Prediction models with survival data: a comparison between machine learning and a Cox proportional hazard regression model

A simulation study and an application to osteosarcoma data from a randomised clinical trial

Audinga-Dea Hazewinkel
Thesis advisor: Dr. Marta Fiocco

Mathematical Institute
Leiden University Medical Center

MASTER THESIS

Defended on September 28th, 2018

STATISTICS FOR THE LIFE AND BEHAVIOURAL SCIENCES
Specialization: Statistical Science
Acknowledgements

I would like to thank my supervisor, Dr Marta Fiocco, for her guidance, support and feedback over the duration of this thesis. I would also like to express my appreciation to Prof. Dr. Hein Putter, Prof. Dr. Hans Gelderblom, and Dr. Carlo Lancia for taking the time to evaluate my work and for taking part in the exam committee. A special thanks to Anja Rüten-Budde (MSc) for her advice and her help with the implementation of the dynamic prediction model of Chapter 2.
Abstract

Over the past years there has been an increased interest in applying machine learning (ML) techniques to medical research. With the growing availability of mixed data - clinical and genomic for instance - ML methods, which have great potential for modelling complex data, have been increasingly applied. Few publications however have seen clinical applications, and the trend towards ML has been criticised for a lack of attention towards proper validation and towards the use of appropriate performance measures to quantify the model performance. Initially, in the context of medical research, machine learning methods were mainly used for diagnosis and detections, but the last years have seen a vast increase in ML modelling for the purpose of cancer prediction and prognosis. The latter trend has given rise to various adaptations of traditional ML approaches to censored survival data. Two such approaches - Biganzoli’s survival neural network and Ishwaran’s random survival forest - are evaluated in this thesis. They are compared to a statistical model - the well-used Cox proportional hazards model - in an application to a clinical dataset with 7 variables, measured on 2025 osteosarcoma patients- the EURAMOS-1 clinical trial.

The purpose of this thesis is two-fold; 1) performing an in-depth comparison of the two ML methods and gaining insight into the potential of ML for clinical data with a limited number of predictors; 2) adding to existing osteosarcoma literature, in which ML methods have a very limited presence. The analyses performed on the EURAMOS data are reinforced by a simulation study, which is novel in the approach it takes to ensure that the simulated data closely mimics the original. This thesis shows that for the EURAMOS-1 osteosarcoma data the Cox proportional hazard model is suitable, and that both ML approaches have limited added benefit. Appropriate performance measures are identified for assessing neural network and random survival forest performance.

For the survival neural network a modification to an existing measure is proposed to aid in identifying network instability - a known neural network pitfall. For the random survival forest it is shown that while suitable for distinguishing high and low risk patients, it results in unreliable individual survival predictions.

An additional, unrelated chapter has been included in this thesis, detailing the application of a dynamic prediction model to the EURAMOS-1 osteosarcoma data.
# Contents

## Introduction

1. Data Description
   1.1 The EURAMOS-1 clinical trial
      1.1.1 Trial protocol
      1.1.2 Consort diagram
   1.2 Patient characteristics and events
      1.2.1 Variables
      1.2.2 Events
   1.3 Data formatting

2. A Dynamic Prediction model for osteosarcoma patients
   2.1 Background
   2.2 Application
      2.2.1 General approach
      2.2.2 Preparing data for landmark analysis
      2.2.3 Fitting a landmark model
      2.2.4 Pooling estimates
      2.2.5 Discriminative ability
      2.2.6 Dynamic predictions
   2.3 Results
      2.3.1 Descriptives
      2.3.2 Dynamic prediction model for overall survival
      2.3.3 Internal calibration
   2.4 Discussion
5.1.3 Simulation of survival and censoring times . . . . . . . . . . . 136
5.2 Cross-validation approach and functions . . . . . . . . . . . . . . 142
  5.2.1 Cox proportional hazards model . . . . . . . . . . . . . . . . 142
  5.2.2 Random Survival Forests . . . . . . . . . . . . . . . . . . . . 145
  5.2.3 Neural network . . . . . . . . . . . . . . . . . . . . . . . . . 152
  5.2.4 Simulation study . . . . . . . . . . . . . . . . . . . . . . . . . 155
5.3 A final comparison between the neural network, random survival forest, 
  and Cox proportional hazards approaches . . . . . . . . . . . . . . 156
  5.3.1 Simulation results . . . . . . . . . . . . . . . . . . . . . . . . 157
  5.3.2 EURAMOS-1 Osteosarcoma data . . . . . . . . . . . . . . . . 162
    5.3.2.1 Performance measures . . . . . . . . . . . . . . . . . 162
    5.3.2.2 Patient-specific predictions . . . . . . . . . . . . . . 164
    5.3.2.3 Variable importance . . . . . . . . . . . . . . . . . . 166
6  Summary and contribution 169
  6.1 Aims . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
  6.2 General approach . . . . . . . . . . . . . . . . . . . . . . . . . . . 170
    6.2.1 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 170
    6.2.2 Data simulation . . . . . . . . . . . . . . . . . . . . . . . . . 170
    6.2.3 Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171
  6.3 Performance measures and variable importance . . . . . . . . . . 172
  6.4 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 175
  6.5 Method comparison and recommendations . . . . . . . . . . . . . 177
  6.6 Future research . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180
    6.6.1 Neural network . . . . . . . . . . . . . . . . . . . . . . . . . 181
    6.6.2 Random survival forest . . . . . . . . . . . . . . . . . . . . . 182
  6.7 Novelty . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
Discussion 184
A  Data description 197
  A.1 Consort diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . . 197
    A.1.1 Consort diagram: EURAMOS trial . . . . . . . . . . . . . . . 197
    A.1.2 Consort diagram: eligibility for analysis . . . . . . . . . . . . 199
  A.2 Events . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 199
  A.3 Variables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 201
  A.4 Data formatting . . . . . . . . . . . . . . . . . . . . . . . . . . . . 204
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>A Dynamic Prediction model for osteosarcoma patients</td>
<td>208</td>
</tr>
<tr>
<td>B.1</td>
<td>Data preparation</td>
<td>208</td>
</tr>
<tr>
<td>B.2</td>
<td>Fitting a landmark model</td>
<td>211</td>
</tr>
<tr>
<td>B.3</td>
<td>Pooling estimates</td>
<td>211</td>
</tr>
<tr>
<td>B.4</td>
<td>Internal validation</td>
<td>213</td>
</tr>
<tr>
<td>B.5</td>
<td>5-year death prediction</td>
<td>216</td>
</tr>
<tr>
<td>B.6</td>
<td>Results</td>
<td>218</td>
</tr>
<tr>
<td>C</td>
<td>A Neural network approach</td>
<td>226</td>
</tr>
<tr>
<td>C.1</td>
<td>Data transformation</td>
<td>226</td>
</tr>
<tr>
<td>C.2</td>
<td>Performance measures and variable importance</td>
<td>229</td>
</tr>
<tr>
<td>C.3</td>
<td>Cross-validation</td>
<td>234</td>
</tr>
<tr>
<td>C.3.1</td>
<td>Cross-validation functions</td>
<td>234</td>
</tr>
<tr>
<td>C.3.2</td>
<td>Cross-validation plots</td>
<td>240</td>
</tr>
<tr>
<td>C.4</td>
<td>Results</td>
<td>256</td>
</tr>
<tr>
<td>C.4.1</td>
<td>Neural Network</td>
<td>256</td>
</tr>
<tr>
<td>C.4.2</td>
<td>Cox model</td>
<td>275</td>
</tr>
<tr>
<td>D</td>
<td>A Random survival forest approach</td>
<td>277</td>
</tr>
<tr>
<td>D.1</td>
<td>rfsrc function</td>
<td>277</td>
</tr>
<tr>
<td>D.2</td>
<td>Performance measures</td>
<td>280</td>
</tr>
<tr>
<td>D.3</td>
<td>Cross-validation</td>
<td>283</td>
</tr>
<tr>
<td>D.3.1</td>
<td>Cross-validation functions</td>
<td>283</td>
</tr>
<tr>
<td>D.3.2</td>
<td>Cross-validation for node size and number of candidate splitting variables</td>
<td>288</td>
</tr>
<tr>
<td>D.3.3</td>
<td>Cross-validation for number of splitting points and node depth</td>
<td>314</td>
</tr>
<tr>
<td>D.3.4</td>
<td>A comparison of different splitting rules</td>
<td>330</td>
</tr>
<tr>
<td>D.3.5</td>
<td>A closer look at the individual tuning parameters</td>
<td>332</td>
</tr>
<tr>
<td>D.3.6</td>
<td>Results</td>
<td>353</td>
</tr>
<tr>
<td>D.4</td>
<td>Additional cross-validation plots</td>
<td>363</td>
</tr>
<tr>
<td>E</td>
<td>Simulation</td>
<td>365</td>
</tr>
<tr>
<td>E.1</td>
<td>Data simulation</td>
<td>365</td>
</tr>
<tr>
<td>E.1.1</td>
<td>Covariate information from original EURAMOS osteosarcoma data</td>
<td>365</td>
</tr>
<tr>
<td>E.1.2</td>
<td>Simulation of covariate data</td>
<td>372</td>
</tr>
<tr>
<td>E.1.3</td>
<td>Simulation of survival and censoring times</td>
<td>372</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 EURAMOS-1 treatment protocol, as taken from Whelan (2015) \[33\] . 5  
1.2 EURAMOS-1 Consort diagram. ........................................ 6  
1.3 Eligibility for analysis. ............................................. 7  
1.4 Distribution of predictors age at registration and absolute tumour volume of death ......................................................... 10  
1.5 Distribution of survival times and death times, for the pooled category of death .......................................................... 11  

2.1 Number of patients at risk at each landmark time point .......... 31  
2.2 Time-varying hazard ratios for histological response .......... 33  
2.3 5-year death predictions for patients with different characteristics .... 36  
2.4 Number of patients at risk and dead from 5 to 8.9 years ........ 37  

3.1 Single hidden layer feed forward neural network ................. 40  
3.2 Lapuerta censoring imputation ..................................... 50  
3.3 Kaplan Meier and event time distribution .......................... 65  
3.4 Cross-validated performance measures for a weight decay of 0 .... 67  
3.5 Cross-validated performance measures for a weight decay of 0.001 .... 67  
3.6 Cross-validated performance measures for a weight decay of 0.01 .... 68  
3.7 Cross-validated performance measures for a weight decay of 0.05) .... 68  
3.8 Cross-validated performance measures for a weight decay of 0.1) .... 69  
3.9 Individual survival curves for different patients for networks with 1-10 nodes and decay values 0.05 and 0.001) ................................. 70  
3.10 Instability of survival predictions for a neural network with 5 nodes, 0.001 decay ................................................................. 70  
3.11 Predicted survival curves of all patients ...................................................... 72  
3.12 Schematic representation of the final neural network: 5 hidden nodes, weight decay 0.05 ......................................................... 73
5.2 Distribution of continuous covariates age and absolute tumour volume in EURAMOS osteosarcoma data ................................. 132
5.3 Distribution of continuous covariates age and absolute tumour volume after imputation .......................................................... 132
5.4 Censoring distributions for original data and imputed data .............. 140
5.5 Survival distributions for original data and imputed data ............... 141
5.6 Distribution of C-index, hazard ratio (HR) and logrank $\chi^2$-statistic for Cox model and RSF model ........................................ 158
5.7 Brier and Kullback-Leibler scores for Cox model, RSF model, and neural network model ..................................................... 159
5.8 Brier and Kullback-Leibler scores for Cox model, RSF model, and neural network model, with a separation of neural network scores in high and low values ............................................................ 160
5.9 Brier and Kullback-Leibler scores for RSF model, obtained via OOB data, and via leave-1%-out cross-validation ............................ 161
5.10 Distribution of survival/censoring times and event times for simulated data ($S = 1000$) .............................................................. 162
5.11 Brier and Kullback-Leibler scores for Cox model, RSF model, and neural network model ..................................................... 163
5.12 Distribution of survival times and event times for the EURAMOS osteosarcoma data ................................................................. 164
5.13 Predicted survival curves for all test set patients for the neural network model, Cox model and RSF model ................................ 165
5.14 Patient-specific predictions for the neural network model, random survival forest model and Cox proportional hazards model ....... 166
5.15 Variable importance for neural network and random forest model .... 167
6.1 Brier and Kullback-Leibler scores for Cox model, RSF model, and neural network model ..................................................... 176
D.1 Cross-validated performance measures for the number of random split-points ($nsplit$) and node depth ($nodedepth$), part 1 ............... 363
D.2 Cross-validated performance measures for the number of random split-points ($nsplit$) and node depth ($nodedepth$), part 2 ............... 364
List of Tables

1.1 Patient characteristics .............................................. 9
1.2 Absorbing event counts ............................................ 10
1.3 Intermediate event counts ......................................... 11
1.4 Data structure ...................................................... 12

2.1 Wide format .......................................................... 17
2.2 Long format .......................................................... 17
2.3 Example landmark dataset .......................................... 19
2.4 Example Patient ..................................................... 29
2.5 HR good versus poor histological response over time .......... 32
2.6 Overview patient characteristics used in 5-year death predictions ... 33
2.7 Dynamic prediction model for overall survival .................... 34

3.1 Neural network characteristics (part 1) .............................. 43
3.2 Neural network characteristics (part 2) .............................. 43
3.3 Street’s neural network approach: defining targets ................ 45
3.4 Ravdin & Clark: data structure ..................................... 46
3.5 Biganzoli: data structure ............................................ 48
3.6 Montes-Torres method: defining targets ........................... 48
3.7 Original data format(wide) ........................................... 53
3.8 Transformed data format for training data (long) .................. 53
3.9 Performance measures for a 5 node neural network with 0.05 decay and the Cox proportional hazards model ............................... 74
3.10 Patient characteristics for reference patient ‘poor hist’ patient and ‘poor hist + lung mets’ patient ........................................ 74
3.11 Cox proportional hazards model: Hazard ratios .................... 76

4.1 Parameters in \textit{rfsr}c function relevant to the construction of a random survival forest ........................................... 92
4.2 Output from the \textit{rfsr}c function ...................................... 94
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Brier score at 1-6 years for 0 split points and 1-15 (by 2) nodes</td>
</tr>
<tr>
<td>4.4</td>
<td>Brier score at 1-6 years for 3 split points and 1-15 (by 2) nodes</td>
</tr>
<tr>
<td>4.5</td>
<td>C-index, hazard ratio and log-rank $\chi^2$-statistic for the three final random survival forests (RSF) models and the Cox proportional hazards model</td>
</tr>
<tr>
<td>5.1</td>
<td>Unique categorical covariate combinations, proportions, and continuous covariate means for sample dataset</td>
</tr>
<tr>
<td>5.2</td>
<td>Unique combinations of categorical covariates, with counts and proportions, and continuous covariate means per unique combination</td>
</tr>
<tr>
<td>5.3</td>
<td>Parameter estimates from log-normal regression on EURAMOS osteosarcoma data (N=2025)</td>
</tr>
<tr>
<td>5.4</td>
<td>Overview of performance measures used to assess the Cox proportional hazards model, the random survival forest model, and the neural network model. For each measure the section describing the theory is given. Hazard ratio refers to the prognostic index/ensemble mortality hazard ratio for the Cox model and random survival forest model, respectively.</td>
</tr>
<tr>
<td>5.5</td>
<td>C-index, hazard ratio, and logrank $\chi^2$-statistic for Cox model and RSF model, for a simulation of $S = 1000$, with the simulation error given. The hazard ratio is calculated for the prognostic index and ensemble mortality for the Cox model and RSF model, respectively.</td>
</tr>
<tr>
<td>5.6</td>
<td>Range of Kullback-Leibler scores for neural network model per year</td>
</tr>
<tr>
<td>5.7</td>
<td>C-index, hazard ratio and logrank $\chi^2$-statistic for RSF model, obtained via OOB data, and via leave-1%-out cross-validation.</td>
</tr>
<tr>
<td>5.8</td>
<td>C-index, hazard ratio (HR) of the prognostic index (Cox) or ensemble mortality (RSF), and the logrank $\chi^2$-statistic, for the random survival forest (RSF) model and the Cox proportional hazards model</td>
</tr>
<tr>
<td>5.9</td>
<td>Patient characteristics for reference patient 'poor hist' patient and 'poor hist + lung mets' patient</td>
</tr>
<tr>
<td>5.10</td>
<td>Cox proportional hazards model: Hazard ratios (HR) with 95% confidence intervals (95% CI)</td>
</tr>
<tr>
<td>6.1</td>
<td>Concordance index for random survival forest (RSF) and Cox proportional hazards model as measured on the EURAMOS-1 osteosarcoma data and as obtained from the simulation ($S = 1000$), with simulation standard error given.</td>
</tr>
</tbody>
</table>
6.2 Predictors identified as imported by neural network (relative importance), random survival forest (VIMP) and Cox proportional hazard model (hazard ratio). Relevant predictors are the presence of lung metastases (Lung), the presence of other metastases (Other), and histological response (Hist). RSF does not distinguish between factor levels.
Introduction

Over the last decade interest and publications on machine learning approaches in medical research and specifically cancer research as well has grown, giving rise to a still ongoing debate on the value of machine learning (ML) approaches versus more traditional statistical modelling (SM). Some examples of statistical models are (penalised) regression models, various semi-parametric models, generalised additive models, and time-to-event models. A statistical model exists within a mathematical framework and makes certain probabilistic assumptions about the data generation process.[12] Generally SM models allow for easy interpretation - consider in the most simple case the parameters from a ordinary linear regression or, in a survival analysis context the coefficients and hazard ratios of the semi-parametric Cox proportional hazards model. In contrast ML approaches makes no assumptions about the data generating process and as such no a priori specified relationship is imposed on the predictors and outcome. ML relies on algorithms and learns relationships from the data itself. While the absence of preconceived structures allows for complicated non-linear effects and high-order interactions between variables, these relationships cannot be accurately quantified, and at best only a general indication of predictor importance can be obtained.

Initially, most cancer-related ML modelling was done with the purpose of cancer diagnosis and detection [1], but in recent years there has been a vast increase in ML modelling for the purpose of cancer prediction and prognosis [1, 34]. In the latter, three main areas can be distinguished: prediction of cancer susceptibility, prediction of recurrence and prediction of survival. The latter is the particular focus of this thesis, which aims to make an in-depth comparison of the performance of two ML approaches - neural networks and random forests - and a more traditional SM choice. The most common choice for modelling time-to-event data - for performing survival analysis - is the Cox proportional hazards model. This semi-parametric approach has the desirable quality of straightforward interpretation, but imposes limitations by assuming a proportional hazard. Machine learning alternatives have been suggested
in the following situations: 1) the variables have an unknown and likely complex relationship with the outcome; 2) the variables have higher order interactions amongst each other.

Applying ML to survival data offers particular challenges due to the censored nature of the data. Various modifications have been proposed to existing methods to accommodate censoring, and two such approaches are compared in this thesis: random survival forests, as described by Ishwaran [19], and an artificial neural network (ANN) approach detailed by Biganzoli. [10]. Random forests are built from individual decision trees, which enact classification in a binary recursive splitting process, splitting a given feature space into increasingly smaller regions, containing observations with similar outcomes. Individual trees have high variability which is resolved by taken their ensemble. The latter is known as bagging, and, under certain constraints, becomes a random forest. ANNs model complex relationships by defining input neurons - nodes - represented by predictor variables, a hidden layer with a number of nodes connected to each of the input nodes, and finally, an output layer with one or more nodes, connected to each of the hidden nodes. An activation function is applied to the hidden layer and determines if a signal is passed or not - analogous to neuron transmissions in the brain. The connections between the nodes are assigned weights, found in an algorithm minimising a predefined loss function.

Random forests and neural networks are but two of a variety of machine learning methods, with other examples including support vector machines and Bayesian networks, each method having a range of varying approaches described. These methods are increasingly applied to medical data, and this newfound interest is largely motivated by the shift towards personalised medicine. Contributing to this is the increased availability of omics data resulting datasets that integrate different types of data, most typically clinical and genomic data, but also increasingly imaging data, and proteomic and metabolomic data. ML methods, not requiring pre-specification of the relationships between predictors and outcomes, have a lot of potential when it comes to complex data, and have been shown to produce comparatively high accuracies for disease prediction [34]. This trend towards machine learning has also given rise to criticism. Frequently ML models are insufficiently validated, the parameters are incorrectly tuned, and unsuitable performance measures are used. [12] The latter is a particular issue as a poorly defined measure of performance - or in this
context prediction accuracy - will affect both the tuning process and the final model evaluation.

This thesis examines the potential of two machine learning techniques - neural networks and random forests- for a clinical dataset with 7 variables, measured on 2025 osteosarcoma patients - data from the EURAMOS-1 osteosarcoma clinical trial. It is a comparatively small dataset, with a limited number of predictors. This type of data has not been a popular choice for ML techniques in the past, and as such it is of interest whether these methods will have an added value when compared to the default Cox proportional hazards model. A more concrete purpose of this thesis is contributing to existing osteosarcoma literature, were machine learning methods have received little attention. This thesis offers an in-depth comparison of two machine-learning methods in which special attention is paid to the details and potential pitfalls of implementation. A variety of performance measures is examined and suitable ones are identified. Parameters for the models are tuned via extensive cross-validation, and the final models are tested on a separate validation set. The results obtained from the EURAMOS-1 osteosarcoma data are reinforced by a simulation study, in which datasets are simulated in a way that closely mimics the original data.

Two parts can be distinguished in this thesis. The first chapter describes the EURAMOS-1 osteosarcoma data in detail, and is relevant to all the remaining chapters. In chapter 2 a standalone chapter is presented, detailing the theory and application of a dynamic prediction model, which manually incorporates non-linear predictor effects. The remaining chapters focus on the comparison of the two machine learning approaches. Chapter 3 details the fitting of a neural network model to the EURAMOS osteosarcoma data, Chapter 4 does the same for random survival forests. Chapter 5 describes the simulation and makes a final comparison between all three models. The description of the data applications in each chapter are preceded by the relevant theory of the methods, giving a general overview of the traditional use and the extensions made to accommodate time-to-event data. Throughout the chapters excerpts of the code used in the R-software environment are given. Each chapter is matched by an appendix, which contains the full code, with the purpose of making every analysis step explicit and enabling complete reproducibility. Chapter 6 offers a succinct summary of the general approach, the most important results, and the advantages and (potential) pitfalls of each model. The thesis is concluded by an overall discussion of the three methods.
Chapter 1

Data Description

This chapter describes the EURAMOS-1 osteosarcoma data, which is used in applications throughout the rest of the thesis. Section 1.1 details the trial protocol and describes which patients were considered suitable for analysis. Section 1.2 describes the event definitions used and a selection of predictors relevant to the thesis.

1.1 The EURAMOS-1 clinical trial

In 2001 EURAMOS (European and American Osteosarcoma Studies) was established—a collaboration between the Childrens Oncology Group (COG), the Cooperative Osteosarcoma Study group (COSS), the European Osteosarcoma Intergroup (EOI), and the Scandinavian Sarcoma Group (SSG), with the purpose of pooling resources and facilitating the study of osteosarcoma. Osteosarcoma is the most common primary bone sarcoma, and is primarily diagnosed in adolescents and young adults. The EURAMOS-1 study is the first of the EURAMOS studies and has recruited a total of 2260 patients from 2005 to 2011.

1.1.1 Trial protocol

A total of 2260 osteosarcoma patients aged < 40 were registered for trial from April 2005 to June 2011. Diagnosis was confirmed via a biopsy. Patients were administered pre-operative chemotherapy involving two 5-week cycles of cisplatin, doxorubicin, methotrexate x 2 (MAP). Patients were randomised after surgery. The patients with a poor histological response (≥ 10 % viable tumour in resected specimen) were administered MAP or MAP with ifosfamide and etoposide (MAPIE), those with a good histological response (< 10 % viable tumour in resected specimen) received MAP or MAP with Ifn over a duration of 28 weeks. Disease progression was documented in
the occurrence of various events: local recurrence, new metastatic disease, secondary malignancy, progression metastatic disease, death, and any combination of events.

The trial protocol is described extensively by Whelan et al. [33] In Figure 2.1 a illustration taken from Whelan [33] is shown, depicting the treatment course. Preceding the surgery are two 5-week cycles. Post surgery, the histological response of the resected specimen is assessed. Good responders are randomised to another 4 cycles of treatment of either MAP or MAP + Ifn. Poor responders are randomised to either 4 cycles of MAP treatment or of MAPIE. All non-randomised patients receive a 4 cycle MAP treatment post surgery.

### 1.1.2 Consort diagram

Figure 1.3 shows the CONSORT diagram, modeled after the CONSORT diagram reported by Whelan. [33] Minor differences are observed, which are assumed to be due to updated dataset versions. The study contains 2260 registered patients, 2209 of which received a diagnostic biopsy, while 2248 received the preoperative treatment. 2056 patients have records of surgery. For 2012 patients, post-operative information on the resected specimen is available, for example on histological response and excision type. For a further 27 patients, information on histological response is present on the treatment forms - this information has been used to supplement the post-operative
Of the randomised patients, 5 good responders have been mistakenly randomised as poor responders, receiving MAP treatment, 7 poor responders have been mistakenly randomised as good responders, receiving MAP+Ifn or MAP treatment.

924 patients were not randomised. At the end of follow-up 622 out of 2260 patients
are classified as dead due to osteosarcoma, 78 as dead due do other causes. For 12 patients no follow-up information post surgery is available.

Of the total 2260 patients a subset of 1965 patients was considered eligible for analysis. Time of surgery was defined as the start of follow-up, with time measured in years since surgery. Patients that had no record of a surgery data were excluded, as it could not be conclusively established whether surgery had been performed. Patients that experienced an event prior to or at the time of surgery were removed, as were patients that had no follow-up post-surgery. In order to make the best use of the available data both randomised and non-randomised patients were included in the analysis. Non-randomised patients were considered to be eligible if the non-randomization was not disease-dependent. Figure 1.3 summarizes the exclusion criteria. The counts for each subsequent category are computed after removing the patients identified in the previous categories. For Chapters 3-6 information on the intermediate events local recurrence and new metastatic disease was not used, and 2025 patients were considered eligible for analysis, as patients that experienced an event prior to or at the time of surgery were not removed.

1.2 Patient characteristics and events

This section describes the patient characteristics for a subset of predictors that have been selected for analysis. In addition, the definitions of several intermediate events
are described, which are relevant to Chapter 2 A Dynamic Prediction model for osteosarcoma patients, where the occurrence of various events preceding the final event of death death is introduced in the form of time-dependent covariates.

1.2.1 Variables

Seven predictors of interest were identified by the clinician: histological response, surgical excision type, presence of lung metastases, presence of other metastases, absolute tumour volume, sex, and age at time of surgery. The presence of lung metastases and the presence of other metastases are baseline characteristics and established before the start of follow-up. The type of surgical excision, the absolute tumour volume and the histological response are established at or shortly after surgery before the post-operative treatment. Histological response distinguishes between two levels - good and poor - where a good response has less than 10% viable tumour in the resected specimen, and a poor response 10% or more. Information on histological response was present in the variable indicating histological response, as well as in the treatment variable, as randomization to treatment is dependent on histological response. A new variable was made for histological response, pooling the information from both. The original excision variable distinguishes between radical, wide, marginal, intralesional and unknown excision. For the purpose of this analysis, and in the interest of having large enough categories, excision types were grouped on recommendation of the consulting clinician. Wide and radical excision were pooled, as were intralesional and unknown excision. The presence of lung metastases and the presence of other metastases are defined as the categories no, yes and possible. Yes and possible were pooled together.

Absolute tumour volume was measured in cm x cm x cm x 0.54. The measurements have been split into two categories, less than 200 cm$^3$ and 200 cm$^3$ or more. Age was determined using the date of surgery and date of birth. The values were pooled into three categories: < 12 years, 12-18 years, ≥ 18. The distribution of patient characteristics for randomised an non-randomised patients is shown in Table 1.1. Figure 1.4 shows the distribution for age at surgery (a) and absolute tumour volume (b,c) prior to the value grouping. Age at surgery ranges from 4.31 to 40.72 years, with most patients less than 20 years of age. Absolute tumour ranges from 0.0052 to 2604 cm$^3$, with most observations smaller than 500 cm$^3$. A closer look at these values is taken in Figure 1.4c
<table>
<thead>
<tr>
<th>Prognostic factor</th>
<th>randomised N</th>
<th>randomised %</th>
<th>Non-randomised N</th>
<th>Non-randomised %</th>
<th>Total N</th>
<th>Total %</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Age</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-18</td>
<td>743</td>
<td>38</td>
<td>346</td>
<td>18</td>
<td>1089</td>
<td>55</td>
</tr>
<tr>
<td>&lt;12</td>
<td>324</td>
<td>16</td>
<td>148</td>
<td>8</td>
<td>472</td>
<td>24</td>
</tr>
<tr>
<td>&gt;18</td>
<td>261</td>
<td>13</td>
<td>143</td>
<td>7</td>
<td>404</td>
<td>21</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Histological response</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good( &lt; 10% tumor)</td>
<td>715</td>
<td>36</td>
<td>281</td>
<td>14</td>
<td>996</td>
<td>51</td>
</tr>
<tr>
<td>Poor( ≥ 10 % tumor)</td>
<td>609</td>
<td>31</td>
<td>306</td>
<td>16</td>
<td>915</td>
<td>47</td>
</tr>
<tr>
<td>Missing</td>
<td>4</td>
<td>0</td>
<td>50</td>
<td>3</td>
<td>54</td>
<td>3</td>
</tr>
<tr>
<td><strong>Excision</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wide/radical</td>
<td>1096</td>
<td>56</td>
<td>510</td>
<td>26</td>
<td>1606</td>
<td>82</td>
</tr>
<tr>
<td>Marginal</td>
<td>175</td>
<td>9</td>
<td>63</td>
<td>3</td>
<td>238</td>
<td>12</td>
</tr>
<tr>
<td>Other</td>
<td>57</td>
<td>3</td>
<td>64</td>
<td>3</td>
<td>121</td>
<td>6</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Volume</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;200</td>
<td>734</td>
<td>37</td>
<td>363</td>
<td>18</td>
<td>1097</td>
<td>56</td>
</tr>
<tr>
<td>≥ 200</td>
<td>341</td>
<td>17</td>
<td>180</td>
<td>9</td>
<td>521</td>
<td>27</td>
</tr>
<tr>
<td>Missing</td>
<td>253</td>
<td>13</td>
<td>94</td>
<td>5</td>
<td>347</td>
<td>18</td>
</tr>
<tr>
<td><strong>Sex</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>546</td>
<td>28</td>
<td>264</td>
<td>13</td>
<td>810</td>
<td>41</td>
</tr>
<tr>
<td>Male</td>
<td>782</td>
<td>40</td>
<td>373</td>
<td>19</td>
<td>1155</td>
<td>59</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Lung metastases</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1083</td>
<td>55</td>
<td>504</td>
<td>26</td>
<td>1587</td>
<td>81</td>
</tr>
<tr>
<td>Yes/Possible</td>
<td>245</td>
<td>12</td>
<td>133</td>
<td>7</td>
<td>378</td>
<td>19</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Other metastases</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1279</td>
<td>65</td>
<td>610</td>
<td>31</td>
<td>1889</td>
<td>96</td>
</tr>
<tr>
<td>Yes/Possible</td>
<td>49</td>
<td>2</td>
<td>27</td>
<td>1</td>
<td>76</td>
<td>4</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


1.2.2 Events

The occurrence of a number of events during follow-up was reported. As information was gathered longitudinally, information on subsequent events is available. In Table 1.2 the frequencies of the following events are shown, occurring as first event, and as second event: death, progression of new metastatic disease (Progression NM), secondary malignancy (Sec. Malignancy), and the combination of any of the previous (Combination). For an additional 78 patients death due to osteosarcoma was recorded at final follow-up. Under advice of the consulting clinician these events were considered to be absorbing events and were pooled together under the category death. This definition of death is used throughout the rest of the thesis.

<table>
<thead>
<tr>
<th>First event</th>
<th>Death</th>
<th>Progression NM</th>
<th>Sec. Malignancy</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>67</td>
<td>27</td>
<td>56</td>
</tr>
<tr>
<td>Second event</td>
<td>356</td>
<td>5</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 1.5 shows the distribution of survival times and event times under the previously defined definition of death. Survival times range from 0.005 to 8.97 years since surgery. Death times range from 0.005 to 7.85 years since surgery and are most frequent in the first half of the follow-up.
For the purpose of Chapter 2 a Dynamic Prediction model for osteosarcoma patients a number of intermediate events were considered. Local recurrence and new metastatic disease were selected as intermediate events of interest. Table 1.3 shows the frequencies for local recurrence (LR), new metastatic disease (NM) and the combination of the two (LR+NM), occurring as a first event, and as a second event. All three events are noticeably less frequent as a second event, with NM being most common. A closer examination of those experiencing NM as a second event shows that 71 of the 88 also experienced NM as a first event. In the interest of a sufficient event count for ensuring reliable estimation, the decision was made to use the information of the first events exclusively, and to disregard the combination category (LR+NM).

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>NM</th>
<th>LR+NM</th>
</tr>
</thead>
<tbody>
<tr>
<td>First event</td>
<td>73</td>
<td>481</td>
<td>39</td>
</tr>
<tr>
<td>Second event</td>
<td>9</td>
<td>88</td>
<td>3</td>
</tr>
</tbody>
</table>

### 1.3 Data formatting

In order to make a dataset suitable for analysis, various modifications were made to the original datafile. For the sake of completeness, the process of converting the file to a working wide format is detailed in appendix A.4. This wide format file is used as a starting point for the analyses detailed in Chapter 2. Below, in Table 1.4, an example...
of the appropriate data structure is given. Here, \( s.death \) and \( t.death \) stand for status and time of death, respectively. \( s.LR \) and \( t.LR \) and \( s.NM \) and \( t.NM \) represent the same for the events of local recurrence (LR) and new metastatic disease (NM). In the absence of an intermediate event, a patient remains at risk for the event until time of death or time of censoring. For the purpose of Chapters 3 to 6 information on the intermediate events local recurrence and new metastatic disease was disregarded. Additionally, for the predictors age and volume no discretization was performed and they were introduced as continuous variables.

Table 1.4: Data structure

<table>
<thead>
<tr>
<th>s.LR</th>
<th>t.LR</th>
<th>s.NM</th>
<th>t.NM</th>
<th>s.death</th>
<th>t.death</th>
<th>Hist. resp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2.88</td>
<td>0</td>
<td>2.88</td>
<td>0</td>
<td>2.88</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>8.23</td>
<td>0</td>
<td>8.23</td>
<td>0</td>
<td>8.23</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3.61</td>
<td>0</td>
<td>3.61</td>
<td>0</td>
<td>3.61</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>7.92</td>
<td>0</td>
<td>7.92</td>
<td>0</td>
<td>7.92</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>6.17</td>
<td>0</td>
<td>6.17</td>
<td>1</td>
<td>6.17</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>5.32</td>
<td>0</td>
<td>5.32</td>
<td>0</td>
<td>5.32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>age</th>
<th>volume</th>
<th>excision</th>
<th>sex</th>
<th>lung met</th>
<th>other met</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 12-18 years</td>
<td>&gt;=200</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2 12-18 years</td>
<td>&gt;=200</td>
<td>Other</td>
<td>Female</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3 12-18 years</td>
<td>&gt;=200</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4 12-18 years</td>
<td></td>
<td>Other</td>
<td>Male</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5 12-18 years</td>
<td>&gt;=200</td>
<td>Marginal</td>
<td>Male</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>6 12-18 years</td>
<td>&lt; 200</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 2

A Dynamic Prediction model for osteosarcoma patients

Dynamic prediction can be defined as the making of a prediction at a certain moment in time, given all the history of events and covariates up until that moment. This can be done by selecting all the individuals at risk at that time, and using solely the information available at that specific time to make a prediction. The sectioning of data for different points in time - prediction times- is referred to as landmarking in medical literature. The dynamic prediction approach makes it possible to extend a simple proportional model so that one can make predictions for different times, while capturing the effect of time-dependent and time-varying covariates. The method was first described by Van Houwelingen in 2006, then expanded on in great detail by Houwelingen & Putter in 2012, in the book Dynamic Prediction in Clinical Survival Analysis. In this chapter a short background on the dynamic prediction approach is given, followed by a detailed application to the EURAMOS-1 trial osteosarcoma data. Both theory and the practical approach are taken from the 2012 book.

2.1 Background

Let $t_{LM}$ define the time from which a prediction is made. At a given prediction time $t_{LM} = s$ a dataset is defined which contains information on events and covariates only up until this time-point. Such sectioned datasets are called landmark datasets, and the prediction times are landmark (LM) times. Then in the dynamic prediction approach a sliding window of width $w$ is defined and predictions are made from time $t_{LM}$ for a fixed horizon $t_{LM} + w$. In the simplest approach, a separate Cox model would be fitted for each landmark time. It is, however, desirable, for reasons of
practicality and ease of interpretation, to obtain one single ‘super-model’. The first thing necessary for that is that the regression coefficients depend on landmark time. Equation (2.1) defines the hazard at time $t$, given a set of covariates, landmark time $t_{LM} = s$ and prediction window $w$. Different from a regular Cox model it conditional on not only on the covariates $x$, but also on landmark and prediction time. This approach is known as a sliding landmark model.

$$h(t|x, t_{LM} = s, w) = h_0(t|s, w) \exp(x^T \beta_{LM}(s))$$ (2.1)

Additionally, the coefficients $\beta_{LM}$ are dependent on landmark time, as shown in formula (2.2).

$$\beta_{LM}(s) = \sum_{j=1}^{m_b} \gamma_j f_j$$ (2.2)

Here, $f_j(s)$ gives the $m_b$ functions of time. There are various possibilities for the choice of function. In the practical application detailed in the second part of this chapter the common choice of the following function has been made: $f_1(s) = 1$, in which no time effect is present, $f_2(s) = s$, in which a linear time effect is present, and $f_3(s) = s^2$, in which a quadratic effect is present.

As is apparent from equation (2.1) the baseline hazard also depends on landmark time and the prediction window. The conceptually most simple approach for introducing this dependence is by stratifying the baseline hazard on landmark time. This will however give a separate baseline hazard for each stratum (2.3), and one overall baseline hazard may be preferred. This can be achieved by explicitly introducing a time-dependent element to the baseline hazard, shown in formulas (2.4) and (2.5), similar to the approach used for the regression coefficients $\beta_{LM}(s)$. Note however that for the baseline hazard $h_0(t)$ the time-dependent parameters depend on event time $t_i$, rather than prediction time $t_{LM} = s$.

$$\hat{h}_0(t_i|s, w) = \frac{1}{\sum_{j \in R(t_i)} \exp(x_j^T \beta_{LM}(s))}, \text{ for } s \leq t \leq s + w$$ (2.3)

$$\hat{h}_0(t|s, w) = h_0(t) \exp(\theta(s))$$ (2.4)
\[ \theta(s) = \sum_{j=1}^{m_h} \eta_j g_j(s) \] (2.5)

A common choice for the functions \( g_j(s) \) is to define them as \( f_{j+1}(s) \). Previously, for the regression coefficients the following functions were defined \( f_1(s) = 1 \), \( f_2(s) = s \), and \( f_3(s) = s^2 \). Then for \( g_j = f_{j+1} \) \( g_1 = s \) and \( g_2 = s^2 \).

In making a landmark model a set of prediction points \( s \) is chosen over a certain range. Typically, the prediction points are equally spaced. For example, in the application discussed later, the landmark points are defined every 3 months for a range from 0 to 5 years. A prediction window \( w \) is defined as well, in the application \( w = 5 \). This means that for every landmark point \( s \) in the defined range \([0,5]\), a prediction can be made for the time \( s + w \), shown in equation (2.6).

\[ \hat{H}(s + w|x, t_{LM} = s) = \exp(x^T \hat{\beta}_{LM}(s) + \hat{\theta}(s)) (\hat{H}_0(s + w) - \hat{H}_0(s-)) \] (2.6)
2.2 Application

2.2.1 General approach

The fitting of a dynamic prediction model is implemented in R, and can be achieved within the survival package\[55\], and the dynpred package \[21\] - the companion package to Dynamic Prediction in Clinical Survival Analysis.\[24\]

The data used in the application is from the EURAMOS-1 clinical trial, and consists of 1966 patients. Seven predictive variables, as selected by the consulting clinician, are used for the analysis. Additionally, two time-dependent binary covariates are introduced for the occurrence of events Local recurrence and New metastatic disease. Details on the EURAMOS data and chosen variables can be found in Chapter 1. The measure of interest for this analysis is overall survival, in which death is defined as a pooled category of actual death, secondary malignancy, and progression of metastatic disease. The follow-up time is defined starting from surgery. A landmark prediction model was made for landmark times from 0 to 5 years in 3 month increments, with a prediction window of 5 years. Linear and quadratic time effects were included in the model, and backwards model selection was used to obtain the final model. Missing data was dealt with by 10-fold multiple imputation using the amelia package.\[26\] Coefficient estimates and covariance matrices were pooled using Rubin’s rule.\[49\] The final model was used to obtain 5-year death probability predictions for patients with various characteristics, with a special focus on the effect of the presence/absence of intermediate events Local recurrence and New metastatic disease. Model calibration and discriminative performance were assessed by means of a heuristic shrinkage factor and dynamic cross-validated concordance indexes, respectively.

This chapter is structured in the following way. The remainder of section 2.2 consists of several sub-sections and is devoted to a step by step explanation of the analysis. Small sections of code are used for illustrative purposes, the full code can be found in appendix B, and is referred to throughout the chapter. The R code is either original, or an adapted version of the code from the Dynamic Prediction book. In the latter case, this is explicitly indicated. In section 2.3 the results of the analysis are reported and discussed.

All analyses were performed in the R-software environment version 3.4.2.\[13\]
2.2.2 Preparing data for landmark analysis

Missing data was imputed for 10 datasets using the **Amelia** package (Code: B.1: 1-7). In order to construct the landmark datasets, the data should be in long format, with multiple entries per individual (Code: B.1: 13-37). Below, in wide format (Table 2.1) and long format (Table 2.2) the information of three patients is shown. *id* identifies the patient, *s.LR* and *t.LR* and *s.NM* and *t.NM* and *s.death* and *t.death* denote the status and time of event for local recurrence (LR), new metastatic disease (NM), and death, respectively. Patient 9 experiences a new metastatic disease (NM) at 0.27 years, and a death at 1.16 years. The patient does not experience a local recurrence but is at risk until the time of death at 1.16 years. This information is represented by two rows in long format, where one row represents the patient before the event of NM from 0 to 0.27 years, the second row the patient after the event of NM, from 0.27 to 1.17 years. In the long format the time is now given by *tstart* and *tstop*, and variables LR, NM and death are the necessary indicator variables.

<table>
<thead>
<tr>
<th>ID</th>
<th>s.LR</th>
<th>t.LR</th>
<th>s.NM</th>
<th>t.NM</th>
<th>s.death</th>
<th>t.death</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>8.16</td>
<td>0</td>
<td>3.35</td>
<td>0</td>
<td>8.16</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1.17</td>
<td>1</td>
<td>0.27</td>
<td>1</td>
<td>1.17</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4.98</td>
<td>0</td>
<td>4.98</td>
<td>0</td>
<td>4.98</td>
</tr>
</tbody>
</table>

Landmark time-points of three months were chosen from zero to 5 years, with a prediction window of 5 years. In the construction of the landmark datasets the state of the time-dependent variables is assessed at the landmark point. Below the construction of a single landmark data set, using the `cutLM()` function from the **dynpred** package, is illustrated. For a landmark time point LM with a fixed horizon of 5 years (w) a dataset is constructed. In this dataset, only patients that are alive at the landmark time point are considered, and their status for local recurrence and new metastatic disease is given by their status at the landmark time. The landmark
data set construction is performed twice, once for each of the time-varying covariates LR and NM, and the datasets are combined.

```r
LR <- cutLM(data=longCoxData, 
outcome=list(time="time.death"), 
status="stat.death", LM=LM, horizon=LM+w, 
covs=list(fixed=c("Hist.resp","age","volume","excision","sex","lung.met","other.met"), 
varying="LR"), 
format="long", id="id", rtime="time.LR")

NM <- cutLM(data=longCoxData, 
outcome=list(time="time.death"), 
status="stat.death", LM=LM, horizon=LM+w, 
covs=list(fixed=c()), varying="NM"), 
format="long", id="id", rtime="time.NM")

LMdataLM <- cbind(LR,DM[,c("NM","time.NM")])
```

This process is repeated for every three months from 0 to 5 years, giving a total of 21 landmark datasets, which are combined into one super dataset, with a landmark column (LM) indexing the landmark points. Finally, A quadratic term LM2 is added. This is repeated for each of the 10 imputed datasets (Appendix B.1, 41-103). Below, an short example of a landmark dataset is shown. As can be observed from the LM1 column these 5 entries are for a landmark point of 3 years. Correspondingly, observation starts at 3 years, as is evidenced by Tstart. LM2 gives the quadratic element, here 9 for $3^2$. Note that in the long format data (Table 2.2) used to construct the landmark datasets, $t.LR$ and $t.NM$ denote the time of LR and NM occurring, respectively. In the landmark data, LR and NM are structured as time-varying covariates dependent on landmark time LM1, and as such explicit times indicating LR and NM occurrence are unnecessary.
Table 2.3: Example landmark dataset

<table>
<thead>
<tr>
<th>id</th>
<th>Tstart</th>
<th>time.death</th>
<th>stat.death</th>
<th>Hist.resp</th>
<th>age</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>1447</td>
<td>1447</td>
<td>3.00</td>
<td>3.15</td>
<td>0.00</td>
<td>Poor hist &gt;= 18 years &gt;=200</td>
<td></td>
</tr>
<tr>
<td>1448</td>
<td>1448</td>
<td>3.00</td>
<td>3.36</td>
<td>0.00</td>
<td>Poor hist &gt;= 18 years &lt; 200</td>
<td></td>
</tr>
<tr>
<td>1449</td>
<td>1449</td>
<td>3.00</td>
<td>3.43</td>
<td>0.00</td>
<td>Poor hist &gt;= 18 years &lt; 200</td>
<td></td>
</tr>
<tr>
<td>1450</td>
<td>1450</td>
<td>3.00</td>
<td>3.36</td>
<td>0.00</td>
<td>Good hist 12-18 years &lt; 200</td>
<td></td>
</tr>
<tr>
<td>1451</td>
<td>1451</td>
<td>3.00</td>
<td>3.22</td>
<td>0.00</td>
<td>Good hist &gt;= 18 years &gt;=200</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>excision</th>
<th>sex</th>
<th>lung.met</th>
<th>other.met</th>
<th>LR</th>
<th>NM</th>
<th>LM1</th>
<th>LM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1447</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>0</td>
<td>3.00</td>
</tr>
<tr>
<td>1448</td>
<td>Radical/wide</td>
<td>Male</td>
<td>Yes/Possible</td>
<td>No</td>
<td>0</td>
<td>1</td>
<td>3.00</td>
</tr>
<tr>
<td>1449</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>3.00</td>
</tr>
<tr>
<td>1450</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>3.00</td>
</tr>
<tr>
<td>1451</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>0</td>
<td>0</td>
<td>3.00</td>
</tr>
</tbody>
</table>
2.2.3 Fitting a landmark model

A landmark supermodel was fitted for each imputed dataset. To account for the dependency of patients being present multiple times in the stacked landmark datasets a cluster term is included (cluster(id)), which gives adjusted standard errors using sandwich estimators. Dependency of the regression coefficients on time was represented by the functions $f_1(s) = 1$, $f_2(s) = s$, and $f_3(s) = s^2$. The dependency of the baseline hazard on time was modelled with the functions $g_1 = s$ and $g_2 = s^2$. In the code below the model as defined in the `Coxph()` function is shown.

```r
LMsuperCox <- Coxph(Surv(Tstart, time.death, stat.death) ~ Hist.resp + Hist.resp*LM1 + Hist.resp*LM2 + age + volume + excision + sex + lung.met + other.met + LR + DM + LM1 + LM2 + cluster(id), data=data, method="breslow")
```

Note that for histological response interactions with linear and quadratic landmark time are included. Initially, all interactions with the linear landmark time ($t$) and quadratic landmark time ($t^2$) were included. Pooled coefficient and standard errors over the 10 imputed datasets were obtained using Rubins rule, as detailed in the next section. Model selection was performed by backwards selection. First, the non-significant quadratic interactions were removed, followed by the non-significant linear interactions. The final obtained model, shown in R code above, is given below in mathematical notation (2.7, 2.8) Here, the time-dependent histological response is defined explicitly, and all the time-constant terms are grouped within $\exp(x_c\beta^T_c)$

$$ h(s|x, t_{LM} = s) = \exp(x_{HR} \ast (\beta_{HR} + \beta_{HR:LM1} \ast t + \beta_{HR:LM2} \ast t^2)) \exp(x_c\beta^T_c) \ast \hat{h}_0(t|s, w) $$ (2.7)

wherein

$$ \hat{h}_0(t|s, w) = h_0(t) \ast \exp(\eta_{LM} \ast t) \ast \exp(\eta_{LM2} \ast t^2). $$ (2.8)

As a measure of model performance a heuristic shrinkage factor was estimated (2.9). A shrinkage factor indicates how much the regression coefficients should be
shrunken towards the mean, in order to ensure a good performance of the model on new data. A shrinkage factor can be calculated using a heuristic formula, as shown below, or by means of cross-validation.

\[
\hat{c}_{\text{heur}} = 1 - \frac{\text{dim}}{\chi^2_{\text{model}}}
\]

(2.9)

Here, \text{dim} is the dimension of the prediction model - the number of coefficients estimated, and \(\chi^2_{\text{model}} = 2(\text{ll}_{\text{model}} - \text{ll}_0)\) - the difference of the log-likelihood (ll) for the estimated model and null model.

```r
chisq <- 2*diff(LMsuperCox$loglik)
p <- length(LMsuperCox$coef)
cheur <- 1-p/chisq
```

#Appendix B.1 (1-7)
2.2.4 Pooling estimates

Coefficients and covariance matrices were estimated for each of the 10 imputed datasets. The robust covariance estimates were selected for use. As specified previously, the introduction of a cluster term in the model gives adjusted standard errors. The results were pooled using the well-established Rubins rule. (reference). For the coefficient estimates, this involved simply averaging over all the combinations (2.10) Here $Q_l$ is the vector of regression coefficients. The vectors are averaged over $m=10$ imputations.

$$\bar{Q}_m = \frac{1}{m} \sum_{l=1}^{m} Q_l$$

(2.10)

For the standard errors, the pooled standard error is given by the within variance of the standard error and the between variance of the standard error. The within variance is given, by the mean of the variance across the estimated values for the 10 imputations for each coefficient (2.11) Here, $U_l$ is the covariance matrix for each imputation.

$$\bar{U}_m = \frac{1}{m} \sum_{l=1}^{m} U_l$$

(2.11)

The between variance is given by the inner product of the squared difference of the coefficients and their respective mean, over the number of imputations $m$ (2.12).

$$B_m = \frac{1}{m-1} \sum_{l=1}^{m} (Q_l - \bar{Q}_m)(Q_l - \bar{Q}_m)'$$

(2.12)

The within variance $\bar{U}_m$ and the between variance $B_m$ are combined into the total variance $T_m$ (2.13).

$$T_m = \bar{U}_m + \frac{m+1}{m} B_m$$

(2.13)

The code for the calculations shown above is given below. Here, coefs is a matrix of coefficients for the 10 imputations, and mods is a list of the 10 estimated models.
coefs is a matrix of coefficients for each imputation
mcoef <- apply(coefs, 1, mean)
m <- 10

the covariances matrices of each model are extracted
vars <- lapply(1:10, function (i) vcov(mods[[i]])

The within variance is the average of each matrix entry over the imputations
withinVAR <- Reduce('+', vars)/m

The between variance
betweenVAR <- lapply(coefs, function(x) (x-mcoef) %*% t(x-mcoef)))
betweenVAR <- Reduce('+',betweenVAR)/(m-1)

The total variance
totalVAR <- withinVAR + ((m+1)/m)*betweenVAR

Pooled standard errors
totalSE <- sqrt(diag(totalVAR))

#Appendix B.3 (1–52)

P-values are calculated from the averaged coefficients and the pooled standard errors.

Coxpv <- mcoef/totalSE
mpv <- vector("numeric", length=length(Coxpv))
for (i in 1:length(Coxpv)){
  if (Coxpv[i]<0){
    mpv[i] <- pnorm(Coxpv[i]) * 2
  } else {
    mpv[i] <- pnorm(Coxpv[i], lower.tail=FALSE) * 2
  }
}

#Appendix B.3 (30–37)
2.2.5 Discriminative ability

The discriminative ability of the model was assessed by means of a cross-validated concordance index (C-index). A model with good discriminative ability will predict a higher risk for patients that experience an event earlier than for those with a later event. The C-index is defined as the proportions of pairs for which the order of survival times is matched by the order of model predictions - when they are concordant. A C-index of 0.5 indicates no discriminative ability of the model, while a C-index of 1 means that the model discriminates perfectly. A dynamic version of the C-index is defined below (2.14). Here, dynamic refers to a C-index obtained within a certain time window. Individuals are evaluated from a given prediction time \( t \) to an end-time \( t + w \), where \( w \) is the prediction window. Observations that exceed the window \( [t, t+w] \) are artificially censored.

\[
C_w(t) = \frac{\sum_{i \in D \, t \leq t_i \leq t+ w} \{\# \{ j \in R(t_i); x_j < x_i \} + 0.5 \cdot \# \{ j \in R(t_i); j \neq i, x_j = x_i \} \}}{\sum_{t \leq t_i \leq t+ w} (Y(t_i) - 1)}. \tag{2.14}
\]

Let \( R(t_i) \) define all those at risk at event time \( t_i \) for \( t \leq t_i \leq t + w \). Let \( D \) be the set of individuals that experience an event. Let the corresponding prognostic index for individual \( i \) be \( x_i \). For each event time \( t_i \) the prognostic index \( x_i \) is compared to the prognostic indexes \( x_j \) of all subjects contained in the risk set. A pair is concordant when the subject experiencing an event at time \( t_i \) has a higher prognostic index than a subject experiencing an event or censoring at a later time; \( x_j < x_i \). The number of such pairs is counted for each event time. For each tie - \( x_j = x_i \) - half a count is added. The C-index is given by proportion of the concordant pairs over the total number of evaluated pairs. Concordance can only be established if at least one of the pair has an observed event time.

In the dynamic prediction framework it is possible to calculate a concordance index for any prediction time. In this application landmark times are defined from 0 to 5 years, with \( w = 5 \). The dynamic concordance index \( C_w(t) \) is calculated for \( t = 0, 1, 2, 3, 4, 5 \).

To obtain the concordance index (C-index) values cross-validated prognostic indexes are calculated. For each individual, a model is estimated from all remaining patients. Coefficients obtained from this model are used to obtain an estimate of the prognostic index for the left-out individual.
The cross-validated prognostic indexes are used to calculate the C-index at six points in time: 0, 1, 2, 3, 4, and 5 years. For each time, only the data corresponding to that specific landmark time is used. Subjects that exceed the limit of the landmark time window of $t + w$ are artificially censored at 5 years. Below, the code is given, in which each of the 10 imputed datasets is cropped to suit each time point.
# Function for obtaining cropped dataset within time window \([t, t+w]\), for \(w=5\)

cropData \leftarrow function(LMdata, \ year)\{  
y \leftarrow \ year  

# starting cut-off \(t\) is defined by selecting data with landmark time \(y\)

dat \leftarrow \ LMdata[LMdata$LM1 == y, ]  

# observations contained within window \(t+5\), are selected

dat <- dat [ dat$Tstart < y+5,]  

# observation exceeding \(t+5\) are censored at \(t+5\)

idx <- dat$time.death > y+5  
dat$time.death [ idx ] <- y+5  

# event indicator for artificially censored observations is set to 0

dat$stat.death [ idx ] <- 0  

return ( dat )  
\}

#Appendix B.4 (52–77)

For a given dataset the C-index can be obtained by counting the number of evaluable pairs, the number of concordant pairs, and calculating the proportion of concordant pairs over the total. The code below is an adapted version of the \(CVcindex()\) function of the dynpred package. First, the event times are sorted. The prognostic index \(x_i\) of each individual experiencing an event at \(t_i\) is compared to all prognostic indexes \(x_j\) for \(j = i + 1\) of individuals who have later event or censoring times. The number of comparisons made gives the total of evaluable pairs. The number of times \(x_i\) exceeds or equals \(x_j\) gives the total of concordant pairs, where in the latter case -the event of a tie- only a half is counted. The overall concordance is expressed as the ratio of the concordant pairs over the total of comparable pairs, as counted for all individuals i who experience an event. C-indexes are calculated for each of the time points 0, 1, 2, 3, 4, and 5, for each of the 10 imputed datasets. The average C-index over the imputations is computed (Appendix B.4, 124-143).
cindex.mod <-

function (data) {

  # Define event time variable, status variable, and prognostic index variable
  time <- data$time.death
  status <- data$stat.death
  x <- data$CV
  n <- length(time)

  # Order variables on time (ascending), and on status (descending- 1s first)
  ord <- order(time, -status)
  time <- time[ord]
  status <- status[ord]
  x <- x[ord]

  # Select all individuals who experience an event
  wh <- which(status == 1)
  total <- concordant <- 0

  # Every individual i with an event is compared to all other individuals
  # with a later event/censoring time j (event times were sorted, ascending)
  for (i in wh) {
    for (j in ((i + 1):n)) {

      # The total number of individuals j with a later event/censoring time
      # than individual i is counted
      if (time[j] > time[i]) {
        total <- total + 1
      }

      # The total number of concordant an tied pairs is counted
      if (x[j] < x[i])
        concordant <- concordant + 1
      if (x[j] == x[i])
        concordant <- concordant + 0.5
    }
  }

  # The proportion of concordant pairs over the total of evaluable pairs
  # gives the C-index
  return(concordant/total)
}

#Appendix B.4: (80–120)
2.2.6 Dynamic predictions

5-year death predictions were estimated for every landmark time \( s \) for prediction window \( w = 5 \) years. The 5 year death prediction \( D(s + w|x, t_{LM} = s) \) is given by 1 minus the survival (2.15) in which the survival is estimated by taking the exponent of the negative cumulative hazard (2.16). The cumulative hazard is obtained by taking the difference between the cumulative hazard estimated at \( s + w \) and \( s \) (2.17).

\[
D(s + w|x, t_{LM} = s) = 1 - S(s + w|x, t_{LM} = s) \tag{2.15}
\]

\[
S(s + w|x, t_{LM} = s) = \exp(-H(s + w|x, t_{LM} = s)) \tag{2.16}
\]

\[
H(s + w|x, t_{LM} = s) = \exp(x^T \hat{\beta}_{LM}(s) + \hat{\theta}(s))(\hat{H}_0(s + w) - \hat{H}_0(s-)) \tag{2.17}
\]

The cumulative hazard is the integral of the hazard (2.18).

\[
H(s + w|x, t_{LM} = s) = \int_{s_{LM}}^{s_{LM}+w} \exp(x^T \hat{\beta}_{LM}(s) + \hat{\theta}(s))h_0(s)ds \tag{2.18}
\]

Below, the hazard for prediction time \( s \) is defined explicitly for the model fitted to the EURAMOS data (2.19). Note that \( x^T \hat{\beta}_{LM}(s) \) from (2.18) has been split out into time-constant and time-varying terms. The quantity \( \exp(x_c \beta^T_c) \) represents the time-constant contributions to the hazard, as do \( \exp(x_{LR} \beta_{LR}) \) and \( \exp(x_{NM} \beta_{NM}) \). The contributions from local recurrence (LR) and new metastatic disease (NM) have been explicitly defined, as they are of particular interest for the predictions made. The remaining terms make up the time-varying contributions for the histological response (HR) and the baseline hazard. The time-constant terms can be estimated directly from the model, while the time-dependent terms need to be estimated for each prediction time separately.

\[
h(s|x, t_{LM} = s) = \exp(x_c \beta^T_c) \ast \exp(x_{LR} \beta_{LR}) \ast \exp(x_{NM} \beta_{NM}) \ast \exp(x_{HR} \ast (\beta_{HR,LM1} \ast t + \beta_{HR,LM2} \ast t^2)) \ast \exp(\eta_{LM} \ast t) \ast \exp(\eta_{LM2} \ast t^2) \ast \hat{h}_0(t) \tag{2.19}
\]

In order to obtain the 5-year death prediction, a number of steps are taken. Below, a simplified version of the R code is shown, reported in full detail in Appendix B.5.
Note that this code is an adapted version of the code accompanying the Dynamic Prediction book.\textsuperscript{[24]} The method is illustrated for a single patient that experiences a local recurrence, shown in Table 2.4

Table 2.4: Example Patient

<table>
<thead>
<tr>
<th>Patient 1</th>
<th>12-18 years</th>
<th>&lt; 200</th>
<th>Radical/wide</th>
<th>Female</th>
<th>No</th>
<th>No</th>
<th>Hist.resp</th>
<th>LR</th>
<th>DM</th>
<th>LM</th>
<th>LM2</th>
</tr>
</thead>
</table>

For the patient-specific predictions, the patient characteristics need to be explicitly defined for all variables but LR and DM. And indicator variable \(xdata\) is later used to distinguish between the presence and absence of LR. The terms linear landmark time LM and quadratic time LM2, shown as zero here, are assigned values based on the prediction time. \(\exp(x_c \beta_T)\) and \(\hat{h}(t)\) from equation (2.19) are estimated directly from the model by using the \texttt{survfit()} function from the \texttt{survival} package to obtain the survival estimates, which then is used to calculate the cumulative hazard, and the small hazard \(h(s|x, t_LM = s)\). For a range of time points from 3 to 5 years, the following are calculated separately, then multiplied with the previously obtained hazard: the hazard of the time-dependent contribution of histological response- \(\exp(x_{HR} \ast \beta_{HR:LM} \ast t + \beta_{HR:LM2} \ast t^2)\) and the linear and quadratic time element terms of the baseline hazard- \(\exp(\eta_{LM} \ast t) \ast \exp(\eta_{LM2} \ast t^2)\). The same is done for the hazard contribution of LR - \(\exp(x_{LR} \beta_{LR})\). Note that this is a time-constant term and could have been included in the first step. It has been defined explicitly as it is more practical for the prediction function, given in full in B.5, 5-115. From the hazard \(h(s|x, t_LM = s)\) the cumulative hazard \(H(s|x, t_LM = s)\) is obtained. The cumulative hazard is evaluated at prediction time \(s\) and \(s + w\). The exponent of the negative difference of the two gives the survival estimate, 1 - the survival estimate gives the 5-year death prediction - the probability of dying within a window of 5 years.
# estimates survival curve for patient with given characteristics
sf <- survfit(mdl, newdata=patient)

# extracting the cumulative hazard
Haz0 <- data.frame(time=sftime, surv=sf$surv)
Haz0$Haz <- -log(Haz0$surv)

# defining time points for estimation, starting at 3 months post surgery
tt <- seq(3/12, 5, by=0.05)
nt <- length(tt)

# indicator variable, when 1 signifies that LR is present
xdata <- rep(1, nt)

# obtaining hazard from cumulative hazard
Haz0$haz <- diff(c(0, Haz0$Haz))

# For each time point probability is computed
Fw <- vector("numeric", length=length(tt))
for (i in 1:nt) {
  sfi <- Haz0  # local copy
  tti <- tt[i]

  # the hazard is multiplied with the time-dependent hazards, and the hazard for LR
  sfi$haz <- sfi$haz * exp(bet[paste(ev)]*xdata[i] + bet["LM2"]*tt[i]^2 + bet["LM1"]*tt[i] + margin*(bet["Hist.respPoor_hist:LM1"]*tt[i] + bet["Hist.respPoor_hist:LM2"]*tt[i]^2))

  # the cumulative hazard is obtained
  sfi$Haz <- cumsum(sfi$haz)

  # the cumulative hazard estimate for the time point and the time point plus
  # the prediction window (5 years) is obtained
  tmp <- evalstep(sftime, sfi$Haz, c(tti, tti+w), subst=0)

  # the difference of the cumulative hazards is taken. The exponent of the negative
  # difference gives the survival prediction, 1-the survival the death prediction
  Fw[i] <- 1-exp(-(tmp[2]-tmp[1]))
}

# Appendix B.5 (5–115)
2.3 Results

The EURAMOS-1 trial data contains information on 2260 patients, of which 1336 were randomized and 924 were not. Both randomized and non-randomized patients were included in the analysis. Non-randomized patients that were non-randomized due to progression of metastatic disease, the development a new metastatic disease, or the presence of an unresectable disease were excluded from the analysis. A total of 295 patients was considered to be ineligible for analysis. Please note that an extensive description of the trial and the patient characteristics can be found in chapter 1: Data description, sections 1.1 - 1.3.

2.3.1 Descriptives

Figure 2.1: Number of patients at risk at each landmark time point. Red, patients with LR (left) or NM (right); blue, patients without LR (left) or without NM (right).

A total of 1965 patients were considered to be eligible for analysis. Median follow-up time since surgery of 4.96 years (95% CI 4.87-5.08) was established using the reverse Kaplan-Meier approach. Figure 2.1 shows the number of patients used at the landmark times, along with their Local recurrence (LR) status (left) and their New metastatic disease (NM) status (right). A total of 73 patients experienced a local recurrence, 481 patients a new metastatic disease. And the end of study, 512 patients had died, while 1462 remained alive.
2.3.2 Dynamic prediction model for overall survival

The hazard ratios (HR) and 95% confidence intervals (95% CI) estimated from the Cox Proportional Hazard model are shown in Table 2.7. All predictors have significant time-constant effects. The strongest predictors are the occurrence of new metastatic disease, with a HR of 7.980 (95% CI 6.778-9.397), followed by local recurrence with a HR of 6.920 (95% CI 4.920-9.732). A poor histological response versus a good histological response gives a HR of 2.475 (95% CI 2.101-2.916). The presence of lung metastases and other metastases give HRs of 2.164 (95% CI 1.808-2.590) and 1.587 (95% CI 1.140-2.207), respectively. An unknown/intralesional excision, compared to a wide/radical one results in a HR of 1.619 (95% CI 1.213-2.163). The remaining predictor categories have more modest effects with hazard ratios smaller than 1.5. Finally, an age of < 12, when compared to an age of 12-18, has a mild protective effect with a HR of 0.729 (95% CI 0.583-0.913). Histological response has a significant time-varying effect, which is modelled by an additional linear term and a quadratic term, the latter of which is borderline non-significant. The hazard ratio for a patient with a poor histological response and a good histological response is calculated using the formula (2.19). Here, $HR_c$ is the time-constant component, $HR_{lin}$ the linear time-dependent component and $HR_{quad}$ the quadratic component, and $s$ is the prediction time in years.

$$HR_c \times HR_{lin}^s \times HR_{quad}^{s^2} = 2.475 \times 0.728^s \times 1.033^{s^2}$$ (2.20)

The calculations for prediction times of 1, 2, 3, 4 and 5 years are illustrated below. At a time of 1 years the HR has changed from 2.475 to 1.861. At the final prediction time of 5 years the HR has further decreased to 1.137.

Table 2.5: Hazard ratio (HR) values for a patient with a poor histological response versus a good histological response at different prediction time points $s$. The constant effect is given, and the linear and quadratic time-varying terms. For the hazard ratio the 95% confidence interval (95% CI) is provided.

<table>
<thead>
<tr>
<th>$s$</th>
<th>constant term</th>
<th>linear term</th>
<th>quadratic term</th>
<th>HR</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.475</td>
<td>0.728</td>
<td>1.033</td>
<td>1.861</td>
<td>(1.52-2.27)</td>
</tr>
<tr>
<td>2</td>
<td>2.475</td>
<td>0.728$^2$</td>
<td>1.033$^4$</td>
<td>1.493</td>
<td>(1.22-1.82)</td>
</tr>
<tr>
<td>3</td>
<td>2.475</td>
<td>0.728$^3$</td>
<td>1.033$^9$</td>
<td>1.278</td>
<td>(1.01-1.51)</td>
</tr>
<tr>
<td>4</td>
<td>2.475</td>
<td>0.728$^4$</td>
<td>1.033$^{16}$</td>
<td>1.167</td>
<td>(0.96-1.43)</td>
</tr>
<tr>
<td>5</td>
<td>2.475</td>
<td>0.728$^5$</td>
<td>1.033$^{25}$</td>
<td>1.137</td>
<td>(1.17-1.74)</td>
</tr>
</tbody>
</table>

32
A visual representation of the time-varying hazard is given in Figure 2.2, starting at 3 months from surgery. The closer the prediction time is to the time of surgery the greater the hazard ratio associated with a poor histological response. The decrease in HR over time is at first more steep, then more gradual. From four years onwards a HR of 1 falls within the confidence interval of poor histological response and there is no longer any significant effect.

Figure 2.2: Time-varying hazard ratio for histological response. Black: good histological response; blue: poor histological response. Dashed line: point-wise 95% confidence interval for poor histological response

In Figure 2.3 the 5-year death-probabilities are shown for a selection of patients. The patient characteristics are summarized in Table 2.6.

Table 2.6: Overview patient characteristics used in 5-year death predictions

<table>
<thead>
<tr>
<th>Patient</th>
<th>age</th>
<th>volume</th>
<th>excision</th>
<th>sex</th>
<th>lung.met</th>
<th>o.met</th>
<th>Hist.resp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient 1</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>Good hist</td>
</tr>
<tr>
<td>Patient 2</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>Poor hist</td>
</tr>
<tr>
<td>Patient 3</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Radical/wide</td>
<td>Female</td>
<td>Yes/Possible</td>
<td>No</td>
<td>Good hist</td>
</tr>
<tr>
<td>Patient 4</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Other</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>Poor hist</td>
</tr>
<tr>
<td>Patient 5</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Other</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>Poor hist</td>
</tr>
<tr>
<td>Patient 6</td>
<td>12-18</td>
<td>&lt; 200</td>
<td>Other</td>
<td>Female</td>
<td>Yes/Possible</td>
<td>No</td>
<td>Good hist</td>
</tr>
</tbody>
</table>
Table 2.7: Dynamic prediction model for overall survival: hazard ratios (HR) alongside 95% confidence intervals (n=1965)

<table>
<thead>
<tr>
<th>Factor</th>
<th>HR</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Histological Response</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Good HR</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Poor HR</td>
<td>2.475</td>
<td>2.10 - 2.92</td>
</tr>
<tr>
<td>Poor HR:s</td>
<td>0.728</td>
<td>0.64 - 0.82</td>
</tr>
<tr>
<td>Poor HR:s²</td>
<td>1.033</td>
<td>1.00 - 1.07</td>
</tr>
<tr>
<td><strong>Local Recurrence</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>6.920</td>
<td>4.92 - 9.73</td>
</tr>
<tr>
<td><strong>New metastatic disease (NM)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>7.980</td>
<td>6.78 - 9.40</td>
</tr>
<tr>
<td><strong>Age</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-18</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>&lt; 12</td>
<td>0.729</td>
<td>0.58 - 0.91</td>
</tr>
<tr>
<td>&gt;18</td>
<td>1.002</td>
<td>0.82 - 1.22</td>
</tr>
<tr>
<td><strong>Excision</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wide/Radical</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Marginal</td>
<td>0.960</td>
<td>0.75 - 1.23</td>
</tr>
<tr>
<td>Other/Intralesional</td>
<td>1.619</td>
<td>1.21 - 2.16</td>
</tr>
<tr>
<td><strong>Sex</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>1.244</td>
<td>1.05 - 1.48</td>
</tr>
<tr>
<td><strong>Volume</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;200</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>&gt;200</td>
<td>1.338</td>
<td>1.11 - 1.62</td>
</tr>
<tr>
<td><strong>Lung metastases</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yes/Possible</td>
<td>2.164</td>
<td>1.81 - 2.59</td>
</tr>
<tr>
<td><strong>Other metastases</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yes/Possible</td>
<td>1.587</td>
<td>1.14 - 2.21</td>
</tr>
<tr>
<td><strong>Prediction time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>0.742</td>
<td>0.68 - 0.81</td>
</tr>
<tr>
<td>s²</td>
<td>1.036</td>
<td>1.02 - 1.06</td>
</tr>
</tbody>
</table>
For these patients 5-year death probabilities were estimated, starting at 3 months post surgery to 5 years post surgery. For each prediction time point, the probability of dying within 5 years is shown. For each patient, estimates were made for that patient given no intermediate event (blue: no NM/LR), given a local recurrence (black: LR), and given a new metastatic disease (red: NM). The occurrence of new metastatic disease and local recurrence increase the 5-year probability of dying drastically for all patients. Only a slight difference in effect between the two intermediate events is observed, this was also evidenced by the previously reported HRs- 7.960 and 6.920 for NM and LR, respectively. Note that the lines for LR and NM are parallel as for neither prognostic factor a time-varying effect was found. In 1A a patient with reference categories is shown. Given that the patient is alive at 1 year after surgery, the probability of dying within 5-years, given that no local recurrence or new metastatic disease occur is 8.2% This probability increases with the occurrence of LR and NM to 44% and 49%, respectively. Poor histological response increases the probability of dying visibly, with a prediction of 24% at 1 year for no LR/NM, 85% for LR, 89% for NM (1B). The combination of poor histological response with an unknown/intralesional excision (2B), leads to an even higher 5-year death probability. The same is observed in 3B where two other unfavourable characteristics are combined- unknown intralesional excision and the presence of lung metastases.

For all patients, the later the prediction time, the lower the 5-year probability of dying. Compare for example a patient with poor histological response (1B) at a prediction time of 1 and 3 years who experiences NM. At 1 year the probability of dying within 5 years is 89%, which at 3 years has decreased to 68%. Notably, for most patients at a prediction time of 5 years, the 5-year probability of dying, has decreased to 1.3-4.3%, given no LR/NM and 8.5-30% given LR/NM. However, for poor histological response, and poor histological response combined with an unknown/intralesional excision, this probability remains considerably higher, at 6.7-11% given no LR/NM, and at 38-59% given LR/NM, for the patients respectively.

It must be noted that at 2.9 years there is a slight increase in the 5-year death probability. This can be explained by looking at the even distribution in the data set. In Figure 2.4 it can be observed that a single isolated event occurs at 7.9 years after surgery, at 5 years from the prediction time s=2.9. This results in a slight increase in the estimated hazard, apparent as a small bump in the 5-year death prediction plots.
Figure 2.3: 5-year death predictions for patients with different characteristics. 1A: Patient 1, reference categories characteristics. 1B: Patient 2, poor histological response. 2A: Patient 3, presence of lung metastases. 2B: Patient 4, poor histological response and an unknown/intralesional excision at surgery. 3A: Patient 5, unknown/intralesional excision at surgery. 3B: Patient 6, unknown/intralesional excision at surgery and lung metastases. Blue: No LR/NM; black: with LR; red: with NM.
2.3.3 Internal calibration

Model calibration was evaluated by means of a heuristic shrinkage factor. A heuristic shrinkage factor of 0.997 was calculated, which indicates that the model is very well calibrated. The discriminative ability of the model was assessed by dynamic cross-validated concordance indices, calculated at prediction times of 0, 1, 2, 3, 4 and 5 years. For these prediction times, C-indices of 0.690, 0.744, 0.807, 0.795, 0.761, and 0.772, were found, respectively, indicating a very good discriminative ability of the model. This is due to the strong predictive value that NM and LR have on survival. The presence of one of these intermediate events results in a much worse prognosis.
2.4 Discussion

This analysis shows that the dynamic prediction approach captures relevant information that would be unavailable from a static prediction model. For instance, it was shown that the prediction time, in years from surgery, strongly influences the probability of dying within 5 years. The longer a patient survives, the smaller the 5-year death probability is. The dynamic prediction approach also allows for dynamic variables - here, local recurrence and new metastatic disease. Both were shown to have a strong impact on overall survival, with hazard ratios of 7.960 and 6.920, respectively.

Other strong predictors are histological response (HR 2.475) and the presence of lung metastases (HR 2.164), which is consistent with previous reports where poor histological response and presence of metastases have been associated with decreased survival.\[14, 16, 32\] Of note is that an unknown/intralesional excision, when compared to a wide/radical excision decreases the overall survival (HR 1.619). Unlike the other predictors considered, excision is not an observed variable and is subject to intervention.

For histological response a time-varying effect was found. The hazard ratio for a poor histological response versus a good one decreases from 2.475 at a prediction time of 0 years to 1.137 at 5 years. While histological response is a strong predictor directly after surgery, it becomes weaker with time, giving no significant effect from 4 years onwards. Histological response is an important and commonly used predictor in the clinical framework, and it would be of relevance to examine the behaviour of histological response with regards time in other datasets.

The model performance was evaluated with cross-validated dynamic C-indexes, ranging from 0.690 to 0.807, which indicates a very good discriminative ability.

The prediction model developed in this study can be used to make reliable predictions of the probability of dying within 5 years of a given prediction time, in which the prediction time can be any time from surgery to 5 years after surgery. This model has accounted for the time-dependent nature of covariates local recurrence (LR) and new metastatic disease (NM), and for the time-varying effect of histological response.
Chapter 3

A Neural network approach

In this chapter a neural network approach for the EURAMOS osteosarcoma data is described. Preceding this a section on the general theory of neural networks is provided, along with a summary of the various approaches used in adapting neural networks to be suitable for censored survival data. The application section consists of subsections on data transformation, the performance measures used, the tuning of hyperparameters by means of cross-validation, and a comparison of the results of the optimal neural network with the Cox model. Findings and remaining avenues of investigation are summarized in the final discussion section. Code excerpts are given throughout the chapter with the full code provided in Appendix C.

3.1 An introduction to Neural Networks

Artificial neural networks (ANN) - also known as multi-layer perceptrons - are a machine learning method capable of modelling non-linear relationships with great flexibility. An artificial neural network relies on nodes - sometimes referred to as neurons. Connections between the nodes are assigned weights, and these weights are the parameters the neural network estimates. The parameters are jointly estimated using the back-propagation algorithm. For the purpose of training a neural network, a target is defined, which is the observed outcome. The neural network attempts to minimise the difference between the observed responses and the output. The objective is to train a neural network that will generalize well to new data.
Figure 3.1: Single hidden layer feed forward neural network. X represents the inputs, H the hidden nodes, and K the output node. The nodes are uni-directionally connected, and the corresponding weights are w. B represents the bias nodes.

In the most simple neural network, three layers of nodes are defined - an input layer, a single hidden layer and an output layer (figure 3.1). The input layer contains nodes for variables of interest and (optionally) a bias node, the connection weight of which is a constant and can be interpreted similarly to an intercept in a regression. The hidden layer contains a number of hidden nodes, and once again a bias node. The output layer is the prediction or classification made by the neural network, and can have one or more nodes. A feed forward neural network is a popular choice of network that has also been frequently used with time-to-event data. Feed forward refers to uni-directional connections - from input to hidden layer, from hidden layer to output. Feed forward artificial neural networks can be considered a non-linear multivariate regression method. [10]

Let $X_j$ ($j = 1, 2, ..., J$) be the input variables, $H_h$ ($h = 1, 2, ..., H$) the hidden nodes, and $K_k$ ($k = 1, 2, ..., K$) the output nodes. The network is trained on a dataset of n subjects. The network estimates an output $\hat{y}_{ik}$ for each subject $i$ and output node $k$. The network is trained until the estimated outputs $\hat{y}_{ik}$ approximate the observed values $y_{oik}$, where the superscript $o$ denotes an observed value, whereas a hat symbol denotes an estimated value. A mathematical representation of a single
layer feed forward artificial neural network is given in (3.1).

$$\hat{y}_k(x_i, w) = \phi_o(\alpha_k + \sum_{h=1}^{H} w_{hk} \phi_h(\alpha_h + \sum_{j=1}^{J} w_{jh} x_{ij}))$$  \hspace{1cm} (3.1)$$

A node in the hidden layer is the sum of the product of the weights $w_{jh}$ and inputs $x_{ij}$ plus a bias term $\alpha_h$. An activation function $\phi_h(u)$ is applied to each node. Common choices are a sigmoid or logistic activation function (3.2).

$$\phi_h(u) = \frac{exp(u)}{1 + exp(u)}$$  \hspace{1cm} (3.2)$$

An activation function is a non-linear transformation applied to the input and is also referred to as a threshold or transfer function. Based on the input and specific choice of function information will be either transmitted or not.[2] The transformed input serves in turn as the input to the output layer. The sum is taken over the hidden neurons $H$ of the product of the transformed input and the weights $w_{hk}$. Once again a bias term $\alpha_k$ is added. The final transformation $\phi_o$ is applied to the output and optional.

In order to estimate the optimal weights $w$ an error function is minimised. A traditional choice is the quadratic error function (3.3). For binary classification the suitable error function is the cross-entropy error (3.4).

$$E = \sum_{k=1}^{K} \sum_{i=1}^{n} (\hat{y}_k(x_i, w) - y_{ik}^o)^2$$  \hspace{1cm} (3.3)$$

$$E = -\sum_{k=1}^{K} \sum_{i=1}^{n} y_{ik}^o \log \hat{y}_k(x_i, w) + (1 - y_{ik}^o) \log[1 - \hat{y}_k(x_i, w)]$$  \hspace{1cm} (3.4)$$

The number of nodes in the hidden layer is variable. With every additional node a more complicated non-linear relationship can be modelled, but also the likelihood of training an over-fitted model increases. An over-fitted model will perform very well on training data, but will not generalize well to new observations. A neural network can be regulated by introducing a decay parameter $\lambda$, which penalizes large weights. The original loss $E$ is modified, given $E^*$ with

$$E^* = E + \lambda \sum w^2.$$  \hspace{1cm} (3.5)$$

Introducing a weight decay penalty will not only decrease the chances of over-fitting the model, it will also improve the convergence of the network, which is especially relevant for large datasets and complicated networks. Common choices for the
\(\lambda\) penalty range from 0 to 0.1. Both the weight decay parameter and the number of hidden neurons need to be optimized, and typically a cross-validation approach is used. Two thirds of the data functions as a training set, while one third is put aside to serve as a final validation set. \(K\)-fold cross-validation, usually 5- or 10-fold, is used on the training set, and the optimally performing hyperparameters are identified. The final model performance is then assessed by using the entire training set to train the network and the left out validation set to test it. Various performance measures can be used for assessing model performance. Of interest in this context is of course a measure suitable for survival data. The standard for assessing survival model performance is the concordance index. This measure is however not suitable in a neural network context as it relies on a ordering of subjects on prognosis. The choice of performance measures is discussed in detail in Section 3.4.2.

While neural networks are capable of modelling complex non-linear relationships, the actual contributions of the input variables cannot be easily assessed. Neural networks are famously known as a black box procedure. While non-linearity and complex interactions are easily accounted for this comes at the cost of interpretability. Various methods, however, have been suggested that attempt to assign importance to variables based on the model weights. Olden, Joy and Death\cite{29} identified the connection weight method \cite{28} as the one providing the most accurate quantification of variable importance. The connection weight approach uses the raw input-hidden and hidden-output connection weights to calculate the variable importance (VI) (3.6). For each input, the connection weight \(w_{jh}\) of the input \(X_j\) with the hidden node \(H_h\) is multiplied with the connection weight \(w_{hk}\) of the hidden neuron \(H_h\) with the output \(K_k\). This is repeated for all hidden nodes that the input \(X_j\) is connected to, after which the products are summed. An absolute measure of variable importance (VI) is obtained, which, given that the variable importance for each input variable is known, is easily converted to a relative measure.

\[
VI_j = \sum_{h=1}^{H} w_{jh}w_{hk}. \tag{3.6}
\]

### 3.2 An overview of previously applied methods

A considerable number of approaches have been suggested for the modelling of survival data. De Laurentiis & Ravdin \cite{41} identified three situations in which neural
networks are valuable alternatives the Cox proportional hazards model: 1) The proportional hazards assumption does not hold. 2) the variables have a complex and unknown relationship with the outcome, and 3) there are interactions between the variables. In this section various approaches are discussed. Tables 3.1 and 3.2 summarize the characteristics of the methods covered in this section. For each method, the estimated quantity of the network is given, the kind of neural network (distinguishing between a feed forward neural network (FFN) and a restricted Boltzmann machine), the activation function applied to the hidden layer, the activation function applied to the outer layer, the error function that is minimised, the number of nodes in the output layer of the network, and the number of networks. Please note that an excellent review article on the subject of neural network approaches to survival analysis is available from Ripley & Ripley.[4]

Table 3.1: Neural network characteristics (part 1)

<table>
<thead>
<tr>
<th>Estimated:</th>
<th>Neural network</th>
<th>Act. hidden</th>
<th>Act. out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street (1998)</td>
<td>$S_t$</td>
<td>FFN</td>
<td>hyperbolic tangent</td>
</tr>
<tr>
<td>Chi &amp; Street (2007)</td>
<td>$S_t$</td>
<td>FFN</td>
<td>sigmoid</td>
</tr>
<tr>
<td>Nilsaz-Dezfouli (2017)</td>
<td>$S_t$</td>
<td>FFN</td>
<td>logistic</td>
</tr>
<tr>
<td>Kappen (1995)</td>
<td>$S_t$</td>
<td>Boltzmann</td>
<td>/</td>
</tr>
<tr>
<td>Theeuwen (1993)</td>
<td>$S_t$</td>
<td>Boltzmann</td>
<td>/</td>
</tr>
<tr>
<td>Ravdin &amp; Clark (1992)</td>
<td>$S_t$</td>
<td>FFN</td>
<td>hyp. tangent</td>
</tr>
<tr>
<td>Biganzoli (1998)</td>
<td>$h_t$</td>
<td>FFN</td>
<td>logistic</td>
</tr>
<tr>
<td>Liestol (1994)</td>
<td>$h_t$</td>
<td>FFN</td>
<td>logistic</td>
</tr>
<tr>
<td>Montes-Torres (2017)</td>
<td>$h_t$</td>
<td>FFN</td>
<td>undefined</td>
</tr>
<tr>
<td>Ohno-Machado (1996)</td>
<td>$p_t$</td>
<td>FFN</td>
<td>undefined</td>
</tr>
<tr>
<td>Lapuerta (1995)</td>
<td>$p_t$</td>
<td>FFN</td>
<td>sigmoid</td>
</tr>
</tbody>
</table>

Table 3.2: Neural network characteristics (part 2)

<table>
<thead>
<tr>
<th>Error function</th>
<th>No. outputs</th>
<th>No. network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street (1998)</td>
<td>relative entropy</td>
<td>multiple</td>
</tr>
<tr>
<td>Chi &amp; Street (2007)</td>
<td>relative entropy</td>
<td>multiple</td>
</tr>
<tr>
<td>Nilsaz-Dezfouli (2017)</td>
<td>cross-entropy</td>
<td>single</td>
</tr>
<tr>
<td>Kappen (1995)</td>
<td>KL</td>
<td>single</td>
</tr>
<tr>
<td>Theeuwen (1993)</td>
<td>KL</td>
<td>single</td>
</tr>
<tr>
<td>Ravdin &amp; Clark</td>
<td>undefined</td>
<td>single</td>
</tr>
<tr>
<td>Biganzoli (1998)</td>
<td>cross-entropy</td>
<td>single</td>
</tr>
<tr>
<td>Liestol (1992)</td>
<td>modified cross-entropy</td>
<td>multiple</td>
</tr>
<tr>
<td>Montes-Torres (2017)</td>
<td>cross-entropy</td>
<td>multiple</td>
</tr>
<tr>
<td>Ohno-Machado (1996)</td>
<td>cross-entropy</td>
<td>multiple</td>
</tr>
<tr>
<td>Lapuerta (1995)</td>
<td>? cross-entropy</td>
<td>multiple</td>
</tr>
</tbody>
</table>
Approaches can be distinguished based on the quantity they estimate - the probability of survival $S_t$, the conditional probability of dying $h_t$ and the unconditional probability of death $p_t$ - and by this the structure of this section is informed. Within any of these categories, additional distinctions can be made. First, the networks can be separated on the type of artificial neural network. Most frequently, feed forward neural networks (FFN) are implemented. Another possibility, less popular in recently, is the usage of a restricted Bolzmann model. FFNs are deterministic structures, whereas Bolzmann is stochastic. Then, distinctions can be made on the structure of the network itself- some authors define networks with k output nodes for k intervals others use a single output node, and other yet define K separate neural networks - or on the transfer functions used - sigmoid, logistic, hyperbolic tangent. Approaches can also be distinguished on how the censoring is dealt with. Kaplan Meier and less frequently Cox models have been used to impute hazard and survival estimates for censored observations. Other methods deal with censoring by modifying the data structure, or by neural network based imputation. Neural networks can also be separated on the error function. Typically the cross-entropy error function is considered appropriate for the estimation of probabilities. Depending on the data and network structure however, modification may be necessary. A final distinction can be made on implementation. Neural network approaches in R are limited to cross-entropy and squared error functions.

