Optimal Scaling regression with Factor-by-Curve interactions

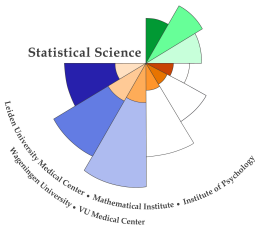
Luc Apon (s1371878)

Thesis advisors:
Sanne Willems, MSc, Dr. Anita van der Kooij
Prof. dr. Jacqueline Meulman & Dr. Elise Dusseldorp

MASTER THESIS

Defended on February 19th, 2020

Specialization: Data Science



STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

In this thesis, two regression models for the nonlinear analysis of interaction effects are proposed. The regression models are based on the Optimal Scaling methodology and specifically target the analysis of Factor-by-Curve interactions between a categorical and a continuous variable. The Optimal Scaling methodology was originally developed for analysis of categorical data, but is also applicable to continuous data. It estimates optimal quantifications for the original observed values in an iterative process by maximising the squared multiple regression coefficient (R^2), thereby transforming the original variable. These quantifications are restricted according to a prespecified scaling level, indicating the stringency of the transformation. These scaling levels can restrict the quantifications to be unsmoothed (non)monotone, or to be smooth (non)monotone. Unsmoothed nonmonotone quantifications are not restricted to any relation between the original observed values, whereas the monotone restriction preserves the ordering of the original observed values in the quantifications. The smooth restrictions are similar, but the quantifications are then also smoothed using a spline function. The quantifications can also be restricted to a linear transformation of the original observed values. This (ordinary) Optimal Scaling regression model, however, does not take into account any interaction effects between the variables.

The type of interactions considered in this thesis are the Factor-by-Curve interactions. Factor-by-Curve interactions are interactions between a categorical variable (factor) and a continuous variable. The models proposed in this thesis will be referred to as the Factor-by-Curve Optimal Scaling regression (FbC-OS-regression) models. Both models fit a separate curve for the continuous variable in the interaction for each level of the factor. For example, an interaction between a continuous variable and a factor of three levels is then fitted with three curves on that continuous variable. The difference between the two proposed models is that they either fit main and interaction effects separately or fit the joint effects in a single term.

The models are illustrated with two applications on real data. The advantage of both FbC-OS-regression models, compared to existing methods for modelling of Factor-by-Curve interactions, is that the Optimal Scaling methodology allows for monotone restrictions of the effects. This is demonstrated using the applications shown in this thesis, which are fitted using monotone spline restrictions. Results for the fitted FbC-OS-regression models are then compared to fitted linear regression models with interactions. Finally, the two approaches of modelling Factor-by-Curve interactions with OS-regression are compared to each other and to the additive model, which is a model suitable for nonlinear analysis of Factor-by-Curve interactions as well, after which suggestions for further study of the proposed models are given.

Acknowledgements

This thesis would not have been possible without the help of my supervisors. The first supervisor I would like to thank is Sanne Willems. I met with Sanne almost every week for more than half a year (seeing her more often than quite a few of my friends). These meetings were always useful and enjoyable, making for a weekly motivational boost! Additionally, her extensive comments were a huge help in making the content of this thesis readable and in the overall shaping of this study. Secondly, Anita van der Kooij has been very helpful in her reviewing of my thesis, but also in the checking of my `R` code when it did not work as needed. Finally, meetings with Jacqueline Meulman and Elise Dusseldorp were also very beneficial. Constructive tips and comments provided by Jacqueline and discussing the concepts of interaction effects with Elise both aided in achieving this final result. My gratitude to all of you for taking the time to meet, to answer all my questions and to review this thesis!

Apart from my supervisors, I would also like to express my gratitude to my family and friends, who have been very supportive from beginning to end.

Contents

Contents **3**

Notation and preliminaries **4**

1 Introduction **6**
1.1 Nonlinear data analysis 8
1.2 Interaction effects 10
1.3 Focus of this thesis 13

2 Optimal Scaling regression **14**
2.1 OS-regression model 14
2.2 Scaling levels 16
2.3 Algorithm 17
2.4 Model fit measure 20

3 Factor-by-Curve interactions in OS-regression **21**
3.1 FbC-OS-regression model 22
3.2 Algorithm 23
3.3 Implementation 26
3.4 Alternative FbC-OS-regression model 27

4 Application of FbC-OS-regression to the BCRP data **29**
4.1 Data description 29
4.2 Analysis of interaction effects 30
4.3 Interaction effects on change in depressive symptoms 32
4.4 Interaction effects on change in physical functioning 38

5 Discussion **43**
5.1 Comparison of the two FbC-OS-regression models 43
5.2 Comparison with additive models 44
5.3 Directions for future study 44
5.4 Conclusions 45

Bibliography **46**

A R Code **50**
A.1 Functions 50
A.2 Data 78
A.3 Plotting 80

Notation and preliminaries

Bold-italic uppercase letters are matrices ($\boldsymbol{Z}$), with a prime indicating their transpose ($\boldsymbol{Z}'$).
Bold-italic lowercase letters are vectors ($\boldsymbol{z}$), with a prime indicating their transpose ($\boldsymbol{z}'$).
Vectors are column vectors, unless specified otherwise.
A vector with an overhead tilde specifically indicates an unstandardised vector ($\tilde{\boldsymbol{z}}$).
A vector with an overhead ring specifically indicates a standardised vector ($\mathring{\boldsymbol{z}}$).

$\cdot$ is the symbol used for the matrix product. The symbol can be omitted to indicate the same, e.g. $\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{AB}$.
$\odot$ is the symbol used for the entrywise product.

N is the number of observations.
P is the number of predictor variables.
j and k are both indices used to indicate variables.
l is an index used to indicate categories of a variable.

C_j is the number of categories, in the case of a categorical variable j, or the number of unique observations, in the case of a continuous variable j.
C_{jk_l} is, for some factor j, continuous variable k, and factor level l, the number of categories observed for continuous variable k conditional on factor level l for factor j.

$\boldsymbol{X}$ is the (N by P) matrix containing N observations with P predictor variables.
$\boldsymbol{x}_j$ is the vector containing values for predictor variable j, i.e. it is the j'th column of matrix $\boldsymbol{X}$.
$\boldsymbol{y}$ is the vector containing outcomes of N observations.
$\hat{\boldsymbol{y}}$ is the vector containing predicted outcomes of N observations, i.e. a predictions vector.
β_j is the regression coefficient for variable j; β_0 specifically denotes the intercept.
β_{jk} is the regression coefficient for the interaction between variables j and k.
$\boldsymbol{\varepsilon}$ is the vector containing the errors.
$\boldsymbol{d}_{j_l}$ is the dummy variable for some factor j and factor level l, using ones to indicate for each entry whether the corresponding observation of factor j has factor level l.

$\boldsymbol{G}_j$ is the (N by C_j) indicator matrix of ones and zeroes for variable j, which indicates the category for that observation with a one in the corresponding column.
$\boldsymbol{G}_{jk}$ is the (N by $\sum_l C_{jk_l}$) indicator matrix of ones and zeroes for some factor j, continuous variable k, which indicates the category for that observation with a one in the corresponding column.
$\boldsymbol{G}_{jk_l}$ is the (N by C_{jk_l}) submatrix of $\boldsymbol{G}_{jk}$, for some factor j, continuous variable k, and factor level l.
$\boldsymbol{v}_j$ is the vector of quantifications for variable j, which has length C_j.

$\boldsymbol{v}_{jk}$ is the vector of quantifications for some factor j, continuous variable k, which has length $\sum_l C_{jk_l}$.
$\boldsymbol{v}_{jk_l}$ is the subvector of $\boldsymbol{v}_{jk}$, for some factor j, continuous variable k and factor level l, which has length C_{jk_l}.
$\varphi_j(\boldsymbol{x}_j)$ is the transformation of variable j, which is the same as $\boldsymbol{G}_j\boldsymbol{v}_j$.
$\varphi_{jk}(\boldsymbol{x}_k)$ is the transformed variable for the interaction effect between some factor j and continuous variable k, which is the same as $\boldsymbol{G}_{jk}\boldsymbol{v}_{jk}$.
$\varphi_o(\boldsymbol{y})$ is the transformation of the outcome variable.

$||\cdot||^2$ denotes the squared Euclidean norm.
$L(\cdot)$ denotes the loss function of a given model.
$\boldsymbol{u}_j$ is the partial residual of a given model for variable j.
$\boldsymbol{u}_{jk}$ is the partial residual of a given model for the interaction effect between variables j and k.

Chapter 1

Introduction

Data analysis can be performed in many ways, depending on the data and purpose. A very general and flexible data analytic system used in, and outside of, statistics is that of (multiple) regression analysis. Broadly speaking, regression analysis is used to analyse some variable of interest as a function of several predictor variables. Such regression models can then be used to study the extent and significance of the relation between these predictors and outcome of interest, or to forecast the outcome based on its predictor variables.

An often used regression model is the linear regression model. This relatively simple model assumes a linear relation between the outcome and predictor variables. For two predictors it can be written as

$$\boldsymbol{y} = \beta_0 + \beta_1 \boldsymbol{x}_1 + \beta_2 \boldsymbol{x}_2 + \boldsymbol{\varepsilon}, \tag{1.1}$$

where $\boldsymbol{y}$ is the outcome vector, $\boldsymbol{x}_1$ represents the first predictor as a column vector containing all observed values for observations $1, \ldots, N$, and $\boldsymbol{x}_2$ similarly represents the second predictor. Scalar β_0 denotes the intercept of the model and scalars β_1 and β_2 denote the regression coefficients for the first and second variables, respectively. Vector $\boldsymbol{\varepsilon}$ denotes the errors.

Additionally, the outcome is in some cases centered or standardised, which results in an intercept of zero, i.e. $\beta_0 = 0$. The model can then be rewritten as

$$\boldsymbol{y} = \beta_1 \boldsymbol{x}_1 + \beta_2 \boldsymbol{x}_2 + \boldsymbol{\varepsilon}. \tag{1.2}$$

The model as such can be used for a simple analysis of the outcome as a function of two predictors. While this simplicity is in some ways an advantage of the linear regression model, e.g. in terms of interpretation of its results, it also comes with limitations. The more obvious limitation is its restriction to linear relations. While the linearity assumption often leads to a decent and simple approximation of the true relationship, there are many relations that cannot be captured accurately by a linear function. In such cases, the linearity assumption is violated, and the linear regression model is unlikely to provide an accurate analysis. Take for example the relation between a driver's age and the average number of car accidents the driver is involved in. Young and old drivers are more likely to be involved in motor vehicle crashes relative to middle-aged drivers (Baker, 1992, 238-239). This results in a U-shaped relation between a driver's age and the average number of motor vehicle crashes they are involved in, which cannot properly be described using a linear function. This problem is demonstrated using simulated data in Figure 1.1. Namely, although there is a strong relation between the variables, a linear regression analysis would only indicate a weak relation. This is shown in Figure 1.1 using a nearly horizontal red line. This limitation of linear models led to a rising popularity of *non*linear analysis methods.

Another limitation of the linear regression model as given in (1.2), is that it does not take into account any interaction effects between the variables. The model defines the relation between

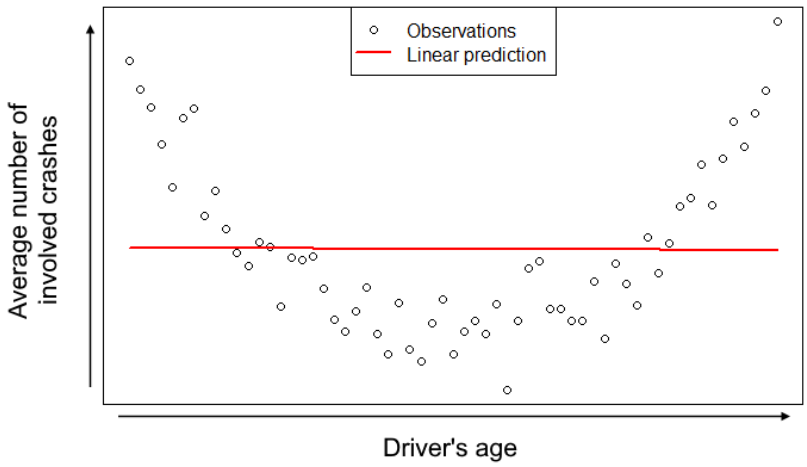
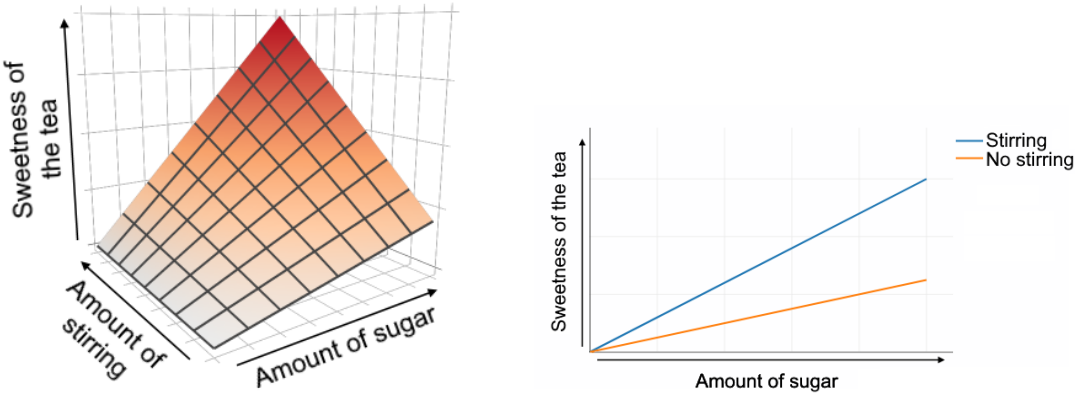


Figure 1.1: Example of a U-shaped relation between driver's age and average number of crashes the driver is involved in for each age, using simulated data, where a linear regression model is unable to adequately capture this relation.

the outcome and predictors as an additive combination of the two predictors, but this is not always sufficient. A very simple example of an interaction effect is the combined effect of both adding sugar and stirring on the sweetness of a cup of tea. Either just adding sugar or stirring will not make much of a sweet tea. Doing both, however, will. This shows that there is an interaction effect between the two actions. In such cases, an extra term is often added to the model to capture this interaction effect. Another example of an interaction, in a medical setting, is the combination effect of genetic variants associated with increased risk of diabetes type II and the Western dietary pattern (high intake of red meats, processed meats, and refined foods) in men (Qi et al., 2009). Intakes of the Western dietary pattern were only found to be significantly associated with higher diabetes risk in men with a high genetic risk score for diabetes, indicating a joint effect between the Western dietary pattern and genetic risk score variables.

Figure 1.2 further illustrates and stresses the significance of accounting for interaction effects. Namely, it shows two examples of strong interaction effects, one between two numeric variables (1.2a) and another between a numeric variable and a factor (1.2b). Both interactions use the sweet tea example as hypothetical illustration. The combination of intensive stirring and addition of sugar results in a sweet tea, while if no sugar is added, stirring does not affect the outcome at all. This is visualised in Figure 1.2a, considering both stirring and added amount of sugar as numeric variables. In the factor – numeric interaction example, whether the tea is stirred clearly affects the effect of the amount of sugar on the sweetness of the tea. If the tea is not stirred, the sugar does not dissolve as easily, hence the sweetness of the tea only slowly increases with the amount of sugar added. An additive model like the linear model in (1.2) would not be able to model these relationships adequately.

Note that, depending on the area of study, the term used to refer to interactions might differ. For example, in behavioural sciences, if a relation between some variable A and an outcome is dependent on another variable B, this is called "moderation". This is essentially an interaction between the two variables, but note that the interpretation for moderation is specifically denoted here as the effect of B on the relation between the outcome and variable A, not of variable A on the relation between the outcome and variable B. Hence B "moderates" the relation between A



(a) Numeric - numeric interaction

(b) Factor - numeric interaction

Figure 1.2: Example plots of interaction effects. (a): Increasingly reddish colour of the surface indicates a sweeter tea. Summation of individual predictor effects would be insufficient to describe the change in sweetness. (b): Sweetness of the tea varies for the amount of sugar added, but the extent of this effect depends on whether the tea is stirred or not. Differences in slopes indicate an interaction effect between these two variables.

and the outcome.

Both nonlinear relations and interactions between predictor variables are common in data. Therefore, appropriate analysis tools are necessary for data analysis. Several methods for analysis of nonlinear relations and interactions will be combined and discussed further in this thesis.

1.1 Nonlinear data analysis

Some regression analysis methods suited for analysis of nonlinear relations are available. Overall, such methods can be divided into one of three major approaches: nonlinear regression, Generalized Linear Models, and regression with transformations. Nonlinear regression avoids a linear combination of the predictors and parameters, but instead combines them in a nonlinear function. An example is the Michaelis-Menten model used to describe enzyme kinetics (Michaelis and Menten, 1913), which is defined as

$$\hat{\boldsymbol{y}} = \frac{\beta_1 \boldsymbol{x}}{\beta_2 + \boldsymbol{x}}. \tag{1.3}$$

The modelled relation here is nonlinear and nonadditive.

Generalized Linear Models (GLM) are extensions of the linear regression model (Nelder and Wedderburn, 1972; MacCullagh and Nelder, 1989). For two variables, the model can be defined as

$$g(E(y_i)) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i}. \tag{1.4}$$

Here, a (nonlinear) link function g of the linear combination of predictors models the expectation of the outcome $E(y_i)$ for each observation i. As a result, depending on the link function, the relation between outcome and predictors is no longer restricted to linearity. Additionally, while the ordinary linear model assumes normally distributed errors, using a link function allows for other distributions, making the GLM more flexible than the ordinary linear model. A variety of

link functions can be used, to accommodate for different types of data. If the link function is the identity function, for example, then the model results in the ordinary linear model, that is to say, ordinary linear regression is also a GLM.