3.2.1 Estimating survival directly:

Street [58] proposed a neural network approach for censored survival data and demonstrated its application to breast cancer data, in which the time to recurrence after surgery was modelled. In this approach $K$ output nodes are defined for $K$ time intervals. In the application 10 output units were defined. The first unit represents the class of subjects with recurrences within one year following surgery, the second recurrences between one and two years, with the last unit representing recurrences between nine and ten years. The input of the neural network consists of the variables of interest, and the neural network target is presented in such a way that the output can be interpreted as the probability of disease free survival up until that time. For a given individual $i$, and for a given time interval $k$ ($k = 1, 2, ..., K$), the probability of disease-free survival is then given by $P(T_i > t_{ki}) = S(t_{ki})$. In order to be able to interpret the output as probabilities, the relative entropy error function is minimised in the process of weight estimation.
Table 3.3: Street’s neural network approach: defining targets

<table>
<thead>
<tr>
<th></th>
<th>(K_1)</th>
<th>(K_2)</th>
<th>(K_3)</th>
<th>(K_4)</th>
<th>(K_5)</th>
<th>(K_6)</th>
<th>(K_7)</th>
<th>(K_8)</th>
<th>(K_9)</th>
<th>(K_{10})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pat 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pat 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.85</td>
<td>0.80</td>
<td>0.78</td>
<td>0.70</td>
<td>0.65</td>
<td>0.60</td>
<td>0.55</td>
</tr>
</tbody>
</table>

An example of a possibly target vector is shown for two patients (Table 3.3). The first patient \(Pat1\) has a recurrence in the interval \(K_4\)- in between three and four years. Up until the interval of recurrence the target vector is assigned 1s, after that 0s. For censored patients, only the disease free survival time (DFS) is known. For the periods following, the probability of recurrence \((risk_k)\) is estimated using the Kaplan Meier estimator, and the risk estimates are used to compute survival probabilities. As such, for censored cases, the target vector at interval \(k\) is given by

\[
S_k = \begin{cases} 
1, & 0 \leq t \leq DFS_i \\
S_{k-1}(1 - risk_k), & t > DFS_i 
\end{cases}
\]  \hspace{1cm} (3.7)

In the example shown in Table 3.3 the patient \(Pat2\) is censored in the third interval \(K_3\). In the original article Street suggested a hyperbolic tangent activation function for the hidden units and a logistic transformation for the output.[58] In a follow-up article, in collaboration with Chi and Wolberg [6] a slight modification was proposed in the definition of the target vectors and the activation function used - sigmoid rather than hyperbolic tangent. The description above, is tailored to this second article, but Street’s original approach is easily derived.

Nilsaz-Dezfouli [20] defined \(K\) separate networks rather than \(K\) different outputs, for four different time periods. In the reported application, the probability of dying at or before 1 year, 2 years, 3 years and 4 years were considered. For each a separate network was fitted. Rather than estimating the survival probabilities \(S_k\) directly, the probability of death \(1 - S_k\) was estimated. For a given network with output node \(k\) a patient experiencing the event in the interval \(k\) was assigned a 1 as target, a patient censored after that interval a 0, and a patient censored before or within the interval \(\hat{p}\), which is the estimated probability of dying before the end of the interval. This quantity was estimated for censored patients using the Cox model. The authors used a logistic activation function for both the transmission from input units to hidden units, and from hidden units to output units. The cross-entropy function was used as error function.
A similar approach of $K$ separate outputs has been used previously by Kappen [25], and Theeuwen [45]. A drawback is that this approach may not result in a monotonically decreasing survival function. For instance, the probability of death at or before two years - in years 1 or 2 - may be reported to be higher than the probability of death at or before three years - in years 1, 2 or 3. It must be noted that the approach described by Kappen and Theeuwen use a restricted Boltzmann machine as a neural network. Instead of using deterministic units (i.e. logistic) it uses stochastic units with a particular distribution. In the other approaches discussed here a deterministic approach is used.

Ravdin & Clark [47] proposed a single neural network with a single output. Patients are replicated for all the considered time intervals. Consider an example where five subsequent time intervals are defined, and a patient experiences an event in the fourth interval. Then a target would be defined as shown in Table 3.4. The patient is assigned a 0 for the intervals in which he is alive, a 1 for the interval of event and all intervals after.

Table 3.4: Ravdin & Clark: data structure

<table>
<thead>
<tr>
<th>id</th>
<th>target</th>
<th>interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

In a way similar to Biganzoli’s approach [10], detailed in the next section, a vector indicating the time interval is included as an input variable in addition to the predictors. A difference from Biganzoli’s approach, however, is that patients are replicated for the intervals based not solely on follow-up time, but also on the Kaplan-Meier survival estimates. Recall that patients who experience an event have targets defined for the interval of event and all intervals following after. Censored patients, however, provide information only up until the interval of censoring. As such, in later intervals patients who experience an event - non-survivors - are overrepresented. The authors give the following example. For a given time interval they observe that the Kaplan Meier estimate measures 50% survival, indicating that there should be an equal number of entries for survivors and non-survivors. Due to censoring, however, in that interval there are 213 observed non-survivors, who died in this interval or
a previous one, and 45 survivors. The bias is corrected for by randomly sampling 45 non-survivors out of the original 213, and removing the remaining ones. For each interval, observations are randomly selected based on the observed Kaplan-Meier estimates. Hyperbolic tangent functions are used both for the hidden and output layer. The authors do not specify the error function that is minimised beyond 'backward propagation of errors', which is the standard algorithm for supervised neural networks.

### 3.2.2 Estimating the hazard:

Let $A_l$ be a time interval for $l = 1, 2, ... L$ disjoint intervals. In a subset of neural network approaches the hazard $h_l$ is estimated - the conditional probability of dying in an interval $l$ given that the patient has survived until interval $l$. The survival function for a given patient $i$ can then be estimated over the specified intervals by

$$S(t_i) = \prod_{l=1}^{t_i} (1 - h_l)$$

One of the most well-used approaches in this category is the partial logistic artificial neural network approach (PLANN), described by Biganzoli et al. (1998). Rather than imputing censored observations with survival/hazard estimates, information is only specified as it is observed. A single output is defined, and a subject is replicated for the number of intervals that it is observed in the study. Consider Table 3.5 shown below. The first patient ($id=1$) experiences an event in the third interval, the second patient ($id=2$) is censored after the fifth interval. Both patients are not repeated for intervals in which they are not observed - they are truncated after the interval of event/censoring. Rather than a target vector per patient, now a single joint vector is provided for all patients. To distinguish between intervals, the interval number is added as an additional input variable. A cross-entropy loss function is used, which is shown to be equal to the negative log-likelihood, and as such minimizing the loss function maximizes the likelihood. A logistic transfer function is used both in the hidden and output layer.

Liestol [35] described a $K$ output neural network, in which each output $K$ corresponds to a time interval, and the conditional death probability is estimated. Similar to Biganzoli’s approach a logistic activation function is used and the cross-entropy error function is minimised. A target vector is defined for each individual, the length
of which varies depending on the number of intervals for which the patient is observed. While it is possible to implement this method, it requires a modification of the error function, and unlike Biganzoli’s PLANN approach cannot be implemented in standard software.

Montes-Torres et al. (2017) reported a $K$ output neural network for the estimation of the hazard. The authors implemented this neural network approach in an online tool for survival predictions, using the R package \texttt{nnet}. Values for censored observations were imputed with the hazard estimates, which are calculated by taking the number of deaths $d_l$ observed for a given interval $A_l$, and dividing it by the number at risk $n_l$: $\frac{d_l}{n_l}$. These estimates are used to impute values for the intervals in which a censored subject is not observed. An illustration of the approach is given in Table 3.6, for 10 intervals. Pat 1 shows a patient with an event in the fourth interval $A_4$. In the three intervals prior the patient is assigned 0s, indicating the he is known to be alive. In the interval of event and the subsequent intervals the patient is assigned 1s, to denote the patient is known to be dead. Pat 2 shows a patient who is censored after the fourth interval. For the first four intervals the patient is known to be alive and accorded 0s. After this, however, it is unknown whether the patient, had he been observed, would be alive or dead, and values are imputed using the hazard. For example, for interval $K_5$ $\frac{d_5}{n_5} = 0.1$

<table>
<thead>
<tr>
<th>Table 3.6: Montes-Torres method: defining targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
</tr>
<tr>
<td>Pat 1</td>
</tr>
<tr>
<td>Pat 2</td>
</tr>
</tbody>
</table>

This approach closely mirrors the targets as specified for Street’s model, shown in Table 3.3. For the patient who experiences an event, however, the 1 and 0 indi-
cators are reversed, and the quantity estimated by Montes-Torres's model could be interpreted as \( F_l = 1 - S_l \). Each 1 would denote the probability of being dead in the corresponding interval. For the censored patient, however, the instantaneous conditional risk is given rather than an estimate of \( F_l \), and as such there appears to be an inconsistency in the target vectors assigned to patients with an event and to censored patients, in that for the former the probability for being dead appears to be specified, and for the latter the probability of dying. It must be noted that the authors calculate the survival using \( S(t_l) = \prod_{l=1}^{t_i} (1 - h_{li}) \). Rodrigo & Tsokos [22] defined target vectors in the same manner as Montes-Torres, but estimate the survival for individual \( i \) in interval \( l \) from output \( y \) as \( S(t_i) = \exp(-\sum_{l=1}^{t_i} y(l | r, X_i)) \). The error function minimised in Rodrigo & Tsokos approach has been described previously by Fornili [42], and is a custom function composed of both the cross-entropy error and quadratic error.

### 3.2.3 Estimating the unconditional death probability:

In the final approach the unconditional probability of dying in a given interval \( k \) \( p_k \) is estimated. As such the survival \( s_k \) is given by \( s_k = 1 - p_1 - p_2 - \ldots - p_k \). Ohno-machado [40] defined a model with four output nodes, representing death at or before 1 year, death in between 1 and 2 years, in between 2 and 3 years, and death after 3 years. The network may return an output of probabilities 0.1, 0.3, 0.2 and 0.4 for the patient dying in each of the respective four intervals. Then, the probability of being dead would be given by 0.1, 0.4, 0.6 and 1, and the probability of survival would be obtained by subtracting the values from 1. For this network, the cross entropy function was minimised for each node. The type of activation function was not reported. It should also be noted that this method appears to offer no apparent way of dealing with censoring. In comparison, Lapuerta [46], prior to fitting a \( K \) node neural network, defined \( K \) separate networks with the purpose of imputing missing data. In the event of a censored patient, this patient would be presented to the networks corresponding to the subsequent intervals until an event was predicted. In the application Lapuerta defined four time periods. A patient may experience an event in period 1, period 2 or period 3, while period 4 represents no event. In Figure 3.2, a patient lost to follow-up in period 1 is shown. Question marks denote the periods for which the information is lacking. This patient is presented to the period 2 network, which will either predict an event or not. In the latter case, the patient will be presented to the next network, the period 3 network, and so forth. After imputation, a neural network with \( K = 4 \)
output nodes is trained for the purpose of classifying the test set. Sigmoid transfer functions were used for all the networks.

This data structure may be reminiscent of the one used by Nilsaz-Dezfouli [20] or Kappen [25] and Theeuwen [45], described previously. These methods, however, estimated the survival directly and utilized a (modified) single time point approach. In the Ohno-Machado [40] and Lapuerta [46] approaches, the probability of dying in or before that interval is estimated.

Beyond the scope of this review are methods utilizing deep multi-layer neural networks and hybrid approaches of neural networks and existing statistical models. In the former category recently a paper has been published by Gensheimer & Narasimham [43], in which the neural network approach is clearly detailed in mathematical terms, and accompanied by source code for implementation in the keras deep learning library. [30] In the latter category, Faraggi & Simon [9] first proposed an adaptation of the Cox proportional hazards model, later applied by Mariani [39] to breast cancer data. Recently, this approach has received increased attention, and two implementations have been developed for Python- DeepSurv by Katzman [31] and Cox-nnet by Ching [51].
3.3 Biganzoli’s PLANN approach in detail

A short introduction to Biganzoli’s approach [10] was given in Section 3.2.2. As stated there, the data is transformed to a longitudinal format and a one-layer feed-forward neural network is defined with a single output node. Survival times are split up into intervals, and each subject is repeated for the number of intervals he/she is observed. The target of the network is a single long indicator vector, with zero entries except for the interval in which an event is observed. Table 3.5 shows the simplified data structure of Biganzoli’s approach. In Section 3.4.1 the data transformation as applied to the EURAMOS osteosarcoma data is illustrated in Tables 3.7 and 3.8. The schematic structure of the network corresponds to the example shown in Section 3.1 in Figure 3.1. Prognostic variables and a time interval variable constitute the input, which connects to a single layer of a variable number of hidden nodes, which in turn connects to a single output node. In the input and hidden layer a bias node is present which connects to the hidden and output layer, respectively. As noted in Section 3.1, the bias node can be interpreted as an intercept and does not have any real value for model interpretation. Weights are estimated for all the connections between nodes by minimizing the cross-entropy loss function. As the cross-entropy loss function is equal to the negative logarithm of the log-likelihood, this results in the estimation of the conditional death probabilities. A trained network can be used to make predictions for a test set. Test subjects are replicated for all the intervals of interest and fed into the network, giving as output a long vector with conditional death probabilities for each time interval for each individual.

More formally, the previously detailed process can be described as follows. Continuous survival times are grouped into \( l=1,2,...,L \) disjoint intervals \( A_l = (t_{l-1}, t_l) \). The model estimates the discrete hazard rate \( h_l \) for each interval \( A_l \), which is the conditional failure probability defined below (3.8) where

\[
h_l = \frac{S(t_{l-1}) - S(t_l)}{S(t_{l-1})},
\]

with

\[
S_{t_l} = P(T > t_l),
\]

from which follows that

\[
S_t = \prod_{l:t_l < t} (1 - h_l).
\]
The conditional failure probabilities for the time intervals $h_l$ approach the continuous hazard function as the intervals become smaller. In terms of the likelihood function, subjects with an event contribute the product of the conditional failure probability $h_l$ for the interval $A_l$ in which the event occurs, and the conditional survival probabilities $(1 - h_l)$ for the preceding intervals in which the subject is observed.

\begin{equation}
P(T_i \in A_l) = h_l \prod_{l=1}^{l_i-1} (1 - h_l). \tag{3.11}
\end{equation}

Uncensored subjects contribute the conditional survival probabilities for the intervals in which they are observed.

\begin{equation}
P(T_i > t_i) = \prod_{l=1}^{l_i-1} (1 - h_l). \tag{3.12}
\end{equation}

Using event indicator vector $d_{ij}$, which per convention denotes an event as 1 and no event as 0, taking the product over $n$ subjects and $i$, (3.11) and (3.12) give

\begin{equation}
L = \prod_{i=1}^{n} \prod_{l=1}^{l_i} h_{dl}^{d_{ij}} (1 - h_{dl})^{1 - d_{ij}}. \tag{3.13}
\end{equation}

Taking the negative logarithm of the likelihood, and adjusting it to include covariate vectors $x_i$ (3.14) is obtained, which is equal to the cross-entropy function given in Section 3.1 (3.4)

\begin{equation}
E = - \sum_{i=1}^{n} \sum_{l=1}^{l_i} d_{il} \log h_l(x_i, a_i) + (1 - d_{il}) \log[1 - h_l(x_i, a_i)]. \tag{3.14}
\end{equation}

As such, minimizing the cross-entropy function is equal to maximizing the likelihood, and is an appropriate choice for estimating the weights of the neural network. Using a logistic transformation on the outputs ensures that the conditional failure probabilities are returned. A transformation for outputs is optional in a neural network, but necessary in this context. An activation function for the hidden layer is essential for any neural network, and for the PLANN approach a logistic function is typically chosen (3.2).
3.4 Application to EURAMOS osteosarcoma data

In this section the application of the neural network approach to the EURAMOS osteosarcoma data is described. First, the necessary data transformation is described, then the performance measures used, followed by a cross-validation using these performance measures to identify the optimal hyperparameters (number of hidden nodes and weight decay parameter). Finally, the results are described and compared to the results from a Cox proportional hazards model. Findings and remaining avenues of investigation are summarized in the final discussion section. All analyses were performed in the R-software environment version 3.4.2. Packages used are nnet, survival, pec, devtools.

3.4.1 Data transformation

In order to fit a neural network as described by Biganzoli, survival times need to be assigned to intervals and the data needs to be in long format. For the data used for training the network each observation is repeated for the number of interval a patient is observed. An interval is defined as a month long period, starting from surgery, to the end of follow-up at 8.99 years.

Table 3.7: Original data format(wide)

<table>
<thead>
<tr>
<th>stat</th>
<th>id</th>
<th>Age</th>
<th>Volume</th>
<th>Hist</th>
<th>Ex</th>
<th>Sex</th>
<th>lung</th>
<th>other</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>353</td>
<td>9.57</td>
<td>205.95</td>
<td>Poor</td>
<td>Rad.</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>0.24</td>
</tr>
<tr>
<td>0</td>
<td>1233</td>
<td>17.14</td>
<td>145.60</td>
<td>Poor</td>
<td>Rad.</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 3.8: Transformed data format for training data (long)

<table>
<thead>
<tr>
<th>Stat</th>
<th>id</th>
<th>Age</th>
<th>Volume</th>
<th>Hist.P</th>
<th>Ex.M</th>
<th>Ex.O</th>
<th>Sex.M</th>
<th>L.y</th>
<th>O.y</th>
<th>Surv</th>
<th>ITvec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>353</td>
<td>-1.04</td>
<td>-0.004</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>353</td>
<td>-1.04</td>
<td>-0.004</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>353</td>
<td>-1.04</td>
<td>-0.004</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1233</td>
<td>0.37</td>
<td>-0.237</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.7 shows the original data format, for subjects 353 and 1233 who were both censored at 0.24 and 0.03 years after surgery, respectively. After the data transformation, shown in Table 3.8, subject 353 has been assigned a survival interval of the 3rd month, and subject 1233 a survival interval of the 1st month (surv). Subjects 353 and 1233 have been replicated for all the intervals they were observed: intervals 1, 2
and 3, and interval 1, respectively \((ITvec)\). Additionally, in the data transformation, continuous variables are normalized to increase training speed. As a neural network does not accept factors, categorical predictors are split out into indicator variables. Below, the data transformation function \(data\.transformer()\) is given, as used on the training data. First the predictors are scaled, then intervals are assigned, and categorical predictors are split out into indicator vectors. Finally, the data is replicated for the number of intervals a subject is observed.

\[
\text{data}\_\text{transformer} \leftarrow \text{function}\(\text{pot}\)\{
    \# scaling of variables
    \text{pot} \leftarrow \text{pot}[\text{order}(\text{pot}\_\text{id}),]
    \text{pot}\_\text{Age} \leftarrow \text{scale}(\text{pot}\_\text{Age}); \text{pot}\_\text{Volume} \leftarrow \text{scale}(\text{pot}\_\text{Volume})
    \# finding interval for each time point
    \text{alt}\_.\text{Int} \leftarrow \text{seq}(0,9,1/12)
    \text{ITvec} \leftarrow \text{vector}("\text{numeric}", \text{length=\text{nrow}(\text{pot})})
    \text{for} (i \in 1: \text{\text{nrow}(\text{pot})})\
    \quad \text{ITvec}[i] \leftarrow \text{findInterval}(\text{pot}\_\text{time}[i], \text{alt}\_.\text{Int})
    \text{pot} \leftarrow \text{cbind}(\text{pot}, \text{ITvec})
    \# creating indicator variables for categorical covariates
    \text{n}\_.\text{mat} \leftarrow \text{\text{as.data.frame}(model.matrix(~ -1 + Hist + Ex + Sex + lung + other, \text{pot})})
    \text{colnames}(\text{n}\_.\text{mat}) \leftarrow \text{c("Hist\_G", "Hist\_P", "Ex\_M", "Ex\_O", "Sex\_M", "L\_y", "O\_y")}
    \text{new}\_.\text{pot} \leftarrow \text{cbind}(\text{pot}\_\text{stat}, \text{pot}\_\text{id}, \text{pot}\_\text{Age}, \text{pot}\_\text{Volume}, \text{pot}\_\text{ITvec}, \text{n}\_.\text{mat})
    \text{uns} \leftarrow \text{\text{sort}(unique(\text{ITvec}))}
    \# function for replicating observation for the number of intervals in which it occurs
    \text{trans}\_.\text{func} \leftarrow \text{function}(\text{x})\{\}\
    \text{pot}\_.\text{l} \leftarrow \text{\text{\text{split}(new.pot, new.pot}\_\text{id})}
    \text{pot}\_.\text{t} \leftarrow \text{\text{\text{do.call("rbind", lapply(pot.l, trans.func))}}}
    \text{return}(\text{pot}\_.\text{t})\}
\]

The function \(\text{trans}\_\text{func}()\) ensures for the training data that each subject is re-
peated for the number of intervals he/she is observed.

```r
trans.func <- function(x){
  # Surv gives the observed interval for each patient
  val <- x$Surv
  # index interval (in case intervals are skipped)
  rept <- which(uns %in% val)
  # replicate patient for each occurrence
  x <- x[rep(seq_len(nrow(x)), rept),]
  # add vector of intervals
  x$ITvec <- uns[1:rept]
  # ensure that event is only indicated for the final
  # replicate corresponding to the event interval.
  if(1 %in% x$Stat){
    x$Stat <- 0
    x$Stat[rept] <- 1
  }
  return(x)
}
```

For test data, as the true survival time and as such the number of intervals a patient is observed is considered to be unknown, each test patient is replicated for the total number of observable intervals (to obtain predictions of the conditional death probabilities for each interval).

```r
# vector of all unique intervals
all.poss <- unique(sort(surv.it))

trans.func <- function(x){
  # number of unique intervals
  rept <- length(all.poss)
  # each subject is repeated for the intervals
  x <- x[rep(seq_len(nrow(x)), rept),]
  # a column ITvec is created to indicate the survival
  # interval for each row
  x$ITvec <- all.poss
  return(x)
}
```

# Appendix C.1 (88–110)

# Appendix C.1 (113–154)
### 3.4.2 Performance measures and variable importance

Neural networks can be trained for different choices of hyperparameters - node size and decay parameter. The purpose of doing so is training a network that generalizes to new data. A cross-validation approach, detailed in the next section, is used to identify the optimal hyper-parameters. For a straightforward neural network classification problem a performance measure such as accuracy is appropriate and easily calculated. In the current survival framework, however, rather than an observed class and predicted class, predictions for a range of time intervals are obtained. For each test subject \( i \) the neural network predicts the hazard - the conditional probability of dying - for each time interval \( l \). Using \( S(t_l) = \prod_{i=1}^{l-1} (1 - h_l) \) the survival probability is calculated. What is known for each test subject is the time of event or censoring. A modified prediction accuracy measure can be calculated by taken the predicted survival probability for the last time interval in which the subject was observed, and comparing it to the observed event or censoring status. A cutoff of 0.5 is utilized for classifying the predicted survival probability as either events (< 0.5) or non-events (≥ 0.5). Having artificially assigned an event or censoring status to the predictions, accuracy is defined as the proportion of predictions in accordance with the observed events/censoring. Correspondingly, specificity is measured by the proportion of non-events that are correctly classified, and sensitivity by the proportion of events that are correctly classified.

The most common measure of model performance in a survival context is the concordance index, which calculated the proportion of concordant pairs over the total of comparable pairs. A pair is considered concordant if the prognostic score for a subject who experiences is an event earlier is higher than for a subject who experiences an event later. The measures is discussed in detail in Section 2.2.5. The concordance index relies on the ordering of subjects by prognosis. As the neural network model is flexible, and the predicted survival curve for each subject can have a different shape, there is no natural unique ordering of the subjects. At one year subject \( i \) may have a higher survival probability than subject \( j \), but this could be reversed for a different time point. As such, the concordance index is not a natural choice for evaluating neural network model performance.

Two other prediction error measures common to survival analysis are the Brier score and the Kullback-Leibler score. These measures focus on the prediction of survival beyond some fixed time point \( t_0 \). Then \( S(t_0|x) \) is the prediction of the model for an individual to survive beyond \( t_0 \) given the predictor or set of predictors \( x \). Let \( y = 1\{t > t_0\} \) be the actual observation: 1 if \( t > t_0 \), 0 otherwise. Temporarily
ignoring the censoring, the Brier score is calculated as follows for one individual for a given fixed time $t_0$:

$$\text{Brier}(y, \hat{S}(t_0|x)) = (y - \hat{S}(t_0|x))^2. \quad (3.15)$$

In a perfect prediction situation the predicted survival probability for subjects with an event will be 0, and for subjects without an event it will be 1, and as such the estimated Brier score 0. Should random guessing be employed, survival probabilities of 0.5 would be assigned. Then, $\text{Brier}(y, S(t_0|x)) = (1 - 0.5)^2 = 0.25$ for no event and equivalently $\text{Brier}(y, S(t_0|x)) = (0 - 0.5)^2 = 0.25$ for an event.

The Kullback-Leibler (KL) score (3.16) is based on the principle that the performance of a predictive model can be measured by evaluating the log likelihood of the model at the observations. The choice between the KL score and the Brier score is considered to be a matter of taste.[24]

$$\text{KL}(y, \hat{S}(t_0|x)) = -[y \ln(\hat{S}(t_0|x)) + (1 - y) \ln(1 - \hat{S}(t_0|x))] \quad (3.16)$$

A possible disadvantage of the KL score is that for predicted probabilities that are 0, when $y = 1$ is observed and probabilities that are 1, when $y = 0$ is observed, the log of 0 is taken, and consequently the KL score blows up as $\ln(0) = \infty$. As shown for the Brier score, the KL score is also zero for perfect prediction. In the event of random guessing with survival probabilities of 0.5 the KL score would be 0.69.

In equations (3.15) and (3.16) censoring is ignored. Censoring can be accounted for by using inverse probability of censoring weighting (IPCW)- described by Graf.[11] Observations censored before $t_0$ are ignored, observations with event time $t \leq t_0$ weighted with $1/C(t_0|x)$, and observations with a censoring/event time $> t_0$ are weighted with $1/C(t_0|x)$. Here $\hat{C}(t|x)$ is the censoring distribution as estimated by the reverse Kaplan Meier estimator.

On the following page, the function for calculating the Brier and KL scores are defined. The function takes as argument a dataframe `surv.preds` - which has a column `surv` for survival time and a column `stat` for status, and `preds`, which is the matrix of predicted survival probabilities. The censoring weights `csurv` are obtained from the reverse Kaplan Meier estimator.
brier.function.neur <- function(surv.preds, preds) {

    # calculating censoring distribution
    so <- Surv(surv.preds$surv, surv.preds$stat)
    time <- so[,1]
    ot <- order(time)
    cens <- so[ot,2]
    time <- time[ot]
    N <- nrow(so)

    hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
    csurv <- predict(hatcdist, times = time, type = "surv")
    csurv[csurv == 0] <- Inf

    btime <- time
    survs <- t(preds)
    btime.n <- unique(time)

    bsc <- rep(0, length(btime.n))
    bsk <- rep(0, length(btime.n))

    for (j in 1:length(btime.n)) {

        # indicator vectors for selecting relevant patients
        help1 <- as.integer(time <= btime.n[j] & cens == 1)
        help2 <- as.integer(time > btime.n[j])

        # modification to avoid log of zero
        inb <- survs[j,]
        inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
        inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)

        # Brier bsc) and KL score
        bsc[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
                        (1 - survs[j,])^2 * help2 * (1/csurv[j]))
        bsk[j] <- -mean((log(1-(inb))*help1*(1/csurv[j]) +
                         log(inb)*(1/csurv[j]) *help2))
    }

    RET <- rbind(Brie, KL)
    return(RET)
}

# Appendix C.2 (7-88)
The calculation of the score for a given time point \((b\text{.time.n}[j])\) is split into two parts. From the prediction matrix \(\text{surv}\) the vector of predictions for each time interval \(j\) is taken: \(\text{surv}[j,]\). Then two situations are distinguished. Either subjects experience an event before or at that time: \(\text{time} \leq b\text{time.n}[j] \& \text{cens} == 1\), where \(\text{time}\) is the event/censoring times of the subjects, and \(\text{cens}\) is the event indicator (1 for an event). Alternatively the event/censoring time of the patient is after the time \(j\): \(\text{time} > b\text{time.n}[j]\), and as such at the time point \(b\text{time.n}[j]\) no event has been observed. In the first case, per the ICPW approach, observations are weighted with the individual censoring weights \(c\text{surv}\), in the latter case the observations are weighted with the censoring distribution in \(j\) \(c\text{surv}[j]\). The total error for a given time interval \(j\) is calculated by applying (3.15) in case of the Brier score and (3.16) in case of the Kullback-Leibler score. The error is calculated for \(y = 1\) and \(y = 0\) and subsequently averaged over.

For the Kullback-Leibler score, for predictions of 0 in \(\log(\text{surv}[j,])\) and 1 in \(\log(1-\text{surv}[j,])\) the log of 0 is taken and the value blows up to infinity. To ensure that values can be calculated, survival predictions of 1 and 0 are modified: values of 1 are lowered by 0.000001, and values of 0 increased by the same amount. If no extreme survival predictions are present the KL score remains unaffected. If 1s or 0s are predicted the KL score gives noticeably higher values that exceed 0.69. With this modification, high KL scores can be considered to be indicative of extreme predictions.

As a final and sixth measure of model performance the error value of the converged network is taken. In the training process of the network weights are estimated by minimizing an error function until convergence - here the cross-entropy error. The final converged error value is selected as a measure of prediction performance. Of note is that, unlike the accuracy measures, the Brier score, and KL score, which are informative as sole measurements, any error value cannot be judged on its own but should be considered relative to other error values obtained from other neural networks trained on the same data.

For the interpretation of the network Olden’s connection weight method is used [28]. This method relies on summing the products of the weights connecting the input and hidden layer and the weights connecting the hidden and output layer for each input variable, described in detail in Section 3.1. On the next page a function is defined which implements this method.
pretty.pictfunc <- function(model.nnet){

  # names input nodes, number input nodes
  names <- model.nnet$coefnames; no.input <- model.nnet$n[1]

  # number hidden nodes, number output nodes
  no.nodes <- model.nnet$n[2], no.outputs <- model.nnet$n[3]

  # dimensions for weight matrices
  ncols.1 <- (no.input + 1); nrows.1 <- (no.nodes)
  ncols.2 <- (no.nodes + 1); nrows.2 <- no.outputs
  length.1 <- ncols.1 * nrows.1; length.2 <- ncols.2 * nrows.2

  # selecting weights
  wts <- model.nnet$wts
  weights1 <- as.data.frame(matrix(wts[1:length.1],
                                  ncol=ncols.1, nrow=nrows.1, byrow=TRUE))
  colnames(weights1) <- c("Bias", names)
  rownames(weights1) <- paste("N", seq(1:no.nodes), sep="")
  weights2 <- t(matrix(wts[(length.1+1):(length.1+length.2)],
                        nrow=nrows.2, ncol=ncols.2, byrow=TRUE))
  rownames(weights2) <- c("Bias", paste("N", seq(1:no.nodes), sep=""))

  # calculating variable imp using connection weight method
  mega.mat <- matrix(NA, ncol=no.input, nrow=no.outputs)
  for (i in 1:no.input){
    for (j in 1:no.outputs){
      mega.mat[j,i] <- sum(weights1[,i+1] * weights2[2:(no.nodes + 1), j])
    }
  }
  colnames(mega.mat) <- names
  mega.mat.abs <- abs(mega.mat)
  totals <- rowSums(mega.mat.abs)
  mega.mat.rel <- as.data.frame(mega.mat.abs/totals)
  rels <- as.vector(as.numeric(mega.mat.rel))

  rainbowcols <- rainbow(length(names), s = 0.5)
  barplot(rels, col=rainbowcols, names.arg=names, ylim=c(0,0.4), ylab="Relative_importance")
}

# Appendix C.2 (144–198)
3.4.3 Cross-validation and evaluation of performance measures

In this section a cross-validation is performed to identify the optimal hyperparameters - number of nodes and decay parameter - and the different measures of performance are evaluated. First, the approach is described, then the results.

3.4.3.1 Cross-validation approach

For the cross-validation the data is divided in a validation ($\frac{1}{3}$) and a training set ($\frac{2}{3}$), while maintaining the event/censoring ratio in the two sets. The cross-validation is performed on the training set, and the performance of the final model is evaluated on the validation set. The EURAMOS-1 trial and data are described extensively in Chapter 1, and to a lesser extent in Chapter 2, where a dynamic prediction model is fitted. The data used here deviates in one aspect - as information on intermediate events local recurrence and new metastatic disease are not relevant to this chapter (or the subsequent ones) patients who experienced an intermediate event at or before surgery were not removed from analysis (Section 1.1.2), giving a total of 2025 patients eligible for analysis.

```r
one.set <- which(pit$stat==1); zero.set <- which(pit$stat==0)
set.seed(987)
ind.ones <- sample(one.set, length(one.set)/3, replace=FALSE)
set.seed(987)
ind.zeros <- sample(zero.set, length(zero.set)/3, replace=FALSE)
validation.set.ind <- c(ind.ones, ind.zeros)
```

A five-fold cross-validation was performed on the training set, for a number of hidden nodes ranging from 1 to 10 and weight decay parameters 0, 0.001, 0.01, 0.05, and 0.1. A seed was set to ensure that for each cross-validation the same subjects were selected for each fold. In the cross-validation four folds are used to train the network, and comprise the training data, while the fifth fold - the test data - is used to assess the network performance. This is repeated until every fold has acted as the test set.
The neural network is trained using the `nnet()` function from the `nnet` R package, as shown above. Here, the target vector is `stat` and the input variables are `Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec`. Note that `ITvec`, a variable indicating the survival interval, is included in addition to the regular predictor variables.

# Appendix C.3 (3−128)
size and decay are used to specify the hyperparameters. As error function the entropy loss function is chosen. The network can be further influenced by defining the maximum number of iteration maxit and restricted by defining the maximum number of weights MaxNWts. In this application, the values specified for maxit and MaxNWts are well in excess of the number of iterations and number of weights, respectively, that are used by the network. As appropriate for a PLANN neural network, a logistic activation function is used and a logistic transformation of the output. In the nnet() function this is the default choice. The predict function is used to obtain the hazard predictions for the test set. Note that the columns of the test dataframe must correspond to the names of the variables specified in the nnet() function. In the event that a dataframe, rather than individual vectors are entered in the training function, the predict() function will be unable to perform the prediction. This is detailed to a greater extent in the annotations of the cross-validation function given in Appendix C.3 (3-128). As the data, and as such the corresponding predictions, are in long format, additional steps need to be taken to isolate the subject-specific predictions. The objective is obtaining a prediction matrix (pred.matrix), containing, for each individual, a row of survival predictions for each of the month long intervals. As described in Section 3.3, a neural network of this structure predicts conditional death probabilities $h_l$ for each interval $l$. From the hazards, survival predictions are obtained using $S(t_{li}) = \prod_{l=1}^{li} (1 - h_l)$. The matrix statt contains the survival prediction corresponding to the observed survival time, the event indicator of the test data, and the observed survival time interval. The former two are necessary to calculate the accuracy, sensitivity, and specificity measures. The event indicator stat, survival time surv and the prediction matrix are necessary to calculate the Brier score and Kullback-Leibler score (see also Section 3.4.2). The two scores are calculated for 8 time points, from 1 to 8 years. The cross-entropy error value is obtained from the neural network object.
# dataframe with probabilities, status, id and surv interval
statt <- as.data.frame(cbind(rel.probs, stat, id, real.surv))
colnames(statt) <- c("pred", "stat", "id", "surv")

# rounding predicted probabilities to classify
# individuals as yes/no event
hmstat <- stat
hmstat$pred <- 1-round(hmstat$pred, 0)

# saving error values
error.val[k] <- new.n.net$value
acc[k] <- length(which(hmstat$pred==hmstat$stat))/nrow(hmstat)
sens[k] <- length(which(hmstat$pred==1 & hmstat$stat==1))/
length(which(hmstat$stat==1))
spec[k] <- length(which(hmstat$pred==0 & hmstat$stat==0))/
length(which(hmstat$stat==0))
brieKL <- brier.function.neur(statt, pred.matrix)
brier <- rbind(brier, brieKL[1,])
KL <- rbind(KL, brieKL[2,])  # Appendix C.3 (3–128)
### 3.4.3.2 Cross-validation results

Figure 3.3 shows the Kaplan Meier distribution (a) and the distribution of the observed event times (b). It can be observed that most patients survive/are censored. The majority of events occur in the first half of the study. Figures 3.4 to 3.8 show the values for the performance measures for neural networks with 1-10 nodes for weight decay values of 0, 0.001, 0.01, 0.05, 0.1. Each figure corresponds to a different decay value.

![Figure 3.3: Kaplan Meier (a) and event time distribution (b)](image)

As can be seen in figure 3.4, when regularization in the form of weight decay is absent, there is no effect observed for different node sizes. When looking at figure 3.4b, it can be noted that a specificity of 1 and a sensitivity of 0 are measured, indicating that the neural network defaults to overestimating the survival, which results in all patients being classified as alive at the 0.5 cutoff that is used for assessing sensitivity and specificity. In the data itself only 27% of the observations are events. As such, a high prediction accuracy can be achieved by assuming there are no events at all. When observing the Kullback-Leibler score in 3.4c, it can be seen that it exceeds the usual range of 0 – 0.69, where 0.69 is the score that would be observed for random guessing with 0.5 survival probabilities. The modifications made to the Kullback-Leibler score (Section 3.4.2) ensure that the score blows up when extreme survival predictions of 1 or 0 are made. In this particular situation survival has been overestimated and an overabundance of $\hat{S}(t|x) = 1$ predictions have been made, resulting in an unrealistic model. Also of note is that both scores steadily increase...
over time, and do not follow the characteristic inverted U-shape pattern. When comparing the cross-validated model performance at the five decay values, it is clear that decay values of 0 and 0.001 (Figure 3.4, Figure 3.5) do not lead to suitable values for the Brier and Kullback-Leibler scores. Performance improves at increased regularization. For a decay of 0.01 and networks with a small number of nodes (1, 2, and 3 nodes) the Kullback-Leibler scores remain below the 0.69 cut-off (Figure 3.6c). Weight decay values of 0.05 (Figure 3.7) and 0.1 (Figure 3.8) have a comparable performance. Notably, just like for 0 decay there is no differential effect observed for the number of nodes. For both decay values the Kullback-Leibler and Brier score are comparatively low and display the characteristic shape. The KL and Brier scores would suggest any network with such decay values is the most suitable. Turning to accuracy, specificity and sensitivity as performance measures, it can be observed there is little variation, both across nodes and decay values. Slightly higher values are observed for a weight decay of 0.001 and 9 and 10 nodes. The cross-entropy error value is the lowest for a decay of 0.001 and 10 nodes (Figure 3.5a), and appears to support the conclusion that would be drawn from the accuracy measures, yet both contradict the Kullback-Leibler and Brier score results.

To gain more insight into the behaviour of the performance measures, survival curves for individual patients were studied. Figure 3.9 shows the predicted survival probabilities for individual patients for a model with 1 to 10 nodes at 0.05 weight decay. For all 10 models low Kullback-Leibler and Brier scores were observed in the cross-validation (Figure 3.9c-d). Figure 3.9a shows a reference patient with the following characteristics: mean volume (203.37 cm$^3$) and age (15.02 years), good histological response, radical/wide excision, female, no lung metastases and no other metastases. Figure 3.9b shows the same patient but with poor histological response, and Figure 3.9c, a patient with both poor histological response and the presence of lung metastases. In each of these plots the survival curves as predicted by the Cox model are also shown. It can be observed that while the neural networks estimate a slightly lower survival than the Cox model for each of the three patients, the curves are comparable. As a contrast, models with 1 to 10 nodes under a less stringent decay value (0.001) are shown in Figure 3.9d, for a patient with poor histological response. Recall that for high numbers of nodes higher accuracy, sensitivity, and specificity values were found in the cross-validation (Figure 3.5b), and a lower cross-entropy value (Figure 3.5a). For this decay value only networks with few nodes (1 and 2) had Kullback-Leibler scores below the 0.69 cutoff. In Figure 3.9d it can be seen that the survival curves predicted for these two node values are comparable to the prediction from the
Figure 3.4: Cross-validated performance measures for a weight decay of 0

Figure 3.5: Cross-validated performance measures for networks with 1-10 nodes and weight decay of 0 (top) and 0.001 (bottom). a) Cross-entropy error; b) Accuracy, sensitivity and specificity; c) Modified Kullback-Leibler score; d) Brier score. Standard deviation across cross-validation folds is given by error bars.
Figure 3.6: Cross-validated performance measures for a weight decay of 0.01

Figure 3.7: Cross-validated performance measures for networks with 1-10 nodes and weight decay of 0.01 (top) and 0.05 (bottom). a) Cross-entropy error; b) Accuracy, sensitivity and specificity; c) Modified Kullback-Leibler score; d) Brier score. Standard deviation across cross-validation folds is given by error bars.
Figure 3.8: Cross-validated performance measures for networks with 1-10 nodes and a weight decay of 0.1. a) Cross-entropy error; b) Accuracy, sensitivity and specificity; c) Modified Kullback-Leibler score; d) Brier score. Standard deviation across cross-validation folds is given by error bars.
Figure 3.9: a: Predicted survival probabilities for a reference patient with a good histological response from a neural network with nodes from 1 to 10, and decay parameter 0.05. b: For a patient with a poor histological response, 1:10 nodes, 0.05 decay. c: For a patient with a poor histological response and lung metastases, 1:10 nodes, 0.05 decay. d: For a patient with a poor histological response, 1:10 nodes, 0.001 decay.

Figure 3.10: Instability of survival predictions for a neural network with 5 nodes, 0.001 decay. A) Variable importance (Olden’s connection weight method) for one converged network. B) Different survival estimates obtained from 5 runs of the same neural network. C) Variable importance for another converged network.
Cox model and to the neural network predictions at a decay of 0.05, shown in Figure 3.9b. The predictions from the remaining networks at 0.001 decay, however, diverge. Models with 4, 6, 8, 9 and 10 nodes give very high survival probability estimates, while 3, 5 and 7 node models have varying values. Of note is that two networks, which have the same decay parameter, and only a difference of a single neuron, may give very different results.

The purpose of Figure 3.10 is to demonstrate how predictions may be unstable within the same neural network. An example is given for a neural network with 5 nodes and 0.001 decay, for which excessively high Kullback-Leibler scores were observed in the cross-validation (Figure 3.5c). Repeatedly re-training the network and predicting from each of these networks results in varying survival estimates, shown in figure 3.10b. In Figures 3.10a and 3.10c the variable importance, as estimated using the connection-weights method (see Sections 3.1, 3.4.2) is shown for two such networks. The estimated effect of the variables changes noticeably. Going from Figure 3.10a to Figure 3.10c, age becomes a more important predictor, as does other/intraleSIONAL excision, while the effect of lung metastases and other metastases decreases. Notably, for both the effect of the time variable denoting the interval of prediction is quite substantial, especially when compared to the variable importance estimated from a stable network, shown in Figure 3.13c in the results section. Figure 3.11, displays, rather than a single predicted survival curve, all the survival curves of the test set patients jointly. The survival curves have been coloured according to the histological response of the patients, with blue curves representing predictions for patients with a good histological response, red curves representing predictions for patients with a poor histological response. The predictor histological response was chosen as it was observed to have a strong effect on survival (Figure 3.9, Figure 3.13). In Figure 3.11a the survival curves are shown for a stable network with 5 nodes and decay parameter 0.05, in Figure 3.11b for an unstable network with 5 nodes and decay parameter 0.001, and in Figure 3.11c for the Cox model. The stable network (Figure 3.11a) shows a separation of patients comparable to the Cox model (Figure 3.11c). For the unstable network (Figure 3.11b) various aberrant curves can be observed.
Figures 3.9, 3.10 and 3.11 show that neural networks can at times be unstable, and that of all the evaluated performance measures - cross-entropy error value, accuracy, sensitivity, specificity, Brier score, Kullback-Leibler score - the Kullback-Leibler score is most indicative of such instability being present in the network. The Brier score gives results in accordance with the Kullback-Leibler score, but displays more subtle differences. Accuracy, sensitivity and specificity vary comparatively little over the networks. The highest values are perhaps observed for the networks with a greater number of nodes at 0.001 decay, yet such networks have been shown to be unstable. Were the cross-entropy value used as an indicator the same unsuitable selection would be made.
3.4.4 Results

Good performance measures were observed for all networks with decay 0.05 and 0.01. As a final model an intermediate choice was made for a network with 5 hidden nodes and 0.05 decay. A schematic representation of the model is shown in Figure 3.12. Blue and red lines indicate positive and negative weights, respectively. Thickness of the connecting lines is proportional to weight magnitude. A total of 56 weights are estimated for the network, with 40 weights for 9 input nodes × 5 hidden nodes, 5 weights for 1 bias node × 5 hidden nodes, 5 weights for 5 hidden nodes × 1 output node, and 1 weight for the final bias node.

Figure 3.12: Schematic representation of the final neural network: 5 hidden nodes, weight decay 0.05
Accuracy, sensitivity and specificity for the final neural network model and the Cox model are shown in Table 3.9 and are almost identical.

Table 3.9: Performance measures for a 5 node neural network with 0.05 decay and the Cox proportional hazards model

<table>
<thead>
<tr>
<th></th>
<th>Neural network (5 nodes)</th>
<th>Cox model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error value</td>
<td>357.61</td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.92</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Figure 3.13a shows model performance in terms of the Brier and Kullback-Leibler scores. Until approximately a time of 4-5 years the neural network and Cox model are close in performance, with slightly higher values for the neural network. At approximately 4 years the Kullback-Leibler score decreases markedly, as does the Brier score to a lesser extent at approximately 5 years. In the previous section the distribution of event times was shown (Figure 3.3b) and it was noted that the majority of the events occur in the first half of the study—before the 4th/5th year. It is of note that the scores decrease around this mark. Figure 3.13b shows individual predictions for a reference patient (green), a poor hist patient (orange), and a poor hist + lung mets patient (purple). Table 3.10 gives the exact patient characteristics for these patients. The neural network predictions (dotted line) are consistently lower than the Cox model predictions (solid line).

Table 3.10: Patient characteristics for reference patient 'poor hist' patient and 'poor hist + lung mets' patient

<table>
<thead>
<tr>
<th></th>
<th>Hist</th>
<th>Lung mets</th>
<th>Other mets</th>
<th>Vol</th>
<th>Age</th>
<th>Excision</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Good</td>
<td>No</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>Wide/radical</td>
<td>Female</td>
</tr>
<tr>
<td>Poor hist</td>
<td>Poor</td>
<td>No</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>Wide/radical</td>
<td>Female</td>
</tr>
<tr>
<td>Poor hist + lung mets</td>
<td>Poor</td>
<td>Yes/possible</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>wide/radical</td>
<td>Female</td>
</tr>
</tbody>
</table>
Figure 3.13: Comparison of Brier scores, Kullback-Leibler scores and individualized predictions for a neural network with 5 nodes and 0.05 decay and the Cox model. a) Brier (solid line) and Kullback-Leibler scores (dotted line) for neural network (red) and Cox model (black). b) Individual predictions for a reference patient (green), a patient with a poor histological response (orange), and a patient with poor histological response and lung metastases (purple), for the neural network (dotted line) and the Cox model (solid line). c) Neural network variable importance as measured by Olden’s connection weight method[28].

Figure 3.13c shows the variable importance calculated using Olden’s connection weight approach.[28] Poor histological response, the presence of lung metastases and the presence of other metastases are identified as the most important predictors. In Table 3.11 the hazard ratios as estimated from the Cox model are shown. Here too, poor histological response (HR: 2.509, 95%CI: 2.02-3.12), the presence of lung metastases (HR:2.621, 95%CI: 2.09-3.29), and the presence of other metastases (1.987, 95%CI: 1.33-2.97) are identified as the most important predictors.

Previously, the choice of a neural network with 5 nodes with 0.05 decay was motivated as a safe, intermediate choice. In the cross-validation, however, no difference was observed between networks with 1 to 10 nodes. This suggests that even a network with only one hidden node would be sufficient for modelling the non-linearities.
Table 3.11: Cox proportional hazards model: Hazard ratios

<table>
<thead>
<tr>
<th></th>
<th>Hazard ratio</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hist response: Good</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hist response: Poor</td>
<td>2.509</td>
<td>2.02-3.12</td>
</tr>
<tr>
<td>Age</td>
<td>1.004</td>
<td>0.98-1.02</td>
</tr>
<tr>
<td>Volume</td>
<td>1.000</td>
<td>1.00-1.00</td>
</tr>
<tr>
<td>Excision: Wide/radical</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Excision: Marginal</td>
<td>0.820</td>
<td>0.60-1.12</td>
</tr>
<tr>
<td>Excision: Other</td>
<td>1.269</td>
<td>0.88-1.83</td>
</tr>
<tr>
<td>Sex: Female</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Sex: Male</td>
<td>1.230</td>
<td>0.99-1.53</td>
</tr>
<tr>
<td>Lung mets: No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Lung mets: Yes/Possible</td>
<td>2.621</td>
<td>2.09-3.29</td>
</tr>
<tr>
<td>Other mets: No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Other mets: Yes/Possible</td>
<td>1.987</td>
<td>1.33-2.97</td>
</tr>
</tbody>
</table>

present. A neural network uses nodes to allow for non-linear relationships- a feed forward neural network without hidden nodes can only find a linear decision boundary. The more hidden nodes, or indeed hidden layers with hidden nodes are included, the more complex a decision boundary can be established. Here, such complexity does not appear to be necessary. A way of assessing non-linearity in the Cox model is by considering the relationship of Schoenfeld residuals with the transformed time, implemented in R-software in the `Cox.zph()` function. Using this measure, the proportional hazard assumption is found to be violated for predictors histological response ($\rho = -0.164$, $p=0.002$), other/intralesional excision ($\rho=0.112$, $p=0.030$), and the presence of lung metastases ($\rho=-0.154$, $p=0.003$). In Chapter 2, where a dynamic prediction model was fitted to the EURAMOS osteosarcoma data, non-linearity of the data was assessed by including linear and quadratic time interactions. These only proved to be significant for the prognostic factor histological response. While differently shaped time dependencies may be present, all this suggests that the non-linearity in the data is not severe.

In Figure 3.14 a final comparison of the neural network versus the Cox model is made. Predictions for all test set patients are shown. The survival curves are coloured based on histological response (Figures 3.14a, 3.14b) and based on the presence of lung metastases (Figures 3.14c, 3.14d) - predictors identified as strong by both the neural network (Figure 3.13c) and the Cox model (Table 3.11). Comparing the predictions from the neural network (Figures 3.14a, 3.14c) to those from the Cox model (Figures 3.14b, 3.14d) it can be observed that while the predicted curves are smoother,
general pattern is similar, both in the overall shape of the curves, as in the distribution of good (blue) versus poor (red) histological response (Figures 3.14a and 3.14b) and in the distribution of the absence (blue) and presence (red) of lung metastases (Figures 3.14c and 3.14d)

Figure 3.14: Predicted survival curves of all test patients for a neural network with 5 nodes, decay 0.05 (a,c) and the Cox proportional hazards model (b,d). In a) and b) patients are coloured according to histological response, with a good histological response in blue and a poor one in red. In c) and d) patients are coloured according to the presence of lung metastases, with blue for no lung metastases and red for yes/possible.

3.4.5 Discussion

In the previously described application Biganzoli’s PLANN approach [10] was used to construct a neural network. The final network consist of 9 input nodes, 7 for the predictors, 1 for the time interval, and 1 bias node which functions as an intercept. For the hidden layer, 5 nodes were chosen to connect the 9 input nodes to a single output node. Six different performance measures were considered - accuracy, sensitivity, specificity, the converged cross-entropy error value, and the Brier and Kullback-Leibler score over a range of time points (1 to 8 years). Accuracy, sensitivity and specificity showed very limited variation over different decay values and node
sizes and when used as a selection criterion gave rise to unstable neural networks. The lack of stability was apparent in vastly different predictions when the node size were varied under a fixed decay parameter, and also in different predictions when the same network was trained multiple times (i.e. retrained with the same node size and decay parameter). The Kullback-Leibler (KL) and Brier score proved superior as selection criteria. The KL score used was modified slightly to ensure values could be calculate for extreme survival probability predictions of 0 and 1. Values for such predictions are large and well out of the 0.69 range. As such, large KL scores especially are indicative of an unsuitable network that makes unrealistic predictions. Given the observations made in the cross-validation, a very strong recommendation would be made to carefully evaluate any performance measure, preferably use several, and to include the KL and Brier scores in that selection. For a better insight into prediction performance it would be advisable to examine a network more closely by looking at individual survival curves, for specific patients (as done in 3.10b) or for all patients jointly (as done in 3.11). While the true survival probabilities of a patient are unknown, unrealistic features and shapes of the survival curve, such as excessively steep or flat predictions, may still serve as a warning sign. For instance, in 3.11b, where predictions for an unstable network are shown, very steep curves can be observed. Another warning sign of an unsuitable network would constitute a lack of variation in predictions under different values of a prognostic factor that is strongly suspected to be relevant. Also demonstrated in Figures 3.10a/b is that variable contribution, as measured by the connection-weight method, varies markedly for unstable networks.

When comparing the performance of the neural network and Cox model as applied to the EURAMOS osteosarcoma data, no great difference is observed. Perhaps a marginal superiority of the neural network is suggested by the lower Brier and Kullback-Leibler scores after the 5th and 4th year mark, respectively. In the results section (Section 3.4.4) it was noted that most events occur in the first half of the study - before 4-5 years -which may be a relevant observation. It would be of interest to investigate the comparative performance of a neural network under differently shaped censoring distributions and also under different censoring percentages.

When discussing the performance measures in Section 3.4.2 it was remarked that the C-index does not lend itself to straightforward use for the evaluation of neural networks, as there is no natural ordering of subjects by prognosis. Gensheimer & Narsimhan used a cutoff survival time to rank the subjects. An idea would be calculate the C-index in such a manner at different cutoff times, and use such a C-index as a time-dependent performance measure.
One of the challenges a neural network poses is in its interpretation. As shown in the application, the connection-weight approach allows for insight into the overall contribution of a prognostic factor. An conceptually simple alternative for evaluating the variable contribution would be by fitting multiple networks that each omit a different variable, and quantifying the decrease in network performance. This however ties in to the previously discussed challenge of establishing a suitable performance measure. While the Brier and Kullback-Leibler scores appear to be reliable, they are calculated for a range of time points, and as such a change in model performance is not easily quantified. One possibility is selecting a single time point - for instance the median follow-up time. Alternatively, a global error could be measured. Van Houwelingen & Putter showed that a global prediction error, either for the Brier or Kullback-Leibler score, is best obtained by integrating the dynamic prediction errors over time. Regardless of exact approach, the network described by Biganzoli and applied in this chapter only specifies a single output node, and as such only an overall measure of variable importance can be obtained. In section 3.2 a range of networks is described which define \( K \) outputs for \( K \) time intervals. A network with this structure would allow for the quantification of variable contribution for each individual output node. As the multiple nodes represent subsequent time intervals, this would allow insight into the variable effect over time, and also allow insight into potential instability in specific time intervals. Of particular interest in this context would be Liestol's approach (Section 3.2.2), which is very close to Biganzoli's approach, similarly relying on logistic activation and cross-entropy error minimisation. However, rather than longitudinally transforming the data, Liestol defines separate target vectors for each individual, which vary in length depending on how long an individual was observed. This method requires a small modification of the error function, and unlike Biganzoli's PLANN approach does not enjoy a relatively straightforward standard software implementation. As Biganzoli's and Liestol's methods estimate the same quantity and are very close in approach, they are eminently suitable for examining the added benefit of defining multiple output nodes versus a single output node.

Another relevant avenue of investigation would be into the comparative performance of methods that estimate different quantities. For instance, it would be of interest to see if an approach such as Biganzoli's PLANN method, which estimates the conditional failure probabilities, is superior or inferior to an approach that estimates survival probabilities directly, such as Streets model. Overall, when evaluating the neural network performance on the EURAMOS osteosarcoma data it appears to perform slightly better than the Cox model for later
time points. Worth noting, however, is that the Cox model uses exact time points, while the neural network relies on the grouping of survival times in intervals—here 1 month long intervals. Such a partitioning in intervals may also give the illusion of better performance of the neural network. It should also be considered that the Cox model fitted here is a simple one. In applications where Cox is used as the primary model, the model would be expanded with variable-time interactions and possibly also variable-variable interactions, which is likely to improve the model performance. Finally, it should be considered that even though a neural network may give an increase in predictive ability, this is achieved at the cost of interpretability. It should also be noted that a neural network requires greater effort in implementation, and that great care should be taken to avoid training an over-fitted and unstable network.
Chapter 4

A Random survival forest approach

In this chapter a random survival forest application to the EURAMOS osteosarcoma data is described. Preceding the application are a number of sections detailing the theory of decision trees and random forests, the extension to survival data, and the available implementation in R software. The usual analyses for survival rely on parametric or semi-parametric methods, for which non-linear and interaction effects need to be explicitly modelled. Classification and regression trees (CART) have proved a valuable resource in modelling complex and non-linear effects for categorical and continuous outcomes, which has motivated a range of attempts to adapt tree methodology to right-censored survival outcomes. In this chapter Ishwaran’s proposed method [19] is implemented for the analysis of the EURAMOS osteosarcoma data. The theoretical approach as well as the practical details of Ishwaran’s randomForestSRC R package will be discussed in Sections 4.2 and 4.3, respectively. The remaining sections follow the structure used in Chapter 3, and will detail the application in Section 4.4, where the relevant performance measures and variable importance are discussed (4.4.1), the cross-validation approach for hyperparameter tuning (4.4.2), the results (4.4.3), followed by a discussion of the advantages and potential pitfalls of the method (4.4.4).

4.1 Binary decision trees and forests

Any tree method relies on the recursive binary partitioning of a given features space into increasingly smaller regions, containing observations with similar responses, until a certain stopping criterion is reached. The regions are referred to as nodes, and the final regions created on termination of the growth of a tree are known as terminal nodes, or leafs, while the initial node is commonly referred to as the root node. There are two necessary features to any tree algorithm: a node splitting rule, which
informs how a partition is best split and a stopping rule, which provides a criterion for termination the growth of a tree. For a given feature space with \( x \) possible variables, and \( c \) possible split values, variable \( x^* \) and split point \( c^* \) are chosen in a way that maximizes the difference between the two daughter nodes. The objective is to enact splitting in such a way that the observations in each node will be increasingly homogeneous. That is to say, when splitting a given node, the observations within each daughter node should be more similar to each other than the other daughter node. The difference in impurity in the parent node and the average daughter node impurity is maximized. In classification trees the impurity is quantified using either the Gini index or the Shannon Entropy measure. In regression trees, the mean squared error is used to define the split rules.

While a tree has great potential for modelling non-linear and interactive effects, as will be detailed later, the very way a tree is grown makes it highly variable and very unstable. Each split of a node is dependent on previous partitioning. Consider a situation in which several samples are taken from a larger dataset and used to grow individual trees. Even a slight variation in the data may result in a selection of a different variable \( x^* \) or splitting point \( c^* \) for one of the early nodes, or even the root node, giving rise to vastly different trees. The sensitivity of a single tree to minor training data variations is likely to result in poor generalization to new data. A way to combat the overfitting reduce the variability - is by introducing a measure of randomness in the way of bootstrapping. In this procedure individual trees are grown for multiple bootstrap samples drawn from the data, and subsequently aggregated over, producing a single ensemble tree. This procedure of bootstrap aggregation, is commonly known as bagging.

While in lone decision trees and in the ensemble bagging approach all variables are considered as candidates for a split, in the random forest approach a random sub-selection of variables is made. This can be especially beneficial in a datasets with a small number of strong predictors and a larger set of weaker ones. For a given split a predictor will be selected that maximally reduces impurity in that specific node. This occurs in an isolated process with no regard for the nodes yet to come. In the event a very strong predictor is present in the dataset, this predictor will be chosen. By defining a random subset of candidate variables, this allows for the selection of less strong predictors as splitting variable, which could result in the inclusion of relevant interaction effects that would be missed in standard bagging procedure. As such,
while the best split found in the variable subset may be suboptimal on a local level, it may increase the overall performance of the ensemble. The introduction of such a random selection step ensures greater diversity, and as such lower correlation between the individual trees of the forest. Breiman showed that a low correlation resulted in a lower upper bound for the generalization error of a tree ensemble. As such, the random forest approach can be argued to be an improvement on straightforward bagging, by 1) introducing weaker variables with potentially relevant interaction effects and by 2) decreasing the overall generalization error by correlation reduction.

A random forests of binary trees is an entirely non-parametric method capable of modelling complex non-linear and even non-monotonous associations, and does not require such a relationship to be specified in advance. A linear regression, for example, draws information from a linear combination of variables, and as such is quite rigid in the associations it can model. Various nonlinear regression approaches, such as the use of smoothing splines, require prior specification of the function shape and the position of knots. In a comparable way, the Cox proportional hazards model assumes proportionality over time, and deviations are modelled using explicitly defined functions of time. For a tree, on the other hand, which is grown recursively, information is obtained from the great variety of combinations of the rectangular partitions in a data-driven process.

Random forests and also, perhaps to a lesser extent, bagged forests are capable of modelling complex associations, even in the situation of a high number of predictors $p$, and have been successfully applied in the context of genetic and epidemiological data. Unlike single trees, forests are a both stable and powerful tool for data analysis. A single tree, however, can easily be interpreted, but an averaged ensemble decidedly less so. As such, while forests resolve variability, this comes at the cost of interpretability. Various measures have been proposed to quantify variable contribution, with the most common method relying on in-variable permutation or random node assignment. The various approaches are discussed in more detail in section 4.4.1. Note, however, while contribution can be quantified, the exact relationships - the interactive and/or non-linear effect - cannot be explicitly defined.

### 4.2 A survival approach to random forests

For random forest, as for neural networks, detailed in the previous chapter, the main difficulty in analysing time-to-event data springs from the uncertainty introduced by
censored observations. In 1984 Breiman formulated a survival tree in terms of a classification tree, as such reducing a problem to one that fits within the Classification and Regression Tree (CART) methodology. Breiman later described an implementation for censored data, but reportedly it was theoretically poorly documented. Unlike classification trees or regression trees, for which the node impurity can be easily assessed by measures such as the Gini index or Shannon entropy and least squares, respectively, there is no straightforward measure of impurity for survival data. Two main approaches for dealing with the censoring problem have been proposed. In 2006 Hothorn described an ensemble survival tree approach relying on weighting observations with inverse probability censoring (IPC) weights, followed by a random forest regression with log-transformed time as the outcome. Bootstrap samples are drawn with probabilities corresponding to the IPC weights derived from the Kaplan Meier estimator, with weights of 0 for censored observations. The weighted average of the observed log survival times is used to calculate the quadratic loss for each partition.

An alternative approach was proposed by Ishwaran in 2008, termed random survival forest (RSF). This approach is unique in that it follows Breimans prescription, which specifies that in all aspects of growing a forest the outcome must be taken into account. In terms of survival data and the censoring problem, this means that both the survival time and the censoring status should be explicitly involved in growing the forest, and as such the optimal partitioning should be found by evaluating the survival difference to measure the effectiveness of a split. The algorithm can be described summarily in 5 steps. Note that the formal notation following the algorithm description is taken directly from Ishwaran.

1) B bootstrap samples are drawn from the data, each sample comprising on average 63% of the observations. 37% is left as out-of-bag (OOB) data.

2) A survival tree is grown for each bootstrap sample. For each node split a subset of variables is randomly selected as candidate. The effectiveness of a split variable and split point is evaluated by comparing the survival curves of the two groups using a) a logrank splitting rule, b) a logrank score splitting rule, or c) a random log-rank splitting rule.

3) The tree is grown under the constraint that the terminal nodes have no less than 1 unique events.
4) The cumulative hazard function (CHF) is calculated for each tree. The individual CHFs are then averaged to obtain the ensemble CHF.

5) Using OOB data, the prediction error is calculated for the ensemble CHF. The prediction error is the C-index.

1) In the first step B bootstrap samples are drawn from the data. (63% from the data). The remaining data is referred to as out-of-bag (OOB) data. As detailed previously, bootstrapping the data introduces randomisation, which decreases the generalization error of an ensemble tree. Additionally, the OOB data can be used to obtain a prediction error measure for the ensemble cumulative hazard (CHF).

2. A survival tree is grown for each bootstrap sample. The splitting variable and split point are established by searching over a subset of variables \( x \), and over all possible split points \( c \). The variable \( x^* \) and split point \( c^* \) are selected such that the survival difference is maximized. For a continuous variable, the split is straightforward and will take the form of \( x \leq c \) and \( x > c \). A subject \( i \) with a value \( x_i \) will be allocated to the left and right daughter node when \( x_i \leq x \) and \( x_i > c \), respectively. In a deterministic splitting approach all possible split points on \( x \) are considered, which are all the unique values observed for \( x \). In a random splitting approach a limited number of potential split points - prespecified - are randomly selected. In the event of a splitting variable that is discrete rather than continuous, all possible groupings of the factor levels are considered. For a discrete variable \( x \) with 4 levels there are two possible level devisions. In the first case, one level would represent the left daughter node, and three levels the second. In the second case, both nodes would be allocated two levels. This means there are \( C(4,1) \) and \( C(4,2) \) split points for each grouping, and 8 split points in total. With an increasing number of levels this number quickly blows up.

The survival difference can be quantified using the log-rank statistic or the log-rank score statistic. The log-rank test has been shown to be a valid test for splitting survival trees given both proportional and non-proportional hazards.\[4\] For a split on a continuous predictor defined as \( x \leq c \) and \( x > c \) let \( t_1 < t_2 < ... < t_m \) be the event times in the parent node \( h \). Let \( d_{k,l} \) and \( Y_{k,l} \) be the number of events and subjects at risk at time \( k \) in the left daughter node, respectively, and let \( d_{k,r} \) and \( Y_{k,r} \)
be the same for the right daughter node. Then \( Y_{k,l} \) will be all the subjects \( i \) at risk at time \( k \) with covariate value \( x_i \leq c \), and \( Y_{k,r} \) the same for subjects with \( x_i > c \) (4.1).

\[
Y_{k,l} = \# T_i \geq t_k, \quad x_i \leq c, \quad Y_{k,r} = \# T_i \geq t_k, \quad x_i > c
\]  

Let \( Y_k \) and \( d_k \) be the total number of subjects at risk at time \( k \) and the total number of events at time \( k \), respectively. Then, the log-rank statistic for the split point \( c \) on the variable \( x \) is given by \([17]\)

\[
L(c, x) = \frac{\sum_{k=1}^{m} \left( d_{k,l} - \frac{Y_{k,l}}{Y_k} \right) d_k}{\sqrt{\sum_{k=1}^{m} \frac{Y_{k,l}}{Y_k} \left( 1 - \frac{Y_{k,l}}{Y_k} \right) \left( \frac{Y_{k,l}}{Y_k} - d_k \right) d_k}}.
\]  

(4.2)

The larger the value of \( |L(c, x)| \), the greater the difference between the survival curves and the greater the node separation. The objective is to find a best variable \( x^* \) with an optimal split point \( c^* \) such that \( |L(c^*, x^*)| \geq |L(c, x)| \) for all variables \( x \) and splitpoints \( c \).

An alternative splitting rule is given by the log-rank score statistic, which differs from the log-rank statistic in the assumption that the variable \( x \) is ordered. This approach was first described by Hothorn and Lausen in 2003.\([52]\). Ishwaran provides a concise formal notation in line with the one given here for the log-rank statistic.\([17]\)

Additionally, Ishwaran describes a log-rank based randomized approach for splitting, where a single random split point \( c \) is selected for each variable \( x \), the log-rank statistic computed, and the variable and split point with the largest statistic selected for splitting. The particular splitting rule used, the number of variables considered at each split along with the number of split points must all be considered when growing a tree.

3) The tree is grown under the minimal constraint that the terminal nodes have no less than 1 unique events. The growth of a tree is terminated when this constraint can no longer be satisfied, or when a node is found to be no longer impure. Additional constraints may be introduced by increasing this threshold - increasing the terminal node size - and by limiting the depth of the tree (the number of nodes counted from the first node to the terminal node). Under these constraints a tree will only continue growing if the current \textit{nodedepth} is less than the maximum \textit{nodedepth} specified, if the current node size is at least twice the specified terminal - \textit{nodesize}, and if the current node is impure. The node size and depth of a tree are hyperparameters that are tuned for optimal tree performance. Further detail is given in section 4.3.
4. The cumulative hazard function (CHF) is calculated for each tree and averaged over to obtain ensemble CHF. The CHF is calculated in the terminal nodes. Denote a single terminal node \( h \) contained in all terminal nodes of a given tree \( h \in T \). The survival time and censoring status for individuals \( i = 1, \ldots, n \) can then be written as \( (T_{1,h}, \delta_{1,h}), \ldots, (T_{n(h),h}, \delta_{n(h),h}) \), where \( \delta_{i,h} = 0 \) denotes a subject right-censored at time \( T_{i,h} \), while \( \delta_{i,h} = 1 \) denotes an event. Define the \( n(h) \) distinct event times as \( t_{1,h} < t_{2,h} < \ldots < t_{n(h),h} \). Then, at time \( t_{l,h} \) \( d_{l,h} \) is the number of deaths, and \( Y_{l,h} \) the subjects at risk. The CHF estimate \( \hat{H}_h(t) \) for a terminal node \( h \) is then given by the Nelson-Aalen estimator

\[
\hat{H}_h(t) = \sum_{t_{l,h} \leq t} \frac{d_{l,h}}{Y_{l,h}}.
\]

(4.3)

To obtain the CHF for a subject \( i \) with predictor vector \( x_i \), the predictor vector \( x_i \) is dropped down the tree. \( x_i \) will fall into a single unique terminal node \( h \), and the CHF of the terminal node \( h \) is then taken as the CHF for the subject \( i \) with covariate vector \( x_i \): \( H(t|x_i) = \hat{H}_h(t) \).

To obtain the ensemble CHF the average over the survival trees is taken. Defining the CHF for a tree grown from a bootstrap sample \( b \) as \( H^*_b(t|x_i) \), the bootstrap ensemble \( H^*_e(t|x_i) \) is then given by

\[
H^*_e(t|x_i) = \frac{1}{B} \sum_{b=1}^{B} H^*_b(t|x_i),
\]

(4.4)

where \( B \) is the number of survival trees. Here (4.4) for a subject \( i \) with covariate vector \( x_i \) all the survival trees are used. An alternative is using only the survival trees for which \( i \) is OOB. This approach gives a CHF estimate which, given a large enough number of trees, is comparable to one that would be obtained with a leave-one-out cross-validation. Such a CHF estimate for a subject \( i \) is valid, as it has been obtained from prediction using only those trees in which \( i \) was not included as an observation.\[15\] \( H^*_e(t|x_i) \) is obtained by once more simply taken the average over the relevant trees. The indicator \( I_{i,b} \) is used to select the correct trees, and will equal 1 if the observations is OOB and 0 if it is in-bag.

\[
H^*_e(t|x_i) = \frac{\sum_{b=1}^{B} I_{i,b} H^*_b(t|x_i)}{\sum_{b=1}^{B} I_{i,b}}.
\]

(4.5)

5) In the final step of the algorithm OOB data is used to calculate the prediction error for the ensemble CHF. The concordance index for the Cox proportional hazards
model is obtained using the prognostic index and observed survival times and status (see also Section 2.2.5). In a random survival forest context, the prognostic index is replaced by a predicted outcome: the ensemble mortality. Note that while OOB data can be used to obtain the OOB ensemble mortality and by extension the OOB prediction error, a valid prediction error can also be obtained in a cross-validation procedure. When choosing the latter option, the ensemble mortality scores for the subjects of each test set are obtained by dropping test observations down the trees of a survival random forest grown from the training set, and calculating their ensemble CHF (4.4) and from the ensemble CHF the ensemble mortality.

Ensemble mortality is based on the conservation of events principle. The conservation of events for a given terminal node \( h \in T \) in a given tree can be written as the following lemma (4.6), which shows that the total number of deaths is conserved within \( h \). Summing the CHF \( \hat{H}_h(T_{i,h}) \) over the observed survival times will give the total number of deaths in terminal node \( h \).

\[
\sum_{i=1}^{n(h)} \hat{H}_h(T_{i,h}) = \sum_{i=1}^{n(h)} \delta_{i,h} \quad \text{for each terminal node} \quad h \in T \quad (4.6)
\]

The conservation of events in the terminal nodes - within the ends of a tree - extends to the conservation of events in the entire tree. Summing the estimated CHF over all observed survival times over all terminal nodes \( h \) amounts to the total number of deaths (4.7).

\[
\sum_{i=1}^{n} H(T_{i}|x_{i}) = \sum_{h \in T} \sum_{i=1}^{n(h)} \hat{H}_h(T_{i,h}) = \sum_{h \in T} \sum_{i=1}^{n(h)} \delta_{i,h} = \sum_{i=1}^{n} \delta_{i} \quad (4.7)
\]

Then, mortality for a given individual \( i \) is defined as the expected value of the CHF, summed over \( T_{i} \), conditioned on \( x_{i} \) (4.8), which is the number of deaths that would be expected under a null hypothesis of similar survival behaviour:

\[
M_{i} = \mathbb{E}_{i} \sum_{j=1}^{n} H(T_{j}|x_{i}). \quad (4.8)
\]

In a survival tree this null hypothesis is naturally enforced in the terminal nodes, as the growth of a tree ensures that all individuals in a terminal have the same CHF. Then, with (4.4) the ensemble mortality can be defined as

\[
\hat{M}_{e,i} = \sum_{j=1}^{n} H_{e}^*(T_{j}|x_{i}), \quad (4.9)
\]
and with (4.5) the OOB ensemble mortality can be defined as

$$\hat{M}^{**}_{e,i} = \sum_{j=1}^{n} H^{**}_{e}(T_j|x_i).$$

(4.10)

The concordance index is calculated by comparing the observed survival times and ensemble mortalities. Let $t^o_1, ..., t^o_m$ be the unique time-points of interest. Then, a subject $i$ is said to have a worse predicted outcome than $j$ when subject $i$ has a higher ensemble mortality than $j$. When calculating the OOB prediction error, this is given by

$$\sum_{l=1}^{m} H^{**}_{e}(t^o_l|x_i) > H^{**}_{e}(t^o_l|x_j).$$

(4.11)

For each pair of subjects where at least one subjects has an observed event time, the order of predicted outcomes is compared to the order of observed survival times. In the above example, where $i$ has a higher ensemble mortality and as such a worse predicted outcome than $j$, the pair of $i$ and $j$ would be concordant if the observed survival time of $i$ is lower than $j$. If this would not hold, the pair would be considered discordant. The number of concordant pairs is counted and the proportion of concordant pairs in the total number of evaluable pairs is calculated, given the concordance index $C$, denoted $C^{**}$ in the event of an OOB estimated. The, the prediction error is given by $1-C^{**}$. The principles behind the concordance index are explained in greater detail in chapter 2.2.5.

As noted earlier, an estimate of the concordance index $C$ can also be obtained by replacing the OOB procedure detailed above with a cross-validation. A prediction error obtained via leave-one-out cross-validation approximates the OOB error very closely. Coarser k-fold cross-validations (e.g. $k = 5$ or $k = 10$) deviate more but can still be considered a reasonably approximation, and are by necessity used for methods that do not have OOB data available. OOB data, and by extent the OOB error, is a natural byproduct of the random forest approach, and, unlike cross-validation, requires a model be only fitted once and not $k$ times, making the OOB approach very desirable from a computational viewpoint. It has been argued however, that the OOB error estimate has a bias which springs from the sub-selection that is necessarily is made in the process. In the standard 0.632 bootstrap procedure, an observation will be used for training in approximately two thirds of the trees, leaving on average about one third of the trees for OOB prediction. In the context of classification
and regression forests it has been reported that the OOB overestimates the true
generalization error \[50, 7\]. It has also been suggested that a dependency is present
between certain parameters, notably the number of variables selected for each split,
and the magnitude of the bias.\[50\] An OOB error can only be obtained from an
unweighted bootstrapping procedure (see 4.3 for a summary of different bootstrap
approaches in Ishwaran's RSF approach). Observations with large weights will almost
always be in the bootstrap sample, as such precluding a fair OOB estimation.\[19\]
In the context of Hothorn's approach, which relies on inverse probability censoring
(IPC) weighting\[52\] - briefly discussed at the beginning of this section - this makes
straightforward OOB estimation impossible. Ishwaran compared the performance of
Hothorn's IPC regression forest with his own proposed RSF method, showing that
while Hothorn's approach performed adequately when censoring was limited, RSF
was superior for data in the presence of moderate to heavy censoring.\[19\] In the
EURAMOS osteosarcoma data only approx. 30% of the events are observed, and as
such Ishwaran's approach has been chosen to model the data.

4.3 Ishwaran’s implementation in R: randomForest-SRC

Ishwaran's Random Survival Forests methodology is implemented in R, in the pack-
age randomForestSRC.\[18\] The rfsrc() function can fit a random forest to regular
classification/regression data, to survival data with a single endpoint, and survival
data with multiple competing endpoints (competing risks). The rfsrc function has
default options available, but also a wide range of customization options. The fully
annotated function is given in Appendix D.1. Table 4.1 gives an overview of a num-
er of arguments most relevant to growing a forest in the context of single endpoint
survival data, and shows what options are available for influencing specific steps of
the RSF algorithm (section 4.2). A short description of the purpose of each argument
is provided, along with the choices available in rfsrc() and, where appropriate, the
default option selected by rfsrc().

The random forest algorithm commences by drawing a number of bootstraps for
an equal number of trees. The argument ntree specifies the number of trees grown,
with a default of 1000. Common practice is to evaluate the prediction error against
the number of trees and choose a number for which the error has stabilized (see
Section 4.4.2, Figure 4.2). Typically, a larger forest will be necessary for data with
more variables, as in the event of fewer trees the random selection of variables for each split point may result in variables being underrepresented in the forest.\[5\]

Several options are available for the bootstrap. The common choice is bootstrapping at the node with the 0.632 bootstrap, which by default is with replacement (by.root). The user has the option of providing weights to observations, through the argument case wt, a vector of non-negative weights of length n for n observations. Complete manual specification of the bootstrap can be achieved through the option by.user. Here, the user must specify a matrix samp of n × ntree observations, specifying for each tree how many times each observation will occur in the bootstrap. Finally, the choice can also be made to bootstrap at each node rather than at the root. Note that in a weighted bootstrap, either by defining case weights case wt, or by specifying a manual bootstrap through by.user and samp option, and in bootstrapping at the node by.node rather than at the root, an OOB error cannot be computed.

The second step of the algorithm involves the splitting of nodes. As detailed in Section 4.1 a random forest differs from regular bagging by introducing an additional measure of variability by making a subselection of candidate splitting variables. The default option in the rfsrc() function is square root p, rounded up, for a total of p variables. The number of candidate variables is specified through mtry. Note that if p candidate splitting variables are selected the random forest reduces to bagging. On each variable, a number of split points are considered. If 0 is specified for mtry, the splitting will be deterministic and all possible split points will be considered, which can be computationally very intensive. Ishwaran recommends tuning the nsplit parameter to find the optimal number of split points, especially in the event of a mix of continuous and discrete variables, as trees are more given to splitting on continuous variables. Additionally, limiting the number of splits considered may reduce bias.\[17\] Finally, a splitting rule (splitrule) needs to be chosen, which can be logrank, logrankscore or random. The former two rely on the log-rank and log-rank-score statistic, respectively, both of which quantify the difference in survival curves between two groups in this context between the two daughter nodes for a potential split point (Section 4.2). When specifying random as the splitting rule, a single split point is randomly chosen on each variable, and the largest log-rank statistic decides the choice of split.

In the third step of algorithm a tree is grown, under the constraint that a final node may not have less than a single unique event. Additional constraints can be imposed by increasing the minimal size of the terminal nodes (nodesize) and by limiting the
Table 4.1: Parameters in \textit{rfsrc} function relevant to the construction of a random survival forest

<table>
<thead>
<tr>
<th>Algorithm step</th>
<th>Number of trees grown</th>
<th>Number of variables randomly selected as candidates for each split, minimum of 1, maximum of $p$ for $p$ variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntree</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>bootstrap</td>
<td>1 by.root</td>
<td>Bootstrap at root node (with replacement: samptype=&quot;swr&quot;, without: samptype=&quot;swor&quot;); bootstrap size (..., default - 0.65)</td>
</tr>
<tr>
<td></td>
<td>by.node</td>
<td>Bootstrap performed at the node</td>
</tr>
<tr>
<td></td>
<td>none</td>
<td>No bootstrapping</td>
</tr>
<tr>
<td></td>
<td>by.user</td>
<td>Manual bootstrapping specification, through argument 'samp'. A $n \times ntree$ array should be specified with the number of times each observation appears in each tree.</td>
</tr>
<tr>
<td></td>
<td>case.wt:</td>
<td>\textit{case.wt}: a vector of weights for each case</td>
</tr>
<tr>
<td>mtry</td>
<td>2</td>
<td>$\sqrt{p}$ 1 to $p$</td>
</tr>
<tr>
<td></td>
<td>$xvar.wt$</td>
<td>$xvar.wt$: vector of selection probabilities for $p$ variables</td>
</tr>
<tr>
<td>splitrule</td>
<td>2 logrank</td>
<td>Uses log-rank statistic for quantifying survival difference for each split.</td>
</tr>
<tr>
<td></td>
<td>logrankscore</td>
<td>Uses log-rank-score statistic</td>
</tr>
<tr>
<td></td>
<td>random</td>
<td>Enacts one random split on each variable and uses logrank statistic for quantification</td>
</tr>
<tr>
<td></td>
<td>$split.wt$</td>
<td>$split.wt$ is an optional vector of length $p$ where weights are specified for weighting the splitrule statistic. A higher weight will give a higher statistic.</td>
</tr>
<tr>
<td>nsplit</td>
<td>2</td>
<td>0 (deterministic)</td>
</tr>
<tr>
<td></td>
<td>$\geq 0$</td>
<td>number of splits considered one each candidate variable. 0 enacts deterministic splitting</td>
</tr>
<tr>
<td>nodesize</td>
<td>3 NULL</td>
<td>$\geq 1$</td>
</tr>
<tr>
<td></td>
<td>$\geq 1$</td>
<td>Ensures number observations per terminal node are on average \textit{nodesize}</td>
</tr>
<tr>
<td>nodedepth</td>
<td>3 NULL</td>
<td>$\geq 1$</td>
</tr>
<tr>
<td></td>
<td>$\geq 1$</td>
<td>Ensures the depth of a tree is no more than \textit{nodedepth}</td>
</tr>
</tbody>
</table>
total number of splits (\textit{nodedepth}). Both are regularization hyperparameters that can protect against overfitting.

\begin{verbatim}
Sample size: 1350
Number of deaths: 360
Number of trees: 750
Forest terminal node size: 3
Average no. of terminal nodes: 58.67333
No. of variables tried at each split: 2
Total no. of variables: 7
Analysis: RSF
Family: surv
Splitting rule: logrank *random*
Number of random split points: 3
Error rate: 30.29%
\end{verbatim}

Figure 4.1: Example output of rfsrc object.

The \textit{rfsrc()} function output contains information on the parameters as specified by the user (a limited number of them shown in Table 4.2), the forest characteristics, and predicted values, in-bag and out-of-bag. On calling the \textit{rfsrc} object some information is readily available. In Figure 4.1 the output is shown for a training set of size 1350 with 360 deaths. The output informs us that 750 trees were grown (\textit{ntree}), that the average terminal node size (\textit{nodesize}) consists of 3 observations, with an average of 58.7 terminal nodes per tree, that 2 out of a total of 7 variables were randomly selected as candidates for a split (\textit{mtry}), that per variable three split points (\textit{nsplit}) were considered, and that a logrank splitting rule (\textit{splitrules}) was used. Note that \textit{logrank ”random”} is noted to make clear that rather than deterministic splitting, a limited number of splitpoints (3) was selected on each variable. This situation is different from choosing the splitrule ”random” which will consider only a single splitpoint per candidate variable. If, however, an \textit{nsplit} of 1 is specified, the \textit{logrank} splitrule reduces to the \textit{random} splitrule. Finally, the OOB error rate is given, which is equal to \(1 - C^{**}\), where \(C^{**}\) is the OOB concordance index.

A subset of additional info that can be extracted from the \textit{rfsrc} object is shown in Table 4.2. The number of terminal nodes and the error rate can be evaluated on a tree basis. \textit{leafcount} gives the number of terminal nodes (leaves) per tree, \textit{err.rate} the prediction error per tree. Calling the \textit{rfsrc} object also returns the CHF, survival,
Table 4.2: Output from the \textit{rfsrc} function

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>leafcount</td>
<td>number of terminal nodes per tree</td>
</tr>
<tr>
<td>err.rate</td>
<td>prediction error, calculated for each tree with OOB data</td>
</tr>
<tr>
<td>chf</td>
<td>cumulative hazard function estimates as obtained for the bootstrap samples (equation (4.4))</td>
</tr>
<tr>
<td>chf.oob</td>
<td>cumulative hazard function estimates as obtained for the OOB samples (equation (4.5))</td>
</tr>
<tr>
<td>survival</td>
<td>survival predictions as obtained for the bootstrap samples</td>
</tr>
<tr>
<td>survival.oob</td>
<td>survival predictions as obtained for the OOB samples</td>
</tr>
<tr>
<td>predicted</td>
<td>Ensemble mortality as obtained for the bootstrap samples (equation (4.9))</td>
</tr>
<tr>
<td>predicted.oob</td>
<td>Ensemble mortality as obtained for the OOB samples (equation (4.10))</td>
</tr>
</tbody>
</table>

and ensemble mortality estimates: \textit{chf} and \textit{survival} are matrices with cumulative hazard function and survival probability functions, respectively, for \( n \) patients for each unique observed event time; \textit{predicted} gives the ensemble mortality estimate for each of \( n \) patients. The ensemble mortality has been scaled to event incidence. For a patient \( i \) the corresponding \textit{predicted} value represents the number of events if all individuals in the dataset had the same predictor vector \( x_i \) as individual \( i \).

For in-tree OOB estimates the output can be obtained as described previously, from the \textit{rfsrc} object. In the event of cross-validation, or training and testing on separate training and testing sets, the same estimates can be obtained from the \textit{predict.rfsrc()} function, with \textit{predict.rfsrc(rfs, newdata = test)}, where \textit{rfs} is the \textit{rfsrc} object.
4.4 Application to EURAMOS osteosarcoma data

In this section the application of Ishwaran's Random Survival Forests method to the EURAMOS osteosarcoma data is detailed. First, the measures for evaluating model performance and variable importance are discussed, followed by the cross-validation procedure, the results, and a final discussion. The sections are illustrated with excerpts from the R code, which is given in its entirety in Appendix D.

4.4.1 Performance measures and variable importance

Various performance measures are calculated to evaluate the random survival forest performance. As detailed in Section 4.2, a concordance index can be calculated using the ensemble mortality. Below, the corresponding code is shown. The function takes as argument a dataframe with survival time `surv`, status `stat`, and the ensemble mortality `pred`. Please see for additional detail Section 2.2.5, where the fully annotated function is given.

```r
# Appendix D. 2 (1–25)
cindex.mod.r <-
  function (data) {
    time <- data$surv; status <- data$stat
    x <- data$pred; n <- length(time)

    ord <- order(time, -status)
    time <- time[ord]; status <- status[ord]
    x <- x[ord]

    wh <- which(status == 1); total <- concordant <- 0

    for (i in wh) {
      for (j in ((i + 1):n)) {
        if (time[j] > time[i]) {
          total <- total + 1
          if (x[j] < x[i])
            concordant <- concordant + 1
          if (x[j] == x[i])
            concordant <- concordant + 0.5
        }
      }
    }
    return(concordant/total)
  }
```

As additional performance measures the Brier and Kullback-Leibler scores are calculated in a similar manner to the one described for neural networks in section 3.4.2. Below, the relevant code is given, after which the modifications are discussed in detail.

```r
# brier/KL function, part 1
brier.function.r <- function(rfs, stat.t, pred.matrix)
{
    # calculating censoring distribution based on training set
    object <- rfs$yvar; so <- Surv(object$time, object$stat)
time <- so[,1]; ot <- order(time)
cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)
    # censoring distribution calculated
    hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
    csurv <- predict(hatcdist, times = time, type = "surv")
    csurv[csurv == 0] <- Inf
    # unique event times are selected
    death.times <- unique(time[which(cens==1)])
    # selecting censoring probabilities at death times
    csurv.t <- csurv[which(cens==1)]
    # index of unique death times
    indexx <- seq(1, length(time[which(cens==1)]))[which(duplicated(time[which(cens==1)]))]
    csurv.tt <- csurv.t[indexx]
    # calculating censoring distribution based on test set.
preds <- pred.matrix; so <- Surv(stat.t$surv, stat.t$stat)
time <- so[,1]; ot <- order(time)
cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)
    hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
    csurv <- predict(hatcdist, times = time, type = "surv")
    csurv[csurv == 0] <- Inf
    # The scores are calculated for each time t0, which is a # unique death time.
btime <- death.times
}
```

96
The structure of the random forest ensures that predictions can only be made for observed event times (see also section 4.2, algorithm step 4). As such, survival probabilities cannot be estimated for every single event/censoring time. In a training set/test set context, either in a cross-validation or when fitting the final model, these
unique event times are given by the training set. The model performance however - through the Brier and Kullback Leibler score - is to be assessed on the test set. As detailed in section 3.4.2, when calculating the scores in a censored setting inverse censoring probability weighting is employed. For a given time \( t_0 \), at which the Brier or Kullback-Leibler score are assessed, the observation is weighted depending on observed survival time and censoring status. Observations censored before the time of interest \( t_0 \) are ignored. Observations with an event time \( t \) for which \( t \leq t_0 \) are weighted with the censoring probability at time \( t \). Observations which either have an event or are censored after the time of interest \( t_0 \) are weighted with the censoring distribution at time \( t_0 \).

As the interest is in assessing performance on the test set, the event/censoring times \( t \) for the relevant observations are obtained from the test set, and in order to weight these observations appropriately the censoring distribution must be obtained from the test set. The censoring distribution at each time of interest \( t_0 \) must also be calculated however, for weighting the observations that have survival times \( > t_0 \). The latter censoring distribution is obtained from the training set, as survival probabilities can only be calculated for the event times as observed in the training set.

In the code the unique death times \( t_0 \) as observed in the training set are given by \( btime \), and the corresponding censoring distribution is given by \( csurv.tt \). The censoring distribution for the test data, assessed at times \( t \) is given by \( csurv \). This is a necessary modification of the Bier and Kullback-Leibler function given for neural networks in 3.4.2, where a single censoring distribution suffices, as the test set performance is assessed at all unique survival times of the test set, rather than the training set.

Two additional measures of performance are considered which both rely on the ensemble mortality estimates. The ensemble mortality can be compared to a prognostic index - as one would obtain from a Cox proportional hazards model, and was discussed earlier in this chapter as a tool for measuring the concordance index and by extension the prediction error. Two other measures relying on the prognostic index have been described in literature, which can also be applied to the ensemble mortality. In the first measure the difference in survival curves of high and low risk patients is quantified using the log-rank \( \chi^2 \)-statistic. Patients with scores above the median are classified as high risk, below the median as low risk. The larger the \( \chi^2 \) value, the better the model can distinguish between these groups. The second measure involves a straightforward Cox regression of the survival time and status on the ensemble mortality index. The ensemble mortality scores are normalized to a range between 0
and 1. The hazard ratio measures the increase in risk when moving from the lowest scoring patient (at 0) to the highest scoring patient (at 1).\[56\]

\[
test\_pred\_d \leftarrow \text{\texttt{predict}}\text{\texttt{.rfs}\texttt{r}}(\text{\texttt{rfs}}, \text{\texttt{newdata}} = \text{\texttt{test}}\_p)\\pred\_scores \leftarrow \text{\texttt{test}}\_pred\_d$\texttt{\$predicted}$
\]

\[
\# \text{calculating hazard ratio from normalized}\\\# \text{prognostic index}\\pred\_scores\_norm \leftarrow (\text{\texttt{pred}}\_\text{\texttt{scores}} - \text{\texttt{min}}(\text{\texttt{pred}}\_\text{\texttt{scores}})) / (\text{\texttt{max}}(\text{\texttt{pred}}\_\text{\texttt{scores}}) - \text{\texttt{min}}(\text{\texttt{pred}}\_\text{\texttt{scores}}))\\cv \leftarrow \text{\texttt{Coxph}}(\text{\texttt{Surv}}(\text{\texttt{time}}, \text{\texttt{stat}}) \sim \text{\texttt{pred}}\_\text{\texttt{scores}}\_\text{\texttt{norm}},\\ \text{\texttt{data}}=\text{\texttt{cbind}}(\text{\texttt{test}}\_p, \text{\texttt{pred}}\_\text{\texttt{scores}}\_\text{\texttt{norm}}))\\\text{HR}[d] \leftarrow \text{\texttt{exp}}(cv$\texttt{\$coefficients})$
\]

\[
\# \text{obtaining logrank Chisq statistic based on median}\\\# \text{prognostic index}\\\text{group} \leftarrow \text{\texttt{rep}}(1, \text{\texttt{nrow}}(\text{\texttt{test}}\_p))\\\text{group}[\text{\texttt{which}}(\text{\texttt{pred}}\_\text{\texttt{scores}} \leq \text{\texttt{median}}(\text{\texttt{pred}}\_\text{\texttt{scores}}))] \leftarrow 2\\\text{group} \leftarrow \text{\texttt{as.factor}}(\text{\texttt{group}})\\\text{temp.dat} \leftarrow \text{\texttt{cbind}}(\text{\texttt{test}}\_p, \text{\texttt{group}})\\\text{LRX}[d] \leftarrow \text{\texttt{survdiff}}(\text{\texttt{Surv}}(\text{\texttt{time}}, \text{\texttt{stat}}) \sim \text{\texttt{group}},\\ \text{\texttt{data}}=\text{\texttt{temp.dat}})\$\texttt{chisq}$
\]

\[
\# \text{Appendix D.3.1 (45–56)}
\]

As detailed in Sections 4.1 and 4.2, by growing a forest of trees the straight-
forward interpretation of a single tree is lost. Two measures of have been proposed to
quantify the effect of a predictor in a forest- minimal depth and variable importance
(VIMP). Minimal depth is calculated by finding the maximal subtrees for a given
predictor and measuring the smallest distance from the maximal subtrees to the
root node. Consider a variable $x$ that has been selected as a split variable two
times. The first split occurs in one branch of the tree, two nodes down from the
root node. The maximal subtree is then the entire branch of the tree following this
split. The second split occurs in a different branch of the tree, one node down from
the root node. Once more, the maximal subtree is the branch from the split on
downwards. The smallest distance from the two maximal subtrees to the root node
is then given by the second split, which is just one node removed from the root
node. The minimal depth is then one. Minimal depths are established for every
tree in the forest and then averaged over. The smaller the value, the better the predictor. An attractive feature of this measure is that it does not depend on the OOB prediction error, whereas the VIMP measure does. The VIMP is a measure of difference in OOB error of the forest as is, and the forest after randomization has been introduced in the variable of interest. In the default approach of calculating VIMP for a variable $x$ the $x$-variable is permuted for the OOB cases, and a new OOB error is calculated. The difference between the original OOB error and the permuted OOB error then gives the VIMP. The larger the value, the more influential the variable. The alternative approach is introducing the randomization in the splitting process itself. Here, for every split on $x$ each OOB case is randomly allocated to a daughter node. In these two approaches the VIMP is calculated by tree, and then averaged over. Alternatively, this can also be applied directly to the forest ensemble. The variable importance can be extracted from an rsf object by calling it in the function \textit{vimp()}, which takes the argument \textit{importance}. VIMP by variable permutation is obtained by specifying "permute", VIMP by random node assignment by specifying "random", and the respective ensemble measures by specifying "permute.ensemble" and "random.ensemble". Minimal depth and the default VIMP can be obtained through calling the rsf object in the function \textit{var.select.rfsrfc()}.
4.4.2 Cross-validation and evaluation of performance measures

As detailed in Section 4.2 the boostrapping procedure gives rise to out of bag (OOB) data, which can be used to estimate the generalization error. Alternatively, the generalization error can be estimated through cross-validation. Both favourable and unfavourable qualities of OOB estimation have been described. The main advantage is the computational speed, as OOB estimation only requires a single model to be fitted rather than the \( k \) models of a cross-validation procedure. It has, however, been suggested that the approach is biased, and that this bias may be associated with the tuning parameters.[50, 7]. Another consideration is that the OOB procedure is only available to random forests - the neural networks described in chapter 3 were tuned through 5-fold cross-validation. In light of potential bias and with the purpose of maintaining consistency across the chapters, the choice was made to tune the random forests with cross-validation.

The EURAMOS osteosarcoma data was divided in a training and validation set as described in 3.4.3. A five-fold cross-validation was performed on the training set. The following parameters were tuned: \textit{mtry} (the number of candidate variables randomly selected at each splitpoint); \textit{nodesize} (the average number of observations per terminal node); \textit{nsplit} (the number of splitpoints randomly selected per candidate variable); \textit{nodedepth} (the maximum number of splits from the rootnode to a single terminal node); \textit{splitrule} (the splitting rule used to evaluate the survival difference of the two daughternodes for a potential split). The parameters were tuned in pairs by means of a grid search. First, all combinations of \textit{mtry} and \textit{nodesize} are evaluated (Section 4.4.2.1). Then, for the optimal \textit{mtry} and \textit{nodesize} the performance of different \textit{nsplit} and \textit{nodedepth} is compared (Section 4.4.2.2). The tuning of these four parameters happens under the default \textit{logrank} splitrule. Finally, for the optimal model all three different splitting rules are compared (Section 4.4.2.3). Following this, a closer look is taken at the effect of each single tuned parameter on the model performance. Additionally, the effect of extreme parameter choices are examined (Section 4.4.2.4).

Prior to the actual cross-validation, the random forest error was examined for random combinations of parameters. A general pattern can be observed for the error. For a smaller number of trees the error fluctuates, remaining stable for all three forest at 500 trees or more. As detailed in section 4.2 the algorithms default choice
is 1000 trees. The choice was made to reduce this to 750 to improve computationally efficiency without compromising accuracy.

Figure 4.2: OOB prediction error against number of trees for three randomly selected forests

The cross-validation functions are given in Appendix D.3.1.
4.4.2.1 Cross-validation for node size and number of candidate splitting variables

In the first cross-validation step different combinations of node size (nodesize) and number of candidate splitting variables (mtry) were considered. Node sizes from 1 to 600 in increments of 50 were evaluated, and 1 to 7 candidate splitting variables. Note that in the event of 7 candidate splitting variables the random forest is reduced to a bagging procedure, as all variables, rather than a subset, are considered at each splitpoint. Results are shown in Figures 4.3 and 4.4. The 13 different node sizes are evaluated for each number of candidate splitting variables. The concordance index (a), the Brier scores from 1 to 6 years after surgery (b), and the Kullback-Leibler scores for the same time period (c) are given. For each observation the standard deviation across the five folds of the training data is given. For all different numbers of candidate variables node sizes of 550 and 600 perform the worst with a C-index of 0.5, indicating models with no discriminative ability. The remaining node sizes have very similar C-index, both within and across the number of candidate variables, with one exception. The smallest node size of 1 performs worse as the number of splitting variables increases, as evidenced by a gradually decreasing C-index. This same pattern can be observed in the Brier and Kullback-Leibler scores. The lines for most node sizes follow a very close pattern, but those for 550 and 600 are noticeably higher (dark green, dark red, respectively, note that they overlap). For a node size of 1 (black) it can be observed that the Brier and Kullback-Leibler scores become worse from two splitting variables onwards. For four to seven candidate variables the scores are comparable to those for 550 and 600 node sizes. Of note is that while the difference in Brier and Kullback-Leibler scores supports what is observed in the C-index, the difference between the better and poorer performing models is less stark. Only a slight increase in score values is observed for node sizes of 550 and 600, while the C-index plummets from almost 0.7 to 0.5. Also of note is that only a slight decrease in C-index is observed for the node size of 1, while the Brier and Kullback-Leibler scores are comparable to those for 550 and 600.

In summary, these results suggest that very small node sizes should not be paired with larger numbers of candidate splitting variables. For larger node sizes, no great effect of the number of candidate splitting variables on performance is observed. Notably, node sizes of 550 and 600 result in models with no discriminative ability. The C-index was used to select the optimal model. A model with an intermediate node size of 200 and 2 candidate splitting variables was chosen. Note, however, that the values for all models save the 1, 550 and 600 node size models, are very similar.
Figure 4.3: Cross-validated performance measures for the number of candidate splitting variables (mtry) and average number of unique cases per terminal node (nodesize). a: concordance index, b: brier score, c: Kullback-Leibler score. For 1, 2, 3 and 4 candidate splitting variables, for 1-600 node sizes, in steps of 50. Standard deviation across cross-validation folds is given by error bars.
4.4.2.2 Cross-validation for number of splitting points and node depth

In the second cross-validation step, having previously established a node size of 200 and 2 candidate splitting variables as optimal, the node depth ($nodedepth$) and the number of split points ($nsplit$) were tuned. For node depth values from 1 to 15 in increments of 2 were considered, for split points 0 to 5 split points. Note that setting $nsplit$ to 0 enacts deterministic splitting, in which all possible split points are considered. The results for 0 and 3 split points are shown in Figure 4.5. For each split point, all 8 node depths are shown. The C-index (a), the Brier score (b), and the
Kullback-Leibler score (c) are given as performance measures. For each observation 
the standard deviation across the five folds of the training data is given. For the 
different node depths, barely any difference was observed across or within the number 
of split points. A node-depth of 1 (black) has a marginally worse performance, with 
slightly higher Brier and Kullback-Leibler scores and a marginally lower C-index. The 
remaining node-depths have almost identical values for the C-index and completely 
overlap in the score plots. The exact values for the Brier score for 0 and 3 split points 
are given in tables 4.3 and 4.4 respectively. The values are given in 8 decimals as the 
differences are very slight. Note that for both little variation due to node depth is 
oberved and that from a nodesize of 9 onwards the scores stay the same within a 
precision of 8 decimals, showing that when a sufficiently large nodedepth is specified, 
the parameter loses effect. An intermediate node-depth of 7 and 3 splitting points 
were chosen as optimal parameter values. The full cross-validation plots are given in 
Appendix D.4.

Figure 4.5: Cross-validated performance measures for the number of candidate split- 
ting variables (mtry) and average number of unique cases per terminal node (nod- 
size). a: concordance index, b: brier score, c: Kullback-Leibler score. For 0 and 
3 splitting points, for 1-15 node depths, in steps of 2. Standard deviation across 
cross-validation folds is given by error bars
Table 4.3: Brier score at 1-6 years for 0 split points and 1-15 (by 2) nodes

<table>
<thead>
<tr>
<th></th>
<th>1 year</th>
<th>2 years</th>
<th>3 years</th>
<th>4 years</th>
<th>5 years</th>
<th>6 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.D. 1</td>
<td>0.04967814</td>
<td>0.11393074</td>
<td>0.15683624</td>
<td>0.17530448</td>
<td>0.18994534</td>
<td>0.20216225</td>
</tr>
<tr>
<td>N.D. 3</td>
<td>0.04855083</td>
<td>0.10916227</td>
<td>0.14993533</td>
<td>0.16683324</td>
<td>0.18157357</td>
<td>0.19538176</td>
</tr>
<tr>
<td>N.D. 5</td>
<td>0.04835616</td>
<td>0.10825016</td>
<td>0.14851083</td>
<td>0.16504330</td>
<td>0.17964381</td>
<td>0.19389761</td>
</tr>
<tr>
<td>N.D. 7</td>
<td>0.04834685</td>
<td>0.10813950</td>
<td>0.14833052</td>
<td>0.16482032</td>
<td>0.17937029</td>
<td>0.19367810</td>
</tr>
<tr>
<td>N.D. 9</td>
<td>0.04834431</td>
<td>0.10812953</td>
<td>0.14831157</td>
<td>0.16480033</td>
<td>0.17933166</td>
<td>0.19363833</td>
</tr>
<tr>
<td>N.D. 11</td>
<td>0.04834422</td>
<td>0.10813032</td>
<td>0.14831180</td>
<td>0.16479792</td>
<td>0.17932873</td>
<td>0.19363499</td>
</tr>
<tr>
<td>N.D. 13</td>
<td>0.04834412</td>
<td>0.10812992</td>
<td>0.14831142</td>
<td>0.16479775</td>
<td>0.17932831</td>
<td>0.19363432</td>
</tr>
<tr>
<td>N.D. 15</td>
<td>0.04834412</td>
<td>0.10812992</td>
<td>0.14831142</td>
<td>0.16479775</td>
<td>0.17932831</td>
<td>0.19363432</td>
</tr>
</tbody>
</table>

Table 4.4: Brier score at 1-6 years for 3 split points and 1-15 (by 2) nodes

<table>
<thead>
<tr>
<th></th>
<th>1 year</th>
<th>2 years</th>
<th>3 years</th>
<th>4 years</th>
<th>5 years</th>
<th>6 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.D. 1</td>
<td>0.04965757</td>
<td>0.11366092</td>
<td>0.15635005</td>
<td>0.17474879</td>
<td>0.18925610</td>
<td>0.20146698</td>
</tr>
<tr>
<td>N.D. 3</td>
<td>0.04857681</td>
<td>0.10904285</td>
<td>0.14952138</td>
<td>0.16637041</td>
<td>0.18068633</td>
<td>0.19407056</td>
</tr>
<tr>
<td>N.D. 5</td>
<td>0.04845074</td>
<td>0.10850748</td>
<td>0.14875203</td>
<td>0.16537561</td>
<td>0.17956902</td>
<td>0.19306399</td>
</tr>
<tr>
<td>N.D. 7</td>
<td>0.04844543</td>
<td>0.10848798</td>
<td>0.14872241</td>
<td>0.16532510</td>
<td>0.17950351</td>
<td>0.19300074</td>
</tr>
<tr>
<td>N.D. 9</td>
<td>0.04844545</td>
<td>0.10848798</td>
<td>0.14872245</td>
<td>0.16532479</td>
<td>0.17950360</td>
<td>0.19300142</td>
</tr>
<tr>
<td>N.D. 11</td>
<td>0.04844545</td>
<td>0.10848798</td>
<td>0.14872245</td>
<td>0.16532479</td>
<td>0.17950360</td>
<td>0.19300142</td>
</tr>
<tr>
<td>N.D. 13</td>
<td>0.04844545</td>
<td>0.10848798</td>
<td>0.14872245</td>
<td>0.16532479</td>
<td>0.17950360</td>
<td>0.19300142</td>
</tr>
<tr>
<td>N.D. 15</td>
<td>0.04844545</td>
<td>0.10848798</td>
<td>0.14872245</td>
<td>0.16532479</td>
<td>0.17950360</td>
<td>0.19300142</td>
</tr>
</tbody>
</table>

4.4.2.3 A comparison of different splitting rules

In the previous two cross-validations the following parameter values were selected: an mtry of 2, a nodesize of 200, an nsplit of 3 and a nodedepth of 7. In the final step, the three possible splitting rules are compared. Note that logrank is the default and was used in the previous cross-validations throughout. logrankscore is a modification of the logrank test statistic and assumes an ordering of the variable. Note that this is not necessarily applicable to the discrete variables of the EURAMOS osteosarcoma data. A random splitting rule, considers just one split point per variable (and makes the nsplit parameter irrelevant). Figure 4.6 shows the C-index (a), Brier score (b) and Kullback-Leibler score (c) for the logrank splitting rule (black), the logrankscore splitting rule (red), and the random splitting rule (blue). For each observation the standard deviation across the five folds of the training data is given. The observed values are very close to one another (note that for the C-index in (a) the y-axis...
represents a very small interval). When looking at both scores in Figures 4.6b and 4.6c it can be observed that the \textit{logrank} splitting rule results in slightly lower values. As such, \textit{logrank} was retained as the splitting rule of choice.

Figure 4.6: Cross-validated performance measures for different splitting rules a: concordance index, b: brier score, c: Kullback-Leibler score. For the \textit{logrank}, \textit{lo-grankscore}, and \textit{random} splitting rule. Standard deviation across cross-validation folds is given by error bars.
4.4.2.4 A closer look at the individual tuning parameters

The cross-validation approach resulted in the selection of a model with parameters \( mtry = 3, \ nsplit = 2, \ nodesize = 200 \) and \( nodedepth = 7 \). In order to gain more insight into the behaviour of the individual parameters a number of additional models were fitted. For the first comparison, for each optimal parameter a model was fitted specifying solely that parameter and leaving the others unrestrained or at the \( \text{rfsrc()} \) function default value (see Section 4.3 for a detailed description of the function). For the second comparison, high and low values were specified for each parameter to gain insight into how extreme values influence model performance and prediction. Model performance was evaluated through the C-index, Brier score, and Kullback-Leibler score, which, as previously described, are well-established measures for right-censored survival data. Two additional measures were introduced, both based on the prognostic index (ensemble mortality in the context of random survival forests). As described in Section 4.4.1, the \( \log\text{-rank} \chi^2 \)-statistic quantifies how well the model can distinguish between low and high risk patients, where low risk patients have ensemble mortality score smaller than the median, high risk patients greater than the median. The second measure is the hazard ratio, as obtained from a simple regression Cox regression of the normalized ensemble mortality scores. While described in previous literature [56], these measures have not been used as a selection criterion, and they are presented here with the purpose of gaining more insight into possible model differences. A closer look at the model performance was taken by evaluating plots of the survival probabilities of individual patients. Figures 4.7 and 4.8 show the performance measures and survival plots, respectively, for the seven models listed below.

1. A model with all parameters tuned in cross-validation: \( mtry = 2, \ nsplit = 3, \ nodesize = 200, \ nodedepth = 7 \). Displayed in 'brown' in Figure 4.7, with survival probabilities shown in Figure 4.8a

2. A model with no parameters tuned. \( \text{rfsrc}() \) default is \( mtry = \sqrt{p} \), rounded up, giving \( \sqrt{7} = 3 \) candidate variables considered at each splitpoint. By default \( nsplit = 0 \), enacting deterministic splitting. \( nodesize \) and \( nodedepth \) are left unrestrained. Displayed in 'orange' in Figure 4.7, with survival probabilities shown in Figure 4.8b.

3. A model with only \( nsplit \) tuned, with \( nsplit = 3 \). Displayed in 'green' in Figure 4.7, with survival probabilities shown in Figure 4.8c
4. A model with only \textit{mtry} tuned, with \textit{mtry}=2. Displayed in 'blue' in Figure 4.7, with survival probabilities shown in Figure 4.8d.

5. A model with only \textit{nodesize} tuned, with \textit{nodesize}=200. Displayed in 'purple' in Figure 4.7, with survival probabilities shown in Figure 4.8e.

6. A model with only \textit{nodedepth} tuned, with \textit{nodedepth}=7. Displayed in 'pink' in Figure 4.7, with survival probabilities shown in Figure 4.8f.

7. A model with all parameters but \textit{nodesize} tuned: \textit{mtry}=2, \textit{nsplit}=3, \textit{nodedepth}=7. Displayed in 'dark red' in Figure 4.7, with survival probabilities shown in Figure 4.8g. Note that this model was added on observing deviating prediction patterns for the tuned \textit{nodesize} parameter in figure 4.8e.

8. Survival probabilities as obtained from the Cox proportional hazards model, shown in Figure 4.8h, not shown in Figure 4.7.

Figure 4.7: Comparison of C-index, Brier scores, Kullback-Leibler scores for different combinations of tuned parameters. a) C-index, b) Brier score, c) Kullback-Leibler score, d) Ensemble mortality hazard ratio, e) Log-rank $\chi^2$ statistic. Standard deviation across cross-validation folds is given by error bars.
The model performance as measured by the C-index (Figure 4.7a) the Brier score (Figure 4.7b) and the Kullback-Leibler score (Figure 4.7c) is consistent over all three measures. The model with all parameters tuned (model 1 on the list, brown in Figure 4.7) has the highest C-index and the lowest score values, closely followed by the model with only node size tuned (model 6, pink) and the model with all parameters but node size tuned (model 7, dark red). Overall, the model performance is comparable for all models, save for the untuned model (model 2, orange) and the model with only the number of candidate splitting variables tuned (model 4, blue), which have noticeably lower and higher values for the C-index and scores, respectively. Notably, when comparing the models on ensemble mortality hazard ratio (Figure 4.7d) and log-rank $\chi^2$ statistic (Figure 4.7e) a different pattern is observed. For the log-rank $\chi^2$ statistic the model performance is comparable for all models, while for the hazard ratio the model with only the number of split points tuned (model 3, green) has a noticeably higher value with a comparatively very large standard error. Overall, the C-index and Brier and Kullback-Leibler scores indicate a slightly better performance for the fully tuned model, a difference which is greatest when compared to an untuned model, or a model with only the number of candidate variables tuned. For the remaining models the performance is similar. Neither the hazard ratio or the log-rank $\chi^2$ statistic consistently support these results.

In Figure 4.8 a different, informal, comparison is made by examining the survival probabilities for all test patients. Here, the model has been trained on the full training data, and predictions have been made on the validation data. Note that this should not be considered a cross-validation step, or a strict model selection step - as it would be inappropriate to use the validation data for this purpose - but merely a quick look at the general predictive patterns associated with the individual models. In the figure the patients are coloured according to the histological response: patients with a poor histological response (red), and patients with a good histological response (blue). Figures 4.8a to 4.8g give the predictions for the seven random forest models, while Figure 4.8h gives the same for the Cox model. Two general prediction patterns can be distinguished: in figures b (model 2), c (model 3), d (model 4), f (model 6) and g (model 7) the predictions are wider spread than in figures a (model 1) and e (model 5), where the predictions are more aggregated. These last two models are the only models for which the node size has been tuned (nodesize=200).
Figure 4.8: Predicted survival curves for all patients for random forest models with different tuned parameters. a) all parameters tuned; b) no parameters tuned; c) number of split points tuned; d) number of candidate variables tuned; e) node size tuned; f) node depth tuned; g) all parameters but nodesize tuned; h) Predicted survival curves for Cox proportional hazards model.
A closer look at the models with a more widely distributed prediction pattern, shows that models 6 (e) and 7 (f) display slightly more clustering. Here, model 6 is the random survival forest model with only node depth tuned, and model 7 the model with node size, split points, and number of candidate variables tuned. The hyperparameters node size and node depth control the extent of forest branching by increasing the minimal terminal node size and limiting the forest depth, respectively. Use of these regularization parameters appears to result in the aggregation of predictions. Going from the overall prediction pattern to the distribution of histological response within each plot, it can be seen that clear, distinguished bands are present in the aggregated plot. In light of the distinct patterning observed for histological response, which has been previously identified as a strong predictor by both the Cox model and the neural network, and in light of the marginally higher performance observed for the aggregated plots, it could be posed that regularization via node size or node depth, while changing the overall prediction shape, retains the relative order of disease severity within the patients.

Figures 4.9 and 4.10 show the performance measures and survival plots for nine models, that have been fitted with extremely low and high values of individual parameters.

1. A model with no parameters tuned. \texttt{rfsrc()} default is \texttt{mtry}=$\sqrt{p}$, rounded up, giving $\sqrt{7} = 3$ candidate variables considered at each split point. By default \texttt{nsplit}=0, enacting deterministic splitting. \texttt{nodesize} and \texttt{nodedepth} are left unrestrained. Displayed in 'brown' in Figure 4.9, with survival probabilities shown in Figure 4.10a.

2. A model with \texttt{nsplit}=5, a comparatively high number of random split points. Displayed in 'orange' in Figure 4.9, with survival probabilities shown in Figure 4.10b.

3. A model with \texttt{nsplit}=1, a comparatively small number of random split points. Displayed in 'orange' in Figure 4.9, with survival probabilities shown in Figure 4.10c.

4. A model with \texttt{mtry}=7. Using the all the variables as candidate variables for a split reduces the random forest to straightforward bagging. Displayed in 'blue' in Figure 4.9, with survival probabilities shown in Figure 4.10d.
5. A model with \( mtry = 1 \). Displayed in 'purple' in Figure 4.9, with survival probabilities shown in Figure 4.10e.

6. A model with \( \text{nodesize} = 400 \). Increasing the minimal terminal node size enacts regularization on the forest. Displayed in 'pink' in Figure 4.9, with survival probabilities shown in Figure 4.10f.

7. A model with \( \text{nodesize} = 1 \). Displayed in 'pink' in Figure 4.9, with survival probabilities shown in Figure 4.10g.

8. A model with \( \text{nodedepth} = 15 \). Displayed in 'pink' in Figure 4.9, with survival probabilities shown in Figure 4.10h.

9. A model with \( \text{nodedepth} = 1 \). Decreasing the maximal node depth enacts regularization on the forest. Displayed in 'turquoise' in Figure 4.9, with survival probabilities shown in Figure 4.10i.

Figure 4.9: Comparison of C-index, Brier scores, Kullback-Leibler scores for random forests with extreme parameter values. a) C-index, b) Brier score, c) Kullback-Leibler score, d) Ensemble mortality hazard ratio, e) Log-rank \( \chi^2 \) statistic. Standard deviation across cross-validation folds is given by error bars.
As in Figure 4.7, once more it can be observed that in Figure 4.9 the C-index (a), Brier score (b) and Kullback-Leibler score (c) are consistent in measuring model performance. Models 5 (purple, 1 candidate variable), 6 (pink, node size of 400), and 9 (turquoise, node depth of 1) have a noticeably higher C-index and lower score values than the remaining models. As detailed before, a higher node size and smaller node depth ensure increased model regularization and protect against overfitting. The random selection of a limited number of variables as candidates for a split is a feature unique to random forests, and has the beneficial quality of improving model performance by decreasing the generalization error and allow for potential weaker variable interactions (see also section 4.1). When contrasting the model that uses only a single candidate splitting variable (model 5, pink) with the model that uses all seven variables (model 4, blue), the former gives a superior performance across the three measures. For the ensemble mortality hazard ratio (d) a different pattern is observed, where all models have similar values, save for model 5 (purple, 1 candidate variable), which has a much higher value and a large standard deviation Model 3 (green, 1 splitting point) also has a comparatively large standard deviation, which was also observed in Figure 4.7d for a model with 3 splitting points. Altogether, this suggests that models in which either the number of candidate variables or the number of split points is limited, have higher variability, when the performance is measured by means of the ensemble mortality hazard ratio. The log-rank $\chi^2$ statistic (4.9e) shows a pattern that is comparable to but less pronounced than the one observed for the C-index (4.9a).

In Figure 4.10 the prediction plots for all nine models are shown. Individual survival curves are shown in red for patients with a poor histological response and in blue for patients with a good histological response. Once more, two general patterns can be distinguished; widespread predictions and clustered predictions. Clustering is observed in e (model 5, 1 candidate variable), f (model 6, node size 400), and i (model 9, node depth of 1). In Figure 4.8f it was observed that a nodesize of 7 resulted in slightly more clustered predictions than the untuned model (Figure 4.8b, 4.10a). Figure 4.10i suggest that decreasing node depth increases the clustering of predictions. Looking back to figure 4.8e, which shows the prediction curves for a model with a node size of 200, it can be observed that while markedly less clustered than for a model with a node size of 400, shown here in Figure 4.10f, the predictions are not nearly as widespread as for a node size of 1 (Figure 4.10g), once more indicating that increasing the node size results in a clustering of predictions.
Figure 4.10: Predicted survival curves for all patients for random forest models with extreme parameter values. a) no parameters tuned; b) 5 random split points; c) 1 random split point; d) 7 candidate variables; e) 1 candidate variable; f) node size of 400; g) node size of 1; h) node depth of 15; i) node depth of 1

Of interest is the clustered pattern observed for the model with 1 candidate variable in Figure 4.10e. Previously, in Figure 4.8d the predictions for a model with two candidate variables were given, displaying no clustering at all, suggesting that the random forest is sensitive to minor changes in the \textit{mtry} parameter. Overall, Figures 4.7 to 4.10 show 1) that increased node size, decreased node depth, and a smaller number of candidate variables result in more clustered predictions and 2) that the models which show the most clustering in predictions are the ones with the highest
C-index and lowest Brier and Kullback-Leibler scores. The effect of varying parameters on prediction is also illustrated in Figure 4.11, where individual predictions of the models are shown for a patient with a poor histological response and lung metastases. Each plot compares the extremes within one parameter. For the number of random split points (b) the predictions for 1 split point and 5 split points are very similar. However, for the number of candidate variables (c) the predicted survival curve is much higher for 1 candidate variable than for 7 candidate variables. The same can be seen for a model with a node size of 400 versus a model with a node size of 1 (d), and for a model with a node depth of 1 versus a node depth of 7. Looking back at the overall prediction plots in 4.10 the higher survival predictions correspond to the plots with the predictions that are clustered and comparatively higher.

Figure 4.11: Patient-specific prediction for random forests with extreme parameter values, for a patient with a poor histological response and lung metastases, and reference/mean values for the remaining predictors. In each plot the random forest prediction is given (green) and the Cox model prediction (black). a) untuned model; b) 1 random split point (dotted line), 5 random split points (solid line); c) 1 candidate splitting variable (dotted line), 7 candidate splitting variables (solid line); d) Node size of 1 (dotted line), node size of 400 (solid line); e) Node depth of 1 (dotted line), node depth of 15 (solid line)
4.4.3 Results

The original cross-validation procedure, described in Sections 4.4.2.1-4.4.2.3, resulted in a model with \( mtry=2 \), \( nsplit=3 \), \( nodesize=200 \), \( nodedepth=7 \), and a logrank splitting rule. A closer look at the behaviour of individual parameters revealed that low values for \( nodedepth \), high values for \( nodesize \), and an \( mtry \) of 1, resulted in relatively high and clustered survival predictions. Comparatively high and low values were observed for the C-index and the Brier and Kullback-Leibler scores, respectively (Sections 4.4.2.4). The same was observed for the originally selected model. A slightly modified model, however, in which \( nodesize \) was left unrestrained, scored closely in performance measures, and displayed markedly less clustering. In this results section, both models are evaluated. Additionally, an untuned random forest model is added. Reportedly, an advantage of the random forests when compared to other machine learning approaches, such as neural networks or support vector machines, is that they perform comparatively well even in the absence of tuning. As such, three random forests models, fitted on the EURAMOS osteosarcoma data, are considered and compared to the Cox proportional hazards model.

1. RSF (1): a model as found through cross-validation, using the C-index as a selection criterion. For this model, \( mtry=2 \), \( nsplit=3 \), \( nodesize=200 \), \( nodedepth=7 \), and a logrank splitting rule is used.

2. RSF (2): a model with \( nodesize \) unrestrained, with the purpose of reducing survival prediction clustering. For this model, \( mtry=2 \), \( nsplit=3 \), \( nodedepth=7 \), and a logrank splitting rule is used.

3. RSF (3): a model with no parameters tuned. For this model, \( mtry \) defaults to 3, \( nsplit \) to 0 (enacting deterministic splitting), \( nodedepth \) and \( nodesize \) are left unrestrained, and a logrank splitting rule is used.

The three models are each fitted to the full training data \((n=1350)\) and their performance is evaluated on the validation data \((n=675)\). The concordance index, the ensemble mortality hazard ratio and the logrank \( \chi^2 \)-statistic for the three models and the Cox proportional hazards model are given in Table 4.5. The C-index is marginally higher for the Cox model than for the RSF(1) model, followed closely by the RSF(2) and RSF(3) models. For the hazard ratio, calculated from the prognostic index and ensemble mortality, for the Cox and RSF model, respectively, a different pattern is observed. The Cox model still has the highest performance, but
now followed by RSF(2), RFS(1) and RFS(3). Note that the relative difference between the models is greater here. According to the logrank $\chi^2$-statistic the optimal model would be RFS(1), closely followed by the Cox model and RSF(2). The hazard ratio is an indication of how predictive the prognostic index is, while the logrank $\chi^2$-statistic quantifies the survival difference between high risk patients ($\geq$ median ensemble mortality) and low risk patients ($< $ median ensemble mortality). Overall, the Cox model outperforms the random forest models slightly according to the C-index, more markedly so when considering the hazard ratio, and not when looking at the logrank $\chi^2$-statistic. Notably, the differences for the C-index are slight, ranging from 0.703 to 0.671, indicating there is not one obviously superior model. In this context it is also of interest to note is that even an untrained random forest- RSF(3) - gives a very good performance with a C-index of 0.671.

Table 4.5: C-index, hazard ratio and log-rank $\chi^2$-statistic for the three final random survival forests (RSF) models and the Cox proportional hazards model

<table>
<thead>
<tr>
<th></th>
<th>RSF (1)</th>
<th>RSF (2)</th>
<th>RSF (3)</th>
<th>Cox</th>
</tr>
</thead>
<tbody>
<tr>
<td>All tuned</td>
<td>0.698</td>
<td>0.689</td>
<td>0.671</td>
<td>0.703</td>
</tr>
<tr>
<td>All but node size</td>
<td>14.00</td>
<td>19.95</td>
<td>9.85</td>
<td>25.98</td>
</tr>
<tr>
<td>Untuned</td>
<td>Logrank $\chi^2$</td>
<td>58.05</td>
<td>51.54</td>
<td>32.07</td>
</tr>
</tbody>
</table>

Figure 4.12 shows the Brier (solid line) and Kullback-Leibler scores (dotted line) for the three RSF models and the Cox model. Both scores for the three random forests model follow a very similar pattern over time, decreasing at around four years after surgery, while the scores for the Cox model continue to increase. Until approximately 3 years, the scores for all four models are very close, after which they start to diverge. No real distinction can be made between RFS(1) and RFS(2), while RFS(3) has slightly higher values, most noticeably for the Kullback-Leibler scores. Based on the Brier and Kullback-Leibler scores the random forest models appear to be superior to the Cox model. Note, however, that the relevant scores have only been obtained for the first six years, as the structure of the random forest only allows for survival prediction at the times events are observed, which is earlier in the study.
Figure 4.12: Comparison of Brier scores (solid line) and Kullback-Leibler scores (dotted line) for a random forest with all parameters tuned (red), with all but node size tuned (green), no parameters tuned (purple) and the Cox model (black).

In Figure 4.13 patient specific predictions are shown for a reference patient (green), a patient with poor histological response (orange) and a patient with poor histological response and the definite/possible presence of lung metastases (purple). The predictions from each random forest model (dotted line) are compared against the Cox model predictions (solid line). When looking at the RFS(1) model (all parameters tuned) a significant divergence from the Cox model predictions can be observed. Survival for the reference patient is underestimated, while it is overestimated for the patient with poor histological response and lung metastases. In contrast, the RFS(2) model, in which the node size has been left unrestrained, closely matches the Cox model predictions. Looking at Figure 4.14, it can be observed that for the RFS(1) model in 1a/1b the predictions are constrained to a narrower range than for the RFS(2) model in 2a/2b, or indeed the untuned RFS(3) model in 3a/3b. The patient specific predictions of the RFS(3) model follow the same general pattern as observed for the RSF(2) model and the Cox model. Notably, the predictions of the untuned model are less smooth than those of either tuned models.
Figure 4.13: Patient-specific predictions for random survival forest models (dotted lines) and Cox proportional hazards model (solid lines). Shown are predictions for a reference patient, with reference categories for categorical variables, and mean values for continuous variables (green), for a patient with poor histological response (orange), and a patient with poor histological response and lung metastases. Predictions are shown for a random forest with all parameters tuned (RFS 1), a random forest with all parameters but node size tuned (RFS 2), and a random forest with no parameters tuned (RFS 3).

Figure 4.14 shows the survival curves of all the patients from the validation set, as predicted by the RFS(1) model (1a,b), the RFS(2) model (2a,b), the RFS(3) model (3a,b) and the Cox model (4a,b). Predictions are classified on histological response (a) and presence of lung metastases (b). As described earlier, the fully tuned RSF(1) model displays clustered observations, while the RSF(2) and RSF(3) predictions are more widely spread, to a lesser and greater extent, respectively. When looking at the separation of patients on histological response and on the presence of lung metastases, it can be observed that the first two models show a stricter separation of classes, whereas for the RSF(3) model there is more overlap. Once more, as in the previous figure (Figure 4.13) the RSF(1) and RSF(2) model have smoother survival curves than the RSF(3) model, where the curves have a more choppy, stepwise pattern. When comparing the RSF models to the Cox model predictions (Figures 14.4a, 14.4b), the RSF(1) model (Figures 14.1a, 14.1b) deviates most from the Cox predictions.
Figure 4.14: Predicted survival curves random survival forest models and the Cox model. a) histological response, blue for a good histological response, red for a poor histological response; b) presence of lung metastases, blue for no lung metastases, red for the possible or definite presence of lung metastases. 1) RFS(1), a fully tuned model; 2) RFS(2), a model with all parameters but node size tuned; 3) RFS(3), an untuned model; 4) The Cox proportional hazards model
Figure 4.15 depicts the variable importance (a) and minimal depth (b) for the seven predictors. A relatively high variable importance (VIMP) and small minimal depth are indicative of a strong predictor (see also section 4.4.1). In both the VIMP and minimal depth plots there are two clear top predictors - histological response (green) followed by the presence of lung metastases (purple). VIMP identifies the next most important predictor for all three models as absolute tumour volume (yellow), while according to minimal depth this is only true for the RFS(2) and RFS(3) models. The next most important predictor is the presence of other metastases. The presence of tumour volume as the third/fourth most important variable is an interesting deviation from the the Cox model results. While the Cox model also identifies histological response (HR=2.509, 95%CI=2.02-3.12) and lung metastases (HR=2.621, 95% CI=2.09-3.29) as most important, absolute tumour volume has a HR of 1.000 (95% CI=1.00-1.00) (see also Table 3.9).
4.4.4 Discussion

In the previously described application Ishwaran’s Random Survival Forest (RSF) approach \cite{19} was used to construct a random forest model for the EURAMOS-1 osteosarcoma data. Five different error measures were considered: the concordance index (C-index), the Brier score, Kullback-Leibler score, the ensemble mortality hazard ratio and the $\log$-rank $\chi^2$-statistic. The C-index, Brier score and Kullback-Leibler score were used for formal model evaluation, with the C-index as main selection criterion. A cross-validation was performed to tune four parameters - $mtry$ (the number of candidate variables considered at each split), $nodesize$ (the average size of the terminal nodes), $nsplit$ (the number of split points selected per variable), and $nodedepth$ (the maximal depth of each node, in terms of branching). Three different splitting rules were considered - logrank, logrankscore and random, and the default logrank option was chosen. Ultimately, three final models were considered. When comparing the Cox proportional hazards model to the RSF models the Cox model is slightly better as measured by the C-index. No difference in the Brier and Kullback-Leibler scores is observed, save for the untuned RSF(3) model, which has slightly higher Kullback-Leibler score values.

1. A fully tuned model with $nodesize=200$, $mtry=2$, $nsplit=3$ and $nodedepth=7$.

2. A tuned model with nodesize unrestrained, $mtry=2$, $nsplit=3$ and $nodedepth=7$.

3. A untuned model with nodesize and nodedepth unrestrained, default $mtry=3$, default deterministic splitting with $nsplit=0$.

The fully tuned model was found through a five-fold cross-validation procedure. The second model was introduced on observing an unusually clustered pattern of predicted survival probabilities, associated with specifying a large node size. While the first model has the largest concordance index (C-index), the second model only has a marginally lower one, yet a drastic difference in prediction patterns can be observed. On closer observation of the behaviour of the individual parameters it was found that enforcing more stringent regularization results in clustering, and also in marginally higher C-indexes. The concordance index is a measure of discriminative ability that relies on comparing pairs of subjects, and counting how many times a subject has a comparatively higher ensemble mortality score (the random survival forest equivalent
to the prognostic index) matched by a comparatively smaller survival time. The C-index is invariant to scale or distance between observations, and relies only on order. As such, two vastly different prediction patterns can be associated with two very similar C-index, given that the ordering remains consistent. Barely any difference was observed in the Brier and Kullback-Leibler scores. Both scores are a measure for the deviation from observed survival probability to known status, as evaluated at $t_0$, with $y = 1 \{ t > t_0 \}$ for event/censoring time $t$, and covariate vector $x$. Then, taking as example the Brier score, with

$$Brier(y, \hat{S}(t_0|x)) = (y - \hat{S}(t_0|x))^2; \quad (4.12)$$

the absence of a distinctive change can be explained as follows. Increased clustering and overall higher survival probability estimates $\hat{S}(t_0|x)$ will reduce the error in the situation that $y = 1$, but increase the error in the situation that $y = 0$, as the calculated deviations will be lower and higher, respectively. The same holds for the Kullback-Leibler score. Altogether, this means that for very similar performance measures, very different predictions can be observed. Of course, the very next question to ask, is which predictions are the reliable ones. Under the assumption that predictions obtained from the Cox proportional hazards model are realistic, a more widespread distribution of survival curves would be appropriate, a condition which holds for the second and third models (all but node size tuned and untuned, respectively), but not for the fully tuned first model. An issue, however, is that the true survival probabilities are not known, and as such it cannot be said with surety that the Cox model predictions are superior. Yet it is suspect that in the random survival forest models minor C-index variations are accompanied by relatively large shifts in the prediction patterns.

In this chapter, a special effort was made to investigate the influence of individual parameters on random forest performance, and after completing a regular cross-validation procedure (Sections 4.4.2.1-4.4.2.3), the individual tuned parameters, and extreme valued parameters were examined (Section 4.4.2.4). The main observations are summarized and motivated below.

- More stringent regularization gives a small increase in performance as measured by the C-index, but results in clustering of the predicted survival curves. Regularization is enacted by parameters nodesize, nodedepth and mtry. nodesize and nodedepth are the most explicit way to influence tree growth.[17] (section 4.4.2.4)
- **nodesize** regulates the average size of the terminal nodes; the higher the specified value, the more observations are present in each terminal node, and the less branched the forest is. Increasing **nodesize** protects against overfitting. A higher **nodesize** was observed to result in a higher C-index and a more clustered prediction pattern.

- **nodedepth** specifies a limit to the depth of a node and halts the branching of a node when this limit is reached. Decreasing **nodedepth** will also reduce the branching of a forest, but enacts no constraint on the terminal node size. A smaller **nodedepth** was observed to result in a higher C-index and a more clustered prediction pattern.

- **mtry** regulates the number of candidate variables considered for a split-point. Setting **mtry** smaller than the total number of predictors introduces additional randomization, and distinguished the random forest from the bagging approach. An **mtry** of 1 was observed to result in a higher C-index and a more clustered prediction pattern.

- Specifying **nodesize** past a (data-specific) threshold will result in a model of no discriminative ability, which is apparent in a C-index of 0.5, but less so in the Bier and Kullback-Leibler scores, for which only a slight increase in value is observed. This was observed for **nodesizes** of 550 and 600. Considering that the training set has 1350 observations in total, specifying such a large node size will allow for very few terminal nodes, which will result in a forest not able to properly represent the data (section 4.4.2.1).

- A small **nodesize** has a poorer performance when combined with a high number of candidate splitting variables (**mtry**). This can likely be explained by overfitting due to the small **nodesize** and by insufficient randomization introduced by constraining the variable selection. (section 4.4.2.1)

- An untuned random forest has a lower performance than a tuned one but an acceptable one (C-index=0.67).

- The performance measures C-index, Brier score, and Kullback-Leibler score consistently report model performance. The ensemble mortality hazard ratio and logrank $\chi^2$-statistic do not. (section 4.4.2.4)
Higher variability in the ensemble mortality hazard ratio is associated with a small number of candidate split variables ($mtry$) and a small number of split-points ($nsplit$). (section 4.4.2.4)

A final issue to consider is the manner in which the generalization error was estimated in the tuning process - via 5-fold cross-validation. Cross-validation is a default choice for parameter tuning as it can be applied to any method. In the context of random forests however, the generalization error can also be estimated using out-of-bag (OOB) estimates. OOB data is a natural by-product of the random forest algorithm, which relies on drawing a $B$ number of bootstrap samples, growing a $B$ trees and then averaging results to obtain an ensemble estimate. OOB estimation is obtained by dropping a given observation $x$ down the trees in which $x$ is not used for training. In the 0.632 bootstrap, used in this application, that means that roughly one third of the trees can be used for each OOB prediction. This implies that as only a relatively small part of the forest can be used for estimation, some accuracy will be lost. Additionally, it should be considered that the distribution on which each tree is grown and the distribution of the OOB predictions are not identical. Some research has been published on the use of OOB estimates versus cross-validation and test set estimates in the context of classification and regression trees [50, 7], which has suggested a bias is present in the OOB error. A dependency between the $mtry$ parameter and the size of the bias has been described [50, 7], and overall it has been suggested that the OOB error overestimates the true generalization error [7]. For classification forests it was shown that class imbalance in the bootstrap samples contributes to the bias. This bias can be reduced by employing stratified sampling, which maintains the class distribution of the original data in each bootstrap sample. [50] Whether all this also applies to random survival forests has not been formally established, but should be considered. And so, while OOB estimation is very attractive from a computational viewpoint, in light of potential bias and in order to maintain consistency with the neural network approach in chapter 3, a cross-validation approach was chosen. Please note that in the simulation chapter OOB and cross-validated performance measures are both obtained.

Overall, a random survival forest approach gives a good performance for the EURAMOS osteosarcoma data, with concordance indexes very close to those of the Cox model, and Brier and Kullback-Leibler scores that are slightly better. There are however, several limitations associated with fitting a RSF model. Firstly, survival predictions for the test data can only be made for the event times observed in the
training data. In the application this is reflected in predictions that do not extend past six years after surgery. In the same way time-dependent performance measures, like the Brier and Kullback-Leibler score, can only be evaluated at these time points. A second concern is the variation observed in individual survival predictions, on varying parameter values - a quality which is not reflected in any of the performance measures. The extent to which this is an issue depends on the purpose of the random forest. As indicated by the high C-indexes associated with the clustered predictions, the relative ordering of patients is maintained, and as such the forest is suitable for dividing patients into risk groups and identifying predictors that may be missed in more restricted models. Such a random forest, however, is not reliable when it comes to individual patient predictions. A well-cited review article notes in the context of random forests that changes in variable importance and predictions indicate forest instability - an issue that can be resolved by increasing the number of trees.\textsuperscript{5} No remarks, however, are made on predictions that are stable within a given forest, but vary over forests with parameter tuning. Implementations of random (survival) forest methodology report model tuning and model evaluation using solely the C-index as a measure of performance, without considering performance in terms of individual survival prediction. Note however that there is no formal summary measure for quantifying the latter. An advantage of the random forest approach is that complex interactions and time-dependencies can be accommodated without a priori specification. In this context, it is of interest to note that the RSF models, through the VIMP and minimal depth measures, identified absolute tumour volume as the third strongest prediction (following histological response and the presence of lung metastases), whereas in the Cox model it is a non-significant predictor. While an overall indication of predictor importance can be obtained, there is no way to quantify the exact way a given predictor impacts survival probability.
Chapter 5

Simulation

In Chapters 3 and 4 the performance of neural networks and random survival forests, respectively, was examined on the EURAMOS osteosarcoma dataset. To gain more insight into model performance a simulation study is performed, in which the standard Cox model, the neural network approach and random survival forests approach are compared on a variety of performance measures. The objective is to simulate data that closely resembles the EURAMOS data, as such providing insight into the potential of different ML models for clinical datasets. The EURAMOS data is described in detail in Chapter 2. In this chapter first the process of simulating comparable data is described (Section 5.1), followed by a description of the cross-validation approach to obtain the relevant performance measures (Section 5.2). In Section 5.3 the simulation results are discussed, and in Section 5.4 a final comparison between the methods is given, as evaluated on the simulated data and the original EURAMOS data. Appendix E is an addendum to this chapter and gives the full and reproducible R code for each step of the simulation process.

5.1 Data simulation

The general approach to simulating data comparable to the EURAMOS osteosarcoma data is described in three steps below.

1. The original EURAMOS data is used to count the occurrence of unique combinations of categorical variables - histological response (good, poor), surgical excision (wide/radical, marginal, intralesional/other), sex (female, male), lung metastases (no, yes/possible), and other metastases (no, yes/possible). For each combination the means of the continuous variables age at registration and absolute tumour volume are determined from the original data. The combination-specific covariance matrices are
also obtained, in order to account for the possible correlation between age and volume. Table 5.1 gives an example of a reduced dataset with categorical covariates histological response (good, poor) and lung metastases (yes/possible, no), each with two levels, and continuous covariates age and volume. For each unique categorical covariate combination the proportion is calculated, as well as the mean age and mean volume. Additionally (not shown in the table), the covariance matrices for age and volume are retained too, to account for dependency across the two covariates.

Table 5.1: Unique categorical covariate combinations, proportions, and continuous covariate means for sample dataset

<table>
<thead>
<tr>
<th>Hist. resp</th>
<th>Lung mets</th>
<th>Proportion</th>
<th>Age (mean)</th>
<th>Volume (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Good hist</td>
<td>Yes/Possible</td>
<td>0.10</td>
<td>15.20</td>
<td>164.00</td>
</tr>
<tr>
<td>2 Good hist</td>
<td>No</td>
<td>0.40</td>
<td>13.60</td>
<td>120.00</td>
</tr>
<tr>
<td>3 Poor hist</td>
<td>Yes/Possible</td>
<td>0.35</td>
<td>14.80</td>
<td>161.00</td>
</tr>
<tr>
<td>4 Poor hist</td>
<td>No</td>
<td>0.15</td>
<td>14.90</td>
<td>155.00</td>
</tr>
</tbody>
</table>

2. Data (n=2000) is simulated by considering all possible combinations of categorical variables, and sampling combinations in accordance to proportion of occurrence in original data. Continuous variables age and volume are sampled from a multivariate normal distribution with mean, variance and covariance specified per categorical variable combination. Following the example given in Table 5.1, the combination Hist.resp:good, Lung mets:yes/possible would be sampled with proportion 0.10 for a total of 2000 draws, giving approx. 200 such observations. Age and volume would be sampled from a multivariate normal distribution with means 15.20 and 164.00, respectively, with the relevant covariance matrix (not shown).

3. Using the original dataset, coefficients for the covariates are obtained from a log-normal regression. The survival times are then given by

\[
\log(T) = \mu + \beta^T X + \sigma \epsilon, \epsilon \sim N(0, 1),
\]

where \( \mu \) is the intercept, \( \beta^T \) the estimated coefficient, \( X \) the covariate matrix for a given simulated dataset, \( \sigma \) the scale parameter, and \( \epsilon \) the error, normally distributed with mean 0 and variance 1. Censoring times are simulated from a uniform distribution, in a manner which mimics the censoring observed in the original dataset.
5.1.1 Covariate information from original EURAMOS osteosarcoma data

In order to obtain representative covariate values, the frequency of occurrence of all possible combinations of the categorical covariates was considered, and combinations observed 20 times or more selected. In the simulation they are sampled according to their proportion in the dataset, after correction for the disregarded low-frequency \((n < 20)\) combinations. The continuous covariates were simulated by drawing from a multivariate distribution using the mean and covariance matrix observed for the respective combinations in the original dataset. Frequencies were evaluated based on the data from 2238 of the original 2260 patients. 22 patients were disregarded due to disease-dependent non-randomization. For histological response and absolute tumour volume 212 (9.5%) and 379 (17%) missing values were observed, respectively. A total of 1673 out of 2238 cases (75%) were complete. A visual overview of the missing values is shown in Figure 5.1.

![Figure 5.1: Missing values for EURAMOS osteosarcoma data (n=2238). Observed values shown in red, missing values shown in white.](image)

In order to make the fullest use of the available data missing values for histological response and volume were imputed. Several restrictions were imposed. For absolute tumour volume, the measured values range from 0.0052 \(cm^3\) to 3952 \(cm^3\), of which 92% are 450 \(cm^3\) or lower. For age at registration, values range from 4 to 41 years. The distributions for the two variables are shown in Figure 5.2.
For both variables, the extremes were disregarded. Missingness and restrictions were both dealt with by multiple imputation. Values exceeding the predetermined cut-offs were removed from the dataset and replaced with NAs. 183 extreme values were removed for absolute tumour volume, and 107 values for age at registration. Multiple imputation (m=10) was used to impute reasonable values. For absolute tumour volume, imputation was bounded between the lowest observed volume 0.0052 cm³ and 450 cm³. For age, imputation was bounded between 5 and 25 years. The new distributions for volume and age are shown in Figure 5.3.

For histological response, the most frequent imputed value over the 10 imputations was selected as a final value. The frequencies of all observed covariate combinations are shown in Figure 5.3.
were counted, and the combinations occurring 20 or more times were selected, given
a total of 18 combinations. Means and covariance matrices for volume and age were
calculated for each combination. Values from the imputed dataset were pooled using
Rubins rule (described in Section 2.2.4)[49], as were the covariance matrices. In the
original data, a correlation of 0.11 is observed between absolute tumour volume and
age at registration. The overall pooled correlation for the imputed datasets is slightly
smaller at 0.85. For 1000 simulated datasets a mean correlation of 0.108 (sd=0.023)
is found, comparable to the correlation in the original data.

Below, the most frequent (N ≥ 20) combinations of covariates are shown, assessed
post-imputation, with corresponding means for volume and age. Notably, the cate-
gory yes/possible for the covariate other metastases is absent. As such this covariate
was removed from the analysis.
Table 5.2: Unique combinations of categorical covariates histological response (Hist), surgical excision (Ex), sex (Sex), lung metastases (Lung), other metastases (Other) observed ≥ 20. Counts and proportions of occurrence are given, and the means of continuous covariates age and volume per unique combination.

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>prop</th>
<th>Hist</th>
<th>Ex</th>
<th>Sex</th>
<th>Lung</th>
<th>Other</th>
<th>Volume</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>403.00</td>
<td>0.19</td>
<td>Good hist</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>133.90</td>
<td>14.60</td>
</tr>
<tr>
<td>2</td>
<td>371.00</td>
<td>0.18</td>
<td>Poor hist</td>
<td>Radical/wide</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>148.50</td>
<td>15.20</td>
</tr>
<tr>
<td>3</td>
<td>292.00</td>
<td>0.14</td>
<td>Good hist</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>118.70</td>
<td>13.60</td>
</tr>
<tr>
<td>4</td>
<td>259.00</td>
<td>0.12</td>
<td>Poor hist</td>
<td>Radical/wide</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>131.20</td>
<td>13.60</td>
</tr>
<tr>
<td>5</td>
<td>96.00</td>
<td>0.05</td>
<td>Good hist</td>
<td>Radical/wide</td>
<td>Male</td>
<td>Yes/Possible</td>
<td>No</td>
<td>168.70</td>
<td>14.90</td>
</tr>
<tr>
<td>6</td>
<td>88.00</td>
<td>0.04</td>
<td>Poor hist</td>
<td>Radical/wide</td>
<td>Male</td>
<td>Yes/Possible</td>
<td>No</td>
<td>160.80</td>
<td>14.80</td>
</tr>
<tr>
<td>7</td>
<td>67.00</td>
<td>0.03</td>
<td>Good hist</td>
<td>Other</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>132.30</td>
<td>15.10</td>
</tr>
<tr>
<td>8</td>
<td>63.00</td>
<td>0.03</td>
<td>Poor hist</td>
<td>Marginal</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>162.70</td>
<td>15.70</td>
</tr>
<tr>
<td>9</td>
<td>57.00</td>
<td>0.03</td>
<td>Good hist</td>
<td>Radical/wide</td>
<td>Female</td>
<td>Yes/Possible</td>
<td>No</td>
<td>149.50</td>
<td>12.80</td>
</tr>
<tr>
<td>10</td>
<td>56.00</td>
<td>0.03</td>
<td>Poor hist</td>
<td>Radical/wide</td>
<td>Female</td>
<td>Yes/Possible</td>
<td>No</td>
<td>141.50</td>
<td>14.00</td>
</tr>
<tr>
<td>11</td>
<td>54.00</td>
<td>0.03</td>
<td>Poor hist</td>
<td>Marginal</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>128.30</td>
<td>13.80</td>
</tr>
<tr>
<td>12</td>
<td>51.00</td>
<td>0.03</td>
<td>Poor hist</td>
<td>Other</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>149.10</td>
<td>15.10</td>
</tr>
<tr>
<td>13</td>
<td>46.00</td>
<td>0.02</td>
<td>Poor hist</td>
<td>Other</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>133.40</td>
<td>12.90</td>
</tr>
<tr>
<td>14</td>
<td>44.00</td>
<td>0.02</td>
<td>Good hist</td>
<td>Other</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>148.60</td>
<td>13.20</td>
</tr>
<tr>
<td>15</td>
<td>43.00</td>
<td>0.02</td>
<td>Good hist</td>
<td>Marginal</td>
<td>Male</td>
<td>No</td>
<td>No</td>
<td>165.30</td>
<td>14.10</td>
</tr>
<tr>
<td>16</td>
<td>42.00</td>
<td>0.02</td>
<td>Good hist</td>
<td>Marginal</td>
<td>Female</td>
<td>No</td>
<td>No</td>
<td>139.10</td>
<td>13.70</td>
</tr>
<tr>
<td>17</td>
<td>21.00</td>
<td>0.01</td>
<td>Good hist</td>
<td>Other</td>
<td>Male</td>
<td>Yes/Possible</td>
<td>No</td>
<td>177.20</td>
<td>15.40</td>
</tr>
<tr>
<td>18</td>
<td>20.00</td>
<td>0.01</td>
<td>Good hist</td>
<td>Marginal</td>
<td>Male</td>
<td>Yes/Possible</td>
<td>No</td>
<td>200.50</td>
<td>14.90</td>
</tr>
</tbody>
</table>
5.1.2 Simulation of covariate data

The data simulation is performed based on the information gathered in Table 5.2. As detailed previously, this table contains 18 unique categorical covariate combinations which were observed with a frequency of $\geq 20$ in the original dataset. The variable other metastases is disregarded as only one factor level (No) is observed. For each unique categorical covariate combination the proportion of occurrence is given, as well as the corresponding means for age and volume. Note that while combination-specific covariance matrices are used, they have not been included in the table. Below, the final function for simulating covariate data is shown, followed by a detailed description of the various steps.

```r
simulation <- function(NN){
  # NN observations are drawn from sample.list
  sim.dat <- sample(sample.list,NN, replace=TRUE, prop)
  new.data <- NULL

  for (i in 1:length(sample.list)){
    # number of times combination occurs (N)
    index.dat <- which(sim.dat==sample.list[i])
    N <- length(index.dat)

    # relevant sample.dat entry selected and repeated N times
    new.dat <- rep(sample.dat[i,], N)

    # Selecting relevant means and covariance matrix
    index.m.v <- i
    vv <- vars[[index.m.v]]
    mm <- means[index.m.v,]

    # values for volume and age are drawn from a multivariate
    # normal distribution.
    sims <- as.data.frame(mvrnorm(N, mm, vv))
    sims[which(sims[,1]<0),] <- 0.0052
    new.dat <- cbind(new.dat, sims)
    new.data <- rbind(new.data, new.dat)
  }
  return(new.data)
}
```

# Appendix E. 1.2 (1−30)

135
The function *simulation* takes as argument the number of entries to be simulated \( NN \). \( NN \) observations are drawn from *sample.list*, which is a vector of 18 names representing all possible unique categorical covariate combinations. They are sampled with proportions \( prop \) as observed in the original data. The number of times \( N \) each unique combination has been drawn is counted. Then, the relevant *sample.dat* entry is selected. *sample.dat* is a dataframe with the same information as *sample.list*. Rather than names for each combination, *sample.dat* has a column per covariate. The relevant entry is selected and repeated \( N \) times- the number of times the unique combination was sampled. The means *means* and covariance matrices *vars* for age and volume, which correspond to each of the unique combinations of *sample.list* and *sample.dat* were created previously. For each unique combination the correct means and covariance matrix for the two variables are selected, and \( N \) values for volume and age are drawn from a multivariate normal distribution. As volume has a comparatively large variance, same values fall below zero - these are set equal to the minimum observed to the original dataset. This process is repeated for every unique covariate combination \( i \), ultimately resulting in a simulated dataset of \( NN \) entries. The full code for performing the steps prior to the definition of the simulation function are given in Appendix E.1.1. Note, however, that this function can be applied to any given dataset with a given number of categorical covariates and, in its current definition, two correlated continuous covariates. The necessary steps preceding the use of the simulation function consist of defining a *sample.list* object (a vector with names representing the possible covariate combinations), a *sample.dat* object (a dataframe with every row corresponding to a name in *sample list*), *means* (a dataframe with the continuous covariate means) and *vars* (a list with the covariance matrices). These objects can be specified manually, or, as done here, obtained from an existing dataset. In the latter case, the process of creating the objects will be largely data specific.

5.1.3 Simulation of survival and censoring times

Survival times were simulated by applying the following formula (5.2), where \( \beta^T \) is the vector of parameter estimates obtained from a lognormal regression on the original data, \( X \) is the covariate matrix of a simulated dataset, \( \sigma \) is the scale parameter, and \( \epsilon \) is the error, normally distributed with mean 0 and variance 1.

\[
\log(T) = \mu + \beta^T X + \sigma \epsilon \sim N(0, 1),
\]  

(5.2)
2025 patients of the original \( n = 2260 \) EURAMOS osteosarcoma data were used to obtain the parameter estimates. The sample size is reduced from the one used to estimate the covariate distribution in the previous section, as now, by necessity only patients with a follow-up after surgery are included. Patients with disease-dependent non-randomization, no registered surgery data, and no follow-up after surgery are excluded. Missing values were imputed using 10-fold multiple imputation, and parameter estimates were pooled using Rubin’s rule\[49\]. In the function shown below a log-normal regression is applied to each imputed dataset.

```r
lognorm.func <- function(imp.datasets){
  lognormlist <- list()
  for(i in 1:length(imp.datasets)){
    data <- imp.datasets[[i]]
    lognorm.obj <- survreg(Surv(time.death,stat.death) ~
                           age.new+ sex+GOOD.POOR. t+new.excis+new.lmetreg+
                           volume, data=data, dist="lognormal")
    name <- paste(i)
    lognormlist[[name]] <- list(lognorm.obj=lognorm.obj)
  }
  return(lognormlist)
}
```

In the following code excerpt the function for pooling the estimates is shown. Per Rubin’s rule, the pooled coefficient estimate is obtained by simply taking the mean over the individual estimates (\( mcoef \)). The pooled covariance estimate is obtained by calculating two components - the within imputation variability (\( withinVAR \)), which is the average over the covariance matrices, and the between imputation variability (\( betweenVAR \)), which measures the deviation of the individual estimates to the grand mean (see also Section 2.2.4). The pooled parameter estimates are shown in Table 5.3. Here, \( mu \) represents the intercept \( \mu \) in equation (5.2), the exponent of \( log(sigma) \) the scale parameter \( \sigma \) and the remaining coefficient estimates \( \beta^T \).
pool.lognorm <- function(lognorm.list){

  # obtain pooled coefficients
  coefs <- lapply(1:10, function(i) {
    c(lognorm.list[[i]]$lognorm.obj$coefficients, 
      log(as.numeric(lognorm.list[[i]]$lognorm.obj$scale)))
  })
  coefs <- data.frame(matrix(unlist(coefs), 
                             ncol=10, byrow=FALSE), 
                      stringsAsFactors=FALSE)
  mcoef <- apply(coefs, 1, mean)

  # obtain covariance matrices
  m <- length(lognorm.list)
  vars <- lapply(1:10, function(i) (vcov(lognorm.list[[i]]$lognorm.obj))

  # obtain within, between and total variance
  withinVAR <- Reduce('+', vars)/m
  betweenVAR <- lapply(coefs, function(x) (x-mcoef) %*% (t(x-mcoef)))
  betweenVAR <- Reduce('+', betweenVAR)/(m-1)
  totalVAR <- withinVAR + ((m+1)/m)*betweenVAR

  # return pooled standard errors and coefficients
  totalSE <- sqrt(diag(totalVAR))
  new.pars <- cbind(mcoef, totalSE)
  return(new.pars) # Appendix E.1.3 (222–263)
}

Table 5.3: Parameter estimates from log-normal regression on EURAMOS osteosarcoma data (N=2025)

<table>
<thead>
<tr>
<th>Coef</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>mu</td>
<td>3.60</td>
</tr>
<tr>
<td>Age</td>
<td>-0.00</td>
</tr>
<tr>
<td>Sex (male)</td>
<td>-0.19</td>
</tr>
<tr>
<td>Hist. resp. (poor)</td>
<td>-0.94</td>
</tr>
<tr>
<td>Ex (marginal)</td>
<td>0.03</td>
</tr>
<tr>
<td>Ex (intralesional/other)</td>
<td>-0.45</td>
</tr>
<tr>
<td>Lung met (Yes/possible)</td>
<td>-1.16</td>
</tr>
<tr>
<td>Volume</td>
<td>-0.00</td>
</tr>
<tr>
<td>log(sigma)</td>
<td>0.44</td>
</tr>
</tbody>
</table>
Below, the final simulation function is shown. The function takes as argument the desired data size \( NN \), and the coefficient estimates \( \text{est.coefs} \), intercept \( mu \) and scaling parameter \( \sigma \) obtained from the lognormal regression on the original EURAMOS osteosarcoma data. Please note that the coefficient estimates \( \text{est.coefs} \) should be in the same order as those of the matrix object \( \text{mat} \) (as shown in Table 1.3). In the first step the function \text{simulation()} \) is used to create a covariate dataset \( \text{sim} \), in turn used to create the object \( \text{mat} \). Survival times are then calculated using the formula (5.2), with \( mu \) for \( \mu \), \( \text{est.coefs} \) for \( \beta^T \), the covariate matrix \( \text{mat} \) for \( X \), and \( \sigma \) for \( \sigma \). The error \( \epsilon \) is drawn from a normal distribution with mean 0 and variance 1 using \text{rnorm()}. Censoring is simulated from a uniform distribution: \( U(1,11) \).

```r
# Function for simulating survival time and data.
# Takes as argument data size (NN), covariate coefficients
# (est.coefs), intercept (mu), and scaling parameter (sigma)
sim.surv <- function(NN, est.coefs, mu, sigma) {

  # A dataset of NN rows is simulated using the previously defined function 'simulation'
  sim <- simulation(NN)

  # An indicator matrix is created for the categorical covariates
  mat <- model.matrix(~Sex + Hist + Ex + lung, sim[,1:4])[,,-1]

  # To this matrix age and volume are added
  mat <- cbind(sim$V2, mat, sim$V1)
  colnames(mat) <- c("Age", colnames(mat[,2:6]), "Vol")

  # Survival time is calculated using equation (3.1)
  Time <- exp(mu + mat %*% est.coefs + sigma*rnorm(NN,0,1))

  # Censoring times are simulated from uniform distribution
  # U(1,11)
  C <- runif(NN,1,11)
  Status <- as.numeric(Time <= C)
  Time.c <- pmin(Time,C)
  id <- seq(1,NN)
  full <- cbind(sim, Time, Status, Time.c, id)
  return(full)
}
```

# Appendix E.1.3 (269–293)
Note that like the previously described covariate simulation function \texttt{simulation()}, the \texttt{sim.surv()} function can easily be adapted to other datasets. Depending on the data created with \texttt{simulation()}, the \texttt{mat} object would need to be modified to represent potentially different covariates. The necessary coefficients (\texttt{est.coefs}, \texttt{mu}, \texttt{sigma}) can be specified manually, and the survival time formula itself can easily be modified. The distribution the censoring times are drawn from can also be changed and the parameters adjusted. In this particular application censoring times are drawn from a uniform distribution with minimum and maximum values of 1 and 11, respectively, with the purpose of simulating a censoring rate of approximately 73%. Figure 5.4a shows the censoring distribution of the original data as estimated using reverse Kaplan Meier. In Figure 5.4b the censoring distribution of a given dataset simulated with the \texttt{sim.surv()} function is shown. Censoring occurs somewhat earlier in the simulated dataset, but overall the distributions are comparable. Figure 5.4c shows the distribution of the censoring percentage for 1000 simulated dataset, which has a mean close to 73%.

Figure 5.4: Censoring distributions for original data and imputed data. a) censoring distribution EURAMOS osteosarcoma data (n=2238). b) censoring distribution for a given imputed dataset. c) Percentage censoring over 1000 imputed datasets

Figure 5.5 shows histograms of the observed and simulated survival times, in the presence of censoring. In 5.5a the distribution of survival/censoring times for the original dataset is shown. In Figure 5.5b the covariate matrix of the original data has been used to calculate new survival times using equation (5.2) with uniform censoring \texttt{U(1,11)}. Compared to the observed times in 5.5a, the survival times in 5.5b have a wider distribution, ranging from approximately 0 to 11 years with a considerable number of observations at both extremes. In contrast, the observed survival times in 5.5.a do not stretch past 9 years and have very few observations on the upper extreme.
Figure 5.5c shows the survival time distribution for a given simulated dataset. It is comparable with the distribution in Figure 5.5b, indicating that the simulated covariate matrices are a good approximation of the original EURAMOS osteosarcoma data.

Figure 5.5: Survival distributions for original data and imputed data. a) survival distribution EURAMOS osteosarcoma data (n=2238). b) survival distribution as calculated by equation (3.1), using the predictor matrix from the original data. c) survival distribution as calculated by equation (3.1), using the predictor matrix from a given imputed dataset.
5.2 Cross-validation approach and functions

A leave-1%-out cross-validation is used to estimate the generalization error for \( S = 1000 \) simulations of \( N = 2000 \) simulated datasets. Three methods are considered—the Cox proportional hazards model; a neural network model, as described by Biganzoli in his PLANN approach \cite{10}; and a random survival forests model, as described by Ishwaran. \cite{19}. For each method the survival probabilities are obtained, and, for the Cox model and the random survival forest model the prognostic index and ensemble mortality, respectively. In the cross-validation iteratively test sets of 1% of the 2000 subjects contained in the data (20 entries each time) are selected. The remaining data is used to train the relevant model. Then, the survival probabilities and/or the prognostic index/ensemble mortality scores are calculated for the test set. Having obtained the relevant cross-validated quantities for all the data, performance measures are calculated. The concordance index is calculated from the observed survival times, censoring status, and the prognostic index/ensemble mortality. The Brier and Kullback-Leibler scores are calculated from the predicted survival probabilities and censoring status, with inverse probability censoring weighting. For the Cox and random survival forest models, for which a prognostic index and ensemble mortality score, respectively, can be obtained, two additional measures are considered. First, the hazard ratio as resulting from a regular Cox regression of the normalized prognostic index/ensemble mortality. Second, the logrank \( \chi^2 \)-statistic, which quantifies the survival difference between high and low risk patients, where high risk is defined as a prognostic index/ensemble mortality equal to or greater than the median.

In the following sections, some notes are made on the details of the cross-validation approach for each of the methods separately. Additionally, for each method the fully annotated cross-validation function is given.

5.2.1 Cox proportional hazards model

Below, the function for applying the Cox proportional hazards model to a given dataset, and obtaining performance measure estimates is given. In the first step an indicator vector \( \text{select.ind} \) is made, where values from 1 to 100 are sampled 20 times each. As such, each unique entry of \( \text{select.ind} \), contained in \( \text{unique.ind} \), represents 1% of all data entries. Two empty objects are created - \( \text{CVPI} \), to hold the cross-validated prognostic index, and \( \text{Pred.mat} \), to hold the cross-validated survival probability estimates. An iterative process is used to obtain estimates for each test set, identified with the index \( \text{unique.ind}[i] \). In each iteration, the Cox model \( \text{cm} \) is trained on all
data but the test set. The test set is then used to construct a model matrix \textit{mm}. The prognostic index for the test set is calculated by taking the product of \textit{mm} and the model coefficients \textit{cmcoefficients}, while the survival probabilities are obtained via the \textit{predictSurvProb()} function. The quantities are assigned to the \textit{CVPI} and \textit{Pred.mat} objects, respectively.

```r
# Final.Cox function, part 1:

final.Cox <- function(data){
  # indicator vector for leave 1% out cross-validation
  select.ind <- sample(rep(1:100,20),nrow(data),
                        replace=FALSE)
  unique.ind <- sort(unique(select.ind))
  data <- cbind(data,select.ind)
  CVPI <- rep(NA,nrow(data))

  # unique times for prediction
  times <- sort(unique(data$time))
  Pred.mat <- matrix(NA, nrow(data), length(times))

  # model - formula necessary
  fit <- Coxph(Surv(time,stat) ~ Hist + Ex + Sex + lung +
               Volume + Age, data=data)

  # performed for each leave out test set
  for(i in 1:length(unique.ind)){
    # index test set
    ind <- unique.ind[i]
    # fitting model on all but test set
    cm <- Coxph(fit$formula, data=data[data$select.ind!=ind,])
    # creating model.matrix for test set
    mm <- model.matrix(fit$formula,
                       data=data[data$select.ind==ind,])
    mm <- mm[,!(colnames(mm) %in% c("(Intercept)"))]
    # calculating prognostic index for test.set
    CVPI[data$select.ind==ind] <- mm %*% cm$coefficients
    # calculating predictions for all unique times
    Pred.mat[data$select.ind==ind,] <- predictSurvProb(cm,
                                                      data[data$select.ind==ind,,]
                                                      , times=times)
  }
}
```
The concordance index $Cox.C$ is calculated from the prognostic index, the survival time and censoring status, here collected in object $Cox.dat$. The Brier and Kullback-Leibler scores are calculated from the survival probability matrix ($Pred.mat$) and the survival time and status (contained in $Cox.dat$). The hazard ratio $HR$ and logrank $\chi^2$-statistic $LRX$ are both estimated from the prognostic index $CVPI$. The concordance index and Brier/Kullback-Leibler score functions are provided alongside the $final.Cox()$ function in Appendix E.2.1.

```r
# Final . Cox function , part 2 :
# making dataframe with survival time , status and # prognostic index
Cox.dat <- as.data.frame(cbind(data$time, data$stat, CVPI))
colnames(Cox.dat) <- c("surv", "stat", "pred")

# calculating C–index and brier/KL scores
Cox.C <- cindex.mod.r(Cox.dat)
Cox.BrierKL <- Brier.KL.Cox(Cox.dat, Pred.mat)

# calculating hazard ratio from normalized # prognostic index
CVPI.norm <- (CVPI - min(CVPI))/(max(CVPI) - min(CVPI))
Cox.cv <- Coxph(Surv(time, stat) ~ CVPI.norm, data=cbind(data, CVPI.norm))
HR <- exp(Cox.cv$coefficients)

# obtaining logrank Chisq statistic using the median # prognostic index
group <- rep(1, nrow(data))
group[which(CVPI<=median(CVPI))] <- 2
group <- as.factor(group)
temp.dat <- cbind(data, group)
LRX <- survdiff(Surv(time, stat)~group, data=temp.dat)$chisq

return(out)
}
# Appendix E.2.1 (1–65)
```
5.2.2 Random Survival Forests

As detailed extensively in chapter 4, random survival forests have several hyperparameters that can be tuned. In the rfsrc() function available from the randomforestRSC package these parameters are known as mtry, nsplit, nodedepth, and nodesize. mtry refers to the number of candidate variables for each node split, randomly selected from the full variable set, and nsplit refers to the number of split points considered on a given variable. The parameter nodedepth controls the maximum depth to which a tree is grown. Finally, nodesize controls the final branching of the tree, setting an average nodesize across the forest. All four of these parameters were tuned in the cross-validation described in section 4.4.2. Ultimately, a model with mtry=2, nsplit=3 and nodedepth=7 was selected, with a logrank splitting rule. The nodesize parameter was left unrestrained, as an unusual clustered prediction pattern was observed when specifying the tuned nodesize (section 4.4.3).

In the context of evaluating model performance on simulated data it would be ideal to perform a separate cross-validation for every single simulated dataset to ensure optimal model performance. Given, however, that this would be computationally very intensive, and given that the simulation is modelled after the EURAMOS osteosarcoma data, the parameter values identified in chapter 4 are used. Previously, in sections 4.2 and 4.4, note was made of the existing OOB error versus test error debate. In section 4.4.2 the choice of using the cross-validation error was motivated with two main arguments: 1) The OOB error has been shown to be biased, and in a classification/regression context has been said to overestimate the true generalization error [7, 50]; 2) Using the cross-validation error is consistent with the approach used in evaluating neural networks in chapter 3. In the simulation context, these two arguments still hold, and as such the leave-1%-out cross-validation, which is used to assess Cox model performance and neural network performance, is also used for the random forest model. For the purpose of gaining insight, however, into the differences between OOB error and cross-validation error, the OOB error will also be obtained.

Below, the process of obtaining the cross-validated survival probabilities and the ensemble mortality scores is described. Of note is that a modification is introduced to the approach described for the Cox model in the previous section is necessary. As detailed extensively in Chapter 4, a random forest model is only capable of predicting survival probabilities for the unique event times observed in the training set. This means that in every iteration of the cross-validation, slightly different prediction times will be considered, making it impossible to obtain a prediction matrix with survival
probabilities for the exact same times. Indeed, ultimately, there is not a single time
point that is common to each of the test set predictions. This problem is resolved by
considering predictions at a limited set of time points, and replacing each time point
of interest with the closest observed time.

```r
# RSF cross-validation function, part 1
final.rand <- function(data){

  # indicator vector for leave 10% out cross-validation
  select.ind <- sample(rep(1:100,20),nrow(data),
    replace=FALSE)

  unique.ind <- sort(unique(select.ind))
  data <- cbind(data, select.ind)
  prog.index <- rep(NA,nrow(data))
  cols <- length(which(data$stat==1))
  pred.list <- list(); time.list <- list(); pat.list <- list()

  # performed for each fold
  for(i in 1:length(unique.ind)){
    # index test set
    ind <- unique.ind[i]
    # fitting model on all but test set
    rfs <- rfsr(Surv(time, stat) ~
      Hist + Age + Volume + Ex +
      Sex + lung, data = data[data$select.ind!=ind,],
      tree.err = FALSE, importance = FALSE, ntree=750)

    # predicting on test set
    pred <- predict.rfsr(rfs,
      newdata = data[data$select.ind==ind,])

    # extracting prediction matrix
    pred.s <- pred$survival
    used.times <- pred$time.interest

    # creating ensemble mortality index
    index <- pred$predicted
    prog.index[data$select.ind==ind] <- index

    # saving predictions (pred.list), along with prediction
    # times (used.times), and survival time (pat.list)
    time.list[[i]] <- used.times
    pred.list[[i]] <- pred.s
    pat.list[[i]] <- data$time[data$select.ind==ind]
  }
}
```

In an initial step, given in the code above, the cross-validated predicted survival probabilities are gathered in a list \( \text{pred.list} \), where each element of the list contains the predictions for each cross-validation fold. As a leave-1%-out cross-validation is performed for a dataset of \( N = 2000 \), each list element has 20 observations, with a total of 100 elements in the list. Simultaneously, another list is created, which for each fold contains the exact times of prediction. In order to obtain the survival probabilities at \( t = 1, 2, \ldots, 10 \) years after surgery, for each list element \( \text{pred.list}[i] \) the predictions closest to these times are selected. It is important to note that this step is taken, as it introduces a measure of uncertainty to the predicted survival probabilities, and to the quantities calculated from these predictions. The product is a matrix \( \text{pred.mat} \) of dimensions \( N \times 10 \), with \( N = 2000 \). Each row contains the survival probabilities of a single individual at approximately the previously specified time points.

```r
# RSF cross-validation function, part 2

# establishing location of 1 to 10 year predictions
new <- lapply(time.list, function(x) {
  c(which.min(abs(x-1)), which.min(abs(x-2)),
    which.min(abs(x-3)), which.min(abs(x-4)),
    which.min(abs(x-5)), which.min(abs(x-6)),
    which.min(abs(x-7)), which.min(abs(x-8)),
    which.min(abs(x-9)), which.min(abs(x-10)))})

# selecting correct predictions
pred.vals <- list()
for (i in 1:length(pred.list)){
  pred.vals[[i]] <- pred.list[[i]][,new[[i]]]
}
pred.mat <- do.call("rbind", pred.vals)

# sorting predictions on patient according to true survival time
pred.mat <- as.data.frame(cbind(pred.mat, unlist(pat.list)))
pred.mat <- pred.mat[order(pred.mat$V11),]

# removing sorting column
pred.mat <- pred.mat[,1:10]
```

Below, the five performance measures are calculated. The Brier and Kullback-
Leibler scores are calculated using the $N \times 10$ prediction matrix `pred.mat` calculated previously, the event status `stat`, and the observed survival times `surv`. The C-index is calculated from the ensemble mortality `pred`, the event status and the observed survival times. The hazard ratio and \textit{logrank}$\chi^2$-statistic are both calculated from the ensemble mortality scores.

```r
# RSF cross-validation function, part 3

# making dataframe with survival time, status and
# prognostic index
rand.dat <- as.data.frame(cbind(data$time, data$stat, prog.index))
colnames(rand.dat) <- c("surv", "stat", "pred")

# calculating C-index and brier/KL scores
rand.C <- cindex.mod.r(rand.dat)
rand.BrierKL <- brier.function.rf(rand.dat, pred.mat)

# calculating hazard ratio from normalized prognostic index
prog.index.norm <- (prog.index - min(prog.index)) / (max(prog.index) - min(prog.index))
rand.cv <- Coxph(Surv(time, stat) ~ prog.index.norm, data=cbind(data, prog.index.norm))
HR <- exp(rand.cv$coefficients)

# obtaining logrank Chisq statistic based on median
# prognostic index
group <- rep(1, nrow(data))
group[which(prog.index<=median(prog.index))] <- 2
group <- as.factor(group); temp.dat <- cbind(data, group)
LRX <- survdiff(Surv(time, stat) ~ group, data=temp.dat)$chisq


return(out)
```

When calculating the Brier and Kullback-Leibler scores the function used is by necessity slightly different than the function previously described for random survival forest models (Section 4.4.1). Note that the latter function in itself is a modified version of the one described for the neural network model (Section 3.4.2). The necessity
for adapting the function springs from the previously discussed limitation of random survival forests - predictions can only be made for event times observed in the training set. In Section 4.4.1 a five-fold cross-validation was performed with the objective of tuning parameters. There, Brier and Kullback-Leibler scores were computed for each fold individually, using the mean value and standard deviation across the folds to gain insight into performance and stability. While the exact event times vary across the folds, they are the same within, and as such, with careful application of inverse censoring probability weighting, the relevant scores can be obtained.