Regression with transformations takes yet another approach. In these models, the predictors and/or outcome themselves are transformed separately from each other. The (transformed) outcome is then modelled as the linear combination of those transformed predictors. Many transformations are possible, and they are often used to improve justification of the assumptions made for regression analysis. A basic transformation could be to use the square of a variable, while a more extensive transformation is a smoothing function such as a spline transformation, which is a combination of piecewise polynomials.

1.1.1 Additive models

An example of a model using the regression with transformations approach, is the additive regression model (Friedman and Stuetzle, 1981; Stone, 1985; Hastie and Tibshirani, 1990; de Leeuw et al., 1976). It can be viewed as an extension of the linear regression model, but with transformed predictors. The model for two predictors and a standardised outcome is formulated as

$$\boldsymbol{y} = f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2) + \boldsymbol{\varepsilon}, \tag{1.5}$$

with $f \in F$ and F being the set of all possible transformation functions for continuous variables. Transformation of the predictor variables provides a more flexible model, while still using a linear combination to describe the relation between variables and outcome. The classical linear regression model can also be viewed as a type of additive model. Namely, when simply using the variable multiplied with the regression coefficient, i.e.

$$f(\boldsymbol{x}_1) = \beta_1 \boldsymbol{x}_1, \quad f(\boldsymbol{x}_2) = \beta_2 \boldsymbol{x}_2, \tag{1.6}$$

the result is the linear regression model as defined in (1.2).

Without further specifications, the additive model as given in (1.5) allows for any transformation function. Finding an optimal solution from an infinite number of possible transformation functions is not feasible. Therefore, some restrictions need to be applied to limit the possible functions. A simple example is restricting f to the function where the variable is multiplied with a coefficient (1.6), resulting in classic linear regression. Usually, however, the additive model uses nonmonotone splines as transformation functions for continuous variables. This results in a smooth linear transformation for those variables. In the case of a categorical variable, dummy variables are included in the model to replace that categorical variable itself. A regression coefficient is then estimated for each dummy variable.

1.1.2 Optimal Scaling regression

Another nonlinear regression model is Optimal Scaling regression (OS-regression) (Young et al., 1976; Gifi, 1990; van der Kooij, 2007; Meulman et al., 2019), which is the focus of this thesis. OS-regression takes the regression with transformations approach, using a linear combination of transformed predictor variables to describe the transformed outcome. The model for two variables is written as

$$\varphi_o(\boldsymbol{y}) = \beta_1 \varphi_1(\boldsymbol{x}_1) + \beta_2 \varphi_2(\boldsymbol{x}_2) + \boldsymbol{\varepsilon}, \tag{1.7}$$

with $\varphi_k(\boldsymbol{x}_k)$ indicating an optimally transformed variable k and $\varphi_o(\boldsymbol{y})$ specifically denoting the transformation for the outcome. It then optimises the linear relation between the transformed

outcome and transformed predictors by iteratively updating the regression coefficients and transformations. This OS-regression model has been implemented in SPSS in the Categories package (van der Kooij and Meulman, 1998).

Although originally designed for analysis of categorical variables, OS-regression can be used for nonlinear analysis of both categorical and numerical variables. As opposed to additive models, OS-regression does not use dummy variables for the categorical variables, but replaces them by optimal numeric values, which are called quantifications. These quantifications are optimal in the sense that they minimise the loss function of the formulated model, which is also where "Optimal" in the model name comes from. Additionally, such quantifications can be estimated for continuous variables. In this case, each unique observed value from the numerical variable is considered as a separate category. The quantifications are then estimated in the same way as for categorical variables.

The transformations for OS-regression can be restricted to preserve some properties of the original variables. The different levels of restrictions are defined by specifying a scaling level. The least restrictive scaling level is the nominal scaling level. For this scaling level, the observations are grouped according to their category in the original variable. Hence, if two observations had the same original value, they will be replaced by the same quantification in the transformed variable. This grouping property is also preserved in all other scaling levels. Another scaling levels is the ordinal scaling level, which is used to retain the ordering of categories. This means that the values of the quantifications are in monotone order, reflecting the order of the categories in the original variable. Two more scaling levels are the nonmonotone and monotone spline scaling levels. Whereas the nominal and ordinal scaling levels are step functions, both spline scaling levels are smooth continuous functions. As a result, transformed values for unobserved continuous values can also be obtained. As the name suggests, the nonmonotone spline scaling level finds nonmonotone spline transformations for the original variable, whereas the monotone spline scaling level finds monotone spline transformations. The latter specifically uses monotone regression splines as described by Ramsay (1988). Finally, the most restrictive scaling level is the numeric scaling level. This scaling level has the additional restriction of retaining the relative spacing of the values in the original variable, which essentially boils down to using the standardised values of the original variable. This scaling level thus results in a linear transformation. When restricting categorical variables using the numeric scaling level, the categories are treated as equally spaced numeric values in the given order.

As can be seen in the model formulation in (1.7), OS-regression includes a coefficient for each variable. In this way, the overall effect of each transformed variable can be quantified, tested for significance, and interpreted. While the outcome may be transformed as well, it is only standardised and not otherwise transformed in the remainder of this thesis for simplicity purposes. More details regarding the OS-regression model, scaling levels, and updating algorithm are given in chapter 2.

1.2 Interaction effects

The linear, additive, and OS-regression models as described previously do not specifically take into account any joint effects, i.e. interaction effects, of variables on the outcome. These models all use a linear combination of (transformed) predictor variables, while interaction is actually a deviation from additivity. To be able to analyse interactions, these models need to be altered or extended.

1.2.1 Multiplicative interactions

In existing models, a multiplicative term is often added to take into account interaction effects. For the ordinary linear regression model, this can be written as

$$\hat{\boldsymbol{y}} = \beta_1 \boldsymbol{x}_1 + \beta_2 \boldsymbol{x}_2 + \beta_{12} \boldsymbol{x}_1 \odot \boldsymbol{x}_2, \tag{1.8}$$

where $\odot$ is the sign for the entrywise product. While the addition of a multiplicative term may not be a suitable approach for all types of interaction effects (Southwood, 1978), it is a straightforward way of modelling a joint effect of predictor variables on the outcome. This method has met with criticism in the past, being deemed too hard to interpret and unreliable (Wright, 1976; Jackman, 1974; Smith and Sasaki, 1979). The former was a concern due to interpretation of the model coefficients; when including this multiplicative interaction term, the coefficients should be interpreted conditionally on the other variables in that interaction. The claims of unreliability of this method arose due to collinearity, since, essentially, a variable $(\boldsymbol{x}_1 \odot \boldsymbol{x}_2)$ is added that is likely to be collinear with the interacting variables ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$). However, both concerns have been disproved by Friedrich (1982) and the method is quite commonly used now.

Adding a multiplicative term for variables that are not both numeric is slightly more complicated. The concept in linear regression for an interaction between a categorical and numerical variable, for example, is essentially to estimate a separate coefficient for the numerical variable, for each level of the categorical variable. In other words, a linear relation between the continuous variable and the outcome is estimated, separately for each category of the categorical variable. This can then be considered as a multiplicative interaction when representing the categorical variable by dummy variables, with the number of dummies equal to the number of categories minus 1. The formulation for the linear model, with a single categorical variable $\boldsymbol{x}_1$ having two levels A and B and a single numerical variable $\boldsymbol{x}_2$, could then be defined as

$$\boldsymbol{y} = \beta_0 + \beta_{1_B} \boldsymbol{d}_{1_B} + \beta_2 \boldsymbol{x}_2 + \beta_{12_B} \boldsymbol{d}_{1_B} \odot \boldsymbol{x}_2 + \boldsymbol{\varepsilon}. \tag{1.9}$$

Here, $\boldsymbol{d}_{1_B}$ is a dummy variable for category B in variable $\boldsymbol{x}_1$. It is a column vector of ones and zeroes. The dummy variable for category l in categorical variable k is then, for each observation i, defined as

$$d_{k_l,i} = \left\{ \begin{array}{ll} 0, & \text{if } x_{k,i} \neq l \\ 1, & \text{if } x_{k,i} = l \end{array} \right\}. \tag{1.10}$$

A dummy variable is defined for each category in the variable, except for one reference category, which is usually the first category. In the example given above, category A is used as a reference for the other categories (in this case only category B). This can be seen from the model formula in (1.9), where only β_0 and $\beta_2 \boldsymbol{x}_2$ are used to describe the outcome if category A is observed. Separate terms for category A are therefore redundant. For each additional category in the categorical variable, however, two new terms do have to be added. One for the main effect and one for the interaction effect, each having its own coefficient. This is also why these coefficients in (1.9) specify the corresponding category, i.e. β_{1_B} and β_{12_B} instead of β_1 and β_{12}. Finally, note that Figure 1.2b is an example of a relation between variables and outcome that can be fitted using the model as formulated in (1.9).

An interaction between two categorical variables can similarly be expressed using multiplicative terms, which also requires the use of dummy variables. This then results in a fitted regression weight for each combination of two categories of the two variables, in addition to the main effects for those variables.

Both multiplicative interaction formulations, (1.8) and (1.9), can be extended to include more predictor variables, either with or without interactions, and to include interactions of a higher

order, i.e. interactions between three or more variables. Due to increased complexity of multiple variables and the uncommon use of higher order interactions, neither will be discussed in this thesis. In other words, this thesis project will only focus on the example with two predictor variables that interact with each other.

1.2.2 Factor-by-Curve interactions

Interactions between a categorical variable (factor) and a continuous variable can specifically be called Factor-by-Curve interactions, as used by Coull et al. (2001). Such interactions have also been mentioned earlier, albeit without the use of this specific term, by Hastie and Tibshirani (1990) for example. Analysis of such interactions, as the name implies, can be performed by modelling the continuous variable as a different curve for each level of the categorical variable. Linear regression is capable of analysing such interactions, when considering linear relations as linear curves. However, similar to the point that there are nonlinear relations between variable and outcome, as discussed earlier in the introduction, it can be limiting to assume a linear interaction effect. Factor-by-Curve interactions can also be fitted using *non*linear curves, thus allowing for nonlinear interaction effects.

The regression with transformations approach is suitable for the analysis of Factor-by-Curve interactions, since continuous predictors are transformed using a smooth transformation function, resulting in a curve. The same approach can then be used to separately transform the continuous variable into multiple curves, one for each category of the categorical variable. Since these curves are fitted separately, they are generally free to vary in form and direction for each category.

This nonlinear concept for interaction effects has been implemented in some models, such as in additive models and nonparametric estimation techniques (Wood, 2017; Sestelo et al., 2017). By fitting and plotting a curve for each level of a factor, this concept of analysing Factor-by-Curve interactions allows for visual comparison of the joint effects between the factor and the numeric variable. Interaction effects can then be determined by dissimilarity of the curves.

1.2.3 Interaction effects in OS-regression

Some research on extending the OS-regression model to allow for modelling of interaction effects has been performed. These studies were on two methods for modelling interactions between two categorical variables. Van der Lans and Heiser (1988) described a method based on adding a new variable with all combinations of the categories of two interacting variables. The resulting model's basic concept is similar to how interactions between two categorical variables are modelled. Van Rosmalen et al. (2009) added a multiplicative term to the model and introduced a visual representation of how the resulting interaction effects can be interpreted.

Methods for analysis of Factor-by-Curve interactions have been implemented in the earlier introduced additive model. However, it has not been priorly studied in OS-regression. Implementing it in the OS-regression model would allow for monotonic restrictions on the interaction effects themselves, which is not an option in other methods modelling Factor-by-Curve interactions. OS-regression then also allows for monotonic transformations of the individual predictors. Additionally, use of dummies can be avoided using OS-regression, by transforming categorical variables. As a result, the particular relations between categories within that variable can be modelled. Both monotone transformations and transformations of categorical variables are not available in the additive model, which uses dummies and only supports nonmonotone splines.

1.3 Focus of this thesis

The aim of this thesis project is to implement the analysis of Factor-by-Curve interactions in the existing OS-regression model to allow for monotone main and interaction effects. To this end, the OS-regression model is explained in more detail in chapter 2, after which the algorithm and implementation of the Factor-by-Curve OS-regression model is discussed in chapter 3. The Factor-by-Curve OS-regression model is then applied to a real data set in chapter 4, where the fitted models will be compared to each other and fitted linear regression models. Finally, the results and implications will be discussed in chapter 5.

Due to the specific interest in interaction effects between a categorical and continuous variable, the models and implementations throughout this thesis will adhere to the format of having one categorical variable and one continuous variable. The use of only two variables is to retain simplicity of equations and concepts, which is more clear and easy to interpret. All models and implementations discussed in this thesis can, however, be extended to models with more predictor variables and interactions. Finally, it is always assumed that the outcome variable is standardised to a mean of zero and a variance of one, i.e. $E(\boldsymbol{y}) = 0$ and $\sigma^2(\boldsymbol{y}) = 1$.

In order to illustrate some of the discussed concepts in this thesis, some very simple toy data will be used. This toy data set is shown in Table 1.1. The first variable ($\boldsymbol{x}_1$) is a categorical variable with three levels A, B, and C. The second variable ($\boldsymbol{x}_2$) is a continuous variable with integers ranging from one to four. If not specified otherwise, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ will be used in examples throughout the thesis to refer to the variables and their values as shown in Table 1.1.

Categorical variable ($\boldsymbol{x}_1$)	Continuous variable ($\boldsymbol{x}_2$)
A	1
B	2
A	4
B	3
C	1
A	2
C	3

Table 1.1: Table with the toy data variables, one is a categorical variable with three levels, A, B, and C, and the other is a continuous variable containing values ranging from one to four. Generally, continuous variables contain more (or consists entirely of) unique values, but these specific values are used for simplicity of the examples given throughout this thesis.

Chapter 2

Optimal Scaling regression

Optimal Scaling regression (OS-regression) is a method that uses a linear combination of transformed predictor variables to perform nonlinear data analysis. It originates from psychometrics, where nonlinear data analysis using optimal scaling methodology is quite extensively explored, notably in the Gifi project (Gifi, 1990), where the optimal scaling methodology is used in multiple techniques. OS-regression was originally named Categorical Regression (CATREG), since this method was designed for nonlinear regression with categorical variables, by optimally scaling the categories in qualitative data. The concept of quantifying qualitative data can be traced back, among others, to the work of Guttman (1941), which later became known as multiple correspondence analysis. Further inspiration for the Gifi project came from the nonlinear version of MONANOVA (Kruskal, 1965), a form of analysis of variance with a monotone transformation of the outcome variable, which was followed up on by various projects, such as ADDALS (de Leeuw et al., 1976), an additive model, MORALS (Young et al., 1976), a multiple regression model, and monotonic transformations to additivity using splines (Winsberg and Ramsay, 1980). Regression using a similar methodology entered the mainstream statistical literature with the paper by Breiman and Friedman (1985) under the name Alternating Conditional Expectations (ACE). The most recently developed OS-regression method (implemented in IBM SPSS Statistics) is extensively described in Meulman et al. (2019).

2.1 OS-regression model

The model for OS-regression with two predictors and disregarding any transformations of the outcome is written as

$$\boldsymbol{y} = \beta_1 \varphi_1(\boldsymbol{x}_1) + \beta_2 \varphi_2(\boldsymbol{x}_2) + \boldsymbol{\varepsilon}, \tag{2.1}$$

with $\varphi_1(\boldsymbol{x}_1)$ and $\varphi_2(\boldsymbol{x}_2)$ representing the optimally transformed variables. To fit this model, both transformations $\varphi_1(\boldsymbol{x}_1)$ and $\varphi_2(\boldsymbol{x}_2)$ and coefficients β_1 and β_2 are simultaneously optimised in each iteration. The optimisation is based on the maximisation of the squared multiple regression coefficient (R^2). The model results in transformed variables with numeric properties, which optimally describe the relation between predictor variables and outcome, while allowing for nonlinearity. It is possible to transform the outcome as well, but this option is not considered throughout this thesis and therefore left out of the OS-regression model equations. The outcome is always standardised to a mean of zero and a variance of one. As a result, no intercept is required in the model. Likewise, the transformed predictors are standardised as well.

In the OS-regression model, the transformed variable $\varphi_k(\boldsymbol{x}_k)$ for some variable k is written

in matrix form as

$$\varphi_k(\boldsymbol{x}_k) = \boldsymbol{G}_k \boldsymbol{v}_k, \tag{2.2}$$

where $\boldsymbol{G}_k$ is the indicator matrix and $\boldsymbol{v}_k$ the vector with the quantifications for each category. This indicator matrix is used to indicate which category each observation belongs to. Namely, each row consists of only zeroes, except for one column each row, where a one represents the category of that observation. Hence, indicator matrix $\boldsymbol{G}_k$ is of dimensions N by C_k, where C_k is used to indicate the number of unique categories for variable k. For example, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ for the toy data in Table 1.1 are

$$\boldsymbol{G}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{G}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.3}$$

with $N = 7$ rows and $C_1 = 3$ and $C_2 = 4$ columns, respectively. Note that the columns of an indicator matrix can be viewed as the dummy variables for each category. The indicator matrix, however, also includes a column for the category which would be used as a reference in the common use of dummy variables.

Quantifications vector $\boldsymbol{v}_k$ is defined as a column vector of length C_k containing the quantifications that are used to replace the categories for that variable. The quantification vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ for the toy data variables would be

$$\boldsymbol{v}_1 = \begin{pmatrix} v_{1_1} \\ v_{1_2} \\ v_{1_3} \end{pmatrix}, \text{ and } \boldsymbol{v}_2 = \begin{pmatrix} v_{2_1} \\ v_{2_2} \\ v_{2_3} \\ v_{2_4} \end{pmatrix}. \tag{2.4}$$

Here, v_{1_1}, v_{1_2} and v_{1_3} represent the quantifications of the first, second and third category of $\boldsymbol{x}_1$, i.e. A, B and C, respectively. Likewise, quantifications v_{2_1}, v_{2_2}, v_{2_3} and v_{2_4} represent the four categories of $\boldsymbol{x}_2$, i.e. numeric values one through four, respectively.

The matrix product of $\boldsymbol{G}_k$ and $\boldsymbol{v}_k$ then results in a column vector $\varphi_k(\boldsymbol{x}_k)$ of length N, containing the quantification for each observation. For example, transformation $\varphi_2(\boldsymbol{x}_2)$ would be a column vector of length $N = 7$, with optimal quantifications for each observation, written as

$$\varphi_2(\boldsymbol{x}_2) = \boldsymbol{G}_2 \cdot \boldsymbol{v}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} v_{2_1} \\ v_{2_2} \\ v_{2_3} \\ v_{2_4} \end{pmatrix} = \begin{pmatrix} v_{2_1} \\ v_{2_2} \\ v_{2_4} \\ v_{2_3} \\ v_{2_1} \\ v_{2_2} \\ v_{2_3} \end{pmatrix}. \tag{2.5}$$

As can be seen in this example, an indicator matrix and quantifications vector can also be constructed for a numerical variable, where each unique value is considered a category. While this numerical example has few unique values, OS-regression can also be used for variables with many categories. Such variables would result in a wider indicator matrix, i.e. more columns, and a longer quantifications vector.

Altogether, the model as written in (2.1) can be rewritten in matrix form as

$$\boldsymbol{y} = \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1 + \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2 + \boldsymbol{\varepsilon}, \tag{2.6}$$

by substituting (2.2).

2.2 Scaling levels

As mentioned earlier, OS-regression is applicable to both numerical and categorical data. It can transform categories to optimal numeric values, which are called quantifications. The quantifications replace the category values, resulting in data with numeric properties. These quantifications are restricted according to a prespecified *scaling level*. All available scaling levels are discussed below, while examples of all scaling levels are shown in Figure 2.1.

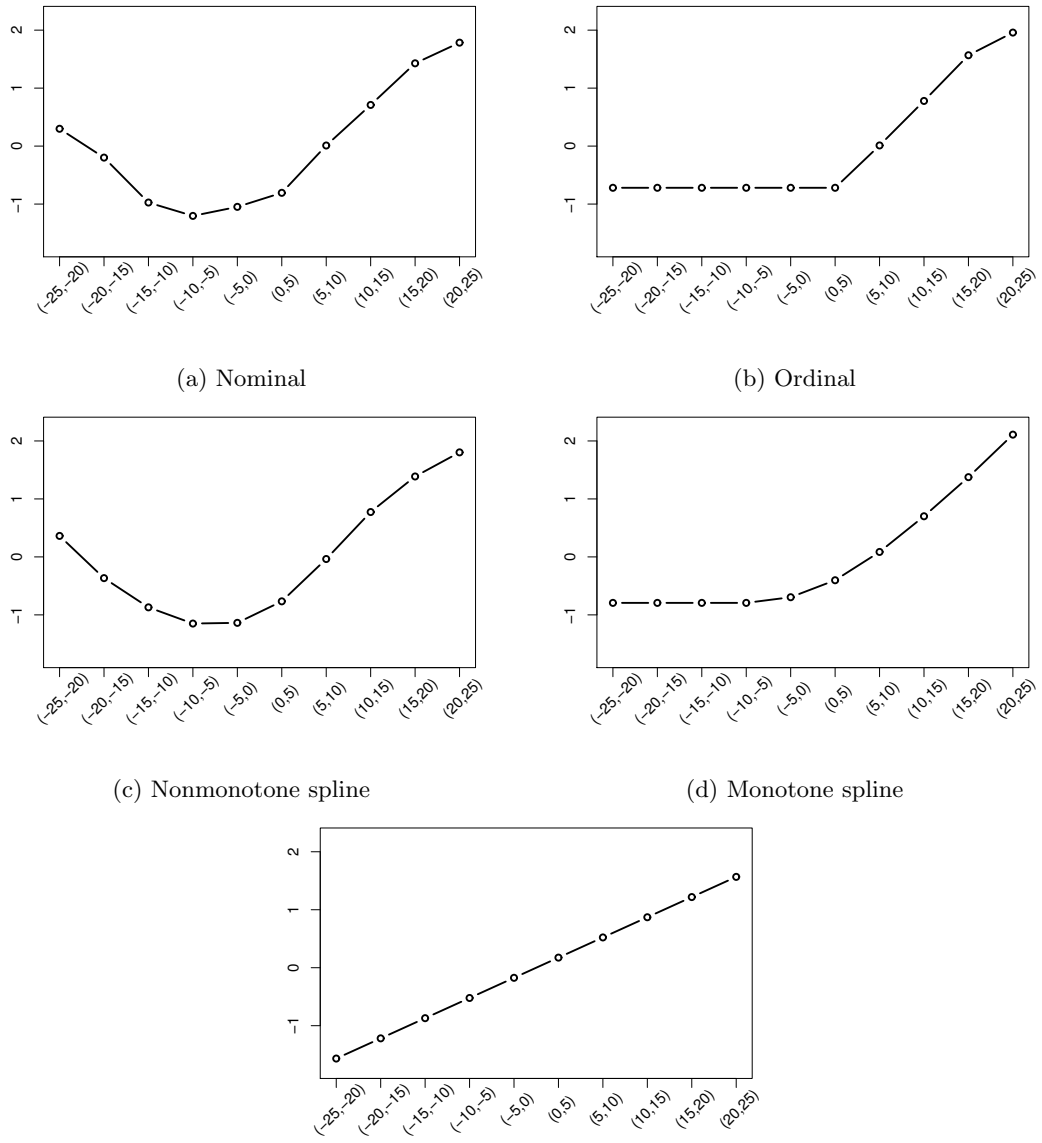
There are five scaling levels. The most basic level is the nominal scaling level, which only groups the quantifications according to the categories for the original variable. No particular relation between the quantifications is then assumed, but observations with the same categories are given the same quantification. The next scaling level is the ordinal scaling level, which assumes a specific ordering within the quantifications. An example is a scale of categories with "low", "medium" and "high". These categories imply a monotone ordering, from "low" to "high". This can also be enforced in the quantifications, with the ordinal quantifications for this example restricted to nondecreasing successive values.

These two scaling levels are not always suitable, however. Especially in cases with many categories, the nominal and ordinal scaling levels may not be restrictive enough. This could lead to overfitting and poorly interpretable results. In such cases, a smooth transformation may be more suitable.

This leads us to restrictions of the nominal and ordinal scaling levels: the nonmonotone and monotone spline scaling levels, which are smoothed versions of the nominal and ordinal scaling levels, respectively. These are more rigorous restrictions, which are more suitable for either numeric variables or categorical variables with many categories. As a rule of thumb, for a variable with eight or more categories, the spline scaling levels are preferred over the nominal and ordinal scaling levels.

As an additional note on the ordinal and monotone spline scaling levels, another restriction is enforced. A monotone transformation can result in both an increasing and decreasing function, but an increasing function is easier for interpretation. Therefore, increasing transformations are enforced by reversing the sign of both the quantifications and the coefficient if the transformation is decreasing. As a result, the overall effect remains the same, while the transformation becomes increasing

The final scaling level is the numeric scaling level. It is mostly suited for numerical variables, but can also be used for categorical variables. This scaling level restricts the order of the quantifications and the distance between them to be the same as in the original variable. This means that the variable is only linearly transformed and only position and scale, i.e. mean and variance, are affected. In practice, this means that a numerical variable scaled as numeric is only standardised and not nonlinearly transformed. An analysis using only the numerical scaling level for all variables would thus give the same results as a linear regression on the standardised variables. For categorical variables, the categories would be replaced by equally spaced numeric values and standardised.



(a) Nominal

(b) Ordinal

(c) Nonmonotone spline

(d) Monotone spline

(e) Numeric

Figure 2.1: Examples of plotted quantifications for all scaling levels. Both splines are fitted with a degree of 2 and have 2 interior knots.

2.3 Algorithm

Generally, the algorithm for OS-regression consists of first initialising the model parameters and then iteratively optimising both quantifications and coefficients for one variable at a time until some convergence criterion has been reached. This criterion involves a loss function and indicates when the algorithm considers the decrease in loss small enough to have converged. Alternatively,

if convergence takes too long, convergence can be based on a maximum number of iterations. After convergence, the algorithm outputs the results and model statistics can be determined.

For subsequent estimation equations for quantifications and coefficients, the first variable $\boldsymbol{x}_1$ is used as a precedent. Equations for the second variable, $\boldsymbol{x}_2$, can be found by swapping the subscripts 1 and 2 of the variables and coefficients in all equations.

2.3.1 Initialisation of parameters

For the OS-regression technique, several parameters need to be initialised prior to the optimisation process. Namely, indicator matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, quantification vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ and coefficients β_1 and β_2. The indicator matrices are constructed from the data, as shown for the example in (2.3). Unlike the indicator matrix consisting of solely ones and zeroes, the quantifications and coefficients are not limited to specific values. Thus, a specific initialisation for those values has to be set.

The default initialisation of the quantifications is as a vector of the unique values after standardising the original variable. Standardisation is performed for a numeric variable by subtracting its mean and dividing by its variance. This results in a variable with a mean of zero and a variance of one, where the dot product with itself equals the total number of observations N. If a variable is categorical, the variable is standardised after replacing categories with equally spaced numerical values. Finally, all coefficients for this model are initialised as ones.

The resulting standardised variable after initialisation is actually the same as the final estimation of the transformed variable for the numeric scaling level, since these standardised values already satisfy the restrictions of the numeric scaling level. As a result, the transformations of variables using this scaling level do not have to be iteratively updated as described for the other scaling levels in further sections. Their coefficients are still updated in the iterative process, however.

2.3.2 Estimation of quantifications

Essential to the algorithm is the estimation of coefficients β_1 and β_2, and quantification vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ from the data. Estimation of these quantifications is based on minimising the loss function of the OS-regression model, which is written as

$$L(\beta_1, \beta_2, \boldsymbol{v}_1, \boldsymbol{v}_2) = ||\boldsymbol{y} - \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1 - \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2||^2, \tag{2.7}$$

where $||\cdot||^2$ denotes the squared Euclidean norm. This loss function cannot be optimised for all coefficients and quantifications simultaneously, so optimisation of β_1, β_2, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is performed for one variable at a time. Hence, the algorithm starts with updating the quantifications and coefficient for the first variable, $\boldsymbol{v}_1$ and β_1. To simplify the equation and subsequent derivations, a partial residual is defined as the residual including all terms except for those including the variable that is updated. For the first predictor, this partial residual $\boldsymbol{u}_1$ is defined as

$$\boldsymbol{u}_1 = \boldsymbol{y} - \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2. \tag{2.8}$$

Substituting the partial residual in the loss function (2.7) results in

$$L(\beta_1, \boldsymbol{v}_1) = ||\boldsymbol{u}_1 - \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1||^2. \tag{2.9}$$

The structure of this equation is the same as for simple linear regression, so the ordinary least squares solution minimises this loss function. Hence, the optimal estimate for $\boldsymbol{v}_1$ is

$$\tilde{\boldsymbol{v}}_1 = (\beta_1 \boldsymbol{G}_1' \beta_1 \boldsymbol{G}_1)^{-1} \beta_1 \boldsymbol{G}_1' \boldsymbol{u}_1 \tag{2.10}$$

$$= (\beta_1 \boldsymbol{D}_1)^{-1} \boldsymbol{G}_1' \boldsymbol{u}_1, \tag{2.11}$$

where $\boldsymbol{D}_1 = \boldsymbol{G}_1'\boldsymbol{G}_1$ is a diagonal matrix with the marginal frequencies of each category. E.g. for $\boldsymbol{x}_1$ in the toy data, the marginal frequencies would be

$$\boldsymbol{D}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}. \tag{2.12}$$

In each iteration, the estimated quantifications vector $\tilde{\boldsymbol{v}}_1$ is then standardised. This standardisation is weighted according to the frequencies for each category. This standardisation is the same as for initialisation of the quantifications and the numeric scaling level, i.e. to $E(\boldsymbol{G}_1\boldsymbol{v}_1) = 0$ and $\sigma^2(\boldsymbol{G}_1\boldsymbol{v}_1) = 1$, and as a result, $(\boldsymbol{G}_1\boldsymbol{v}_1)'(\boldsymbol{G}_1\boldsymbol{v}_1) = N$ as well. The standardised quantifications vector is denoted as $\mathring{\boldsymbol{v}}_1$ and these are then the optimal quantifications for the nominal scaling level. They have not been restricted in any way, apart from the fact that there is only one quantification for each category. To achieve the ordering and/or smoothing restrictions, the unstandardised, nominal quantifications $\tilde{\boldsymbol{v}}_1$ are first calculated as above and then restricted and standardised.

Since a (weighted) standardisation is used for the quantifications, the resulting quantifications after standardisation will be the same, regardless of inclusion of β_1 in the update equation. Hence, β_1 can be left out of the equation for the optimal quantifications estimate and it can thus be simplified to

$$\tilde{\boldsymbol{v}}_1 = \boldsymbol{D}_1^{-1}\boldsymbol{G}_1'\boldsymbol{u}_1. \tag{2.13}$$

This formula is then actually the same as calculating the weighted mean for each category of the partial residual.

Ordinal quantifications are estimated using weighted monotonic regression (Kruskal, 1965), which replaces any nonincreasing nominal quantifications of consecutive categories by their weighted average. (Non)monotone spline scaling level restrictions are both estimated using an I-spline with some user-specified degree and number of knots, and, for the monotone spline scaling level, these splines are restricted to monotonicity as described in Ramsay (1988).

Additionally, if a monotone restriction results in a monotonically decreasing function, the sign of the quantifications and the coefficient is reversed, enforcing monotonic increasing quantifications.

2.3.3 Estimation of coefficients

After updating the quantifications of a variable, its coefficient is updated as well. An optimal solution is found based on the same loss function as when updating the quantifications (2.9), in which the partial residual is substituted, i.e.

$$L(\beta_1, \boldsymbol{v}_1) = ||\boldsymbol{u}_1 - \beta_1\boldsymbol{G}_1\boldsymbol{v}_1||^2. \tag{2.14}$$

The coefficient β_1 that minimises this loss function is again the same as the solution for ordinary least squares in linear regression. Hence, the update equation, using updated and standardised quantifications $\mathring{\boldsymbol{v}}_1$, is then

$$\hat{\beta}_1 = ((\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)'\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)^{-1}(\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)'\boldsymbol{u}_1. \tag{2.15}$$

Since the quantifications are standardised before updating the coefficient, the inverse part of the equation, $((\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)'\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)^{-1}$, can be simplified to N^{-1} and the update formula can be rewritten

as

$$\hat{\beta}_1 = N^{-1}(\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)'\boldsymbol{u}_1. \tag{2.16}$$

After updating the quantifications and coefficient for the first variable, $\boldsymbol{v}_1$ and β_1, the quantifications and coefficient for the second variable, $\boldsymbol{v}_2$ and β_2, are updated. This process is then iterated until the convergence criteria are met.

2.3.4 Algorithm summary

The algorithm for estimating the OS-regression model with two variables can be summarised as follows.

Initialisation: Construct indicator matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ and the matrices containing the marginal frequencies $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ from the data and initialise coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ and quantifications $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$.

Update main effects: For $\{j, k\} = \{1, 2\}$ and $\{2, 1\}$, do:

- **Step 1:** If j has a numeric scaling level, go to step 6. Otherwise, calculate the partial residual as $\boldsymbol{u}_j = \boldsymbol{y} - \hat{\beta}_k\boldsymbol{G}_k\hat{\boldsymbol{v}}_k$.
- **Step 2:** Update the quantifications as $\tilde{\boldsymbol{v}}_j = \boldsymbol{D}_j^{-1}\boldsymbol{G}_j'\boldsymbol{u}_j$.
- **Step 3:** For variables restricted to scaling levels other than nominal, apply restrictions according to the respective scaling levels.
- **Step 4:** Standardise quantifications and denote standardised quantifications as $\mathring{\boldsymbol{v}}_j$.
- **Step 5:** Set $\hat{\boldsymbol{v}}_j = \mathring{\boldsymbol{v}}_j$.
- **Step 6:** Update the coefficient as $\hat{\beta}_j = N^{-1}(\boldsymbol{G}_j\hat{\boldsymbol{v}}_j)'\boldsymbol{u}_j$.

Convergence: Repeat the update step until the change in total loss $L(\hat{\beta}_1, \hat{\beta}_2, \hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2) = ||\boldsymbol{y} - \hat{\beta}_1\boldsymbol{G}_1\hat{\boldsymbol{v}}_1 - \hat{\beta}_2\boldsymbol{G}_2\hat{\boldsymbol{v}}_2||^2$ compared to the previous iteration is smaller than the convergence criterion or until the maximum number of iterations has been reached.

2.4 Model fit measure

To evaluate how well the model has fit the data, the Apparent Prediction Error (APE) is used. The APE is defined as

$$APE = 1 - R^2, \tag{2.17}$$

where R^2 is the squared multiple regression coefficient. This equation can be rewritten to

$$APE = \frac{||\boldsymbol{y} - \hat{\boldsymbol{y}}||^2}{N}. \tag{2.18}$$

The APE is then interpreted as a measure of how well the model can predict the data used to fit the model. It can range from 0 to 1, where a value of 0 means that the model fits the data perfectly with no error at all.

Chapter 3

Modelling Factor-by-Curve interactions with Optimal Scaling regression

To model interactions in OS-regression, the concept of Factor-by-Curve (FbC) interactions is used. The term Factor-by-Curve is usually used to describe interactions between a categorical and a continuous variable. Conceptually, the idea of extending a regression model to allow for FbC interactions is to fit a separate curve for the continuous variable for each level of the categorical variable. By plotting these curves together, interactions between the two variables can be visualised. Clear differences between these curves are an indication of an interaction effect between the two variables. A simulated example of a FbC interaction is shown in Figure 3.1.