In this instance, the objective is not to obtain a separate score for each fold, but to calculate the score for the overall cross-validated survival probabilities. As described previously, the issue with cross-validation is that in every fold approximately 1% of the observations will not contribute to the event time pool. Repeating this 100 times ultimately means that not a single event time is present in all folds - that there is no intersect of common event times. This is resolved by selecting predictions for each fold with event times as close as possible to the time of interest. In the context of the score calculations the times of interest are then defined as \( t_0 = 1, 2, \ldots, 10 \) years. As a score is calculated for the cross-validated survival probabilities on the whole, rather than per fold, a single censoring distribution, taken from the entire data, can be used to weight the observations. This is in contrast to the approach detailed in 4.4.1, where the censoring distribution at times \( t_0 \) (the observed event times) was taken from the training data, and the censoring distribution for the test observations at times \( t \) was taken from the test data. In the function below the full data censoring distribution \( csurv \) is obtained. Then, for the censoring weights at \( t_0 \) \( csurv.tt \) is created. Observations with a censoring event time after a given \( t_0 \) are weighted with the relevant \( csurv.tt \) entry, while observations with an event time prior to a given \( t_0 \) are weighted with the relevant \( csurv \) entry. For detail on weighting theory see Section 4.4.1.
brier.function.rf <- function(stat.t, pred.matrix){

  # censoring distribution for entire data is obtained
  object <- stat.t; so <- Surv(object$surv, object$stat)
  time <- so[,1]; ot <- order(time)
  cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)
  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf

  # censoring distribution at times t=1,2,...,10 is obtained
  index.1 <- list()
  for (i in 1:length(1:10)){
    index.1[[i]] <- which.min(abs(time-i))
  }
  csurv.tt <- csurv[unlist(index.1)]

  # times at which scores are evaluated and predictions
  btime <- seq(1,10)
  survs <- t(as.matrix(pred.matrix))

  bsc <- rep(0, length(btime))
  bsk <- rep(0, length(btime))
  for (j in 1:length(btime)) {
    help1 <- as.integer(time <= btime[j] & cens == 1)
    help2 <- as.integer(time > btime[j])
    bsk[j] <- -mean((log(1-(survs[j,]))*help1*(1/csurv) + log(surv[j,])*(1/csurv.tt[j]) *help2))
  }
  RET <- rbind(bsc, bsk)
  return(RET)
}

# Appendix E.2.2 (98–134)

As noted previously, for the purpose of comparing the cross-validated error with the OOB error, the latter is also obtained for the simulated dataset. The function is given on the next page. Notably, as OOB data is a natural byproduct of the random forest process, in order to obtain the error for a given dataset a model needs only to be fitted once, making for a markedly less computationally intensive function that the previously discussed final.rand() function that iterates over 10 folds. Performance measures are calculated as before. Note that the Brier and Kullback-Leibler scores are obtained with the function described in section 4.4.1. In contrast to the leave-1%-
cross-validation approach described previously, the OOB-predictions for each subject are obtained for the exact same event times, making a rounding step unnecessary. However, in contrast to the cross-validation approach of chapter 4, the censoring distribution used for weighting is calculated from the full dataset.

```r
new.func.rand <- function(data){
  # fitting model
  rfs.final <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung, data = data, nsplit = 3,
                     splitrule="logrank", mtry=2, nodedsize=NULL,
                     nodedepth=7, tree.err = TRUE, importance = TRUE,
                     ntree=750)

  # obtaining OOB survival and ensemble mortality
  surv <- rfs.final$survival.oob
  prog <- rfs.final$predicted.oob

  # obtaining survival time and censoring status
  stat.mat <- rfs.final$yvar
  stat.mat <- as.data.frame(cbind(stat.mat, prog))
  colnames(stat.mat) <- c("surv", "stat", "pred")

  # calculating C–index and Brier/Kullback–Leibler score
  C.ind <- cindex.mod.r(stat.mat)
  BR <- brier.function.r.mod(rfs.final, stat.mat, surv)

  # calculating hazard ratio
  pred.scores.norm <- (prog - min(prog))/
                      (max(prog) - min(prog))
  HR <- exp(Coxph(Surv(surv, stat) ~ pred.scores.norm,
                   data=cbind(stat.mat, pred.scores.norm))$coefficients)

  # obtaining logrank Chisq statistic
  group <- rep(1, nrow(stat.mat));
  group[which(prog<=median(prog))] <- 2
  group <- as.factor(group)
  temp.dat <- cbind(stat.mat, group)
  LRX <- survdiff(Surv(surv, stat) ~ group,
                  data=temp.dat)$chisq

  out <- c(C.ind, HR, LRX, BR[1,], BR[2,])
  return(out)
}
```

#Appendix E.2.2 (138–179)
5.2.3 Neural network

As described in Chapter 3, in order to fit a neural network the number of hidden nodes, the weight decay parameter, and activation functions must be specified. The final neural network model (Section 3.4.4) is a feed forward network with one hidden layer, 5 nodes in the hidden layer, two bias nodes (for the input and hidden layer, respectively) and a weight decay parameter of 0.05. Both activations functions are logistic ones, as recommended by Biganzoli.[10] The optimal number of nodes and optimal decay value were identified by means of cross-validation. As described in Section 3.4.1, in order to fit a neural network a data transformation is necessary, with slightly different versions for the training and test set. The functions used here have been adapted to the format of the simulated data (see Appendix E.2.3 (5-107)). The function `finalneur()` below gives the leave-1%-out cross-validation function for obtaining the cross-validated Brier- and Kullback-Leibler scores. The iterative process is analogous to the one described for the Cox model (5.2.1) and random survival forest model (5.2.2), with two modifications. As explained in Appendix C.3.1, in order to be able to predict from a `nnet` object, the predictors need to be specified as separate objects, rather than be contained in a dataframe.

```
# finalneur() function, part 1
finalneur <- function(data){
  # indicator vector for leave-1%-out cross-validation
  select.ind <- sample(rep(1:100,20),nrow(data),replace=F)
  unique.ind <- sort(unique(select.ind))
  data <- cbind(data,select.ind); pat.seq <- rep(NA,nrow(data))
  data <- cbind(data, select.ind); data <- data[order(data$Id),]
  Pred.mat <- NULL

  # transforming full data to training and test format
  data.train <- data.transformer.fin(data)
  data.test <- special.test.transformer.fin(data)

  # defining variables for training
  Vol <- data.train$Volume; Ag <- data.train$Age
  His.P <- data.train$Hist.P; ExM <- data.train$Ex.M
  ExO <- data.train$Ex.O; SexM <- data.train$Sex.M
  Ly <- data.train$L.y; ITvec <- data.train$ITvec
  statt <- data.train$Stat
```
The second modification involves converting the long format predictions (here, \texttt{predict.neur}) to a prediction matrix with one row per patient (\texttt{pred.matrix}). The \texttt{pred.matrix} objects for each iteration (each test set) are collected in one overall matrix \texttt{Pred.mat}.

```r
# final.neur() function part 2

for (i in 1:length(unique.ind)){
    # index test set
    ind <- unique.ind[i]

    # defining variables for training on all but test fold
    stat <- stat[t(data.train$select.ind! = ind]
    Age <- Ag[data.train$select.ind! = ind]
    Volume <- Vol[data.train$select.ind! = ind]
    Hist.P <- His.P[data.train$select.ind! = ind]
    Ex.M <- ExM[data.train$select.ind! = ind]
    Ex.O <- ExO[data.train$select.ind! = ind]
    Sex.M <- SexM[data.train$select.ind! = ind]
    L.y <- Ly[data.train$select.ind! = ind]
    ITvec <- ITvecc[data.train$select.ind! = ind]

    # training the neural network
    net <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + ITvec, size = 5, maxit = 2000, MaxNWts = 10000, decay = 0.05, entropy = TRUE)

    # defining the test set and obtaining predictions
    test <- data.test[data.test$select.ind = ind,]
    predict.neur <- predict(net, test, type = "raw")

    # obtaining predictions and id in wide format
    coll <- as.data.frame(cbind(predict.neur, test$id))
    colnames(coll) <- c("prob", "id")

    coll.s <- split(coll, coll$id)
    coll.t <- lapply(coll.s, function(x) {
        x <- cumprod(1 - x$prob)
    })
    pred.matrix <- do.call("rbind", coll.t)
    Pred.mat <- rbind(Pred.mat, pred.matrix)
}
# finalneur() function part 3

# sorting predictions according to id
Pred.mat <- as.data.frame(Pred.mat)

# obtaining survival times (in interval format)
temp <- as.data.frame(cbind(data.train$id, data.train$Surv))
colnames(temp) <- c("id", "surv")
temp.s <- split(temp, temp$id)
surv.times <- lapply(temp.s, function(x){
a <- x$surv[1]
return(a)
})
surv.times <- unlist(surv.times)

# Making matrix with survival interval times and status
Stat.mat <- as.data.frame(cbind(data$stat, surv.times))
colnames(Stat.mat) <- c("stat", "surv")

# calculating brier/KL scores
brieKL <- brier.function.neur.mod(Stat.mat, Pred.mat)
out <- c(brieKL[1,], brieKL[2,])
return(out)

# Appendix E, 2.3 (189–279)
5.2.4 Simulation study

In the previous three sections (5.2.1-5.2.3) the functions for obtaining the relevant cross-validated performance measures from a given simulated dataset were given. Prior to that (Section 5.1.3) the \texttt{sim.surv()} function for simulating datasets was defined. The final simulation was performed in two steps. First, a list of 1000 simulated datasets (\texttt{simul.data}) was created. Then, the relevant functions were applied. The latter step was performed on the Tukey server, which is a Unix supercomputer with an x86 operating system available from the Leiden Mathematical Institute. 20 cores were used in a FORK cluster to perform an explicit parallelization.

```r
# loading relevant packages
library(survival); library(MASS); library(pec)
library(randomForestSRC); library(nnet)
library(doParallel)

# creating list of 1000 simulated datasets
# with sim.surv() function
simul.data <- replicate(1000, simplify=FALSE,
                        sim.surv(2000, est.coefs, mu, sigma))

# (not shown: loading functions from 5.2.1–5.2.3)

# defining number of cores and making cluster
cores <- 20; cl <- makeCluster(cores, type="FORK")

# applying functions and saving output
# neural network
neur.vals <- parLapply(cl, simul.data, final.neur)
save(neur.vals, file="neur.vals")

# random survival forest (cross. val)
rf.vals <- parLapply(cl, simul.data, final.rand)
save(rf.vals, file="rf.vals")

# Cox
Cox.vals <- parLapply(cl, simul.data, final.Cox)
save(Cox.vals, file="Cox.vals")

# random survival forest (OOB)
good.rand <- parLapply(cl, simul.data, new.func.rand)
save(good.rand, file="good.rand")

# terminating cluster
stopCluster(cl)
```

155
5.3 A final comparison between the neural network, random survival forest, and Cox proportional hazards approaches

In this section a final comparison is made between the three methods, based on the results from previous chapters and the results from an $S = 1000$ simulation of $N = 2000$ datasets. Section 5.3.1 gives the simulation results, while section 5.3.2 gives a recap of the final results of chapters 3 and 4, where a neural network and random survival forest models, respectively, were fitted to the EURAMOS osteosarcoma data. Table 5.4 gives an overview of performance measures used to assess each method, and the sections in which theoretical details are given.

Table 5.4: Overview of performance measures used to assess the Cox proportional hazards model, the random survival forest model, and the neural network model. For each measure the section describing the theory is given. Hazard ratio refers to the prognostic index/ensemble mortality hazard ratio for the Cox model and random survival forest model, respectively.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Cox</th>
<th>RSF</th>
<th>Neural</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-index</td>
<td>X</td>
<td>X</td>
<td>/</td>
<td>2.2.5 (Cox), 4.2 (RSF)</td>
</tr>
<tr>
<td>Brier score</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>3.4.2</td>
</tr>
<tr>
<td>Kullback-Leibler score</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>3.4.2</td>
</tr>
<tr>
<td>Hazard ratio</td>
<td>X</td>
<td>X</td>
<td>/</td>
<td>4.4.1</td>
</tr>
<tr>
<td>logrank $\chi^2$-statistic</td>
<td>X</td>
<td>X</td>
<td>/</td>
<td>4.4.1</td>
</tr>
</tbody>
</table>

The concordance index, hazard ratio, and logrank $\chi^2$-statistic can only be obtained for Cox and random survival forest models, and not for neural networks, as in the latter case there is no natural unique ordering of subjects. For the Cox and RSF model, however, there is, and a prognostic index and ensemble mortality score, respectively, can be obtained. The hazard ratio and the logrank $\chi^2$-statistic are both calculated from the prognostic index/ensemble mortality score, and as such are also unsuitable for evaluating neural network performance. The Brier and Kullback-Leibler scores are the sole two measures that can be used to assess all three methods. In Chapter 3 additional measures were considered in an attempt to evaluate neural network performance: accuracy, sensitivity and specificity. These are obtained by comparing the event status at the observed survival times with the predicted survival probability at that time. If this probability is $\geq 0.5$ the observation is be classified as a non-event, if $< 0.5$ as an event. It was, however, shown, that these measures vary
little over different neural networks and lead to instable models when used for model selection. As such they have been excluded from the simulation in Section 5.3.1, and from the result summary in Section 5.3.2.

5.3.1 Simulation results

A simulation of $S = 1000$ of $N = 2000$ datasets was performed. The datasets were created in a way to mimic the original EURAMOS osteosarcoma data. Survival times were simulated using a log-normal distribution, and censoring was imposed using a uniform distribution. A log-normal survival distribution was chosen with the purpose of avoiding a positive bias towards Cox model performance. The assessed performance measures per method are summarized in Table 5.4. Table 5.5 gives the concordance index, hazard ratio, and the logrank $\chi^2$-statistic for the Cox model and the random survival forest model ($mtry=2$, $nsplit=3$, $nodedepth=7$, $splitrule=logrank$). Two values are given per measure: the mean over 1000 simulations and the simulation error - the standard deviation. For the Cox model a slightly higher mean C-index value is observed. When considering the spread, the values overlap within one standard deviation. The same holds for the hazard ratio and the logrank $\chi^2$-statistic.

Table 5.5: C-index, hazard ratio, and logrank $\chi^2$-statistic for Cox model and RSF model, for a simulation of $S = 1000$, with the simulation error given. The hazard ratio is calculated for the prognostic index and ensemble mortality for the Cox model and RSF model, respectively

<table>
<thead>
<tr>
<th></th>
<th>Cox</th>
<th>RSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-index</td>
<td>0.638 (0.011)</td>
<td>0.629 (0.012)</td>
</tr>
<tr>
<td>Hazard ratio</td>
<td>8.71 (2.04)</td>
<td>7.04 (1.53)</td>
</tr>
<tr>
<td>logrank $\chi^2$-statistic</td>
<td>111.1 (20.9)</td>
<td>94.3 (24.4)</td>
</tr>
</tbody>
</table>

Notably, the logrank $\chi^2$-statistic and the hazard ratio, in that order, have a relatively much higher standard deviation than the C-index. In Chapter 4, it was observed for random survival forest models that increases and decreases in C-indexes were not always matched by corresponding changes in these two statistics (section 4.4.2.4). In Figure 5.6 the distributions are shown for the simulations ($S = 1000$), with all values sorted according to C-index in ascending order. The C-index, hazard ratio and logrank $\chi^2$-statistic are shown in Figures 5.6a, 5.6b and 5.6c, respectively. The upper panels (1) show the values for the random survival forest model, the lower panels (2) for the Cox model. The hazard ratio and logrank $\chi^2$-statistic of both
models show gradually higher values as the C-index increases, but there is no strict one-on-one relationship. This is in accordance with the observations made for random survival forests in Chapter 4. Both measures demonstrate too much variability to be considered reliable performance measures and to be used as model selection tools.

Figure 5.6: Distribution of C-index (a), hazard ratio (HR) of the prognostic index (Cox) and ensemble mortality (RSF) (b), and logrank $\chi^2$-statistic (c) for Cox model (1, black) and RSF model (2, red). All values have been sorted according to C-index in ascending order.

Figure 5.7 shows the Brier (a) and Kullback-Leibler scores for the three models, for times of 1 to 10 years after surgery. The Cox model (black) and the random survival forest model (red) have nearly identical values and comparable standard deviations - in line with the observations made previously for the C-index, hazard ratio and logrank $\chi^2$-statistic. For the neural network model (blue) a different pattern can be observed. For both scores, the neural network values are initially higher, then distinctly lower than those of the Cox and RSF models. Another contrast between the neural network model and the Cox and RSF models is the size of the simulation error for the Kullback-Leibler score, which is substantially bigger for the network model. Table 5.6 shows the range of Kullback-Leibler scores for the neural network per year. The maximum values are approximately twice the size of the minimum values.
Figure 5.7: Brier (a) and Kullback-Leibler (b) scores for Cox model (black), RSF model (red), and neural network model (blue). Simulation standard deviation is shown in the error bars.

Table 5.6: Range of Kullback-Leibler scores for neural network model per year

<table>
<thead>
<tr>
<th>Year</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.30</td>
<td>0.67</td>
</tr>
<tr>
<td>2</td>
<td>0.46</td>
<td>0.87</td>
</tr>
<tr>
<td>3</td>
<td>0.52</td>
<td>0.94</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>0.96</td>
</tr>
<tr>
<td>5</td>
<td>0.55</td>
<td>1.07</td>
</tr>
<tr>
<td>6</td>
<td>0.53</td>
<td>1.04</td>
</tr>
<tr>
<td>7</td>
<td>0.51</td>
<td>1.02</td>
</tr>
<tr>
<td>8</td>
<td>0.47</td>
<td>0.99</td>
</tr>
<tr>
<td>9</td>
<td>0.42</td>
<td>0.95</td>
</tr>
<tr>
<td>10</td>
<td>0.38</td>
<td>0.91</td>
</tr>
</tbody>
</table>

As detailed in 3.4.2, a modification has been made to the Kullback-Leibler score calculations, to allow for extreme observations. Predicted survival probabilities of 0 and of 1 are replaced with probabilities of 0.00001 and 0.99999, to ensure that the scores can be computed. This also ensures that that in the presence of extreme -unrealistic - survival probabilities the Kullback-Leibler score values blow up. In a regular situation, under random guessing, the maximum observed score would be approximately 0.69. Figure 5.8 shows the score plots of 5.7 again but with the neural network scores split out into two parts. "Neural (high)” (dotted line, blue) refers to the neural network observations for which the 5-year Kullback-Leibler score exceeds 0.69 (n = 293), "Neural (low)” (solid line, blue) refers to the observations with a 5-year score under 0.69 (n = 707). In Figure 5.8b two distinct lines can be observed for the neural Kullback-Leibler scores. The lower one matches the Cox (black) and RSF (red) scores closely before decreasing. For the Brier scores (5.8a) a much slighter difference is observed. As detailed in Chapter 3, high Kullback-Leibler scores are indicative of an instable network that makes extreme predictions. Table 5.6 and Figure 5.8 suggest that there is a subset of unstable networks, indicating that a 5-node hidden network with 0.05 decay is not appropriate for all simulated datasets.
Figure 5.8: Brier (a) and Kullback-Leibler (b) scores for Cox model (black), RSF model (red), and neural network model (blue), with a separation of neural network scores in high (dotted line) and low (solid line) values. Simulation standard deviation is shown in the error bars.

Previously the discussion of using out-of-bag (OOB) estimation versus cross-validation was broached (4.4.2, 4.4). Cross-validation was chosen over OOB in Chapter 4 as it has been reported using OOB data may result in a biased estimate of the generalization error, and because cross-validation was used in the previous chapter for neural network parameter tuning. To gain insight into the differences between the two approaches OOB estimates were obtained from the simulated data. The mean C-index, hazard ratio, and logrank $\chi^2$-statistic along with simulation error are given in Table 5.6. All three values are almost identical. A slight variation in the latter two measures can be observed, but for the C-index the first difference is found in the fourth decimal (0.628656 and 0.628748 for OOB and leave-1%-out cross-validation, respectively).

Table 5.7: C-index, hazard ratio and logrank $\chi^2$-statistic for RSF model, obtained via OOB data, and via leave-1%-out cross-validation.

<table>
<thead>
<tr>
<th></th>
<th>OOB (RSF)</th>
<th>Cross-val (RSF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-index</td>
<td>0.629 (0.0118)</td>
<td>0.629 (0.0117)</td>
</tr>
<tr>
<td>Hazard ratio</td>
<td>7.09 (1.52)</td>
<td>7.04 (1.53)</td>
</tr>
<tr>
<td>logrank $\chi^2$-statistic</td>
<td>94.7 (24.1)</td>
<td>94.3 (24.4)</td>
</tr>
</tbody>
</table>

Figure 5.7 shows the OOB (purple) and leave-1%-out cross-validation estimates (red) for the Brier (a) and Kullback-Leibler scores (b). Once more, the values are almost identical. OOB estimation for the first 9 years after surgery was straightforward.
For the 10th year, for 96 simulated datasets score values could not be obtained, and the mean value and standard deviation shown are those of the remaining 904 datasets. This can be explained by considering the distribution of event times in the simulated datasets. Figure 5.8 shows the pooled survival times of all 1000 datasets, with all survival times in 5.8a and only event times in 5.8b. In the latter plot it can be observed that events are very sparse from the 10th year onwards. When obtaining an OOB estimate, approximately one third of the trees is used. In the event of rare observations they may be excluded from this tree set. As a random forest can only predict for observed event times, in the event of extreme event times OOB predictions are likely to be limited to a shorter time range.

Figure 5.9: Brier (a) and Kullback-Leibler scores (b) for RSF model, obtained via OOB data (purple), and via leave-1%-out cross-validation (red).

All together, these results show that for this kind of data OOB estimation can replace cross-validation, which is desirable from a computational viewpoint. With the current approach obtaining the performance measures for a single simulated dataset takes approximately 45 minutes, requiring 100 models to be fitted for the leave-1%-out cross-validations. The computation time can be drastically reduced if OOB estimation is used. This option is not available in a neural network section, where a single cross-validated simulation takes approximately 15 minutes. Do note that for networks with more hidden nodes, variables, and subjects, running time will increase drastically. The Cox model is computationally most efficient, taking less than a minute for a single dataset cross-validation.
5.3.2 EURAMOS-1 Osteosarcoma data

This section makes a final cohesive comparison between the three different models fitted to the EURAMOS osteosarcoma data, which were previously discussed in Chapter 3 (A Neural network approach) and Chapter 4 (A Random survival forest approach).

5.3.2.1 Performance measures

As shown in Table 5.8, the Cox and RSF model have comparable C-indexes, with a marginally higher value for the Cox model. Both the hazard ratio and logrank $\chi^2$-statistic are higher for the Cox model with a relatively bigger difference than observed for the C-index. As discussed in the previous section, however, these two measures show high variability, and as such any interpretation should be made with some caution.

Table 5.8: C-index, hazard ratio (HR) of the prognostic index (Cox) or ensemble mortality (RSF), and the logrank $\chi^2$-statistic, for the random survival forest (RSF) model and the Cox proportional hazards model

<table>
<thead>
<tr>
<th></th>
<th>RSF</th>
<th>Cox</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-index</td>
<td>0.689</td>
<td>0.703</td>
</tr>
<tr>
<td>Hazard ratio</td>
<td>19.95</td>
<td>25.98</td>
</tr>
<tr>
<td>logrank $\chi^2$</td>
<td>51.54</td>
<td>55.69</td>
</tr>
</tbody>
</table>

In Figure 5.10, where all three models are compared on Brier (solid line) and Kullback-Leibler scores (dotted line), it can be observed that the estimates for the
Cox model (black) and the RSF model (red) are nearly indistinguishable. Notably, the scores for RSF are truncated after 6 years.

Figure 5.11: Brier (solid line) and Kullback-Leibler (dotted line) scores are shown for the Cox model (black), RSF model (red), and the neural network model (blue)

A RSF model can only make predictions for event times observed in the training set, which has its latest event at 6.935 years. The follow-up time in the entire dataset ranges from 0.005 years to 8.975 years after surgery (Figure 5.12a), with most events occurring in the first half of the follow-up (Figure 5.12b), and only a single event observed past the seven year mark. While both the Cox model and neural network are capable of making predictions for any survival time observed in the training set, be it censored or an event, the RSF model is due to the event time constraint limited by necessity to predictions before 7 years.
5.3.2.2 Patient-specific predictions

Figure 5.13 shows the survival curves of all test set patients, as predicted by the neural network (a), the Cox model (b) and the RSF model (c). Please note that the RSF predictions are truncated just before 7 years. Looking at the general shape of the individual survival curves it can be observed that the neural network gives very smooth curves, while stricter and more parallel lines are produced by the simple Cox model, and the survival probabilities of the RSF model have a more stepwise nature. These observations are in accordance with the characteristics of the individual methods - a neural network fits complicated non-linear relationships (here via logistic activation functions), a Cox model has an implicit linearity, and an RFS model relies on recursive binary partitioning.

The survival curves have been coloured according to predictor values. 5.13(1) distinguishes between good (blue) and poor histological response (red), while 5.13(2) distinguishes between the absence (blue) and possible/definite presence of lung metastases (red). All three models appear to properly separate the different patients. When comparing the models on overall prediction pattern, the neural network and Cox model have a similar spread, while the RSF model predictions are closer together-more clustered.

In Chapter 4 three different RSF models were examined. The fully tuned model with the highest C-index had the most clustered prediction pattern, while the untuned model predictions resembled the Cox predictions the most. An intermediate model that had an almost equal performance to the fully tuned model, but less clustered predictions, was ultimately selected, and has been discussed throughout this chapter.
Figure 5.13: Predicted survival curves for all test set patients for the neural network model (a), Cox model (b) and RSF model (c). (1) Patients are distinguished on good (blue) and poor (red) histological response. (2) Patients are distinguished on the absence (blue) and presence of lung metastases (red).

While the predicted probabilities of the RSF model are different from the Cox model and neural network, the high value of the C-index and the low Brier and Kullback-Leibler score values indicate a discriminative model, capable of identifying risk factors and patient groups.

Table 5.9: Patient characteristics for reference patient ’poor hist’ patient and ’poor hist + lung mets’ patient

<table>
<thead>
<tr>
<th></th>
<th>Hist</th>
<th>Lung mets</th>
<th>Other mets</th>
<th>Vol</th>
<th>Age</th>
<th>Excision</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Good</td>
<td>No</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>Wide/radical</td>
<td>Female</td>
</tr>
<tr>
<td>Poor hist</td>
<td>Poor</td>
<td>No</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>Wide/radical</td>
<td>Female</td>
</tr>
<tr>
<td>Poor hist + lung mets</td>
<td>Poor</td>
<td>Yes/possible</td>
<td>No</td>
<td>203.37</td>
<td>15.02</td>
<td>wide/radical</td>
<td>Female</td>
</tr>
</tbody>
</table>

Figure 5.14 shows the predicted survival for three specific patients - a reference patient (green), a poor hist patient (orange), and a poor hist + lung mets patient (purple), where the latter two refer to a patient with poor histological response and
a patient with poor histological response and the definite or possible presence of lung metastases, respectively. The reference patient has reference categories for each categorical variable and mean values for the continuous variables. Table 5.9 gives the exact patient characteristics for the three patients. In Figure 5.14a the neural network predictions (dashed line) are compared to the Cox model predictions (solid line), in Figure 5.14b the same comparison is made for the RSF model (dashed line). Overall, compared to the Cox model, the neural networks predicts slightly lower survival probabilities for the poor hist and poor hist + lung mets patients, after the four year mark. The RSF predictions follow the Cox model more closely, but are less smooth than the Cox and neural network predictions. Of interest to note is that in Figure 5.11, where the Brier and Kullback-Leibler scores are shown for the three models, it is also at the four year mark that the neural network scores deviate from the Cox scores - more specifically, they become lower, which suggests that the neural network performs better. In light of this, it is a possibility that the Cox model slightly underestimates survival for later years. Note, once more, that the RSF predictions are truncated before 7 years.

Figure 5.14: Patient-specific predictions for the neural network model (a, dotted lines), random survival forest model (b, dotted lines) and Cox proportional hazards model (a,b, solid lines). Shown are predictions for a reference patient, with reference categories for categorical variables, and mean values for continuous variables (green), for a patient with poor histological response (orange), and a patient with poor histological response and lung metastases.

5.3.2.3 Variable importance

Figure 5.15 shows the variable importance for the neural network model (a) and the random survival forest model (b,c). The relative variable importance in a neural
network is calculated using Olden’s connection weight method [28], which considers
the weights from input to hidden layer and hidden layer to output layer for each
variable (section 3.4.2). The presence of lung metastases, histological response and
the presence of other metastases are identified as strongest predictors, in that order.

Figure 5.15: Variable importance for neural network and random forest model. For
the neural network Olden’s connection weight method is used to measure variable
importance (a), for the random survival forest VIMP (b) and minimal depth (c).

Variable importance in a random survival forest context is assessed by the VIMP
score and minimal depth. VIMP is most commonly reported in use. VIMP gives the
difference in OOB error (1−C-index) before and after the permutation of observations
in the variable of interest, while minimal depth measures how close to the root node a
variable is first chosen for a split (Section 4.4.1). The latter is in part a chance process,
as in the random forest algorithm a subselection of variables is made (here, mtry=2).
In the situation where two weaker predictors are randomly selected, a less important
predictor can still have a small minimal depth, giving the appearance of importance.
Indeed, when comparing the VIMP (Figure 5.15b) and minimal depth (Figure 5.15c)
plots, in the latter the differences between predictors is less stark. Overall, however,
the two measures display the same pattern of variable importance. The strongest
predictors are histological response and the presence of lung metastases, in that order.
The third strongest predictor - at considerable distance - is absolute tumour volume. While slight when compared to the two strongest predictors, it is still noticeably bigger than the remaining ones. When considering the variable importance in a Cox model (Table 5.10) - given by hazard ratios, which can be interpreted directly as the predictor effects - the most important predictors correspond to those of the neural network: the presence of lung metastases, followed by histological response, by the presence of other metastases. Notably, the predictor absolute tumour volume, while of importance in the RSF model, has a hazard ratio of 1 in the Cox model, indicating no significant predictor effect, and is also not identified as a strong predictor in the neural network model.

Table 5.10: Cox proportional hazards model: Hazard ratios (HR) with 95% confidence intervals (95% CI)

<table>
<thead>
<tr>
<th></th>
<th>HR</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hist response: Good</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Hist response: Poor</td>
<td>2.509</td>
<td>2.02-3.12</td>
</tr>
<tr>
<td>Age</td>
<td>1.004</td>
<td>0.98-1.02</td>
</tr>
<tr>
<td>Volume</td>
<td>1.000</td>
<td>1.00-1.00</td>
</tr>
<tr>
<td>Excision: Wide/radical</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Excision: Marginal</td>
<td>0.820</td>
<td>0.6-1.12</td>
</tr>
<tr>
<td>Excision: Other</td>
<td>1.269</td>
<td>0.88-1.83</td>
</tr>
<tr>
<td>Sex: Female</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Sex: Male</td>
<td>1.230</td>
<td>0.99-1.53</td>
</tr>
<tr>
<td>Lung mets: No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Lung mets: Yes/Possible</td>
<td>2.621</td>
<td>2.09-3.29</td>
</tr>
<tr>
<td>Other mets: No</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Other mets: Yes/Possible</td>
<td>1.987</td>
<td>1.33-2.97</td>
</tr>
</tbody>
</table>
Chapter 6

Summary and contribution

This chapter offers a concise recap of Chapters 2 to 5, summarizing the most important observations and offering recommendations for model fitting and future research. Sections 6.6 and 6.7, pertaining to avenues of future research and the novelty of this thesis, respectively, in content largely overlap with the Discussion chapter. They have been included here with the purpose of providing a chapter that summarises the most important elements of the thesis.

6.1 Aims

This thesis investigates the performance of Ishwaran’s random survival forest model [19] (Section 4.2-4.3) and Biganzoli’s PLANN survival neural network [10] (Section 3.3) on the EURAMOS-1 osteosarcoma data (Chapter 1). These two approaches have adapted traditional random forest and neural network methodology, respectively, to function with censored time-to-event data. A comparison is made between the two machine learning (ML) methods and a statistical model commonly used for survival analysis- the Cox proportional hazards model. The purpose of this thesis is two-fold: 1) perform and in-depth comparison of the ML methods neural networks and random survival forests, gaining insight into the potential of ML for clinical data with a limited number of predictors; 2) add to osteosarcoma literature, in which ML methods have not yet been extensively examined. The analyses performed on the EURAMOS osteosarcoma data are reinforced by a simulation. Suitable performance measures are identified, the model hyperparameters are investigated, and the advantages and (potential) pitfalls of the models are documented.
6.2 General approach

In this section the analysis approach, described in detail in Chapters 3, 4 and 5, is summarised. First, the data is described, then the simulation study, and finally the implementation.

6.2.1 Data

The EURAMOS-1 osteosarcoma data, as examined here, has observations on seven variables for 2025 patients, five of which are categorical, two are continuous: histological response (good, poor), presence of lung metastases (no, yes/possible), presence of other metastases (no, yes/possible), surgical excision (wide/radical, marginal, intraleisional/other), sex (male, female), absolute tumour volume, and age. The patients are followed after surgery, up to a maximum of 8.9 years. Most events occur in the first half of the study and 73% of the observations are censored. See Chapter 2 for more detail on the EURAMOS-1 osteosarcoma data.

6.2.2 Data simulation

The data simulation was designed to mimic the characteristics of the original data. Three steps can be distinguished (section 5.1).

1. The original EURAMOS data is used to count the occurrence of unique combinations of categorical variables. Combinations occurring $\geq 20$ are retained. For each combination the means and the covariance matrices of the continuous variables are determined.

2. Data ($N = 2000$) is simulated by sampling the unique categorical covariate combinations in accordance to proportion of occurrence in the original data. For each combination, continuous predictor values are sampled from a multivariate normal distribution with mean, variance and covariance obtained from the original data.

3. Using the original dataset, coefficients for the covariates are obtained from a log-normal regression. The survival times are then given by

$$\log(T) = \mu + \beta^T X + \sigma \epsilon, \epsilon \sim N(0, 1),$$

(6.1)
where $\mu$ is the intercept, $\beta^T$ the estimated coefficient, $X$ the covariate matrix for a given simulated dataset, $\sigma$ the scale parameter, and $\epsilon$ the error, normally distributed with mean 0 and variance 1. Survival times are simulated from a log-normal distribution to avoid introducing bias towards any of the considered methods. Censoring times are simulated from a uniform distribution with $U(1, 11)$, in a manner which mimics the censoring observed in the original dataset.

### 6.2.3 Method

Both machine learning approaches - the neural network and random survival forest - have parameters that require tuning. For this purpose the data was divided into a training and a test set, containing two thirds and one third of the observations, respectively. The original event/censoring proportion was retained in both sets. The training set was used to perform a five-fold cross-validation, using the performance measures detailed in Section 6.3 to evaluate the methods. For neural networks two parameters were tuned (see Section 3.4.3)

1. **Node number.** The node number refers to the number of nodes present in the hidden layer of the 3-layer feed-forward artificial neural network. Increasing the number of nodes allows the network to accommodate more complex relationships between predictors amongst each other and predictors and outcome. In context of a survival neural network the latter includes time dependencies. Increasing node number may also result in overfitting, ensuring poor generalization to new data. To give an indication of range: usually nodesizes are considered in between the number of input variables and output variables.

2. **Weight decay parameter.** The weight decay parameter regulates the network by penalizing large weights and guarding the network against overfitting. An added advantage is that weight decay will also improve network convergence, saving on computational time. Typically decay values between 0 and 0.1 are considered (see Section 3.1).

For random survival forests four parameters were tuned (see section 4.3).

1. **mtry.** The mtry parameters determines the number of predictors that are randomly selected to be considered for a given node splitting. This measure
introduce an additional source of randomization into the process, and distinguishes it from the bagging tree approach. If \( mtry \) is set to equal the total number of predictors the random survival forest is reduced to bagging.

2. \textit{nsplit} The \( nsplit \) parameter determines the number of split points that are randomly selected for a given predictor. When left unspecified, all possible split points are considered, enacting deterministic, rather than random, splitting.

3. \textit{nodesize}. The \( nodesize \) parameter regulates the forest growth by imposing a constraint on the minimal number of observations per terminal node. The larger the specified \( nodesize \), the less branched a forest will be.

4. \textit{nodedepth}. The \( nodedepth \) parameter regulates the growth by limiting the number of splits per node. The depth of a tree is measured from root node to terminal nodes. Like \( nodesize \), it will reduce forest branching, but it does not impose a number of observations per terminal node.

For both methods, the optimal models were evaluated on the validation data. The parameters found through tuning on the EURAMOS osteosarcoma data were retained when applying the models to the 1000 simulated datasets of \( N = 2000 \). A leave-1%-out cross-validation approach was used to obtain cross-validated performance measures. The mean over the 1000 datasets was taken, as well as the standard deviation to obtain the simulation error.

6.3 Performance measures and variable importance

For each method a range of performance measures was considered. For neural networks the potential of the measures listed below was examined. On investigation of the measures, the Brier score and Kullback-Leibler score were identified as consistent performance measures leading to the selection of stable networks. Accuracy, sensitivity and specificity were also considered, as in previous literature variations on the measures have been utilized.\[4, 48\] The final measure considered was the error value, obtained from the minimized neural network loss function. All four measures were shown to lead to poor, instable, model choices.

1. \textbf{Brier score}. The Brier score can be calculated for individual time points, here for every year of the study. It is a measure for the deviation of the predicted survival probability to the observed status. Observations are weighted using
inverse probability censoring weighting. A brier score will be 0 for a perfect model and 0.25 for a random one. (see Section 3.4.2).

2. **Kullback-Leibler score.** The Kullback-Leibler score is comparable to the Brier score, but is based on the principle that the performance of a predictive model can be measured by evaluated the log likelihood of the model at the observations. The score ranges from 0 to 0.69. Important to note is that in the event of extreme survival probability predictions of 0 and 1 the score cannot be computed, as the logarithm of 0 cannot be taken. A modification is introduced, that slightly increases and decreases the extreme predictions 0 and 1, respectively. The consequence of this is that calculated value, in the event of extreme observations, will be noticeably larger than usual. This property is of particular value in a neural network context, as instable networks are given to extreme predictions, and the score then serves as a warning sign of instability. In the rest of this chapter it is referred to as the modified Kullback-Leibler score. (see Section 3.4.2)

3. **Accuracy.** Accuracy is an overall measure for model performance and unlike the previous two scores independent of time. Accuracy is assessed by comparing the predicted survival probability at the observed survival time to the event status. In the event that the predicted probability $< 0.5$ it is considered an event. If the observed event status also indicates an event the observation is considered accurate. Sensitivity and specificity rely on the same principle as accuracy. **Sensitivity** measures the proportion of correct events, while **specificity** measures the proportion of correctly classified non-events. (see Section 3.4.2)

4. **Error value.** The error value refers to the value given by the cross-entropy loss function when the neural network algorithm has converged. Minimizing the cross-entropy loss function has been shown to be equal to maximizing the log likelihood [10] (see Section 3.3).

For **random survival forests** the following measures were considered. The concordance index, Brier score and Kullback-Leibler score were shown to be reliable and consistent performance measures. For the ensemble mortality hazard ratio and the logrank $\chi^2$-statistic great variability was observed in the simulation, and an increase in C-index only had a spurious association with increases in either of the two measures.
1. **Concordance index.** The concordance index (C-index) is the most frequently reported performance measure for survival data. It calculates the proportion of concordant pairs, where a pair of observations is considered concordant if the subject with the highest predicted prognostic index of the two has the shortest survival time. In random survival forest context the concordance index is replaced by the ensemble mortality. The ensemble mortality score for a given subject can be interpreted as the number of deaths that would be expected given that all other subjects had the same predictor values. Both the ensemble mortality score and the Cox prognostic index have a similar interpretation—the higher, the worse the prognosis.

2. **Brier score.** Definition given for neural networks applies.

3. **Kullback-Leibler score.** Definition given for neural networks applies. Note that for random survival forests no difference was observed between the regular and modified Kullback-Leibler score, as random survival forests, unlike neural networks, are not given to within-model instability.

4. **Hazard ratio.** The hazard ratio is calculated by performing a simple Cox regression on the ensemble mortality score, which is normalized to a range between 0 and 1. The obtained hazard ratio measures the increase in risk when moving from the lowest scoring patient (at 0) to the highest scoring patient (at 1). The normalization allows for comparison across datasets and between the random survival forest and Cox model. (see Section 4.4.1)

5. **Logrank $\chi^2$-statistic.** This statistic measures the difference between the survival curves of low and high risk patients. A high risk patient has an ensemble mortality score— or indeed for the Cox model a prognostic index value—equal to or higher than the median score. (see Section 4.4.1)

All measures detailed in the above two lists can also be applied to the Cox proportional hazards model. The three measures found to be satisfactory—the Brier score, Kullback-Leibler score and C-index—were used for formal comparison.

The importance of a predictor in the Cox proportional hazards model is readily available in the form of a coefficient or the more popular **hazard ratio**. For machine learning approaches variable importance is not as straightforward and additional measures need to be taken to obtain such a measure. A neural network
algorithm on convergence produces weights. One approach to gain insight into the relative importance of the predictors is by using the connection-weight method proposed by Olden [28]. This approach uses all the weights involved in connecting a given predictor input node to the output node to quantify its importance (see Section 3.4.2). A random forest as is contains little information that can directly be used to obtain variable importance. One approach, which makes use of the individual trees, measures how close to the root node the predictor was first used as a split. This is called the minimal depth and while it is attractive in that it can be obtained from an existing forest, it is considered less reliable than the VIMP measure. VIMP stands for variable importance and is calculated for a given predictor by refitting the forest after permuting all observations in the relevant predictor. The difference in error prior and post permutation gives the variable importance. Of note is that for a random forest only the overall importance of a variable can be obtained (see Section 4.4.1). In contrast, for the Cox model each factor level will have an associated hazard ratio, and each continuous variable will have a hazard ratio associated with a one unit increase. The neural network does distinguish between factor levels for the simple reason that factor variables cannot be included in the model and the factors need to be entered separately, binary coded.

6.4 Results

The tuning cross-validation resulted in the following two models.

1. A neural network model which minimises the cross-entropy loss function, with 5 nodes in the hidden layer and a weight decay parameter of 0.05, with a logistic activation functions

2. A random survival forest with $\text{mtry}=2, \text{nsplit}=3$ and $\text{nodedepth}=7$

As detailed in the the Method section (6.2.3), the models were applied two times - first to the original EURAMOS-1 osteosarcoma data, then to 1000 simulated datasets, designed to mimic the original data. Table 6.1 shows the concordance index (C-index) obtained for the Cox models and the random survival forest models. For both the original and the simulated data the C-index of the Cox model is slightly higher. When considering the spread observed in the simulation the values overlap within one standard deviation. Overall, the C-indexes obtained for the original data are approximately 0.06 higher than those from the simulation.
Table 6.1: Concordance index for random survival forest (RSF) and Cox proportional hazards model as measured on the EURAMOS-1 osteosarcoma data and as obtained from the simulation ($S = 1000$), with simulation standard error given.

<table>
<thead>
<tr>
<th>EURAMOS-1 data</th>
<th>Simulation ($S = 1000$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSF</td>
<td>0.689</td>
</tr>
<tr>
<td>Cox</td>
<td>0.703</td>
</tr>
</tbody>
</table>

Figure 6.1 shows the Brier (solid line) and Kullback-Leibler (dotted line) scores for the original (a) and simulated data (b). Once more, the pattern observed for the two is similar, with the original data having slightly lower values. The random survival forest (red) and the Cox model (black) have near identical values. The neural network scores are initially comparable, then decrease later in the follow-up. Notably, in contrast to the other two models, for the neural network Kullback-Leibler scores a considerable spread is observed in the simulation. On closer look (detailed in section 5.3.1) a subset of 293 datasets gives considerably higher Kullback-Leibler scores, indicating extreme predictions and by extent instable networks.

![Figure 6.1: Brier (solid line) and Kullback-Leibler (dotted line) scores for Cox model (black), RSF model (red), and neural network model (blue). a) EURAMOS-1 osteosarcoma data. b) Simulation ($S = 1000$). Simulation standard error is shown in the error bars.](image)

In Table 6.2 the three most important predictors are given for each method. For the neural network, variable importance is given by the relative importance as measured by Olden’s connection weight method [28], for the random survival forest the variable importance is quantified by VIMP, and for the Cox model the hazard ratios with associated 95% confidence intervals are provided. For the neural network and
Cox model the same predictors in the same order are identified as most important - the possible/definitive presence of lung metastases, poor histological response, and the possible/definitive presence of other metastases. In contrast, the random forest identifies histological response as most important, followed by lung metastases and absolute tumour volume.

Table 6.2: Predictors identified as imported by neural network (relative importance), random survival forest (VIMP) and Cox proportional hazard model (hazard ratio). Relevant predictors are the presence of lung metastases (Lung), the presence of other metastases (Other), and histological response (Hist). RSF does not distinguish between factor levels.

<table>
<thead>
<tr>
<th>Neural network</th>
<th>(rel. imp)</th>
<th>RSF (VIMP)</th>
<th>Cox (HR, 95% CI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Lung (yes)</td>
<td>0.2854</td>
<td>Hist 0.0694</td>
<td>Lung (yes) 2.621 (2.09-3.29)</td>
</tr>
<tr>
<td>2 Hist (poor)</td>
<td>0.2582</td>
<td>Lung 0.0496</td>
<td>Hist. (poor) 2.509 (2.02-3.12)</td>
</tr>
<tr>
<td>3 Other (yes)</td>
<td>0.2222</td>
<td>Volume 0.0113</td>
<td>Other (yes) 1.987 (1.33-2.97)</td>
</tr>
</tbody>
</table>

Please note that the full model comparison for the EURAMOS-1 osteosarcoma data and the simulation can be found in sections 5.3.2 and 5.3.1, respectively.

### 6.5 Method comparison and recommendations

In conclusion it can be said that the methods perform in a comparable manner, with the neural network showing slight superiority, as measured by the Brier and Kullback-Leibler scores, in the second half of the follow-up. Variable importance as measured by the Cox model and neural network are consistent with each other, while the random forest deviates, notably by including absolute tumour volume - a weak variable in the other models - in its top three predictors. The comparison between the two machine learning methods can be extended beyond performance measures and variable importance. This section gives a number of observations on the methods, detailing the advantages, and possible pitfalls, alongside some recommendations for implementing and evaluating the models.

For the neural network two parameters are tuned, the number of nodes and the weight decay. In absence of an imposed weight decay, or with the specification of an insufficient weight decay instability was observed within network. The final neural network found stable for the original EURAMOS data, when applied to the simulated datasets, showed instability for a subset of datasets, indicating that tuning is essential,
and that a network will be sensitive to even slight changes in the data. A way to observe in-network instability is by computing the modified Kullback-Leibler score, which will give high values in such a situation. A less formal, yet also informative approach, is refitting the same network multiple times and seeing whether noticeable shifts are present in the prediction curve pattern and whether there is variability in the variable importance measures. Overall, the recommendation would be made to employ both options. The latter, however, is not very practical in a cross-validation tuning context.

The Brier and modified Kullback-Leibler score were used for parameter tuning and for evaluating model performance. In the neural network, unlike the Cox model and random survival forest model, there is no natural ordering of predictions and as such a C-index cannot be computed. The score values, while consistent in performance, are calculated over time, and currently, there appears to be no single overall performance measure readily available to quantify model performance.

The survival neural network, as described by Biganzoli, in its implementation requires a specific data transformation to longitudinal format for both training and test data prior to neural network fitting. All described neural network approaches for survival data, however, necessitate the division of observations into time intervals - here month-long intervals were defined. This reduces the model accuracy and may have implications when comparing a neural network model to other approaches that do use exact survival times.

For a neural network, given a stable model, the prediction curves are smoother but comparable to those from the Cox model, suggesting they are reliable. This is in contrast to the predictions obtained from a random survival forest, where for the optimally tuned forest severe prediction clustering was observed. On investigation, it was established that high values for the parameter nodesize, low values for the parameter nodedepth and a low value for mtry are associated with such clustering. Notably, all these parameters impose regularization on the forest, with the first two explicitly reducing branching. The clustered predictions are associated with good - even superior - C-indexes and good Brier and Kullback-Leibler scores. In light of this, when fitting a RSF model, it is recommended to perform a visual inspection of the prediction plots in addition to evaluating the performance measures. In general, it would be unwise to rely on a random survival forest model for individual patient survival predictions. The good performance measures do indicate that the model is able to distinguish between high and low risk patients. A considerable advantage
of the random survival forest model is that, in contrast to the neural network, it is stable with a decent performance even in the absence of tuning.

An issue unique to the random survival forest model is that it can only predict for observed event times. As detailed in Sections 4.4.1 and 5.5.2 this has implications for calculating the Brier and Kullback-Leibler scores. One of the consequences is that in the event of an uneven censoring distribution, when events later in the study are rare, scores can only be calculated for a limited window of the follow-up time. For the EURAMOS data, for instance, scores could only be obtained for the first 6 years. A way to mitigate this in a training set/test set context is by ensuring that the survival time distributions are as comparable as possible in the two sets. This dependency on observed event times is also an issue for out-of-bag (OOB) estimation, where on average only a third of the trees can be used to make predictions for a given observation. OOB estimation is a natural extension of the random survival forest structure that can replace cross-validation, and is less computationally intensive. In the simulation it was shown that for this particular kind of data OOB estimation and leave-1%-out cross-validation produce near identical performance measure values. This is not a given for every dataset, and as such it would be recommended to run a small cross-validation and compare the results to OOB performance measures, to verify whether the cross-validation can be disregarded. Note that while prediction accuracy, derived from the C-index, can be obtained directly from the fitted model in the event of OOB estimation, but in the event of cross-validation a manual implementation is necessary (see Section 4.4.1).

Both models, when compared to the Cox model, are more difficult to implement in R-software. Survival neural networks require extensive data pre-processing. Random survival forests have an R-software implementation, but like neural networks require manual implementation of the performance measure function. Of the three, the Cox model is the most efficient from a computational viewpoint, followed by the neural network and random survival forest. Note that for the latter two, when increasing the number of subjects and variables, the computation time will increase, and, on varying parameters for a given dataset, computation time may increase too.

In summary, the main drawbacks and advantages of Biganzoli's survival neural network are the following.

**Drawbacks**
• within-network instability
• the necessity of defining time intervals instead of exact survival times
• the lack of a ready implementation in R and the extensive data transformations required prior to and after the model fitting
• the absence of an overall performance measure

Advantages

• superior performance as measured by the Brier and Kullback-Leibler scores later in the follow-up, as compared to the Cox and random survival forest model
• (seemingly) reliable patient-specific prediction

Accordingly, for Ishwaran’s random survival forest the following drawbacks and advantages can be distinguished.

Drawbacks

• the clustering of prediction curves and the consequently unreliable patient-specific prediction
• the restriction of predictions to observed event times

Advantages

• the presence of a natural ordering of subjects, which allows the ensemble mortality score to be obtained (comparable to the Cox model prognostic index) and consequently an overall performance measure to be computed - the C-index
• in-model stability
• good and stable performance even in the absence of parameter tuning
• Available R implementation

6.6 Future research

In this section ideas for possible future avenues of investigation are posed. These ideas are also covered in the Discussion section in greater detail.

180
6.6.1 Neural network

A general drawback of both machine learning approaches when compared to the Cox model is interpretability - only an overall measure of variable importance can be obtained. For random survival forest variable importance is quantified by the VIMP measure, which measures the difference in overall error rate prior to and after predictor permutation. In that regard it could be considered superior to the neural network, for which variable importance is quantified based on weight. The permutation approach as used for random survival forests is currently not feasible, as a single overall performance measure, such as the C-index, is not available for survival neural networks. It would be desirable to define such a measure, not only in order to provide an alternative way of quantifying variable importance, but also to facilitate automation of the parameter tuning process, and for overall model evaluation.

Alternatively, it could be investigated whether predictor permutation can be used in combination with the existing connection weight importance measure. The relative change in this measure on permutation could be calculated. In its current structure, only a singly output node is defined, and as such only a single importance measures per predictor can be calculated. Alternative survival neural networks have been proposed that define an output node per time interval, most notably by Liestol [35], who describes an approach that in theory is very close to that of Biganzoli. With a separate output node defined for every time interval, variable importance could be tracked over time, allowing insight into the non-linearities that are otherwise implicit in the model. Note, however, that Liestol's approach, as well as many other described approaches use modified loss functions and would necessitate adjustments of existing R implementation.

Biganzoli's [10] and Liestol's [35] approaches both truncate the data on censoring and estimate the conditional hazard, from which survival probabilities can be calculated. Other methods have been proposed, most notably by Street [58], who estimates survival probabilities directly and imputes censored observations with Kaplan Meier estimates. It would be of interest to compare the performance of these methods. As they use different approaches to deal with censoring it would be relevant to examine them under different censoring distributions.

Common to all these methods is that exact survival times are not used, and time intervals are defined. In most studies these intervals are sizable chunks of time, up to a year long [4]. In this thesis, in the interest of staying as close to the original survival times as possible, month-long intervals were defined. It would be relevant to
examine the effect of defining different intervals for a given dataset. The implications of comparing the neural network to a model that does use exact times also needs to be investigated.

### 6.6.2 Random survival forest

In the simulation it was established that performance measures obtained from the leave-1%-out cross-validation and those obtained using OOB estimation were near equal. This, of course, is not conclusive proof that OOB can replace cross-validation, and merely shows that it is an appropriate choice for this particular data, and this particular cross-validation. In the context of regression and classification forests, some concerns have been raised on a potential dependency between bias in OOB estimates and the \textit{mtry} parameter. For random survival forests specifically no research has been done into OOB bias. It would be relevant to compare OOB versus cross-validation behaviour for different kinds of survival data, under different parameter specifications, while varying sample size and potentially also the censoring distribution. The coarseness of the cross-validation should also be considered. OOB is an attractive alternative, as it requires a model to be fitted only once, saving on computational time.

The main drawback of the random survival forest is the observed prediction clustering, making it unsuitable for individual patient survival probability predictions. Such visual evaluation has been absent from literature detailing ML implementations, and further investigation into what methods are suitable/unsuitable for such predictions is required. For random survival forests it was observed that none of the performance measures were indicative of prediction clustering. Possibly, a way of formally quantifying prediction spread could be established.

### 6.7 Novelty

With the growth in interest and application of machine learning (ML) techniques in medical research a complementary increase of poor conception and execution is observed.\cite{[12], [33]} Issues include a lack of validation and the use of improper performance measures, both in parameter tuning - a necessity for most ML methods - and in model evaluation.\cite{[12]} This thesis describes an in-depth analysis of two ML methods - neural networks and random survival forests - in which parameters are tuned via cross-validation and the models are validated on a separate test set. Various performance measures are compared and three reliable measures are identified:
the concordance index, the Brier score and the Kullback-Leibler score, the latter two of which are measured over time. A modification is proposed to the Kullback-Leibler score when applied to neural networks, allowing it to function as a proxy to instability - a common issue in the neural network framework. The importance of visual evaluation, frequently neglected in literature, is shown to be valuable both in neural network and random survival forest context. For the former, it can identify instability, for the latter, it can showcase prediction clustering behaviour. Clustering is not identified by any of the performance measures, yet is important to take note of, as in that event a model will not be suitable for individual survival predictions. Results obtained from the EURAMOS-1 osteosarcoma data are reinforced by a simulation study. The design of the simulation is particular in that it closely mimics the original data, with survival times generated from a log-normal distribution to avoid biasing the data to any of the considered methods.

In summary, this thesis has

- given a consistent evaluation of the methods ensuring fair comparison that is entirely reproducible,
- shown for the 2025 subject, 7 variable EURAMOS osteosarcoma data that a simple Cox model is suitable,
- supported these results with a novel simulation approach,
- demonstrated the advantages and pitfalls of the individual methods,
- added to the discussion on appropriate performance measures in ML and proposed a modification to an existing measure, demonstrating the relevance of the modified Kullback-Leibler score for neural networks,
- demonstrated the relevance of visual inspections of prediction plots,
- shown the potential of random survival forests for distinguishing patients on risk, but cast doubt on the reliability of individual survival predictions,
- identified relevant avenues of investigation in the application of neural networks and random survival forests.
Discussion

In this thesis the potential of two machine learning methods - neural networks and random forest - were examined for their potential to model the EURAMOS-1 osteosarcoma data. The performance of both methods was compared to that of the Cox proportional hazards model, which is a well-established way of modelling time to event data. The Cox model has the advantage of easy implementation and straightforward interpretation, but it imposes the assumption of proportional hazards. While time-dependencies can be modelled, they require a priori specification, which is an arduous task when the dependency is complex and cannot be represented by a simple transformation. In the same way, any interactions between variables need to be manually specified, which can be problematic in the presence of many variables and multi-way interactions. Machine learning methods, such as neural networks and random forest, are able to accommodate such relationships implicitly, and have been widely used in classification and regression problems. Due to the uncertainty introduced in time-to-event data by censored observations, the extension of these machine learning methods to survival analysis is not straightforward, and over the years a variety of methods has been proposed. In Chapters 3 and 4 the particulars of this are discussed for neural networks and random forests, respectively. For both methods, a single approach has been selected for the data application. For fitting the neural network, Biganzoli's PLANN model [10] was chosen, for the random forest, Ishwaran's random survival forest (RSF).[19]

The methods were applied to the EURAMOS osteosarcoma data directly, and subsequently to 1000 simulated datasets. The EURAMOS data, as examined here, consists of 2025 subjects with observations for 7 variables: histological response (good, poor), presence of lung metastases (no, yes/possible), presence of other metastases (no, yes/possible), surgical excision (wide/radical, marginal, intralesional/other), sex (male, female), absolute tumour volume, and age. Follow-up time is defined in years since surgery and ranges from 0.005 to 8.9 years, with 73% of the observations censored and most events occurring in the first half of the study. The data simulation was
designed to mimic the characteristics of the original data. The categorical covariates were simulated by replicating the frequency of combinations as observed in the data. For each unique categorical covariate combination, the continuous covariates were simulated from the observed means and covariance matrices, the latter used to account for in-variable spread and inter-variable dependency. Survival times were simulated from a log-normal model, this choice motivated by the need to ensure that the data would not be predisposed a priori to any of the considered methods.

A variety of performance measures were used to evaluate model performance. The concordance index relies on comparing the ordering of observations in observed survival times with the ordering in prognostic index and ensemble mortality, for the Cox model and random survival forest model, respectively. The Brier and Kullback-Leibler scores are calculated for different times, and gives a measure for the deviation from predicted survival probability to observed status. The choice was made to introduce a modification to the Kullback-Leibler score, to enable calculation in the event of extreme predictions. Ordinarily, when survival probabilities of 0 or 1 are predicted, the scores cannot be computed. By imposing a slight increase and decrease, respectively, on these extreme predictions, score values can be obtained, which are distinctly higher than those calculated for intermediate survival predictions. With this modification the Kullback-Leibler score serves as an indicator of unrealistic predictions. The concordance index, Brier and Kullback-Leibler score are used to evaluate the model performance of the Cox and random survival forest models. Neural network models are assessed solely with the Brier and Kullback-Leibler scores, as a concordance index can only be calculated when there is a natural ordering of subjects. This quality is absent from a neural network, where one subject may have a comparatively higher predicted survival at one time, yet the the reverse may be true at another time. For neural networks several additional measures were considered - accuracy, sensitivity and specificity, which were determined by comparing the predicted survival probability at the observed survival time to the censoring status. It was, however, shown that these measures vary little over different models and lead to improper model choices when used as selection criteria. For the Cox model and random forest model two additional measures were considered. The prognostic index and ensemble mortality, respectively, were used to calculate a hazard ratio and logrank $\chi^2$-statistic. The former is obtained by a simple Cox regression of the normalised prognostic index/ensemble mortality scores, the second by comparing the survival curves of high and low risk patients, where the respective categories are determined by the median score value.
While overall an increase in C-index is associated with an increase in these two measures it is not a strict association, and a comparatively high variability is found in these measures. In contrast, the C-index, Brier score and Kullback-Leibler score were found to be in accordance with each other, and are judged as suitable performance measures - the C-index as a well-used overall indication of performance, the Brier and Kullback-Leibler scores to assess performance over time.

The EURAMOS data was split into a training set, containing two thirds of the observations, and a test set containing the rest of the data, while conserving the event/censoring ratio. A five-fold cross-validation was performed on the training set with the purpose of tuning the neural network and random forest model. Final model performance was evaluated on the test set. In the simulation a leave-1%-out cross-validation approach was used to obtain cross-validated performance measures. For both the neural network and random survival forest models, the parameters obtained from the cross-validation process were used in the simulation. A neural network with 5 hidden nodes and 0.05 weight decay was specified, and a random survival forest with $mtry=2$, $nsplit=3$, $nodedepth=7$.

The same general pattern in performance measure behaviour was observed when applying the methods to the EURAMOS osteosarcoma data and when applying them to simulated data, indicating that the simulation process accurately captures the nature of the original data. Overall, for the Cox model and random survival forest model almost identical behaviour was observed, with similar C-indexes and similar Brier and Kullback-Leibler score patterns over time. For both methods a gradual increase of the scores over time was observed. For neural networks the Brier and Kullback-Leibler scores were initially close to those of the other two models, but become noticeably lower in the second half of the follow-up. Notably, in the simulation larger standard errors were present for the neural network scores, most distinctly for the Kullback-Leibler scores. On closer examination, for a subset of the simulated datasets, inflated values were observed - evidence of extreme predictions, which in neural network context indicates an unstable model.

Instability in machine learning models can be considered to be of two flavours. The first is an acceptable and necessary part of the very way the models function, and is born from the inherent randomness in the methods. Consider the neural network, where weights are randomly initialised and the final parameter values are found on convergence of the specified algorithm - here, when the cross-entropy error falls below
a certain predefined limit. In the same way, a random forest also relies on a chance process - starting with the very first algorithm step, where bootstrap samples are drawn, then subsequent steps in which the tree is grown, and random selections of variables and potential splitting points are made. Stability and good performance are achieved by specifying appropriate hyper parameter values - mtry, nsplit, nodesize and nodedepth for a random forest; hidden node number and weight decay for a neural network. Incorrect specification, or, alternatively, the use of data unsuited to the method in question, will give rise to a second kind of instability. In this situation, on repeatedly fitting models with the exact same parametrisation and data, distinctly different results will be observed, rather than the small variations that are expected and considered acceptable in machine learning approaches.

In the cross-validation tuning process, performed on the EURAMOS osteosarcoma data, this second kind of severe instability was observed most strongly for the neural network, in the absence of weight decay or on specification of insufficient weight decay. A final model with 5 hidden nodes and a weight decay of 0.05 was chosen. For this particular weight decay value no difference in performance was observed for varying node sizes, and a 5 node network was selected as a safe intermediate choice. For the purpose of the simulation, the same parametrisation was retained. While the same general score pattern is observed for the simulated data as for the original data, the wide spread in Kullback-Leibler scores in the former, suggests that such a network is not suitable for all simulated datasets, and that performance is sensitive to minor variations in data. Each dataset ideally requires a custom parameter tuning procedure, in which the modified Kullback-Leibler score is accorded special attention to identify unstable networks.

For the random survival forest simulation the parametrisation used for the EURAMOS data was also retained, and, in contrast to the neural network - displayed consistent performance across the simulations. While performance varied for different parameter combinations in the parameter tuning process, no instability was observed for individual forests. In previous literature random forests have been described as comparatively stable methods, which instability only occurring in the event of data with smaller sample sizes and a greater number of predictors. In such an instance, the issue can be mitigated by increasing the number trees in the forest. In the data here, which is a clinical dataset with a limited number of predictors (7) for over 2000 observations, this is not a relevant concern. Indeed, for most types of data random forests are very stable when compared to other machine learning approaches such
as neural networks and support vector machines - stable to the extent that a forest can function without explicit parameter tuning. This property was verified for an untuned forest grown from the EURAMOS data, where the lack of tuning resulted in a very small decrease (0.02) in C-index. While within-model instability is not an issue in random survival forests, substantial differences in predictions are observed on varying parameter values. Of note is that the variation in prediction pattern is not reflected in changes in C-index, Brier scores or Kullback-Leibler scores. The cross-validation process initially resulted in a model with $mtry=2$, $nsplit=3$, $nodesize=200$. On examining predicted survival curves, however, a clustered patterned was observed, which was in part mitigated by leaving $nodesize$ unrestrained, motivating the choice of a model without a specified $nodesize$. On further investigation of parameter behaviour, three of the four parameters were found to be relevant: a large $nodesize$, a small $nodedepth$, and a small $mtry$ all were associated with aberrant prediction patterns in the form of increased survival curve clustering. $Nodesize$ and $nodedepth$ both regulate tree growth, the former by imposing a limit to the maximum number of node splits, as measured from root node to terminal node, the latter by assigning a minimum number of observations per terminal node. The parameter $mtry$ specifies the number of variables randomly selected as candidates for splitting variables, and the inclusion of this parameter distinguishes random forests from bagging. Reducing the number of candidate variables adds an additional source of randomisation to the forest algorithm, decreasing the generalisation error. All three parameters enact regularisation, and the previously described observations suggest an at least tangential connection between increased regularisation and prediction clustering. Notably, increased regularisation and clustering are also associated with marginally higher C-indexes, and as such increased model performance. The variation is minor however, and two nearly identical values can be associated with wildly different prediction patterns. As the C-index relies on the order of observations, the absolute distance between them is irrelevant, and consequently an equal, or indeed higher, C-index value can be accorded to a model which makes clustered predictions. The Brier and Kullback-Leibler scores were observed to follow the same pattern as the C-indexes (explained in detail in section 4.4.4), indicating that none of the considered performance measures are capable of indicating the presence of clustered predictions. In previous literature (and bell sth), when comparing random survival forests to other models, they are tuned and evaluated on C-index only. No report is made of evaluating the models on predictive pattern, visual or otherwise.
A concern unique to the random survival forest approach is that of out-of-bag (OOB) estimation. As a random forest relies on taking the ensemble of trees grown from a number of bootstrap samples only a subset of the subjects will be included in each tree. The excluded subjects are referred to as OOB. An OOB estimate for a subject is obtained by using only those trees for which the subject is out-of-bag. In context of the 0.632 bootstrap used here, a subject will be on average present in 63.2% of the trees and excluded from 36.8%. OOB estimation has been reported as a way to replace cross-validation, and is an attractive option as it requires only a single model to be fitted, rather than the $k$ models of a $k$-fold cross-validation, reducing computation time considerably. In the context of regression and classification forests, however, bias in the OOB error has been reported. In the simulation study no difference was observed between the performance measures (C-index, Brier scores, Kullback-Leibler scores) obtained from OOB estimation and those obtained from the leave-1%-out cross-validation. In order to explain the consistency between the results from the two approaches, the conceptual differences should be considered in more detail. In OOB estimation approximately one third of the trees in the forest is used to estimate the survival probabilities/ensemble mortality score. In a cross-validation an entire forest is constructed using all subjects except those for which an estimate is to be obtained (here, 1% of the data). In the latter case, given that the cross-validation is sufficiently fine, a more precise estimate will be obtained. As the approaches were found to give practically identical performance measure estimates, this suggests that a mere third of the trees is sufficient for obtaining an valid estimate. Indeed, in Section 4.4.2, where the OOB error was examined under different forest sizes, ranging from 1 to 1000 trees, the error was found to stabilise early on, from 250 trees onwards. This suggests that given a large enough forest, OOB estimation can replace cross-validation without ill consequence.

A drawback that needs to be considered is that in the event of extreme and rare observations there is an increased chance of such observations not being present in the trees used for OOB estimation, as they only comprise approximately a third of the forest. By its very structure, a decision tree, and by extent a random survival forest - is only able to predict for observed classes, here for observed event times. Indeed, when calculating OOB Brier and Kullback-Leibler scores on the simulated data, for approximately 10% of the datasets the final time of interest (11 years after surgery) could not be computed.

When looking at the EURAMOS osteosarcoma data itself, estimates were only obtained for the first six years after surgery, this limit imposed by the event time
distribution in the training set, for which the latest event time was 6.9 years. Consequently, test set predictions could only be made for a limited time period. In the latter case, this could have been partly mitigated by ensuring similar event time distributions in both training and test set. In a bootstrap context this could also be achieved, by using a weighted approach in which rare observations are drawn more frequently. As commented on by Ishwaran [19], however, this makes fair OOB estimation impossible.

While OOB error versus cross-validation error has been investigated for classification and regression forests, this has not been done for random survival forests specifically. The simulation results suggests that for the EURAMOS osteosarcoma data OOB error can replace cross-validation error, when assessing model performance. This however cannot be assumed to hold for other datasets, which may have a different sample size, censoring percentage, and number of variables. Another thing to consider is that for regression and classification forests a dependency has been reported between the \( \textit{mtry} \) parameter and the OOB error bias, which may affect the tuning process. For the tuning cross-validation performed on the EURAMOS data no comparison was made between the two errors, and the possibility of bias was not investigated. It would be relevant to perform a study of the two errors for a range of datasets with the purpose of establishing whether in a survival context OOB estimation can be used both in model tuning and in final model evaluation. Alternatively, when analysing a given dataset, a small-scale comparison between the two errors could be made. Should OOB estimation prove reliable, this will much facilitate the process of fitting a RSF model.

In the previous paragraphs, the models were compared on performance measures, and the various issues relating to these measures were discussed. Having considered performance measure behaviour, the next comparison to be made is on which predictors the respective methods identify as important. In contrast to the Cox proportional hazards model, straightforward measures of variable importance are not available for neural networks or random forests. For the latter two various options are available and the following have been considered; For the neural network Oldens connection weight approach [28], for the random survival forest VIMP and minimal depth [19]. Both the Cox model and the neural network identified the presence of lung metastases, histological response and the presence of other metastases as the strongest predictors, in that order. The random forest model identified histological response, the presence of lung metastases and tumour volume as most important predictors.
A possible explanation for the difference with the other methods may be found in the way a random forest is grown. At each node a random forest makes a random subsection of variables from which the final splitting variable is chosen. This ensures that less strong predictors are also given the opportunity to be present in the model, which can potentially result in relevant interactions being revealed that otherwise, if no preselection was made, would be lost. This approach may result in a weaker split on a node level, but in an overall better prediction performance.

Even though random forest and neural networks both, in their own way, accommodate variable interactions and non-linear effects, the exact effect cannot be quantified. For neural networks however, in a survival context, there may be a way to gain additional information on the predictor contribution. The approach applied in this thesis is Biganzoli’s PLANN method, which relies on a data transformation to represent censoring. This data is fed into a neural network with a single hidden layer, containing a number of hidden nodes, and a single output node. Predictions are made in longitudinal format and a second transformation is necessary to obtain a prediction vector per patient. Variable importance for a neural network is quantified by considering the weights from the input to the output. As only a single output node is present, only a single measure for variable importance can be obtained per predictor.

Now consider an alternative approach, which, rather than having one output has \( k \) outputs specified for the \( k \) relevant time intervals. Liestol offers such an approach, which is near identical to Biganzoli’s, and also estimates the conditional hazard, but through a modification of the loss function allows for the specification of targets with varying lengths, with the length corresponding to the number of time intervals in which a given subject is observed. Such an implementation would offer the theoretical advantages of Biganzoli, but through multiple output nodes would allow one to track the variable importance over time, giving an indication of the shape of time dependencies. Note however, that there is no implementation available in R-software, and as such this would require manually specifying a neural network and explicitly defining the modified loss function. Liestol’s approach is one of many approaches that define \( k \) different outputs. Section 3.2 gives a detailed review of the various ways that have been proposed for the modelling of survival data in a neural network framework. They can be distinguished on number of output nodes, as has been just done, but also on the quantity that is estimated and the way censoring is dealt with. For the latter there are two options available- either, as per Biganzoli and Liestol, the observations are truncated after censoring, alternatively some manner of imputation is used for
the censored observations. Street proposed a method which estimates survival probabilities directly, and imputes censored observations with Kaplan Meier estimates. It would be relevant to examine model performance for these approaches on different datasets, and especially under different amounts of censoring. A comparison would be made between models that estimate hazard and those that estimate survival directly, and models that deal with censoring via truncation versus those that use imputation. For the models defining \( k \) output nodes a comparison could be made not only on performance measures, but also on variable importance over time.

In context of all these neural network approaches it is important to note that they rely on defining time intervals. In the application and simulation month-long intervals were defined. It should be considered that this may introduce a bias in the performance measure values when comparing them to those obtained from Cox or RSF. This is another area that requires investigation. In literature, for survival neural networks, typically larger intervals are defined of 1 or more years. Some authors recommend defining intervals of varying lengths with equal numbers of observations, others implement various weighting approaches (Section 3.2).

Overall, a substantial number of approaches has been described for modelling survival data with neural networks, yet no consistent, in-depth application comparison has been made, as has been done in this thesis for Biganzolis neural network and Ishwarans random survival forest. There is insufficient information on how all these different neural network approaches behave relative to each other, under different amounts of censoring, and relative to the Cox proportional hazards model. In this thesis two performance measures are considered to be relevant indicators of neural network performance - the Brier score and Kullback-Leibler score. As detailed previously, due to the absence of a natural ordering amongst subjects a C-index cannot be computed. In previous literature, an artificial ordering has been reported. For example Gensheimer ranked the observations on predicted 1-year survival.\[43\]

In this thesis, an extensive comparison between the Cox model and two machine learning approaches has been made. The Cox model and random survival forest model were shown to have very similar performance, while the neural network was shown to be superior later on in the follow-up, as measured by the Brier and Kullback-Leibler score. Three main performance measures were considered - the concordance index, the Brier score and the Kullback-Leibler score - and they were observed to be consistent with each other in assessing model performance. Several issues are associated with the
neural network and the random survival forest both. The neural network is given to instability, which is indicated by large values of the modified Kullback-Leibler score, and can be verified by refitting a model multiple times and observing the variation of the prediction curves and variable importance. Another consideration is that for fitting the network by necessity time intervals are defined and as such predictions are made on a monthly basis rather than for exact times. The random survival forest was found to be stable in-model, and even to give an acceptable performance in the absence of parameter tuning, but on varying parameter values vastly different prediction curves were observed across forests, which was not reflected in any of the considered performance measures. Additionally, the RSF approach is limited to making predictions for observed event times, which will limit the prediction window if most events take place early on in the study - a not infrequent occurrence in cancer studies. For both neural networks and RSF the interpretation of variable importance is limited. Both methods can in theory accommodate complex interactions and non-linear relationships, but they cannot be accurately quantified - the gain in flexibility is accompanied by a loss of interpretability. In light of the observed results, a final question to ask is whether, for this particular data, it is of added value to consider machine learning approaches. For the purpose of answering this question a number of observations are relevant. Firstly, an informal comparison can be made with the model of chapter 2 - a dynamic prediction model in which a time-dependent effect was observed to be significant for the histological response predictor, but for none of the other variables. Secondly, when fitting neural network it was observed that models with only 1 and 2 hidden nodes had a performance almost identical to that of the final 5-node network, indicating an absence of distinct non-linearity. Finally, when comparing the machine learning methods to Cox, a improved performance was only observed for the neural network. This observation, however, should be interpreted with caution, and more extensive research into survival neural networks is necessary.

When comparing the three models on ease of use, the Cox model is the most easy one to implement. Random survival forests are more difficult, as parameter tuning is required. Using OOB estimation rather than cross-validation will considerably facilitate the process. Neural networks for survival data do not have a ready made implementation available in R, and require the performance of a range of manual steps before and after the network fitting to obtain the relevant survival probabilities. When considering computational time the Cox model is the most efficient, followed by the neural network, with the random survival forest being the most time intensive of the three approaches.
Having considered the three methods in regard to each other, and having examined their possible pitfalls and discussed their relevance in context of the EURAMOS osteosarcoma data, it is perhaps also of interest to consider the findings of this thesis in light of the more general debate concerning machine learning (ML) versus the more traditional statistical modelling (SM) in medical research. Two main topics of discussion can be distinguished within this debate. The first focuses on the correctness of ML implementation, with a frequent point of critique being insufficient validation, incorrect parameter tuning, and the use of unsuitable performance measures [1, 12]. This thesis gives a consistent and critical evaluation of two ML methods versus the Cox proportional hazards model, supporting the conclusions with a simulation study, and offering an in-depth study of various performance measures. The concordance index, Brier score and Kullback-Leibler score were identified as reliable and consistent performance measures, suitable for model assessment and parameter tuning. In contrast, accuracy, sensitivity and specificity were found to be unsuitable for neural network assessment. Here, accuracy was calculated by comparing the predicted survival probability for the observed survival time with the observed status, using a 0.5 cut-off to classify predictions as events and non-events. In previous literature, a similar approach has been reported for assessing the accuracy of the 5-year predictions of a survival neural network [48]. In this thesis it was shown that the three measures are unsuitable for model selection, giving rise to distinctly unstable models, even though excellent accuracy values were measured. Such a verification of the suitability of a performance measure is frequently absent from literature. Authors are quick to use the term ”accuracy” for a performance measure, yet no clear definition of the measure is given, nor is the validity of the choice investigated [59, 12]. This is a considerable oversight, which compromises both the parameter tuning process and the evaluation of the final model performance.

The second topic in the ML versus SM debate is concerned with the the purpose of the analyses, the types of data used, model interpretability and intervention potential. While over the last 10 years a substantial increase in machine learning articles in medical research can be observed, there are few discoveries actually implemented in clinical practice. This is partly due to validation issues, discussed previously, but is also caused by the interpretability - or lack thereof - of machine learning approaches. Both methods considered in this thesis - neural network and random survival forest - are able to accommodate complex interactions and non-linearities. While various measures exist to get an overall idea of the importance of a predictor, an exact
relationship cannot be defined. Consider the observation made for the predictor absolute tumour volume in this study. The random survival forest model identified it as a moderately important predictor, yet this was not found by the neural network, nor the Cox model. The effect the RSF model measures could certainly be valid, and could be explained by supposing some kind of interaction effect, non-linear time effect or even a combination of the two for the predictor. Yet when evaluating a patient with a relatively high tumour volume, in absence of the exact model, and measurements for the other variables, it cannot be said whether this patient would be at increased risk or not. With an increase in variables and model complexity this becomes an increasingly pertinent issue. In the absence of clear interpretation, translating ML findings to something concretely actionable in the clinical field is difficult. The black box nature of ML will also complicate the detection of potential biases.

In a survival analysis context it should be noted that while methods have been adapted to deal with right-censored data, there is a lack of research on what happens in the event of left- or interval-censoring, truncation, or in the situation of a dependency between the survival and censoring distributions.

Another issue to be considered, in light of the observations made in this thesis, is to what extent machine learning methods are suitable for predicting survival probabilities on a patient level. The limitation of the random survival forests in this regard, discovered through a visual inspection of the predicted survival curves, has been discussed extensively. Whether similar issues may be present in other ML methods has not been established. Were one to err on the side of caution, the recommendation would be to use ML chiefly for risk assessment - for distinguishing between high and low risk patients, and potentially identifying patients that would benefit from additional monitoring in their disease progression.

A final and fair point of critique that has been made in the ML versus SM debate is that most ML comparisons are made against the most naive SM model possible, and that due consideration is not given to non-linearities and interactions that can be incorporated manually - for instance by the addition of splines to the Cox model [12, 4]. This thesis is admittedly guilty of comparing the ML methods to a simple Cox regression, yet there is a strength in showing that for this particular kind of osteosarcoma data even a simple Cox model does not under-perform when compared to the evaluated ML models. In a next step, however, it would indeed be relevant to include a more complicated model into the comparison. For this particular data, it would not be unreasonable to think that such a model would outperform the ML alternatives.
Overall, in order to comprehensively establish the potential of ML for survival analysis of clinical data, there is a need for increased research, which is consistent and transparent in approach, and uses appropriate performance measures to evaluate model performance.
Appendix A

Data description

A.1 Consort diagrams

A.1.1 Consort diagram: EURAMOS trial

Information for the consort diagram was obtained from the raw data. Information on histological response is contained both in the histological response variable hresp1 and the treatment variable trt, as treatment is dependent on randomization, which in turn is dependent on histological response. The variable hresp1 has 248 missing entries, and for these missing values information on treatment is used to supplement. Below, good and poor responders are identified from both variables. Patients who were incorrectly randomized are identified, with the purpose of creating a variable accurately representing histological response. 27 missing entries in histological response (hresp1) are filled in using treatment information.

```r
# Identifying good responders: information available from treatment
# and from hresp1
ind.trt.g.r <- which(data$trt=="MAP×(good responder)" | data$trt=="MAPIFN×(good responder)")
ind.hist.g <- which(data$hresp1==
"Good response <10% viable tumour")
ind.good.resp <- unique(c(ind.trt.g.r, ind.hist.g))

# Identifying poor responders
ind.trt.p.r <- which(data$trt=="MAP×(poor responder)" | data$trt=="MAPI×(poor responder)")
ind.hist.p <- which(data$hresp1==
"Poor response ≥10% viable tumour")
ind.poor.resp <- unique(c(ind.trt.p.r, ind.hist.p))
```
17 # A variable is created summarizing the randomization of patients.
18 rand <- rep(0, nrow(data))
19 rand[which(data$trt=="Not randomised")]<-1
20 rand <- as.factor(rand)
21 levels(rand) <- c("Randomised", "Not randomised")
22 sum(summary(rand))
23
24 # misassigned randomisation for good/poor responders
25 trouble_cases.tt <- unique(c(ind.good.resp[which(ind.good.resp%in%ind.poor.resp)],
                              ind.poor.resp[which(ind.poor.resp %in%
                              ind.good.resp)]))
26
27 # 5 good responders randomised as poor responders
28 ind.dat.hresp.good <- which(data$hresp1==
                              "Good response<10% viable tumour")
29
30 # 7 poor responders randomised as good responders
31 ind.dat.hresp.bad <- which(data$hresp1==
                             "Poor response>=10% viable tumour")
32
33 # Making variable with correct assignments
34 ind.miss.good <- ind.dat.hresp.good[which(which(data$hresp1==
                                          "Good response<10% viable tumour")
                                          %in% trouble_cases.tt)]
35 ind.comp.good <- ind.good.resp
36
37 ind.miss.poor <- ind.dat.hresp.bad[which(which(data$hresp1==
                                          "Poor response>=10% viable tumour")
                                          %in% trouble_cases.tt)]
38 ind.comp.poor <- ind.poor.resp
39
40 ind.comp.good <- ind.comp.good[\-which(ind.comp.good
                                  %in% ind.miss.good)]
41 ind.comp.poor <- ind.comp.poor[\-which(ind.comp.poor
                                     %in% ind.miss.good)]
42
43 index.hist.resp <- seq(1,nrow(data))
44 index.known <- which(index.hist.resp %in%
                        c(ind.comp.good, ind.comp.poor))
45 index.unkown <- index.hist.resp[\-index.known]
46
47 index.hist.resp[ind.comp.good] <- 0
48 index.hist.resp[ind.comp.poor] <- 1
\texttt{index.hist.resp[index.unkown] <- NA}

\texttt{hist.resp <- \texttt{as.factor(index.hist.resp)}}

\texttt{levels(hist.resp) <- c(”Good\_hist”, ”Poor\_hist”, NA)}

\texttt{# Summary information consort diagram}

\texttt{count.pats <- \texttt{nrow(data)}}

\texttt{count.no.biop <- \texttt{length(\texttt{which(data$who1==””})}}

\texttt{count.no.surg <- \texttt{length(\texttt{which(is.na(data$dosurg)})}}

\texttt{count.no.preop <- \texttt{length(\texttt{which(is.na(data$doprc)})}}

\texttt{count.no.hist <- \texttt{length(\texttt{which(data$hresp1==””})}}

\texttt{count.rand <- \texttt{length(\texttt{which(rand==”Randomised”})}}

\texttt{count.not.rand <- \texttt{length(\texttt{which(rand==”Not\_randomised”})}}

\texttt{count.wrong.rand.g <- \texttt{length(ind.miss.good)}}

\texttt{count.wrong.rand.p <- \texttt{length(ind.miss.poor)}}

\textbf{A.1.2 Consort diagram: eligibility for analysis}

\texttt{inf.fail <- \texttt{which(data$doevent <= data$dosurg)}}

\texttt{# no surgery date}

\texttt{inf.surg <- \texttt{which(is.na(data$dosurg)}}

\texttt{# no follow-up}

\texttt{inf.na <- \texttt{which(is.na(data$dofu) \& is.na(data$locrec) \& is.na(data$locrec_1)}}

\texttt{# disease-dependent non-randomization}

\texttt{inf.not.rand.w <- \texttt{which(data$ranin== ”Progression\_of\_metastatic\_disease\_or\_new\_metastatic\_disease” \& data$ranin ==”Unresectable\_disease\_or\_primary\_metastatic\_or\_both”)}}

\texttt{ind.total.rem <- \texttt{unique(c(inf.fail, inf.surg, inf.na, inf.not.rand.w))}}

\texttt{data.t <- data[-ind.total.rem,]}

\textbf{A.2 Events}

\texttt{# index LR, NM and LR+NM as first event}
ind.LR <- c(which(data.t$s$locrec==2),
   which(data.t$s$locrec==6 & data.t$s$multev==2))
ind.NM <- c(which(data.t$s$locrec==3),
   which(data.t$s$locrec==6 & data.t$s$multev==3))
ind.LRN <- c(which(data.t$s$multev==23))
# index LR, NM and LR+NM as second event
ind.LR2 <- c(which(data.t$s$locrec_1==2),
   which(data.t$s$locrec_1==6 & data.t$s$multev_1==2))
ind.NM2 <- c(which(data.t$s$locrec_1==3),
   which(data.t$s$locrec_1==6 & data.t$s$multev_1==3))
ind.LRN2 <- c(which(data.t$s$multev_1==23))
length(which(ind.LRN2 %in% ind.NM))  #77
# event table
ev.tab <- rbind(cbind(length(ind.LR), length(ind.NM),
   length(ind.LRN)), cbind(length(ind.LR2), length(ind.NM2),
   length(ind.LRN2)))
colnames(ev.tab) <- c("LR", "NM", "LR+NM")
rownames(ev.tab) <- c("First event", "Second event")
# index pooled death as first event
death.ev1 <- which(data.t$s$locrec==1 | data.t$s$locrec==4 |
   data.t$s$locrec==5, data.t$s$multev==24 | data.t$s$multev==14 |
   data.t$s$multev==34 | data.t$s$multev==234 | data.t$s$multev==25 |
   data.t$s$multev==35)
# index pooled death as second event
death.ev2 <- which(data.t$s$locrec_1==1 | data.t$s$locrec_1==4 |
   data.t$s$locrec_1==5, data.t$s$multev_1==13 | data.t$s$multev_1==14 |
   data.t$s$multev_1==34 | data.t$s$multev_1==134)  
deadth.ev2 <- death.ev2[!which(death.ev2 %in% death.ev1)]
# index variable for death due to osteosarcoma at final follow-up
death.final <- which(data.t$s$surv==2)
death.final <- death.final[!which(death.final %in% c(death.ev1, death.ev2))]
length(death.final)
# table pooled death
death.tab <- rbind(c(table(data.t$s$locrec)[c(1,4,5)]),
A.3 Variables

# Pooling variables
Hist.resp <- histresp[-ind.total.rem]

age1 <- (data.t$dosurg-data.t$dob)/365.25
age <- vector("numeric", nrow(data.t))
age[which(age1<12)] <- "<12 years"
age[which(age1>=12 & age1<18)] <- "12-18 years"
age[which(age1>=18)] <- ">=18 years"
age <- as.factor(age)
age <- as.data.frame(age)

excision <- vector("numeric", nrow(data.t))
excision[which(data.t$excis1=="Marginal")] <- "Marginal"
excision[which(data.t$excis1=="Intralesional")] <- "Other"
excision[which(data.t$excis1=="

data.t$excis1=="Not known")]<-"Other"
excision[which(data.t$excis1=="Radical" |
data.t$excis1=="Wide")]<-"Radical/wide"
excision <- as.factor(excision)
excision <- as.data.frame(excision)

volume <- data.t$avol
volume[which(data.t$avol<200)] <- "<200"
volume[which(data.t$avol>=200)] <- ">=200"
volume <- as.factor(volume)
volume <- as.data.frame(volume)

lung.met <- as.character(data.t$lmetreg)
ind.lmet.comb <- which(data.t$lmetreg=="Yes" |
data.t$lmetreg=="Possible")
lung.met[ind.lmet.comb] <- "Yes/Possible"
lung.met <- as.factor(lung.met)
lung.met <- as.data.frame(lung.met)

other.met <- as.character(data.t$ometreg)
ind.omet.comb <- which(data.t$ometreg=="Yes" |
data.t$ometreg=="Possible")
other.met[ind.omet.comb] <- "Yes/Possible"
other.met <- as.factor(other.met)
other.met <- as.data.frame(other.met)

sex <- as.data.frame(data.t$sex)

var.set <- cbind(Hist.resp, age, volume,
excision, sex, lung.met, other.met, rand)

# Tables patient characteristics
rownames.n.table <- c("Age", "12-18", "<12", ">18", "Missing",
"Histological response", "Good(<10% tumor)",
"Poor(>=10% tumor)", "Missing",
"Wide/radical", ",Marginal", "Other",
"Sex", "Female", "Male", "Missing",
"Lung metastases", "No", "Yes/Possible",
"Missing", ",Other metastases", "No",
"Yes/Possible", "Missing")

N <- nrow(var.set)

var.set$excision <- relevel(var.set$excision,"Radical/wide")

table <- c("", as.vector(summary(var.set$age)),0,"",
as.vector(summary(var.set$Hist.resp)), ",",
as.vector(summary(var.set$excision)),0,"",
as.vector(summary(var.set$volume)), ",",
as.vector(summary(var.set$sex)), 0,"",
as.vector(summary(var.set$lung.met)), 0,"",
as.vector(summary(var.set$other.met)),0)

table.2 <- c("", round(as.vector(summary(var.set$age))
/N*100, 0),0,"",
round(as.vector(summary(var.set$Hist.resp))
/N*100, 0), ",",

202
round(as.vector(summary(var.set$excision)))
/N*100, 0), 0, "", 
round(as.vector(summary(var.set$volume)))
/N*100, 0), ", ",
round(as.vector(summary(var.set$sex)))
/N*100, 0), 0, "", 
round(as.vector(summary(var.set$lung.met)))
/N*100, 0), 0, "", 
round(as.vector(summary(var.set$other.met)))
/N*100, 0)

index.rand <- which(var.set$rand=="Randomised")
index.n.rand <- which(var.set$rand=="Not randomised")
length(index.rand) + length(index.n.rand)
data.rand <- var.set[index.rand,]
data.n.rand <- var.set[index.n.rand,]

# Tables patient characteristics, randomised
table.r <- c("", as.vector(summary(data.rand$age)), 0, "", 
as.vector(summary(data.rand$Hist.resp)), ", "
as.vector(summary(data.rand$excision)), 0, "", 
as.vector(summary(data.rand$volume)), ", "
as.vector(summary(data.rand$sex)), 0,"", 
as.vector(summary(data.rand$lung.met)), 0,"", 
as.vector(summary(data.rand$other.met)), 0)
table.2.r <- c("",
round(as.vector(summary(data.rand$age)))/
N*100, 0), 0, "", 
round(as.vector(summary(data.rand$Hist.resp)))/
N*100, 0), "", 
round(as.vector(summary(data.rand$excision)))/
N*100, 0), 0, "", 
round(as.vector(summary(data.rand$volume)))/
N*100, 0), "", 
round(as.vector(summary(data.rand$sex)))/
N*100, 0), 0, "", 
round(as.vector(summary(data.rand$lung.met)))/
N*100, 0), 0, "", 
round(as.vector(summary(data.rand$other.met)))/
N*100, 0), 0)
# Tables patient characteristics, not randomised

```r
table.nr <- c("", 
    as.vector(summary(data.n.rand$age)), 0, "", 
    as.vector(summary(data.n.rand$Hist.resp)), "", 
    as.vector(summary(data.n.rand$excision)), 0, "", 
    as.vector(summary(data.n.rand$volume)), "", 
    as.vector(summary(data.n.rand$sex)), "", 
    as.vector(summary(data.n.rand$lung.met)), 0,"", 
    as.vector(summary(data.n.rand$other.met)), 0)

round(as.vector(summary(data.n.rand$age))/N*100, 0),0,"", 
round(as.vector(summary(data.n.rand$Hist.resp))/N*100, 0,"", 
round(as.vector(summary(data.n.rand$excision))/N*100, 0,"", 
round(as.vector(summary(data.n.rand$volume))/N*100, 0,"", 
round(as.vector(summary(data.n.rand$sex))/N*100, 0,"", 
round(as.vector(summary(data.n.rand$lung.met))/N*100, 0,"", 
round(as.vector(summary(data.n.rand$other.met))/N*100, 0,0)
```

n.table <- cbind(table.r, table.2.r, table.nr, table.2.nr, table,table.2)
rownames(n.table) <- rownames.n.table
colnames(n.table) <- c("N", "%", "N", "%", "N", "%")

## Data formatting

```r
# time of event occurrence in years since surgery

time.doevent <- (data.t$doevent - data.t$dosurg)/365.25

time.doevent1 <- (data.t$doevent_1 - data.t$dosurg)/365.25

time.surv <- (data.t$dofu - data.t$dosurg)/365.25

# index variable for death as first event

death.ev1 <- which(data.t$locrec==1 | data.t$locrec==4 | 
data.t$locrec==5, data.t$multev==24 | data.t$multev==14 | 
data.t$multev==34 | data.t$multev==234 | data.t$multev==25 |
data.t$multev==35)

# index variable for death as second event
defeat.ev2 <- which(data.t$locrec_1==1 | data.t$locrec_1==4 | data.t$locrec_1==5, data.t$multev_1==13 | data.t$multev_1==14 | data.t$multev_1==34 | data.t$multev_1==134)
defat.ev2 <- defeat.ev2[!which(defat.ev2 %in% defeat.ev1)]

# index variable for death due to osteosarcoma at final follow-up
defat.fina <- which(data.t$surv==2)
defat.fina <- defeat.fina[!which(defat.fina %in% c(defat.ev1, defeat.ev2))]

# event status death
stat.death <- rep(0, nrow(data.t))
stat.death[unique(c(defat.ev1, defeat.ev2, defeat.fina))] <- 1

# time status death
time.death <- vector("numeric", nrow(data.t))
time.death[defat.ev1] <- time.doevent[defat.ev1]
time.death[defat.ev2] <- time.doevent1[defat.ev2]
time.death[defat.fina] <- time.surv[defat.fina]

# in absence of death, the final date of follow-up is used
time.death[!c(defat.ev1, defeat.ev2, defeat.fina)] <-
time.surv[!c(defat.ev1, defeat.ev2, defeat.fina)]

# event status and time of local recurrence
ind.LR <- c(which(data.t$locrec==2), which(data.t$locrec==6 & data.t$multev==2))
time.LR <- vector("numeric", nrow(data.t))
time.LR[ind.LR] <- time.doevent[ind.LR]
time.LR[!ind.LR] <- time.death[!ind.LR]
stat.LR <- rep(0, nrow(data.t))
stat.LR[ind.LR] <- 1

# event status and time of new metastatic disease
ind.NM <- c(which(data.t$locrec==3), which(data.t$locrec==6 & data.t$multev==3))
time.NM <- vector("numeric", nrow(data.t))
time.NM[ind.NM] <- time.doevent[ind.NM]
time.NM[!ind.NM] <- time.death[!ind.NM]
stat.NM <- rep(0, nrow(data.t))
stat.NM[ind.NM] <- 1
# Problem cases:

# 1: if only one event registered, but no follow-up available
(time event is taken as censoring time)

```r
#data.t [which(is.na(time.death)) ,]
```

```r
prob.case <- which(is.na(time.death))
time.death[prob.case] <- time.NM[prob.case]
time.LR[prob.case] <- time.NM[prob.case]
```

# 2: if event and death occur simultaneously: death takes precedence
(status other events set to 0)

```r
prob.case.simult.death <- which(time.NM==time.death & stat.NM==1 & stat.death==1)
stat.NM[prob.case.simult.death] <- 0
```

# 3: Final follow-up date prior to event

```r
#data.t [prob.case.wrong.follow ,]
```

# LR occurs after final follow up time. On inspection, a second event is recorded after LR event time. As the model does not allow two consecutive events, save when the second one is death, this event time is used as time of final follow-up.

```r
time.NM[prob.case.f] <- time.doevent1[prob.case.f]
time.death[prob.case.f] <- time.doevent1[prob.case.f]
```

```r
comp.dat.1 <- cbind(stat.LR, time.LR, stat.NM, time.NM, stat.death, time.death)
```

Creating final dataset

```r
compp <- cbind(comp.dat.1, var.set[,1:7])
```

```r
compp$excision <- relevel(compp$excision, "Radical/wide")
```

```r
dat.d <- compp
dat.d$id <- seq(1,nrow(dat.d))
```

```r
colnames(dat.d) <- c("stat.LR","time.LR","stat.NM","time.NM","stat.death","time.death","Hist.resp","age","volume","excision","sex","lung.met","other.met","id")
```
Appendix B

A Dynamic Prediction model for osteosarcoma patients

B.1 Data preparation

```r
# Imputing 10 data sets
a.out <- amelia(dat.d, m=10, p2s=0,
    noms=c("Hist. resp", "excision", "sex", "lung.met",
        "other.met", "volume"),
    ords="age",
    idvars=c("stat.LR", "time.LR", "stat.NM", "time.NM",
        "stat.death", "time.death"))

# Function creating long format from wide format (with wide format as defined
# in A.4 Data Formatting)
longCoxmod <- function(checkedData){
    n <- nrow(checkedData)
    tstart <- tstop <- NULL
    length <- vector("numeric", n)
    for(i in 1:n){
        times <- c(checkedData$time.LR[i],
            checkedData$time.NM[i],
            checkedData$time.death[i])
        events <- c(checkedData$stat.LR[i],
            checkedData$stat.NM[i],1)
        eventTimes <- sort(c(0, times[events==1]))
        tstart <- c(tstart, eventTimes[-length(eventTimes)])
        tstop <- c(tstop, eventTimes[-1])
        length[i] <- length(eventTimes[-length(eventTimes)])
    }
}```
d <- checkedData[,]

death <- as.numeric(tstop==death & d$stat.death)
LR <- as.numeric(tstop>LR & d$stat.LR)
NM <- as.numeric(tstop>NM & d$stat.NM)

longCoxData <- data.frame(cbind(d, tstart, tstop, LR, NM, death))
return(longCoxData)

LMdata_imp <-
# Creates landmark datasets for every imputed dataset, returns list, with every
# list element containing all landmark datasets for a given imputation
function (a.out) {
LMdataList <- list()

for (i in 1:length(a.out$imputations)){
  data <- a.out$imputations[i]
  longCoxData <- longCoxmod(data)
  longCoxData$LR <- longCoxData$tstart # for cutLM
  longCoxData$NM <- longCoxData$tstart # for cutLM

  #landmark dynamic prediction data set from long format
  # Landmark datasets are created for every gridpoint 'grd' from 0 to 5 with a
  prediction window 'w' of 5 years of
  w <- 5
  grd <- 3/12
  LMdata <- NULL
  LMs <- seq(0, 5, by=grd)
  for (LM in LMs) {
    print(LM)
  }

  # Landmark datasets are created for one time-varying variable at a time
  # Here it is done for LR
  LMdataLM_LR <- cutLM(data=longCoxData,
    outcome=list(time="time.death", status="stat.death"),
    LM=LM, horizon=LM+w,
    covs=list(fixed=c("Hist.resp", "age", "volume",
                       "excision", "sex", "lung.met",
                       "other.met"),
              varying="LR"),
    format="long", id="id", rtime="time.LR")
}
# Here it is done for NM
LMdataLM_NM <- cutLM(data=longCoxData, outcome=list(time="time.death", status="stat.death"), LM=LM, horizon=LM+w, covs=list(fixed=c(), varying="NM"), format="long", id="id", rtime="time.NM")

# The two are combined
LMdataLM <- cbind(LMdataLM_LR, LMdataLM_NM[, c("NM", "time.NM")])
LMdata <- rbind(LMdata, LMdataLM)

# The smallest time of each landmark dataset is the landmark cutoff point
LMdata$Tstart <- LMdata$LM

# Patients at time zero (time of surgery) cannot have events
idx <- LMdata$LM==0
LMdata$LR[idx] <- LMdata$NM[idx] <- 0
LMdata$time.LR[idx] <- LMdata$time.NM[idx] <-
LMdata$time.death[idx]

# Introduce landmark time variable, and squared landmark time variable
LMdata$LM1 <- LMdata$LM
LMdata$LM2 <- LMdata$LM^2
name <- paste(i)
LMdataList[[name]] <- LMdata

return(LMdataList)

# List of landmark datasets
LMdata.U <- LMdata_imp(a.out)

# Removing unnecessary variables
LMdata.red <-
function(LMdatas){
LMreds <- list()
for(i in 1:length(LMdatas)){
data <- LMdatas[[i]]
datared <- data[, -c(12, 13, 15)]
name <- paste(i)
LMreds[[name]] <- datared
}
return(LMreds)
B.2 Fitting a landmark model

```r
# computes landmark models and heuristic shrinkage factors for dynamic
# prediction using the landmark dataset. This is done for each imputation.
# Returns list of lists with model and heuristic shrinkage factor
imputedModel3 <-
function(LMdatas) {
  cmList <- list()
  for (i in 1:length(LMdatas)) {
    data <- LMdatas[[i]]
    Fitting model on each imputation
    LMsupercox <- coxph(Surv(Tstart, time.death, stat.death) ~
      Hist.resp + Hist.resp*LM1 +
      Hist.resp*LM2 + age +
      volume + excision +
      sex + lung.met +
      other.met + LR +
      DM + LM1 + LM2 +
      cluster(id), data=data, 
      method="breslow")
    Obtaining heuristic shrinkage factor
    chisq5 <- 2 * diff(LMsupercox$loglik)
    p <- length(LMsupercox$coef)
    cheur <- 1 - p/chisq5
    name <- paste(i)
    cmList[[name]] <- list(LMsupercox=LMsupercox, cheur=cheur)
  }
  return(cmList)
}
```

B.3 Pooling estimates

```r
pooling <- function(mdls) {
  # extract models from list with models and heuristic shrinkage factors
  mods <- lapply(1:10, function (i) mdls[[i]][[1]])
  # obtain pooled coefficients
  coefs <- lapply(1:10, function (i) coef(mods[[i]]))
  coefs <- data.frame(matrix(unlist(coefs), nrow=15,
```
byrow=FALSE), stringsAsFactors=FALSE)
mcoef <- apply(coefs, 1, mean)
length(mcoef)

# obtain pooled covariance matrix
m <- 10
vars <- lapply(1:10, function(i) vcov(mods[[i]]))
withinVAR <- Reduce(’+’, vars)/m
betweenVAR <- lapply(coefs, function(x) (x-mcoef) %*% t(x-mcoef))
betweenVAR <- Reduce(’+’, betweenVAR)/(m-1)
totalVAR <- withinVAR + ((m+1)/m)*betweenVAR

# extract pooled standard errors for coefficients
totalSE <- sqrt(diag(totalVAR))

# calculate hazard ratio from coefficients and corresponding CI
HR <- exp(mcoef)
lower <- exp(mcoef-1.96*totalSE)
upper <- exp(mcoef+1.96*totalSE)

# calculate P-values from pooled coefficients and SEs
covp <- mcoef/totalSE
mpv <- vector(”numeric”, length=length(covp))
for (i in 1:length(covp)){
  if(covp[i]<0){
    mpv[i] <- pnorm(covp[i]) * 2
  } else {
    mpv[i] <- pnorm(covp[i], lower.tail=FALSE) * 2
  }
}

# make table
tab2 <- cbind(format(round(HR, 3), nsmall = 3),
paste(format(round(lower, 3), nsmall = 3),”–”
format(round(upper, 3), nsmall = 3)),
format.pval(mpv, eps=0.001, digits =3),mcoef)
colnames(tab2) <- c(”HR”, ”95% CI”, ”P-value”, ”Coef”)
rownames(tab2) <- rownames(as.data.frame(coef(mods[[1]])))
return(tab2)

# with coefficients
coeffs.U.1 <- pooling(imp.mod)

# without coefficients for table
# Function for calculating prognostic indexes for a prediction model 'model',
for a list of imputed datasets 'data', for element 'j' of this list

cvpi <- function(model, data, j){
    ids <- unique(data$id)
    n <- length(ids)
    CVPI <- rep(NA, nrow(data))
    for (i in 1:n){
        select patient i
        id <- ids[i]

        fit model on remaining patients
        cm <- coxph(model$formula, data=data[data$id!=id,])

        obtain model matrix for patient i
        mm <- model.matrix(model$formula, data=data[data$id==id,])

        remove intercept and ID from model matrix, as these are not covariates
        mm <- mm[,!(colnames(mm) %in% c("(Intercept)", "cluster(id)"))]

        Patient covariates * model coefficients gives the prognostic index
        CVPI[data$id==id & !is.na(data$LR)] <- mm %*% cm$coefficients
        if (i %% 100 == 0){
            Save results for every 100 patients
            save(CVPI, file=paste("/Directory/ CVPIU", i))
        }
    }
    Save results for all patients
    save(CVPI, file=paste("/Directory/ CVPIoth", j))
    return(CVPI)
}

Making copy list landmark datasets
LMdatas <- LMdata.U

Calculating prognostic indexes
CVlist <- lapply(1:10, function(i), cvip(mdl, LMdatas[[i]], i))
Attaching prognostic indexes to datasets

```r
for(i in 1:10) {
  LMdatas[[i]]$CVlist <- CVlist[[i]]
}
save(LMdatas, file="LMdatasCV")

# Function for obtaining cropped dataset within time window [t, t+w], for w=5

cropData <- function(LMdata, year) {
  y <- year

  # starting cut-off t is defined by selecting data with landmark time y
  dat <- LMdata[LMdata$LM1==y,]

  # observations contained within window t+5, are selected
  dat <- dat[dat$Tstart < y+5,]  # redundant?

  # observation exceeding t+5 are censored at t+5
  idx <- dat$time.death > y+5
  dat$time.death[idx] <- y+5

  # event indicator for artifically censored observations is set to 0
  dat$stat.death[idx] <- 0
  return(dat)
}

# cropped datasets for prediction time t=0, 1, 2, 3, 4, 5
LMdatas0 <- lapply(LMdatas, function(x) cropData(x, 0))
LMdatas1 <- lapply(LMdatas, function(x) cropData(x, 1))
LMdatas2 <- lapply(LMdatas, function(x) cropData(x, 2))
LMdatas3 <- lapply(LMdatas, function(x) cropData(x, 3))
LMdatas4 <- lapply(LMdatas, function(x) cropData(x, 4))
LMdatas5 <- lapply(LMdatas, function(x) cropData(x, 5))

# function for calculating C-index

cindex.mod <- function(data) {
  time <- data$time.death
  status <- data$stat.death
  x <- data$CV
  n <- length(time)
```
# Order variables on time (ascending), and on status (descending- 1s first)
ord <- order(time, -status)
time <- time[ord]
status <- status[ord]
x <- x[ord]

# Select all individuals who experience an event
wh <- which(status == 1)
total <- concordant <- 0

# Every individual i with an event is compared to all other individuals
# with a later event/censoring time j (event times were sorted, ascending)
for (i in wh) {
    for (j in ((i + 1):n)) {

        # The total number of individuals j with a later event/censoring time
        # than individual i is counted
        if (time[j] > time[i]) {
            total <- total + 1

        # The total number of concordant an tied pairs is counted
        if (x[j] < x[i])
            concordant <- concordant + 1
        if (x[j] == x[i])
            concordant <- concordant + 0.5
        }
    }
}

# The proportion of concordant pairs over the total of evaluable pairs
# gives the C-index
return(concordant/total)

# Calculating C-index for the 6 prediction times for 10 imputations
indices0 <- lapply(LMdatas0, cindex.mod)
indices1 <- lapply(LMdatas1, cindex.mod)
indices2 <- lapply(LMdatas2, cindex.mod)
indices3 <- lapply(LMdatas3, cindex.mod)
indices4 <- lapply(LMdatas4, cindex.mod)
indices5 <- lapply(LMdatas5, cindex.mod)

# Calculating average C-index per prediction time
allCs <- rbind(indices0, indices1, indices2, indices3, indices4, indices5)
cindices5)
colnames(allCs) <- c("Imp1", "Imp2", "Imp3", "Imp4", "Imp5", "Imp6", "Imp7", "Imp8", "Imp9", "Imp10")
cindex0 <- mean(unlist(cindices0))
cindex1 <- mean(unlist(cindices1))
cindex2 <- mean(unlist(cindices2))
cindex3 <- mean(unlist(cindices3))
cindex4 <- mean(unlist(cindices4))
cindex5 <- mean(unlist(cindices5))
Cs <- c(cindex0, cindex1, cindex2, cindex3, cindex4, cindex5)

B.5 5-year death prediction

# Function for 5-year death prediction for patient with predefined characteristics, applied to list of imputed landmark datasets LMdata.U, for patient with LR; ev1="LR", or NM; ev1="NM"
deadth5year <- function(patient, LMdata.U, ev1){
  FW0list <- list()
  FW1list <- list()
  # Predictions are made for each of the 10 imputed datasets
  for (i in 1:length(LMdata.U)){
    LMdata <- LMdata.U[[i]]
    # Model is fitted
    mdl <- coxph(Surv(Tstart, time, death, stat.death) ~ Hist.resp + Hist.resp*LM1 + Hist.resp*LM2 + age + volume + excision + sex + lung.met + other.met + LR + NM + LM1 + LM2 + cluster(id), data=LMdata, method="breslow")
    # estimates survival curve for patient with given characteristics
    sf <- survfit(mdl, newdata=patient)
    # extracting the cumulative hazard
    Haz0 <- data.frame(time=sf$time, surv=sf$surv)
    Haz0$Haz <- -log(Haz0$surv)
}
# defining time points for estimation, starting at 3 months post surgery to 5 years
	tt1 <- seq(3/12,5,by=0.05)

t1 <- length(tt1)

# indicator variable, when 0 no LR/NM
	xdata <- rep(0,nt1)

# prediction is made for patient without LR/NM

Fw0 <- Fwpredict(bet=mdl$coef, Haz0=Haz0, xdata, tt=tt1, marginfree=patient$Hist.resp=="Good_hist", ev=ev1)

	tt2<- seq(3/12,5,by=0.05)

nt2 <- length(tt2)

# indicator variable, when 1 patient has LR/NM
	xdata <- rep(1,nt2)

# prediction made for patient with LR/NM

Fw1 <- Fwpredict(mdl$coef, Haz0, xdata, tt2, marginfree=patient$Hist.resp=="Good_hist", ev=ev1)

name <- paste(i)

FW0list[[name]] <- Fw0

FW1list[[name]] <- Fw1

}

# The means of the predictions over the 10 imputed datasets are taken

FW0mean <- data.frame(time=tt1, Fw=apply(sapply(FW0list, function(x) x$Fw),1,mean))

FW1mean <- data.frame(time=tt2, Fw=apply(sapply(FW1list, function(x) x$Fw),1,mean))

return(list(no=FW0mean, yes=FW1mean))

# Prediction function, gives 5-year death predictions for range of prediction points (tt), in presence absence intermediate event (xdata), for LR or NM (ev).
The hazard is partly given by Haz0 (computed in the death5year function).
The remaining elements of the hazard are given inside the Fwpredict function below
Fwpredict <-

function(bet, Haz0, xdata, tt, marginfree=TRUE, w=5, ev){

# If a good histological response is specified in the patient dataframe,
margin <- ifelse(marginfree, 1, 0)
nt <- length(tt)

# obtaining small hazard from cumulative hazard
Haz0$Haz <- diff(c(0, Haz0$Haz))

# For each time point death probability is computed
Fw <- data.frame(time=tt, Fw=NA)
for (i in 1:nt) {
  sfi <- Haz0 # local copy
tti <- tt[i]

  # the hazard calculated in the death5year function is multiplied with the
time-dependent hazards, and the hazard for LR
  sfi$haz <- sfi$haz *
    exp(bet[ paste(ev) ] * xdata[i] + bet["LM2"] * tt[i]^2
       + bet["LM1"] * tt[i] +
    margin*(bet["Hist.respPoor.hist:LM1"] * tt[i] +
       bet["Hist.respPoor.hist:LM2"] * tt[i]^2))

  # the cumulative hazard is obtained
  sfi$Haz <- cumsum(sfi$haz)

  # the cumulative hazard estimate for the time point and the time point plus
  # the prediction window (5 years) is obtained
  tmp <- evalstep(sfi$time, sfi$Haz, c(tti, tti+w), subst=0)

  # the difference of the cumulative hazards is taken. The exponent of the
  # difference gives the survival prediction, 1-the survival the death prediction
  Fw$Fw[i] <- 1-exp(-(tmp[2]-tmp[1]))
}

return(Fw)

B.6 Results
# Follow up time using reverse kaplan meier

```
rev.k <- as.data.frame(cbind(compp$stat.death, compp$time.death))
colnames(rev.k) <- c("stat", "time")
rev.k[which(rev.k$stat==1),1] <- 2
rev.k[which(rev.k$stat==0),1] <- 1
rev.k[which(rev.k$stat==2),1] <- 0
obj.rev.k <- survfit(Surv(time, stat)~1, data=rev.k)

# median, min and max follow-up time
obj.rev.k$min(summary(obj.rev.k)$time)
max(summary(obj.rev.k)$time)

#event counts
sum(compp$stat.LR)
sum(compp$stat.NM)
sum(compp$stat.death)
sum(compp$stat.death==0)

# Counting number of LR and NM events, and patients at risk per landmark
timepoint
data <- dat.d
w <- 5
grd <- 3/12
LMs <- seq(0,5,by=grd)
attrisk <- haveLR <- haveNM <- NULL
for(LM in LMs){
  idx <- data$time.death > LM
  idx2 <- data$time.LR<LM
  idx3 <- data$time.NM<LM
  attrisk <- c(attrisk, sum(idx))
  haveLR <- c(haveLR, sum(idx&idx2))
  haveNM <- c(haveNM, sum(idx&idx3))
}

# Making barplots
xlb <- LMs
xlb[!(xlb %in% c(0,1,2,3,4,5))] <- NA
lrplot <- rbind(attrisk-haveLR,haveLR)
barplot(lrplot, width=3/12,ylab="Number of patients",
       xlab="Years since surgery",

# Counting number of LR and NM events, and patients at risk per landmark
timepoint
data <- dat.d
w <- 5
grd <- 3/12
LMs <- seq(0,5,by=grd)
attrisk <- haveLR <- haveNM <- NULL
for(LM in LMs){
  idx <- data$time.death > LM
  idx2 <- data$time.LR<LM
  idx3 <- data$time.NM<LM
  attrisk <- c(attrisk, sum(idx))
  haveLR <- c(haveLR, sum(idx&idx2))
  haveNM <- c(haveNM, sum(idx&idx3))
}

# Making barplots
xlb <- LMs
xlb[!(xlb %in% c(0,1,2,3,4,5))] <- NA
lrplot <- rbind(attrisk-haveLR,haveLR)
barplot(lrplot, width=3/12,ylab="Number of patients",
       xlab="Years since surgery",

```

219
names.arg=xlb,
col=c("darkblue","red"),
xlim==NULL,
ylim=c(0,2000))
legend("topright",legend=c("no LR","LR"),
col=c("darkblue","red"),
pch=15, bty="n")
nmplot <- rbind(atrisk~haveNM,haveNM)
barplot(nmplot, width=3/12,ylab="Number of patients",
xlab="Years since surgery",
names.arg=xlb,
col=c("darkblue","red"),
xlim= NULL,
ylim= c(0,2000))
legend("topright",legend=c("no NM","NM"),
col=c("darkblue","red"),
pch=15, bty="n")

# Making hazard ratio table
cc <- cbind(1, "",
cc, coefs.U[4:5,], cc, coefs.U[7:8,], cc,
rownames(new.coefs) <- c("Good HR", "Poor HR", "Poor HR: s",
"Poor HR: s2", "Local recurrence (LR)",
"New metastatic disease (NM)",
"Age 12–18", "Age < 12", "Age > 18",
"Excision Wide/Radical", "Excision Marginal",
"Excision Other/Intralésional",
"Sex female", "Sex male", "Volume < 200",
"Volume > 200", "Lung met. No",
"Lung met. Yes/Possible", "Other met. No",
"Other met. Yes/Possible", "s", "s2")
xtable(new.coefs)

# Making time-varying hazard plot for histological response
# Edit pooling function B.3 to obtain pooled covariance matrix
vars <- function(mdls){
  #extract models from list with models and heuristic shrinkage factors
  mods <- lapply(1:10, function (i) mdls[[i]][[1]])
  #obtain pooled coefficients
coefs <- lapply(1:10, function (i) coef(mods[[i]]))

#obtain pooled covariance matrix
m <- 10
vars <- lapply(1:10, function (i) vcov(mods[[i]]))
withinVAR <- Reduce('+', vars)/m
betweenVAR <- lapply(coefs, function(x) (x - mcoef) %*% t(x - mcoef))
betweenVAR <- Reduce('+', betweenVAR)/(m-1)
totalVAR <- withinVAR + ((m+1)/m)*betweenVAR

return(totalVAR)

tot.vars <- vars(imp.mod)

#Extract relevant elements and compute standard deviation
rel.bits.GP <- tot.vars[c(1,14,15),c(1,14,15)]
var.2 <- rel.bits.GP[1,1] + rel.bits.GP[2,2] + rel.bits.GP[3,3] +
2*rel.bits.GP[1,2] + 2*rel.bits.GP[2,3] + 2*rel.bits.GP[1,3]
sd.2 <- sqrt(var.2)

#Make plot
coef.c <- as.numeric(coefs.U.1[1,4])
coef.t1 <- as.numeric(coefs.U.1[14,4])
coef.t2 <- as.numeric(coefs.U.1[15,4])

ts <- seq(3/12,5,1/12)
tt <- ts
ts <- as.matrix(ts, ncol=1)

HZ <- apply(ts,1,function(ts) exp(coef.c + coef.t1 * ts[i] +
coef.t2*ts[i]^2))
HZ.u <- apply(ts,1,function(ts) exp(coef.c + coef.t1 * ts[i] +
coef.t2*ts[i]^2 - sd.2*qnorm(0.975)))
HZ.l <- apply(ts,1,function(ts) exp(coef.c + coef.t1 * ts[i] +
coef.t2*ts[i]^2 + sd.2*qnorm(0.975)))

plot(HZ ~ tt, type="l", col="blue", ylim=c(0,3), ylab="Hazard ratio",
xlab="Years since surgery")
lines(HZ.u ~ tt, type="l", col="blue", lty="dashed")

lines(HZ.l ~ tt, type="l", col="blue", lty="dashed")
lines(HZ.1 ~ tt, type="l", col="blue", lty="dashed")
lines(rep(1, length(tt)) ~ tt, col="black")
legend("topright", c("Poor HR", "Good HR"), col=c("blue", "black"), lty=1, bty="n", cex=0.9)

patient1 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Radical/wide", sex="Female"
                        lung.met="No", other.met="No",
                        Hist.resp="Good hist",
                        LR=0,NM=0,LM=0,LM1=0,LM2=0)

patient1 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Radical/wide", sex="Female"
                        lung.met="No", other.met="No",
                        Hist.resp="Poor hist",
                        LR=0,NM=0,LM=0,LM1=0,LM2=0)

patient3 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Radical/wide", sex="Female"
                        lung.met="Yes/Possible", other.met="No",
                        Hist.resp="Good hist",
                        LR=0,NM=0,LM=0,LM1=0,LM2=0)

patient4 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Other", sex="Female"
                        lung.met="No", other.met="No",
                        Hist.resp="Poor hist",
                        LR=0,NM=0,LM=0,LM1=0,LM2=0)

patient5 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Other", sex="Female"
                        lung.met="No", other.met="No",
                        Hist.resp="Good hist",
                        LR=0,NM=0,LM=0,LM1=0,LM2=0)

patient6 <- data.frame(age="12-18 years", volume="< 200",
                        excision="Other", sex="Female"
                        lung.met="Yes/Possible", other.met="No",
                        Hist.resp="Good hist",
                        LR=0,NM,LM=0,LM1=0,LM2=0)

# Patient: reference patient
LRG <- death5year(patient1, LMdata.U, ev1="LR")
NMG <- death5year(patient1, LMdata.U, ev1="NM")
plot(NMG$no$time, NMG$no$Fw, type="l", lwd=2, ylim=c(0, 1), xlim=c(0, 5),
    xlab="Prediction in years since surgery", 
    main="Good histological response", 
    ylab="5-year probability of death", col=c("darkblue"))
lines(NMG$yes[NMG$yes$time>0.25,]$time, 
    NMG$yes[NMG$yes$time>0.25,]$Fw, lwd=2, col=c("red"))
lines(LRG$yes[LRG$yes$time>0.25,]$time, 
    LRG$yes[LRG$yes$time>0.25,]$Fw, lwd=2, col=c("black"))
legend("topright", c("no NM/ LR", "NM", "LR"),
    col=c("darkblue", "red", "black"), bty="n", pch=16)

# Patient 2: poor histological response
LRP <- death5year(patient2, LMdata.U, ev1="LR")
NMP <- death5year(patient2, LMdata.U, ev1="NM")
plot(NMP$no$time, NMP$no$Fw, type="l", lwd=2, ylim=c(0, 1),
    xlab="Prediction in years since surgery", 
    main="Poor histological response", 
    ylab="5-year probability of death", col=c("darkblue"))
lines(NMP$yes[NMP$yes$time>0.25,]$time, 
    NMP$yes[NMP$yes$time>0.25,]$Fw, lwd=2, col=c("red"))
lines(LRP$yes[LRP$yes$time>0.25,]$time, 
    LRP$yes[LRP$yes$time>0.25,]$Fw, lwd=2, col=c("black"))
legend("topright", c("no NM/ LR", "NM", "LR"),
    col=c("darkblue", "red", "black"), bty="n", pch=16)

# Patient 3: lung met, good histological response
LRG.lm <- death5year(patient3, LMdata.U, ev1="LR")
NMG.lm <- death5year(patient3, LMdata.U, ev1="NM")
plot(NMG.lm$no$time, NMG.lm$no$Fw, type="l",
    lwd=2, ylim=c(0, 1), xlim=c(0, 5),
    xlab="Prediction in years since surgery", 
    main="Presence lung metastases", 
    ylab="5-year probability of death", col=c("darkblue"))
lines(NMG.lm$yes[NMG.lm$yes$time>0.25,]$time, 
    NMG.lm$yes[NMG.lm$yes$time>0.25,]$Fw, lwd=2, col=c("red"))
lines(LRG.lm$yes[LRG.lm$yes$time>0.25,]$time, 
    LRG.lm$yes[LRG.lm$yes$time>0.25,]$Fw, lwd=2, col=c("black"))
legend("topright", c("no NM/ LR", "NM", "LR"),
    col=c("darkblue", "red", "black"), bty="n", pch=16)
# patient 4: poor hist, other excision

\[
\text{LRP. ex. o} < \text{death5year( patient4 , LMdata.U, ev1="LR")}
\]

\[
\text{NMP. ex. o} < \text{death5year( patient4 , LMdata.U, ev1="N M")}
\]

\[
\text{plot(NMP. ex. o$no$\text{time} , NMP. ex. o$no$\text{Fw}, type="1", lwd=2, ylim=c(0,1),}
\]

\[
xlim=c(0,5), xlab="\text{Prediction time in years since surgery}",
\]

\[
\text{main="Poor histological response.}
\]

\[
\text{unknown/intralesional excision","}
\]

\[
ylab="5-year probability of death", col=c("darkblue"))
\]

\[
\text{lines(NMP. ex. o$yes [NMP. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{NMP. ex. o$yes [NMP. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("red"))}
\]

\[
\text{lines(LRP. ex. o$yes [LRP. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{LRP. ex. o$yes [LRP. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("black"))}
\]

\[
\text{legend("topright","c("no NM/LR","NM","LR"),}
\]

\[
col=c("darkblue","red","black"), bty="n", pch=16, cex=0.9)
\]

# patient 5: good hist, other excision

\[
\text{LRG. ex. o} < \text{death5year( patient5 , LMdata.U, ev1="LR")}
\]

\[
\text{NMG. ex. o} < \text{death5year( patient5 , LMdata.U, ev1="N M")}
\]

\[
\text{plot(NMG. ex. o$no$\text{time} , NMG. ex. o$no$\text{Fw}, type="1", lwd=2, ylim=c(0,1),}
\]

\[
xlim=c(0,5), xlab="\text{Prediction time in years since surgery}",
\]

\[
\text{main="Unknown/intralesional excision","}
\]

\[
ylab="5-year probability of death", col=c("darkblue"))
\]

\[
\text{lines(NMG. ex. o$yes [NGM. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{NMG. ex. o$yes [NGM. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("red"))}
\]

\[
\text{lines(LRG. ex. o$yes [LRG. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{LRG. ex. o$yes [LRG. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("black"))}
\]

\[
\text{legend("topright","c("no NM/LR","NM","LR"),}
\]

\[
col=c("darkblue","red","black"), bty="n", pch=16)
\]

# patient 6: good hist, other excision, lung mets

\[
\text{LRG. ex. o. l. met} < \text{death5year( patient6 , LMdata.U, ev1="LR")}
\]

\[
\text{NMG. ex. o. l. met} < \text{death5year( patient6 , LMdata.U, ev1="N M")}
\]

\[
\text{plot(NMG. ex. o$no$\text{time} , NMG. ex. o$no$\text{Fw}, type="1", lwd=2, ylim=c(0,1),}
\]

\[
xlim=c(0,5), xlab="\text{Prediction time in years since surgery}",
\]

\[
\text{main="Unknown/intralesional excision","}
\]

\[
ylab="5-year probability of death", col=c("darkblue"))
\]

\[
\text{lines(NMG. ex. o$yes [NGM. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{NGM. ex. o$yes [NGM. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("red"))}
\]

\[
\text{lines(LRG. ex. o$yes [LRG. ex. o$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{LRG. ex. o$yes [LRG. ex. o$yes$\text{time} >0.25,$ \text{Fw}, lwd=2, col=c("black"))}
\]

\[
\text{legend("topright","c("no NM/LR","NM","LR"),}
\]

\[
col=c("darkblue","red","black"), bty="n", pch=16)
\]

\[
\text{plot(NMG. ex. o. l. met$no$\text{time} , NMG. ex. o. l. met$no$\text{Fw}, type="1",}
\]

\[
lwd=2, ylim=c(0,1),
\]

\[
xlim=c(0,5), xlab="\text{Prediction time in years since surgery}",
\]

\[
\text{main="Unknown/lung metastases","}
\]

\[
ylab="5-year probability of death", col=c("darkblue"))
\]

\[
\text{lines(NMG. ex. o. l. met$yes [NGM. ex. o. l. met$yes$\text{time} >0.25,$ \text{time},}
\]

\[
\text{NGM. ex. o. l. met$yes [NGM. ex. o. l. met$yes$\text{time} >0.25,$ \text{Fw},}
\]

\[
lwd=2, col=c("red"))
\]
lines(LRG.ex.o.l.met$yes[LRG.ex.o.l.met$yes$time > 0.25], time)
LWD=2, COL=C("black")
legend("topright", C("no NM/LR", "NM", "LR"),
col=C("darkblue", "red", "black"), bty="n", pch=16)

# event plot (number) for events in final years.
evs <- as.data.frame(cbind(compp$time, death, compp$stat.death))

colnames(evs) <- c("time", "stat")
ev$s$Cums <- cumsum(evs$s$Stat)
ev$s$Risk <- nrow(compp) - ev$s$Cums

evs$s$new <- NA

length(which(compp$time.death > 5))

vals <- seq(0, 9.3/12)

for (i in 1:length(vals)) {
  ind <- which(evs$time > vals[i] & evs$time <= vals[i+1])
  ev$s$new[ind] <- vals[i]
}

bb <- aggregate(evs$risk, by=list(evs$new), mean)
cc <- aggregate(evs$Stat, by=list(evs$new), sum)

fin <- as.data.frame(cbind(bb$x, cc$x))
colnames(fin) <- c("event", "atrisk")
fin <- t(fin)

finredd <- fin
finredd[1,] <- finredd[1,] - 1450
finred <- fin[, 21:36]
finred[1,] <- finred[1,] - 1450

ww <- c(5, NA, NA, NA, 6, NA, NA, NA, 7, NA, NA, NA, 8, NA, NA, NA, 9)

barplot(finred, width=3/12, ylab="Number of patients",
  xlab="Years since surgery",
  xlim=NULL,
  ylim=NULL,
  col=C("darkblue", "red"),
  yaxt="n",
  names.arg=ww)

axis(side=2, at=seq(0, 45, 5), labels=seq(0, 45, 5) + seq(1450, 1495, 5))

legend("topright", legend=c("At risk", "Dead"),
col=C("darkblue", "red"), pch=15, bty="n")
Appendix C

A Neural network approach

C.1 Data transformation

# For the purpose of comparability, imputed values for
# histological response and volume were pooled.
# For histological response:

pit <- comp.dat.par[[1]]

gp <- cbind(comp.dat.par[[1]]$GOOD.POOR.t,
            comp.dat.par[[2]]$GOOD.POOR.t,
            comp.dat.par[[3]]$GOOD.POOR.t,
            comp.dat.par[[4]]$GOOD.POOR.t,
            comp.dat.par[[5]]$GOOD.POOR.t,
            comp.dat.par[[6]]$GOOD.POOR.t,
            comp.dat.par[[7]]$GOOD.POOR.t,
            comp.dat.par[[8]]$GOOD.POOR.t,
            comp.dat.par[[9]]$GOOD.POOR.t,
            comp.dat.par[[10]]$GOOD.POOR.t)

means <- rowMeans(gp-1)
means <- round(means, 0)
levels(means) <- c("Good.hist", "Poor.hist")
gp <- means

# For volume:
vol.mat <- matrix(unlist(list.vol), ncol=10, byrow=FALSE)
vols <- rowMeans(vol.mat)

colnames(pit) <- c("stat", "time", "Hist", "Age",
"Volume", "Ex", "Sex", "lung", "other")
```
30 pit$Volume <- vols
31 pit$Hist <- gp
32 head(pit)
33
34 id <- seq(1:nrow(pit),1)
35 pit <- cbind(pit, id)
36 pit <- pit[order(pit$id),]
37 colnames(pit) <- c("stat", "time", "Hist", "Age", "Volume", "Ex", "Sex", "lung", "other", "id")
38
39 pot <- pit
40
41 # defining validation set and training set ,
42 # with equal proportions of events.
43 one.set <- which(pit$stat==1)
44 zero.set <- which(pit$stat==0)
45 set.seed(987)
46 ind.ones <- sample(one.set, length(one.set)/3, replace=FALSE)
47 set.seed(987)
48 ind.zeros <- sample(zero.set, length(zero.set)/3, replace=FALSE)
49 validation.set.ind <- c(ind.ones, ind.zeros)
50
51 # data.transformer function, applied to dataset
52 # of pooled imputed values
data.transformer <- function(pot){
53
54 # scaling of variables
55 pot <- pot[order(pot$id),]
56 pot$Age <- scale(pot$Age)
57 pot$Volume <- scale(pot$Volume)
58
59 # finding interval for each time point
60 alt.Int <- seq(0,9,1/12)
61 ITvec <- vector("numeric", length=nrow(pot))
62 for (i in 1:nrow(pot)){
63 ITvec[i] <- findInterval(pot$time[i], alt.Int)
64 }
65 pot <- cbind(pot, ITvec)
66
67 # creating indicator variables for categorical covariates
68 n.mat <- as.data.frame(model.matrix(~ -1 + Hist + Ex +
69 Sex + lung+ other + pot))
71
72 227
```
new.pot <- cbind(pot$stat, pot$id, pot$Age, pot$Volume, pot$ITvec, n.mat)
pot.l <- split(new.pot, new.pot$id)

uns <- sort(unique(ITvec))

# function for replicating observation for the number of 
# intervals in which it occurs
trans.func <- function(x){
  # Surv gives the interval of event/censoring for each patient
  val <- x$Surv
  # finding index interval (in case intervals are skipped)
  rept <- which(uns %in% val)
  # replicate patient for each occurrence
  x <- x[rep(seq_len(nrow(x)), rept),]
  # add vector of intervals
  x$ITvec <- uns[1:rept]
  # ensure that event is only indicated for the final
  # replicate corresponding to the event interval.
  if(1 %in% x$Stat){
    x$Stat <- 0
    x$Stat[rept] <- 1
  }
  return(x)
}

pot.t <- lapply(pot.l, trans.func)
pot.t <- do.call("rbind", pot.t)

return(pot.t)}

# Data transformer for test function
special.test.transformer <- function(pat){
  pat <- pat[order(pat$id),]
  pat$Age <- scale(pat$Age)
  pat$Volume <- scale(pat$Volume)
  alt.Int <- seq(0, 0.9, 0.1/12)
surv.it <- vector("numeric", length=nrow(pat))
for (i in 1:nrow(pat)){
  surv.it[i] <- findInterval(pat$time[i], alt.Int)
}

all.poss <- unique(sort(surv.it))
pat <- cbind(pat, surv.it)
n.mat <- as.data.frame(model.matrix(~ -1 + Hist + Ex + Sex + lung + other, pat))
"Ex.O", "Sex.M", "L.y", "O.y")

new.pat <- cbind(pat$stat, pat$id, pat$Age, pat$Volume, 
pat$surv.it, n.mat)
rownames(new.pat) <- c("Stat", "id", "Age", "Volume", 
"L.y", "O.y")
pat.l <- split(new.pat, new.pat$id)

trans.func <- function(x){
  val <- x$Surv
  rept <- length(all.poss)
  x <- x[rep(seq_len(nrow(x)), rept),]
  x$ITvec <- all.poss
  return(x)
}

pat.t <- lapply(pat.l, trans.func)
pat.t <- do.call("rbind", pat.t)

C.2 Performance measures and variable importance

# Function for calculating Brier and KL score for neural network.

# function for calculating the brier and KL scores.
# takes as argument data.frame surv.preds which has 
# a column surv for survival time and stat for status, 
# and preds, which is the matrix of predictions
brier.function.neur <- function(surv.preds, preds)
\begin{verbatim}
{
    # calculating censoring distribution
    so <- Surv(surv.preds$surv, surv.preds$stat)
    time <- so[,1]
    ot <- order(time)
    cens <- so[ot,2]
    time <- time[ot]
    N <- nrow(so)

    hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
    csurv <- predict(hatcdist, times = time, type = "surv")
    csurv[csurv == 0] <- Inf

    btime <- time

    # transposing prediction matrix, so that predictions for
    # a given time are given by row
    survs <- t(preds)

    # unique prediction times
    btime.n <- unique(time)

    # defining empty vectors for brier score (bsc)
    # and KL score (bsk)
    bsc <- rep(0, length(btime.n))
    bsk <- rep(0, length(btime.n))

    for (j in 1:length(btime.n)) {
        # indicator vector for selecting relevant patients
        # help1 selects those individuals that have an event
        # at or before the time of interest. The predictions for
        # these observations are compared
        help1 <- as.integer(time <= btime.n[j] & cens == 1)
        help2 <- as.integer(time > btime.n[j])

        # As the log of zero cannot be taken, values of 1 and 0
        # need to be modified: values of 1 are lowered by
        # 0.000001, values of 0 increased.
        inb <- survs[j,]
        inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
        inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)

        # calculating brier scores for each unique time j, weighted by
\end{verbatim}
# the censoring distribution csurv

csurv[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
(1 - survs[j,])^2 * help2 * (1/csurv[j]))

# calculating KL scores for each unique time j, weighted
# by the censoring distribution csurv. survs[j,] has been
# replaced by inb (defined above) to solve the problem of
# extreme predictions.

bsk[j] <- -mean((log(1-(inb))*help1*(1/csurv) +
                log(inb)*(1/csurv[j]) *help2))

# Estimates are made for every month-long interval.
# Predictions at 1 to 8 years are selected by subtracting
# the relevant number of months (i.e 12 for 1 year,
# 24 for 2 years, etc.), and selecting the relevant
# score (brier or KL) at the minimum of the resulting vector.
# This roundabout approach is taken for the event that an
# exact value cannot be established: the procedure will
# then default to the nearest available interval.

ind.values <- seq(1,8)

rel.vals <- vector("numeric", length(ind.values))
for(i in 1:length(ind.values)){
  rel.vals[[i]] <- which.min(abs(btime-ind.values[i]))
}

Brie <- bsc[rel.vals]
KL <- bsk[rel.vals]

RET <- rbind(Brie,KL)

return(RET)

# Brier/KL score function slightly modified to suit Cox
brier.function.cox.m <- function(imp, preds)
{
  so <- Surv(imp$surv, imp$stat)
time <- so[,1]
  ot <- order(time)
  cens <- so[ot,2]
```r
98  time <- time[ot]
99  N <- nrow(so)
100  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
101  csurv <- predict(hatcdist, times = time, type = "surv")
102  csurv[csurv == 0] <- Inf
103
104  btime <- time
105
106  survs <- t(as.matrix(preds))
107
108  bsc <- rep(0, nrow(survs))
109  bsk <- rep(0, nrow(survs))
110  bsp <- rep(0, nrow(survs))
111  for (j in 1:nrow(survs)) {
112    help1 <- as.integer(time <= btime[j] & cens == 1)
113    help2 <- as.integer(time > btime[j])
114    inb <- survs[j,]
115    inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
116    inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)
117    bsc[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
118                     (1 - survs[j,])^2 * help2 * (1/csurv[j]))
119    bsk[j] <- -mean((log((1-(inb)))*help1*(1/csurv) +
120                     log(inb)*(1/csurv[j]) * help2))
121    bsp[j] <- -mean((log(1-survs[j,]) * help1 * (1/csurv) +
122                     log(survs[j,]) * help2 * (1/csurv[j])))
123  }
124
125  ind.values <- seq(1,8)
126
127  rel.vals <- vector("numeric", length(ind.values))
128  for (i in 1:length(ind.values)) {
129    rel.vals[[i]] <- which.min(abs(btime-ind.values[i]))
130  }
131
132  Brie <- bsc[rel.vals]
133  KL <- bsk[rel.vals]
134
135  RET <- rbind(Brie, KL)
136
137  return(RET)
138  }
```
# Variable importance function

```r
pretty.picture.func <- function(model.nnet){

  # names input nodes
  names <- model.nnet$coefnames

  # number input nodes
  no.input <- model.nnet$n[1]

  # number hidden nodes
  no.nodes <- model.nnet$n[2]

  # number output nodes
  no.outputs <- model.nnet$n[3]
  rainbowcols <- rainbow(length(names), s = 0.5)

  # dimensions for weight matrices
  ncols.1 <- (no.input + 1)
  nrows.1 <- (no.nodes)
  ncols.2 <- (no.nodes + 1)
  nrows.2 <- no.outputs
  length.1 <- ncols.1 * nrows.1
  length.2 <- ncols.2 * nrows.2

  # selecting weights
  wts <- model.nnet$wts
  weights1 <- as.data.frame(matrix(wts[1:length.1],
                                  ncol=ncols.1, nrow=nrows.1, byrow=TRUE))
  colnames(weights1) <- c("Bias", names)
  rownames(weights1) <- paste("N", seq(1:no.nodes), sep="")
  weights2 <- t(matrix(wts[(length.1+1):(length.1+length.2)],
                       nrow=nrows.2, ncol=ncols.2, byrow=TRUE))
  rownames(weights2) <- c("Bias", paste("N", seq(1:no.nodes),
                                      sep=""))

  # calculating variable contribution using the
  # connection weight approach
  mega.mat <- matrix(NA, ncol=no.input, nrow=no.outputs)
  for (i in 1:no.input){
    for (j in 1:no.outputs){
      mega.mat[j,i] <- sum(weights1[,i+1] *
                           weights2[2:(no.nodes + 1), j])
    }
  }
  colnames(mega.mat) <- names
```

233
mega.mat.abs <- abs(mega.mat)
totals <- rowSums(mega.mat.abs)
mega.mat.rel <- as.data.frame(mega.mat.abs/totals)
rainbowcols <- rainbow(length(names), s = 0.5)
rels <- as.vector(as.numeric(mega.mat.rel))
barplot(rels, col=rainbowcols, names.arg=names, ylim=c(0, 0.4), ylab="Relative importance")

C.3 Cross-validation

C.3.1 Cross-validation functions

# Cross-validation neural network function. Takes as arguments
# a training set, node size and decay parameter.
other.neur <- function(dat.train, node.size, decay.par){
dat.train <- dat.train[order(dat.train$id),]

# transforming training data
dat.try <- data.transformer(dat.train)
dat.try <- dat.try[order(dat.try$id, dat.try$ITvec),]

# Defining folds
K <- 5
index <- rep(1:K, floor(nrow(dat.train)/K)+1)[1:nrow(dat.train)]
set.seed(92874)
fold.index <- sample(index)

# empty vectors for error measures
error.val <- vector("numeric", K)
acc <- vector("numeric", K)
sens <- vector("numeric", K)
spec <- vector("numeric", K)
brier <- NULL
KL <- NULL

for (k in 1:K){
# indicator vector for training set in wide format data
inda <- which(fold.index!=k)
# selecting individuals using the id vector
ids <- dat.train$id[inda]
# finding individuals in long format
ind.p <- which(dat.try$id %in% ids)

training <- dat.try[ind.p,]

# Selecting test set (test.dat) from wide format data
ind.ts <- which(fold.index==k)
test.dat <- dat.train[ind.ts,]
# Transforming test using the test transformation function
test <- special.test.transformer(test.dat)

# Assigning interval column to separate vector
# prior to normalization
IT.test <- test$ITvec
test$ITvec <- scale(test$ITvec)

# Vectors are defined for the variables, because specifying
# a dataframe instead causes the predict() function to mal-
# function
Volume <- training$Volume; Age <- training$Age
Hist.P <- training$Hist.P; Ex.M <- training$Ex.M
Ex.O <- training$Ex.O; Sex.M <- training$Sex.M
L.y <- training$L.y; O.y <- training$O.y
ITvec <- scale(training$ITvec); stat <- training$Stat

new.n.net <- nnet(cbind(stat) ~ Age + Volume + Hist.P +
Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec,  
size=node.size, maxit=2000, MaxNWts=10000,  
decay=decay.par, entropy=TRUE)
predict.neur <- predict(new.n.net, test, type="raw")

# data frame with prediction, prediction times, id, 
# status, and original survival time in long format
coll <- as.data.frame(cbind(predict.neur, test$Surv,  
test$id, test$Stat))
colnames(coll) <- c("prob", "surv", "id", "stat")
stat <- test.dat$stat
idd <- unique(coll$id)
# obtaining survival interval
surv.id <- col[,[c(2,3)]]
real.surv <- unlist(lapply(split(surv.id, surv.id$id),
  function(x){
a <- x$surv
b <- a[1]
return(b)
}))

# obtaining survival estimates
coll.s <- col[,[c(1,3)]]
coll.s <- split(coll.s, coll$id)
coll.t <- lapply(coll.s, function(x) {
x <- cumprod(1-x$prob)
})

# obtaining prediction matrix
pred.matrix <- do.call("rbind", coll.t)

# obtaining relevant survival estimate
rel.probs <- vector("numeric", length(coll.t))
for (i in 1:length(coll.t)){
temp <- coll.t[[i]]
ind <- which(sort(unique(IT.test))== real.surv[i])
rel.probs[i] <- temp[ind]
}

# dataframe with probabilities, status, id and survival interval
statt <- as.data.frame(cbind(as.data.frame(rel.probs),
  stat, idd, real.surv))
colnames(statt) <- c("pred", "stat", "id","surv")

hmstat <- statt

# rounding predicted probabilities to classify individuals as yes/no event
hmstat$pred <- 1-round(hmstat$pred,0)

# saving error values
ero.error.val[k] <- new.n.net$value
acc[k] <- length(which(hmstat$pred==hmstat$stat))/nrow(hmstat)
sens[k] <- length(which(hmstat$pred==1 & hmstat$stat==1))/length(which(hmstat$stat==1))
spec[k] <- length(which(hmstat$pred==0 &
hmstat$stat==0)/length(which(hmstat$stat==0))
brieKL <- brier.function.neur(statt, pred.matrix)
brier <- rbind(brier, brieKL[1,])
KL <- rbind(KL, brieKL[2,])
}
lst <- list()
lst[[1]] <- cbind(error.val, acc, sens, spec)
lst[[2]] <- brier
lst[[3]] <- KL
obj <- lst
return(obj)
}

# function for applying previously defined cross-validation
# function over different numbers of nodes for a given
# decay parameter. Returns a list with accuracy, sensitivity,
# specificity and the cross-entropy error value as the first
# element, the Brier scores for 1–8 years as the second, and
# the Kullback–Leibler scores for 1–8 years as the third list
# element.
cross.val.plots <- function(decay, neuron.range) {

  # for the specified decay, an object is created for each
  # number of hidden nodes, saved in a list.
  # each list element contains a list with the cross-validated
  # values of the measures, in elements 1, 2, 3:
  # 1) error value, accuracy, sensitivity, specificity,
  # 2) brier scores for every year from 1 to 8 years
  # 3) KL scores for every year from 1 to 8 years
  objects <- list()
  for (i in 1:length(neuron.range)) {
    objects[[i]] <- other.neur(dat.train, neuron.range[i], decay)
  }

  # the means of the measures are calculated over the folds
  means <- list()
  for (i in 1:length(objects)) {
    means[[i]] <- lapply(objects[[i]], colMeans)
  }

  # the sds of the measures are calculated over the folds
  sds <- list()
for (i in 1:length(objects)){
  sds[[i]] <- lapply(objects[[i]], function(x) {
    sds <- vector("numeric", ncol(x))
    for (i in 1:ncol(x)){
      sds[i] <- sd(x[,i])
    }
    return(sds))
  }
}

# the means for error value, accuracy, sensitivity, specificity are collected in a matrix
single.vals.0.m <- matrix(NA, nrow=length(neuron.range), ncol=4)
for (i in 1:length(neuron.range)){
  single.vals.0.m[i,] <- means[[i]][[1]]
}

# the sds for error value, accuracy, sensitivity, specificity are collected in a matrix
single.vals.0.s <- matrix(NA, nrow=length(neuron.range), ncol=4)
for (i in 1:length(neuron.range)){
  single.vals.0.s[i,] <- sds[[i]][[1]]
}

# the means for the brier scores are collected in a matrix
briers <- matrix(NA, nrow=length(neuron.range), ncol=8)
for (i in 1:length(neuron.range)){
  briers[i,] <- means[[i]][[2]]
}

# the sds for the brier scores are collected in a matrix
briers.s <- matrix(NA, nrow=length(neuron.range), ncol=8)
for (i in 1:length(neuron.range)){
  briers.s[i,] <- sds[[i]][[2]]
}

# the means for the KL scores are collected in a matrix
KLs <- matrix(NA, nrow=length(neuron.range), ncol=8)
for (i in 1:length(neuron.range)){
  KLs[i,] <- means[[i]][[3]]
}

# the sds for the KL scores are collected in a matrix
KLs.s <- matrix(NA, nrow=length(neuron.range), ncol=8)
for (i in 1:length(neuron.range)){
  KLs.s[i,] <- sds[[i]][[3]]
}
```r
# all the matrices are saved in a list
bothlist <- list()
bothlist[[1]] <- single.vals.0.m
bothlist[[2]] <- briers
bothlist[[3]] <- KLs

both.list.s <- list()
both.list.s[[1]] <- single.vals.0.s
both.list.s[[2]] <- briers.s
both.list.s[[3]] <- KLs.s

all.list <- list()
all.list[[1]] <- bothlist
all.list[[2]] <- both.list.s

return(all.list)
```

# Applying the function for different decay values, and
# gathering the individual performance measures in
# separate matrices.

outt <- cross.val.plots(0, c(1,2,3,4,5,6,7,8,9,10))
means.0 <- outt[1]
sds.0 <- outt[2]

outt.1 <- cross.val.plots(0.001, c(1,2,3,4,5,6,7,8,9,10))
means.1 <- outt.1[1]
sds.1 <- outt.1[2]

outt.2 <- cross.val.plots(0.01, c(1,2,3,4,5,6,7,8,9,10))
means.2 <- outt.2[1]
sds.2 <- outt.2[2]

outt.3 <- cross.val.plots(0.05, c(1,2,3,4,5,6,7,8,9,10))
means.3 <- outt.3[1]
sds.3 <- outt.3[2]

outt.4 <- cross.val.plots(0.1, c(1,2,3,4,5,6,7,8,9,10))
means.4 <- outt.4[1]
```
C.3.2 Cross-validation plots

# Kaplan Meier plot and event time distribution plot
par(mfrow=c(1,2))
plot(survfit(Surv(dat.train$time, dat.train$stat) ~ 1),
     xlab="Time (years since surgery)",
     ylab="Survival probability")
hist(dat.train$time[which(dat.train$stat == 1)],
     xlab="Time (years since surgery)",
     main="")

# For every measure, the values for each of the decay parameters are collected in a matrix. Then, for each value the standard error over the folds (5) is used to calculate the ±1 and −1 sd values.
vall <- cbind(means.0[[1]][[1]][,1], means.1[[1]][[1]][,1],
              means.2[[1]][[1]][,1], means.3[[1]][[1]][,1],
              means.4[[1]][[1]][,1])
vall.sd <- cbind(sds.0[[1]][[1]][,1], sds.1[[1]][[1]][,1],
                 sds.2[[1]][[1]][,1], sds.3[[1]][[1]][,1],
                 sds.4[[1]][[1]][,1])
vall1 <- vall + vall.sd
vall2 <- vall - vall.sd

acc <- cbind(means.0[[1]][[1]][,2], means.1[[1]][[1]][,2],
              means.2[[1]][[1]][,2], means.3[[1]][[1]][,2],
              means.4[[1]][[1]][,2])
acc.sd <- cbind(sds.0[[1]][[1]][,2], sds.1[[1]][[1]][,2],
                sds.2[[1]][[1]][,2], sds.3[[1]][[1]][,2],
                sds.4[[1]][[1]][,2])
acc1 <- acc + acc.sd
acc2 <- acc - acc.sd

sens <- cbind(means.0[[1]][[1]][,3], means.1[[1]][[1]][,3],
               means.2[[1]][[1]][,3], means.3[[1]][[1]][,3],
               means.4[[1]][[1]][,3])
sens.sd <- cbind(sds.0[[1]][[1]][,3], sds.1[[1]][[1]][,3],
                 sds.2[[1]][[1]][,3], sds.3[[1]][[1]][,3],
                 sds.4[[1]][[1]][,3])
sens1 <- sens + sens.sd
sens2 <- sens - sens.sd
spec <- cbind(means.0[[1]][[1]][,4], means.1[[1]][[1]][,4],
               means.2[[1]][[1]][,4], means.3[[1]][[1]][,4],
               means.4[[1]][[1]][,4])
spec.sd <- cbind(sds.0[[1]][[1]][,4], sds.1[[1]][[1]][,4],
                 sds.2[[1]][[1]][,4], sds.3[[1]][[1]][,4],
                 sds.4[[1]][[1]][,4])
spec1 <- spec + spec.sd
spec2 <- spec - spec.sd
brier <- list()
brier[[1]] <- means.0[[1]][[2]]
brier[[2]] <- means.1[[1]][[2]]
brier[[3]] <- means.2[[1]][[2]]
brier[[4]] <- means.3[[1]][[2]]
brier[[5]] <- means.4[[1]][[2]]
brier.s1 <- list()
brier.s1[[1]] <- means.0[[1]][[2]] + sds.0[[1]][[2]]
brier.s1[[2]] <- means.1[[1]][[2]] + sds.1[[1]][[2]]
brier.s1[[3]] <- means.2[[1]][[2]] + sds.2[[1]][[2]]
brier.s1[[4]] <- means.3[[1]][[2]] + sds.3[[1]][[2]]
brier.s1[[5]] <- means.4[[1]][[2]] + sds.4[[1]][[2]]
brier.s2 <- list()
brier.s2[[1]] <- means.0[[1]][[2]] - sds.0[[1]][[2]]
brier.s2[[2]] <- means.1[[1]][[2]] - sds.1[[1]][[2]]
brier.s2[[3]] <- means.2[[1]][[2]] - sds.2[[1]][[2]]
brier.s2[[4]] <- means.3[[1]][[2]] - sds.3[[1]][[2]]
brier.s2[[5]] <- means.4[[1]][[2]] - sds.4[[1]][[2]]
KLr <- list()
KLr[[1]] <- means.0[[1]][[3]]
KLr[[2]] <- means.1[[1]][[3]]
KLr[[3]] <- means.2[[1]][[3]]
KLr[[4]] <- means.3[[1]][[3]]
KLr[[5]] <- means.4[[1]][[3]]
KL.s1 <- list()
KL.s1[[1]] <- means.0[[1]][[3]] + sds.0[[1]][[3]]
KL.s1[[2]] <- means.1[[1]][[3]] + sds.1[[1]][[3]]
KL.s1[[3]] <- means.2[[1]][[3]] + sds.2[[1]][[3]]
KL.s1[[4]] <- means.3[[1]][[3]] + sds.3[[1]][[3]]
```r
KL.s1[5] <- means.4[[1]][[3]] + sds.4[[1]][[3]]
KL.s2 <- list()
KL.s2[1] <- means.0[[1]][[3]] - sds.0[[1]][[3]]
KL.s2[2] <- means.1[[1]][[3]] - sds.1[[1]][[3]]
KL.s2[3] <- means.2[[1]][[3]] - sds.2[[1]][[3]]
KL.s2[4] <- means.3[[1]][[3]] - sds.3[[1]][[3]]
KL.s2[5] <- means.4[[1]][[3]] - sds.4[[1]][[3]]

# Making plots for Brier and KL scores, and for accuracy, sensitivity, specificity and the error value

## legends

legend("topright", legend=c(paste(1:10, "Nodes", sep=" ")), col="black", rainbows, lty=1, lwd=2)
legend("topright", legend=c("Accuracy", "Sensitivity", "Specificity"), col="black", "red", "blue", lty=1, lwd=2)

xx <- seq(1,8)

plot((brierr[[1]][1,])~xx, type="l", ylab="Brier score", xlab="Time (days)", lwd=2, main="Weight decay : 0")

arrows(xx, brierr.s2[[1]][1,], xx, brierr.s1[[1]][1,], length=0.05, angle=90, code=3)
lines((brierr[[1]][2,])~xx, type="l", ylim=c(0,0.5), col=rainbows[1], lwd=2)

arrows(xx, brierr.s2[[1]][2,], xx, brierr.s1[[1]][2,], length=0.05, angle=90, code=3, col=rainbows[1])
lines((brierr[[1]][3,])~xx, type="l", ylim=c(0,0.5), col=rainbows[2], lwd=2)

arrows(xx, brierr.s2[[1]][3,], xx, brierr.s1[[1]][3,], length=0.05, angle=90, code=3, col=rainbows[2])
lines((brierr[[1]][4,])~xx, type="l", ylim=c(0,0.5), col=rainbows[3], lwd=2)
```

arrows(xx, brier.s2[[1]][4,], xx, brier.s1[[1]][4,],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((brier [[1]][5,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[4]
, lwd=2)
arrows(xx, brier.s2[[1]][5,], xx, brier.s1[[1]][5,],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((brier [[1]][6,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[5]
, lwd=2)
arrows(xx, brier.s2[[1]][6,], xx, brier.s1[[1]][6,],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((brier [[1]][7,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[6]
, lwd=2)
arrows(xx, brier.s2[[1]][7,], xx, brier.s1[[1]][7,],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((brier [[1]][8,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[7]
, lwd=2)
arrows(xx, brier.s2[[1]][8,], xx, brier.s1[[1]][8,],
length=0.05,
angle=90, code=3, col=rainbows[7])
lines((brier [[1]][9,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[8]
, lwd=2)
arrows(xx, brier.s2[[1]][9,], xx, brier.s1[[1]][9,],
length=0.05,
angle=90, code=3, col=rainbows[8])
lines((brier [[1]][10,])~xx, type="l", ylim=c(0, 0.5),
col=rainbows[9]
, lwd=2)
arrows(xx, brier.s2[[1]][10,], xx, brier.s1[[1]][10,],
length=0.05,
angle=90, code=3, col=rainbows[9])

plot((brier [[2]][1,])~xx, type="l", ylim=c(0, 0.5), lwd=2,
ylab="Brier score",
xlab="Time (days)", main="Weight decay: 0.001")
arrows(xx, brier.s2[[2]][1,], xx, brier.s1[[2]][1,],

243
```r
length = 0.05, angle = 90, code = 3
lines((brierrr[[2]][2,])^xx, type = "l", ylim = c(0, 0.5), lwd = 2,
col = rainbows[1])
arrows(xx, brier.s2[[2]][2,], xx, brier.s1[[2]][2,],
length = 0.05,
angle = 90, code = 3, col = rainbows[1])
lines((brierrr[[2]][3,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[2])
arrows(xx, brier.s2[[2]][3,], xx, brier.s1[[2]][3,],
length = 0.05,
angle = 90, code = 3, col = rainbows[2])
lines((brierrr[[2]][4,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[3])
arrows(xx, brier.s2[[2]][4,], xx, brier.s1[[2]][4,],
length = 0.05,
angle = 90, code = 3, col = rainbows[3])
lines((brierrr[[2]][5,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[4])
arrows(xx, brier.s2[[2]][5,], xx, brier.s1[[2]][5,],
length = 0.05,
angle = 90, code = 3, col = rainbows[4])
lines((brierrr[[2]][6,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[5])
arrows(xx, brier.s2[[2]][6,], xx, brier.s1[[2]][6,],
length = 0.05,
angle = 90, code = 3, col = rainbows[5])
lines((brierrr[[2]][7,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[6])
arrows(xx, brier.s2[[2]][7,], xx, brier.s1[[2]][7,],
length = 0.05,
angle = 90, code = 3, col = rainbows[6])
lines((brierrr[[2]][8,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[7])
arrows(xx, brier.s2[[2]][8,], xx, brier.s1[[2]][8,],
length = 0.05,
angle = 90, code = 3, col = rainbows[7])
lines((brierrr[[2]][9,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[8])
arrows(xx, brier.s2[[2]][9,], xx, brier.s1[[2]][9,],
length = 0.05,
angle = 90, code = 3, col = rainbows[8])
lines((brierrr[[2]][10,])^xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[9])
arrows(xx, brier.s2[[2]][10,], xx, brier.s1[[2]][10,],
length = 0.05,
angle = 90, code = 3, col = rainbows[9])
```

angle = 90, code = 3, col = rainbows[9])

plot((brierr[[3]][1,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, ylab = "Brier score",
  xlab = "Time (days)", main = "Weight decay: \omega=0.01")

arrows(xx, brier.s2[[3]][1,], xx, brier.s1[[3]][1,],
  length = 0.05, angle = 90, code = 3)

lines((brierr[[3]][2,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[1])

arrows(xx, brier.s2[[3]][2,], xx, brier.s1[[3]][2,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[1])

lines((brierr[[3]][3,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[2])

arrows(xx, brier.s2[[3]][3,], xx, brier.s1[[3]][3,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[2])

lines((brierr[[3]][4,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[3])

arrows(xx, brier.s2[[3]][4,], xx, brier.s1[[3]][4,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[3])

lines((brierr[[3]][5,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[4])

arrows(xx, brier.s2[[3]][5,], xx, brier.s1[[3]][5,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[4])

lines((brierr[[3]][6,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[5])

arrows(xx, brier.s2[[3]][6,], xx, brier.s1[[3]][6,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[5])

lines((brierr[[3]][7,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[6])

arrows(xx, brier.s2[[3]][7,], xx, brier.s1[[3]][7,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[6])

lines((brierr[[3]][8,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[7])

arrows(xx, brier.s2[[3]][8,], xx, brier.s1[[3]][8,],
  length = 0.05,
  angle = 90, code = 3, col = rainbows[7])

lines((brierr[[3]][9,])~xx, type = "l", ylim = c(0, 0.5),
lwd = 2, col = rainbows[8])
arrows(xx, brier.s2[[3]][9,], xx, brier.s1[[3]][9,],
length=0.05,
    angle=90, code=3, col=rainbows[8])
lines((brier[[3]][10,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[9])
arrows(xx, brier.s2[[3]][10,], xx, brier.s1[[3]][10,],
length=0.05,
    angle=90, code=3, col=rainbows[9])

plot((brier[[4]][1,])~xx, type="l", ylim=c(0,0.5),
lwd=2, ylab="Brier score",
    xlab="Time (days)", main="Weight decay: 0.05")
arrows(xx, brier.s2[[4]][1,], xx, brier.s1[[4]][1,],
length=0.05, angle=90, code=3)
lines((brier[[4]][2,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[1])
arrows(xx, brier.s2[[4]][2,], xx, brier.s1[[4]][2,],
length=0.05,
    angle=90, code=3, col=rainbows[1])
lines((brier[[4]][3,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[2])
arrows(xx, brier.s2[[4]][3,], xx, brier.s1[[4]][3,],
length=0.05,
    angle=90, code=3, col=rainbows[2])
lines((brier[[4]][4,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[3])
arrows(xx, brier.s2[[4]][4,], xx, brier.s1[[4]][4,],
length=0.05,
    angle=90, code=3, col=rainbows[3])
lines((brier[[4]][5,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[4])
arrows(xx, brier.s2[[4]][5,], xx, brier.s1[[4]][5,],
length=0.05,
    angle=90, code=3, col=rainbows[4])
lines((brier[[4]][6,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[5])
arrows(xx, brier.s2[[4]][6,], xx, brier.s1[[4]][6,],
length=0.05,
    angle=90, code=3, col=rainbows[5])
lines((brier[[4]][7,])~xx, type="l", ylim=c(0,0.5),
lwd=2, col=rainbows[6])
arrows(xx, brier.s2[[4]][7,], xx, brier.s1[[4]][7,],
length=0.05,
\begin{verbatim}
angle=90, code=3, \texttt{col}=\texttt{rainbows[6])
lines((\texttt{brier[[4]][8,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[7])
arrows(xx, \texttt{brier.s2[[4]][8,]}), xx, \texttt{brier.s1[[4]][8,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[7])
lines((\texttt{brier[[4]][9,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[8])
arrows(xx, \texttt{brier.s2[[4]][9,]}), xx, \texttt{brier.s1[[4]][9,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[8])
lines((\texttt{brier[[5]][1,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[1])
arrows(xx, \texttt{brier.s2[[5]][1,]}), xx, \texttt{brier.s1[[5]][1,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[1])
lines((\texttt{brier[[5]][2,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[2])
arrows(xx, \texttt{brier.s2[[5]][2,]}), xx, \texttt{brier.s1[[5]][2,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[2])
lines((\texttt{brier[[5]][3,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[3])
arrows(xx, \texttt{brier.s2[[5]][3,]}), xx, \texttt{brier.s1[[5]][3,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[3])
lines((\texttt{brier[[5]][4,]})*xx, type="1", ylim=c(0,0.5), lwd=2, \texttt{col}=\texttt{rainbows[4])
arrows(xx, \texttt{brier.s2[[5]][4,]}), xx, \texttt{brier.s1[[5]][4,]}), length=0.05,
\texttt{angle}=90, code=3, \texttt{col}=\texttt{rainbows[4])
lines((\texttt{brier[[5]][6,]})*xx, type="1", ylim=c(0,0.5),
\end{verbatim}
lwd=2, col=rainbows[5])

arrows(xx, brier.s2[[5]][6,], xx, brier.s1[[5]][6,], length=0.05,
      angle=90, code=3, col=rainbows[5])

lines((brier[[5]][7,])~xx, type="l", ylim=c(0,0.5),
      lwd=2, col=rainbows[6])

arrows(xx, brier.s2[[5]][7,], xx, brier.s1[[5]][7,],
      length=0.05,
      angle=90, code=3, col=rainbows[6])

lines((brier[[5]][8,])~xx, type="l", ylim=c(0,0.5),
      lwd=2, col=rainbows[7])

arrows(xx, brier.s2[[5]][8,], xx, brier.s1[[5]][8,],
      length=0.05,
      angle=90, code=3, col=rainbows[7])

lines((brier[[5]][9,])~xx, type="l", ylim=c(0,0.5),
      lwd=2, col=rainbows[8])

arrows(xx, brier.s2[[5]][9,], xx, brier.s1[[5]][9,],
      length=0.05,
      angle=90, code=3, col=rainbows[8])

plot((KLr[[1]][1,])~xx, type="l", ylim=c(0,5), lwd=2,
     ylab="Kullback-Leibler score",
     xlab="Time (days)", main="Weight decay: 0"
     length=0.05, angle=90, code=3)

lines((KLr[[1]][2,])~xx, type="l", ylim=c(0,5),
      lwd=2, col=rainbows[1])

arrows(xx, KL.s2[[1]][2,], xx, KL.s1[[1]][2,],
      length=0.05,
      angle=90, code=3, col=rainbows[1])

lines((KLr[[1]][3,])~xx, type="l", ylim=c(0,5),
      lwd=2, col=rainbows[2])

arrows(xx, KL.s2[[1]][3,], xx, KL.s1[[1]][3,],
      length=0.05,
      angle=90, code=3, col=rainbows[2])

lines((KLr[[1]][4,])~xx, type="l", ylim=c(0,5),
      lwd=2, col=rainbows[3])
arrows(xx, KL.s2[[1]][4], xx, KL.s1[[1]][4],
  length=0.05,
  angle=90, code=3, col=rainbows[3])
lines((KLr[[1]][5]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[4])
arrows(xx, KL.s2[[1]][5], xx, KL.s1[[1]][5],
  length=0.05,
  angle=90, code=3, col=rainbows[4])
lines((KLr[[1]][6]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[5])
arrows(xx, KL.s2[[1]][6], xx, KL.s1[[1]][6],
  length=0.05,
  angle=90, code=3, col=rainbows[5])
lines((KLr[[1]][7]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[6])
arrows(xx, KL.s2[[1]][7], xx, KL.s1[[1]][7],
  length=0.05,
  angle=90, code=3, col=rainbows[6])
lines((KLr[[1]][8]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[7])
arrows(xx, KL.s2[[1]][8], xx, KL.s1[[1]][8],
  length=0.05,
  angle=90, code=3, col=rainbows[7])
lines((KLr[[1]][9]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[8])
arrows(xx, KL.s2[[1]][9], xx, KL.s1[[1]][9],
  length=0.05,
  angle=90, code=3, col=rainbows[8])
lines((KLr[[1]][10]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[9])
arrows(xx, KL.s2[[1]][10], xx, KL.s1[[1]][10],
  length=0.05,
  angle=90, code=3, col=rainbows[9])

plot((KLr[[2]][1]), xx, type="l", ylim=c(0,5),
  lwd=2, ylab="Kullback-Leibler score",
  xlab="Time (days)", main="Weight decay: 0.001")
arrows(xx, KL.s2[[2]][1], xx, KL.s1[[2]][1],
  length=0.05, angle=90, code=3)
lines((KLr[[2]][2]), xx, type="l", ylim=c(0,5),
  lwd=2, col=rainbows[1])
arrows(xx, KL.s2[[2]][2], xx, KL.s1[[2]][2],
  length=0.05,
  angle=90, code=3, col=rainbows[1])
```r
lines((KLr[[2]][3,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[2])
arrows(xx, KL.s2[[2]][3,], xx, KL.s1[[2]][3,],
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((KLr[[2]][4,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[3])
arrows(xx, KL.s2[[2]][4,], xx, KL.s1[[2]][4,],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((KLr[[2]][5,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[4])
arrows(xx, KL.s2[[2]][5,], xx, KL.s1[[2]][5,],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((KLr[[2]][6,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[5])
arrows(xx, KL.s2[[2]][6,], xx, KL.s1[[2]][6,],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((KLr[[2]][7,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[6])
arrows(xx, KL.s2[[2]][7,], xx, KL.s1[[2]][7,],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((KLr[[2]][8,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[7])
arrows(xx, KL.s2[[2]][8,], xx, KL.s1[[2]][8,],
length=0.05,
angle=90, code=3, col=rainbows[7])
lines((KLr[[2]][9,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[8])
arrows(xx, KL.s2[[2]][9,], xx, KL.s1[[2]][9,],
length=0.05,
angle=90, code=3, col=rainbows[8])
lines((KLr[[2]][10,])~xx, type="l", ylim=c(0,5),
lwd=2, col=rainbows[9])
arrows(xx, KL.s2[[2]][10,], xx, KL.s1[[2]][10,],
length=0.05,
angle=90, code=3, col=rainbows[9])

plot((KLr[[3]][1,])~xx, type="l", lwd=2, ylim=c(0,5),
ylab="Kullback-Leibler score",

250
```
\begin{verbatim}
537     length=0.05, angle=90, code=3)
538     lines ((KLr [[4]] [2, ]))~xx, type="l", lwd=2, ylim=c(0,1),
539     col=rainbows [1])
540     arrows(xx, KL.s2 [[4]] [2 ,], xx, KL.s1 [[4]] [2 ,], length=0.05,
541           angle=90, code=3, col=rainbows [1])
542     lines ((KLr [[4]] [3 ,])~xx, type="l", lwd=2, ylim=c(0,1),
543           col=rainbows [2])
544     arrows(xx, KL.s2 [[4]] [3 ,], xx, KL.s1 [[4]] [3 ,], length=0.05,
545           angle=90, code=3, col=rainbows [2])
546     lines ((KLr [[4]] [4 ,])~xx, type="l", lwd=2, ylim=c(0,1),
547           col=rainbows [3])
548     arrows(xx, KL.s2 [[4]] [4 ,], xx, KL.s1 [[4]] [4 ,], length=0.05,
549           angle=90, code=3, col=rainbows [3])
550     lines ((KLr [[4]] [5 ,])~xx, type="l", lwd=2, ylim=c(0,1),
551           col=rainbows [4])
552     arrows(xx, KL.s2 [[4]] [5 ,], xx, KL.s1 [[4]] [5 ,], length=0.05,
553           angle=90, code=3, col=rainbows [4])
554     lines ((KLr [[4]] [6 ,])~xx, type="l", lwd=2, ylim=c(0,1),
555           col=rainbows [5])
556     arrows(xx, KL.s2 [[4]] [6 ,], xx, KL.s1 [[4]] [6 ,], length=0.05,
557           angle=90, code=3, col=rainbows [5])
558     lines ((KLr [[4]] [7 ,])~xx, type="l", lwd=2, ylim=c(0,1),
559           col=rainbows [6])
560     arrows(xx, KL.s2 [[4]] [7 ,], xx, KL.s1 [[4]] [7 ,], length=0.05,
561           angle=90, code=3, col=rainbows [6])
562     lines ((KLr [[4]] [8 ,])~xx, type="l", lwd=2, ylim=c(0,1),
563           col=rainbows [7])
564     arrows(xx, KL.s2 [[4]] [8 ,], xx, KL.s1 [[4]] [8 ,], length=0.05,
565           angle=90, code=3, col=rainbows [7])
566     lines ((KLr [[4]] [9 ,])~xx, type="l", lwd=2, ylim=c(0,1),
567           col=rainbows [8])
568     arrows(xx, KL.s2 [[4]] [9 ,], xx, KL.s1 [[4]] [9 ,], length=0.05,
569           angle=90, code=3, col=rainbows [8])
570     lines ((KLr [[4]] [10 ,])~xx, type="l", lwd=2, ylim=c(0,1),
571           col=rainbows [9])
572     arrows(xx, KL.s2 [[4]] [10 ,], xx, KL.s1 [[4]] [10 ,], length=0.05,
573           angle=90, code=3, col=rainbows [9])
574
575
576     plot ((KLr [[5]] [1 ,]))~xx, type="l", ylab="Kullback Leibler score",
577     xlab="Time (days)", main="Weight decay : 0.1")
578     arrows(xx, KL.s2 [[5]] [1 ,], xx, KL.s1 [[5]] [1 ,],
579           length=0.05, angle=90, code=3)
252
\end{verbatim}
lines((KLr[5][2,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[1])

arrows(xx, KL.s2[[5]][2,], xx, KL.s1[[5]][2,], length=0.05, angle=90, code=3, col=rainbows[1])

lines((KLr[5][3,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[2])

arrows(xx, KL.s2[[5]][3,], xx, KL.s1[[5]][3,], length=0.05, angle=90, code=3, col=rainbows[2])

lines((KLr[5][4,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[3])

arrows(xx, KL.s2[[5]][4,], xx, KL.s1[[5]][4,], length=0.05, angle=90, code=3, col=rainbows[3])

lines((KLr[5][5,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[4])

arrows(xx, KL.s2[[5]][5,], xx, KL.s1[[5]][5,], length=0.05, angle=90, code=3, col=rainbows[4])

lines((KLr[5][6,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[5])

arrows(xx, KL.s2[[5]][6,], xx, KL.s1[[5]][6,], length=0.05, angle=90, code=3, col=rainbows[5])

lines((KLr[5][7,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[6])

arrows(xx, KL.s2[[5]][7,], xx, KL.s1[[5]][7,], length=0.05, angle=90, code=3, col=rainbows[6])

lines((KLr[5][8,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[7])

arrows(xx, KL.s2[[5]][8,], xx, KL.s1[[5]][8,], length=0.05, angle=90, code=3, col=rainbows[7])

lines((KLr[5][9,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[8])

arrows(xx, KL.s2[[5]][9,], xx, KL.s1[[5]][9,], length=0.05, angle=90, code=3, col=rainbows[8])

lines((KLr[5][10,])~xx, type="l", ylim=c(0,1), lwd=2, col=rainbows[9])

arrows(xx, KL.s2[[5]][10,], xx, KL.s1[[5]][10,], length=0.05, angle=90, code=3, col=rainbows[9])

x <- 1:10

plot(vall[,1]~x, pch=16, ylim=c(250,300), ylab="Error value", xlab="Number of nodes", main="Weight decay: 0")
arrows(x, vall2[,1], x, vall1[,1], length=0.05, angle=90, code=3)
plot(acc[,1] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", main="Weight decay : 0")

arrows(x, acc2[,1], x, acc1[,1], length=0.05, angle=90, code=3)

points(sens[,1] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="red")

arrows(x, sens2[,1], x, sens1[,1], length=0.05, angle=90, code=3, col="red")

points(spec[,1] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="blue")

arrows(x, spec2[,1], x, spec1[,1], length=0.05, angle=90, code=3, col="blue")

plot(val2[,2] ~ x, pch=16, ylim=c(250,300), ylab="Error value", xlab="Number of nodes", main="Weight decay : 0.001")

arrows(x, val2[,2], x, val1[,2], length=0.05, angle=90, code=3)

points(sens[,2] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="red")

arrows(x, sens2[,2], x, sens1[,2], length=0.05, angle=90, code=3, col="red")

points(spec[,2] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="blue")

arrows(x, spec2[,2], x, spec1[,2], length=0.05, angle=90, code=3, col="blue")

plot(val2[,3] ~ x, pch=16, ylim=c(250,300), ylab="Error value", xlab="Number of nodes", main="Weight decay : 0.01")

arrows(x, val2[,3], x, val1[,3], length=0.05, angle=90, code=3)

points(sens[,3] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="red")

arrows(x, sens2[,3], x, sens1[,3], length=0.05, angle=90, code=3)

points(spec[,3] ~ x, pch=16, ylim=c(0,1), ylab="Accuracy", xlab="Number of nodes", col="red")
672 arrows(x, sens2[,3], x, sens1[,3], length=0.05, angle=90,
673   code=3, col="red")
674 points(spec[,3]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
675   xlab="Number of nodes", col="blue")
676 arrows(x, spec2[,3], x, spec1[,3], length=0.05, angle=90,
677   code=3, col="blue")
678
679 plot(vall[,4]~x,pch=16, ylim=c(250,300), ylab="Error value",
680   xlab="Number of nodes", main="Weight decay : 0.05")
681 arrows(x, vall2[,4], x, vall1[,4], length=0.05, angle=90,
682   code=3)
683
684 plot(acc[,4]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
685   xlab="Number of nodes", main="Weight decay : 0.05")
686 arrows(x, acc2[,4], x, acc1[,4], length=0.05, angle=90,
687   code=3)
688
689 points(sens[,4]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
690   xlab="Number of nodes", col="red")
691 arrows(x, sens2[,4], x, sens1[,4], length=0.05, angle=90,
692   code=3, col="red")
693 points(spec[,4]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
694   xlab="Number of nodes", col="blue")
695 arrows(x, spec2[,4], x, spec1[,4], length=0.05, angle=90,
696   code=3, col="blue")
697
698 plot(vall[,5]~x,pch=16, ylim=c(250,300), ylab="Error value",
699   xlab="Number of nodes", main="Weight decay : 0.1")
700 arrows(x, vall2[,5], x, vall1[,5], length=0.05, angle=90,
701   code=3)
702
703 plot(acc[,5]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
704   xlab="Number of nodes", main="Weight decay : 0.1")
705 arrows(x, acc2[,5], x, acc1[,5], length=0.05, angle=90,
706   code=3)
707
708 points(sens[,5]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
709   xlab="Number of nodes", col="red")
710 arrows(x, sens2[,5], x, sens1[,5], length=0.05, angle=90,
711   code=3, col="red")
712 points(spec[,5]~x,pch=16, ylim=c(0,1), ylab="Accuracy",
713   xlab="Number of nodes", col="blue")
714 arrows(x, spec2[,5], x, spec1[,5], length=0.05, angle=90,
715   code=3, col="blue")
C.4 Results