Compared to interactions between two continuous variables, FbC interactions can, in some ways, be viewed as a more simple type of interaction. The former can be analysed using methods such as interaction splines (Wahba, 1990; Chen, 1993), multivariate adaptive regression splines (MARS) (Friedman, 1991), and tensor product smooths as used in implementations of additive models (Wood, 2006). All three methods use combinations of some basis functions, often smooth, to allow estimation of a regression surface of the two continuous variables; Figure 1.2a shows an example of a regression surface for two variables. The analysis of an interaction between two continuous variables could, rudimentary, be viewed as using an analysis of a FbC interaction for a factor with many levels. However, since those categories are actually numeric values, the distance between those values is known. As a result, the methods should consider these interrelations when estimating the regression surface and are thus more complex. Additionally, to accurately estimate a regression surface, an adequate number of observations spread all over the combinations of the two continuous variables, within the desired range, is necessary. While the same is true for the analysis of FbC interactions, the number of combinations is reduced if the categorical variable has only few categories, meaning it generally has lower requirements in terms of number of observations. Estimation is also simpler, since no information on interrelations between the categories, like distances, is used for the interaction.

While the analysis of FbC interactions has been implemented in other statistical techniques, such as in implementations of the additive model and nonparametric estimation techniques (Wood, 2017; Sestelo et al., 2017), there are some differences in doing so with OS-regression. Subsequent sections will describe the Factor-by-Curve Optimal Scaling regression (FbC-OS-regression) model and detail its implementation.

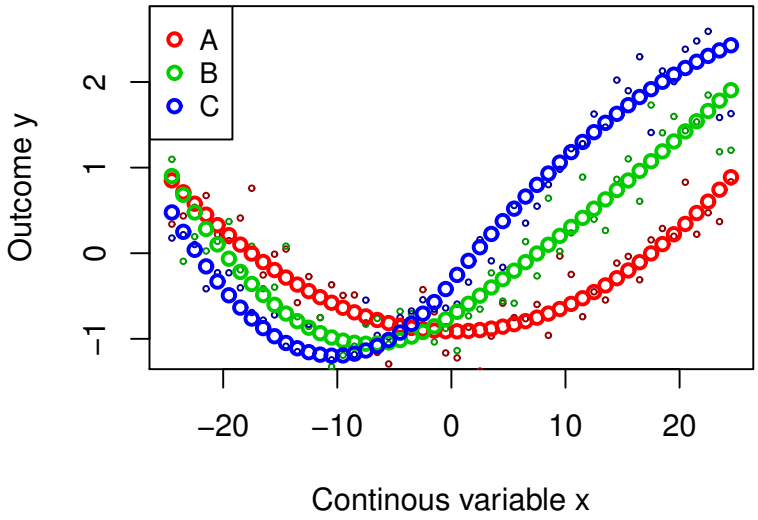


Figure 3.1: Example of a Factor-by-Curve interaction plot depicting predicted values for simulated data. Curves are fitted separately for observations of different levels, in this case as shown for the three levels A, B and C, using Factor-by-Curve OS-regression as detailed in this chapter. Smaller dots represent the observed outcome values.

3.1 FbC-OS-regression model

The model for FbC-OS-regression is quite similar to ordinary OS-regression. The main difference is the addition of a new bivariate term, which models the joint effects. This term also consists of a coefficient, β_{12}, an indicator matrix, $\boldsymbol{G}_{12}$, and a quantifications vector, $\boldsymbol{v}_{12}$. Using this bivariate term, the FbC-OS-regression model can be written as

$$\boldsymbol{y} = \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1 + \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2 + \beta_{12} \boldsymbol{G}_{12} \boldsymbol{v}_{12} + \boldsymbol{\varepsilon}. \tag{3.1}$$

The indicator matrix and quantifications vector for the interactions term are slightly different from those used in ordinary OS-regression and will therefore be detailed below.

For this chapter, the categorical variable will be assumed to contain three levels, here denoted by A, B and C. This can easily be generalised to more categories. For an interaction between a continuous variable and a factor of three levels, three curves are then fitted on the continuous variable, besides the main effects for both variables. These curves are represented using an indicator matrix, $\boldsymbol{G}_{12}$, and a quantifications vector, $\boldsymbol{v}_{12}$, as can be seen in (3.1). The indicator matrix and quantifications vector can both be considered to consist of submatrices and subvectors, respectively, with the number of submatrices and subvectors depending on the number of factor levels. For levels A, B and C, indicator matrix $\boldsymbol{G}_{12}$ consists of three submatrices $\boldsymbol{G}_{12_A}$, $\boldsymbol{G}_{12_B}$ and $\boldsymbol{G}_{12_C}$. Likewise, three subvectors $\boldsymbol{v}_{12_A}$, $\boldsymbol{v}_{12_B}$ and $\boldsymbol{v}_{12_C}$, for levels A, B and C respectively, are considered for quantifications vector $\boldsymbol{v}_{12}$.

The submatrices for the indicator matrix are specific for each factor level, since they only include the values observed for the continuous variable within that level. Thus, the number of columns for each submatrix is not necessarily equal to the total number of unique values in the continuous variable, but can be any positive integer equal to or lower than that number of categories. This number of categories for the continuous variable, conditional on the factor

level, is then denoted as C_{12_A}, C_{12_B} and C_{12_C}, respectively for levels A, B and C. For example, submatrix $\boldsymbol{G}_{12_A}$ has N rows and C_{12_A} columns. It consists solely of ones and zeroes, only indicating categories with a one if that observation is of the relevant factor level, i.e. for submatrix $\boldsymbol{G}_{12_A}$, only observations of category A contain a one to specify the numeric value. For the toy data, these indicator matrices are then

$$\boldsymbol{G}_{12_A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \boldsymbol{G}_{12_B} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \text{ and } \boldsymbol{G}_{12_C} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.2}$$

For indicator matrix $\boldsymbol{G}_{12_A}$, the first and second columns respectively indicate numeric values 1 and 2 from continuous toy data variable $\boldsymbol{x}_2$. The third column does not indicate numeric value 3, instead it indicates numeric value 4, since none of the observations with factor level A were found to have a value of 3 for the continuous variable.

These separate indicator matrices, $\boldsymbol{G}_{12_A}$, $\boldsymbol{G}_{12_B}$ and $\boldsymbol{G}_{12_C}$, can be combined into the overall indicator matrix $\boldsymbol{G}_{12}$, by concatenating the matrices into a single matrix with N rows and $C_{12_A} + C_{12_B} + C_{12_C}$ columns. This indicator matrix can then, theoretically, represent an infinite number of factor levels.

Similarly, quantification subvectors $\boldsymbol{v}_{12_A}$, $\boldsymbol{v}_{12_B}$ and $\boldsymbol{v}_{12_C}$ are vectors of lengths C_{12_A}, C_{12_B} and C_{12_C} respectively. They can be concatenated to form the overall quantifications vector $\boldsymbol{v}_{12}$ having length $C_{12_A} + C_{12_B} + C_{12_C}$.

For the toy data example, indicator matrix $\boldsymbol{G}_{12}$ and quantifications vector $\boldsymbol{v}_{12}$ are thus

$$\boldsymbol{G}_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \text{ and } \boldsymbol{v}_{12} = \begin{pmatrix} \boldsymbol{v}_{12_A,1} \\ \boldsymbol{v}_{12_A,2} \\ \boldsymbol{v}_{12_A,3} \\ \boldsymbol{v}_{12_B,1} \\ \boldsymbol{v}_{12_B,2} \\ \boldsymbol{v}_{12_C,1} \\ \boldsymbol{v}_{12_C,2} \end{pmatrix}. \tag{3.3}$$

Note that the number 3 in $\boldsymbol{v}_{12_A,3}$ is used as an index of the third value in the vector and is not referring to the numeric value 3, just like how the third column of indicator matrix $\boldsymbol{G}_{12_A}$ indicates numeric value 4. The same holds for the other subscripts in the example as well.

While the FbC-OS-regression model as defined in (3.1) is similar to the ordinary OS-regression model, addition of the bivariate term for modelling of joint effects leads to changes in the algorithm and implementation. These differences with the ordinary OS-regression model are detailed in subsequent sections.

3.2 Algorithm

As with the algorithm for ordinary OS-regression, the updating algorithm for FbC-OS-regression consists of initialisation of necessary parameters, iteratively optimising both quantifications and coefficients until some criterion has been reached, either based on a sufficiently small decrease of the model's loss function or a maximum number of iterations. However, as can be concluded from the model formulation, the algorithm is extended, since the quantifications for the interaction

effect and its coefficient need to be updated as well. Namely, in each iteration, FbC-OS-regression first updates the quantifications and coefficients for each of the main effects, after which the quantifications and coefficient for the interaction term are updated. The loss function used for this model is

$$L(\beta_1, \beta_2, \beta_{12}, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_{12}) = ||\boldsymbol{y} - \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1 - \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2 - \beta_{12} \boldsymbol{G}_{12} \boldsymbol{v}_{12}||^2. \tag{3.4}$$

3.2.1 Initialisation of parameters

For FbC-OS-regression, the parameters that need to be initialised prior to the optimisation process are indicator matrices $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ and $\boldsymbol{G}_{12}$, quantification vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_{12}$, and coefficients β_1, β_2 and β_{12}. Parameters for the main effects are initialised as described in subsection 2.3.1 for ordinary OS-regression.

The indicator matrix for the interaction term, $\boldsymbol{G}_{12}$, is constructed from the data as in the example shown in (3.3), whereas quantifications vector $\boldsymbol{v}_{12}$ is initialised as a vector of zeroes. The coefficient for the interaction, β_{12}, is initialised as the coefficients for the main effects, so as a scalar of value 1.

3.2.2 Estimation of quantifications

The quantifications for the main effects are updated using the same updating formula as for ordinary OS-regression, but, since this model uses a different loss function, the partial residual is redefined to include both the other main effect and the interaction effect. Hence, for the main effect of the first variable, the partial residual is defined as

$$\boldsymbol{u}_1 = \boldsymbol{y} - \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2 - \beta_{12} \boldsymbol{G}_{12} \boldsymbol{v}_{12}. \tag{3.5}$$

The quantifications for the main effect of the first predictor is then updated as

$$\tilde{\boldsymbol{v}}_1 = \boldsymbol{D}_1^{-1} \boldsymbol{G}_1' \boldsymbol{u}_1, \tag{3.6}$$

which is the same equation as for ordinary OS-regression, as defined in (2.13), but partial residual $\boldsymbol{u}_1$ is thus different for ordinary OS-regression and FbC-OS-regression.

Estimation of the quantifications for the interaction term, $\boldsymbol{v}_{12}$, is similar to estimation of the main effect quantifications. They are first fitted as nominal quantifications, with quantifications for each combination of observed factor level and observed numeric value in that level. The partial residual for the interaction term can be defined as

$$\boldsymbol{u}_{12} = \boldsymbol{y} - \beta_1 \boldsymbol{G}_1 \boldsymbol{v}_1 - \beta_2 \boldsymbol{G}_2 \boldsymbol{v}_2, \tag{3.7}$$

and hence loss function (3.4) can be rewritten as

$$L(\beta_{12}, \boldsymbol{v}_{12}) = ||\boldsymbol{u}_{12} - \beta_{12} \boldsymbol{G}_{12} \boldsymbol{v}_{12}||^2. \tag{3.8}$$

The nominal quantifications for the interaction are then derived using the least squares solution and estimated as

$$\tilde{\boldsymbol{v}}_{12} = \boldsymbol{D}_{12}^{-1} \boldsymbol{G}_{12}' \boldsymbol{u}_{12}, \tag{3.9}$$

with $\boldsymbol{D}_{12} = \boldsymbol{G}_{12}' \boldsymbol{G}_{12}$. From update formula (3.9), it can be seen that all quantifications for the interaction term are updated simultaneously, and separate values are estimated for categories of the continuous variable in each level of the categorical variable.

The interaction quantifications for each factor are then separately restricted to (non)monotone splines using the spline restrictions as in ordinary OS-regression. After restriction, the quantifications are standardised. This standardisation is performed on the complete transformed interaction variable, i.e. on $\boldsymbol{G}_{12} \boldsymbol{v}_{12}$.

3.2.3 Estimation of coefficients

After estimating the quantifications for a main effect, its coefficient is estimated. Again, the optimal solution for the coefficient of the main effects is the same as in ordinary OS-regression, but using the new definition for the partial residual as given in (3.5). The update formula for the coefficient of the first predictor is thus

$$\hat{\beta}_1 = N^{-1}(\boldsymbol{G}_1\mathring{\boldsymbol{v}}_1)'\boldsymbol{u}_1. \tag{3.10}$$

Similarly, after updating the quantifications for the interaction term, coefficient $\hat{\beta}_{12}$ is updated. The update formula for this interaction coefficient is of the same form as (3.10), but using total standardised transformed variable $\boldsymbol{G}_{12}\mathring{\boldsymbol{v}}_{12}$ and earlier defined partial residual $\boldsymbol{u}_{12}$. The coefficient is then updated as

$$\hat{\beta}_{12} = ((\boldsymbol{G}_{12}\mathring{\boldsymbol{v}}_{12})'\boldsymbol{G}_{12}\mathring{\boldsymbol{v}}_{12})^{-1}(\boldsymbol{G}_{12}\mathring{\boldsymbol{v}}_{12})'\boldsymbol{u}_{12} \tag{3.11}$$

$$= N^{-1}(\boldsymbol{G}_{12}\mathring{\boldsymbol{v}}_{12})'\boldsymbol{u}_{12}, \tag{3.12}$$

again simplifying the inner product in the inverse to N, as is done in ordinary OS-regression in (2.16), due to the standardisation of the quantifications.

3.2.4 Algorithm summary

The algorithm for two variables and their interaction can be summarised as shown below.

Initialisation: Construct indicator matrices $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ and $\boldsymbol{G}_{12}$ and the matrices containing the marginal frequencies $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ and $\boldsymbol{D}_{12}$ from the data and initialise coefficients $\hat{\beta}_1$, $\hat{\beta}_2$ and $\hat{\beta}_{12}$ and quantifications $\hat{\boldsymbol{v}}_1$, $\hat{\boldsymbol{v}}_2$ and $\hat{\boldsymbol{v}}_{12}$.

Update main effects: For $\{j,k\} = \{1,2\}$ and $\{2,1\}$, do:

- **Step 1:** If j has a numeric scaling level, go to step 6. Otherwise calculate the partial residual as $\boldsymbol{u}_j = \boldsymbol{y} - \hat{\beta}_k \boldsymbol{G}_k \hat{\boldsymbol{v}}_k - \hat{\beta}_{12} \boldsymbol{G}_{12} \hat{\boldsymbol{v}}_{12}$.
- **Step 2:** Update the quantifications as $\tilde{\boldsymbol{v}}_j = \boldsymbol{D}_j^{-1} \boldsymbol{G}_j' \boldsymbol{u}_j$.
- **Step 3:** For variables restricted to scaling levels other than nominal, apply restrictions according to the respective scaling levels.
- **Step 4:** Standardise quantifications and denote standardised quantifications as $\mathring{\boldsymbol{v}}_j$.
- **Step 5:** Set $\hat{\boldsymbol{v}}_j = \mathring{\boldsymbol{v}}_j$.
- **Step 6:** Update the coefficient as $\tilde{\beta}_j = N^{-1} (\boldsymbol{G}_j \hat{\boldsymbol{v}}_j)' \boldsymbol{u}_j$.
- **Step 7:** Set $\hat{\beta}_j = \tilde{\beta}_j$.

Update interaction effect: Do:

- **Step 1:** Define partial residual $\boldsymbol{u}_{12} = \boldsymbol{y} - \hat{\beta}_1 \boldsymbol{G}_1 \hat{\boldsymbol{v}}_1 - \hat{\beta}_2 \boldsymbol{G}_2 \hat{\boldsymbol{v}}_2$.
- **Step 2:** Update the quantifications as $\tilde{\boldsymbol{v}}_{12} = \boldsymbol{D}_{12}^{-1} \boldsymbol{G}_{12}' \boldsymbol{u}_{12}$.
- **Step 3:** Apply restrictions according to either monotone or nonmonotone spline scaling level, separately for each level of the factor.
- **Step 4:** Standardise quantifications and denote standardised quantifications as $\mathring{\boldsymbol{v}}_{12}$.
- **Step 5:** Set $\hat{\boldsymbol{v}}_{12} = \mathring{\boldsymbol{v}}_{12}$.
- **Step 6:** Update the coefficient as $\hat{\beta}_{12} = N^{-1} (\boldsymbol{G}_{12} \hat{\boldsymbol{v}}_{12})' \boldsymbol{u}_{12}$.

Convergence: Repeat updating steps for both main and interaction effects until the change in total loss $L(\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_{12}, \hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2, \hat{\boldsymbol{v}}_{12}) = ||\boldsymbol{y} - \hat{\beta}_1 \boldsymbol{G}_1 \hat{\boldsymbol{v}}_1 - \hat{\beta}_2 \boldsymbol{G}_2 \hat{\boldsymbol{v}}_2 - \hat{\beta}_{12} \boldsymbol{G}_{12} \hat{\boldsymbol{v}}_{12}||^2$ compared to the previous iteration is smaller than the convergence criterion or until the maximum number of iterations has been reached.

3.3 Implementation

The FbC-OS-regression model was implemented in R (R Core Team, 2019), also using the `splines2` package to create I-spline bases (Wang and Yan, 2018). The implementation is based on the model formulae described in previous sections, but is optimised to, among others, reduce the use of the sparse indicator matrices. This optimisation is also used in ordinary OS-regression to reduce necessary memory and processing time. Since the FbC-OS-regression model is naturally based on the ordinary OS-regression model, many parts of the implementation are similar. However, not all parts of the implementation are directly transferable to the FbC-OS-regression model. Three main changes were made and these will be discussed in this section.