C.4.1 Neural Network

```r
# Fitting neural networks with nodes 1–10 and decay=0.05

dat.test <- dat.test[order(dat.test$id),]
training <- data.transformer(dat.train)
test <- special.test.transformer(dat.test)

IT.test <- test$ITvec
Surv.test <- test$Surv

test$ITvec <- scale(test$ITvec)

Volume <- training$Volume
Age <- training$Age
Hist.P <- training$Hist.P
Ex.M <- training$Ex.M
Ex.O <- training$Ex.O
Sex.M <- training$Sex.M
L.y <- training$L.y
O.y <- training$O.y
Surv <- training$Surv

#Surv <- scale(Surv)
ITvec <- training$ITvec
ITvec <- scale(training$ITvec)

stat <- training$Stat
neg.stat <- 1-stat

# Neural networks with 0.05 decay

new.n.net.1 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
                      Ex.O + Sex.M +
                      L.y + O.y + ITvec,
                      size=1, maxit=2000, MaxNWts=10000, decay=0.05)

new.n.net.2 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
                      Ex.O + Sex.M +
                      L.y + O.y + ITvec,
                      size=2, maxit=2000, MaxNWts=10000, decay=0.05)
```
new.n.net.3 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=3, maxit=2000, MaxNWts=10000, decay=0.05
 )

new.n.net.4 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=4, maxit=2000, MaxNWts=10000, decay=0.05
 )

new.n.net.5 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=5, maxit=2000, MaxNWts=10000, decay=0.05
 )

new.n.net.6 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=6, maxit=2000, MaxNWts=10000, decay=0.1
 )

new.n.net.7 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=7, maxit=2000, MaxNWts=10000, decay=0.05
 )

new.n.net.8 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=8, maxit=2000, MaxNWts=10000, decay=0.05
 )

new.n.net.9 <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
    Ex.O + Sex.M +
    L.y + O.y + ITvec,
    size=9, maxit=2000, MaxNWts=10000, decay=0.05
 )

257
new.n.net.10 <- nnet(formula = stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, size = 10, maxit = 2000, MaxNWts = 10000, decay = 0.05)

# Defining reference patient for neural network and cox model
eur.pat.good <- test[1,]
eur.pat.good$Hist.G <- 1
eur.pat.good$Hist.P <- 0
eur.pat.good$Ex.M <- 0
eur.pat.good$Ex.O <- 0
eur.pat.good$Sex.M <- 0
eur.pat.good$Sex.O <- 0
eur.pat.good$L.y <- 0
eur.pat.good$O.y <- 0
eur.pat.good$Age <- 0
eur.pat.good$Volume <- 0
eur.pat.good <- neur.pat.good[rep(seq_len(nrow(eur.pat.good)), length(times)),]
eur.pat.good$ITvec <- scale(sort(unique(IT.test)))

cox.pat <- dat.test[1,]
cox.pat$Hist <- "Good_Hist"
cox.pat$Age <- mean(dat.test$Age)
cox.pat$Volume <- mean(dat.test$Volume)
cox.pat$lung <- "No"
cox.pat$other <- "No"
cox.pat$Sex <- "Female"
eur.cox.good <- predictSurvProb(cox.eur, cox.pat, times = dat.test$time)

# Making plots for reference patient
times <- sort(unique(test$Surv))
times <- times/12

pred.eur.temp1 <- predict(new.n.net.1, eur.pat.good, type = "raw")
pred.eur.temp1 <- cumprod(1 - pred.eur.temp1)
plot(pred.eur.temp1~times, type="l", ylim=c(0,1), xlim=c(0,9),
  xlab="Time (Years since surgery)",
  ylab="Survival probability", lty=2, col="black", lwd=2)

pred.eur.temp2 <- predict(new.n.net.2, eur.pat.good, type="raw")
lines(pred.eur.temp2~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[1], lwd=2)

pred.eur.temp3 <- predict(new.n.net.3, eur.pat.good, type="raw")
lines(pred.eur.temp3~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[2], lwd=2)

pred.eur.temp4 <- predict(new.n.net.4, eur.pat.good, type="raw")
lines(pred.eur.temp4~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[3], lwd=2)

pred.eur.temp5 <- predict(new.n.net.5, eur.pat.good, type="raw")
lines(pred.eur.temp5~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[4], lwd=2)

pred.eur.temp6 <- predict(new.n.net.6, eur.pat.good, type="raw")
lines(pred.eur.temp6~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[5], lwd=2)

pred.eur.temp7 <- predict(new.n.net.7, eur.pat.good, type="raw")
lines(pred.eur.temp7~times, type="l", ylim=c(0,1),
  xlab="Time (days)",
  ylab="Survival probability", lty=2,
  col=rainbows[6], lwd=2)
lines(pred.eur.temp7~times, type="l", ylim=c(0,1),
xlab="Time_(days)", ylab="Survival\_probability",
lty=2, col=rainbows[6], lwd=2)

pred.eur.temp8 <- predict(new.n.net.8, eur.pat.good, type="raw")
lines(pred.eur.temp8~times, type="l", ylim=c(0,1),
xlab="Time_(days)", ylab="Survival\_probability",
lty=2, col=rainbows[7], lwd=2)

pred.eur.temp9 <- predict(new.n.net.9, eur.pat.good, type="raw")
lines(pred.eur.temp9~times, type="l", ylim=c(0,1),
xlab="Time_(days)", ylab="Survival\_probability",
lty=2, col=rainbows[8], lwd=2)

pred.eur.temp10 <- predict(new.n.net.10, eur.pat.good,
type="raw")
lines(as.vector(eur.cox.good) ~ as.vector(dat.test\$_\$time),
type="l", ylim=c(0,1), xlab="Time_(days)",
ylab="Survival\_probability", lty=1,
lwd=2, col="purple")

legend("bottomleft", legend=c("Cox", "Nodes", sep=" "),
"Cox")

eur.pat.poor.hist <- test[1,]
eur.pat.poor.hist$Hist.G <- 0
eur.pat.poor.hist$Hist.P <- 1
eur.pat.poor.hist$Ex.M <- 0
eur.pat.poor.hist$Ex.O <- 0
eur.pat.poor.hist$Sex.M <- 0
eur.pat.poor.hist$Sex.O <- 0
eur.pat.poor.hist$Age <- 0
eur.pat.poor.hist$Volume <- 0
times <- sort(unique(test$Surv))
eur.pat.poor.hist <- eur.pat.poor.hist[rep(seq_len(nrow(eur.pat.poor.hist)), length(times)),]
eur.pat.poor.hist$ITvec <- scale(sort(unique(IT.test)))

cox.pat.poor <- dat.test[1,]
cox.pat.poor$Age <- mean(dat.test$Age)
cox.pat.poor$Volume <- mean(dat.test$Volume)
cox.pat.poor$lung <- "No"
cox.pat.poor$other <- "No"
cox.pat.poor$Sex <- "Female"

eur.cox.poor <- predictSurvProb(cox.eur, cox.pat.poor, times=dat.test$time)
pred.eur.temp1 <- predict(new.n.net.1, eur.pat.poor.hist, type="raw")
pred.eur.temp1 <- cumprod(1-pred.eur.temp1)
plot(pred.eur.temp1~times, type="l", ylim=c(0,1),
xlim=c(0,9), xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col="black", lwd=2)
pred.eur.temp2 <- predict(new.n.net.2, eur.pat.poor.hist, type="raw")
pred.eur.temp2 <- cumprod(1-pred.eur.temp2)
lines(pred.eur.temp2~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[1], lwd=2)
pred.eur.temp3 <- predict(new.n.net.3, eur.pat.poor.hist, type="raw")
pred.eur.temp3 <- cumprod(1-pred.eur.temp3)
lines(pred.eur.temp3~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[2], lwd=2)
pred.eur.temp4 <- predict(new.n.net.4, eur.pat.poor.hist, type="raw")
pred.eur.temp4 <- cumprod(1-pred.eur.temp4)
lines(pred.eur.temp4~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[3], lwd=2)

pred.eur.temp5 <- predict(new.n.net.5, eur.pat.poor.hist, type="raw")
pred.eur.temp5 <- cumprod(1-pred.eur.temp5)
lines(pred.eur.temp5~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[4], lwd=2)

pred.eur.temp6 <- predict(new.n.net.6, eur.pat.poor.hist, type="raw")
pred.eur.temp6 <- cumprod(1-pred.eur.temp6)
lines(pred.eur.temp6~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[5], lwd=2)

pred.eur.temp7 <- predict(new.n.net.7, eur.pat.poor.hist, type="raw")
pred.eur.temp7 <- cumprod(1-pred.eur.temp7)
lines(pred.eur.temp7~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[6], lwd=2)

pred.eur.temp8 <- predict(new.n.net.8, eur.pat.poor.hist, type="raw")
pred.eur.temp8 <- cumprod(1-pred.eur.temp8)
lines(pred.eur.temp8~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[7], lwd=2)

pred.eur.temp9 <- predict(new.n.net.9, eur.pat.poor.hist, type="raw")
pred.eur.temp9 <- cumprod(1-pred.eur.temp9)
lines(pred.eur.temp9~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[8], lwd=2)

pred.eur.temp10 <- predict(new.n.net.10, eur.pat.poor.hist, type="raw")
pred.eur.temp10 <- cumprod(1-pred.eur.temp10)
lines(pred.eur.temp10^times, type="l", ylim=c(0,1),
xlab="Time\(\text{(Years since surgery)}\)",
    ylab="Survival probability", lty=2,
    col=rainbows[9], lwd=2)

lines(as.vector(eur.cox.poor) ~ as.vector(dat.test$time),
    type="l", ylim=c(0,1), xlab="Time\(\text{(Years since surgery)}\)",
    ylab="Survival probability", lty=1,
    lwd=2, col="purple")

# Defining patient with poor histological response and lung metastases for neural network and cox model

eur.pat.poor.lung <- test[1,]
eur.pat.poor.lung$Hist.G <- 0
eur.pat.poor.lung$Hist.P <- 1
eur.pat.poor.lung$Ex.M <- 0
eur.pat.poor.lung$Ex.O <- 0
eur.pat.poor.lung$Sex.M <- 0
eur.pat.poor.lung$Sex.O <- 1
eur.pat.poor.lung$L.y <- 1
eur.pat.poor.lung$O.y <- 0
eur.pat.poor.lung$Age <- 0
eur.pat.poor.lung$Volume <- 0
times <- sort(unique(test$Surv))
eur.pat.poor.lung <- rep(seq_len(nrow(eur.pat.poor.lung)), length(times),]
eur.pat.poor.lung$ITvec <- scale(sort(unique(IT.test)))

cox.pat.poor.lung <- dat.test[1,]
cox.pat.poor.lung$Age <- mean(dat.test$Age)
cox.pat.poor.lung$Volume <- mean(dat.test$Volume)
cox.pat.poor.lung$lung <- "No"
cox.pat.poor.lung$other <- "No"
cox.pat.poor.lung$Sex <- "Female"
cox.pat.poor.lung$lung <- "Yes/Possible"

eur.cox.poor.lung <- predictSurvProb(cox.eur,
    cox.pat.poor.lung, times=dat.test$time)
times <- sort(unique(IT.test))
times <- times/12
pred.eur.pl.temp1 <- predict(new.n.net.1,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp1 <- cumprod(1-pred.eur.pl.temp1)
plot(pred.eur.pl.temp1~times, type="l", ylim=c(0,1),
   xlab="Time (Years since surgery)",
   ylab="Survival probability", lty=2, col="black", lwd=2)
pred.eur.pl.temp2 <- predict(new.n.net.2,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp2 <- cumprod(1-pred.eur.pl.temp2)
lines(pred.eur.pl.temp2~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[1], lwd=2)
pred.eur.pl.temp3 <- predict(new.n.net.3,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp3 <- cumprod(1-pred.eur.pl.temp3)
lines(pred.eur.pl.temp3~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[2], lwd=2)
pred.eur.pl.temp4 <- predict(new.n.net.4,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp4 <- cumprod(1-pred.eur.pl.temp4)
lines(pred.eur.pl.temp4~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[3], lwd=2)
pred.eur.pl.temp5 <- predict(new.n.net.5,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp5 <- cumprod(1-pred.eur.pl.temp5)
lines(pred.eur.pl.temp5~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[4], lwd=2)
pred.eur.pl.temp6 <- predict(new.n.net.6,eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp6 <- cumprod(1-pred.eur.pl.temp6)
```r
lines(pred.eur.pl.temp6~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[5],
lwd=2)

pred.eur.pl.temp7 <- predict(new.n.net.7, eur.pat.poor.lung,
type="raw")
lines(pred.eur.pl.temp7~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[6],
lwd=2)

pred.eur.pl.temp8 <- predict(new.n.net.8, eur.pat.poor.lung,
type="raw")
lines(pred.eur.pl.temp8~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2, col=rainbows[7],
lwd=2)

pred.eur.pl.temp9 <- predict(new.n.net.9, eur.pat.poor.lung,
type="raw")
lines(pred.eur.pl.temp9~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2,
col=rainbows[8], lwd=2)

lines(as.vector(eur.cox.poor.lung) ~ as.vector(sort(dat.test$time)),
type="l",
ylim=c(0,1), xlab="Time (days)",
ylab="Survival probability", lty=1,
lwd=2, col="purple")
```
new.n.net.1.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=1, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.2.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=2, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.3.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=3, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.4.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=4, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.5.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=5, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.6.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=6, maxit=2000, MaxNWts=10000, decay=0.001 )

new.n.net.7.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + Ex.O + Sex.M + L.y + O.y + ITvec, 
size=7, maxit=2000, MaxNWts=10000, decay=0.001 )

# Plot patient poor histological response at decay 0.001
\textbf{new.n.net.8.i} <- \texttt{nnet(\texttt{stat} \sim \texttt{Age} + \texttt{Volume} + \texttt{Hist.P} + \texttt{Ex.M} +}
\texttt{\texttt{Ex.O} + \texttt{Sex.M} +}
\texttt{\texttt{L.y} + \texttt{O.y} + \texttt{ITvec},}
\texttt{size=8, maxit=2000, MaxNWts=10000, decay=0.001)}

\textbf{new.n.net.9.i} <- \texttt{nnet(\texttt{stat} \sim \texttt{Age} + \texttt{Volume} + \texttt{Hist.P} + \texttt{Ex.M} +}
\texttt{\texttt{Ex.O} + \texttt{Sex.M} +}
\texttt{\texttt{L.y} + \texttt{O.y} + \texttt{ITvec},}
\texttt{size=9, maxit=2000, MaxNWts=10000, decay=0.001)}

\textbf{new.n.net.10.i} <- \texttt{nnet(\texttt{stat} \sim \texttt{Age} + \texttt{Volume} + \texttt{Hist.P} + \texttt{Ex.M} +}
\texttt{\texttt{Ex.O} + \texttt{Sex.M} +}
\texttt{\texttt{L.y} + \texttt{O.y} + \texttt{ITvec},}
\texttt{size=10, maxit=2000, MaxNWts=10000, decay=0.001)}

\texttt{pred.eur.temp1.i} <- \texttt{predict(new.n.net.1.i, eur.pat.poor.hist,}
\texttt{type="raw")}
\texttt{pred.eur.temp1.i} <- \texttt{cumprod(1-pred.eur.temp1.i)}
\texttt{plot(pred.eur.temp1.i \sim \texttt{times, type="l", ylim=c(0,1), xlim=c(0,9),}
\texttt{xlab="Time\_\textbackslash{(Years\_since\_surgery)}",}
\texttt{ylab="Survival\_probability", lty=2, \texttt{col="black", lwd=2})}

\texttt{pred.eur.temp2.i} <- \texttt{predict(new.n.net.2.i, eur.pat.poor.hist,}
\texttt{type="raw")}
\texttt{pred.eur.temp2.i} <- \texttt{cumprod(1-pred.eur.temp2.i)}
\texttt{lines(pred.eur.temp2.i \sim \texttt{times, type="l", ylim=c(0,1),}
\texttt{xlim=c(0,9), xlab="Time\_\textbackslash{(Years\_since\_surgery)}",}
\texttt{ylab="Survival\_probability", lty=2,}
\texttt{col=rainbows[1], lwd=2})

\texttt{pred.eur.temp3.i} <- \texttt{predict(new.n.net.3.i, eur.pat.poor.hist,}
\texttt{type="raw")}
\texttt{pred.eur.temp3.i} <- \texttt{cumprod(1-pred.eur.temp3.i)}
\texttt{lines(pred.eur.temp3.i \sim \texttt{times, type="l", ylim=c(0,1),}
\texttt{xlim=c(0,9), xlab="Time\_\textbackslash{(Years\_since\_surgery)}",}
\texttt{ylab="Survival\_probability", lty=2,}
\texttt{col=rainbows[2], lwd=2})
pred.eur.temp4.i <- predict(new.n.net.4.i, eur.pat.poor.hist, type="raw")
pred.eur.temp4.i <- cumprod(1 - pred.eur.temp4.i)
lines(pred.eur.temp4.i * times, type="1", ylim=c(0,1),
xlim=c(0,9), xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[3], lwd=2)

pred.eur.temp5.i <- predict(new.n.net.5.i, eur.pat.poor.hist, type="raw")
pred.eur.temp5.i <- cumprod(1 - pred.eur.temp5.i)
lines(pred.eur.temp5.i * times, type="1", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2,
col=rainbows[4], lwd=2)

pred.eur.temp6.i <- predict(new.n.net.6.i, eur.pat.poor.hist, type="raw")
pred.eur.temp6.i <- cumprod(1 - pred.eur.temp6.i)
lines(pred.eur.temp6.i * times, type="1", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2,
col=rainbows[5], lwd=2)

pred.eur.temp7.i <- predict(new.n.net.7.i, eur.pat.poor.hist, type="raw")
pred.eur.temp7.i <- cumprod(1 - pred.eur.temp7.i)
lines(pred.eur.temp7.i * times, type="1", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[6], lwd=2)

pred.eur.temp8.i <- predict(new.n.net.8.i, eur.pat.poor.hist, type="raw")
pred.eur.temp8.i <- cumprod(1 - pred.eur.temp8.i)
lines(pred.eur.temp8.i * times, type="1", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[7], lwd=2)

pred.eur.temp9.i <- predict(new.n.net.9.i, eur.pat.poor.hist, type="raw")
pred.eur.temp9.i <- cumprod(1 - pred.eur.temp9.i)
lines(pred.eur.temp9.i * times, type="1", ylim=c(0,1),
xlim=c(0,9), xlab="Time (Years since surgery)", ylab="Survival probability", lty=2,
col=rainbows[8], lwd=2

pred.eur.temp10.i <- predict(new.n.net.10.i, eur.pat.poor.hist,
type="raw")
pred.eur.temp10.i <- cumprod(1-pred.eur.temp10.i)
lines(pred.eur.temp10.i~times, type="l", ylim=c(0,1),
xlab="Time (days)", ylab="Survival probability", lty=2, col=rainbows[9], lwd=2)

lines(as.vector(eur.cox.poor) ~ as.vector(sort(dat.test$time)),
type="l",
ylim=c(0,1), xlab="Time (days)", ylab="Survival probability", lty=1,
lwd=2, col="purple")

# Instability plot for network with 5 hidden nodes, decay 0.001
times <- sort(unique(IT.test))/12
new.n.net.5.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
                        Ex.O + Sex.M + L.y + O.y + ITvec,
                        size=4, maxit=2000, MaxNWts=10000, decay=0.001
                        )
pred.eur.temp5.i <- predict(new.n.net.5.i, eur.pat.poor.hist,
type="raw")
pred.eur.temp5.i <- cumprod(1-pred.eur.temp5.i)
plot(pred.eur.temp5.i~as.vector(times), type="l",
ylim=c(0,1), xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[3], lwd=2)

new.n.net.5.i <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
                        Ex.O + Sex.M + L.y + O.y + ITvec,
                        size=4, maxit=2000, MaxNWts=10000, decay=0.001
                        )
pred.eur.temp5.i <- predict(new.n.net.5.i, eur.pat.poor.hist,
type="raw")
pred.eur.temp5.i <- cumprod(1-pred.eur.temp5.i)
lines(pred.eur.temp5.i~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[3],
lwd=2)
# network is rerun every time and a new lines()
# object is plotted

# function for calculating performance measures
final.other.neur <- function(dat.train, dat.test, 
node.size, decay.par){

dat.train <- dat.train[order(dat.train$id),]

dat.test <- dat.test[order(dat.test$id),]

training <- data.transformer(dat.train)
test <- special.test.transformer(dat.test)

IT.test <- test$ITvec
Surv.test <- test$Surv

test$ITvec <- scale(test$ITvec)

Volume <- training$Volume
Age <- training$Age
Hist.P <- training$Hist.P
Ex.M <- training$Ex.M
Ex.O <- training$Ex.O
Sex.M <- training$Sex.M
L.y <- training$L.y
O.y <- training$O.y
Surv <- training$Surv

#Surv <- scale(Surv)
ITvec <- training$ITvec
ITvec <- scale(training$ITvec)
stat <- training$Stat

new.n.net <- nnet(stat ~ Age + Volume + Hist.P + Ex.M + 
Ex.O + Sex.M +
L.y + O.y + ITvec,
size=5, maxit=2000, MaxNWts=10000, decay=0.05,
entropy=TRUE)

predict.neur <- predict(new.n.net, test, type="raw")
predict.neur <- predict.neur[,1]
# data frame with prediction, prediction times, id, status, and original survival time in long format

coll <- as.data.frame(cbind(predict.neur, test$Surv, test$id, test$Stat))

ITs <- sort(unique(dat.try$Surv), decreasing=FALSE)

colnames(coll) <- c("prob", "surv", "id", "stat")

stat <- dat.test$stat
idd <- unique(coll$id)

surv.id <- coll[,c(2,3)] #5
real.surv <- unlist(lapply(split(surv.id, surv.id$id),
    function(x){
a <- x$Surv
b <- a[1]
return(b)
})

# obtaining survival estimates

coll.s <- coll[,c(1,3)]
coll.s <- split(coll.s, coll$id)
coll.t <- lapply(coll.s, function(x){
x <- cumprod(1-x$prob)
})
pred.matrix <- do.call("rbind", coll.t)
pred.matrix <- pred.matrix[order(real.surv),]

rel.probs <- vector("numeric", length(coll.t))
for (i in 1:length(coll.t)){
temp <- coll.t[[i]]
ind <- which(sort(unique(IT.test))== real.surv[i])
rel.probs[i] <- temp[ind]
}

statt <- as.data.frame(cbind(as.data.frame(rel.probs),
    stat, idd, real.surv))
colnames(stat) <- c("pred", "stat", "id", "surv")

hmstat <- statt
hmstat$pred <- 1-round(hmstat$pred,0)
error.val <- new.n.net$value
acc <- length(which(hmstat$pred==hmstat$stat))/nrow(hmstat)
sens <- length(which(hmstat$pred==1 & hmstat$stat==1))/
  length(which(hmstat$stat==1))
spec <- length(which(hmstat$pred==0 & hmstat$stat==0))/
  length(which(hmstat$stat==0))
brieKL <- brier.function.neur(statt, pred.matrix)
brier <- brieKL[1,]
KL <- brieKL[2,]
lst <- list()
lst[[1]] <- cbind(error.val, acc, sens, spec)
lst[[2]] <- brier
lst[[3]] <- KL
obj <- lst
return(obj)

# final result plot
cox.eur.brier.KL <- brier.function.cox.m(cox.statt, cox.pred.matrix)
final.eur <- final.other.neur(dat.train, dat.test, 5, 0.05)
plot(cox.eur.brier.KL[1,]~seq(1,8), type="l", ylim=c(0,1),
ylab="Brier/KL score",
xlab="Time (days)", lwd=2)
lines(final.eur[[2]]~seq(1,8), type="l", ylim=c(0,0.5), col="red",
  lwd=2)
lines(final.eur[[3]]~seq(1,8), type="l", ylim=c(0,0.5), col="red",
  lwd=2, lty=2)
lines(cox.eur.brier.KL[2,]~seq(1,8), type="l", ylim=c(0,0.5),
  col="black",
  lwd=2, lty=2)
legend("topright", legend=c("Cox.Brier","Cox.KL", "Neural.Brier",
  "Neural.KL"),
  col=c("black", "black", "red", "red"), lty=c(1,2,1,2),
  lwd=2,bty="n")
# final result patient
times <- sort(unique(IT.test))/12
pred.eur.temp5 <- predict(new.n.net.5, eur.pat.good, type="raw")
pred.eur.temp5 <- cumprod(1-pred.eur.temp5)
plot(pred.eur.temp5~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[4], lwd=2)
lines(as.vector(eur.cox.good) ~ as.vector(sort(dat.test$time)),
type="l",
ylim=c(0,1), xlab="Time (days)",
ylab="Survival probability", lty=1,
lwd=2, col=rainbows[4])
pred.eur.temp5 <- predict(new.n.net.5, eur.pat.poor.hist,
type="raw")
pred.eur.temp5 <- cumprod(1-pred.eur.temp5)
lines(pred.eur.temp5~times, type="l", ylim=c(0,1),
xlab="Time (Years since surgery)",
ylab="Survival probability", lty=2, col=rainbows[5], lwd=2)
lines(as.vector(eur.cox.poor) ~ as.vector(sort(dat.test$time)),
type="l",
ylim=c(0,1), xlab="Time (days)",
ylab="Survival probability", lty=1,
lwd=2, col=rainbows[5])
pred.eur.pl.temp5 <- predict(new.n.net.5, eur.pat.poor.lung,
type="raw")
pred.eur.pl.temp5 <- cumprod(1-pred.eur.pl.temp5)
lines(pred.eur.pl.temp5~times, type="l", ylim=c(0,1),
xlab="Time (days)",
ylab="Survival probability", lty=2,
col=rainbows[6], lwd=2)
lines(as.vector(eur.cox.poor.lung) ~ as.vector(sort(dat.test$time)),
type="l",
ylim=c(0,1), xlab="Time (days)",
ylab="Survival probability", lty=1,
lwd=2, col=rainbows[6])
legend("bottomleft", legend=c("Reference", "Poor Hist", ...)
"Poor Hist + lung mets", "Cox", "Neural"),
    col=c(rainbows[4:6], "black", "black"),
    lty=c(1,1,1,1,2), lwd=2,bty="n")

# Table accuracy, sensitivity, specificity neural and Cox
final.eur <- final.other.neur(dat.train, dat.test, 5, 0.05)
vals.fin.eur <- cbind(as.vector(final.eur[[1]]),
    c(NA, cox.eur.acc, cox.eur.sens, cox.eur.spec))
rownames(vals.fin.eur) <- c("Error value", "Accuracy",
    "Sensitivity", "Specificity")
colnames(vals.fin.eur) <- c("Neural (5 nodes)", "Cox model")

# illustration neural network itself
library(devtools)
source_url('https://gist.githubusercontent.com/
fawda123/7471137/raw/
466c1474d0a505ff044412703516c34f1a4684a5/
nnet_plot_update.r')
new.n.net.5$coefnames
plot.nnet(d, x.lab=c("Age", "Volume",
    "Hist. poor", "Ex. marginal", "Ex. other",
    "Sex_male", "Lung mets", "Other mets", "Time interval"),
    y.lab="Hazard", pos.col="darkblue", neg.col="red",
    circle.col="paleturquoise")

# prediction matrix for network with 5 nodes, 0.05 decay
special.matrix.r <- pred.matrix

# prediction matrix for network with 5 nodes, 0.001 decay
# pred matrix is obtained by running network on training set,
# testing on test set (as described in cross-validation section)
special.matrix <- pred.matrix

# making colour vector based on histological response
col.vec.hist <- as.numeric(dat.test$Hist)
col.vec.hist[which(col.vec.hist==1)] <- "#0000FF66"
col.vec.hist[which(col.vec.hist==2)] <- "#FF000066"
# individual survival probabilities hist: network 5 nodes, 0.05
obj <- matplot(t(special.matrix.r), type="l", lwd=1, lty=1,
col=col.vec.hist, xaxt="n", ylim=c(0,1),
ylab="Survival probability", xlab="Time (years since surgery)"
axis(side=1, at=(seq(0.96, by=12)), labels=seq(0,8))

# individual survival probabilities hist: network 5 nodes, 0.001
obja <- matplot(t(special.matrix), type="l", lwd=1, lty=1,
col=col.vec.hist, xaxt="n", ylim=c(0,1),
ylab="Survival probability", xlab="Time (years since surgery)"
axis(side=1, at=(seq(0.96, by=12)), labels=seq(0,8))

# individual survival probabilities cox
obj2 <- matplot(t(preds.eur.b), type="l", lwd=1,
lty=1, col=col.vec.hist, xaxt="n", ylim=c(0,1),
ylab="Survival probability", xlab="Time (years since surgery)"
axis(side=1, at= (seq(0.650, by=81.25)), labels=seq(0,8))

# making colour vector based on lung metastases
col.vec.lung <- as.numeric(dat.test$lung)
col.vec.lung[which(col.vec.lung==1)] <- "#0000FF66"
col.vec.lung[which(col.vec.lung==2)] <- "#FF000066"

# individual survival probabilities lung: network 5 nodes, 0.05
obj3 <- matplot(t(special.matrix.r), type="l", lwd=1,
lty=1, col=col.vec.lung, xaxt="n", ylim=c(0,1),
ylab="Survival probability", xlab="Time (years since surgery)"
axis(side=1, at=(seq(0.96,by=12)), labels=seq(0,8))

# individual survival probabilities hist: cox
obj4 <- matplot(t(preds.eur.b), type="l", lwd=1, lty=1,
col=col.vec.lung, xaxt="n", ylim=c(0,1),
ylab="Survival probability", xlab="Time (years since surgery)"
axis(side=1, at=(seq(0.650, by=81.25)), labels=seq(0,8))

C.4.2 Cox model

# Fitting cox model
cox.eur <- coxph(Surv(time, stat) ~ Hist + Age + Volume +
Ex + Sex + lung + other, data=dat.train)

# obtaining prediction matrix
preds.eur.b <- predictSurvProb(cox.eur, dat.test, times=dat.test$time)

# Making table with hazard ratios and confidence intervals
tempcoefs <- summary(cox.eur)$coefficients[,c(1,3)]
HRs <- exp(tempcoefs[,1])
HRs <- round(HRs, 3)
lower <- round(exp(tempcoefs[,1]-1.96*tempcoefs[,2]),2)
upper <- round(exp(tempcoefs[,1]+1.96*tempcoefs[,2]),2)
confint <- paste(lower, upper, sep="-")
table.coefs <- cbind(HRs, confint)
colnames(table.coefs) <- c("Hazard ratio", "95% CI")

# Obtaining acc, sens and spec for Cox
preds.eur <- diag(preds.eur.b)
stats.cox.eur <- as.data.frame(cbind(preds.eur, dat.test$time, dat.test$stat))
colnames(stats.cox.eur) <- c("pred", "surv", "stat")
mod.stats.cox.eur <- stats.cox.eur
mod.stats.cox.eur$pred <- 1-round(mod.stats.cox.eur$pred,0)
cox.eur.acc <- length(which(mod.stats.cox.eur$pred==mod.stats.cox.eur$stat))/nrow(mod.stats.cox.eur)
cox.eur.sens <- length(which(mod.stats.cox.eur$pred==1 & mod.stats.cox.eur$stat==1))
cox.eur.spec <- length(which(mod.stats.cox.eur$pred==0 & mod.stats.cox.eur$stat==0))
cox.stattd <- as.data.frame(cbind(dat.test$time, dat.test$stat))
colnames(cox.stattd) <- c("surv", "stat")
cox.pred.matrix <- predictSurvProb(cox.eur, dat.test, times=dat.test$time)
Appendix D

A Random survival forest approach

D.1 \textit{rfsrc} function

\begin{verbatim}
# rfsr function (part 1)
rfsr(formula, data,
    # number of trees grown, 1000 is default
    ntree = 1000,
    # choice of bootstrap approach
    bootstrap = c("by.root", "by.node", "none", "by.user"),
    # parameters to be tuned. mtry specifies number of
    # candidate variables for each split, nodesize the size
    # of the terminal nodes, nodedepth the maximal depth of
    # each node, splitrule the splitting rule and nsplit the
    # number of splitpoints randomly selected on each variable
    mtry = NULL,
    nodesize = NULL,
    nodedepth = NULL,
    splitrule = NULL,
    nsplit = 0,
    # for unsupervised forest (not relevant)
    ytry = NULL,
    yvar.wt = NULL,
\end{verbatim}
# rfsr src function (part 2)

# quantifying variable importance.
# "permute" permutes the OOB cases in the relevant
# variable x. "random" assigns a random daughter node
# at every split involving variable x. "anti" assigns
# the opposite daughter node in the same situation.
# For these three methods the difference in original and
# modified OOB error is calculated per tree and averaged
# over. The two ensemble methods use the same approach but
# directly on the ensemble forest.
importance = c(FALSE, TRUE, "none", "permute", "random",
               "anti", "permute.ensemble", "random.ensemble",
               "anti.ensemble"),

# missing value imputation (not applied)
na.action = c("na.omit", "na.impute"),
nimpute = 1,

# Default is making ensemble calculations for
# observed event times. ntime can be used to specify
# a reduced number of such event times. For each
# specified ntime timepoint, the closest event time
# will be taken.
ntime,

# for competing risk (not relevant)
cause,

# proximity matrix (n*n, with n=number individuals),
# the proximity of two cases is measured by the
# frequency of being in the same terminal node.
# This can be done for in bag, oob, or both at
# the same time.
proximity = FALSE,

# bootstrap size (default is .)
sampsize = NULL,

# for by root bootstrap: sampling with replacement
# (swr) and sampling without replacement (swor).
samptype = c("swr", "swor"),
# rfsr function (part 3)

# additional option for user-defined bootstrap
samp = NULL,

# optional weights for each of n cases, probability
# of being selected for bootstrap.
case.wt = NULL,

# weights, based on normalised entry n, used as a
# multiplier for the split statistic: If entry is
# large, the split statistic will be large,
# encouraging a split on that variable
split.wt = NULL,

# forest weight matrix
forest.wt = FALSE,

# vector of probabilities for selecting variable x as
# candidate for splitting node.
xvar.wt = NULL,
forest = TRUE,
var.used = c(FALSE, "all.trees", "by.tree"),
split.depth = c(FALSE, "all.trees", "by.tree"),
seed = NULL,
do.trace = FALSE,
membership = FALSE,
statistics = FALSE,
tree.err = FALSE,
...)

279
D.2 Performance measures

# Concordance index

cindex.mod.r <- function (data) {
  time <- data$surv; status <- data$stat
  x <- data$pred; n <- length(time)
  ord <- order(time, -status)
  time <- time[ord]; status <- status[ord]
  x <- x[ord]

  wh <- which(status == 1); total <- concordant <- 0

  for (i in wh) {
    for (j in ((i + 1):n)) {
      if (time[j] > time[i]) {
        total <- total + 1
        if (x[j] < x[i])
          concordant <- concordant + 1
        if (x[j] == x[i])
          concordant <- concordant + 0.5
      }
    }
  }

  return (concordant/total)
}

# Brier KL function adapted for Random survival forests

brier.function.r <- function(rfs, stat.t, pred.matrix) {
  # calculating censoring distribution based on training set
  object <- rfs$yvar; so <- Surv(object$time, object$stat)
  time <- so[,1]; ot <- order(time)
  cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)

  # censoring distribution calculated
  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf

  # unique event times are selected
death.times <- unique(time[which(cens==1)])

# selecting censoring probabilities at death times

# index of unique death times

indexx <- seq(1,length(time[which(cens==1)]))

# calculating censoring distribution based on test set.

preds <- pred.matrix; so <- Surv(stat.t$surv, stat.t$stat)
time <- so[,1]; ot <- order(time)
cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)

hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
csurv <- predict(hatcdist, times = time, type = "surv")
csurv[csurv == 0] <- Inf

# The scores are calculated for each time t0, which is a
# unique death time.

time <- death.times

# transposing the prediction matrix

survs <- t(as.matrix(preds))

# the observed times and status for each individual

time.t <- stat.t$surv; cens.t <- stat.t$stat

bsc <- rep(0, length(btime)); bsk <- rep(0, length(btime))

# weighting patient survival predictions with appropriate
# censoring distribution.

for (j in 1:length(btime)) {
    help1 <- as.integer(time.t <= btime[j] & cens.t == 1)
    help2 <- as.integer(time.t > btime[j])
    bsc[j] <- mean((0 - survs[j, ])^2 * help1 * (1/csurv) +
                    (1 - survs[j, ])^2 * help2 * (1/csurv.tt[j]))
    inb <- survs[j, ]
    inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
    inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)
    bsk[j] <- -mean((log(1-(inb))*(help1*(1/csurv) +
                      log(inb)*(1/csurv.tt[j]) *help2))
}
\(\text{ind.
list} \leftarrow \text{list()}\)

\[
\text{for}(i \text{ in } 1: \text{length(rel.vals)})\{
\text{ind.
list}[[i]] \leftarrow \text{which.min}(\text{abs}(\text{btime}-\text{rel.vals}[[i]]))
\}
\]

\(\text{inds} \leftarrow \text{unlist(ind.
list)}\)

\(\text{Brie} \leftarrow \text{bsc[inds]}; \text{KL} \leftarrow \text{bsk[inds]}\)

\(\text{RET} \leftarrow \text{rbind(Brie,KL)}\)

\(\text{colnames}(\text{RET}) \leftarrow \text{round(\text{death.times}[\text{inds}],0)}\)

\(\text{return}(\text{RET})\)

\# Brier–KL function for Cox.

\text{brier.function.cox.m} \leftarrow \text{function}(\text{imp}, \text{preds})\{
\text{so} \leftarrow \text{Surv(imp$\text{surv}, imp$\text{stat})}\)
\text{time} \leftarrow \text{so[,1]}
\text{ot} \leftarrow \text{order(time)}
\text{cens} \leftarrow \text{so[ot,2]}
\text{time} \leftarrow \text{time[ot]}
\text{N} \leftarrow \text{nrow(so)}
\text{hatcdist} \leftarrow \text{prodlim(Surv(time, cens) \sim 1, reverse = TRUE)}\)
\text{csurv} \leftarrow \text{predict(hatcdist, times = time, type = "surv")}\)
\text{csurv[csurv == 0]} \leftarrow \text{Inf}\)
\text{btime} \leftarrow \text{time}
\text{survs} \leftarrow \text{t(as.matrix(preds))}\)
\text{bsc} \leftarrow \text{rep(0, nrow(survs))}\)
\text{bsk} \leftarrow \text{rep(0, nrow(survs))}\)
\text{bsp} \leftarrow \text{rep(0,nrow(survs))}\)
\text{for (j in 1:nrow(survs)) \{\}}
\text{help1} \leftarrow \text{as.integer(time} \leq \text{btime}[j] \& \text{cens} == 1)\)
\text{help2} \leftarrow \text{as.integer(time} > \text{btime}[j])\)
\text{inb} \leftarrow \text{survs}[j,]\)
\text{inb\[\text{which(inb==1)}\]} \leftarrow \text{(inb\[\text{which(inb==1)}\]-0.000001)}\)
\text{inb\[\text{which(inb==0)}\]} \leftarrow \text{(inb\[\text{which(inb==0)}\]+0.000001)}\)
\text{bsc[j]} \leftarrow \text{mean((0} - \text{survs[j, ]})^2 * \text{help1} * (1/csurv) +}
(1 − survs[j,])^2 * help2 * (1/csdrv[j]))

bsk[j] <- -mean((log(1-(inb))*help1*(1/csdrv) +
log(inb)*(1/csdrv[j]) *help2))

bsp[j] <- -mean((log(1-survs[j,])*help1*(1/csdrv) +
log(surv[j,]) *help2*(1/csdrv[j])))

ind.values <- seq(1,8)

rel.vals <- vector("numeric", length(ind.values))
for(i in 1:length(ind.values)){
  rel.vals[[i]] <- which.min(abs(btime-ind.values[i]))
}

Brie <- bsc[rel.vals]
KL <- bsk[rel.vals]

RET <- rbind(Brie,KL)
return(RET)

D.3 Cross-validation

D.3.1 Cross-validation functions

# Cross-validation function
cross.val.rand.for <- function(training.s,
splits=0, mtrys=3,
splitrules="logrank",
nodesizes=NULL, nodedepths=NULL){
  D <- 5
  index <- rep(1:D, floor(nrow(training.s)/D)+1)[1:nrow(training.s)]
  set.seed(92874)
  fold.index <- sample(index)
  vals <- vector("numeric", D)
  briers <- NULL
  KLs <- NULL
  HR <- vector("numeric", D)
  LRX<- vector("numeric", D)
  for (d in 1:D){
    ind.train <- which(fold.index!=d)
ind.test <- which(fold.index==d)

training.p <- training.s[ind.train,]
test.p <- training.s[ind.test,]
test.p <- test.p[order(test.p$time),]

rfs <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = training.p, 
nsplit = nsplits, splitrule=splitrules, 
mtry=mtry, nodesize=nodesizes, nodedepth=nodedepths, 
tree.err = FALSE, importance = FALSE, ntree=750)

test.pred.d <- predict.rfsrc(rfs, newdata = test.p)
pred.s <- test.pred.d$survival

pred.scores <- test.pred.d$predicted

stat.t <- as.data.frame(cbind(test.p$time, test.p$stat, pred.scores)
colnames(stat.t) <- c("surv", "stat", "preds")

vals[d] <- cindex.mod.r(stat.t)
brierKL <- brier.function.r(rfs, stat.t, pred.s)
briers <- rbind(briers, brierKL[1,])
KLs <- rbind(KLs, brierKL[2,])

# calculating hazard ratio from normalized prognostic index
pred.scores.norm <- (pred.scores - min(pred.scores)) / 
(max(pred.scores) - min(pred.scores))
cv <- coxph(Surv(time, stat) ~ pred.scores.norm, 
data=cbind(test.p, stat) ~ pred.scores.norm)
HR[d] <- exp(cv$coefficients)

# obtaining logrank Chisq statistic based on median prognostic index

group <- rep(1,nrow(test.p))
group[which(pred.scores<=median(pred.scores))] <- 2
group <- as.factor(group); temp.dat <- cbind(test.p, group)
LRX[d] <- survdiff(Surv(time, stat)~group, data=temp.dat)$chisq

}

lst <- list()
lst[[1]] <- as.data.frame(vals)
lst[[2]] <- briers
lst[[3]] <- KLs
1st[[4]] <- as.data.frame(HR)
1st[[5]] <- as.data.frame(LRX)

obj <- 1st
return(obj)
}

# wrap function for cross-validation

cross.val.plots.rand <- function(mtrys, nodesizes, nsplits=0,
                       nodedepths=NULL, option=c("first", "second")){
    if(option=="first"){
        objects <- list()
        for (i in 1:length(nodesizes)){
            objects[[i]] <- cross.val.rand.for(dat.train,mtrys=mtrys,
                                               nodesizes=nodesizes[i],nsplits=nsplits)
        }
    }else{
        objects <- list()
        for (i in 1:length(nodedepths)){
            objects[[i]] <- cross.val.rand.for(dat.train,mtrys=mtrys,
                                               nodesizes=nodesizes, nsplits=nsplits,
                                               nodedepths=nodedepths[i])
        }
        nodedepths <- nodedepths
    }

    # the means of the measures are calculated over the folds
    means <- list()
    for (i in 1:length(objects)){
        means[[i]] <- lapply(objects[[i]], colMeans)
    }

    # the sds of the measures are calculated over the folds
    sds <- list()
    for (i in 1:length(objects)){
        sds[[i]] <- lapply(objects[[i]], function(x) {
            sds <- vector("numeric", ncol(x))
            for (i in 1:ncol(x)){
                sds[i] <- sd(x[,i])
            }
            return(sds)
        })
    }
# the means for error value, accuracy, sensitivity, specificity are collected in a matrix
single.vals.0.m <- matrix(NA, nrow=length(nodesizes), ncol=1)
for (i in 1:length(nodesizes)){
  single.vals.0.m[i,] <- means[[i]][[1]]
}

# the sds for error value, accuracy, sensitivity, specificity are collected in a matrix
single.vals.0.s <- matrix(NA, nrow=length(nodesizes), ncol=1)
for (i in 1:length(nodesizes)){
  single.vals.0.s[i,] <- sds[[i]][[1]]
}

# the means for the brier scores are collected in a matrix
briers <- matrix(NA, nrow=length(nodesizes), ncol=6)
for (i in 1:length(nodesizes)){
  briers[i,] <- means[[i]][[2]]
}

# the sds for the brier scores are collected in a matrix
briers.s <- matrix(NA, nrow=length(nodesizes), ncol=6)
for (i in 1:length(nodesizes)){
  briers.s[i,] <- sds[[i]][[2]]
}

# the means for the KL scores are collected in a matrix
KLs <- matrix(NA, nrow=length(nodesizes), ncol=6)
for (i in 1:length(nodesizes)){
  KLs[i,] <- means[[i]][[3]]
}

# the sds for the KL scores are collected in a matrix
KLs.s <- matrix(NA, nrow=length(nodesizes), ncol=6)
for (i in 1:length(nodesizes)){
  KLs.s[i,] <- sds[[i]][[3]]
}

# all the matrices are saved in a list
bothlist <- list()
bothlist[[1]] <- single.vals.0.m
bothlist[[2]] <- briers
bothlist[[3]] <- KLs
both.list.s <- list()
both.list.s[[1]] <- single.vals.0.s
both.list.s[[2]] <- briers.s
both.list.s[[3]] <- KLs.s

all.list <- list()
all.list[[1]] <- bothlist
all.list[[2]] <- both.list.s

return(all.list)

# function for training random forest on train data, and
# testing on test data (for after cross-validation)
final.rand.for <- function(train, test, nsplits=3, mtrys=2,
    splitrules="logrank", nodesizes=200, nodedepths=7){
    rfs <- rfs.src(Surv(time, stat) ~ Hist + Age + Volume + Ex +
        Sex + lung + other, data = train,
        nsplit = nsplits, splitrule=splitrules,
        mtry=mtrys, nodesize=nodesizes, nodedepth=nodedepths,
        tree.err = TRUE, importance = TRUE, ntree =750)

test.pred.d <- predict.rfs.src(rfs, newdata = test)
pred.s <- test.pred.d$predicted
pred.s <- pred.s[order(test$time),]
pred.scores <- test.pred.d$predicted
stat.t <- as.data.frame(cbind(test$time, test$stat, pred.scores))
colnames(stat.t) <- c("surv", "stat", "preds")
vals <- cindex.mod.r(stat.t)
brierKL <- brier.function.r(rfs, stat.t, pred.s)
briers <- brierKL[1,]
KLs <- brierKL[2,]

# calculating hazard ratio from normalized prognostic index
pred.scores.norm <- (pred.scores - min(pred.scores))/
    (max(pred.scores) - min(pred.scores))
cv <- coxph(Surv(time, stat) ~ pred.scores.norm,
        data=cbind(test, pred.scores.norm))
HR <- \exp (cv$coefficients)

# obtaining logrank Chisq statistic based on median prognostic index
group <- rep(1, nrow(test))
group[which(pred.scores<=\text{median}(pred.scores))] <- 2
group <- \text{as.factor}(group)
temp.dat <- cbind(test, group)
LRX <- \text{survdiff}(\text{Surv}(time, stat) \sim \text{group}, \text{data}=temp.dat)$\text{chisq}

lst <- \text{list}()
lst[1] <- \text{as.data.frame}(vals)
lst[2] <- briers
lst[3] <- KLs
lst[4] <- HR
lst[5] <- LRX
obj <- lst

\text{return}(obj)

\textbf{D.3.2 Cross-validation for node size and number of candidate splitting variables}

\texttt{out.r} <- \text{cross.val.plots.rand}(\text{mtry}=mtry[1],
\text{nodesizes}=nodesizes, \text{nslits}=0,
\text{nodedepths}=\text{NULL}, \text{option}="\text{first}"
)\text{means.r.0} <- \text{out.r}[1]
\text{sds.r.0} <- \text{out.r}[2]

\text{out.r.1} <- \text{cross.val.plots.rand}(\text{mtry}=mtry[2],
\text{nodesizes}=nodesizes, \text{nslits}=0,
\text{nodedepths}=\text{NULL}, \text{option}="\text{first}"
)\text{means.r.1} <- \text{out.r.1}[1]
\text{sds.r.1} <- \text{out.r.1}[2]

\text{out.r.2} <- \text{cross.val.plots.rand}(\text{mtry}=mtry[3],
\text{nodesizes}=nodesizes, \text{nslits}=0,
\text{nodedepths}=\text{NULL}, \text{option}="\text{first}"
)\text{means.r.2} <- \text{out.r.2}[1]
\text{sds.r.2} <- \text{out.r.2}[2]

\text{out.r.3} <- \text{cross.val.plots.rand}(\text{mtry}=mtry[4],}
nodesizes=nodesizes, nsplits=0,
nodedepths=NULL, option="first")
means.r.3 <- out.r.3[1]
sds.r.3 <- out.r.3[2]

out.r.4 <- cross.val.plots.rand(mtry[5],
nodesizes=nodesizes, nsplits=0,
nodedepths=NULL, option="first")
means.r.4 <- out.r.4[1]
sds.r.4 <- out.r.4[2]

out.r.5 <- cross.val.plots.rand(mtry[6],
nodesizes=nodesizes, nsplits=0,
nodedepths=NULL, option="first")
means.r.5 <- out.r.5[1]
sds.r.5 <- out.r.5[2]

out.r.6 <- cross.val.plots.rand(mtry[7],
nodesizes=nodesizes, nsplits=0,
nodedepths=NULL, option="first")
means.r.6 <- out.r.6[1]
sds.r.6 <- out.r.6[2]

vall <- cbind(means.r.0[1][1][1], means.r.1[1][1][1],
means.r.2[1][1][1], means.r.3[1][1][1],
means.r.4[1][1][1], means.r.5[1][1][1],
means.r.6[1][1][1])
vall.sd <- cbind(sds.r.0[1][1][1], sds.r.1[1][1][1],
sds.r.2[1][1][1], sds.r.3[1][1][1],
sds.r.4[1][1][1], sds.r.5[1][1][1],
sds.r.6[1][1][1])

vall1 <- vall + vall.sd
vall2 <- vall - vall.sd

plot(vall[,1]~nodesizes, pch=16, ylab="C-index",
xlab="Node size",
main="1 candidate splitting variable", ylim=c(0.45,0.75))
arrows(nodesizes, vall2[,1], nodesizes, vall1[,1],
length=0.05, angle=90, code=3)

plot(vall[,2]~nodesizes, pch=16, ylab="C-index",
xlab="Node size",
main="2 candidate splitting variables", ylim=c(0.45,0.75))
```r
arrows(nodesizes, vall2[,2], nodesizes, vall1[,2],
  length=0.05, angle=90, code=3)
plot(vall[,3]~nodesizes, pch=16, ylab="C-index",
  xlab="Node size",
  main="3 candidates splitting variables", ylim=c(0.45, 0.75))
arrows(nodesizes, vall2[,3], nodesizes, vall1[,3],
  length=0.05, angle=90, code=3)
plot(vall[,4]~nodesizes, pch=16, ylab="C-index",
  xlab="Node size",
  main="4 candidates splitting variables", ylim=c(0.45, 0.75))
arrows(nodesizes, vall2[,4], nodesizes, vall1[,4],
  length=0.05, angle=90, code=3)
plot(vall[,5]~nodesizes, pch=16, ylab="C-index",
  xlab="Node size",
  main="5 candidates splitting variables", ylim=c(0.45, 0.75))
arrows(nodesizes, vall2[,5], nodesizes, vall1[,5],
  length=0.05, angle=90, code=3)
plot(vall[,6]~nodesizes, pch=16, ylab="C-index",
  xlab="Node size",
  main="6 candidates splitting variables", ylim=c(0.45, 0.75))
arrows(nodesizes, vall2[,6], nodesizes, vall1[,6],
  length=0.05, angle=90, code=3)
plot(vall[,7]~nodesizes, pch=16, ylab="C-index",
  xlab="Node size",
  main="7 candidates splitting variables", ylim=c(0.45, 0.75))
arrows(nodesizes, vall2[,7], nodesizes, vall1[,7],
  length=0.05, angle=90, code=3)

# brier/KL scores: per variable size. colours for node size.

brier.rf <- list()
brier.rf[[1]] <- means.r.0[[1]][[2]]
brier.rf[[2]] <- means.r.1[[1]][[2]]
brier.rf[[3]] <- means.r.2[[1]][[2]]
brier.rf[[4]] <- means.r.3[[1]][[2]]
brier.rf[[5]] <- means.r.4[[1]][[2]]
brier.rf[[6]] <- means.r.5[[1]][[2]]
brier.rf[[7]] <- means.r.6[[1]][[2]]
```

brier.rf.s1 <- list()
  brier.rf.s1[1] <- means.r.0[1][2] + sds.r.0[1][2]
  brier.rf.s1[2] <- means.r.1[1][2] + sds.r.1[1][2]
  brier.rf.s1[3] <- means.r.2[1][2] + sds.r.2[1][2]
  brier.rf.s1[4] <- means.r.3[1][2] + sds.r.3[1][2]
  brier.rf.s1[5] <- means.r.4[1][2] + sds.r.4[1][2]
  brier.rf.s1[6] <- means.r.5[1][2] + sds.r.5[1][2]
  brier.rf.s1[7] <- means.r.6[1][2] + sds.r.6[1][2]

brier.rf.s2 <- list()
  brier.rf.s2[1] <- means.r.0[1][2] - sds.r.0[1][2]
  brier.rf.s2[3] <- means.r.2[1][2] - sds.r.2[1][2]
  brier.rf.s2[4] <- means.r.3[1][2] - sds.r.3[1][2]
  brier.rf.s2[5] <- means.r.4[1][2] - sds.r.4[1][2]
  brier.rf.s2[6] <- means.r.5[1][2] - sds.r.5[1][2]
  brier.rf.s2[7] <- means.r.6[1][2] - sds.r.6[1][2]

KL.rf <- list()
  KL.rf[1] <- means.r.0[1][3]
  KL.rf[2] <- means.r.1[1][3]
  KL.rf[3] <- means.r.2[1][3]
  KL.rf[4] <- means.r.3[1][3]
  KL.rf[5] <- means.r.4[1][3]
  KL.rf[6] <- means.r.5[1][3]
  KL.rf[7] <- means.r.6[1][3]

KL.rf.s1 <- list()
  KL.rf.s1[1] <- means.r.0[1][3] + sds.r.0[1][3]
  KL.rf.s1[2] <- means.r.1[1][3] + sds.r.1[1][3]
  KL.rf.s1[3] <- means.r.2[1][3] + sds.r.2[1][3]
  KL.rf.s1[4] <- means.r.3[1][3] + sds.r.3[1][3]
  KL.rf.s1[5] <- means.r.4[1][3] + sds.r.4[1][3]
  KL.rf.s1[6] <- means.r.5[1][3] + sds.r.5[1][3]
  KL.rf.s1[7] <- means.r.6[1][3] + sds.r.6[1][3]

KL.rf.s2 <- list()
  KL.rf.s2[1] <- means.r.0[1][3] - sds.r.0[1][3]
  KL.rf.s2[2] <- means.r.1[1][3] - sds.r.1[1][3]
  KL.rf.s2[3] <- means.r.2[1][3] - sds.r.2[1][3]
```r
KL.rf.s2[[4]] <- means.r.3[[1]][[3]] - sds.r.3[[1]][[3]]
KL.rf.s2[[5]] <- means.r.4[[1]][[3]] - sds.r.4[[1]][[3]]
KL.rf.s2[[6]] <- means.r.5[[1]][[3]] - sds.r.5[[1]][[3]]
KL.rf.s2[[7]] <- means.r.6[[1]][[3]] - sds.r.6[[1]][[3]]

rainbows <- rainbow(12)

rainboww <- c("tan4", rainboww[c(2, 4, 7, 8, 10, 11, 12)], 
               "pink", "darkgreen", "darkred")

legend("bottomleft", legend=nodesizes, col=c("black", rainbows), 
       lwd=10, ncol=3)

xx <- seq(1, 6)

plot((brier.rf[[1]][1,]) ~ xx, type="l", ylim=c(0, 0.3), 
     ylab="Brier score", xlab="Time (years since surgery)", lwd=2, 
     main="1 candidate splitting variable")

arrows(xx, brier.rf.s2[[1]][1,], xx, brier.rf.s1[[1]][1,], 
       length=0.05, angle=90, code=3)

lines((brier.rf[[1]][2,]) ~ xx, type="l", ylim=c(0, 0.3), 
      col=rainbows[1], lwd=2)

arrows(xx, brier.rf.s2[[1]][2,], xx, brier.rf.s1[[1]][2,], 
       length=0.05, 
       angle=90, code=3, col=rainbows[1])

lines((brier.rf[[1]][3,]) ~ xx, type="l", ylim=c(0, 0.3), 
      col=rainbows[2]
      , lwd=2)

arrows(xx, brier.rf.s2[[1]][3,], xx, brier.rf.s1[[1]][3,], 
       length=0.05, 
       angle=90, code=3, col=rainbows[2])

lines((brier.rf[[1]][4,]) ~ xx, type="l", ylim=c(0, 0.3), 
      col=rainbows[3]
      , lwd=2)

arrows(xx, brier.rf.s2[[1]][4,], xx, brier.rf.s1[[1]][4,], 
       length=0.05, 
       angle=90, code=3, col=rainbows[3])

lines((brier.rf[[1]][5,]) ~ xx, type="l", ylim=c(0, 0.3), 
      col=rainbows[4]
      , lwd=2)

arrows(xx, brier.rf.s2[[1]][5,], xx, brier.rf.s1[[1]][5,], 
       length=0.05, 
       angle=90, code=3, col=rainbows[4])

lines((brier.rf[[1]][6,]) ~ xx, type="l", ylim=c(0, 0.3), 
      col=rainbows[5])
```
arrows(xx, brier.rf.s2[[1]][6], xx, brier.rf.s1[[1]][6],
  length=0.05,
  angle=90, code=3, col=rainbows[5])
lines((brier.rf[[1]][7])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[6]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][7], xx, brier.rf.s1[[1]][7],
  length=0.05,
  angle=90, code=3, col=rainbows[6])
lines((brier.rf[[1]][8])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[7]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][8], xx, brier.rf.s1[[1]][8],
  length=0.05,
  angle=90, code=3, col=rainbows[7])
lines((brier.rf[[1]][9])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[8]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][9], xx, brier.rf.s1[[1]][9],
  length=0.05,
  angle=90, code=3, col=rainbows[8])
lines((brier.rf[[1]][10])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[9]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][10], xx, brier.rf.s1[[1]][10],
  length=0.05,
  angle=90, code=3, col=rainbows[9])
lines((brier.rf[[1]][11])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[10]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][11], xx, brier.rf.s1[[1]][11],
  length=0.05,
  angle=90, code=3, col=rainbows[10])
lines((brier.rf[[1]][12])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[11]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][12], xx, brier.rf.s1[[1]][12],
  length=0.05,
  angle=90, code=3, col=rainbows[11])
lines((brier.rf[[1]][13])~xx, type="l", ylim=c(0, 0.3),
  col=rainbows[12]
, lwd=2)
arrows(xx, brier.rf.s2[[1]][13], xx, brier.rf.s1[[1]][13],
  length=0.05,
angle=90, code=3, col=rainbows[12])

plot((brier.rf[[2]][1,])~xx, type="l", ylim=c(0,0.3),
ylab="Brier score", xlab="Time (years since surgery)", lwd=2,
main="2 candidate splitting variables")

arrows(xx, brier.rf.s2[[2]][1,], xx, brier.rf.s1[[2]][1,],
length=0.05, angle=90, code=3)

lines((brier.rf[[2]][2,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[1], lwd=2)

arrows(xx, brier.rf.s2[[2]][2,], xx, brier.rf.s1[[2]][2,],
length=0.05, angle=90, code=3, col=rainbows[1])

lines((brier.rf[[2]][3,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[2], lwd=2)

arrows(xx, brier.rf.s2[[2]][3,], xx, brier.rf.s1[[2]][3,],
length=0.05, angle=90, code=3, col=rainbows[2])

lines((brier.rf[[2]][4,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[3], lwd=2)

arrows(xx, brier.rf.s2[[2]][4,], xx, brier.rf.s1[[2]][4,],
length=0.05, angle=90, code=3, col=rainbows[3])

lines((brier.rf[[2]][5,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[4], lwd=2)

arrows(xx, brier.rf.s2[[2]][5,], xx, brier.rf.s1[[2]][5,],
length=0.05, angle=90, code=3, col=rainbows[4])

lines((brier.rf[[2]][6,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[5], lwd=2)

arrows(xx, brier.rf.s2[[2]][6,], xx, brier.rf.s1[[2]][6,],
length=0.05, angle=90, code=3, col=rainbows[5])

lines((brier.rf[[2]][7,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[6], lwd=2)

arrows(xx, brier.rf.s2[[2]][7,], xx, brier.rf.s1[[2]][7,],
length=0.05, angle=90, code=3, col=rainbows[6])

lines((brier.rf[[2]][8,])~xx, type="l", ylim=c(0,0.3),
```r
col = rainbows[7]

arrows(xx, brier.rf.s2[[2]][8], xx, brier.rf.s1[[2]][8],
length = 0.05,
angle = 90, code = 3, col = rainbows[7])
lines((brier.rf[[2]][9]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[8], lwd = 2)

arrows(xx, brier.rf.s2[[2]][9], xx, brier.rf.s1[[2]][9],
length = 0.05,
angle = 90, code = 3, col = rainbows[8])
lines((brier.rf[[2]][10]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[9], lwd = 2)

arrows(xx, brier.rf.s2[[2]][10], xx, brier.rf.s1[[2]][10],
length = 0.05,
angle = 90, code = 3, col = rainbows[9])
lines((brier.rf[[2]][11]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[10], lwd = 2)

arrows(xx, brier.rf.s2[[2]][11], xx, brier.rf.s1[[2]][11],
length = 0.05,
angle = 90, code = 3, col = rainbows[10])
lines((brier.rf[[2]][12]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[11], lwd = 2)

arrows(xx, brier.rf.s2[[2]][12], xx, brier.rf.s1[[2]][12],
length = 0.05,
angle = 90, code = 3, col = rainbows[11])
lines((brier.rf[[2]][13]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[12], lwd = 2)

plot((brier.rf[[3]][1]) ~ xx, type = "1", ylim = c(0, 0.3),
ylab = "Brier score", xlab = "Time (years since surgery)", lwd = 2,
main = "3 candidate splitting variables")

arrows(xx, brier.rf.s2[[3]][1], xx, brier.rf.s1[[3]][1],
length = 0.05, angle = 90, code = 3)
lines((brier.rf[[3]][2]) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[1], lwd = 2)
```
arrows(xx, brier.rf.s2[[3]][2], xx, brier.rf.s1[[3]][2]),
length=0.05,
angle=90, code=3, col=rainbows[1])
lines((brier.rf[[3]][3,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[2], lwd=2)
arrows(xx, brier.rf.s2[[3]][3], xx, brier.rf.s1[[3]][3]),
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((brier.rf[[3]][4,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[3], lwd=2)
arrows(xx, brier.rf.s2[[3]][4], xx, brier.rf.s1[[3]][4]),
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((brier.rf[[3]][5,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[4])
arrows(xx, brier.rf.s2[[3]][5], xx, brier.rf.s1[[3]][5]),
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((brier.rf[[3]][6,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[5], lwd=2)
arrows(xx, brier.rf.s2[[3]][6], xx, brier.rf.s1[[3]][6]),
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((brier.rf[[3]][7,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[6], lwd=2)
arrows(xx, brier.rf.s2[[3]][7], xx, brier.rf.s1[[3]][7]),
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((brier.rf[[3]][8,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[7])
arrows(xx, brier.rf.s2[[3]][8], xx, brier.rf.s1[[3]][8]),
length=0.05,
angle=90, code=3, col=rainbows[7])
lines((brier.rf[[3]][9,])~xx, type="1", ylim=c(0,0.3),
col=rainbows[8], lwd=2)
arrows(xx, brier.rf.s2[[3]][9], xx, brier.rf.s1[[3]][9]),
length=0.05,
angle=90, code=3, col=rainbows[8])
lines((brier.rf[[3]][10,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[9], lwd=2)
arrows(xx, brier.rf.s2[[3]][10,], xx, brier.rf.s1[[3]][10,],
  length=0.05,
  angle=90, code=3, col=rainbows[9])
lines((brier.rf[[3]][11,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[10], lwd=2)
arrows(xx, brier.rf.s2[[3]][11,], xx, brier.rf.s1[[3]][11,],
  length=0.05,
  angle=90, code=3, col=rainbows[10])
lines((brier.rf[[3]][12,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[11], lwd=2)
arrows(xx, brier.rf.s2[[3]][12,], xx, brier.rf.s1[[3]][12,],
  length=0.05,
  angle=90, code=3, col=rainbows[11])
lines((brier.rf[[3]][13,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[12], lwd=2)
arrows(xx, brier.rf.s2[[3]][13,], xx, brier.rf.s1[[3]][13,],
  length=0.05,
  angle=90, code=3, col=rainbows[12])

plot((brier.rf[[4]][1,])~xx, type="l", ylim=c(0,0.3),
  ylab="Brier score", xlab="Time (years since surgery)", lwd=2,
  main="4 candidate splitting variables")
arrows(xx, brier.rf.s2[[4]][1,], xx, brier.rf.s1[[4]][1,],
  length=0.05, angle=90, code=3)
lines((brier.rf[[4]][2,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[1], lwd=2)
arrows(xx, brier.rf.s2[[4]][2,], xx, brier.rf.s1[[4]][2,],
  length=0.05,
  angle=90, code=3, col=rainbows[1])
lines((brier.rf[[4]][3,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[2], lwd=2)
arrows(xx, brier.rf.s2[[4]][3,], xx, brier.rf.s1[[4]][3,],
  length=0.05,
  angle=90, code=3, col=rainbows[2])
lines((brier.rf[[4]][4,])~xx, type="l", ylim=c(0,0.3),
  col=rainbows[3]
428 , lwd=2)
429 arrows(xx, brier.rf.s2[[4]][4], xx, brier.rf.s1[[4]][4],
430 length=0.05,
431 angle=90, code=3, col=rainbows[3])
432 lines((brier.rf[[4]][5,])~xx, type="1", ylim=c(0,0.3),
433 col=rainbows[4]
434 , lwd=2)
435 arrows(xx, brier.rf.s2[[4]][5], xx, brier.rf.s1[[4]][5],
436 length=0.05,
437 angle=90, code=3, col=rainbows[4])
438 lines((brier.rf[[4]][6,])~xx, type="1", ylim=c(0,0.3),
439 col=rainbows[5]
440 , lwd=2)
441 arrows(xx, brier.rf.s2[[4]][6], xx, brier.rf.s1[[4]][6],
442 length=0.05,
443 angle=90, code=3, col=rainbows[5])
444 lines((brier.rf[[4]][7,])~xx, type="1", ylim=c(0,0.3),
445 col=rainbows[6]
446 , lwd=2)
447 arrows(xx, brier.rf.s2[[4]][7], xx, brier.rf.s1[[4]][7],
448 length=0.05,
449 angle=90, code=3, col=rainbows[6])
450 lines((brier.rf[[4]][8,])~xx, type="1", ylim=c(0,0.3),
451 col=rainbows[7]
452 , lwd=2)
453 arrows(xx, brier.rf.s2[[4]][8], xx, brier.rf.s1[[4]][8],
454 length=0.05,
455 angle=90, code=3, col=rainbows[7])
456 lines((brier.rf[[4]][9,])~xx, type="1", ylim=c(0,0.3),
457 col=rainbows[8]
458 , lwd=2)
459 arrows(xx, brier.rf.s2[[4]][9], xx, brier.rf.s1[[4]][9],
460 length=0.05,
461 angle=90, code=3, col=rainbows[8])
462 lines((brier.rf[[4]][10,])~xx, type="1", ylim=c(0,0.3),
463 col=rainbows[9]
464 , lwd=2)
465 arrows(xx, brier.rf.s2[[4]][10], xx, brier.rf.s1[[4]][10],
466 length=0.05,
467 angle=90, code=3, col=rainbows[9])
468 lines((brier.rf[[4]][11,])~xx, type="1", ylim=c(0,0.3),
469 col=rainbows[10]
470 , lwd=2)
471 arrows(xx, brier.rf.s2[[4]][11], xx, brier.rf.s1[[4]][11],
472 length=0.05,
angle = 90, code = 3, col = rainbows[10]
lines((brier.rf[[4]][12],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[11]
, lwd = 2)
arrows(xx, brier.rf.s2[[4]][12], xx, brier.rf.s1[[4]][12],
length = 0.05,
angle = 90, code = 3, col = rainbows[11])
lines((brier.rf[[4]][13],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[12]
, lwd = 2)
arrows(xx, brier.rf.s2[[4]][13], xx, brier.rf.s1[[4]][13],
length = 0.05,
angle = 90, code = 3, col = rainbows[12])

plot((brier.rf[[5]][1],) ~ xx, type = "1", ylim = c(0, 0.3),
ylab = "Brier score", xlab = "Time (years since surgery)", lwd = 2,
main = "5 candidate splitting variables")
arrows(xx, brier.rf.s2[[5]][1], xx, brier.rf.s1[[5]][1],
length = 0.05, angle = 90, code = 3)
lines((brier.rf[[5]][2],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[1]
, lwd = 2)
arrows(xx, brier.rf.s2[[5]][2], xx, brier.rf.s1[[5]][2],
length = 0.05,
angle = 90, code = 3, col = rainbows[1])
lines((brier.rf[[5]][3],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[2]
, lwd = 2)
arrows(xx, brier.rf.s2[[5]][3], xx, brier.rf.s1[[5]][3],
length = 0.05,
angle = 90, code = 3, col = rainbows[2])
lines((brier.rf[[5]][4],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[3]
, lwd = 2)
arrows(xx, brier.rf.s2[[5]][4], xx, brier.rf.s1[[5]][4],
length = 0.05,
angle = 90, code = 3, col = rainbows[3])
lines((brier.rf[[5]][5],) ~ xx, type = "1", ylim = c(0, 0.3),
col = rainbows[4]
, lwd = 2)
arrows(xx, brier.rf.s2[[5]][5], xx, brier.rf.s1[[5]][5],
length = 0.05,
angle = 90, code = 3, col = rainbows[4])
lines((brier.rf[[5]][6],) ~ xx, type = "1", ylim = c(0, 0.3),
col=rainbows[5], lwd=2)
      arrows(xx, brier.rf.s2[[5]][6,], xx, brier.rf.s1[[5]][6,],
             length=0.05,
             angle=90, code=3, col=rainbows[5])
      lines((brier.rf[[5]][7,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[6], lwd=2)
      arrows(xx, brier.rf.s2[[5]][7,], xx, brier.rf.s1[[5]][7,],
             length=0.05,
             angle=90, code=3, col=rainbows[6])
      lines((brier.rf[[5]][8,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[7], lwd=2)
      arrows(xx, brier.rf.s2[[5]][8,], xx, brier.rf.s1[[5]][8,],
             length=0.05,
             angle=90, code=3, col=rainbows[7])
      lines((brier.rf[[5]][9,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[8], lwd=2)
      arrows(xx, brier.rf.s2[[5]][9,], xx, brier.rf.s1[[5]][9,],
             length=0.05,
             angle=90, code=3, col=rainbows[8])
      lines((brier.rf[[5]][10,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[9], lwd=2)
      arrows(xx, brier.rf.s2[[5]][10,], xx, brier.rf.s1[[5]][10,],
             length=0.05,
             angle=90, code=3, col=rainbows[9])
      lines((brier.rf[[5]][11,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[10], lwd=2)
      arrows(xx, brier.rf.s2[[5]][11,], xx, brier.rf.s1[[5]][11,],
             length=0.05,
             angle=90, code=3, col=rainbows[10])
      lines((brier.rf[[5]][12,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[11], lwd=2)
      arrows(xx, brier.rf.s2[[5]][12,], xx, brier.rf.s1[[5]][12,],
             length=0.05,
             angle=90, code=3, col=rainbows[11])
      lines((brier.rf[[5]][13,])~xx, type="l", ylim=c(0,0.3),
             col=rainbows[12], lwd=2)
      arrows(xx, brier.rf.s2[[5]][13,], xx, brier.rf.s1[[5]][13],
             col=rainbows[12], lwd=2)
length=0.05, angle=90, code=3, col=rainbows[12])

plot((brier.rf[[6]][1,])~xx, type="l", ylim=c(0,0.3), ylab="Brier score", xlab="Time (years since surgery)", lwd=2, main="6 candidate splitting variables")

arrows(xx, brier.rf.s2[[6]][1,], xx, brier.rf.s1[[6]][1,], length=0.05, angle=90, code=3)

lines((brier.rf[[6]][2,])~xx, type="l", ylim=c(0,0.3), col=rainbows[1], lwd=2)

arrows(xx, brier.rf.s2[[6]][2,], xx, brier.rf.s1[[6]][2,], length=0.05, angle=90, code=3, col=rainbows[1])

lines((brier.rf[[6]][3,])~xx, type="l", ylim=c(0,0.3), col=rainbows[2], lwd=2)

arrows(xx, brier.rf.s2[[6]][3,], xx, brier.rf.s1[[6]][3,], length=0.05, angle=90, code=3, col=rainbows[2])

lines((brier.rf[[6]][4,])~xx, type="l", ylim=c(0,0.3), col=rainbows[3], lwd=2)

arrows(xx, brier.rf.s2[[6]][4,], xx, brier.rf.s1[[6]][4,], length=0.05, angle=90, code=3, col=rainbows[3])

lines((brier.rf[[6]][5,])~xx, type="l", ylim=c(0,0.3), col=rainbows[4], lwd=2)

arrows(xx, brier.rf.s2[[6]][5,], xx, brier.rf.s1[[6]][5,], length=0.05, angle=90, code=3, col=rainbows[4])

lines((brier.rf[[6]][6,])~xx, type="l", ylim=c(0,0.3), col=rainbows[5], lwd=2)

arrows(xx, brier.rf.s2[[6]][6,], xx, brier.rf.s1[[6]][6,], length=0.05, angle=90, code=3, col=rainbows[5])

lines((brier.rf[[6]][7,])~xx, type="l", ylim=c(0,0.3), col=rainbows[6], lwd=2)

arrows(xx, brier.rf.s2[[6]][7,], xx, brier.rf.s1[[6]][7,], length=0.05,
angle = 90, code = 3, col = rainbows[6])

lines((brier.rf[[6]][8,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[7]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][8,], xx, brier.rf.s1[[6]][8,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[7])

lines((brier.rf[[6]][9,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[8]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][9,], xx, brier.rf.s1[[6]][9,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[8])

lines((brier.rf[[6]][9,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[9]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][9,], xx, brier.rf.s1[[6]][9,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[9])

lines((brier.rf[[6]][10,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[10]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][10,], xx, brier.rf.s1[[6]][10,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[10])

lines((brier.rf[[6]][11,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[11]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][11,], xx, brier.rf.s1[[6]][11,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[11])

lines((brier.rf[[6]][12,]) ~ xx, type = "1", ylim = c(0, 0.3),
    col = rainbows[12]
    , lwd = 2)

arrows(xx, brier.rf.s2[[6]][12,], xx, brier.rf.s1[[6]][12,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[12])

plot((brier.rf[[7]][1,]) ~ xx, type = "1", ylim = c(0, 0.3),
    ylab = "Brier score" , xlab = "Time (years since surgery)" ,
    lwd = 2,
    main = "7 candidates splitting variables"
)

arrows(xx, brier.rf.s2[[7]][1,], xx, brier.rf.s1[[7]][1,],
    length = 0.05, angle = 90, code = 3)

lines((brier.rf[[7]][2,]) ~ xx, type = "1", ylim = c(0, 0.3),
    lwd = 2,
col=rainbows[1], lwd=2
arrows(xx, brier.rf.s2[[7]][2,], xx, brier.rf.s1[[7]][2,],
length=0.05,
angle=90, code=3, col=rainbows[1])
lines((brier.rf[[7]][3,])"xx, type=", ylim=c(0,0.3),
col=rainbows[2]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][3,], xx, brier.rf.s1[[7]][3,],
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((brier.rf[[7]][4,])"xx, type=", ylim=c(0,0.3),
col=rainbows[3]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][4,], xx, brier.rf.s1[[7]][4,],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((brier.rf[[7]][5,])"xx, type=", ylim=c(0,0.3),
col=rainbows[4]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][5,], xx, brier.rf.s1[[7]][5,],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((brier.rf[[7]][6,])"xx, type=", ylim=c(0,0.3),
col=rainbows[5]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][6,], xx, brier.rf.s1[[7]][6,],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((brier.rf[[7]][7,])"xx, type=", ylim=c(0,0.3),
col=rainbows[6]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][7,], xx, brier.rf.s1[[7]][7,],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((brier.rf[[7]][8,])"xx, type=", ylim=c(0,0.3),
col=rainbows[7]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][8,], xx, brier.rf.s1[[7]][8,],
length=0.05,
angle=90, code=3, col=rainbows[7])
lines((brier.rf[[7]][9,])"xx, type=", ylim=c(0,0.3),
col=rainbows[8]
, lwd=2)
arrows(xx, brier.rf.s2[[7]][9,], xx, brier.rf.s1[[7]][9,],
length=0.05,
\begin{verbatim}
    angle=90, code=3, col=rainbows[8])
    lines((brier.rf[[7]][10,])~xx, type="1", ylim=c(0,0.3),
    col=rainbows[9]

    arrows(xx, brier.rf.s2[[7]][10,], xx, brier.rf.s1[[7]][10,],
    length=0.05,
    angle=90, code=3, col=rainbows[9])
    lines((brier.rf[[7]][11,])~xx, type="1", ylim=c(0,0.3),
    col=rainbows[10]

    arrows(xx, brier.rf.s2[[7]][11,], xx, brier.rf.s1[[7]][11,],
    length=0.05,
    angle=90, code=3, col=rainbows[10])
    lines((brier.rf[[7]][12,])~xx, type="1", ylim=c(0,0.3),
    col=rainbows[11]

    arrows(xx, brier.rf.s2[[7]][12,], xx, brier.rf.s1[[7]][12,],
    length=0.05,
    angle=90, code=3, col=rainbows[11])
    lines((brier.rf[[7]][13,])~xx, type="1", ylim=c(0,0.3),
    col=rainbows[12]

    plot((KL.rf[[1]][1,])~xx, type="1", ylim=c(0,0.8),
    ylab="Kullback-Leibler score", xlab="Time (years since surgery)",
    lwd=2,
    main="1 candidate splitting variable")
    arrows(xx, KL.rf.s2[[1]][1,], xx, KL.rf.s1[[1]][1,],
    length=0.05, angle=90, code=3)
    lines((KL.rf[[1]][2,])~xx, type="1", ylim=c(0,0.8),
    col=rainbows[1], lwd=2)
    arrows(xx, KL.rf.s2[[1]][2,], xx, KL.rf.s1[[1]][2,],
    length=0.05,
    angle=90, code=3, col=rainbows[1])
    lines((KL.rf[[1]][3,])~xx, type="1", ylim=c(0,0.8),
    col=rainbows[2]

    arrows(xx, KL.rf.s2[[1]][3,], xx, KL.rf.s1[[1]][3,],
    length=0.05,
\end{verbatim}
angle = 90, code = 3, col = rainbows[2])

lines((KL.rf[[1]][4,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[3], lwd = 2)

arrows(xx, KL.rf.s2[[1]][4,], xx, KL.rf.s1[[1]][4,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[3])

lines((KL.rf[[1]][5,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[4], lwd = 2)

arrows(xx, KL.rf.s2[[1]][5,], xx, KL.rf.s1[[1]][5,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[4])

lines((KL.rf[[1]][6,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[5], lwd = 2)

arrows(xx, KL.rf.s2[[1]][6,], xx, KL.rf.s1[[1]][6,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[5])

lines((KL.rf[[1]][7,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[6], lwd = 2)

arrows(xx, KL.rf.s2[[1]][7,], xx, KL.rf.s1[[1]][7,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[6])

lines((KL.rf[[1]][8,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[7], lwd = 2)

arrows(xx, KL.rf.s2[[1]][8,], xx, KL.rf.s1[[1]][8,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[7])

lines((KL.rf[[1]][9,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[8], lwd = 2)

arrows(xx, KL.rf.s2[[1]][9,], xx, KL.rf.s1[[1]][9,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[8])

lines((KL.rf[[1]][10,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[9], lwd = 2)

arrows(xx, KL.rf.s2[[1]][10,], xx, KL.rf.s1[[1]][10,],
    length = 0.05,
    angle = 90, code = 3, col = rainbows[9])

lines((KL.rf[[1]][11,]), xx, type = “l”, ylim = c(0, 0.8),
    col = rainbows[10]
arrows(xx, KL.rf.s2[[1]][[1]], xx, KL.rf.s1[[1]][[1]], length=0.05, angle=90, code=3, col=rainbows[10])
lines((KL.rf[[1]][[12]],)~xx, type="l", ylim=c(0,0.8), col=rainbows[11], lwd=2)

arrows(xx, KL.rf.s2[[1]][[12]], xx, KL.rf.s1[[1]][[12]], length=0.05, angle=90, code=3, col=rainbows[11])
lines((KL.rf[[1]][[13]],)~xx, type="l", ylim=c(0,0.8), col=rainbows[12], lwd=2)

plot((KL.rf[[2]][1,]),~xx, type="l", ylab="Kullback-Leibler score", xlab="Time (years since surgery)", col=rainbows[1], lwd=2, main="2 candidate splitting variables")

arrows(xx, KL.rf.s2[[2]][1], xx, KL.rf.s1[[2]][1], length=0.05, angle=90, code=3)
lines((KL.rf[[2]][2,])~xx, type="l", ylim=c(0,0.8), col=rainbows[1], lwd=2)

arrows(xx, KL.rf.s2[[2]][2], xx, KL.rf.s1[[2]][2], length=0.05, angle=90, code=3, col=rainbows[2])
lines((KL.rf[[2]][3,])~xx, type="l", ylim=c(0,0.8), col=rainbows[2], lwd=2)

arrows(xx, KL.rf.s2[[2]][3], xx, KL.rf.s1[[2]][3], length=0.05, angle=90, code=3, col=rainbows[3])
lines((KL.rf[[2]][4,])~xx, type="l", ylim=c(0,0.8), col=rainbows[3], lwd=2)

arrows(xx, KL.rf.s2[[2]][4], xx, KL.rf.s1[[2]][4], length=0.05, angle=90, code=3, col=rainbows[3])
lines((KL.rf[[2]][5,])~xx, type="l", ylim=c(0,0.8), col=rainbows[4], lwd=2)

arrows(xx, KL.rf.s2[[2]][5], xx, KL.rf.s1[[2]][5], length=0.05, angle=90, code=3, col=rainbows[4])
lines((KL.rf[[2]][6,])~xx, type="l", ylim=c(0,0.8), col=rainbows[5], lwd=2)

arrows(xx, KL.rf.s2[[2]][6], xx, KL.rf.s1[[2]][6], length=0.05, angle=90, code=3, col=rainbows[5])
lines((KL.rf[[2]][7,])~xx, type="l", ylim=c(0,0.8), col=rainbows[6]
\begin{verbatim}
833 , lwd=2)
834 arrows(xx, KL.rf.s2[[2]][7,], xx, KL.rf.s1[[2]][7,], length=0.05,
835 angle=90, code=3, col=rainbows[6])
836 lines((KL.rf[[2]][8,])"xx", type=1", ylim=c(0,0.8), col=rainbows[7]
837 , lwd=2)
838 arrows(xx, KL.rf.s2[[2]][8,], xx, KL.rf.s1[[2]][8,], length=0.05,
839 angle=90, code=3, col=rainbows[7])
840 lines((KL.rf[[2]][9,])"xx", type=1", ylim=c(0,0.8), col=rainbows[8]
841 , lwd=2)
842 arrows(xx, KL.rf.s2[[2]][9,], xx, KL.rf.s1[[2]][9,], length=0.05,
843 angle=90, code=3, col=rainbows[8])
844 lines((KL.rf[[2]][10,])"xx", type=1", ylim=c(0,0.8), col=rainbows[9]
845 , lwd=2)
846 arrows(xx, KL.rf.s2[[2]][10,], xx, KL.rf.s1[[2]][10,], length=0.05,
847 angle=90, code=3, col=rainbows[9])
848 lines((KL.rf[[2]][11,])"xx", type=1", ylim=c(0,0.8), col=rainbows[10]
849 , lwd=2)
850 arrows(xx, KL.rf.s2[[2]][11,], xx, KL.rf.s1[[2]][11,], length=0.05,
851 angle=90, code=3, col=rainbows[10])
852 lines((KL.rf[[2]][12,])"xx", type=1", ylim=c(0,0.8), col=rainbows[11]
853 , lwd=2)
854 arrows(xx, KL.rf.s2[[2]][12,], xx, KL.rf.s1[[2]][12,], length=0.05,
855 angle=90, code=3, col=rainbows[11])
856 lines((KL.rf[[2]][13,])"xx", type=1", ylim=c(0,0.8), col=rainbows[12]
857 , lwd=2)
858 arrows(xx, KL.rf.s2[[2]][13,], xx, KL.rf.s1[[2]][13,], length=0.05,
859 angle=90, code=3, col=rainbows[12])
860
861
862 plot((KL.rf[[3]][1,])"xx", type=1", ylim=c(0,0.8),
863 ylab=Kullback–Leibler_score",
864 xlab=Time (years since surgery", lwd=2,
865 main="3 candidate splitting variables")
866 arrows(xx, KL.rf.s2[[3]][1,], xx, KL.rf.s1[[3]][1,],
867 length=0.05, angle=90, code=3)
868 lines((KL.rf[[3]][2,])"xx", type=1", ylim=c(0,0.8),
869 col=rainbows[1], lwd=2)
870 arrows(xx, KL.rf.s2[[3]][2,], xx, KL.rf.s1[[3]][2,], length=0.05,
871 angle=90, code=3, col=rainbows[1])
872 lines((KL.rf[[3]][3,])"xx", type=1", ylim=c(0,0.8), col=rainbows[2]
873 , lwd=2)
874 arrows(xx, KL.rf.s2[[3]][3,], xx, KL.rf.s1[[3]][3,], length=0.05,
875 angle=90, code=3, col=rainbows[2])
876 lines((KL.rf[[3]][4,])"xx", type=1", ylim=c(0,0.8), col=rainbows[3]
877)
307
\end{verbatim}
arrows(xx, KL.rf.s2[[3]][4], xx, KL.rf.s1[[3]][4], length=0.05, angle=90, code=3, col=rainbows[3])
lines((KL.rf[[3]][5]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[4], lwd=2)
arrows(xx, KL.rf.s2[[3]][5], xx, KL.rf.s1[[3]][5], length=0.05, angle=90, code=3, col=rainbows[4])
lines((KL.rf[[3]][6]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[5], lwd=2)
arrows(xx, KL.rf.s2[[3]][6], xx, KL.rf.s1[[3]][6], length=0.05, angle=90, code=3, col=rainbows[5])
lines((KL.rf[[3]][7]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[6], lwd=2)
arrows(xx, KL.rf.s2[[3]][7], xx, KL.rf.s1[[3]][7], length=0.05, angle=90, code=3, col=rainbows[6])
lines((KL.rf[[3]][8]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[7], lwd=2)
arrows(xx, KL.rf.s2[[3]][8], xx, KL.rf.s1[[3]][8], length=0.05, angle=90, code=3, col=rainbows[7])
lines((KL.rf[[3]][9]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[8], lwd=2)
arrows(xx, KL.rf.s2[[3]][9], xx, KL.rf.s1[[3]][9], length=0.05, angle=90, code=3, col=rainbows[8])
lines((KL.rf[[3]][10]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[9], lwd=2)
arrows(xx, KL.rf.s2[[3]][10], xx, KL.rf.s1[[3]][10], length=0.05, angle=90, code=3, col=rainbows[9])
lines((KL.rf[[3]][11]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[10], lwd=2)
arrows(xx, KL.rf.s2[[3]][11], xx, KL.rf.s1[[3]][11], length=0.05, angle=90, code=3, col=rainbows[10])
lines((KL.rf[[3]][12]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[11], lwd=2)
arrows(xx, KL.rf.s2[[3]][12], xx, KL.rf.s1[[3]][12], length=0.05, angle=90, code=3, col=rainbows[11])
lines((KL.rf[[3]][13]) ~xx, type="l", ylim=c(0,0.8), col=rainbows[12], lwd=2)
arrows(xx, KL.rf.s2[[3]][13], xx, KL.rf.s1[[3]][13], length=0.05, angle=90, code=3, col=rainbows[12])
plot((KL.rf[[4]][1]) ~xx, type="l", ylim=c(0,0.8), ylab="Kullback-Leibler score", xlab="Time \text{(years since surgery)}")
arrows(xx, KL.rf.s2[[4]][12,], xx, KL.rf.s1[[4]][12,], length=0.05, angle=90, code=3, col=rainbows[11])
lines((KL.rf[[4]][13,])~xx, type="l", ylim=c(0,0.8), col=rainbows[12], lwd=2)
arrows(xx, KL.rf.s2[[4]][13,], xx, KL.rf.s1[[4]][13,], length=0.05, angle=90, code=3, col=rainbows[11])

plot((KL.rf[[5]][1,])~xx, type="l", ylim=c(0,0.8), ylab="Kullback-Leibler score", xlab="Time (years since surgery)", lwd=2, main="5 candidate splitting variables")
arrows(xx, KL.rf.s2[[5]][1,], xx, KL.rf.s1[[5]][1,], length=0.05, angle=90, code=3, col=rainbows[1])
lines((KL.rf[[5]][2,])~xx, type="l", ylim=c(0,0.8), col=rainbows[1], lwd=2)
arrows(xx, KL.rf.s2[[5]][2,], xx, KL.rf.s1[[5]][2,], length=0.05, angle=90, code=3, col=rainbows[2])
lines((KL.rf[[5]][3,])~xx, type="l", ylim=c(0,0.8), col=rainbows[2], lwd=2)
arrows(xx, KL.rf.s2[[5]][3,], xx, KL.rf.s1[[5]][3,], length=0.05, angle=90, code=3, col=rainbows[3])
lines((KL.rf[[5]][4,])~xx, type="l", ylim=c(0,0.8), col=rainbows[3], lwd=2)
arrows(xx, KL.rf.s2[[5]][4,], xx, KL.rf.s1[[5]][4,], length=0.05, angle=90, code=3, col=rainbows[4])
lines((KL.rf[[5]][5,])~xx, type="l", ylim=c(0,0.8), col=rainbows[4], lwd=2)
arrows(xx, KL.rf.s2[[5]][5,], xx, KL.rf.s1[[5]][5,], length=0.05, angle=90, code=3, col=rainbows[5])
lines((KL.rf[[5]][6,])~xx, type="l", ylim=c(0,0.8), col=rainbows[5], lwd=2)
arrows(xx, KL.rf.s2[[5]][6,], xx, KL.rf.s1[[5]][6,], length=0.05, angle=90, code=3, col=rainbows[6])
lines((KL.rf[[5]][7,])~xx, type="l", ylim=c(0,0.8), col=rainbows[6], lwd=2)
arrows(xx, KL.rf.s2[[5]][7,], xx, KL.rf.s1[[5]][7,], length=0.05, angle=90, code=3, col=rainbows[7])
lines((KL.rf[[5]][8,])~xx, type="l", ylim=c(0,0.8), col=rainbows[7], lwd=2)
arrows(xx, KL.rf.s2[[5]][8,], xx, KL.rf.s1[[5]][8,], length=0.05, angle=90, code=3, col=rainbows[7])
lines((KL.rf[[5]][9,])~xx, type="l", ylim=c(0,0.8), col=rainbows[8], lwd=2)
arrows(xx, KL.rf.s2[[5]][9], xx, KL.rf.s1[[5]][9], length=0.05, angle=90, code=3, col=rainbows[8])
lines((KL.rf[[5]][10,])~xx, type="l", ylim=c(0,0.8), col=rainbows[9], lwd=2)
arrows(xx, KL.rf.s2[[5]][10], xx, KL.rf.s1[[5]][10], length=0.05, angle=90, code=3, col=rainbows[9])
lines((KL.rf[[5]][11,])~xx, type="l", ylim=c(0,0.8), col=rainbows[10], lwd=2)
arrows(xx, KL.rf.s2[[5]][11], xx, KL.rf.s1[[5]][11], length=0.05, angle=90, code=3, col=rainbows[10])
lines((KL.rf[[5]][12,])~xx, type="l", ylim=c(0,0.8), col=rainbows[11], lwd=2)
arrows(xx, KL.rf.s2[[5]][12], xx, KL.rf.s1[[5]][12], length=0.05, angle=90, code=3, col=rainbows[11])
lines((KL.rf[[5]][13,])~xx, type="l", ylim=c(0,0.8), col=rainbows[12], lwd=2)
arrows(xx, KL.rf.s2[[5]][13], xx, KL.rf.s1[[5]][13], length=0.05, angle=90, code=3, col=rainbows[12])
plot((KL.rf[[6]][1,])~xx, type="l", ylim=c(0,0.8), ylab="Kullback-Leibler score", xlab="Time (years since surgery)", lwd=2, main="6 candidate splitting variables")
arrows(xx, KL.rf.s2[[6]][1], xx, KL.rf.s1[[6]][1], length=0.05, angle=90, code=3, col=rainbows[1])
lines((KL.rf[[6]][2,])~xx, type="l", ylim=c(0,0.8), col=rainbows[1], lwd=2)
arrows(xx, KL.rf.s2[[6]][2], xx, KL.rf.s1[[6]][2], length=0.05, angle=90, code=3, col=rainbows[1])
lines((KL.rf[[6]][3,])~xx, type="l", ylim=c(0,0.8), col=rainbows[2], lwd=2)
arrows(xx, KL.rf.s2[[6]][3], xx, KL.rf.s1[[6]][3], length=0.05, angle=90, code=3, col=rainbows[2])
lines((KL.rf[[6]][4,])~xx, type="l", ylim=c(0,0.8), col=rainbows[3], lwd=2)
arrows(xx, KL.rf.s2[[6]][4], xx, KL.rf.s1[[6]][4], length=0.05, angle=90, code=3, col=rainbows[3])
lines((KL.rf[[6]][5,])~xx, type="l", ylim=c(0,0.8), col=rainbows[4], lwd=2)
arrows(xx, KL.rf.s2[[6]][5], xx, KL.rf.s1[[6]][5], length=0.05, angle=90, code=3, col=rainbows[4])
lines((KL.rf[[6]][6,])~xx, type="l", ylim=c(0,0.8), col=rainbows[5], lwd=2)
\begin{verbatim}
1058  , lwd=2)
1059  arrows(xx, KL.rf.s2[[6]][6], xx, KL.rf.s1[[6]][6], length=0.05,  
1060     angle=90, code=3, col=rainbows[5])
1061  lines((KL.rf[[6]][7]), xx, type="l", ylim=c(0,0.8), col=rainbows[6]  
1062     , lwd=2)
1063  arrows(xx, KL.rf.s2[[6]][7], xx, KL.rf.s1[[6]][7], length=0.05,  
1064     angle=90, code=3, col=rainbows[6])
1065  lines((KL.rf[[6]][8]), xx, type="l", ylim=c(0,0.8), col=rainbows[7]  
1066     , lwd=2)
1067  arrows(xx, KL.rf.s2[[6]][8], xx, KL.rf.s1[[6]][8], length=0.05,  
1068     angle=90, code=3, col=rainbows[7])
1069  lines((KL.rf[[6]][9]), xx, type="l", ylim=c(0,0.8), col=rainbows[8]  
1070     , lwd=2)
1071  arrows(xx, KL.rf.s2[[6]][9], xx, KL.rf.s1[[6]][9], length=0.05,  
1072     angle=90, code=3, col=rainbows[8])
1073  lines((KL.rf[[6]][10]), xx, type="l", ylim=c(0,0.8), col=rainbows[9]  
1074     , lwd=2)
1075  arrows(xx, KL.rf.s2[[6]][10], xx, KL.rf.s1[[6]][10], length=0.05,  
1076     angle=90, code=3, col=rainbows[9])
1077  lines((KL.rf[[6]][11]), xx, type="l", ylim=c(0,0.8), col=rainbows[10]  
1078     , lwd=2)
1079  arrows(xx, KL.rf.s2[[6]][11], xx, KL.rf.s1[[6]][11], length=0.05,  
1080     angle=90, code=3, col=rainbows[10])
1081  lines((KL.rf[[6]][12]), xx, type="l", ylim=c(0,0.8), col=rainbows[11]  
1082     , lwd=2)
1083  arrows(xx, KL.rf.s2[[6]][12], xx, KL.rf.s1[[6]][12], length=0.05,  
1084     angle=90, code=3, col=rainbows[11])
1085  lines((KL.rf[[6]][13]), xx, type="l", ylim=c(0,0.8), col=rainbows[12]  
1086     , lwd=2)
1087  arrows(xx, KL.rf.s2[[6]][13], xx, KL.rf.s1[[6]][13], length=0.05,  
1088     angle=90, code=3, col=rainbows[12])
1089
1090  plot((KL.rf[[7]][1]), xx, type="l", ylim=c(0,0.8),  
1091      ylab="Kullback-Leibler score",  
1092      xlab="Time (years since surgery)", lwd=2,  
1093      main="7 candidate splitting variables")
1094  arrows(xx, KL.rf.s2[[7]][1], xx, KL.rf.s1[[7]][1],  
1095      length=0.05, angle=90, code=3)
1096  lines((KL.rf[[7]][2]), xx, type="l", ylim=c(0,0.8),  
1097      col=rainbows[1], lwd=2)
1098  arrows(xx, KL.rf.s2[[7]][2], xx, KL.rf.s1[[7]][2], length=0.05,  
1099      angle=90, code=3, col=rainbows[1])
1100  lines((KL.rf[[7]][3]), xx, type="l", ylim=c(0,0.8), col=rainbows[2]  
1101     , lwd=2)
312
\end{verbatim}
D.3.3 Cross-validation for number of splitting points and node depth

```r
out.r.2.0 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[1], nodedepths=nodedepths, option="second")
means.r.2.0 <- out.r.2.0[1]
sds.r.2.0 <- out.r.2.0[2]

out.r.2.1 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[2], nodedepths=nodedepths, option="second")
means.r.2.1 <- out.r.2.1[1]
sds.r.2.1 <- out.r.2.1[2]

out.r.2.2 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[3], nodedepths=nodedepths, option="second")
means.r.2.2 <- out.r.2.2[1]
sds.r.2.2 <- out.r.2.2[2]

out.r.2.3 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[4], nodedepths=nodedepths, option="second")
means.r.2.3 <- out.r.2.3[1]
sds.r.2.3 <- out.r.2.3[2]

out.r.2.4 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[5], nodedepths=nodedepths, option="second")
means.r.2.4 <- out.r.2.4[1]
sds.r.2.4 <- out.r.2.4[2]

out.r.2.5 <- cross.val.plots.rand(mtrys=2, nodesizes=200,
    nsplits=splits[6], nodedepths=nodedepths, option="second")
means.r.2.5 <- out.r.2.5[1]
sds.r.2.5 <- out.r.2.5[2]

vall <- cbind(means.r.2.0[1][1], means.r.2.1[1][1],
    means.r.2.2[1][1], means.r.2.3[1][1],
    means.r.2.4[1][1], means.r.2.5[1][1])
vall.sd <- cbind(sds.r.2.0[1][1], sds.r.2.1[1][1],
    sds.r.2.2[1][1], sds.r.2.3[1][1],
    sds.r.2.4[1][1], sds.r.2.5[1][1])
vall1 <- vall + vall.sd
vall2 <- vall - vall.sd

plot(vall[,1]~nodedepths,pch=16, ylab="C-index")
```

314
```r
xlab="Node_depth",
main="0_random_split_points", ylim=c(0.45, 0.75)
arrows(nodedepths, vall2[,1], nodedepths, vall1[,1],
       length=0.05, angle=90, code=3)

plot(vall[,2]~nodedepths, pch=16, ylab="C-index",
xlab="Node_depth",
main="1_random_split_point", ylim=c(0.45, 0.75))
arrows(nodedepths, vall2[,2], nodedepths, vall1[,2],
       length=0.05, angle=90, code=3)

plot(vall[,3]~nodedepths, pch=16, ylab="C-index",
xlab="Node_depth",
main="2_random_split_points", ylim=c(0.45, 0.75))
arrows(nodedepths, vall2[,3], nodedepths, vall1[,3],
       length=0.05, angle=90, code=3)

plot(vall[,4]~nodedepths, pch=16, ylab="C-index",
xlab="Node_depth",
main="3_random_split_points", ylim=c(0.45, 0.75))
arrows(nodedepths, vall2[,4], nodedepths, vall1[,4],
       length=0.05, angle=90, code=3)

plot(vall[,5]~nodedepths, pch=16, ylab="C-index",
xlab="Node_depth",
main="4_random_split_points", ylim=c(0.45, 0.75))
arrows(nodedepths, vall2[,5], nodedepths, vall1[,5],
       length=0.05, angle=90, code=3)

plot(vall[,6]~nodedepths, pch=16, ylab="C-index",
xlab="Node_depth",
main="5_random_split_points", ylim=c(0.45, 0.75))
arrows(nodedepths, vall2[,6], nodedepths, vall1[,6],
       length=0.05, angle=90, code=3)

brier.rf2 <- list()
brier.rf2[[1]] <- means.r.2.0[[1]][[2]]
brier.rf2[[2]] <- means.r.2.1[[1]][[2]]
brier.rf2[[3]] <- means.r.2.2[[1]][[2]]
brier.rf2[[4]] <- means.r.2.3[[1]][[2]]
brier.rf2[[5]] <- means.r.2.4[[1]][[2]]
brier.rf2[[6]] <- means.r.2.5[[1]][[2]]

# variations 3rd, 4th decimal, after that exact.
```
```r
table.b.0 <- brier.rf2[[1]]
table.b.2 <- brier.rf2[[3]]
rownames(table.b.0) <- paste("N.D.", nodedepths, sep=" ")
rownames(table.b.2) <- paste("N.D.", nodedepths, sep=" ")
colnames(table.b.0) <- paste(seq(1,6), " years", sep=" ")
colnames(table.b.2) <- paste(seq(1,6), " years", sep=" ")
library(xtable)
xtable(table.b.2, digits=8)

brier.rf2.s1 <- list()
brier.rf2.s1[[1]] <- means.r.2.0[[1]][[2]] + sds.r.2.0[[1]][[2]]
brier.rf2.s1[[2]] <- means.r.2.1[[1]][[2]] + sds.r.2.1[[1]][[2]]
brier.rf2.s1[[3]] <- means.r.2.2[[1]][[2]] + sds.r.2.2[[1]][[2]]
brier.rf2.s1[[4]] <- means.r.2.3[[1]][[2]] + sds.r.2.3[[1]][[2]]
brier.rf2.s1[[5]] <- means.r.2.4[[1]][[2]] + sds.r.2.4[[1]][[2]]
brier.rf2.s1[[6]] <- means.r.2.5[[1]][[2]] + sds.r.2.5[[1]][[2]]

brier.rf2.s2 <- list()
brier.rf2.s2[[1]] <- means.r.2.0[[1]][[2]] - sds.r.2.0[[1]][[2]]
brier.rf2.s2[[2]] <- means.r.2.1[[1]][[2]] - sds.r.2.1[[1]][[2]]
brier.rf2.s2[[3]] <- means.r.2.2[[1]][[2]] - sds.r.2.2[[1]][[2]]
brier.rf2.s2[[4]] <- means.r.2.3[[1]][[2]] - sds.r.2.3[[1]][[2]]
brier.rf2.s2[[5]] <- means.r.2.4[[1]][[2]] - sds.r.2.4[[1]][[2]]
brier.rf2.s2[[6]] <- means.r.2.5[[1]][[2]] - sds.r.2.5[[1]][[2]]

KL.rf2 <- list()
KL.rf2[[1]] <- means.r.2.0[[1]][[3]]
KL.rf2[[2]] <- means.r.2.1[[1]][[3]]
KL.rf2[[3]] <- means.r.2.2[[1]][[3]]
KL.rf2[[4]] <- means.r.2.3[[1]][[3]]
KL.rf2[[5]] <- means.r.2.4[[1]][[3]]
KL.rf2[[6]] <- means.r.2.5[[1]][[3]]

KL.rf2.s1 <- list()
KL.rf2.s1[[1]] <- means.r.2.0[[1]][[3]] + sds.r.2.0[[1]][[3]]
KL.rf2.s1[[2]] <- means.r.2.1[[1]][[3]] + sds.r.2.1[[1]][[3]]
KL.rf2.s1[[3]] <- means.r.2.2[[1]][[3]] + sds.r.2.2[[1]][[3]]
KL.rf2.s1[[4]] <- means.r.2.3[[1]][[3]] + sds.r.2.3[[1]][[3]]
KL.rf2.s1[[5]] <- means.r.2.4[[1]][[3]] + sds.r.2.4[[1]][[3]]
KL.rf2.s1[[6]] <- means.r.2.5[[1]][[3]] + sds.r.2.5[[1]][[3]]
```
```
KL.rf2.s2 <- list()
KL.rf2.s2[[1]] <- means.r.2.0[[1]][[3]] - sds.r.2.0[[1]][[3]]
KL.rf2.s2[[2]] <- means.r.2.1[[1]][[3]] - sds.r.2.1[[1]][[3]]
KL.rf2.s2[[3]] <- means.r.2.2[[1]][[3]] - sds.r.2.2[[1]][[3]]
KL.rf2.s2[[4]] <- means.r.2.3[[1]][[3]] - sds.r.2.3[[1]][[3]]
KL.rf2.s2[[5]] <- means.r.2.4[[1]][[3]] - sds.r.2.4[[1]][[3]]
KL.rf2.s2[[6]] <- means.r.2.5[[1]][[3]] - sds.r.2.5[[1]][[3]]

legend("bottomleft", legend=nodedepths, col=c("black", rainbows[1:7]), lwd=10, ncol=3)
xx <- seq(1,6)
plot((brier.rf2[[1]][1,])~xx, type="l", ylim=c(0,0.3), ylab="Brier score", xlab="Time (years since surgery)", lwd=2, main=0 random split points")
arrows(xx, brier.rf2.s2[[1]][1,], xx, brier.rf2.s1[[1]][1,], length=0.05, angle=90, code=3)
lines((brier.rf2[[1]][2,])~xx, type="l", ylim=c(0,0.3), col=rainbows[1], lwd=2)
arrows(xx, brier.rf2.s2[[1]][2,], xx, brier.rf2.s1[[1]][2,], length=0.05, angle=90, code=3, col=rainbows[1])
lines((brier.rf2[[1]][3,])~xx, type="l", ylim=c(0,0.3), col=rainbows[2], lwd=2)
arrows(xx, brier.rf2.s2[[1]][3,], xx, brier.rf2.s1[[1]][3,], length=0.05, angle=90, code=3, col=rainbows[2])
lines((brier.rf2[[1]][4,])~xx, type="l", ylim=c(0,0.3), col=rainbows[3], lwd=2)
arrows(xx, brier.rf2.s2[[1]][4,], xx, brier.rf2.s1[[1]][4,], length=0.05, angle=90, code=3, col=rainbows[3])
lines((brier.rf2[[1]][5,])~xx, type="l", ylim=c(0,0.3), col=rainbows[4], lwd=2)
arrows(xx, brier.rf2.s2[[1]][5,], xx, brier.rf2.s1[[1]][5,], length=0.05, angle=90, code=3, col=rainbows[4])
lines((brier.rf2[[1]][6,])~xx, type="l", ylim=c(0,0.3),
```

317
177  col=rainbows[5]
178  , lwd=2)
179 arrows(xx, brier.rf2.s2[[1]][6,], xx, brier.rf2.s1[[1]][6,],
180 length=0.05,
181  angle=90, code=3, col=rainbows[5])
182 lines((brier.rf2[[1]][7,])~xx, type="l", ylim=c(0,0.3),
183 col=rainbows[6]
184 , lwd=2)
185 arrows(xx, brier.rf2.s2[[1]][7,], xx, brier.rf2.s1[[1]][7,],
186 length=0.05,
187  angle=90, code=3, col=rainbows[6])
188 lines((brier.rf2[[1]][8,])~xx, type="l", ylim=c(0,0.3),
189 col=rainbows[7]
190 , lwd=2)
191 arrows(xx, brier.rf2.s2[[1]][8,], xx, brier.rf2.s1[[1]][8,],
192 length=0.05,
193  angle=90, code=3, col=rainbows[7])
194
195
196 plot((brier.rf2[[2]][1,])~xx, type="l", ylim=c(0,0.3),
197 ylab="Brier score", xlab="Time (years since surgery)", lwd=2,
198 main="1 random split point")
199 arrows(xx, brier.rf2.s2[[2]][1,], xx, brier.rf2.s1[[2]][1,],
200 length=0.05, angle=90, code=3)
201 lines((brier.rf2[[2]][2,])~xx, type="l", ylim=c(0,0.3),
202 col=rainbows[1], lwd=2)
203 arrows(xx, brier.rf2.s2[[2]][2,], xx, brier.rf2.s1[[2]][2,],
204 length=0.05,
205  angle=90, code=3, col=rainbows[1])
206 lines((brier.rf2[[2]][3,])~xx, type="l", ylim=c(0,0.3),
207 col=rainbows[2]
208 , lwd=2)
209 arrows(xx, brier.rf2.s2[[2]][3,], xx, brier.rf2.s1[[2]][3,],
210 length=0.05,
211  angle=90, code=3, col=rainbows[2])
212 lines((brier.rf2[[2]][4,])~xx, type="l", ylim=c(0,0.3),
213 col=rainbows[3]
214 , lwd=2)
215 arrows(xx, brier.rf2.s2[[2]][4,], xx, brier.rf2.s1[[2]][4,],
216 length=0.05,
217  angle=90, code=3, col=rainbows[3])
218 lines((brier.rf2[[2]][5,])~xx, type="l", ylim=c(0,0.3),
219 col=rainbows[4]
220 , lwd=2)
221
318
arrows(xx, brier.rf2.s2[[2]][5,], xx, brier.rf2.s1[[2]][5,], length=0.05, angle=90, code=3, col=rainbows[4])
lines((brier.rf2[[2]][6,])~xx, type="l", ylim=c(0,0.3), col=rainbows[5])

arrows(xx, brier.rf2.s2[[2]][6,], xx, brier.rf2.s1[[2]][6,], length=0.05, angle=90, code=3, col=rainbows[5])
lines((brier.rf2[[2]][7,])~xx, type="l", ylim=c(0,0.3), col=rainbows[6])

arrows(xx, brier.rf2.s2[[2]][7,], xx, brier.rf2.s1[[2]][7,], length=0.05, angle=90, code=3, col=rainbows[6])
lines((brier.rf2[[2]][8,])~xx, type="l", ylim=c(0,0.3), col=rainbows[7])

plot((brier.rf2[[3]][1,])~xx, type="l", ylim=c(0,0.3), xlab="Time (years since surgery)", lwd=2, main="2 random split points")
arrows(xx, brier.rf2.s2[[3]][1,], xx, brier.rf2.s1[[3]][1,], length=0.05, angle=90, code=3)
lines((brier.rf2[[3]][2,])~xx, type="l", ylim=c(0,0.3), col=rainbows[1], lwd=2)
arrows(xx, brier.rf2.s2[[3]][2,], xx, brier.rf2.s1[[3]][2,], length=0.05, angle=90, code=3, col=rainbows[1])
lines((brier.rf2[[3]][3,])~xx, type="l", ylim=c(0,0.3), col=rainbows[2])

arrows(xx, brier.rf2.s2[[3]][3,], xx, brier.rf2.s1[[3]][3,], length=0.05, angle=90, code=3, col=rainbows[2])
lines((brier.rf2[[3]][4,])~xx, type="l", ylim=c(0,0.3), col=rainbows[3])

arrows(xx, brier.rf2.s2[[3]][4,], xx, brier.rf2.s1[[3]][4,], length=0.05)
\begin{verbatim}
angle = 90, code = 3, col = rainbows[3]
lines((brier.rf2[[3]][5,])~xx, type = "1", ylim = c(0, 0.3),
col = rainbows[4], lwd = 2)

arrows(xx, brier.rf2.s2[[3]][5,], xx, brier.rf2.s1[[3]][5,],
length = 0.05,
angle = 90, code = 3, col = rainbows[4])
lines((brier.rf2[[3]][6,])~xx, type = "1", ylim = c(0, 0.3),
col = rainbows[5],
length = 0.05,
angle = 90, code = 3, col = rainbows[5])

arrows(xx, brier.rf2.s2[[3]][6,], xx, brier.rf2.s1[[3]][6,],
length = 0.05,
angle = 90, code = 3, col = rainbows[6])

lines((brier.rf2[[3]][7,])~xx, type = "1", ylim = c(0, 0.3),
col = rainbows[7],
length = 0.05,
angle = 90, code = 3, col = rainbows[7])

plot((brier.rf2[[4]][1,])~xx, type = "1", ylim = c(0, 0.3),

ylab = "Brier_score",
xlab = "Time(years since surgery)", lwd = 2,
main = "3 random split points")
arrows(xx, brier.rf2.s2[[4]][1,], xx, brier.rf2.s1[[4]][1,],
length = 0.05, angle = 90, code = 3)
lines((brier.rf2[[4]][2,])~xx, type = "1", ylim = c(0, 0.3),
col = rainbows[1], lwd = 2)
arrows(xx, brier.rf2.s2[[4]][2,], xx, brier.rf2.s1[[4]][2,],
length = 0.05,
angle = 90, code = 3, col = rainbows[1])

lines((brier.rf2[[4]][3,])~xx, type = "1", ylim = c(0, 0.3),
col = rainbows[2],
length = 0.05,
angle = 90, code = 3, col = rainbows[2])
lines((brier.rf2[[4]][4,])~xx, type = "1", ylim = c(0, 0.3),

}
\end{verbatim}
col=rainbows[3]

arrows(xx, brier.rf2.s2[[4]][4], xx, brier.rf2.s1[[4]][4],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((brier.rf2[[4]][5,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[4]

arrows(xx, brier.rf2.s2[[4]][5], xx, brier.rf2.s1[[4]][5],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((brier.rf2[[4]][6,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[5]

arrows(xx, brier.rf2.s2[[4]][6], xx, brier.rf2.s1[[4]][6],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((brier.rf2[[4]][7,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[6]

arrows(xx, brier.rf2.s2[[4]][7], xx, brier.rf2.s1[[4]][7],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((brier.rf2[[4]][8,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[7]

arrows(xx, brier.rf2.s2[[4]][8], xx, brier.rf2.s1[[4]][8],
length=0.05,
angle=90, code=3, col=rainbows[7])

plot((brier.rf2[[5]][1,])~xx, type="l", ylim=c(0,0.3),
ylab="Brier score",
xlab="Time (years since surgery)", lwd=2,
main="4 random split points")
arrows(xx, brier.rf2.s2[[5]][1], xx, brier.rf2.s1[[5]][1],
length=0.05, angle=90, code=3)
lines((brier.rf2[[5]][2,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[1], lwd=2)
arrows(xx, brier.rf2.s2[[5]][2], xx, brier.rf2.s1[[5]][2],
length=0.05,
angle=90, code=3, col=rainbows[1])
lines((brier.rf2[[5]][3,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[2]

, lwd=2)
arrows(xx, brier.rf2.s2[[5]][3], xx, brier.rf2.s1[[5]][3],
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((brier.rf2[[5]][4,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[3]
, lwd=2)
arrows(xx, brier.rf2.s2[[5]][4], xx, brier.rf2.s1[[5]][4],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((brier.rf2[[5]][5,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[4]
, lwd=2)
arrows(xx, brier.rf2.s2[[5]][5], xx, brier.rf2.s1[[5]][5],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((brier.rf2[[5]][6,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[5]
, lwd=2)
arrows(xx, brier.rf2.s2[[5]][6], xx, brier.rf2.s1[[5]][6],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((brier.rf2[[5]][7,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[6]
, lwd=2)
arrows(xx, brier.rf2.s2[[5]][7], xx, brier.rf2.s1[[5]][7],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((brier.rf2[[5]][8,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[7]
, lwd=2)
arrows(xx, brier.rf2.s2[[5]][8], xx, brier.rf2.s1[[5]][8],
length=0.05,
angle=90, code=3, col=rainbows[7])

plot((brier.rf2[[6]][1,])~xx, type="l", ylim=c(0,0.3),
ylab="Brier score",
xlab="Time(years since surgery)", lwd=2,
main="5 random split points")
arrows(xx, brier.rf2.s2[[6]][1], xx, brier.rf2.s1[[6]][1],
length=0.05, angle=90, code=3)
lines((brier.rf2[[6]][2,])~xx, type="l", ylim=c(0,0.3),
col=rainbows[1], lwd=2)
arrows(xx, brier.rf2.s2[[6]][2], xx, brier.rf2.s1[[6]][2],
length=0.05,
```r
402 angle = 90, code = 3, col = rainbows[1]
403 lines((brier.rf2[[6]][3,]) ~ xx, type = "l", ylim = c(0, 0.3),
404 col = rainbows[2]
405 , lwd = 2)
406 arrows(xx, brier.rf2.s2[[6]][3,], xx, brier.rf2.s1[[6]][3,],
407 length = 0.05,
408 angle = 90, code = 3, col = rainbows[2])
409 lines((brier.rf2[[6]][4,]) ~ xx, type = "l", ylim = c(0, 0.3),
410 col = rainbows[3]
411 , lwd = 2)
412 arrows(xx, brier.rf2.s2[[6]][4,], xx, brier.rf2.s1[[6]][4,],
413 length = 0.05,
414 angle = 90, code = 3, col = rainbows[3])
415 lines((brier.rf2[[6]][5,]) ~ xx, type = "l", ylim = c(0, 0.3),
416 col = rainbows[4]
417 , lwd = 2)
418 arrows(xx, brier.rf2.s2[[6]][5,], xx, brier.rf2.s1[[6]][5,],
419 length = 0.05,
420 angle = 90, code = 3, col = rainbows[4])
421 lines((brier.rf2[[6]][6,]) ~ xx, type = "l", ylim = c(0, 0.3),
422 col = rainbows[5]
423 , lwd = 2)
424 arrows(xx, brier.rf2.s2[[6]][6,], xx, brier.rf2.s1[[6]][6,],
425 length = 0.05,
426 angle = 90, code = 3, col = rainbows[5])
427 lines((brier.rf2[[6]][7,]) ~ xx, type = "l", ylim = c(0, 0.3),
428 col = rainbows[6]
429 , lwd = 2)
430 arrows(xx, brier.rf2.s2[[6]][7,], xx, brier.rf2.s1[[6]][7,],
431 length = 0.05,
432 angle = 90, code = 3, col = rainbows[6])
433 lines((brier.rf2[[6]][8,]) ~ xx, type = "l", ylim = c(0, 0.3),
434 col = rainbows[7]
435 , lwd = 2)
436 arrows(xx, brier.rf2.s2[[6]][8,], xx, brier.rf2.s1[[6]][8,],
437 length = 0.05,
438 angle = 90, code = 3, col = rainbows[7])
439
440
441 plot((KL.rf2[[1]][1,]) ~ xx, type = "l", ylim = c(0, 0.8),
442 ylab = "Kullback-Leibler score",
443 xlab = "Time (years since surgery)", lwd = 2,
444 main = "0_random_split_points")
445 arrows(xx, KL.rf2.s2[[1]][1,], xx, KL.rf2.s1[[1]][1,],
446
323
```
plot((KL.rf2[[2]][1,])~xx, type="l", ylim=c(0,0.8),
ylab="Kullback–Leibler score",
xlab="Time (years since surgery)",
lwd=2,
main="1 random split point")
arrows(xx, KL.rf2.s2[[2]][1,], xx, KL.rf2.s1[[2]][1,],
  length=0.05, angle=90, code=3)
lines((KL.rf2[[2]][2,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[1], lwd=2)
arrows(xx, KL.rf2.s2[[2]][2,], xx, KL.rf2.s1[[2]][2,],
  length=0.05,
  angle=90, code=3, col=rainbows[1])
lines((KL.rf2[[2]][3,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[2]
  , lwd=2)
arrows(xx, KL.rf2.s2[[2]][3,], xx, KL.rf2.s1[[2]][3,],
  length=0.05,
  angle=90, code=3, col=rainbows[2])
lines((KL.rf2[[2]][4,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[3]
  , lwd=2)
arrows(xx, KL.rf2.s2[[2]][4,], xx, KL.rf2.s1[[2]][4,],
  length=0.05,
  angle=90, code=3, col=rainbows[3])
lines((KL.rf2[[2]][5,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[4]
  , lwd=2)
arrows(xx, KL.rf2.s2[[2]][5,], xx, KL.rf2.s1[[2]][5,],
  length=0.05,
  angle=90, code=3, col=rainbows[4])
lines((KL.rf2[[2]][6,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[5]
  , lwd=2)
arrows(xx, KL.rf2.s2[[2]][6,], xx, KL.rf2.s1[[2]][6,],
  length=0.05,
  angle=90, code=3, col=rainbows[5])
lines((KL.rf2[[2]][7,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[6]
  , lwd=2)
arrows(xx, KL.rf2.s2[[2]][7,], xx, KL.rf2.s1[[2]][7,],
  length=0.05,
  angle=90, code=3, col=rainbows[6])
lines((KL.rf2[[2]][8,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[7]
arrows(xx, KL.rf2.s2[[2]][8], xx, KL.rf2.s1[[2]][8],
        length=0.05,
        angle=90, code=3, col=rainbows[7])

plot((KL.rf2[[3]][1,])~xx, type="l", ylim=c(0,0.8),
      ylab="Kullback-Leibler score",
      xlab="Time (years since surgery)", lwd=2,
      main="2 random split points"
lines((KL.rf2[[3]][1]),xx, KL.rf2.s1[[3]][1],
      length=0.05, angle=90, code=3)
lines((KL.rf2[[3]][2]),xx, KL.rf2.s1[[3]][2],
      length=0.05,
      angle=90, code=3, col=rainbows[1])
lines((KL.rf2[[3]][3]),xx, type="l", ylim=c(0,0.8),
      col=rainbows[2],
      length=0.05,
      angle=90, code=3, col=rainbows[2])
lines((KL.rf2[[3]][4]),xx, type="l", ylim=c(0,0.8),
      col=rainbows[3],
      length=0.05,
      angle=90, code=3, col=rainbows[3])
lines((KL.rf2[[3]][5]),xx, type="l", ylim=c(0,0.8),
      col=rainbows[4],
      length=0.05,
      angle=90, code=3, col=rainbows[4])
lines((KL.rf2[[3]][6]),xx, type="l", ylim=c(0,0.8),
      col=rainbows[5],
      length=0.05,
      angle=90, code=3, col=rainbows[5])
lines((KL.rf2[[3]][7]),xx, type="l", ylim=c(0,0.8),
      col=rainbows[6],
      length=0.05,
      angle=90, code=3, col=rainbows[6])
arrows(xx, KL.rf2.s2[[3]][7], xx, KL.rf2.s1[[3]][7],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((KL.rf2[[3]][8,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[7], lwd=2)
arrows(xx, KL.rf2.s2[[3]][8], xx, KL.rf2.s1[[3]][8],
length=0.05,
angle=90, code=3, col=rainbows[7])

plot((KL.rf2[[4]][1,])~xx, type="l", ylim=c(0,0.8),
ylab="Kullback–Leibler score",
xlab="Time (years since surgery)", lwd=2,
main="3 random split points")
arrows(xx, KL.rf2.s2[[4]][1], xx, KL.rf2.s1[[4]][1],
length=0.05, angle=90, code=3)
lines((KL.rf2[[4]][2,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[1], lwd=2)
arrows(xx, KL.rf2.s2[[4]][2], xx, KL.rf2.s1[[4]][2],
length=0.05,
angle=90, code=3, col=rainbows[1])
lines((KL.rf2[[4]][3,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[2], lwd=2)
arrows(xx, KL.rf2.s2[[4]][3], xx, KL.rf2.s1[[4]][3],
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((KL.rf2[[4]][4,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[3], lwd=2)
arrows(xx, KL.rf2.s2[[4]][4], xx, KL.rf2.s1[[4]][4],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((KL.rf2[[4]][5,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[4], lwd=2)
arrows(xx, KL.rf2.s2[[4]][5], xx, KL.rf2.s1[[4]][5],
length=0.05,
angle=90, code=3, col=rainbows[4])
lines((KL.rf2[[4]][6,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[5], lwd=2)
arrows(xx, KL.rf2.s2[[4]][6], xx, KL.rf2.s1[[4]][6],
length=0.05,
```r
angle = 90, code = 3, col = rainbow(5)
lines((KL.rf2[[4]][7,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(6), lwd = 2)
arrows(xx, KL.rf2$s2[[4]][7,], xx, KL.rf2$s1[[4]][7,],
length = 0.05,
angle = 90, code = 3, col = rainbow(6))
lines((KL.rf2[[4]][8,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(7), lwd = 2)
arrows(xx, KL.rf2$s2[[4]][8,], xx, KL.rf2$s1[[4]][8,],
length = 0.05,
angle = 90, code = 3, col = rainbow(7))
plot((KL.rf2[[5]][1,]) %>% xx, type = "l", ylim = c(0, 0.8),
ylab = "Kullback-Leibler score",
xlab = "Time (years since surgery)", lwd = 2,
main = "4 random split points")
arrows(xx, KL.rf2$s2[[5]][1,], xx, KL.rf2$s1[[5]][1,],
length = 0.05, angle = 90, code = 3)
lines((KL.rf2[[5]][2,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(1), lwd = 2)
arrows(xx, KL.rf2$s2[[5]][2,], xx, KL.rf2$s1[[5]][2,],
length = 0.05,
angle = 90, code = 3, col = rainbow(1))
lines((KL.rf2[[5]][3,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(2), lwd = 2)
arrows(xx, KL.rf2$s2[[5]][3,], xx, KL.rf2$s1[[5]][3,],
length = 0.05,
angle = 90, code = 3, col = rainbow(2))
lines((KL.rf2[[5]][4,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(3), lwd = 2)
arrows(xx, KL.rf2$s2[[5]][4,], xx, KL.rf2$s1[[5]][4,],
length = 0.05,
angle = 90, code = 3, col = rainbow(3))
lines((KL.rf2[[5]][5,]) %>% xx, type = "l", ylim = c(0, 0.8),
col = rainbow(4), lwd = 2)
arrows(xx, KL.rf2$s2[[5]][5,], xx, KL.rf2$s1[[5]][5,],
length = 0.05,
angle = 90, code = 3, col = rainbow(4))
```
lines((KL.rf2[[5]][6,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[5], lwd=2)
arrows(xx, KL.rf2.s2[[5]][6,], xx, KL.rf2.s1[[5]][6,],
length=0.05,
angle=90, code=3, col=rainbows[5])
lines((KL.rf2[[5]][7,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[6], lwd=2)
arrows(xx, KL.rf2.s2[[5]][7,], xx, KL.rf2.s1[[5]][7,],
length=0.05,
angle=90, code=3, col=rainbows[6])
lines((KL.rf2[[5]][8,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[7], lwd=2)
arrows(xx, KL.rf2.s2[[5]][8,], xx, KL.rf2.s1[[5]][8,],
length=0.05,
angle=90, code=3, col=rainbows[7])

plot((KL.rf2[[6]][1,])~xx, type="l", ylim=c(0,0.8),
ylab="Kullback–Leibler score",
"Time(years since surgery)", lwd=2,
main="5 random split points")
arrows(xx, KL.rf2.s2[[6]][1,], xx, KL.rf2.s1[[6]][1,],
length=0.05, angle=90, code=3)
lines((KL.rf2[[6]][2,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[1], lwd=2)
arrows(xx, KL.rf2.s2[[6]][2,], xx, KL.rf2.s1[[6]][2,],
length=0.05,
angle=90, code=3, col=rainbows[1])
lines((KL.rf2[[6]][3,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[2])

arrows(xx, KL.rf2.s2[[6]][3,], xx, KL.rf2.s1[[6]][3,],
length=0.05,
angle=90, code=3, col=rainbows[2])
lines((KL.rf2[[6]][4,])~xx, type="l", ylim=c(0,0.8),
col=rainbows[3], lwd=2)
arrows(xx, KL.rf2.s2[[6]][4,], xx, KL.rf2.s1[[6]][4,],
length=0.05,
angle=90, code=3, col=rainbows[3])
lines((KL.rf2[[6]][5,])~xx, type="l", ylim=c(0,0.8),


D.3.4 A comparison of different splitting rules

```r
logranktest <- cross.val.rand.for(dat.train, nsplits=3, mtrys=2,
  splitrules="logrank", nodedsizes=200, nodedepths=7)

logrankscoretest <- cross.val.rand.for(dat.train, nsplits=3, mtrys=2,
  splitrules="logrankscore", nodedsizes=200, nodedepths=7)

randomtest <- cross.val.rand.for(dat.train, nsplits=3, mtrys=2,
  splitrules="random", nodedsizes=200, nodedepths=7)

time.logrank <- system.time(cross.val.rand.for(dat.train, nsplits=3, mtrys=2,
  splitrules="logrank", nodedsizes=200, nodedepths=7))

val.means <- rbind(colMeans(logranktest[[1]]),
  colMeans(logrankscoretest[[1]]),
  colMeans(randomtest[[1]]))
val.sds <- rbind(sd(t(logranktest[[1]])),
```

330
sd(t(logrankscoretest[[1]])),
sd(t(randomtest[[1]]))
val.means.u <- val.means + val.sds
val.means.l <- val.means - val.sds

# C index plot for different splitting rule
plot(as.vector(t(val.means))~c(1,2,3), type="p",
ylab="C–index", xaxt="n", lwd=2, xlab="", pch=16,
col=c("black", "red", "blue"))
axis(1, at=c(1,2,3), labels=c("logrank", "logrankscore", "random"))
arrows(c(1,2,3), as.vector(t(val.means.u)), c(1,2,3),
length=0.05, angle=90, code=3,
col=c("black", "red", "blue"))

brier.means <- rbind(colMeans(logranktest[[2]]),
colMeans(logrankscoretest[[2]]),
colMeans(randomtest[[2]]))
brier.sds <- rbind(apply(logranktest[[2]], 2, sd),
apply(logrankscoretest[[2]], 2, sd),
apply(randomtest[[2]], 2, sd))
brier.means.u <- brier.means + brier.sds
brier.means.l <- brier.means - brier.sds

plot(brier.means[1,]~xx, type="l", ylim=c(0,0.25),
ylab="Brier score", xlab="Time (years since surgery)", lwd=2)
arrows(xx, brier.means.u[1,], xx, brier.means.l[1,],
length=0.05, angle=90, code=3)
lines(brier.means[2,]~xx, type="l", ylim=c(0,0.3),
col="red", lwd=2)
arrows(xx, brier.means.u[2,], xx, brier.means.l[2,],
length=0.05, angle=90, code=3, col="red")
lines(brier.means[3,]~xx, type="l", ylim=c(0,0.3),
col="blue", lwd=2)
arrows(xx, brier.means.u[3,], xx, brier.means.l[3,],
length=0.05, angle=90, code=3, col="blue")

legend("bottomright", legend=c("logrank", "logrankscore", "random"),
col=c("black", "red", "blue"),
bty="n", lwd=3)
KL. means <- rbind(colMeans(logranktest[[3]]),
                   colMeans(logrankscoretest[[3]]),
                   colMeans(randomtest[[3]]))
KL. sds <- rbind(apply(logranktest[[3]], 2, sd),
                 apply(logrankscoretest[[3]], 2, sd),
                 apply(randomtest[[3]], 2, sd))
KL. means.u <- KL. means + KL. sds
KL. means.l <- KL. means - KL. sds
plot(KL. means[1,] ~ xx, type="l", ylim=c(0,0.65),
ylab="Kullback–Leibler score",
xlab="Time (years since surgery)", lwd=2)
arrows(xx, KL. means.u[1,] , xx, KL. means.l[1,],
      length=0.05, angle=90, code=3)
lines(KL. means[2,] ~ xx, type="l",
      col="red", lwd=2)
arrows(xx, KL. means.u[2,] , xx, KL. means.l[2,],
      length=0.05, angle=90, code=3, col="red")
lines(KL. means[3,] ~ xx, type="l",
      col="blue", lwd=2)
arrows(xx, KL. means.u[3,] , xx, KL. means.l[3,],
      length=0.05, angle=90, code=3, col="blue")
legend("bottomright", legend=c("logrank", "logrankscore",
                               "random"), col=c("black", "red", "blue"),
       bty="n", lwd=3)

D.3.5 A closer look at the individual tuning parameters

# Looking at individually tuned parameters

# Making cross-validated performance measure plots (Figure 4.7)

all <- cross.val.rand.for(dat.train, nsplits=3, mtrys=2,
                          splitrules="logrank", nodesizes=200, nodedepths=7)
unr <- cross.val.rand.for(dat.train, nsplits=0, mtrys=3,
                          splitrules="logrank", nodesizes=NULL, nodedepths=NULL)
nspl <- cross.val.rand.for(dat.train, nsplits=3, mtrys=0,
                          splitrules="logrank", nodesizes=NULL, nodedepths=NULL)
mtryy <- cross.val.rand.for(dat.train, nsplits=0, mtrys=2, 
  splitrules=“logrank”, nodesizes=NULL, nodedepths=NULL)

nodesiz <- cross.val.rand.for(dat.train, nsplits=0, mtrys=3, 
  splitrules=“logrank”, nodesizes=200, nodedepths=NULL)

nodedep <- cross.val.rand.for(dat.train, nsplits=0, mtrys=3, 
  splitrules=“logrank”, nodesizes=NULL, nodedepths=7)

allbut <- cross.val.rand.for(dat.train, nsplits=3, mtrys=2, 
  splitrules=“logrank”, nodesizes=NULL, nodedepths=7)

val.means.n <- rbind(colMeans(all[[1]]), 
  colMeans(unr[[1]]), 
  colMeans(nspl[[1]]), 
  colMeans(mtryy[[1]]), 
  colMeans(nodesiz[[1]]), 
  colMeans(nodedep[[1]]), 
  colMeans(allbut[[1]]))

val.sds.n <- rbind(sd(t(all[[1]])), 
  sd(t(unr[[1]])), 
  sd(t(nspl[[1]])), 
  sd(t(mtryy[[1]])), 
  sd(t(nodesiz[[1]])), 
  sd(t(nodedep[[1]])), 
  sd(t(allbut[[1]])))

val.means.n.u <- val.means.n + val.sds.n

val.means.n.l <- val.means.n - val.sds.n

HR.means.n <- rbind(colMeans(all[[4]]), 
  colMeans(unr[[4]]), 
  colMeans(nspl[[4]]), 
  colMeans(mtryy[[4]]), 
  colMeans(nodesiz[[4]]), 
  colMeans(nodedep[[4]]), 
  colMeans(allbut[[4]]))

HR.sds.n <- rbind(sd(t(all[[4]])), 
  sd(t(unr[[4]])), 
  sd(t(nspl[[4]])), 
  sd(t(mtryy[[4]])), 
  sd(t(nodesiz[[4]])), 
  sd(t(nodedep[[4]])), 
  sd(t(allbut[[4]])))

HR.means.n.u <- HR.means.n + HR.sds.n

333
HR.m = HR.m - HR.s

LRX.m <- rbind(colMeans(all[[5]]),
               colMeans(unr[[5]]),
               colMeans(nspl[[5]]),
               colMeans(mtryy[[5]]),
               colMeans(nodesiz[[5]]),
               colMeans(nodedep[[5]]),
               colMeans(allbut[[5]]))

LRX.s <- rbind(sd(t(all[[5]])),
                sd(t(unr[[5]])),
                sd(t(nspl[[5]])),
                sd(t(mtryy[[5]])),
                sd(t(nodesiz[[5]])),
                sd(t(nodedep[[5]])),
                sd(t(allbut[[5]])))

LRX.m.u <- LRX.m + LRX.s

# C index (Figure 4.7a)

rainbows2 <- rainbows[1,2,11,6,7,9,12]

plot(as.vector(t(val.m))~seq(1,7), type="p",
     ylim=c(0.65,0.75), cex=1.5,
     ylab="C-index", xaxt="n", lwd=2, xlab="", pch=16,
     col=rainbows2[1:7])

axis(1, at=seq(1,7), labels=FALSE)

text(x=seq(1,7), y=par()$usr[3] - 0.1*(par()$usr[4] - par()$usr[3]),
     labels=c("all.tuned", "untuned", "split.points",
     "no.variables", "node.size", "node.depth",
     "all.but.node.size"), srt=25, adj=1, xpd=TRUE)

arrows(seq(1,7), as.vector(t(val.m.u)), seq(1,7),
       as.vector(t(val.m.1)),
       length=0.05, angle=90, code=3,
       col=rainbows2[1:7])

# HR plots (Figure 4.7d)

plot(as.vector(t(HR.m))~seq(1,7), type="p",
     ylim=c(0,50), cex=1.5,
     ylab="Hazard.ratio", xaxt="n", lwd=2, xlab="", pch=16,
     col=rainbows2[1:7])

axis(1, at=seq(1,7), labels=FALSE)
text(x=seq(1,7), y=par($)usr[3]-0.1*(par($)usr[4]-par($)usr[3]),
     labels=c("all\_tuned", "untuned", "split\_points",
           "no\_variables", "node\_size", "node\_depth",
           "all\_but\_node\_size"), srt=25, adj=1, xpd=TRUE)
     arrows(seq(1,7), as\.vector(t(HR. means\.n\.u)), seq(1,7),
           as\.vector(t(HR. means\.n\.l)),
           length=0.05, angle=90, code=3,
           col=rainbows2[1:7])

# logrank chisquare plots (Figure 4.7e)
plot(as\.vector(t(LRX. means\.n))~seq(1,7), type="p",
     ylim=c(0,50), cex=1.5,
     ylab="LR\_Chisq", xaxt="n", lwd=2, xlab="", pch=16,
     col=rainbows2[1:7])
axis(1, at=seq(1,7), labels=FALSE)
text(x=seq(1,7), y=par($)usr[3]-0.1*(par($)usr[4]-par($)usr[3]),
     labels=c("all\_tuned", "untuned", "split\_points",
           "no\_variables", "node\_size", "node\_depth",
           "all\_but\_node\_size"), srt=25, adj=1, xpd=TRUE)
     arrows(seq(1,7), as\.vector(t(LRX. means\.n\.u)), seq(1,7),
           as\.vector(t(LRX. means\.n\.l)),
           length=0.05, angle=90, code=3,
           col=rainbows2[1:7])

# Brier plots (Figure 4.7b)
brier\.n\.means <- rbind(colMeans(all[[2]]),
                          colMeans(unr[[2]]),
                          colMeans(nspl[[2]]),
                          colMeans(mtryy[[2]]),
                          colMeans(nodesiz[[2]]),
                          colMeans(nodedep[[2]]),
                          colMeans(allbut[[2]]))
brier\.n\.sds <- rbind(apply(all[[2]], 2, sd),
                       apply(unr[[2]], 2, sd),
                       apply(nspl[[2]], 2, sd),
                       apply(mtryy[[2]], 2, sd),
                       apply(nodesiz[[2]], 2, sd),
                       apply(nodedep[[2]], 2, sd),
                       apply(allbut[[2]], 2, sd))
brier\.n\.means\.u <- brier\.n\.means + brier\.n\.sds
brier\.n\.means\.l <- brier\.n\.means - brier\.n\.sds

xx <- seq(1,6)
plot(brier\.n\.means[2,]~xx, type="l", ylim=c(0,0.25),
     ylab="Brier score", xlab="Time (years since surgery)", lwd=2,


```r
150 col = rainbows2[2])
151 arrows(xx, brier.n.means.u[2], xx, brier.n.means.l[2],
152 length = 0.05, angle = 90, code = 3, rainbows2[2])
153 lines(brier.n.means[3], xx, type = "1", ylim = c(0, 0.3),
154 lwd = 2, col = rainbows2[3])
155 arrows(xx, brier.n.means.u[3], xx, brier.n.means.l[3],
156 length = 0.05, angle = 90, code = 3, col = rainbows2[3])
157 lines(brier.n.means[4], xx, type = "1", ylim = c(0, 0.3),
158 lwd = 2, col = rainbows2[4])
159 arrows(xx, brier.n.means.u[4], xx, brier.n.means.l[4],
160 length = 0.05, angle = 90, code = 3, col = rainbows2[4])
161 lines(brier.n.means[5], xx, type = "1", ylim = c(0, 0.3),
162 col = rainbows2[5], lwd = 2)
163 arrows(xx, brier.n.means.u[5], xx, brier.n.means.l[5],
164 length = 0.05, angle = 90, code = 3, col = rainbows2[5])
165 lines(brier.n.means[6], xx, type = "1", ylim = c(0, 0.3),
166 col = rainbows2[6], lwd = 2)
167 arrows(xx, brier.n.means.u[6], xx, brier.n.means.l[6],
168 length = 0.05, angle = 90, code = 3, col = rainbows2[6])
169 lines(brier.n.means[7], xx, type = "1", ylim = c(0, 0.3),
170 col = rainbows2[7], lwd = 2)
171 arrows(xx, brier.n.means.u[7], xx, brier.n.means.l[7],
172 length = 0.05, angle = 90, code = 3, col = rainbows2[7])
173 lines(brier.n.means[1], xx, type = "1", ylim = c(0, 0.3),
174 lwd = 2, col = rainbows2[1])
175 arrows(xx, brier.n.means.u[1], xx, brier.n.means.l[1],
176 length = 0.05, angle = 90, code = 3, col = rainbows2[1])
177 legend("bottomright", legend = c("all\_tuned", "untuned",
178 "split\_points",
179 "no\_variables", "node\_size", "node\_depth",
180 "all\_but\_node\_size"), col = rainbows2,
181 bty = "n", lwd = 5, ncol = 2)

# Kullback–Leibler plots (Figure 4.7e)
185 KL.n.means <- rbind(colMeans(all[[3]]),
186 colMeans(unr[[3]]),
187 colMeans(nspl[[3]]),
188 colMeans(mtryy[[3]]),
189 colMeans(nodesiz[[3]]),
190 colMeans(nodedep[[3]]),
191 colMeans(allbut[[3]]))
194 KL.n.sds <- rbind(apply(all[[3]], 2, sd),
195 336
```
apply(unr[[3]], 2, sd),
apply(nspl[[3]], 2, sd),
apply(mtryy[[3]], 2, sd),
apply(nodesiz[[3]], 2, sd),
apply(nodedep[[3]], 2, sd),
apply(allbut[[3]], 2, sd))

KL.n.means.u <- KL.n.means + KL.n.sds
KL.n.means.l <- KL.n.means - KL.n.sds

plot(KL.n.means[2,]~xx, type="l", ylab="Kullback-Leibler score",
      xlab="Time (years since surgery)", lwd=2,
      col=rainbows2[2])

arrows(xx, KL.n.means.u[2,], xx, KL.n.means.l[2,],
       length=0.05, angle=90, code=3,rainbows2[2])

lines(KL.n.means[3,]~xx, type="l", ylim=c(0,0.8),
      lwd=2, col=rainbows2[3])

arrows(xx, KL.n.means.u[3,], xx, KL.n.means.l[3,],
       length=0.05, angle=90, code=3, col=rainbows2[3])

lines(KL.n.means[4,]~xx, type="l", ylim=c(0,0.8),
      lwd=2, col=rainbows2[4])

arrows(xx, KL.n.means.u[4,], xx, KL.n.means.l[4,],
       length=0.05, angle=90, code=3, col=rainbows2[4])

lines(KL.n.means[5,]~xx, type="l", ylim=c(0,0.8),
      lwd=2, col=rainbows2[5])

arrows(xx, KL.n.means.u[5,], xx, KL.n.means.l[5,],
       length=0.05, angle=90, code=3, col=rainbows2[5])

lines(KL.n.means[6,]~xx, type="l", ylim=c(0,0.8),
      col=rainbows2[6], lwd=2)

arrows(xx, KL.n.means.u[6,], xx, KL.n.means.l[6,],
       length=0.05, angle=90, code=3, col=rainbows2[6])

lines(KL.n.means[7,]~xx, type="l", ylim=c(0,0.8),
      col=rainbows2[7], lwd=2)

arrows(xx, KL.n.means.u[7,], xx, KL.n.means.l[7,],
       length=0.05, angle=90, code=3, col=rainbows2[7])

lines(KL.n.means[1,]~xx, type="l", ylim=c(0,0.8),
      lwd=1, col=rainbows2[1])

arrows(xx, KL.n.means.u[1,], xx, KL.n.means.l[1,],
       length=0.05, angle=90, code=3, col=rainbows2[1])

legend("bottomright", legend=c("all tuned", "untuned",
                               "split points", 
                               "no variables", "node size", "node depth",
                               "all but node size"), col=rainbows2,
# Fitting models to full training data

```r
final.rand.test.unr <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=0,mtry=3,nodesize=NULL, nodedepth=NULL, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

```r
final.rand.test.unr.nsplits <- rfsrc(Surv(time, stat) ~ 
                                Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=3,mtry=3,nodesize=NULL, nodedepth=NULL, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

```r
final.rand.test.unr.mtry <- rfsrc(Surv(time, stat) ~ 
                                Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=0,mtry=2,nodesize=NULL, nodedepth=NULL, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

```r
final.rand.test.unr.nodesize <- rfsrc(Surv(time, stat) ~ 
                                Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=0,mtry=3,nodesize=200, nodedepth=NULL, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

```r
final.rand.test.unr.nodedepth <- rfsrc(Surv(time, stat) ~ 
                                Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=0,mtry=3,nodesize=NULL, nodedepth=7, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

```r
final.rand.test.all.but <- rfsrc(Surv(time, stat) ~ 
                                Hist + Age + Volume + Ex + 
                                Sex + lung + other, data = dat.train, 
                                nsplit=3,mtry=2,nodesize=NULL, nodedepth=7, 
                                tree.err = TRUE,importance = TRUE, ntree=750)
```

# Making predictions for probability plots (Figure 4.8)

```r
test.predss <- predict.rfsrc(rfs.final, 
                            newdata = dat.test)
test.preds <- test.predss$survival
```

338
test.pred.unr <- predict.rfsrc(final.rand.test.unr, dat.test)
test.pred.unr <- test.pred.unrs$survival
test.pred.unr.nspls <- predict.rfsrc(final.rand.test.unr.nspl, dat.test)
test.pred.unr.nspl <- test.pred.unr.nspls$survival
test.pred.unr.mtrys <- predict.rfsrc(final.rand.test.unr.mtry, dat.test)
test.pred.unr.mtry <- test.pred.unr.mtrys$survival
test.pred.unr.nodesizes <- predict.rfsrc(final.rand.test.unr.nodesiz, dat.test)
test.pred.unr.nodesiz <- test.pred.unr.nodesizes$survival
test.pred.unr.nodedeps <- predict.rfsrc(final.rand.test.unr.nodedep, dat.test)
test.pred.unr.nodedep <- test.pred.unr.nodedeps$survival
test.pred.unr.all.but.s <- predict.rfsrc(final.rand.test.all.but, dat.test)
test.pred.unr.all.but <- test.pred.unr.all.but.s$survival

# Making all patient prediction plots

col.vec.hist <- as.numeric(dat.test$Hist)
col.vec.hist [which(col.vec.hist==1)] <- "#0000FF66"
col.vec.hist [which(col.vec.hist==2)] <- "#FF000066"

matplot(t(test.preds), type="l", lwd=1, lty=1, col=col.vec.hist, ylim=c(0,1), ylab="Survival probability", xlab="Time (years since surgery)", xaxt="n", main="All parameters tuned")
axis(side=1, at=seq(0,350,by=50), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"), col=c("blue", "red"), lty=1, lwd=2, bty="n")

matplot(t(test.pred.unr), type="l", lwd=1, lty=1, col=col.vec.hist, ylim=c(0,1), ylab="Survival probability", xlab="Time (years since surgery)", xaxt="n", main="No parameters tuned (default)"
```r
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.pred.unr.nspl), type="l", lwd=1, lty=1,
  col=col.vec.hist, ylim=c(0,1),ylab="Survival probability",
  xlab="Time (years since surgery)",
  xaxt="n", main="Random split points tuned")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.pred.unr.mtry), type="l", lwd=1, lty=1,
  col=col.vec.hist, ylim=c(0,1),ylab="Survival probability",
  xlab="Time (years since surgery)",
  xaxt="n", main="No. variables tuned")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.pred.unr.nodesiz), type="l", lwd=1, lty=1,
  col=col.vec.hist, ylim=c(0,1),ylab="Survival probability",
  xlab="Time (years since surgery)",
  xaxt="n", main="Node size tuned")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.pred.unr.nodedep), type="l", lwd=1, lty=1,
  col=col.vec.hist, ylim=c(0,1),ylab="Survival probability",
  xlab="Time (years since surgery)",
  xaxt="n", main="Node depth tuned")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.pred.unr.all.but), type="l", lwd=1, lty=1,
  col=col.vec.hist, ylim=c(0,1),ylab="Survival probability",
  xlab="Time (years since surgery)",
  xaxt="n", main="All but node size tuned")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
  col=c("blue", "red"), lty=1, lwd=2, bty="n")
```
# Looking at extreme values for parameters

# Training models with extreme parameter values on training data

unr <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nsplit=0, mtry=3, nodesize=NULL, nodedepth=NULL, tree.err = TRUE, importance = TRUE, ntree=750)

min.nsplits <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nsplit=1, tree.err = TRUE, importance = TRUE, ntree=750)

ext.nsplits <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nsplit=5, tree.err = TRUE, importance = TRUE, ntree=750)

ext.mtry <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, mtry=7, tree.err = TRUE, importance = TRUE, ntree=750)

min.mtry <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, mtry=1, nodesize=200, tree.err = TRUE, importance = TRUE, ntree=750)

ext.nodes <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nodesize=400, tree.err = TRUE, importance = TRUE, ntree=750)

min.nodes <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nodesize=1, tree.err = TRUE, importance = TRUE, ntree=750)

ext.depth <- rfsrc(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train, nodedepth=15,
```r
tree.err = TRUE, importance = TRUE, ntree=750)

min.depth <- rfsr(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data = dat.train,
                   nodedepth=1,
                   tree.err = TRUE, importance = TRUE, ntree=750)

# individual predictions patient with poor histological response, # as predicted on test set. # for comparison Cox model predictions included # (Figure 4.11)

# Patient and cox prediction

cox.pat.poor.lung <- dat.test [1 ,]
cox.pat.poor.lung$Age <- mean(dat.test$Age)
cox.pat.poor.lung$Volume <- mean(dat.test$Volume)
cox.pat.poor.lung$other <- "No"
cox.pat.poor.lung$other<- as.factor(cox.pat.poor.lung$other)
cox.pat.poor.lung$Sex <- "Female"
cox.pat.poor.lung$Sex<- as.factor(cox.pat.poor.lung$Sex)
cox.pat.poor.lung$lung <- "Yes/Possible"
cox.pat.poor.lung$lung <- as.factor(cox.pat.poor.lung$lung)

eur.cox.poor.lung <- predictSurvProb(cox.eur, cox.pat.poor.lung , times=dat.train$time[which(dat.train$stat==1)])

time.points <- unique(dat.train$time[which(dat.train$stat==1)])

plot(as.vector(eur.cox.poor.lung~sort(dat.test$time)),
     type="l", ylim=c(0,1), lty=1, col="black", lwd=2,
     ylab="Survival probability",
     xlab="Time (Years since surgery)",
     main="No parameters modified")

poor.hist <- predict.rfsr(unr, newdata = cox.pat.poor.lung)
poor.hist$survival

lines(as.vector(poor.hist~sort(time.points)),
      type="l", ylim=c(0,1), col="green", lwd=2, lty=1)

legend("bottomleft", legend=c("Cox", "RFS"),
        col=c("black", "green"),
        lty=c(1,1), bty="n", lwd=c(3,3))

plot(as.vector(eur.cox.poor.lung~sort(dat.test$time)),
     type="l", ylim=c(0,1), lty=1, col="black", lwd=2,
     ylab="Survival probability",
```

342
poor.hist <- predict.rfsrce(ext.nsplit,
                          newdata = cox.pat.poor.lung)
lines(as.vector(poor.hist)$survival
       type="1", ylim=c(0,1), col="green", lwd=2, lty=1)
poor.hist <- predict.rfsrce(min.nsplit,
                          newdata = cox.pat.poor.lung)
lines(as.vector(poor.hist)$survival
       type="1", ylim=c(0,1), col="green", lwd=2, lty=2)
legend("bottomleft", legend=c("Cox", "RFS", "5.split.points",
                              "1.split.point"),
       col=c("black", "green", "green", "green"),
       lty=c(1,1,1,2), bty="n", lwd=c(3,3,3,3))

plot(as.vector(eur.cox.poor.lung)$sort(dat.test$time),
      type="l", ylim=c(0,1), lty=1, col="black", lwd=2,
      ylab="Survival probability",
      xlab="Time (Years since surgery)",
      main="Candidate variables")
poor.hist <- predict.rfsrce(ext.mtry,
                          newdata = cox.pat.poor.lung)
lines(as.vector(poor.hist)$survival
       type="1", ylim=c(0,1), col="green", lwd=2, lty=1)
poor.hist <- predict.rfsrce(min.mtry,
                          newdata = cox.pat.poor.lung)
lines(as.vector(poor.hist)$survival
       type="1", ylim=c(0,1), col="green", lwd=2, lty=2)
legend("bottomleft",
       legend=c("Cox", "RFS", "7.candidates", "1.candidate"),
       col=c("black", "green", "green", "green"),
       lty=c(1,1,1,2), bty="n", lwd=c(3,3,3,3))
plot(as.vector(eur.cox.poor.lung)$sort(dat.test$time),
      type="l", ylim=c(0,1), lty=1, col="black", lwd=2,
```r
poor.hist <- predict.rfsrcc( ext.nodes ,
  newdata = cox.pat.poor.lung )
poor.hist <- poor.hist$survival
lines( as.vector(poor.hist)~sort(time.points) ,
  type="l" , ylim=c(0,1) , col="green" , lwd=2, lty=1)

poor.hist <- predict.rfsrcc( min.nodes ,
  newdata = cox.pat.poor.lung )
poor.hist <- poor.hist$survival
lines( as.vector(poor.hist)~sort(time.points) ,
  type="l" , ylim=c(0,1) , col="green" , lwd=2, lty=2)

legend("bottomleft" ,
  legend=c("Cox" , "RFS" , "Node\_size\_400" , "Node\_size\_1" ) ,
  col=c("black" , "green" , "green" , "green" ) ,
  lty=c(1,1,1,2) , bty="n" , lwd=c(3,3,3,3))

plot( as.vector(eur.cox.poor.lung)~sort(dat.test$time) ,
  type="l" , ylim=c(0,1) , lty=1 , col="black" , lwd=2 ,
  ylab="Survival probability" ,
  xlab="Time\_ (Years\_ since\_ surgery ) " ,
  main="Node depth")
poor.hist <- predict.rfsrcc( ext.depth, newdata = cox.pat.poor.lung )
poor.hist <- poor.hist$survival
lines( as.vector(poor.hist)~sort(time.points) ,
  type="l" , ylim=c(0,1) , col="green" , lwd=2, lty=1)

poor.hist <- predict.rfsrcc( min.depth ,
  newdata = cox.pat.poor.lung )
poor.hist <- poor.hist$survival
lines( as.vector(poor.hist)~sort(time.points) ,
  type="l" , ylim=c(0,1) , col="green" , lwd=2, lty=2)

legend("bottomleft" ,
  legend=c("Cox" , "RFS" , "Node\_ depth\_15" , "Node\_ depth\_1") ,
  col=c("black" , "green" , "green" , "green") ,
  lty=c(1,1,1,2) , bty="n" , lwd=c(3,3,3,3))

# making predictions for each model for all patients
```
test.unr <- predict.rfsrc(unr, dat.test)
test.unr <- test.unr$survival
test.ext.nsplit <- predict.rfsrc(ext.nsplit, dat.test)
test.ext.nsplit <- test.pred.unrs$survival
test.min.nsplit <- predict.rfsrc(min.nsplit, dat.test)
test.min.nsplit <- test.min.nsplit$survival
test.ext.mtry <- predict.rfsrc(ext.mtry, dat.test)
test.ext.mtry <- test.ext.mtry$survival
test.min.mtry <- predict.rfsrc(min.mtry, dat.test)
test.min.mtry <- test.min.mtry$survival
test.ext.nodes <- predict.rfsrc(ext.nodes, dat.test)
test.ext.nodes <- test.ext.nodes$survival
test.min.nodes <- predict.rfsrc(min.nodes, dat.test)
test.min.nodes <- test.min.nodes$survival
test.ext.depth <- predict.rfsrc(ext.depth, dat.test)
test.ext.depth <- test.ext.depth$survival
test.min.depth <- predict.rfsrc(min.depth, dat.test)
test.min.depth <- test.min.depth$survival

# making prediction plots for all patients, with colour
# indication according to histological response
# (Figure 4.10)
col.vec.hist <- as.numeric(dat.test$Hist)
col.vec.hist [which(col.vec.hist==1)] <- "#0000FF66"
col.vec.hist [which(col.vec.hist==2)] <- "#FF000066"

matplot(t(test.unr), type="l", lwd=1, lty=1, col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
         xlab="Time (years since surgery)",
         xaxt="n", main="No parameters modified")
axis(side=1, at=seq(0,350,by=50), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
        col=c("blue", "red"), lty=1, lwd=2, bty="n")

matplot(t(test.ext.nsplit), type="l", lwd=1, lty=1, col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
         xlab="Time (years since surgery)",
         xaxt="n", main="No parameters modified")
axis(side=1, at=seq(0,350,by=50), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
        col=c("blue", "red"), lty=1, lwd=2, bty="n")
xlab="Time (years since surgery)",
xaxt="n", main="5 random split points"
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.min.nsplit), type="l", lwd=1, lty=1,
col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
xlab="Time (years since surgery)",
xaxt="n", main="1 random split point")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.ext.mtry), type="l", lwd=1, lty=1,
col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
xlab="Time (years since surgery)",
xaxt="n", main="7 candidate variables")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.min.mtry), type="l", lwd=1, lty=1,
col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
xlab="Time (years since surgery)",
xaxt="n", main="1 candidate variable")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.ext.nodes), type="l", lwd=1, lty=1,
col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
xlab="Time (years since surgery)",
xaxt="n", main="Node size of 400")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
matplot(t(test.min.nodes), type="l", lwd=1, lty=1,
col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
xlab="Time (years since surgery)",
xaxt="n", main="Node size of 1")
axis(side=1, at=(seq(0,350,by=50)), labels=seq(0,7))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
col=c("blue", "red"), lty=1, lwd=2, bty="n")
```r
matplot(t(test.ext.depth), type="l", lwd=1, lty=1,
        col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
        xlab="Time (years since surgery)",
        xaxt="n", main="Node depth of 15")
axis(side=1, at=seq(0,350,by=50)), labels=seq(0,7)
legend("bottomleft", legend=c("Good hist", "Poor hist"),
        col=c("blue", "red"), lty=1, lwd=2, bty="n")

matplot(t(test.min.depth), type="l", lwd=1, lty=1,
        col=col.vec.hist, ylim=c(0,1), ylab="Survival probability",
        xlab="Time (years since surgery)",
        xaxt="n", main="Node depth of 1")
axis(side=1, at=seq(0,350,by=50)), labels=seq(0,7)
legend("bottomleft", legend=c("Good hist", "Poor hist"),
        col=c("blue", "red"), lty=1, lwd=2, bty="n")

# obtaining cross−validated performance measure estimates
unr.c <- cross.val.rand.for(dat.train, mtrys=NULL,
                              splitrules="logrank", nodesizes=NULL, nodedepths=NULL)

min.nsplit.c <- cross.val.rand.for(dat.train, nsplit=1, mtrys=NULL,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=NULL)

ext.nsplit.c <- cross.val.rand.for(dat.train, nsplit=5, mtrys=NULL,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=NULL)

ext.mtry.c <- cross.val.rand.for(dat.train, mtrys=7,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=NULL)

min.mtry.c <- cross.val.rand.for(dat.train, mtrys=1,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=NULL)

ext.nodes.c <- cross.val.rand.for(dat.train, mtrys=NULL,
                                   splitrules="logrank", nodesizes=400, nodedepths=NULL)

min.nodes.c <- cross.val.rand.for(dat.train, mtrys=NULL,
                                   splitrules="logrank", nodesizes=1, nodedepths=NULL)

ext.depth.c <- cross.val.rand.for(dat.train, mtrys=NULL,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=15)

min.depth.c <- cross.val.rand.for(dat.train, mtrys=NULL,
                                   splitrules="logrank", nodesizes=NULL, nodedepths=15)
```

347
```r
splitrules="logrank", nodesizes=NULL, nodedepths=1)

val.means.n <- rbind(colMeans(unr.c[[1]]),
                      colMeans(ext.nsplit.c[[1]]),
                      colMeans(min.nsplit.c[[1]]),
                      colMeans(ext.mtry.c[[1]]),
                      colMeans(min.mtry.c[[1]]),
                      colMeans(ext.nodes.c[[1]]),
                      colMeans(min.nodes.c[[1]]),
                      colMeans(ext.depth.c[[1]]),
                      colMeans(min.depth.c[[1]])
val.sds.n <- rbind(sd(t(unr.c[[1]])),
                   sd(t(ext.nsplit.c[[1]])),
                   sd(t(min.nsplit.c[[1]])),
                   sd(t(ext.mtry.c[[1]])),
                   sd(t(min.mtry.c[[1]])),
                   sd(t(ext.nodes.c[[1]])),
                   sd(t(min.nodes.c[[1]])),
                   sd(t(ext.depth.c[[1]])),
                   sd(t(min.depth.c[[1]])))
val.means.n.u <- val.means.n + val.sds.n
val.means.n.l <- val.means.n - val.sds.n

HR.means.n <- rbind(colMeans(unr.c[[4]]),
                    colMeans(ext.nsplit.c[[4]]),
                    colMeans(min.nsplit.c[[4]]),
                    colMeans(ext.mtry.c[[4]]),
                    colMeans(min.mtry.c[[4]]),
                    colMeans(ext.nodes.c[[4]]),
                    colMeans(min.nodes.c[[4]]),
                    colMeans(ext.depth.c[[4]]),
                    colMeans(min.depth.c[[4]]))
HR.sds.n <- rbind(sd(t(unr.c[[4]])),
                  sd(t(ext.nsplit.c[[4]])),
                  sd(t(min.nsplit.c[[4]])),
                  sd(t(ext.mtry.c[[4]])),
                  sd(t(min.mtry.c[[4]])),
                  sd(t(ext.nodes.c[[4]])),
                  sd(t(min.nodes.c[[4]])),
                  sd(t(min.depth.c[[4]])))
```

348
HR. means. n . u <- HR. means. n + HR. sds. n
HR. means. n . l <- HR. means. n - HR. sds. n
LRX. means. n <- rbind(colMeans(unr. c[[5]]),
colMeans(ext. nsplt. c[[5]]),
colMeans(min. nsplt. c[[5]]),
colMeans(ext. mtry. c[[5]]),
colMeans(min. mtry. c[[5]]),
colMeans(ext. nodes. c[[5]]),
colMeans(min. nodes. c[[5]]),
colMeans(ext. depth. c[[5]]),
colMeans(min. depth. c[[5]])
)
LRX. sds. n <- rbind(sd(t(unr. c[[5]])),
sd(t(ext. nsplt. c[[5]])),
sd(t(min. nsplt. c[[5]])),
sd(t(ext. mtry. c[[5]])),
sd(t(min. mtry. c[[5]])),
sd(t(ext. nodes. c[[5]])),
sd(t(min. nodes. c[[5]])),
sd(t(ext. depth. c[[5]])),
sd(t(min. depth. c[[5]]))
)
LRX. means. n . u <- LRX. means. n + LRX. sds. n
LRX. means. n . l <- LRX. means. n - LRX. sds. n

# C−index plot (Figure 4.9a)
ww <- seq(1,9)
rainbows3 <- c(rainbows[c(1,2,11,6,7,9,12)], "green", "turquoise")
plot(as.vector(t(val. means. n))~ww, type="p",
ylim=c(0.65,0.72),
ylab="C−index", xlab="n", lwd=2, xlab="", pch=16,
col=rainbows3[1:9], cex=1.5)
axis(1, at=ww, labels=FALSE)
text(x=ww, y=par()$usr[3]-0.1*(par()$usr[4]-par()$usr[3]),
labels=c("unmodified", "5 split points", "1 split point",
"7 variables", "1 variable", "node size 400",
"node size 1", "node depth 15", "node depth 1"),
srt=25, adj=1, xpd=TRUE)
arrows(ww, as.vector(t(val.means.n.u)), ww,
as.vector(t(val.means.n.1)), length=0.05, angle=90, code=3,
col=rainbows3[1:9])

# Hazard ratio plot (Figure 4.9d)
plot(as.vector(t(HR.means.n))~ww, type="p",
ylim=c(0,40),
ylab="Hazard ratio", xaxt="n", lwd=2, xlab="", pch=16,
col=rainbows3[1:9], cex=1.5)
axis(1, at=ww, labels=FALSE)
text(x=ww, y=par()$usr[3]-0.1*(par()$usr[4]-par()$usr[3]),
labels=c("unmodified", "5 split points", "1 split point",
"7 variables", "1 variable", "node size 400",
"node size 1", "node depth 15", "node depth 1"),
srt=25, adj=1, xpd=TRUE)
arrows(ww, as.vector(t(HR.means.n.u)), ww,
as.vector(t(HR.means.n.1)), length=0.05, angle=90, code=3,
col=rainbows3[1:9])

# Logrank chisquare plot (Figure 4.9e)
plot(as.vector(t(LRX.means.n))~ww, type="p",
ylim=c(0,40),
ylab="LR Chisq", xaxt="n", lwd=2, xlab="", pch=16,
col=rainbows3[1:9], cex=1.5)
axis(1, at=ww, labels=FALSE)
text(x=ww, y=par()$usr[3]-0.1*(par()$usr[4]-par()$usr[3]),
labels=c("unmodified", "5 split points", "1 split point",
"7 variables", "1 variable", "node size 400",
"node size 1", "node depth 15", "node depth 1"),
srt=25, adj=1, xpd=TRUE)
arrows(ww, as.vector(t(LRX.means.n.u)), ww,
as.vector(t(LRX.means.n.1)), length=0.05, angle=90, code=3,
col=rainbows3[1:9])

# brier plot (Figure 4.9b)
brier.n.means <- rbind(colMeans(unr.c[[2]]),
colMeans(ext.nsplits.c[[2]]),
colMeans(min.nsplits.c[[2]]),
colMeans(ext.mtry.c[[2]]),
colMeans(min.mtry.c[[2]]),
colMeans(ext.nodes.c[[2]])

350
colMeans(mn.1),
colMeans(ed.1),
colMeans(md.1)

brier.n.sds <- rbind(
apPLY(unr.1,2,sd),
apPLY(edn.1,2,sd),
apPLY(md.1,2,sd),
apPLY(mdn.1,2,sd),
apPLY(ed.1,2,sd),
apPLY(md.1,2,sd),
apPLY(mdn.1,2,sd))
brier.n.means.u <- brier.n.means + brier.n.sds
brier.n.means.l <- brier.n.means - brier.n.sds

legend("bottomright", legend=c("unmodified", "5 split points", "7 variables", "1 variable", "node size 400", "node size 1", "node depth 15", "node depth 1"), col=rainbows3, bty="n", lwd=5, ncol=2)

plot(brier.n.means[1,] ~ xx, type="l", ylim=c(0,0.25), ylab="Brier score", xlab="Time (years since surgery)", lwd=2, col=rainbows3[1])
arrows(xx, brier.n.means.u[1,], xx, brier.n.means.l[1,], length=0.05, angle=90, code=3, rainbows3[1])
lines(brier.n.means[2,] ~ xx, type="l", ylim=c(0,0.3), lwd=2, col=rainbows3[2])
arrows(xx, brier.n.means.u[2,], xx, brier.n.means.l[2,], length=0.05, angle=90, code=3, col=rainbows3[3])
lines(brier.n.means[3,] ~ xx, type="l", ylim=c(0,0.3), lwd=2, col=rainbows3[3])
arrows(xx, brier.n.means.u[3,], xx, brier.n.means.l[3,], length=0.05, angle=90, code=3, col=rainbows3[3])
lines(brier.n.means[4,] ~ xx, type="l", ylim=c(0,0.3), lwd=2, col=rainbows3[4])
arrows(xx, brier.n.means.u[4,], xx, brier.n.means.l[4,], length=0.05, angle=90, code=3, col=rainbows3[4])
lines(brier.n.means[5,] ~ xx, type="l", ylim=c(0,0.3), col=rainbows3[5], lwd=2)
arrows(xx, brier.n.means.u[5,], xx, brier.n.means.l[5,], length=0.05, angle=90, code=3, col=rainbows3[5])
lines(brier.n.means[6,] ~ xx, type="l", ylim=c(0,0.3), col=rainbows3[6], lwd=2)
arrows(xx, brier.n.means[6], xx, brier.n.means.1[6], 
length=0.05, angle=90, code=3, col=rainbows3[6])
lines(brier.n.means[7] xx, type="l", ylim=c(0,0.3), 
col=rainbows3[7], lwd=2)
arrows(xx, brier.n.means.u[7], xx, brier.n.means.1[7], 
length=0.05, angle=90, code=3, col=rainbows3[7])
lines(brier.n.means[8] xx, type="l", ylim=c(0,0.3), 
lwd=2, col=rainbows3[8])
arrows(xx, brier.n.means.u[8], xx, brier.n.means.1[8], 
length=0.05, angle=90, code=3, col=rainbows3[8])
lines(brier.n.means[9] xx, type="l", ylim=c(0,0.3), 
lwd=2, col=rainbows3[9])
arrows(xx, brier.n.means.u[9], xx, brier.n.means.1[9], 
length=0.05, angle=90, code=3, col=rainbows3[9])

# kullback−leibler plots (Figure 4.9c)
KL.n.means <- rbind(colMeans(unr.c[[3]]), 
colMeans(min.nsplt.c[[3]]), 
colMeans(ext.mtry.c[[3]]), 
colMeans(min.mtry.c[[3]]), 
colMeans(ext.nodes.c[[3]]), 
colMeans(min.nodes.c[[3]]), 
colMeans(ext.depth.c[[3]]), 
colMeans(min.depth.c[[3]]))
KL.n.sds <- rbind(apply(unr.c[[3]], 2, sd), 
apply(min.nsplt.c[[3]], 2, sd), 
apply(ext.mtry.c[[3]], 2, sd), 
apply(min.mtry.c[[3]], 2, sd), 
apply(ext.nodes.c[[3]], 2, sd), 
apply(min.nodes.c[[3]], 2, sd), 
apply(ext.depth.c[[3]], 2, sd), 
apply(min.depth.c[[3]], 2, sd))
KL.n.means.u <- KL.n.means + KL.n.sds
KL.n.means.1 <- KL.n.means - KL.n.sds

plot(KL.n.means[1,] xx, type="l", ylim=c(0,0.8), 
ylab="Kullback−Leibler score", 
mlab="Time (years since surgery)", lwd=2, 
col=rainbows3[1])
arrows(xx, KL.n.means.u[1], xx, KL.n.means.1[1], 
length=0.05, angle=90, code=3, rainbows3[1])
lines(KL.n.means[2,] xx, type="l", ylim=c(0,0.3),
D.3.6 Results

# Cox performance measure values for comparison

cox.eur <- coxph(Surv(time, stat) ~ Hist + Age + Volume + Ex + Sex + lung + other, data=dat.train)
preds.eur.b <- predictSurvProb(cox.eur, dat.test, times=sort(dat.test$stat))

cox.stat <- as.data.frame(cbind(dat.test$time, dat.test$stat))
colnames(cox.stat) <- c("surv", "stat")
cox.pred.matrix <- predictSurvProb(cox.eur, dat.test, times=sort(dat.test$stat))

mm <- model.matrix(cox.eur$formula, dat.test)
```r
mm <- mm[, !(colnames(mm) %in% "(Intercept)")]
prog.cox <- mm %>% cox.eur$coefficients

cox.dat <- as.data.frame(cbind(dat.test$time, dat.test$stat, prog.cox))
colnames(cox.dat) <- c("surv", "stat", "pred")

cox.C <- cindex.mod.r(cox.dat)

cox.brierKL <- brier.function.cox.m(cox.stat, cox.pred.matrix)

prog.cox.norm <- (prog.cox - min(prog.cox))/
(max(prog.cox) - min(prog.cox))