Firstly, in ordinary OS-regression, monotone splines are restricted to monotonically increasing splines (instead of decreasing), compensating with the sign of its quantifications and coefficient if necessary. However, this approach cannot be adopted for monotone splines in the interaction, since the direction of the interaction should be allowed to differ for each curve, allowing both monotonically increasing and decreasing curves. Additionally, a single coefficient is preferably still used to indicate the overall effect of the interaction. Therefore, the coefficient for monotone spline curves in the Factor-by-Curve interactions is restricted to being positive, while the splines are allowed to vary in being either monotonically increasing or decreasing. Any main effects included in the model using the monotone spline scaling level are still restricted to be monotonically increasing, while allowing the coefficient to vary in sign, as with ordinary OS-regression.

Secondly, a problem with monotone restrictions is that they are liable to local minima. Namely, there are multiple solutions for the restriction of nominal quantifications to ordinal quantifications or nonmonotone spline quantifications to monotone spline quantifications. In short, this means that the iterative procedure can converge to multiple solutions, while one of those is the global minimizer of the loss function. The OS-regression model can thus converge to multiple sets quantifications, of which one is the overall optimal solution. This global minimum can be found by estimating all possible solutions and determining which of those is the global optimal solution. For a full overview on this problem, the reader is referred to van der Kooij (2007). Testing all possible solutions of quantifications increases the performance time greatly, exponentially even, for the number of variables included having an ordinal or monotone spline scaling level. Therefore, the current default for ordinary OS-regression is not to test for additional possible solutions, unless requested by the user. In the implementation of FbC-OS-regression as presented in this thesis it does not take a lot of time to calculate all possible solutions, since the specified FbC-OS-regression model is small. Thus, the decision was made to determine all these possible optimal solutions for monotone curves in the interaction term, allowing for selection of the global optimal solution. The optimal solution is determined by calculating the sum of squares of the difference between all possible monotone spline quantifications and the nonmonotone spline quantifications from which they were derived, selecting the solution with the smallest sum of squares.

Lastly, the decision was made to standardise the interaction quantifications as a whole, as described in the previous section on estimating the quantifications. It would have been an option to standardise them separately for each curve, but the current standardisation method was chosen so that the interaction coefficient is then interpreted similarly to the coefficients for the main effects. Additionally, the separate curves can be interpreted relative to each other when standardised similarly.

An additional note on the FbC-OS-regression model, is that the model itself does not necessarily restrict the interaction curves all to the same spline scaling level. The current implementation, however, restricts the curves in the interaction term to either all monotone or all nonmonotone splines.

3.4 Alternative FbC-OS-regression model

In the previous sections, the FbC-OS-regression model was defined such that the main and interaction effects are all estimated separately. Alternatively, the model can be formulated to include both main effects and the interaction effect in a single term. The FbC-OS-regression model with this combined term can be formulated as

$$\boldsymbol{y} = \beta_{12}\boldsymbol{G}_{12}\boldsymbol{v}_{12} + \boldsymbol{\varepsilon}. \tag{3.13}$$

In this way, the total effect of the two variables can be restricted to (non)monotone splines, as opposed to restricting all effects separately. The algorithm for this model is mostly the same as for the other FbC-OS-regression model formulation, but without the steps related to the main effects. It can be summarised as below. Note that no partial residual is necessary for this model, since its loss function is defined as

$$L(\beta_{12}, \boldsymbol{v}_{12}) = ||\boldsymbol{y} - \beta_{12}\boldsymbol{G}_{12}\boldsymbol{v}_{12}||^2, \tag{3.14}$$

which only contains a single term with parameters that need to be updated. As a result, the partial residual in this model would actually equate to the outcome, i.e. $\boldsymbol{u}_{12} = \boldsymbol{y}$, so this step is omitted.

Initialisation: Construct indicator matrix $\boldsymbol{G}_{12}$ and marginal frequencies matrix $\boldsymbol{D}_{12}$ from the data and initialise coefficient $\hat{\beta}_{12}$ and quantifications vector $\hat{\boldsymbol{v}}_{12}$.

Update overall effect: Do:

Step 1: Update the quantifications as $\tilde{\boldsymbol{v}}_{12} = \boldsymbol{D}_{12}^{-1}\boldsymbol{G}_{12}'\boldsymbol{y}$.

Step 2: Apply restrictions according to either monotone or nonmonotone spline scaling level, separately for each level of the factor.

Step 3: Standardise quantifications and denote standardised quantifications as $\mathring{\boldsymbol{v}}_{12}$.

Step 4: Set $\hat{\boldsymbol{v}}_{12} = \mathring{\boldsymbol{v}}_{12}$.

Step 5: Update the coefficient as $\hat{\beta}_{12} = N^{-1}(\boldsymbol{G}_{12}\hat{\boldsymbol{v}}_{12})'\boldsymbol{y}$.

Convergence: Repeat the update step until the change in total loss $L(\hat{\beta}_{12}, \hat{\boldsymbol{v}}_{12}) = ||\boldsymbol{y} - \hat{\beta}_{12}\boldsymbol{G}_{12}\hat{\boldsymbol{v}}_{12}||^2$ compared to the previous iteration is smaller than the convergence criterion or until the maximum number of iterations has been reached.

Chapter 4

Application of FbC-OS-regression to the BCRP data

Both implementations of the FbC-OS-regression models are applied to the Breast Cancer Recovery Project (BCRP) data set (Scheier et al., 2007). The BCRP data were used to study the impact of certain psychosocial interventions on physical and psychological functioning in women completing treatment for early-stage breast cancer. It has previously been shown by Scheier et al. (2007) that there are two-way interaction effects between the categorical predictor variable of interest, which is the variable indicating the psychosocial intervention patients received, and some of the numerical predictor variables in this data set. Given the presence of interactions, this real data set is suitable to illustrate Factor-by-Curve interactions in OS-regression. The interaction effects between two combinations of a categorical and numerical variable are fitted using the two FbC-OS-regression models and the ordinary linear regression model. Subsequently, the results of the fitted models for both variable combinations are compared.

4.1 Data description

The Breast Cancer Recovery Project data set consists of 252 female patients with stage 0, I, or II breast cancer. Each patient has finished a nonhormonal adjuvant therapy at most two months prior to the start of the study. All patients were randomly assigned to one of three groups. The first group that patients could be assigned to was a control group, in which patients received standard medical care. The other two groups received a psychosocial intervention, either nutritional or educational. The nutritional intervention consisted of information on how to adopt and adhere to a low-fat, high-fruit, and high-vegetable diet. The educational intervention informed patients about their disease and its treatment, including relevant coping skills. Meetings for both interventions were held once a month for four consecutive months and all sessions lasted two hours. The *experimental condition* variable indicates the group to which each patient was assigned.

In this study, two outcomes are of interest. The first indicates the *change in depressive symptoms* of the patient and the second indicates the *change in physical functioning*. Both depressive symptoms and physical functioning were measured at baseline, i.e. before assignment to an intervention type or standard care, at 4 months after baseline, i.e. after end of the intervention, and at 9 months after that, i.e. 13 months after baseline. The difference between the measurements at baseline and at 9 months postintervention follow-up is used as outcome in this thesis. More specifically, the baseline measurement values are subtracted from the values measured at the 9 months postintervention follow-up, which is why the outcomes of interest are

Variable	Description
Change in Depressive Symptoms	Depressive symptoms of the patient, twice assessed based on a ten-item survey, with change in score used as the variable.
Change in Physical Functioning	Physical functioning of the patient, twice assessed based on a thirty-six-item survey, with change in score used as the variable.
Unmitigated Communion	Extent of placing others' needs above patient's own, based on a nine-item scale.
Number of Comorbidities	Counts of comorbidities, e.g. diabetes, migraines et cetera.
Experimental Condition	Randomised treatment group.

Table 4.1: Descriptions for the variables in the BCRP data set used in the analyses in this thesis.

defined as *change in depressive symptoms* and *change in physical functioning*. Positive values then indicate an increase in outcome after 9 months postintervention follow-up, compared to the baseline measurement. For the *change in depressive symptoms variable*, positive values indicate an increase in depressive symptoms, which is an undesirable outcome. Positive values for the *change in physical functioning* variable, however, are desirable, since they indicate an increase in physical functioning.

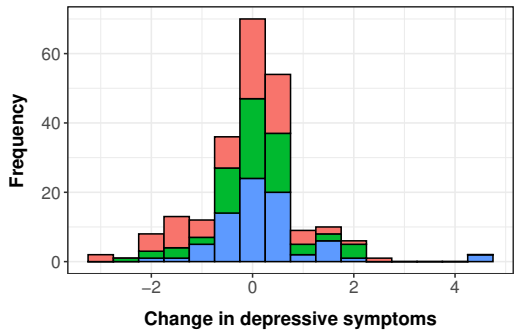
Some data are missing from this data set, namely for 28 patients the measurements at 9 months postintervention follow-up were not recorded and therefore omitted in the analysis. As a result, complete data are available for 224 patients. The distributions of the patients over the outcome values, including within each intervention group, are shown in the stacked histograms in Figure 4.1a and Figure 4.1b. The number of patients in each treatment group for which complete data are available resulted in 78 patients that received the nutritional intervention, 70 patients that received the educational intervention, and 76 patients that received standard care.

For the analyses described in the next section, the numerical predictor variables of interest are *unmitigated communion* and *number of comorbidities*. Both were assessed at baseline. These two predictor variables are expected to have an interaction effect with *experimental condition* on *change in depressive symptoms* and *change in physical functioning*, respectively. Further descriptions for these predictor variables and earlier described outcomes are shown in Table 4.1. The distributions of the patients over these predictor variable values, across each intervention group, are shown in the histograms in Figure 4.1c and Figure 4.1d.

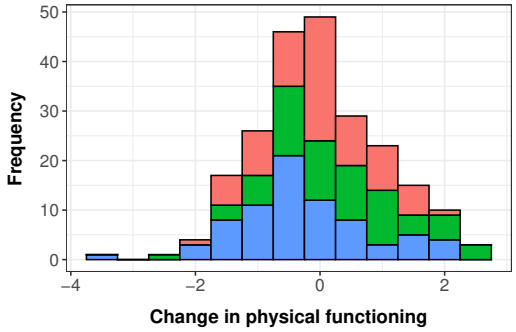
For more details on patient inclusion, the protocol of the BCR Project, and (assessment of) its variables, the reader is referred to Scheier et al. (2005, 2007).

4.2 Analysis of interaction effects

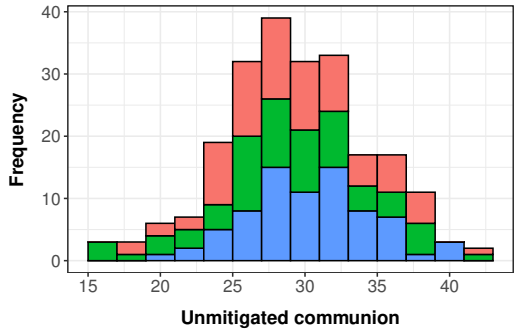
The previous study on interaction effects in the BCRP data by Scheier et al. (2007) found multiple two-way interactions between the *experimental condition* variable and some of the numerical variables on both outcome variables. However, Scheier et al. (2007) defined a dummy variable for either the nutritional or the educational intervention types. For example, for the analysis in which the nutritional intervention dummy variable was used, the effect size of the nutritional intervention is relative to those not in the nutritional intervention group, i.e. relative to those in either the control group or the educational intervention group. In each of our analyses, any one treatment group is not compared to the combination of the two other treatment groups, but the



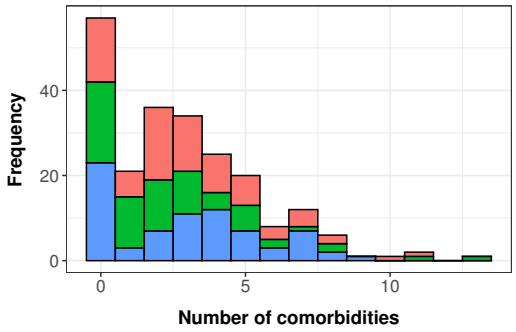
(a) Stacked histogram for outcome variable *change in depressive symptoms*



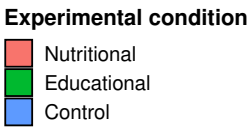
(b) Stacked histogram for outcome variable *change in physical functioning*



(c) Stacked histogram for predictor variable *unmitigated communion*



(d) Stacked histogram for predictor variable *number of comorbidities*



(e) Legend

Figure 4.1: Stacked histograms for both outcome and binned continuous variables used in the analyses. The legend for all histograms is given in (e).

three treatments are simultaneously analysed as three separate groups. Therefore, the results described in this thesis are not directly comparable to those by Scheier et al. (2007).

The study by Scheier et al. (2007) focused on how the numerical variables affect the relation between *experimental condition* and either outcome. This is more specifically represented by the term "moderation", in which the numerical variables of the BCRP study were hypothesized to moderate the effect of *experimental condition* on either outcome. Interpretations of the results are also formulated in this manner.

One significant interaction effect found by Scheier et al. (2007) was the interaction between

the dummy representing the nutritional intervention and *unmitigated communion* on *change in depressive symptoms*. Another interaction was that of the nutritional intervention dummy with *number of comorbidities* on *change in physical functioning*. Similarly, the educational intervention type dummy was found to interact with *number of comorbidities* on *change in physical functioning*. This chapter studies all these interactions, fitting them with the two FbC-OS-regression models and the linear regression model with interactions.

Since the application of the FbC-OS-regression models to these interactions includes the *experimental condition* variable directly, as opposed to the use of a dummy variable by Scheier et al. (2007), no separate analyses are needed to study the effect of the nutritional and educational interventions. The interactions that are tested are thus, firstly, the interaction effect of *experimental condition* and *unmitigated communion* on *change in depressive symptoms*, and secondly, the interaction effect of *experimental condition* and *number of comorbidities* on *change in physical functioning*.

For the FbC-OS-regression models, the interaction curves are fitted using a monotone spline with degree two and two interior knots. In the separate effects FbC-OS-regression models, the main effects of the numerical variables, i.e. *unmitigated communion* and *number of comorbidities*, are also fitted using a monotone spline with the same degree and number of interior knots. The *experimental condition* is fitted using the nominal scaling level, since the categories are assumed to have no ordering.

The results from the analyses using FbC-OS-regression are compared to those for ordinary linear regression. In the linear regression analyses, *experimental condition* is considered a factor, resulting in two dummies that define the three categories.

FbC-OS-regression models are considered converged if the decrease in total loss is smaller than 10^{-5}. All analyses are performed in `R` (R Core Team, 2019). Also note that these applications are exploratory and only to demonstrate the model, hence no substantive conclusions on relations in the BCRP data are to be drawn from the results. However, interpretations of the results are given to illustrate how this is done for FbC-OS-regression.

Finally, the quantifications in the separate effects FbC-OS-regression model should be interpreted as being associated with an effect on the residuals of the outcome when accounting for the other terms in the model if an interaction effect is present, as opposed to a direct association with the outcome. In the interpretations of both separate effects FbC-OS-regression models for the applications discussed in this chapter, however, this is simplified to interpreting the quantifications as the effect on the outcome itself, to improve overall clarity of the results. Contrary to the separate effects FbC-OS-regression model, the quantifications for the combined effect FbC-OS-regression model can be directly interpreted as the effect on the outcome, since this model only has a single overall effect term.

4.3 Interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*

First, the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms* is modelled. The estimates of the FbC-OS-regression models are given and they are compared to the results of the linear regression model. Note that for interpretation of the outcome, a positive outcome means an increase in the patient's depressive symptoms. Thus, in this case, a negative outcome is desirable.

4.3.1 Separate effects FbC-OS-regression model

The quantification plots for the separate effects FbC-OS-regression model are shown in Figure 4.2, including the coefficients for each effect. All three coefficients are estimated as positive values, and the coefficient for the interaction effect is the largest. This means that, overall, the interaction effect has the strongest relative impact on the predicted outcome. Additionally, standardisation affects the interpretation of the coefficient sizes on their own. For example, the interaction coefficient of value 0.222 indicates an increase of 0.222 times the standard deviation of the outcome for each increase of 1 standard deviation of the transformed variable of the interaction.

The transformation plots show in- and decreasing curves for the interaction variable displayed in Figure 4.2c, with a relatively high coefficient, indicating that interaction is present. This complicates the interpretation, as it does in linear regression with interactions. For example, the educational intervention has the highest positive quantification for the main effect of the *experimental condition* variable, as shown in Figure 4.2a, meaning it is most associated with an increase of the residuals of *change in depressive symptoms* when the effects of the transformed *unmitigated communion* and interaction variables are accounted for[1]. However, when looking at the interaction plot displayed in Figure 4.2c, the quantifications of this intervention type for numeric values of *unmitigated communion* up to about 30 are lower than for the other types. As a result, the overall predicted values for the educational intervention are not necessarily the highest of the three intervention types; they are only the highest for patients with a numeric value higher than about 30 of *unmitigated communion*. The nutritional intervention has the lowest quantified value for the main effect, but the highest quantified values for the interaction effect for numeric values of *unmitigated communion* up to about 30. So, up to category 30 of *unmitigated communion*, the overall predicted values for the nutritional and educational interventions will be about the same, as is confirmed by the predicted values shown in Figure 4.4b, while for patients with a numeric value of *unmitigated communion* that is higher than about 30, the predicted values for these two intervention types increasingly differ. In contrast, the effect of the standard care and educational intervention groups on the outcome are increasingly similar for these patients.