HR.cox <- exp(coxph(Surv(time, stat) ~ prog.cox.norm, 
data=cbind(dat.test, prog.cox.norm))$coef)

groupC <- rep(1, nrow(dat.test))

objC <- survdiff(Surv(time, stat) ~ groupC, data=temp.datC)
LRX.C <- objC$chisq

# Performance measures for three final models
# performance measures three final models

final.rand.test <- final.rand.for(dat.train, dat.test, nsplits=3, 
mtrys=2, splitrules="logrank", 
nodesizes=200, nodedepths=7)
final.rand.test.2 <- final.rand.for(dat.train, dat.test, nsplits=3, 
mtrys=2, splitrules="logrank", 
nodesizes=NULL, nodedepths=7)
final.untrained <- final.rand.for(dat.train, dat.test, nsplits=0, 
mtrys=3, splitrules="logrank", 
nodesizes=NULL, nodedepths=NULL)

C.all <- final.rand.test[[1]]
C.allbut <- final.rand.test.2[[1]]
C.Unt <- final.untrained[[1]]

HR.all <- final.rand.test[[4]]
HR.allbut <- final.rand.test.2[[4]]
HR.Unt <- final.untrained[[4]]

LRX.all <- final.rand.test[[5]]
```
LRX.allbut <- final.rand.test.2[[5]]
LRX.Untr <- final.untrained[[5]]

plot(cox.brierKL[1,]~seq(1,8), type="l", ylab=c("Brier/KLscore"), xlab="Time(Years since surgery)", ylim=c(0,0.8), lwd=2, lty=1)
lines(final.rand.test[[2]], lwd=2, lty=1, col="red")
lines(final.rand.test.2[[2]], lwd=2, lty=1, col="green")
lines(final.untrained[[2]], lwd=2, lty=1, col="purple")
lines(cox.brierKL[2,]~seq(1,8), type="l", ylab=c("Brier/KLscore"), xlab="Time(Years since surgery)", ylim=c(0,0.8), lwd=2, lty=2, col="black")
lines(final.rand.test[[3]], lwd=2, col="red", lty=2)
lines(final.rand.test.2[[3]], lwd=2, col="green", lty=2)
lines(final.untrained[[3]], lwd=2, col="purple", lty=2)
legend("topleft", legend=c("Cox Brier", "Cox KL", "RSF Brier(1)", "RSF KL(1)", "RSF Brier(2)", "RSF KL(2)", "RSF Brier(3)", "RSF KL(3)", col=c("black", "black", "red", "red", "green", "green", "purple", "purple"), lty=c(1,2,1,2,1,2), bty="n", lwd=2)

# Making prediction plots for three RFS models, and Cox

# Patients

#good hist
cox.pat <- dat.test[1,]
cox.pat$Hist <- "Good Hist"
cox.pat$Hist <- as.factor(cox.pat$Hist)
cox.pat$Age <- mean(dat.test$Age)
cox.pat$Volume <- mean(dat.test$Volume)
cox.pat$lung <- "No"
cox.pat$lung <- as.factor(cox.pat$lung)
cox.pat$other <- "No"
cox.pat$other <- as.factor(cox.pat$other)
cox.pat$Sex <- "Female"
cox.pat$Sex <- as.factor(cox.pat$Sex)
eur.cox.good <- predictSurvProb(cox.eur, cox.pat, times=dat.test$time)
# poor hist
cox.pat.poor <- dat.test[1,]
cox.pat.poor$Age <- mean(dat.test$Age)
cox.pat.poor$Volume <- mean(dat.test$Volume)
cox.pat.poor$lung <- "No"
cox.pat.poor$other <- as.factor(cox.pat.poor$other)
cox.pat.poor$Sex <- "Female"
cox.pat.poor$lung <- as.factor(cox.pat.poor$lung)
cox.pat.poor$other <- as.factor(cox.pat.poor$other)
cox.pat.poor$Sex <- as.factor(cox.pat.poor$Sex)

eur.cox.poor <- predictSurvProb(cox.eur, cox.pat.poor, times=dat.test$time)

#poor hist, lungmet.
cox.pat.poor.lung <- dat.test[1,]
cox.pat.poor.lung$Age <- mean(dat.test$Age)
cox.pat.poor.lung$Volume <- mean(dat.test$Volume)
cox.pat.poor.lung$other <- "No"
cox.pat.poor.lung$other <- as.factor(cox.pat.poor.lung$other)
cox.pat.poor.lung$Sex <- "Female"
cox.pat.poor.lung$Sex <- as.factor(cox.pat.poor.lung$Sex)
cox.pat.poor.lung$lung <- "Yes/Possible"
cox.pat.poor.lung$lung <- as.factor(cox.pat.poor.lung$lung)

eur.cox.poor.lung <- predictSurvProb(cox.eur, cox.pat.poor.lung, times=dat.test$time)

# RFS model with all parameters tuned

good.hist <- predict.rfs.rcc(rfs.final, newdata = cox.pat)
good.hist <- good.hist$survival

poor.hist <- predict.rfs.rcc(rfs.final, newdata = cox.pat.poor)
poor.hist <- poor.hist$survival

poor.hist.lung <- predict.rfs.rcc(rfs.final, newdata = cox.pat.poor.lung)
poor.hist.lung <- poor.hist.lung$survival

time.points <- unique(dat.train$time[which(dat.train$stat==1)])
length(time.points)
#main: RFS 1

```r
plot(as.vector(eur.cox.good)~sort(dat.test$time),
     type="l", ylim=c(0,1), lty=1, col="green", lwd=2,
     ylab="Survival probability",
     xlab="Time (Years since surgery)"
)
lines(as.vector(good.hist)~sort(time.points), type="l",
      ylim=c(0,1), col="green", lwd=2, lty=2)
lines(as.vector(eur.cox.poor)~sort(dat.test$time),
      type="l", ylim=c(0,1), lty=1, col="orange", lwd=2)
lines(as.vector(poor.hist)~sort(time.points), type="l",
      ylim=c(0,1), col="orange", lwd=2, lty=2)
lines(as.vector(eur.cox.poor.lung)~sort(dat.test$time),
      type="l", ylim=c(0,1), lty=1, col="purple",
      lwd=2)
lines(as.vector(poor.hist.lung)~sort(time.points),
      type="l", ylim=c(0,1), col="purple", lwd=2, lty=2)
legend("bottomleft", legend=c("Reference", "Poor hist",
                              "Poor hist + lung mets", "Cox", "RFS"),
        col=c("green", "orange", "purple", "black", "black"),
        lty=c(1,1,1,1,3), bty="n", lwd=3)
```

# all parameters but node size tuned

```r
good.hist <- predict.rfsr(fina.rand.test.all.but,
                              newdata = cox.pat)
good.hist <- good.hist$survival
poor.hist <- predict.rfsr(fina.rand.test.all.but,
                              newdata = cox.pat.poor)
poor.hist <- poor.hist$survival
poor.hist.lung <- predict.rfsr(fina.rand.test.all.but,
                               newdata = cox.pat.poor.lung)
poor.hist.lung <- poor.hist.lung$survival
time.points <- unique(dat.train$time[which(dat.train$stat==1)])
length(time.points)
plot(as.vector(eur.cox.good)~sort(dat.test$time),
     type="l", ylim=c(0,1), lty=1, col="green", lwd=2,
     ylab="Survival probability",
     xlab="Time (Years since surgery)"
)
lines(as.vector(good.hist)~sort(time.points), type="l",
      ylim=c(0,1), col="green", lwd=2, lty=2)
```
ylim=c(0,1), col="green", lwd=2, lty=2)
lines(as.vector(eur.cox.poor)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="orange", lwd=2)
lines(as.vector(poor.hist)~sort(time.points), type="l"
, ylim=c(0,1), col="orange", lwd=2, lty=2)
lines(as.vector(eur.cox.poor.lung)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="purple",
lwd=2)
lines(as.vector(poor.hist.lung)~sort(time.points),
type="l", ylim=c(0,1), col="purple", lwd=2, lty=2)

time.points <- unique(dat.train$time[which(dat.train$stat==1)])
length(time.points)

plot(as.vector(eur.cox.good)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="green", lwd=2,
  ylab="Survival probability", xlab="Time (Years since surgery)"
)lines(as.vector(good.hist)~sort(time.points), type="l",
  ylim=c(0,1), col="green", lwd=2, lty=2)
lines(as.vector(eur.cox.poor)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="orange", lwd=2)
lines(as.vector(poor.hist)~sort(time.points), type="l",
  ylim=c(0,1), col="orange", lwd=2, lty=2)
lines(as.vector(eur.cox.poor.lung)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="purple",
lwd=2)
lines(as.vector(poor.hist.lung)~sort(time.points),

# for untrained random forest

good.hist <- predict.rfsr.c(final.rand.test.unr,
  newdata = cox.pat)
good.hist <- good.hist$survival

poor.hist <- predict.rfsr.c(final.rand.test.unr,
  newdata = cox.pat.poor)
poor.hist <- poor.hist$survival

poor.hist.lung <- predict.rfsr.c(final.rand.test.unr,
  newdata = cox.pat.poor.lung)
poor.hist.lung <- poor.hist.lung$survival

# for untrained random forest

plot (as.vector(eur.cox.good)~sort(dat.test$time),
type="l", ylim=c(0,1), lty=1, col="green", lwd=2,
  ylab="Survival probability", xlab="Time (Years since surgery)"
)lines (as.vector (good.hist) ~ sort (time.points),
  type="l",
  ylim=c(0,1), col="green", lwd=2, lty=2)
lines (as.vector (eur.cox.poor) ~ sort (dat.test$time),
type="l", ylim=c(0,1), lty=1, col="orange", lwd=2)
lines (as.vector (poor.hist) ~ sort (time.points),
  type="l",
  ylim=c(0,1), col="orange", lwd=2, lty=2)
lines (as.vector (eur.cox.poor.lung) ~ sort (dat.test$time),
type="l", ylim=c(0,1), lty=1, col="purple",
lwd=2)
lines (as.vector (poor.hist.lung) ~ sort (time.points),

}
type=" l " , ylim=c( 0 , 1 ) , col=" purple " , lwd=2, l t y =2)
legend(" bottomleft " , legend=c(" Reference " , " Poor hist ",
" Poor hist + lung mets " , " Cox " , " RFS " ),
col=c(" green " , " orange " , " purple " , " black " , " black " ),
lty=c(1,1,1,1,3) , bty=" n " , lwd=3)

# Making probability plots for all patients
# For good versus poor histological response

col.vec.hist <- as.numeric(dat.test$Hist)
col.vec.hist[which(col.vec.hist==1)] <- "#0000FF66"
col.vec.hist[which(col.vec.hist==2)] <- "#FF000066"

# individual survival probabilities hist: all tuned
obj <- matplot(t(test.preds) , type=" l " , lwd=1, lty=1,
col=col.vec.hist , ylim=c(0,1),
ylab="Survival probability ",
 TCLab=" Time (years since surgery) ",
 xlab="n")
axis(side=1, at=(seq(0,350,by=50)) , labels=seq(0,7))
legend(" bottomleft " , legend=c(" Good hist ", " Poor hist " ),
col=c(" blue " , " red " ) , lty=1, lwd=2, bty=" n " )

# individual survival probabilities hist: all but tuned
obj <- matplot(t(test.pred.unr.all.but) , type=" l " , lwd=1, lty=1,
col=col.vec.hist , ylim=c(0,1),
ylab="Survival probability ",
 TCLab=" Time (years since surgery) ",
 xlab="n")
axis(side=1, at=(seq(0,350,by=50)) , labels=seq(0,7))
legend(" bottomleft " , legend=c(" Good hist ", " Poor hist " ),
col=c(" blue " , " red " ) , lty=1, lwd=2, bty=" n " )

## individual survival probabilities hist: nothing tuned
obj <- matplot(t(test.pred.unr) , type=" l " , lwd=1, lty=1,
col=col.vec.hist , ylim=c(0,1),
ylab="Survival probability ",
 TCLab=" Time (years since surgery) ",
 xlab="n", main="Untuned")
axis(side=1, at=(seq(0,350,by=50)) , labels=seq(0,7))
legend(" bottomleft " , legend=c(" Good hist " , " Poor hist " ),
col=c("blue", "red"), lty=1, lwd=2, bty="n")

# individual survival probabilities histcox
obj2 <- matplot(t(preds.eur.b), type="l", lwd=1,
    lty=1, col=col.vec.hist, xaxt="n", ylim=c(0,1),
    ylab="Survival probability",
    xlab="Time(years since surgery)", main="Cox model")
axis(side=1, at=seq(0,650, by=81.25), labels=seq(0,8))
legend("bottomleft", legend=c("Good hist", "Poor hist"),
    col=c("blue", "red"), lty=1, lwd=2, bty="n")

col.vec.lung <- as.numeric(dat.test$lung)
col.vec.lung[which(col.vec.lung==1)] <- "#0000FF66"
col.vec.lung[which(col.vec.lung==2)] <- "#FF000066"

# probability plots yes versus no lung metastases

# all but node size tuned
obj3 <- matplot(t(test.pred.unr.all.but), type="l",
    lwd=1, lty=1, col=col.vec.lung, xaxt="n",
    ylim=c(0,1), ylab="Survival probability",
    xlab="Time(years since surgery)",
    main="All but node size tuned")
axis(side=1, at=seq(0,350, by=50), labels=seq(0,7))
legend("bottomleft", legend=c("No lung mets", "Lung mets"),
    col=c("blue", "red"), lty=1, lwd=2, bty="n")

# individual survival probabilities lung: everything tuned
obj3 <- matplot(t(test.preds), type="l",
    lwd=1, lty=1, col=col.vec.lung, xaxt="n",
    ylim=c(0,1), ylab="Survival probability",
    xlab="Time(years since surgery)",
    main="All parameters tuned")
axis(side=1, at=seq(0,350, by=50), labels=seq(0,7))
legend("bottomleft", legend=c("No lung mets", "Lung mets"),
    col=c("blue", "red"), lty=1, lwd=2, bty="n")

# individual survival probabilities lung: nothing tuned
obj <- matplot(t(test.pred.unr), type="l",
    lwd=1, lty=1, col=col.vec.lung, ylim=c(0,1),
    ylab="Survival probability",
    xlab="Time(years since surgery)"
xaxt="n", main="Untuned")

axis(side=1, at=seq(0,350,by=50), labels=seq(0,7))

legend("bottomleft", legend=c("No_lung_mets", "Lung_mets"),
       col=c("blue", "red"), lty=1, lwd=2, bty="n")

# individual survival probabilities lung: cox
obj4 <- matplot(t(preds.eur.b), type="l", lwd=1,
                 lty=1, col=col.vec.lung, xaxt="n",
                 ylim=c(0,1), ylab="Survival probability",
                 xlab="Time(years since surgery)",
                 main="Cox model")

axis(side=1, at=seq(0,650,by=81.25), labels=seq(0,8))
legend("bottomleft", legend=c("No_lung_mets", "Lung_mets"),
       col=c("blue", "red"), lty=1, lwd=2, bty="n")

# Variable importance plots: all tuned, all but tuned, non tuned
rfs.final <- rfs src(Surv(time, stat) ~ Hist + Age + Volume + Ex +
                      Sex + lung + other, data = dat.train,
                      nsplit = 3, splitrule="logrank",
                      mtry=2,nodesize=200,nodedepth=7,
                      tree.err = TRUE,importance = TRUE, ntree=750)

final.rand.test.all.but <- rfs src(Surv(time, stat) ~
                                   Hist + Age + Volume + Ex +
                                   Sex + lung + other, data = dat.train,
                                   nsplit = 3, splitrule="logrank",
                                   mtry=2,nodesize=NULL,nodedepth=7,
                                   tree.err = TRUE,importance = TRUE, ntree=750)

final.rand.test.unr <- rfs src(Surv(time, stat) ~
                                  Hist + Age + Volume + Ex +
                                  Sex + lung + other, data = dat.train,
                                  nsplit=0,mtry=3,nodesize=NULL, nodedepth=NULL,
                                  tree.err = TRUE,importance = TRUE, ntree=750)

#sort colours according to neural network
rain <- rainbow(7, s=0.5)

objet <- var.select.rfs src(rfs.final)
varimp <- objet$varselect
varimp <- varimp[6,4,2,5,7,1,3,]
labs <- rownames(varimp)
labs <- c(labs[1:5], "Lung", "Other")

barplot(varimp[,2], ylim=c(0.00,0.07), names.arg=labs, col=rain,
         ylab="vimp")

361
```r
372 barplot(varimp[,1], ylim=c(0,3.5), names.arg= labs, col=rain,
            ylab="minimal depth")
374
375 objet2 <- var.select.rfsr(final.rand.test.all.but)
376 varimp2 <- objet2$varselect
377 varimp2 <- varimp2[6,3,2,5,7,1,4,]
378 labs2 <- rownames(varimp2)
379 labs2 <- c(labs[1:5], "Lung", "Other")
380 barplot(varimp2[,2], ylim=c(0.00,0.07), names.arg= labs, col=rain,
            ylab="vimp")
382 barplot(varimp2[,1], ylim=c(0,3.5), names.arg= labs, col=rain,
            ylab="minimal depth")
384
385 objet3 <- var.select.rfsr(final.rand.test.unr)
386 varimp3 <- objet3$varselect
387 varimp3 <- varimp3[4,3,2,6,7,1,5,]
388 labs3 <- rownames(varimp3)
389 labs3 <- c(labs[1:5], "Lung", "Other")
390 barplot(varimp3[,2], ylim=c(0.00,0.07), names.arg= labs, col=rain,
            ylab="vimp")
392 barplot(varimp3[,1], ylim=c(0,3.5), names.arg= labs, col=rain,
            ylab="minimal depth")
```
D.4 Additional cross-validation plots

Figure D.1: Cross-validated performance measures for the number of random split-points (\textit{nsplit}) and node depth (\textit{nodedepth}). a) concordance index, b) brier score, c) Kullback-Leibler score. For 0, 1, 2, and 3 random split points, for 1-15 node depth, in steps of 2.
Figure D.2: Cross-validated performance measures for the number of random split-points (\( nsplit \)) and node depth (\( nodedepth \)). a) concordance index, b) brier score, c) Kullback-Leibler score. For 4 and 5 candidate random split points, for 1-15 node depths, in steps of 2.
Appendix E

Simulation

E.1 Data simulation

E.1.1 Covariate information from original EURAMOS osteosarcoma data

```r
# loading data and selecting all patients except those
# non-randomized due to disease
# product is data.frame var.set.s, with all seven covariates
# in correct format

data <- read.csv("data.csv")
data$dob <- as.Date(data$dob)
data$doreg <- as.Date(data$doreg)
data$doibiop <- as.Date(data$doibiop)
data$doprc <- as.Date(data$doprc)
data$dosurg <- as.Date(data$dosurg)
data$dorand <- as.Date(data$dorand)
data$dopoc <- as.Date(data$dopoc)
data$dolc <- as.Date(data$dolc)
data$doevent[1:4] <- NA

data$doevent <- as.Date(data$doevent)
data$dofu <- as.Date(data$dofu)
data$doevent_1[1:4] <- NA

data$doevent_1 <- as.Date(data$doevent_1)

# Selecting data for covariate distributions
inf.not.rand.w <- which(data$ranin=="Progression of metastatic disease or new metastatic disease" |
data$ranin=="Unresectable disease, primary, metastatic, or both")
data.s <- data[-inf.not.rand.w]
```
age.new.a <- (data.s$doreg-data.s$dob)/365.25

# Identifying good responders
indtt.trt.g.r <- which(data.s$trt=="MAP(0, good.responder)" | data.s$trt=="MAPIE(0, good.responder)")
indtt.hist.g <- which(data.s$hist=="Good, response <10% viable tumour")
indtt.good.resp <- unique(c(indtt.trt.g.r, indtt.hist.g))

# Identifying poor responders
indtt.trt.p.r <- which(data.s$trt=="MAP(0, poor.responder)" | data.s$trt=="MAPIE(0, poor.responder)")
indtt.hist.p <- which(data.s$hist=="Poor, response >=10% viable tumour")
indtt.poor.resp <- unique(c(indtt.trt.p.r, indtt.hist.p))

# Making variable for yes/no randomized
RANDdtt <- rep(0, nrow(data.s))
RANDdtt[which(data.s$trt=="Not randomised")]<- 1
RANDdtt <- as.factor(RANDdtt)
levels(RANDdtt) <- c("Randomised", "Not randomised")

## misassigned randomisation for good/poor responders
trouble.cases.tt <- unique(c(indtt.good.resp[which(indtt.good.resp %in% indtt.poor.resp)], indtt.poor.resp[which(indtt.poor.resp %in% indtt.good.resp)]))

# 5 good responders randomised as poor responders
indtt.dat.hresp.good <- which(data.s$hist=="Good, response <10% viable tumour")

# 7 poor responders randomised as good responders
indtt.dat.hresp.bad <- which(data.s$hist=="Poor, response >=10% viable tumour")

# Making variable with correct assignments
indtt.miss.good <- indtt.dat.hresp.good[which(whic(h( data.s$hist=="Good, response <10% viable tumour") %in% trouble.cases.tt))]

366
```r
indtt.comp.good <- indtt.good.resp

indtt.miss.poor <- indtt.dat.hresp.bad[which(do.call(`which`,
  c(data.s$hresp1=="Poor_response", >=10\% viable_tumour" ))
  %in% trouble.cases.tt)]

indtt.comp.poor <- indtt.poor.resp

indtt.comp.good <- indtt.comp.good[-which(indtt.comp.good
  %in% indtt.miss.poor)]

indtt.comp.poor <- indtt.comp.poor[-which(indtt.comp.poor
  %in% indtt.miss.good)]

indttex.GOOD.POOR.t <- seq(1, nrow(data.s))

indttex.known <- which(indttex.GOOD.POOR.t %in%
  c(indtt.comp.good, indtt.comp.poor))

indttex.unkown <- indttex.GOOD.POOR.t[-indttex.known]

indttex.GOOD.POOR.t[indtt.comp.good] <- 0

indttex.GOOD.POOR.t[indtt.comp.poor] <- 1

indttex.GOOD.POOR.t[indttex.unkown] <- NA

GOOD.POOR.t <- as.factor(indttex.GOOD.POOR.t)

levels(GOOD.POOR.t) <- c("Good_hist", "Poor_hist", NA)

new.excis <- vector("numeric", nrow(data.s))

new.excis[which(data.s$excis1=="Marginal")]
  <- "Marginal"

new.excis[which(data.s$excis1=="Intralional")]
  <- "Other"

new.excis[which(data.s$excis1==""
  | data.s$excis1=="Not_known")]
  <- "Other"

new.excis[which(data.s$excis1=="Radical" |
  data.s$excis1=="Wide")]
  <- "Radical/wide"

new.excis <- as.factor(new.excis)

new.excis <- as.data.frame(new.excis)

new.lmetreg <- as.character(data.s$lmetreg)

ind.lmet.comb <- which(data.s$lmetreg=="Yes" |
  data.s$lmetreg=="Possible")

new.lmetreg[ind.lmet.comb] <- "Yes/Possible"

new.lmetreg <- as.factor(new.lmetreg)

new.lmetreg <- as.data.frame(new.lmetreg)

new.ometreg <- as.character(data.s$ometreg)

ind.omet.comb <- which(data.s$ometreg=="Yes" |
  data.s$ometreg=="Possible")

new.ometreg[ind.omet.comb] <- "Yes/Possible"

new.ometreg <- as.factor(new.ometreg)

new.ometreg <- as.data.frame(new.ometreg)
```

367
sex <- as.data.frame(data.s$sex)
volume <- as.data.frame(data.s$avol)

#age.new <- (data.s$dosurg-data.s$dob)/365.25

var.set.s <- cbind(GOOD.POOR.t, age.new.a, volume,
                  new.excis, sex, new.lmetreg,
                  new.ometreg)

colnames(var.set.s) <- c("GOOD.POOR.t", "age.new.a",
                      "volume", "new.excis", "sex",
                      "new.lmetreg", "new.ometreg")

# To make full use of available data, missing values for
# Volume and histological response are imputed. Values for
# volume and age that exceed the bounds, 5–25 for age,
# 0.005–450 for volume, and imputed. Bounds are set within the
# imputation process to ensure appropriate values for age and
# volume

# setting extreme values for age and volume to NA > to be imputed
mod.set <- var.set.s
mod.set$age.new.a[which(mod.set$age.new.a>=5 &
                      mod.set$age.new.a<=25)] <- NA
mod.set$volume[which(mod.set$volume<=450)] <- NA
mod.set$age.new.a <- as.numeric(mod.set$age.new.a)

# setting bounds for age and volume imputations
bounds <- as.matrix(rbind(c(3,0.0052000,450),c(2.5,25)))

Completed_data<-amelia(mod.set, m=10,p2s=0,
noms=c("GOOD.POOR.t", "new.excis", "sex", "new.lmetreg",
       "new.ometreg"), bounds=bounds)

imps <- Completed_data$imputations

# taking average prediction for good poor hist response for filling in
good.poor.mat <- cbind(imps[[1]]$GOOD.POOR.t, imps[[2]]$GOOD.POOR.t,
                      imps[[3]]$GOOD.POOR.t, imps[[4]]$GOOD.POOR.t,
                      imps[[5]]$GOOD.POOR.t, imps[[6]]$GOOD.POOR.t,
                      imps[[7]]$GOOD.POOR.t, imps[[8]]$GOOD.POOR.t,
                      imps[[9]]$GOOD.POOR.t, imps[[10]]$GOOD.POOR.t)
means <- rowMeans(good.poor.mat)
means <- round(means, 0)
levels(means) <- c("Good_hist", "Poor_hist")
GOOD.POOR.t <- means

mod.set.2 <- mod.set
mod.set.2$GOOD.POOR.t <- GOOD.POOR.t

# imputation matrices for volume and age
vol.mat <- cbind(imps[[1]]$volume, imps[[2]]$volume,
                 imps[[3]]$volume, imps[[4]]$volume,
                 imps[[5]]$volume, imps[[6]]$volume,
                 imps[[7]]$volume, imps[[8]]$volume,
                 imps[[9]]$volume, imps[[10]]$volume)

age.mat <- cbind(imps[[1]]$age.new.a, imps[[2]]$age.new.a,
                 imps[[3]]$age.new.a, imps[[4]]$age.new.a,
                 imps[[5]]$age.new.a, imps[[6]]$age.new.a,
                 imps[[7]]$age.new.a, imps[[8]]$age.new.a,
                 imps[[9]]$age.new.a, imps[[10]]$age.new.a)

# overall correlation between age and volume before imputation
comp.cases <- mod.set[complete.cases(mod.set),]
cor(comp.cases$volume, comp.cases$age.new.a)

# overall correlation between age and volume after imputation
vcovs.f <- lapply(1:10, function(i) cov(cbind(vol.mat[,i], age.mat[,i])))
m <- 10
withinVAR.f <- Reduce(’+’, vcovs.f)/m
vol.means.f <- colMeans(vol.mat)
age.means.f <- colMeans(age.mat)

betweenVAR.f <- as.matrix(rbind(c(var(vol.means.f), 0),
c(0, var(age.means.f))))
totalVAR.f <- withinVAR.f + ((m+1)/m)*betweenVAR.f
corr <- totalVAR.f[1,2]/(sqrt(totalVAR.f[1,1])*sqrt(totalVAR.f[2,2]))

# Function for calculating category appropriate means and
# covariance matrices for age and volume
f.g.set <- function(index){

369
# selecting relevant entries of imputation matrices
vols ← vol.mat[index,]
age.s ← age.mat[index,]

# extracting covariance matrices for age/volume for each
# imputed dataset
vcovs ← lapply(1:10, function(i) cov(cbind(vols[,i],age.s[,i])))

# within variance is calculated as the mean over the imputed
data sets
m ← 10
withinVAR ← Reduce(’+’, vcovs)/m

# between var is calculated
vol.means ← colMeans(vols)
age.means ← colMeans(age.s)

betweenVAR ← as.matrix(rbind(c(var(vol.means),0),
                           c(0,var(age.means))))

# variance matrix of subset of entries for volume and age
totalVAR ← withinVAR + ((m+1)/m)*betweenVAR

# mean for subset of entries for volume and age
big.vol.mean ← mean(vols)
big.age.mean ← mean(age.s)

tot.MEANS ← c(big.vol.mean, big.age.mean)

ret.vec ← rbind(tot.MEANS, totalVAR)
return(ret.vec)

# selecting categorical variables
mod.set.3 ← mod.set.2[,c(1,4,5,6,7)]

# counting category combinations, with frequency >=20
pst ← paste(mod.set.3$GOOD.POOR.t, mod.set.3$new.excis,mod.set.3$sex,
             mod.set.3$new.1metreg,mod.set.3$new.ometreg,sep=””)
tt ← as.data.frame(table(pst))
bb ← tt[which(tt$Freq>=20),]

# obtaining combinations proportions, plus matching means
# and variances for volume and age
means <- matrix(NA, nrow=nrow(bb), ncol=2)
vars <- list()
prop <- vector("numeric", nrow(bb))
count <- vector("numeric", nrow(bb))

for(i in 1:nrow(bb)){
  index <- which(pst==bb[i,1])
  prop[i] <- length(index)/sum(bb$Freq)
  count[i] <- length(index)
  obj <- f.g.set(index)
  means[i,1:2] <- obj[1,]
  vars[[i]] <- obj[2:3,]
}

# vector of possible combinations
sample.list <- as.vector(bb$pst)

# creating matrix of possible combinations
l1 <- c(rep("Good_hist",10), rep("Poor_hist",8))
l2 <- c(rep("Marginal",3), rep("Other",3), rep("Radical/wide",4),
       rep("Marginal",2), rep("Other",2), rep("Radical/wide",4))
l3 <- c("Female", rep("Male",2), "Female", rep("Male",2),
       "Female", "Male", "Female", "Male", rep("Female",2),
       rep("Female",2), rep("Male",2))
l4 <- c(rep("No",2), "Yes/Possible", rep("No",2),"Yes/Possible",
       "No", "Yes/Possible", "No", "Yes/Possible", rep("No",5),
       "Yes/Possible", "No", "Yes/Possible")
l5 <- c(rep("No",18))

# making sample dat, which is a dataframe of the relevant
# possible combinations, with one column per variable
sample.dat <- cbind(l1,l2,l3,l4,l5)
colnames(sample.dat) <- c("Hist", "Ex", "Sex", "lung", "other")
sample.dat <- as.data.frame(sample.dat)
sample.dat$Ex <- relevel(sample.dat$Ex, "Radical/wide")

# making figure 6.4
sum.rep <- as.data.frame(cbind(count,round(prop,3),
                        sample.dat,round(means,1)))
colnames(sum.rep) <- c("count", "prop", "Hist", "Ex", "Sex", "Lung",
                      "Other", "Volume", "Age")
sum.rep <- sum.rep[order(count, decreasing=TRUE),]
rownames(sum.rep) <- NULL

# removing 'other mets' from data, as in all of the combinations
# only "No other metastases" was present.
sample.dat <- sample.dat[,1:4]

E.1.2 Simulation of covariate data

# function for simulating the data
simulation <- function(NN){

  # NN observations are drawn from sample.list
  sim.dat <- sample(sample.list,NN, replace=TRUE, prop)
  new.data <- NULL

  for (i in 1:length(sample.list)){
    # number of times combination occurs (N)
    index.dat <- which(sim.dat==sample.list[i])
    N <- length(index.dat)

    # relevant sample.dat entry selected and repeated N times
    new.dat <- rep(sample.dat[i,],N)

    # Selecting relevant means and covariance matrix
    index.m.v <- i
    vv <- vars[[index.m.v]]
    mm <- means[[index.m.v]]

    # values for volume and age are drawn from a multivariate
    # normal distribution.
    sims <- as.data.frame(mvrnorm(N,mm, vv))
    sims[which(sims[,1]<0),] <- 0.0052
    new.dat <- cbind(new.dat,sims)
    new.data <- rbind(new.data,new.dat)
  }
  return(new.data)
}

E.1.3 Simulation of survival and censoring times

  # loading data and selecting all patients except those
  # non–randomized due to disease, those with no surgery datee,
  # and those without follow–up information.
  # product is data.frame compp, with all seven covariates
# in correct format, along with survival time and status.

```r
data <- read.csv("data.csv")
data$dob <- as.Date(data$dob)
data$doreg <- as.Date(data$doreg)
data$dobiop <- as.Date(data$dobiop)
data$doprc <- as.Date(data$doprc)
data$dosurg <- as.Date(data$dosurg)
data$dorand <- as.Date(data$dorand)
data$dopoc <- as.Date(data$dopoc)
data$dolc <- as.Date(data$dolc)
data$doevent[1:4] <- NA
data$doevent[0x0][1:4] <- NA
inf.surg <- which(is.na(data$dosurg))
# no follow-up
inf.na <- which(is.na(data$dofu) & is.na(data$locrec) &
    is.na(data$locrec_1))
# disease-dependent non-randomization
inf.not.rand.w <- which(data$ranin==
    "Progression of metastatic disease or new metastatic disease" |
    data$ranin=="Unresectable disease, primary, metastatic or both")
ind.tot.al.rem <- unique(c( inf.surg, inf.na, inf.not.rand.w))
data.p <- data[-ind.tot.al.rem,]

# Identifying good responders: information available
# from treatment, and from hresp1
indtt.trt.g.r <- which(data.p$trt=="MAP (good responder)" |
    data.p$trt=="MAPIFN (good responder)"")
indtt.hist.g <- which(data.p$hresp1==
    "Good response, <10% viable tumour")
indtt.good.resp <- unique(c(indtt.trt.g.r, indtt.hist.g))

# Identifying poor responders
indtt.trt.p.r <- which(data.p$trt=="MAP (poor responder)" |
    data.p$trt=="MAPIE (poor responder)"")
indtt.hist.p <- which(data.p$hresp1==
    "Poor response, >10% viable tumour")
indtt.poor.resp <- unique(c(indtt.trt.p.r, indtt.hist.p))

# Making variable for yes/no randomized
```

373
# misassigned randomisation for good/poor responders

```r
trouble.cases.tt <- unique(c(indtt.good.resp[which(indtt.good.resp %in% indtt.poor.resp)],
                           indtt.poor.resp[which(indtt.poorresp %in% indtt.good.resp)]))
```

# 5 good responders randomised as poor responders
```r
indtt.dat.hresp.good <- which(data.p$hresp1==
                             "Good response <10% viable tumour")
```

# 7 poor responders randomised as good responders
```r
indtt.dat.hresp.bad <- which(data.p$hresp1==
                         "Poor response >= 10% viable tumour")
```

# Making variable with correct assignments
```r
indtt.miss.good <- indtt.dat.hresp.good[which(which(data.p$hresp1==
                                                 "Good response <10% viable tumour")
              %in% trouble.cases.tt)]
indtt.comp.good <- indtt.good.resp
```

```r
indtt.miss.poor <- indtt.dat.hresp.bad[which(which(data.p$hresp1==
                                                 "Poor response >= 10% viable tumour")
              %in% trouble.cases.tt)]
indtt.comp.poor <- indtt.poor.resp
```

```r
indtt.comp.good <- indtt.comp.good[!which(indtt.comp.good %in% indtt.miss.poor)]
indtt.comp.poor <- indtt.comp.poor[!which(indtt.comp.poor %in% indtt.miss.good)]
```
age.new <- as.numeric((data.p$doreg-data.p$dob))/365.25
age.new <- as.data.frame(age.new)

new.excis <- vector("numeric", nrow(data.p))
new.excis[which(data.p$excis1=="Marginal")]<-"Marginal"
new.excis[which(data.p$excis1=="Intrallesional")]<-"Other"
new.excis[which(data.p$excis1=="Not known")]<-"Other"
new.excis <- as.factor(new.excis)
new.excis <- as.data.frame(new.excis)

volume <- data.p$avol
volume <- as.data.frame(volume)

new.lmetreg <- as.character(data.p$lmetreg)
ind.lmet.comb <- which(data.p$lmetreg=="Yes" |
                      data.p$lmetreg=="Possible")
new.lmetreg[ind.lmet.comb] <- "Yes/Possible"
new.lmetreg <- as.factor(new.lmetreg)
new.lmetreg <- as.data.frame(new.lmetreg)

new.ometreg <- as.character(data.p$ometreg)
ind.omet.comb <- which(data.p$ometreg=="Yes" |
                      data.p$ometreg=="Possible")
new.ometreg[ind.omet.comb] <- "Yes/Possible"
new.ometreg <- as.factor(new.ometreg)
new.ometreg <- as.data.frame(new.ometreg)

sex <- as.data.frame(data.p$sex)

var.set <- cbind(GOOD.POOR.t, age.new, volume,
                 new.excis, sex, new.lmetreg,
                 new.ometreg, RANDdtt)
colnames(var.set) <- c("GOOD.POOR.t", "age.new",
                   "volume", "new.excis", "sex",
                   "new.lmetreg", "new.ometreg", "RANDdtt")

# time of event occurrence in years since surgery
time.doevent <- (data.p$doevent - data.p$dosurg)/365.25
time.doevent1 <- (data.p$doevent1 - data.p$dosurg)/365.25
time.surv <- (data.p$dofu - data.p$dosurg)/365.25
# index variable for death as first event
death.ev1 <- which(data$p$locrec==1 | data$p$locrec==4 | data$p$locrec==5, data$p$multev==24 | data$p$multev==14 | data$p$multev==234 | data$p$multev==25 | data$p$multev==35)

# index variable for death as second event
death.ev2 <- which(data$p$locrec==1 | data$p$locrec==4 | data$p$locrec==5, data$p$multev==13 | data$p$multev==14 | data$p$multev==34 | data$p$multev==134)
death.ev2 <- death.ev2[-which(death.ev2 %in% death.ev1)]

# index variable for death due to osteosarcoma at final follow-up
death.final <- which(data$p$surv==2)
death.final <- death.final[-which(death.final %in% c(death.ev1, death.ev2))]

# event status death
stat.death <- rep(0, nrow(data.p))
stat.death[unique(c(death.ev1, death.ev2, death.final))] <- 1

# time status death
time.death <- vector("numeric", nrow(data.p))
time.death[death.ev1] <- time.doevent[death.ev1]
time.death[death.ev2] <- time.doevent1[death.ev2]
time.death[death.final] <- time.surv[death.final]

# in absence of death, the final date of follow-up is used
time.death[-c(death.ev1, death.ev2, death.final)] <-
time.surv[-c(death.ev1, death.ev2, death.final)]

# trouble case: no last follow-up: last information
# intermediate event 3.
data.p[which(is.na(time.death)),]
time.death[which(is.na(time.death))] <-
time.doevent[which(is.na(time.death))]

# trouble case: time < 0
ind.wrong <- which(time.death<0)

comp.dat.1 <- cbind(stat.death, time.death)
compp <- cbind(comp.dat.1, var.set[,1:7])
compp$\texttt{new.excis}$ ← \texttt{relevel}(compp$\texttt{new.excis}$, "Radical/wide")
compp ← compp[-ind.wrong,]

\texttt{par.dat}$ ← \texttt{compp}$

\texttt{#multiple.imputation}

Completed \texttt{data.par}$ ← \texttt{amelia}(\texttt{par.dat}$, m=10, p2s=0, 
noms=c("GOOD.POOR.t", "new.excis", 
"sex", "new.lmetreg", 
"new.ometreg"),
idvars=c("stat.death","time.death"))

\texttt{comp.dat.par}$ ← \texttt{Completed.data.par}$\$\texttt{imputations}$

\texttt{# A lognormal regression is performed on each imputed} 
\texttt{# dataset, and the results are saved in a list}

\texttt{lognorm.func}$ ← \texttt{function}(\texttt{imp.datasets})$

\texttt{lognormlist}$ ← \texttt{list()}$

\texttt{for}(\texttt{i}$ in 1:length(\texttt{imp.datasets}))$

\texttt{data}$ ← \texttt{imp.datasets}[\texttt{[i]}]$

\texttt{lognorm.obj}$ ← \texttt{survreg}(\texttt{Surv(time.death, stat.death)}~
\texttt{age.new+ sex+GOOD.POOR.t+new.excis+new.lmetreg+}
\texttt{volume, data=\texttt{data}, dist="lognormal"})

\texttt{name}$ ← \texttt{paste(\texttt{i})}$

\texttt{lognormlist}[\texttt{[\texttt{name}]}]$ ← \texttt{list(\texttt{lognorm.obj}=\texttt{lognorm.obj})}$

\texttt{return(\texttt{lognormlist})}$

\texttt{lognorm.list}$ ← \texttt{lognorm.func(\texttt{comp.dat.par})}$

\texttt{# function for pooling results}

\texttt{pool.lognorm}$ ← \texttt{function(\texttt{lognorm.list})}$

\texttt{# obtain pooled coefficients}

\texttt{coefs}$ ← \texttt{lapply}(1:10, \texttt{function}(\texttt{i})$

\texttt{c(\texttt{lognorm.list}[\texttt{[i]}]$\texttt{lognorm.obj}$\$\texttt{coefficients}$, 
\texttt{log(\texttt{as.numeric(\texttt{lognorm.list}[\texttt{[i]}]$\texttt{lognorm.obj}$\$\texttt{scale}))))}$

\texttt{)}}

\texttt{coefs}$ ← \texttt{data.frame(matrix(\texttt{unlist(\texttt{coefs}),}
\texttt{ncol=10, byrow=FALSE},}$
stringsAsFactors=FALSE)
mcoef <- apply(coefs, 1, mean)

# obtain covariance matrices
m <- length(lognorm.list)
vars <- lapply(1:10, function(i)
  (vcov(lognorm.list[[i]]$lognorm.obj)))

# obtain within, between and total variance
withinVAR <- Reduce('+', vars)/m
betweenVAR <- lapply(coefs, function(x)
  (x-mcoef) %*% t(x-mcoef))
betweenVAR <- Reduce('+', betweenVAR)/(m-1)
totalVAR <- withinVAR + ((m+1)/m)*betweenVAR

# extract pooled standard errors for coefficients
totalSE <- sqrt(diag(totalVAR))

# calculate P-values from pooled coefficients and SEs
new.pars <- cbind(mcoef, totalSE)
return(new.pars)
}

lognorm.pars <- pool.lognorm(lognorm.list)

colnames(lognorm.pars) <- c("Coef", "SE")
rownames(lognorm.pars) <- c("mu", "age", "sex(male)", "Hist(poor)", "Ex(marginal)", "Ex(other)", "Lung.met(Yes/possible)", "volume", "Log(scale)")
est.coefs <- lognorm.pars[2:8,1]
sigma <- exp(lognorm.pars[9,1])
mu <- lognorm.pars[1,1]

# Function for simulating survival time and data.
# Takes as argument data size (NN), covariate coefficients
# (est.coefs), intercept (mu), and scaling parameter (sigma)
sim.surv <- function(NN, est.coefs, mu, sigma){
  # A dataset of NN rows is simulated using the previously
  # defined function 'simulation'
sim <- simulation(NN)
# An indicator matrix is created for the categorical covariates
mat <- model.matrix(~ Sex + Hist + Ex + lung, sim[,1:4])[,−1]

# To this matrix age and volume are added
mat <- cbind(sim$V2,mat, sim$V1)
colnames(mat) <- c("Age", colnames(mat[,2:6]), "Vol")

# Survival time is calculated using equation (3.1)
Time <- exp(mu + mat %*% est.coefs + sigma*rnorm(NN,0,1))

# Censoring times are simulated from uniform distribution
# U(1,11)
C <- runif(NN,1,11)
Status <- as.numeric(Time <= C)
Time.c <- pmin(Time,C)
id <- seq(1,NN)
full <- cbind(sim, Time, Status, Time.c, id)
return(full)

E.2 Cross-validation approach and functions

E.2.1 Cox proportional hazards model

# Cross-validation function
final.cox <- function(data){
  # indicator vector for leave 10% out cross-validation
  select.ind <- sample(rep(1:100,20),nrow(data), replace=FALSE)
  unique.ind <- sort(unique(select.ind))
  data <- cbind(data,select.ind)
  CVPI <- rep(NA,nrow(data))
  pat.seq <- rep(NA,nrow(data))

  # unique times for prediction
times <- sort(unique(data$time))
  Pred.mat <- matrix(NA, nrow(data), length(times))

  # model - formula necessary
  fit <- coxph(Surv(time,stat) ~ Hist + Ex + Sex + lung +
    Volume + Age, data=data)

  # performed for each leave out testset
  for(i in 1:length(unique.ind)){
    # index testset
```
ind <- unique.ind[i]
# fitting model on all but test set
cm <- coxph(formula, data=data[data$select.ind!=ind,])
# creating model.matrix for test set
mm <- model.matrix(formula, data=data[data$select.ind==ind,])
mm <- mm[,!(colnames(mm) %in% c("(Intercept)"))]
# calculating prognostic index for test set
CVPI[data$select.ind==ind] <- mm %*% cm$coefficients
# calculating predictions for all unique times
Pred.mat[data$select.ind==ind] <- predictSurvProb(cm, data[data$select.ind==ind,], times
pat.seq[data$select.ind==ind] <- data$time[data$select.ind==ind]
]

# making dataframe with survival time, status and prognostic index
cox.dat <- as.data.frame(cbind(data$time, data$stat, CVPI))
colnames(cox.dat) <- c("surv", "stat", "pred")

# sorting prediction matrix entries on time
Pred.mat <- as.data.frame(cbind(Pred.mat, pat.seq))
Pred.mat <- Pred.mat[order(Pred.mat$pat.seq),]
Pred.mat <- Pred.mat[,1:(ncol(Pred.mat)-1)]

# calculating C-index and brier/KL scores
cox.C <- cindex_mod.r(cox.dat)
cox.BrierKL <- Brier.KL.cox(cox.dat, Pred.mat)

# calculating hazard ratio from normalized prognostic index
CVPI.norm <- (CVPI - min(CVPI))/(max(CVPI)-min(CVPI))
cox.cv <- coxph(Surv(time, stat) ~ CVPI.norm, data=cbind(data, CVPI.norm))
HR <- exp(cox.cv$coefficients)

# obtaining logrank Chisq statistic based on median
# prognostic index
group <- rep(1,nrow(data)); group[which(CVPI<=median(CVPI))] <- 2
group <- as.factor(group); temp.dat <- cbind(data, group)
LRX <- survdiff(Surv(time, stat)~group, data=temp.dat)$chisq
return(out)
```

# Brier/Kullback–Leibler score function

```
Brier.KL.cox <- function(imp, preds){
  so <- Surv(imp$surv, imp$stat)
  time <- so[,1]
  ot <- order(time)
  cens <- so[ot,2]
  time <- time[ot]
  N <- nrow(so)
  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf
  btime <- time
  survs <- t(as.matrix(preds))
  bsc <- rep(0, nrow(survs))
  bsk <- rep(0, nrow(survs))
  for (j in 1:nrow(survs)) {
    help1 <- as.integer(time <= btime[j] & cens == 1)
    help2 <- as.integer(time > btime[j])
    inb <- survs[j,]
    inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
    inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)
    bsc[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
                   (1 - survs[j,])^2 * help2 * (1/csurv[j]))
    bsk[j] <- -mean((log(1-(inb))*help1*(1/csurv) +
                               log(inb)*(1/csurv[j]) *help2))
  }
  ind.values <- seq(1,10)
  rel.vals <- vector("numeric", length(ind.values))
  for(i in 1:length(ind.values)){
    rel.vals[[i]] <- which.min(abs(btime-ind.values[i]))
  }
  Brie <- bsc[rel.vals]
  KL <- bsk[rel.vals]
  RET <- rbind(Brie,KL)
```
# C–index (see also .... SEARCH)
cindex.mod.r <-
  function (data) {
    time <- data$ surviv
    status <- data$ stat
    x <- data$ pred
    n <- length (time)

    ord <- order(time, -status)
    time <- time[ord]
    status <- status[ord]
    x <- x[ord]

    wh <- which(status == 1)
    total <- concordant <- 0

    for (i in wh) {
      for (j in ((i + 1):n)) {
        if (time[j] > time[i]) {
          total <- total + 1
          if (x[j] < x[i])
            concordant <- concordant + 1
          if (x[j] == x[i])
            concordant <- concordant + 0.5
        }
      }
    }
    return (concordant/total)
  }

E.2.2 Random Survival forests

# Cross–validation function
final.rand <- function (data) {
  # indicator vector for leave 10% out cross–validation
  select.ind <- sample(rep(1:100, 20), nrow(data), replace=FALSE)
  unique.ind <- sort(unique(select.ind))
  data <- cbind(data, select.ind)
prog.index <- rep(NA, nrow(data))
cols <- length(which(data$stat==1))
pred.list <- list(); time.list <- list(); pat.list <- list()

# performed for each leave out testset
for(i in 1:length(unique.ind)){
  # index testset
  ind <- unique.ind[i]
  # fitting model on all but testset
  rfsrc <- rfsrc(Surv(time, stat) ~
                      Hist + Age + Volume + Ex +
                      Sex + lung, data = data[data$select.ind!=ind, ],
                      tree.err = FALSE, importance = FALSE, ntree=750)
  # predicting on test.set
  pred <- predict.rfsrc(rfsrc, newdata = data[data$select.ind==ind, ])
  # extracting prediction matrix
  pred.s <- pred$survival
  used.times <- pred$time.interest

  # creating ensemble mortality index
  index <- pred$predicted
  prog.index[data$select.ind==ind] <- index

  # saving predictions (pred.list), along with prediction
  # times (used.times), and survival time (pat.list)
  time.list[[i]] <- used.times
  pred.list[[i]] <- pred.s
  pat.list[[i]] <- data$time[data$select.ind==ind]
}

# establishing location of 1 to 10 year predictions
new <- lapply(time.list, function(x) {
  c(which.min(abs(x-1)), which.min(abs(x-2)),
     which.min(abs(x-3)), which.min(abs(x-4)),
     which.min(abs(x-5)), which.min(abs(x-6)),
     which.min(abs(x-7)), which.min(abs(x-8)),
     which.min(abs(x-9)), which.min(abs(x-10))})
}

# selecting correct predictions
pred.vals <- list()
for (i in 1:length(pred.list)){
  pred.vals[[i]] <- pred.list[[i]][,new[[i]]]
}
pred.mat <- do.call("rbind", pred.vals)
# sorting predictions on patient according to true survival time
pred.mat <- as.data.frame(cbind(pred.mat,
              unlist(pat.list)))
pred.mat <- pred.mat[order(pred.mat$V11),]

# removing sorting column
pred.mat <- pred.mat[,1:10]

# making dataframe with survival time, status and prognostic index
rand.dat <- as.data.frame(cbind(data$time, data$stat, prog.index))
colnames(rand.dat) <- c("surv", "stat", "pred")

# calculating C-index and brier/KL scores
rand.C <- cindex.mod.r(rand.dat)
rand.BrierKL <- brier.function.rf(rand.dat, pred.mat)

# calculating hazard ratio from normalized prognostic index
prog.index.norm <- (prog.index - min(prog.index))/
              (max(prog.index)-min(prog.index))
rand.cv <- coxph(Surv(time, stat) ~ prog.index.norm,
              data=cbind(data, prog.index.norm))
HR <- exp(rand.cv$coefficients)

# obtaining logrank Chisq statistic based on median prognostic index
group <- rep(1,nrow(data))
group[which(prog.index<=median(prog.index))] <- 2
group <- as.factor(group); temp.dat <- cbind(data, group)
LRX <- survdiff(Surv(time, stat)~group, data=temp.dat)$chisq

out <- c(rand.C, HR, LRX, rand.BrierKL[1,],
rand.BrierKL[2,])

return(out)

# Modified Brier function for 'final.rand' cross-validation # function

brier.function.rf <- function(stat.t, pred.matrix){
# censoring distribution for entire data is obtained
object <- stat.t; so <- Surv(object$surv, object$stat)
time <- so[,1]; ot <- order(time)
cens <- so[ot,2]; time <- time[ot]; N <- nrow(so)
hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
csurv <- predict(hatcdist, times = time, type = "surv")
csurv[csurv == 0] <- Inf

# censoring distribution at times t=1,2,...,10 is obtained
index.l <- list()
for (i in 1:length(1:10)) {
  index.l[[i]] <- which.min(abs(time-i))
}
csurv.tt <- csurv[unlist(index.l)]

# times at which scores are evaluated
btime <- seq(1,10)

# predictions at times 1 to 10
survs <- t(as.matrix(pred.matrix))

bsc <- rep(0, length(btime))
bsk <- rep(0, length(btime))
for (j in 1:(length(btime))) {
  help1 <- as.integer(time <= btime[j] & cens == 1)
  help2 <- as.integer(time > btime[j])
  bsc[j] <- mean((0 - survs[j, ])^2 * help1 * (1/csurv) +
                   (1 - survs[j, ])^2 * help2 * (1/csurv.tt[j]))
  bsk[j] <- -mean((log(1-(survs[j, ]))*help1*(1/csurv) +
                   log(survs[j, ]))*(1/csurv.tt[j]) *help2))
}
RET <- rbind(bsc,bsk)
return(RET)

# Function for obtaining OOB based performance measures
new.func.rand <- function(data){
  # fitting model
  rfs.final <- rfsr(Surv(time, stat) ~
                   Hist + Age + Volume + Ex +
                   Sex + lung, data = data,
                   nsplit = 3, splitrule="logrank",
                   mtry=2,nodesize=NULL,nodedepth=7,
tree.err = TRUE, importance = TRUE, ntree=750)

# obtaining OOB survival probabilities and ensemble mortality
surv <- rfs.final$survival.oob
prog <- rfs.final$predicted.oob

# obtaining survival time and censoring status
stat.mat <- rfs.final$yvar
stat.mat <- as.data.frame(cbind(stat.mat, prog))
colnames(stat.mat) <- c("surv", "stat", "pred")

# ordering survival prediction according to survival time (!!)
order <- order(stat.mat$surv)
surv <- surv[order,]

# calculating C-index and Brier/Kullback-Leibler score
C.ind <- cindex.mod.r(stat.mat)
BR <- brier.function.r.mod(rfs.final, stat.mat, survv)

# calculating hazard ratio
pred.scores.norm <- (prog - min(prog))/
(max(prog)-min(prog))
cv <- coxph(Surv(surv,stat) ~ pred.scores.norm,
data=cbind(stat.mat,pred.scores.norm))
HR <- exp(cv$coefficients)

# obtaining logrank Chisq statistic based on median prognostic index
group <- rep(1,nrow(stat.mat));
group[which(prog<=median(prog))] <- 2
group <- as.factor(group); temp.dat <- cbind(stat.mat, group)
LRX <- survdiff(Surv(surv,stat)~group, data=temp.dat)$chisq

out <- c(C.ind, HR, LRX, BR[1,], BR[2,])
return(out)

# Brier score function used in previous function (note that
# it is the same as the one used in the cross-validation of
# chapter 4, see section 4.4.1
brier.function.r.mod <- function(rfs, stat.t, pred.matrix)
{
  object <- rfs$yvar
  so <- Surv(object$time, object$stat)
  time <- so[,1]
  ot <- order(time)
  cens <- so[ot,2]
  time <- time[ot]
  N <- nrow(so)
  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf
  death.times <- unique(time[which(cens==1)])
  csurv.t <- csurv[which(cens==1)]
  indexx <- seq(1, length(time[which(cens==1)]))[
    -which(duplicated(time[which(cens==1)]))] 
  if (length(indexx)==0){
    csurv.tt <- csurv.t
  } else {
    csurv.tt <- csurv.t[indexx]
  }
  preds <- pred.matrix
  so <- Surv(stat.t$surv, stat.t$stat)
  time <- so[,1]
  ot <- order(time)
  cens <- so[ot,2]
  time <- time[ot]
  N <- nrow(so)
  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf
  btime <- death.times
  survs <- t(as.matrix(preds))
  time.t <- time
  cens.t <- cens
\begin{verbatim}
bsc <- rep(0, length(btime))
bsk <- rep(0, length(btime))
for (j in 1:(length(btime))) {
  help1 <- as.integer(time.t <= btime[j] & cens.t == 1)
  help2 <- as.integer(time.t > btime[j])
  bsc[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
                  (1 - survs[j,])^2 * help2 * (1/csurv.tt[j]))
  inb <- survs[j,]
  inb[which(inb==1)] <- (inb[which(inb==1)] - 0.000001)
  inb[which(inb==0)] <- (inb[which(inb==0)] + 0.000001)
  bsk[j] <- -mean((log(1-(inb))*help1*(1/csurv) +
                   log(inb)*(1/csurv.tt[j]) *help2))
}
pos.v vals <- range(btime)
rel.v vals <- seq(1,floor(pos.v vals[2]),1)
ind.list <- list()
for(i in 1:length(rel.v vals)){
  ind.list[[i]] <- which.min(abs(btime-rel.v vals[[i]]))
}
inds <- unlist(ind.list)
Brie <- bsc[inds]
KL <- bsk[inds]
RET <- rbind(Brie,KL)
colnames(RET) <- round(death.times[inds],0)
return(RET)
}\end{verbatim}

\subsection*{E.2.3 Neural networks}

\begin{verbatim}
data.transformer.fin <- function(pot) {
# scaling of variables
  pot <- pot[order(pot$id),]
  pot$Age <- scale(pot$Age)
  pot$Volume <- scale(pot$Volume)
\end{verbatim}
# finding interval for each time point
alt.Int <- seq(0, 11, 1/12)
ITvec <- vector("numeric", length=nrow(pot))
for (i in 1:nrow(pot)) {
  ITvec[i] <- findInterval(pot$time[i], alt.Int)
}
pot <- cbind(pot, ITvec)

## creating indicator variables for categorical covariates
n.mat <- as.data.frame(model.matrix(~ -1 + Hist + Ex + Sex + lung, pot))
new.pot <- cbind(pot$stat, pot$id, pot$Age, pot$Volume, pot$ITvec, n.mat, pot$select.ind)
pot.1 <- split(new.pot, new.pot$id)
uns <- sort(unique(ITvec))

## function for replicating observation for the number of
## intervals in which it occurs
trans.func <- function(x){
  # Surv gives the interval of event/censoring
  # for each patient
  val <- x$Surv
  # finding index interval
  rept <- which(uns %in% val)
  # replicate patient for each occurrence
  x <- x[rep(seq_len(nrow(x)), rept),]
  # add vector of intervals
  x$ITvec <- uns[1:rept]
  # ensure that event is only indicated for the final
  # replicate
  if(1 %in% x$Stat){
    x$Stat <- 0
    x$Stat[rept] <- 1
  }
  return(x)
}
pot.t <- lapply(pot.l, trans.func)
pot.t <- do.call("rbind", pot.t)
return(pot.t)}

# Data transformation function for test data
# see also section 3.4.1 and Appendix C.1 (112–153)
# the function below is adapted to the simulated data

special.test.transformer.fin <- function(pat){
  pat <- pat[order(pat$id),]
  pat$Age <- scale(pat$Age)
  pat$Volume <- scale(pat$Volume)
  alt.Int <- seq(0,11,1/12)
  surv.it <- vector("numeric", length=nrow(pat))
  for (i in 1:nrow(pat)){
    surv.it[i] <- findInterval(pat$time[i], alt.Int)
  }
  all.poss <- unique(sort(surv.it))
  pat <- cbind(pat, surv.it)
  n.mat <- as.data.frame(model.matrix(~ -1 + Hist +
                                      Ex + Sex + lung, pat))
                       "Ex.O", "Sex.M", "L.y")
  new.pat <- cbind(pat$stat, pat$id, pat$Age, pat$Volume,
                   pat$surv.it, n.mat, pat$select.ind)
  colnames(new.pat) <- c("Stat", "id", "Age", "Volume",
                         "Surv", "Hist.G",
                         "Sex.M", "L.y", "select.ind")
  pat.l <- split(new.pat, new.pat$id)
  trans.func <- function(x){
    val <- x$Surv
    rept <- length(all.poss)
\begin{verbatim}
x <- x[rep(seq_len(nrow(x)), rept),]
x$ITvec <- all.poss
  return(x)
}

pat.t <- lapply(pat.1, trans.func)
pat.t <- do.call("rbind", pat.t)

# Brier–KL function
# See also section 3.4.2 and Appendix (7–89)
brier.function.neur.mod <- function(surv.preds, preds)
{
  # calculating censoring distribution
  so <- Surv(surv.preds$surv, surv.preds$stat)
time <- so[,1]
  ot <- order(time)
cens <- so[ot,2]
time <- time[ot]
  N <- nrow(so)

  hatcdist <- prodlim(Surv(time, cens) ~ 1, reverse = TRUE)
  csurv <- predict(hatcdist, times = time, type = "surv")
  csurv[csurv == 0] <- Inf

  btime <- time

  # transposing prediction matrix, so that predictions
  # for a given time are given by row
  survs <- t(preds)

  # unique prediction times
  btime.n <- unique(time)

  # defining empty vectors for bier score (bsc)
  # and KL score (bsk)
  bsc <- rep(0, length(btime.n))
  bsk <- rep(0, length(btime.n))

  for (j in 1:length(btime.n)) {
    # indicator vector for selecting relevant patients
    # help1 selects those individuals that have an event
\end{verbatim}
# at or before the time of interest. The predictions for
# these observations are compared
help1 <- as.integer(time <= btime.n[j] & cens == 1)
help2 <- as.integer(time > btime.n[j])

# As the log of zero cannot be taken, values of 1 and 0 need
# to be modified: values of 1 are lowered by 0.000001, values
# of 0 increased.
inb <- survs[j,]
inb[which(inb==1)] <- (inb[which(inb==1)]-0.000001)
inb[which(inb==0)] <- (inb[which(inb==0)]+0.000001)

# calculating brier scores for each unique time j,
# weighted by the censoring distribution csurv
bsc[j] <- mean((0 - survs[j,])^2 * help1 * (1/csurv) +
               (1 - survs[j,])^2 * help2 * (1/csurv[j]))

# calculating KL scores for each unique time j, weighted
# by the censoring distribution csurv. survs[j,] has
# been replaced by inb (defined above) to solve the
# problem of extreme predictions.
bsk[j] <- -mean((log(1-(inb)))*help1*(1/csurv) +
               log(inb)*(1/csurv[j]) *help2))

ind.values <- seq(1,11)
rel.vals <- vector("numeric", length(ind.values))
for(i in 1:length(ind.values)){
  rel.vals[i] <- which.min(abs((btime.n/12)-ind.values[i]))
}
Brie <- bsc[rel.vals]
KL <- bsk[rel.vals]
RET <- rbind(Brie,KL)
return(RET)

# Cross-validation function
finalneur <- function(data){
  # indicator vector for leave 10% out cross-validation
```r
select.ind <- sample(rep(1:100,20),nrow(data),
    replace=FALSE)
unique.ind <- sort(unique(select.ind))
data <- cbind(data,select.ind)
pat.seq <- rep(NA,nrow(data))

Pred.mat <- NULL

data <- cbind(data,select.ind)
data <- data[order(data$id),]

# transforming full data to training and test format
data.train <- data.transformer.fin(data)
data.test <- special.test.transformer.fin(data)

# defining variables for training
# on all folds but the test fold
stat <- statt[data.train$select.ind!=ind]
Age <- Ag[data.train$select.ind!=ind]
Volume <- Vol[data.train$select.ind!=ind]
Hist.P <- Hist.P[data.train$select.ind!=ind]
Ex.M <- ExM[data.train$select.ind!=ind]
Ex.O <- ExO[data.train$select.ind!=ind]
Sex.M <- SexM[data.train$select.ind!=ind]
Ly <- Ly[data.train$L.y; ITvecc <- data.train$ITvec
statt <- data.train$Stat

for(i in 1:length(unique.ind)){
    # index test set
    ind <- unique.ind[i]

    # defining variables for training the neural network
    # on all folds but the test fold
    stat <- statt[data.train$select.ind!=ind]
    Age <- Ag[data.train$select.ind!=ind]
    Volume <- Vol[data.train$select.ind!=ind]
    Hist.P <- Hist.P[data.train$select.ind!=ind]
    Ex.M <- ExM[data.train$select.ind!=ind]
    Ex.O <- ExO[data.train$select.ind!=ind]
    Sex.M <- SexM[data.train$select.ind!=ind]
    Ly <- Ly[data.train$L.y; ITvecc <- data.train$ITvec
    
    # training the neural network
    net <- nnet(stat ~ Age + Volume + Hist.P + Ex.M +
        Ex.O + Sex.M + Ly + ITvec,
        size=5, maxit=2000, MaxNWts=10000, decay=0.05,
        entropy=TRUE)

    # defining the test set:
```

393
test <- data.test[data.test$select.ind==ind,]

# predicting on test.set
predict.neur <- predict(net, test, type="raw")

data frame with prediction and id in long format
coll <- as.data.frame(cbind(predict.neur, test$id))
colnames(coll) <- c("prob", "id")

# obtaining survival estimates in wide format
coll.s <- split(coll, coll$id)
coll.t <- lapply(coll.s, function(x) {
x <- cumprod(1-x$prob)
})
pred.matrix <- do.call("rbind", coll.t)
Pred.mat <- rbind(Pred.mat, pred.matrix)

}

# sorting predictions according to id
Pred.mat <- as.data.frame(Pred.mat)

# obtaining survival times (in interval format)
temp <- as.data.frame(cbind(data.train$id, data.train$Surv))
colnames(temp) <- c("id", "surv")
temp.s <- split(temp, temp$id)
surv.times <- lapply(temp.s, function(x){a <- x$surv[1]
return(a)})
surv.times <- unlist(surv.times)

# Making matrix with survival interval times and status
Stat.mat <- as.data.frame(cbind(data$stat, surv.times))
colnames(Stat.mat) <- c("stat", "surv")

# calculating brier/KL scores
brieKL <- brier.function.neur.mod(Stat.mat, Pred.mat)
out <- c(brieKL[1,], brieKL[2,])

return(out)
E.2.4 Performing the simulation

```r
# loading relevant packages
library(survival)
library(MASS)
library(pec)
library(randomForestSRC)
library(nnet)
library(doParallel)

# creating list of 1000 simulated datasets
# with sim.surv() function
simul.data <- replicate(1000, simplify=FALSE,
    sim.surv(2000, est.coefs, mu, sigma))

# (not shown: loading all functions defined in
# sections (5.2.1-5.2.3) ... )

# defining number of cores and making cluster
cores <- 20
cl <- makeCluster(cores, type="FORK")

# applying functions and saving output
# neural network
neur.vals <- parLapply(cl, simul.data, final.neur)
save(neur.vals, file="neur.vals")

# random survival forest (cross.val)
rf.vals <- parLapply(cl, simul.data, final.rand)
save(rf.vals, file="rf.vals")

# Cox
cox.vals <- parLapply(cl, simul.data, final.cox)
save(cox.vals, file="cox.vals")

# random survival forest (OOB)
good.rand <- parLapply(cl, simul.data, new.func.rand)
save(good.rand, file="good.rand")

# terminating cluster
stopCluster(cl)
```

E.3 Results

Put in code excerpt where code is applied to same simulated dataset.

```r
# RSF simulations per performance measure
```
```r
load(file="rf.vals")
rfs.d <- do.call("rbind", rf vals)
rfs.C <- rfs.d[,1]; rfs.HR <- rfs.d[,2]
rfs.Kl <- rfs.d[,14:23]

# RSF performance measure means
rfs.C.m <- mean(rfs.C); rfs.C.sd <- sd(rfs.C)
rfs.LR.m <- mean(rfs.LR); rfs.LR.sd <- sd(rfs.LR)
rfs.HR.m <- mean(rfs.HR); rfs.HR.sd <- sd(rfs.HR)
rfs.Br.m <- colMeans(rfs.Br)
rfs.Kl.m <- colMeans(rfs.Kl)
rfs.Br.sd <- apply(rfs.Br, 2, sd)
rfs.Kl.sd <- apply(rfs.Kl, 2, sd)
rfs.Br.2 <- rfs.Br.m + rfs.Br.sd
rfs.Kl.1 <- rfs.Kl.m - rfs.Kl.sd
rfs.Kl.2 <- rfs.Kl.m + rfs.Kl.sd

# RSF C-index, LR, HR distribution, ordered by C-index
od <- order(rfs.C)
rfs.LR.ill <- rfs.LR[od]
rfs.C.ill <- rfs.C[od]
rfs.HR.ill <- rfs.HR[od]
plot(rfs.LR.ill ~ seq(1:1000), pch=16, xlab="Simulations", ylab="logrank Chi^2", col="red")
plot(rfs.C.ill ~ seq(1:1000), pch=16, xlab="Simulations", ylab="C-index", col="red")
plot(rfs.HR.ill ~ seq(1,1000), pch=16, xlab="Simulations", ylab="HR", col="red")

# Cox performance measures simulation values
load(file="cox.vals")
cox.d <- do.call("rbind", cox.vals)
cox.C <- cox.d[,1]; cox.HR <- cox.d[,2]
cox.Kl <- cox.d[,14:23]

# Cox performance measure means
cox.C.m <- mean(cox.C); cox.C.sd <- sd(cox.C)

396
```
```r
47 cox.LR.m <- mean(cox.LR); cox.LR.sd <- sd(cox.LR)
48 cox.HR.m <- mean(cox.HR); cox.HR.sd <- sd(cox.HR)
49
50 cox.Br.m <- colMeans(cox.Br)
51 cox.Kl.m <- colMeans(cox.Kl)
52 cox.Br.sd <- apply(cox.Br, 2, sd)
53 cox.Kl.sd <- apply(cox.Kl, 2, sd)
54
56 cox.Br.2 <- cox.Br.m + cox.Br.sd
57
58 cox.Kl.1 <- cox.Kl.m - cox.Kl.sd
59 cox.Kl.2 <- cox.Kl.m + cox.Kl.sd
60
61 # Cox C-index, LR, HR distribution, order to C-index
62 od <- order(cox.C)
63 cox.LR.ill <- cox.LR[od]
64 cox.C.ill <- cox.C[od]
65 cox.HR.ill <- cox.HR[od]
66
67 plot(cox.LR.ill ~ seq(1:1000), pch=16, xlab="Simulations", ylab="logrank Chi^2")
68 plot(cox.C.ill ~ seq(1:1000), pch=16, xlab="Simulations", ylab="C-index")
69 plot(cox.HR.ill ~ seq(1,1000), pch=16, xlab="Simulations", ylab="HR")
70
71 # Final results table for Cox, HR and loglik chisq
72 fin.res <- rbind(c(paste(round(cox.C.m,3),
73    paste(paste("(",round(cox.C.sd,3),sep=""), "")","sep="")
74    ), sep="")
75 c(paste(round(rfs.C.m,3),
76    paste(paste("(",round(rfs.C.sd,3),sep=""), "")","sep="")
77    )
78 c(paste(round(cox.HR.m,2),
79    paste(paste("(",round(cox.HR.sd,2),sep=""), "")","sep="")
80    )
81 c(paste(round(cox.LR.m,1),
82    paste(paste("(",round(cox.LR.sd,1),sep=""), "")","sep="")
83    ))
84
table(cox.C.ill, cox.LR.ill, cox.HR.ill)
```

397
92 \texttt{paste(round(rfs.LR.m,1),} \\
93 \quad \texttt{paste(paste("(",round(rfs.LR.sd,1),sep=""),} \\
94 \quad \quad \texttt{")",sep=""), sep="\n")}) \\
95 \texttt{colnames(fin.res) <- c("Cox", "RSF")} \\
96 \texttt{rownames(fin.res) <- c("C-index", "HR", "logrank\_Chisq")} \\
97 \\
98 \# Neural performance measures simulation values \\
99 \texttt{load(file="neur.vals")} \\
100 \texttt{neur.d <- do.call("rbind", neur.vals)} \\
102 \\
103 \# Neural performance measures means \\
104 \texttt{neur.Br.m <- colMeans(neur.Br)} \\
105 \texttt{neur.Kl.m <- colMeans(neur.Kl)} \\
106 \texttt{neur.Br.sd <- apply(neur.Br,2, sd)} \\
107 \texttt{neur.Kl.sd <- apply(neur.Kl,2, sd)} \\
109 \texttt{neur.Br.2 <- neur.Br.m + neur.Br.sd} \\
110 \texttt{neur.Kl.1 <- neur.Kl.m - neur.Kl.sd} \\
111 \texttt{neur.Kl.2 <- neur.Kl.m + neur.Kl.sd} \\
112 \\
113 \# Brier plots \\
114 x <- 1:10 \\
115 \texttt{plot(neur.Br.m \sim x, type="l", ylim=c(0,0.3), col="blue", lwd=2,} \\
116 \quad \texttt{ylab=c("Brier\_score"),} \\
117 \quad \texttt{xlab="Time\_\_years\_since\_surgery", xaxt="n")} \\
118 \texttt{axis(1, at=seq(1,10), labels=seq(1,10))} \\
119 \texttt{arrows(x, neur.Br.1, x, neur.Br.2,} \\
120 \quad \texttt{length=0.05, angle=90, code=3, col="blue")} \\
121 \texttt{lines(x, cox.Br.1, x, cox.Br.2,} \\
122 \quad \texttt{length=0.05, angle=90, code=3, col="black")} \\
123 \texttt{lines(rfs.Br.1, x, type="1", col="red", lwd=2)} \\
124 \texttt{arrows(x, rfs.Br.1, x, rfs.Br.2,} \\
125 \quad \texttt{length=0.05, angle=90, code=3, col="red")} \\
126 \texttt{legend("bottomright", legend=c("Cox", "Neural", "RSF"),} \\
127 \quad \texttt{col=c("black", "blue", "red")},} \\
128 \quad \texttt{bty="n", lty=c(1,1,1), lwd=c(3,3,3))} \\
129 \\
130 \# Kullback–Leibler plots \\
131 x <- 1:10 \\
132 \texttt{plot(neur.Kl.m \sim x, type="l", ylim=c(0,0.9), col="blue", lwd=2,} \\
133 \quad \texttt{ylab=c("Kullback–Leibler\_score"),} \\
134 \quad \texttt{xlab="Time\_\_years\_since\_surgery", xaxt="n")} \\
135 \\
398
137 axis(1, at=seq(1,10), labels=seq(1,10))
138 arrows(x, neur.Kl.1, x, neur.Kl.2, length=0.05, angle=90, code=3, col="blue")
139 lines(cox.Kl.m ~ x, type="l", col="black", lwd=2)
140 arrows(x, cox.Kl.1, x, cox.Kl.2, length=0.05, angle=90, code=3, col="black")
141 lines(rfs.Kl.m ~ x, type="l", col="red", lwd=2)
142 arrows(x, rfs.Kl.1, x, rfs.Kl.2, length=0.05, angle=90, code=3, col="red")
143 legend("bottomright", legend=c("Cox", "Neural", "RSF"), col=c("black", "blue", "red"), bty="n", lty=c(1,1,1), lwd=c(3,3,3))

# Taking a closer look at distribution neural
# network scores

# Table range Kl scores
range.Kl.n <- apply(neur.Kl,2,range)
range.Kl.n <- round(range.Kl.n,3)
colnames(range.Kl.n) <- seq(1,10)
rownames(range.Kl.n) <- c("minimum", "maximum")

high.neur <- neur.Kl[which(neur.Kl[,5] > 0.69),]
nrow(high.neur) #257

other.neur <- neur.Kl[which(neur.Kl[,5] > 0.69),]
nrow(other.neur) #743

high.neur.m <- colMeans(high.neur)
high.neur.sd <- apply(high.neur,2,sd)
high.neur.1 <- high.neur.m - high.neur.sd
high.neur.2 <- high.neur.m + high.neur.sd

other.neur.m <- colMeans(other.neur)
other.neur.sd <- apply(other.neur,2,sd)
other.neur.1 <- other.neur.m - other.neur.sd
other.neur.2 <- other.neur.m + other.neur.sd

x <- 1:10
plot(other.neur.m ~ x, type="l", ylim=c(0,0.9), col="blue", lwd=2, ylab=c("Kullback-Leibler score"), xlab="Time (years since surgery)", xaxt="n")
```r
axis(1, at=seq(1,10), labels=seq(1,10))
```
```r
arrows(x, other.neur.1, x, other.neur.2,
    length=0.05, angle=90, code=3, col="blue")
```
```r
lines(high.neur.m ~ x, type="l", col="blue", lwd=2, lty=2)
```
```r
arrows(x, high.neur.1, x, high.neur.2,
    length=0.05, angle=90, code=3, col="blue")
```
```r
lines(cox.Kl.m ~ x, type="l", col="black", lwd=2)
```
```r
arrows(x, cox.Kl.1, x, cox.Kl.2,
    length=0.05, angle=90, code=3, col="black")
```
```r
lines(rfs.Kl.m ~ x, type="l", col="red", lwd=2)
```
```r
arrows(x, rfs.Kl.1, x, rfs.Kl.2,
    length=0.05, angle=90, code=3, col="red")
```
```r
legend("bottomright", legend=c("Cox", "Neural_{high}", "Neural_{low}", "RSF"),
    col=c("black", "blue", "blue", "red"),
    bty="n", lty=c(1,2,1,1), lwd=c(3,3,3,3))
```
```r
highB.neur <- neur.Br[which(neur.Kl[,5] > 0.69),]
nrow(highB.neur) #257
```
```r
otherB.neur <- neur.Br[-which(neur.Kl[,5] > 0.69),]
nrow(otherB.neur) #743
```
```r
highB.neur.m <- colMeans(highB.neur)
highB.neur.sd <- apply(highB.neur, 2, sd)
highB.neur.1 <- highB.neur.m - highB.neur.sd
highB.neur.2 <- highB.neur.m + highB.neur.sd
```
```r
otherB.neur.m <- colMeans(otherB.neur)
otherB.neur.sd <- apply(otherB.neur, 2, sd)
otherB.neur.1 <- otherB.neur.m - otherB.neur.sd
otherB.neur.2 <- otherB.neur.m + otherB.neur.sd
```
```r
x <- 1:10
plot(otherB.neur.m ~ x, type="l", ylim=c(0, 0.3),
    col="blue", lwd=2,
    ylab="Brier score",
    xlab="Time (years since surgery)", xaxt="n")
axis(1, at=seq(1,10), labels=seq(1,10))
arrows(x, otherB.neur.1, x, otherB.neur.2,
    length=0.05, angle=90, code=3, col="blue")
```
lines(highB.neur.m ~ x, type="l", col="blue", lwd=2, lty=2)
arrows(x, highB.neur.1, x, highB.neur.2,
length=0.05, angle=90, code=3, col="blue")
lines(cox.Br.m ~ x, type="l", col="black", lwd=2)
arrows(x, cox.Br.1, x, cox.Br.2,
length=0.05, angle=90, code=3, col="black")
lines(rfs.Br.m ~ x, type="l", col="red", lwd=2)
arrows(x, rfs.Br.1, x, rfs.Br.2,
length=0.05, angle=90, code=3, col="red")
legend("bottomright", legend=c("Cox", "Neural\(_{\text{high}}\)",
"Neural\(_{\text{low}}\)", "RSF"),
col=c("black", "blue", "blue", "red"),
bty="n", lty=c(1,2,1,1), lwd=c(3,3,3,3))

# RF OOB estimation
load(file="good.rand")

# Splitting OOB Brier/KL value up in two sets:
# good.rand.1 containing observations with estimates for only
# 9 years, good.rand.2 for all 10 years.
inf <- lapply(good.rand, length)
inf <- unlist(inf)
ind.short <- which(inf<23)
good.rand.1 <-good.rand[ind.short]
good.rand.1 <- do.call("rbind",good.rand.1)
good.rand.2 <-good.rand[-ind.short]
good.rand.2 <- do.call("rbind",good.rand.2)

# obtaining C-index, HR, and logrank chisq for all simulations
temp.rand <- lapply(good.rand, function(x){
a <- x[1:3]
return(a)
})
temp.rand <- do.call("rbind", temp.rand)

# defining Brier and KL vectors
rfs2.C <- temp.rand[,1]; rfs2.HR <- temp.rand[,2]
rfs2.LR <- temp.rand[,3];
rfs2.Br <- good.rand.2[,4:13]
rfs2.Br.sh <- good.rand.1[,4:12]
272  rfs2.Kl <- good.rand.2[14:23]
274
275  # RSF performance measure means
276  rfs2.C.m <- mean(rfs2.C)
277  rfs2.C.sd <- sd(rfs2.C)
278  rfs2.LR.m <- mean(rfs2.LR)
279  rfs2.LR.sd <- sd(rfs2.LR)
280  rfs2.HR.m <- mean(rfs2.HR)
281  rfs2.HR.sd <- sd(rfs2.HR)
282
283  # obtaining means and sd for Brier and KL
286  rfs2.Br.m.10 <- mean(rfs2.Br[10])
287  rfs2.Kl.m.10 <- mean(rfs2.Kl[10])
288  rfs2.Br.m <- c(rfs2.Br.m, rfs2.Br.m.10)
289  rfs2.Kl.m <- c(rfs2.Kl.m, rfs2.Kl.m.10)
290
292  rfs2.Kl.sd <- apply(rbind(rfs2.Kl[1:9], rfs2.Kl.sh), 2, sd)
293  rfs2.Br.sd.10 <- sd(rfs2.Br[10])
294  rfs2.Kl.sd.10 <- sd(rfs2.Kl[10])
296  rfs2.Kl.sd <- c(rfs2.Kl.sd, rfs2.Kl.sd.10)
297
299  rfs2.Br.2 <- rfs2.Br.m + rfs2.Br.sd
300  rfs2.Kl.1 <- rfs2.Kl.m - rfs2.Kl.sd
301  rfs2.Kl.2 <- rfs2.Kl.m + rfs2.Kl.sd
302
303  # OOB RFS Kullback-Leibler plot
304  x <- 1:10
305  plot(rfs.Kl.m ~ x, type="l", ylim=c(0,0.8), col="red", lwd=2,
306       ylab="KL score", xlab="Time (years since surgery)",
307       xaxt="n")
308  axis(at=seq(1,10), 1, labels=seq(1,10))
309  arrows(x, rfs.Kl.1, x, rfs.Kl.2,
310         length=0.05, angle=90, code=3, col="red")
311  lines(rfs2.Kl.m ~ x, type="l", col="purple",lwd=2, lty=1)
312  arrows(x, rfs2.Kl.1, x, rfs2.Kl.2,
313         length=0.05, angle=90, code=3, col="purple")
314  legend("bottomright", legend=c("Cross-val", "OOB"),
315         fill=col2plasma(1:10), cex=0.7, bty="n")
\texttt{col=c("red","purple"), lty=c(1,1),}
\texttt{lwd=c(3,3), bty="n")}

# OOB RFS Brier plot
\texttt{plot(rfs.Br.m ~ x, type="l", ylim=c(0,0.3), col="red", lwd=2,}
\texttt{ylab=c("Brier score"), xlab="Time (years since surgery)",}
\texttt{xaxt="n")}
\texttt{axis(at=seq(1,10), 1, labels=seq(1,10))}
\texttt{arrows(x, rfs.Br.1, x, rfs.Br.2,}
\texttt{length=0.05, angle=90, code=3, col="red")}
\texttt{lines(rfs2.Br.m ~ x, type="l", col="purple", lwd=2, lty=1)}
\texttt{arrows(x, rfs2.Br.1, x, rfs2.Br.2,}
\texttt{length=0.05, angle=90, code=3, col="purple")}
\texttt{legend("bottomright", legend=c("Cross-val", "OOB"),}
\texttt{col=c("red","purple"), lty=c(1,1),}
\texttt{lwd=c(3,3), bty="n")}

# OOB RSF and regular RSF C−index, HR, and logrank chisq
\texttt{fin.rsf <- rbind(c(paste(round(rfs2.C.m,3),}
\texttt{paste(paste("(" ,"round(rfs2.C.sd,3),sep="" ), "")",}
\texttt{sep="" ), sep="" )",}
\texttt{paste(round(rfs.C.m,3),}
\texttt{paste(paste("(" ,"round(rfs.C.sd,3),sep="" ), "")",}
\texttt{sep="" ), sep="" )",}
\texttt{c(paste(round(rfs2.HR.m,2),}
\texttt{paste(paste("(" ,"round(rfs2.HR.sd,2),sep="" ), "")",}
\texttt{sep="" ), sep="" )",}
\texttt{paste(round(rfs2.LR.m,1),}
\texttt{paste(paste("(" ,"round(rfs2.LR.sd,1),sep="" ), "")",}
\texttt{sep="" ), sep="" )")})}
\texttt{colnames(fin.rsf) <- c("OOB(RSF)" , "Cross−val(RSF)")}
\texttt{rownames(fin.rsf) <- c("C−index", "HR", "logrank Chisq")}
Bibliography