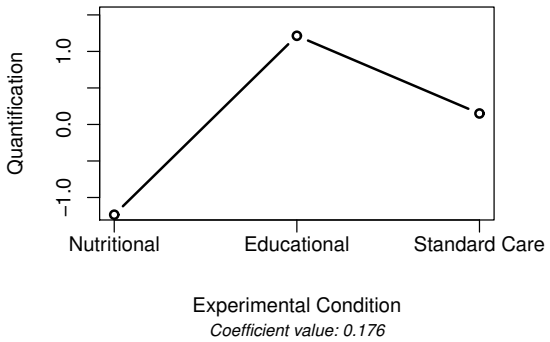
The quantifications for the main effect of *unmitigated communion* shown in Figure 4.2b follow a monotonically increasing function, as specified by the scaling level. This indicates a higher *change in depressive symptoms* for patients showing a stronger degree of *unmitigated communion*. Note, however, that the coefficient of this effect has a value of 0.007, whereas the coefficients of the factor and interaction effects have values 0.176 and 0.222, respectively. In other words, the effect size for this main effect is very small compared to the other effects, meaning that the continuous main effect contributes little to the predictions of the model. Therefore, the continuous main effect does not require substantive interpretation.

Summarising, the overall interpretation is that the educational intervention is similar to standard care, while the nutritional intervention has a more beneficial effect on the *change in depressive symptoms* than standard care, but only for patients with values of *unmitigated communion* higher than 30.

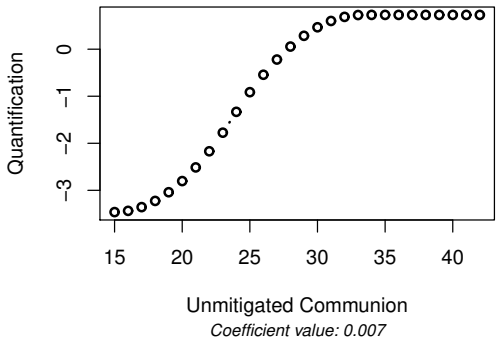
4.3.2 Combined effect FbC-OS-regression model

The combined effect FbC-OS-regression model has only one set of quantifications, which represents the overall effect of both variables and their interaction. Its quantification plot and coefficient is shown in Figure 4.3. Due to the single set of quantifications, these are easier to

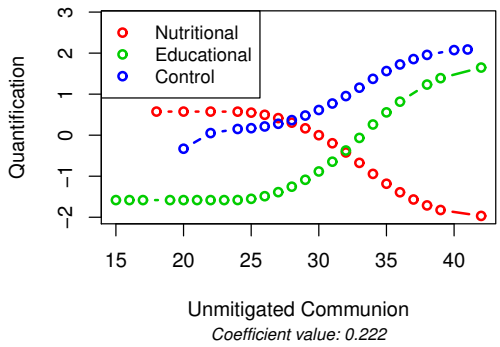
[1]Note that this is the correct interpretation of the quantifications for the separate effects FbC-OS-regression model if an interaction effect is present. Subsequent interpretations of quantifications are simplified, but should be interpreted in earlier described manner.



(a) Quantification plot for *experimental condition*



(b) Quantification plot for *unmitigated communion*



(c) Quantification plot for the interaction effect between *unmitigated communion* and *experimental condition*

Figure 4.2: Quantification plots of the fitted separate effects FbC-OS-regression model for the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*. When fitting this model, the algorithm converged after 66 iterations.

interpret compared to the separate effects FbC-OS-regression model. For patients with a lower value of *unmitigated communion*, the three treatments show a similar effect on the *change in depressive symptoms*. For patients with high values of *unmitigated communion*, however, the nutritional intervention seems to have a more beneficial effect on depressive symptoms, whereas the educational intervention and standard care both show a counterproductive effect. This is indicative of an interaction effect between the treatment groups and *unmitigated communion*, specifically of the nutritional intervention compared to the two other groups. Due to lack of other coefficients, the coefficient cannot be interpreted relatively, only numerically as described in the previous subsection.

4.3.3 Model comparison

Besides fitting the two FbC-OS-regression models, a linear regression model is also fitted to compare the results. The three models can be compared based on their predictions. Plots of the predicted values for all three models are given in Figure 4.4. Density estimates of the *unmitigated communion* variable are also included to indicate its distribution. The model results are most

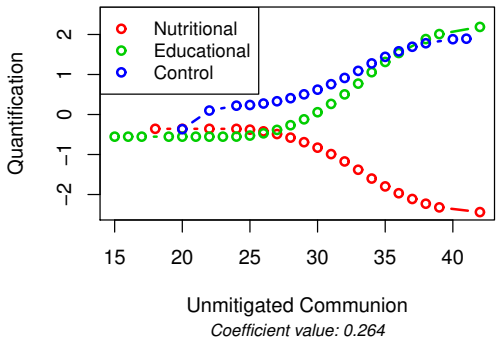


Figure 4.3: Quantification plot of the fitted combined effect FbC-OS-regression model for the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*. The algorithm converged after 40 iterations.

reliable in the regions of high density.

Overall, the three models give a similar indication of the interaction effect. More explicitly, patients with lower values of *unmitigated communion* experience similar outcome values over the three treatment groups, while the treatments have different effects on patients with high values of *unmitigated communion*. Specifically, the nutritional intervention has a more beneficial effect on *change in depressive symptoms* compared to the two other treatment groups.

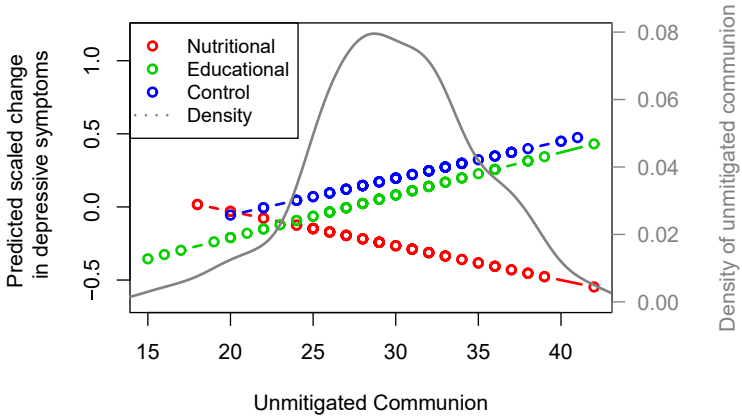
The separate and combined effects FbC-OS-regression models give very similar predictions. Compared to the linear model, the FbC-OS-regression models mainly differ in that they predict quite constant outcome values, i.e. a horizontal line, for patients with lower values (15-25) of *unmitigated communion*. For these patients, predictions are quite similar for all three treatment groups. In contrast, the linear model does differ more in predictions for the three treatment groups for patients with those values of *unmitigated communion*, although the differences are still quite small. Hence, except for a slight increase in flexibility, the predictions of the FbC-OS-regression models compared to those of the linear model are very similar as well.

Note that the quantifications for the combined effect FbC-OS-regression model as shown in Figure 4.3 only differ from the predicted values by a factor equal to the coefficient for that model. This is a result of the model formulation (see (3.13)); predictions are determined by simply multiplying the transformed variable by its coefficient. For the separate effects FbC-OS-regression model, the overall functions of the predicted values look very similar to the quantifications of the interaction effect (Figure 4.2c). This is because the effect size of *unmitigated communion* in this model is very small and hence the overall shape of the individual curves is hardly affected. However, when disregarding the shapes, the curves are different from the interaction quantifications when comparing their overall effect on the outcome, i.e. their positioning on the y-axis. This is due to the effect of the *experimental condition* variable, which has a coefficient more similar in size to that of the interaction effect.

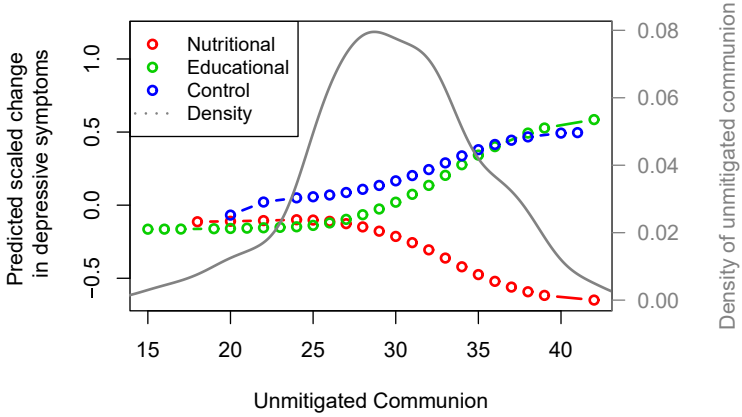
Apparent prediction errors of the three models are given in Table 4.2. All three models resulted in very high APE values, which indicates that none of the models describe the relation very well. These high APE values might be a result of using only two variables to describe changes in depression, which is a complex condition. Apart from all the models having high APE values, the APEs for both FbC-OS-regression models are the same for the given precision. Given the likeness of the predicted value plots, this was to be expected. However, the APEs for both FbC-OS-regression models are slightly better than the APE for the linear model.

	Linear model	Separate effects FbC-OS-regression	Combined effect FbC-OS-regression
APE	.943	.930	.930

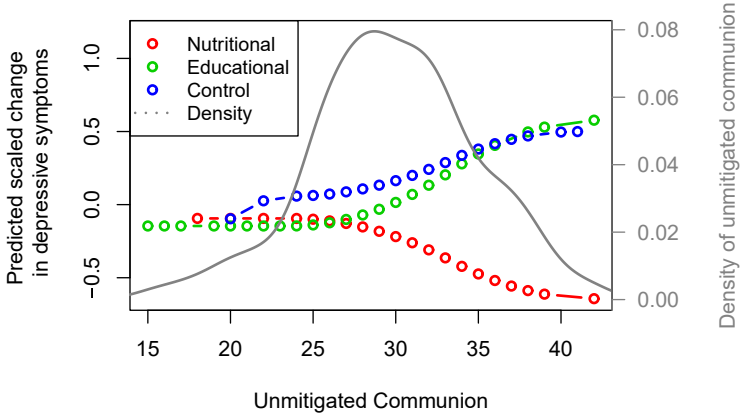
Table 4.2: Apparent prediction errors of the three models fitting the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*.



(a) Predicted values for the linear model with interactions



(b) Predicted values for the FbC-OS-regression model with separate main effects



(c) Predicted values for the FbC-OS-regression model with a combined total effect

Figure 4.4: Predicted value plots for the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*, for linear regression with interactions and both FbC-OS-regression models. A density estimate of *unmitigated communion* is also included. The axis for the density is displayed on the right.

4.4 Interaction effect of *number of comorbidities* and *experimental condition* on *change in physical functioning*

This section focuses on the interaction effect of the *number of comorbidities* and *experimental condition* on *change in physical functioning*. As with the previously fitted interaction effect, the two FbC-OS-regression models and the ordinary linear regression model are fitted and compared. Interpretation for this outcome is slightly different, however. Now, a positive outcome for *change in physical functioning* means an increase in physical functioning of the patient and a positive outcome is therefore desirable.

4.4.1 Separate effects FbC-OS-regression model

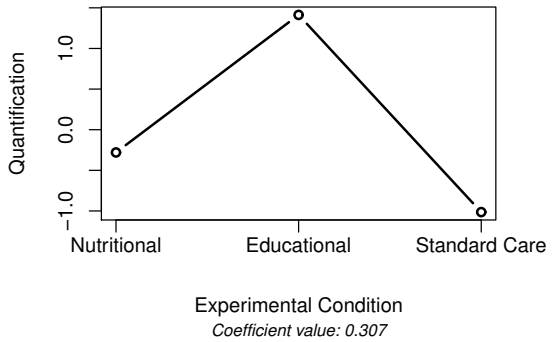
Quantification plots for the separate effects FbC-OS-regression model and the corresponding coefficients are shown in Figure 4.5. Coefficients for all effects are positive, with the *experimental condition* variable having the largest effect size. The *number of comorbidities* variable, however, has a relatively small coefficient, meaning that this main effect on its own has a small effect on the outcome. The effect sizes for the interaction effect and *experimental condition* are more similar.

The transformation plot for the interaction in Figure 4.5c shows in- and decreasing curves, indicating that an interaction effect is present, thus again complicating the interpretation of the quantifications. The quantifications for the *experimental condition* variable, plotted in Figure 4.5a, show that the educational intervention group is most strongly associated with an increase in *change in physical functioning*. However, for the interaction effect quantifications as shown in Figure 4.5c, the educational intervention group shows the lowest effect on the outcome on average.

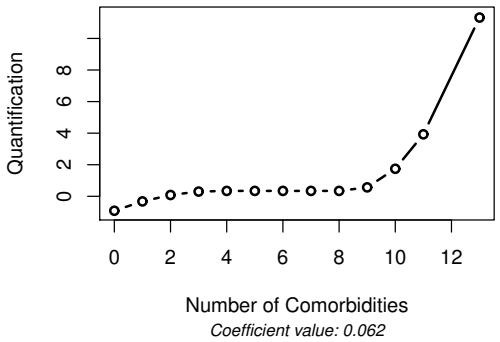
Figure 4.5b shows that the main effect of *number of comorbidities* is increasing. This is unintuitive, since a patient having more comorbidities seems likely to have a worse physical functioning. Note also the high quantified value for the last category; the transformed variables are Z-scores, i.e. having a mean of zero and a variance of one, so a quantification value of about 11 is extreme. This is an indication that the one patient in this category is an outlier that has caused the transformation to be dominated by the contrast between this patient, with the most comorbidities but not physically functioning badly, and all other patients. A possible remedy could be to merge the highest categories of the *number of comorbidities* variable. However, even with merged categories the quantifications range from about -1 to 4, indicating that the outlying patient has very little influence on the overall result. All in all, *number of comorbidities* has a low main effect and we will not involve this effect in the interpretation.

The interaction effects for the educational intervention and standard care groups, as opposed to the continuous main effect, do indicate higher predictions for patients with fewer comorbidities and lower predictions for patients with many comorbidities. The nutritional intervention has an inverse interaction effect compared to the other two treatments, that is, higher predictions for patients with the most comorbidities.

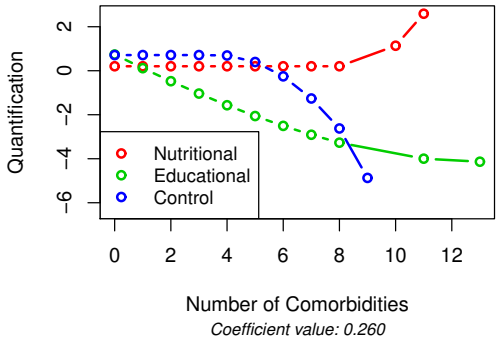
The educational intervention and standard care treatment groups can be interpreted to, overall, have a decreasing effect on *change in physical functioning* for patients with many comorbidities compared to patients having fewer comorbidities. The predictions for this model, as shown in Figure 4.7b, verify that the overall effects of the educational intervention and standard care treatment group show mostly decreasing trends. A stronger beneficial effect of the nutritional intervention for patients with many comorbidities, relative to the effect of the educational intervention and standard care treatments, can also be observed.



(a) Quantification plot for *experimental condition*



(b) Quantification plot for *number of comorbidities*



(c) Quantification plot for the interaction effect between *number of comorbidities* and *experimental condition*

Figure 4.5: Quantification plots of the fitted FbC-OS-regression model for the interaction effect of *number of comorbidities* and *experimental condition* on *change in physical functioning*, with separately fitted main effects. The algorithm converged after 26 iterations.

There are few patients with more than 5 comorbidities, however, as can be determined from the density estimate plotted in Figure 4.7b. Therefore, interpretations of the effect of the *number of comorbidities* variable on the outcome are more reliable in the range of 0 to 5 comorbidities. In this range, the nutritional intervention and standard care treatments are very similar and can both be interpreted to have a constant effect on the outcome, as opposed to these groups having an increasing and decreasing effect, respectively, when interpreted for the total range of *number of comorbidities*. Finally, the interpretation for the educational intervention is not as affected, it is still interpreted to have a more beneficial effect on *change in physical functioning* for patients with zero comorbidities compared to patients with multiple comorbidities.

4.4.2 Combined effect FbC-OS-regression model

The quantification plot and coefficient for the fitted combined effect FbC-OS-regression model can be found in Figure 4.6. The curves are very similar to the plot for the interaction effect of the separate effects FbC-OS-regression model as shown in Figure 4.2c. Patients with fewer comorbidities show similar results for the nutritional intervention and standard care treatment

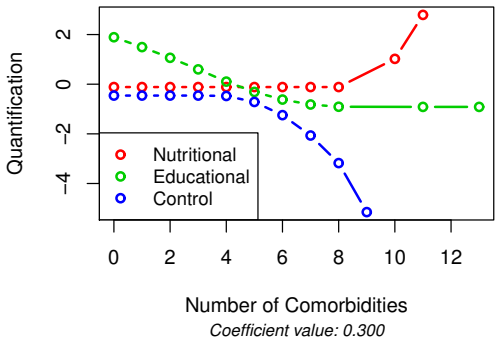


Figure 4.6: Quantification plot of the fitted combined effect FbC-OS-regression model for the interaction effect of *unmitigated communion* and *experimental condition* on *change in depressive symptoms*. The algorithm converged in 2 iterations.

groups on the *change in physical functioning*, whereas the educational intervention seems more beneficial for these patients. Patients with many comorbidities, however, seem to show a stronger difference between the three treatment groups compared to patients with fewer comorbidities, indicating an interaction effect between the two variables. Specifically, the nutritional intervention seems the most beneficial for physical functioning in patients with many comorbidities.

As was the case for the separate effects FbC-OS-regression model, there are few patients having more than 5 comorbidities. Thus, interpretation is more reliable in the range of 0 to 5 comorbidities. The educational intervention still shows a decreasing effect in this range; the effect on the *change in physical functioning* is more beneficial for patients with zero or very few comorbidities than patients having around five comorbidities. The nutritional intervention and standard care treatments result in a different interpretation in the range of 0 to 5 comorbidities compared to the total range. Both groups then show a constant effect on the outcome, i.e. a horizontal line, and this constant effect is very similar for the two groups. Finally, the coefficient, again, cannot be interpreted relatively, only numerically.

4.4.3 Model comparison

After fitting the two FbC-OS-regression models, an ordinary linear regression model is fitted as well, to compare the results for the three models. Predicted values for these models are shown in Figure 4.7. It can be seen that the predicted values for the two FbC-OS-regression models are very similar, except for the shape of the curve for the educational intervention, which is clearly nonmonotone in the separate effects FbC-OS-regression model. Since it separately restricts the main and interaction effects, a nonmonotone prediction curve is possible.

All three models resulted in linear functions for patients with fewer comorbidities, i.e. about 0 to 5 comorbidities, which is also the range containing the most observations. These linear functions do differ slightly for the three models. The two FbC-OS-regression models predict a nearly flat curve for patients with fewer comorbidities receiving either the nutritional intervention or standard care. The linear model, however, only predicts a nearly flat line for patients receiving the nutritional intervention group. The standard care treatment is a decreasing line, as opposed to being flat. This could lead to interpreting a difference between the control and nutritional intervention groups in patients with fewer comorbidities for the linear model, whereas the more flexible FbC-OS-regression models do not suggest such a difference. This difference in results

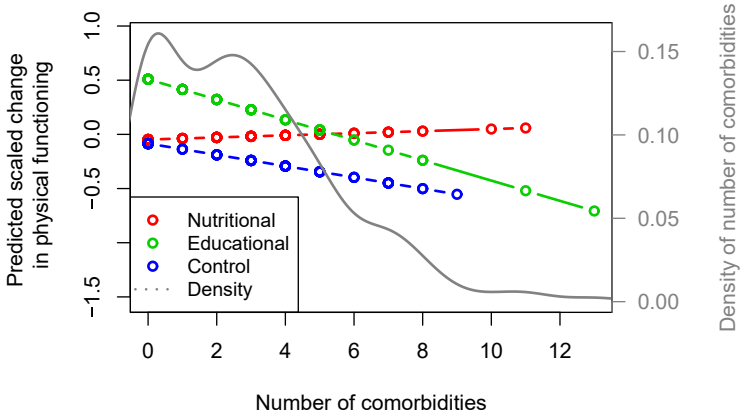
is caused by the restriction of the linear model to a linear relation, where extreme outcome values for patients with many comorbidities can also affect the predictions for patients with fewer comorbidities. The more flexible nonlinear curves are less affected in such cases and hence they may lead to a different conclusion on the relation between predictors and outcome. In this case, this conclusion is that the nutritional intervention and standard care have a similar effect on patients with few comorbidities.

The APEs for the three models are shown in Table 4.3. As is visible in the predicted value plots, both FbC-OS-regression models are very similar and therefore show, for the given precision, the same APE. The APE for the linear model is higher, which makes sense, since the other models are more flexible. Overall, the three APE values are very high, so the relation between outcome and predictors is not adequately described by any of the models. The two predictor variables used in this analysis possibly do not yield enough information to interpret the observed changes in physical functioning sufficiently.

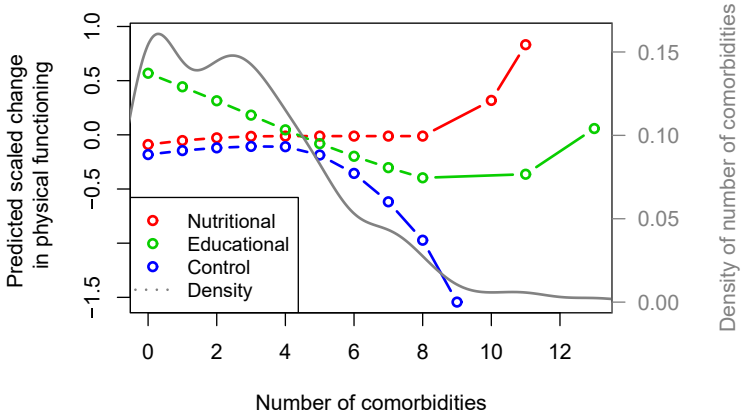
Overall, the interpretation of the predicted values for all models is that for patients with fewer comorbidities, the educational intervention type has the most positive effect on the *change in physical functioning*. For patients with more than 5 comorbidities, however, the effect cannot be properly interpreted due to the low number of observations.

	Linear model	Separate effects FbC-OS-regression	Combined effect FbC-OS-regression
APE	.931	.910	.910

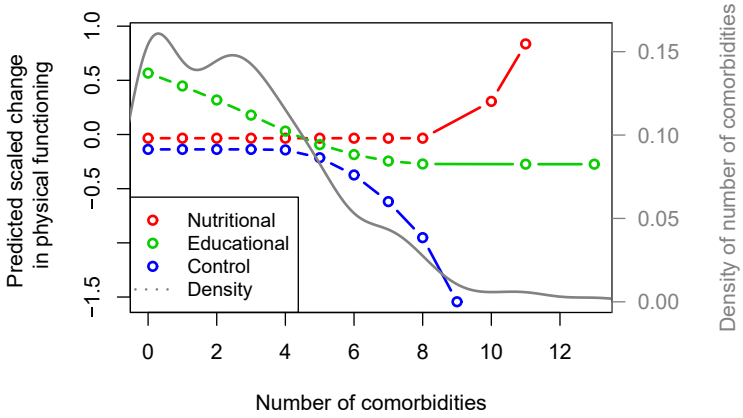
Table 4.3: Apparent prediction errors of the three models fitting the interaction effect of *number of comorbidities* and *experimental condition* on *change in physical functioning*.



(a) Predicted values for the linear model with interactions



(b) Predicted values for the FbC-OS-regression model with separate main effects



(c) Predicted values for the FbC-OS-regression model with a combined total effect

Figure 4.7: Predicted value plots for the interaction effect of *number of comorbidities* and *experimental condition* on *change in physical functioning*, for linear regression with interactions and both FbC-OS-regression models. A density estimate of the *number of comorbidities* is also included. The axis for the density is displayed on the right.

Chapter 5

Discussion

In this thesis, two approaches are presented to model Factor-by-Curve interactions in Optimal Scaling regression. With these FbC-OS-regression models, the interaction effect between a categorical predictor variable and a continuous predictor variable on an outcome of interest can be studied. This model allows for monotonic restrictions of main and interaction effects, a feature that is not available in other models for analysis of FbC interactions.

It is shown with two examples using a real data set that the presented FbC-OS-regression models are able to flexibly model main and interaction effects, either separately or combined. The flexibility of both FbC-OS-regression models resulted in a better fit to the data compared to the linear model, as based on their APEs.

5.1 Comparison of the two FbC-OS-regression models

The two FbC-OS-regression models themselves, while similar in many aspects, differ in the model definition, as one model includes the two main effects and the interaction effect as separate effects, while the other model includes them as a single combined effect. As a result, for each set of two interacting variables, the separate effects FbC-OS-regression model estimates three coefficients and three sets of quantifications, while the combined effect FbC-OS-regression model estimates a single coefficient and a single set of quantifications. Any (monotone) restrictions are thus also enforced on either the separate effects or the combined effect.

Since the variables and their interaction effects are modelled differently, the models can result in different predicted values. In both applications detailed in the previous chapter, however, differences between the predictions are very minor. The number of iterations needed to reach those similar predictions does differ. The separate effects FbC-OS-regression model takes longer to converge, since it needs to optimise two additional coefficients and sets of quantifications, which are all dependent on each other.

Another difference between the two models is the interpretation of the quantifications. The combined effect FbC-OS-regression model is simplest in interpretation. It provides only a single set of quantifications for the overall effect of the two variables on the outcome. In contrast, the separate effects FbC-OS-regression model results in multiple sets of quantifications. As a result, interpretation of the quantifications is more complex, since the main and interaction effects are conditional on each other. This means that any interpretation of a quantified variable depends on the other two variables. The same goes for the effect sizes of the separate effects. Although the interpretation is more complicated, a benefit of this model is that it does allow for separate examination and testing of the main and interaction effects, as opposed to the overall combined effect.

When extending either one of the FbC-OS-regression models to include any additional variables, besides the two interacting variables, interpretation becomes more complex. In that case, interpretation of the combined effect FbC-OS-regression model is still simpler compared to the separate effects FbC-OS-regression model. Since the combined interaction effect can be plotted on its own more easily, it can be interpreted separately from any variables not included in that interaction.

For both applications of the separate effects FbC-OS-regression model, the continuous main effect showed a very small coefficient. This was because the effect for the continuous variable was mostly included in the interaction effect, as is indicated by the close resemblance of the plots for the interaction effects in the separate effects model to the plots for the combined effect model. Whether this is also the case for other data sets was not investigated.

5.2 Comparison with additive models

There are multiple differences between the two FbC-OS-regression models studied in this thesis and existing statistical methods that implement analysis of FbC interactions, such as the additive model. A few differences with the additive model are discussed here.

Interpretation of the FbC interaction in both additive and OS-regression models is based on comparison of the curves. However, a difference concerning the curves is that, in additive models as implemented in the `mgcv` package, a FbC interaction is fitted with only a separate main effect for the factor and a combined effect of continuous variable and interaction, resulting in two separate terms. In other words, when compared to the separate effects FbC-OS-regression with three terms, one less term is included, namely the separate term for the continuous variable. Curves are then centered individually, and therefore the main effect of the factor variable is required as a separate effect. As a result of the individual centering, the average effect of each factor level can be interpreted more easily, since it is not conditional on the interaction effect. Alternatively, the additive model can fit an overall combined effect, similar to the combined effect FbC-OS-regression model.

The methods used for fitting the smooth functions for the additive and OS-regression models are also different. Multiple types of smooth functions are available in the additive model implementation of the `mgcv` package. These smooth functions include, among others, several variants of spline functions. All smooth functions for the additive model are generally non-monotone. In contrast, (FbC-)OS-regression specifically uses I-splines (Ramsay, 1988), which allow for straightforward restriction to monotone spline functions. The overall concept for the smooth transformation functions is thus quite similar, but due to differences in the use of smooth functions, the results for the two models can differ, even if the model terms are similar.

5.3 Directions for future study

In this thesis, an extended implementation of OS-regression is proposed that allows for analysis of FbC interactions. However, multiple aspects of the proposed models should be studied further and extended, in order to, for example, assess the performance of the models, and to validate the chosen implementation of the models as presented in this thesis. A few aspects are discussed in subsequent paragraphs.

Firstly, the current implementation of the FbC-OS-regression models does not validate the model fit. Implementing a bootstrap or cross-validation procedure would be beneficial for testing stability of the model estimates, and accuracy of prediction for test data.

Secondly, the implementations of both FbC-OS-regression models as presented in this thesis allow an exploratory analysis of FbC interactions, by visually determining differences between the

resulting curves. However, the significance of these differences cannot be tested yet. Thus, even if the curve functions differ strongly from each other and, consequently, imply an interaction effect, no strict conclusions can be drawn. To be able to interpret significance of the results implied by the fitted FbC-OS-regression models, some form of hypothesis testing is necessary.

Thirdly, while three-way interactions and interactions of even higher orders are not used as often as two-way interactions, the model could be extended to allow for such interactions. An interaction between two categorical variables and a continuous variable, for example, can be modelled as an extended FbC interaction, where a curve could be fitted for each combination of the two categorical variables.

Finally, the apparent intermingling of the main effect with the interaction effect quantifications, as observed in the separate effects FbC-OS-regression model, needs further investigation.

5.4 Conclusions

Overall, the presented implementations of the Factor-by-Curve Optimal Scaling regression models are useful methods to visually analyse Factor-by-Curve interactions, drawing on advantages of the Optimal Scaling methodology. The Factor-by-Curve Optimal Scaling regression models are mainly favourable for analyses requiring monotone restrictions, either for main or interaction effects. While many aspects of these models still require further study, this thesis sets a starting point for further implementation of Optimal Scaling regression models for the analysis of Factor-by-Curve interactions.

Bibliography

Baker, S. P.
1992. *The Injury Fact Book.*, volume 2nd ed. Oxford University Press.

Breiman, L. and J. H. Friedman
1985. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–598.

Chen, Z.
1993. Fitting multivariate regression functions by interaction spline models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(2):473–491.

Coull, B. A., D. Ruppert, and M. P. Wand
2001. Simple incorporation of interactions into additive models. *Biometrics*, 57:539–45.

de Leeuw, W., F. Young, and Y. Takane
1976. Additive structure in qualitative data: An alternating least squares method with optimal scaling features. *Psychometrika*, 41:471–503.

Friedman, J. H.
1991. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67.

Friedman, J. H. and W. Stuetzle
1981. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

Friedrich, R. J.
1982. In defense of multiplicative terms in multiple regression equations. *American Journal of Political Science*, 26(4):797–833.

Gifi, A.
1990. *Nonlinear multivariate analysis*, Wiley series in probability and mathematical statistics, [rev. ed.]. edition. Chichester [etc.]: Wiley.

Guttman, L.
1941. The quantification of a class of attributes: A theory and method of scale construction. *The Prediction of Personal Adjustment.*

Hastie, T. and R. Tibshirani
1990. *Generalized additive models*, Monographs on statistics and applied probability; 43 842820124. London [etc.]: Chapman and Hall.

Jackman, R.
1974. Political democracy and social equality: a comparative analysis. *American sociological review*, 39(1):29–45.

Kruskal, J. B.
1965. Analysis of factorial experiments by estimating monotone transformations of the data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 27(2):251–263.

MacCullagh, P. and J. Nelder
1989. *Generalized linear models*, Monographs on statistics and applied probability; 37 842820124, 2nd ed.. edition. London [etc.]: Chapman & Hall.

Meulman, J., A. van Der Kooij, and K. Duisters
2019. Ros regression: Integrating regularization with optimal scaling regression. *Statistical Science*, P. 361.

Michaelis, L. and M. L. Menten
1913. Kinetik der invertinwirkung. *Biochemische Zeitung*, 49:333369.

Nelder, J. A. and R. W. M. Wedderburn
1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384.

Qi, L., M. C. Cornelis, C. Zhang, R. M. van Dam, and F. B. Hu
2009. Genetic predisposition, Western dietary pattern, and the risk of type 2 diabetes in men. *The American Journal of Clinical Nutrition*, 89(5):1453–1458.

R Core Team
2019. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ramsay, J. O.
1988. Monotone regression splines in action. *Statistical Science*, 3(4):425–441.

Scheier, M. F., V. S. Helgeson, R. Schulz, S. Colvin, S. Berga, M. W. Bridges, J. Knapp, K. Gerszten, and W. S. Pappert
2005. Interventions to enhance physical and psychological functioning among younger women who are ending nonhormonal adjuvant treatment for early-stage breast cancer. *Journal of clinical oncology: official journal of the American Society of Clinical Oncology*, 23(19):4298.

Scheier, M. F., V. S. Helgeson, R. Schulz, S. Colvin, S. L. Berga, J. Knapp, and K. Gerszten
2007. Moderators of interventions designed to enhance physical and psychological functioning among younger women with early-stage breast cancer. *Journal of clinical oncology: official journal of the American Society of Clinical Oncology*, 25(36):5710.

Sestelo, M., N. M. Villanueva, L. Meira-Machado, and J. Roca-Pardias
2017. npregfast: An r package for nonparametric estimation and inference in life sciences. *Journal of Statistical Software*, 82(1):1–27.

Smith, K. W. and M. Sasaki
1979. Decreasing multicollinearity: A method for models with multiplicative functions. *Sociological Methods & Research*, 8(1):35–56.

Southwood, K. E.
1978. Substantive theory and statistical interaction: Five models. *American Journal of Sociology*, 83(5):1154–1203.

Stone, C. J.
1985. Additive regression and other nonparametric models. *The Annals of Statistics*, 13(2):689–705.

van der Kooij, A. J.
2007. Prediction accuracy and stability of regression with optimal scaling transformations.

van der Kooij, A. J. and J. J. Meulman
1998. Regression with optimal scaling. In *SPSS Categories 8.0*, J. J. Meulman, W. J. Heiser, and SPSS Inc., eds., Pp. 1–8, 77–101, 239–246. Chicago: SPSS Inc.

van der Lans, I. A. and W. J. Heiser
1988. Nonlinear analysis of multiplicative rules in expectancy-value models.

van Rosmalen, J., A. J. Koning, and P. J. F. Groenen
2009. Optimal scaling of interaction effects in generalized linear models. *Multivariate Behavioral Research*, 44(1):59–81.

Wahba, G.
1990. Additive and interaction splines. In *Spline Models for Observational Data*, Pp. 127–133. Society for Industrial and Applied Mathematics.

Wang, W. and J. Yan
2018. *splines2: Regression Spline Functions and Classes*. R package version 0.2.8.

Winsberg, S. and J. O. Ramsay
1980. Monotonic transformations to additivity using splines. *Biometrika*, 67(3):669–674.

Wood, S. N.
2006. Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics*, 62(4):1025–1036.

Wood, S. N.
2017. *Generalized Additive Models: An Introduction with R*, 2 edition. Chapman and Hall/CRC.

Wright, G.
1976. Linear models for evaluating conditional relationships. *American Journal of Political Science*, 20(2):349.

Young, F., W. de Leeuw, and Y. Takane
1976. Regression with qualitative and quantitative variables: An alternating least squares method with optimal scaling features. *Psychometrika*, 41:505–529.

Appendix A

R Code

A.1 Functions

A.1.1 Supplementary

```r
autoscale <- function(x) {
  # Short function which standardizes over N instead of N-1.
  x <- as.matrix(x)
  N <- nrow(x)
  scaled <- scale(x)

  for(k in 1:ncol(scaled)){
    if(all(is.nan(scaled[,k]))){
      scaled[,k] <- 0
    }
  }

  autoscaled <- scaled * sqrt(N/(N-1)) # correct for N-1
  return(autoscaled)
}

library(splines2)
fitNonMonotoneSpline <- function(unrestrictedQuantifications,
                                 splineBase,
                                 weightsQuantifications =
                                   rep(1, length(unrestrictedQuantifications)))
  # Respective inputs represent:
    # Current unrestricted quantifications
    # Spline base fitted at beginning of algorithm
    # Vector weights of categories
  {
    splineFit <- lm(unrestrictedQuantifications ~ splineBase,
                    weights = weightsQuantifications)

    splineResult <- predict(splineFit)
    splineCoeffs <- splineFit$coef

    return(list(restrictedQuantifications = splineResult,
                splineCoefficients = splineCoeffs))
```

```r
  }

fitMonotoneSpline <- function(unrestrictedSplineQuantifications,
                              splineBase,
                              initialSplineCoefficients,
                              weightsQuantifications =
                                rep(1, length(unrestrictedSplineQuantifications)),
                              maxSplineIter = 2,
                              FbC = FALSE)
  # Respective inputs represent:
    # Current unrestricted quantifications
    # Spline base fitted at beginning of algorithm
    # Initial spline coefficients (from previous updates)
    # Vector weights of categories
    # Maximum number of iterations for fitting the spline
    # Whether to use the procedure checking for the optimal monotone spline fit

  {
  # Create column for intercept:
  splineBase <- cbind(1, splineBase)

  coeffs <- initialSplineCoefficients
  if(FbC) coeffs2 <- initialSplineCoefficients

  for(j in 1:maxSplineIter){
    coeffsPrevious <- coeffs

    for(l in 1:length(coeffs)){
      # Create partial residual u_l
      ul <- unrestrictedSplineQuantifications -
        as.matrix(splineBase[,-l])%*%coeffs[-l]
      linearFit_l <- lm(ul ~ -1 + splineBase[,l],
                        weights = weightsQuantifications)


      # for all coefficients except for the intercept, set to 0 if negative
      if(l == 1){
        coeffs[l] <- linearFit_l$coefficients
      } else{
        coeffs[l] <- max(0, linearFit_l$coefficients)
      }
    }

    # check convergence:
    if(sum((coeffsPrevious - coeffs)^2) < 0.000000001){
      break # stop if converged
    }
  }

  if(FbC){

    for(j in 1:maxSplineIter){
      coeffsPrevious <- coeffs2
```

```
88
89        for(l in 1:length(coeffs2)){
90          # Create partial residual u_l
91          ul <- unrestrictedSplineQuantifications -
92            as.matrix(splineBase[,-l])%*%coeffs2[-l]
93          linearFit_l <- lm(ul ~ -1 + splineBase[,l],
94                            weights = weightsQuantifications)
95
96
97          # for all coefficients except for the intercept, set to 0 if negative
98          if(l == 1){
99            coeffs2[l] <- linearFit_l$coefficients
100         } else{
101           coeffs2[l] <- min(0, linearFit_l$coefficients)
102         }
103       }
104
105       # check convergence:
106       if(sum((coeffsPrevious - coeffs2)^2) < 0.000000001){
107         break # stop if converged
108       }
109     }
110
111     losspos <- sum((unrestrictedSplineQuantifications - splineBase%*%coeffs)^2)
112     lossneg <- sum((unrestrictedSplineQuantifications - splineBase%*%coeffs2)^2)
113
114     if(losspos <= lossneg){
115       return(list(restrictedQuantifications = splineBase%*%coeffs,
116                   splineCoefficients = coeffs))
117     } else{
118       return(list(restrictedQuantifications = splineBase%*%coeffs2,
119                   splineCoefficients = coeffs2))
120     }
121   }
122
123
124   return(list(restrictedQuantifications = splineBase%*%coeffs,
125               splineCoefficients = coeffs))
126 }
127
128 restrictOrdinal <- function(unrestrictedQuantifications,
129                             cat.weights =
130                               rep(1, length(unrestrictedQuantifications))){
131   # Function to restrict quantifications to ordinal (monotone)
132
133   C <- length(unrestrictedQuantifications) # Number of categories
134
135   # Start with unrestricted quantifications:
136   restrictedQuantifications <- unrestrictedQuantifications
137
138   groups <- 1:C
139
140   finished <- FALSE
```

```
141
142   while(!finished){
143     for(i in 2:C){
144       if(restrictedQuantifications[i] >= restrictedQuantifications[i-1]){
145         # Skip iteration if i'th and (i-1)'th quantification are monotone
146         next()
147       }
148
149       # if order is incorrect, group i with i-1:
150       groups[i-1] <- groups[i]
151
152       # Then the value for each of the members of that group will get the same value
153       # (= weighted average)
154
155       groupMembers <- which(groups == groups[i])
156
157       restrictedQuantifications[groupMembers] <-
158         weighted.mean(unrestrictedQuantifications[groupMembers],
159                       cat.weights[groupMembers])
160
161     }
162
163     if(!is.unsorted(restrictedQuantifications)){
164         finished <- TRUE
165     }
166   }
167
168   return(restrictedQuantifications)
169
170 }
171
172 SLcheck <- function(ScalingLevels){
173   # Function to check if any of the inputted scaling levels is incorrect
174   levelnames <- c("nominal", "numeric", "nonmonotonespline",
175                   "monotonespline", "ordinal", "fbc")
176   results <- sapply(1:length(ScalingLevels), function(k){
177     !(tolower(ScalingLevels[k]) %in% levelnames)
178   })
179
180   return(any(results))
181 }
```

A.1.2 FbC-OS-regression

A.1.2.1 Separate effects FbC-OS-regression model

```
1 FbC.OS.SE <- function(formula, fcmat = NULL, data,
2                       SL = rep("Nominal", length(all.vars(formula))-1),
3                       nDegrees, nInteriorKnots, crititer=0.00001, maxiter=1000) {
4   # Function to perform Optimal Scaling on independent variables, including the
5   #   possibility for Factor-by-Curve interactions.
6   # Optimal scaling level of the outcome is restricted to "numeric".
7   #
8   # Arguments:
```

```
#   formula: an object of class "formula": a symbolic description of the model
#            to be fitted.
#   fcmat: a matrix with, for each row, five values, the first number indicating
#          the factor variable and the second indicating the curve variable. Third
#          and fourth are spline degree and interior knots for the curve. The
#          fifth value indicates whether to fit nonmonotone (1) or monotone (2)
#          interaction curves.
#   data: a data frame containing the variables in the model.
#   SL: a vector containing the scaling levels, all nominal if not specified
#   nDegrees: a vector containing the degrees of the splines for the main effects
#   nInteriorKnots: a vector containing the number of interior knots of the
#                   splines for the main effects
#   crititer: critical value for the decrease in loss between two iterations
#   maxiter: maximum number of iterations
#
# Returns:
#   a list object with: a) The final regression coefficients for any main effects
#                       b) The final regression coefficients for the interactions
#                       c) Total sum of squares
#                       d) Regression sum of squares
#                       e) Apparent prediction error (APE)
#                       f) Loss for each iteration
#                       g) The category quantifications
#                       h) Some output that can be used for plots

# Perform some preliminary tests, checking argument validity
if (class(formula)!="formula"){
  stop("Argument formula is not of the right class.")
} else if (!is.data.frame(data)){
  stop("Argument data is not a dataframe.")
} else if (!is.vector(SL) | (length(SL) == 0)){
  stop("Argument SL is not a vector or has a length of zero.")
} else if (SLcheck(SL)){
  stop("Argument SL contains unrecognised scaling levels.
       Use only the following (case insensitive):
       Nominal, Ordinal, Numeric, NonMonotoneSpline or MonotoneSpline")
}

# Construct a variable for the outcome and a matrix with the predictors
outcome.name <- all.vars(formula)[1]
pred.names <- all.vars(formula)[-1]

N <- nrow(data) # Number of observations

y <- data[, outcome.name] # Outcome, to be called y in the rest of the code
y <- autoscale(y) # Use numeric scaling level for the outcome (i.e. standardise)

X <- as.data.frame(data[, pred.names]) # Independent variables, not scaled
colnames(X) <- pred.names

P <- length(pred.names) # Number of predictors


```

```
62  SLnum <- # Vector of which variables are to be scaled numerically
63    sapply(SL, function(k) tolower(k) == "numeric")
64
65
66  categories <- # Category labels
67    lapply(pred.names, function(k) sort(unique(X[, k])))
68  nr.categories <- # Number of categories for each variable
69    lapply(pred.names, function(k) length(unique(X[, k])))
70
71  # If interaction is included, specify indices for variables to include
72  # as interactions
73  if(!is.null(fcmat)) {
74    fcmat <- matrix(fcmat, ncol = 5) # Ensure matrix format
75    intinds <- matrix(fcmat[,1:2], ncol = 2)
76  } else {
77    intinds <- matrix(NA, ncol = 2, nrow = 1)
78  }
79
80  # Numeric values for each category for all observations
81  XNum <- lapply(1:P, function(k) {
82    if(is.numeric(X[, pred.names[k]])){
83      return(X[,pred.names[k]])
84    } else {
85      return(as.numeric(as.factor(X[,pred.names[k]])))
86      # As.factor() also sorts the order, ensuring the same order as
87      # for the indicator matrix
88    }
89  })
90
91  # Numeric values for each combination of category and numeric value
92  # for all observations for each Factor-by-Curve
93  if(!is.null(fcmat)){
94    XNum_FbC <- lapply(1:nrow(fcmat), function(k){
95      fctr <- intinds[k,1]; crv <- intinds[k,2]
96      comb.factor <- interaction(X[,fctr],X[,crv], drop = TRUE,
97                                 sep = "@", lex.order = TRUE)
98      return(as.numeric(comb.factor))
99    })
100 }
101
102 # Unique numerical values for categories
103 categoriesnum <- lapply(XNum, function(k) sort(unique(k)))
104
105 # Unique numerical values for each combination of category and numeric value
106 # for Factor-by-Curve
107 if(!is.null(fcmat)){
108   categories_FbC <- lapply(1:nrow(fcmat), function(k) {
109     fctr <- intinds[k,1]; crv <- intinds[k,2]
110     comb.factor <- interaction(X[,fctr],X[,crv], drop = TRUE,
111                                sep = "@", lex.order = TRUE)
112     return(as.character(levels(comb.factor)))
113   })
114 }
```

```
115
116 # Initialize transformed variables as autoscaled numeric variables
117 PhiX <- matrix(NA, nrow = N, ncol = P)
118 Fctrs <- Crvs <- MEs <- NULL
119
120 for(k in 1:P){
121   PhiX[,k] <- autoscale(XNum[[k]])
122 }
123
124 # Initialize a transformed variable for the interaction(s) as a vector of zeroes
125 if(!is.null(fcmat)){
126   PhiX_FbC <- matrix(NA, nrow = N, ncol = nrow(fcmat))
127   for(k in 1:nrow(fcmat)){
128     PhiX_FbC[,k] <- rep(0, N)
129   }
130 } else {
131   PhiX_FbC <- NULL
132 }
133
134
135 # Construct the indicator matrices, as vectors containing indices
136 # to reduce redundancy
137 g <- lapply(1:P, function(k) return(as.numeric(as.factor(XNum[[k]]))))
138
139 if(!is.null(fcmat)){
140   g_FbC <- lapply(1:nrow(fcmat), function(k) as.numeric(as.factor(XNum_FbC[[k]])))
141 }
142
143 # And the univariate marginals
144 d <- lapply(1:P, function(k){
145   ifelse(!SLnum[k], return(as.numeric(table(XNum[[k]]))), return(NULL))
146 })
147
148 # Initialize the coefficients:
149 betas <- numeric(P) + 1
150 betasIE <- matrix(0, nrow = P, ncol = P)
151
152 if(!is.null(fcmat)){
153   for(i in split(intinds, seq(nrow(intinds)))){
154     betasIE[i[1],i[2]] <- 1
155   }
156 }
157
158
159 # Initialize splines for main effects if necessary
160 splineBases <- list()
161 splineCoefficients <- vector("list", P)
162
163 for(k in 1:P){
164   if(tolower(SL[k]) %in% c("nonmonotonespline","monotonespline")){
165     # Use quantiles for selecting the knot placement
166       # Using unique categories, with unweighted knot placement as a result
167     interiorKnots_k <- quantile(categoriesnum[[k]],
```

```
168                                   probs =
169                                     seq(0,1, length.out = nInteriorKnots[k] +
    2),
170                                   type = 6)[-c(1, nInteriorKnots[k] +2)]
171                                            # +2 to exclude the boundaries
172
173       splineBases[[k]] <- iSpline(XNum[[k]],
174                                   knots = interiorKnots_k,
175                                   degree = nDegrees[k]-1,
176                                     # Degree -1, due to usage of I-splines
177                                   intercept = T)
178     } else {
179       splineBases[[k]] <- NULL
180     }
181   }
182
183
184   # Initialize splines for interaction effects if included in the model
185   if(!is.null(fcmat)){
186     splineBasesFbC <- list()
187     splineCoefficientsFbC <- list()
188     row.indices <- list()
189
190     for(l in 1:nrow(intinds)){
191       a <- intinds[l,1]; b <- intinds[l,2]
192       row.indices[[l]] <- list()
193       splineBasesFbC[[l]] <- list()
194       splineCoefficientsFbC[[l]] <- list()
195
196       # Use separate spline bases for each factor level
197       for(m in 1:nr.categories[[a]]){
198         row.indices[[l]][[m]] <- rows <- which(g[[a]]==m)
199
200         interiorKnots_k <- quantile(categoriesnum[[b]],
201                                     probs = seq(0,1, length.out = fcmat[l,4] +
    2),
202                                     type = 6)[-c(1, fcmat[l,4] +2)]
203
204         splineBasesFbC[[l]][[m]] <- iSpline(XNum[[b]][rows],
205                                             knots = interiorKnots_k,
206                                             degree = fcmat[l,3]-1,
207                                             intercept = T)
208       }
209     }
210   }
211
212
213   # Initialize a vector to save the loss (=squared euclidean norm)
214   # after each iteration
215   loss <- numeric(maxiter+1)
216   # Use as starting value for the loss:
217   loss[1] <- Inf
218
```

```r
# We set our index i at 2, because we already used the first element
# of the loss vector
i <- 2 # Note that the actual iteration number of our algorithm is i - 1

# Convergence variable to indicate convergence (due to either reaching
# max number of iterations or reaching the loss criterion)
conv <- FALSE


# Start of iterative algorithm
while(!conv) { # I.e. while not converged

  # Main effects #
  for (j in 1:P) {

    if(P == 1){
      u <- y
    } else {
      # Calculate partial residuals
      exj <- (1:P)[-j]

      # Partial residual for main effect(s)
      exj_terms <- lapply(exj, function(k) return(as.matrix(betas[k]*PhiX[,k])))

      if(!is.null(fcmat)){
        # Partial residual for interaction effect(s)
        exjIE_terms <- lapply(1:nrow(fcmat), function(k) {
          a <- fcmat[k,1]; b <- fcmat[k,2]
          return(as.matrix(betasIE[a,b] * PhiX_FbC[,k]))
        })
        # Sum partial regression terms and calculate the residual
          # (including interaction effects in u)
        u <- y - Reduce("+", exj_terms) - Reduce("+",exjIE_terms)

      } else {
        # Sum partial regression terms and calculate the residual
          # (no interaction effects in u)
        u <- y - Reduce("+", exj_terms)
      }
    }



    # Calculate quantifications and/or betas for given scaling level of variable j
    if(!SLnum[j]){
      ## Quantifications ##

      ### Nominal ###

      # Calculation of updated transformed variable:
      Phix_j <- u # Fully unrestricted phi_j(x_j)

```

```
271         # Update quantifications
272         vs <- aggregate(Phix_j, XNum[j], mean)
273
274         # Construct new phi_j(x_j), which is now restricted to nominal
275         nominal.phi <- as.numeric(vs[g[[j]],2])
276
277         restricted.phi <- nominal.phi
278
279         if(tolower(SL[j]) == "nonmonotonespline"){
280           ### NonMonotone Spline ###
281
282           # Restrict transformed variable using nonmonotone spline
283           restricted.phi <- fitNonMonotoneSpline(nominal.phi,
284                                                  splineBases[[j]]
285                                                  )$restrictedQuantifications
286
287         } else if(tolower(SL[j]) == "monotonespline"){
288           ### Monotone Spline ###
289
290           # Restrict transformed variable using monotone spline
291           if(i == 2){
292             # In the first iteration, use the splineCoefficients from the nominal
293             # spline solution as starting values for the ordinal spline fit.
294             # After the first iteration, continue with coefficients from previous
295             # ordinal updates.
296
297             splinefit <- fitNonMonotoneSpline(nominal.phi,
298                                               splineBases[[j]])
299
300             splineCoefficients[[j]] <- splinefit$splineCoefficients
301           }
302
303           splinefit <- fitMonotoneSpline(nominal.phi,
304                                          splineBases[[j]],
305                                          splineCoefficients[[j]],
306                                          maxSplineIter = ifelse(i == 2, 200, 2))
307
308           splineCoefficients[[j]] <- splinefit$splineCoefficients
309
310           if(all(splineCoefficients[[j]][-1] == 0)){
311             # If all spline coefficients are zero, the direction of the
312             # monotone function should be changed. This is done by changing
313             # the sign of the corresponding transformed variable and spline
314             # coefficients.
315
316             splinefit <- fitMonotoneSpline(-nominal.phi,
317                                            splineBases[[j]],
318                                            -splineCoefficients[[j]])
319
320             splineCoefficients[[j]] <- splinefit$splineCoefficients
321           }
322
323           restricted.phi <- splinefit$restrictedQuantifications
```

```
324
325         } else if (tolower(SL[j]) == "ordinal"){
326           ### Ordinal ###
327
328           # Restrict quantifications to monotonic increasing
329           restricted.v <- restrictOrdinal(vs[,2], d[[j]])
330
331           # If first and last quantifications are equal (i.e. all
332           # quantifications are equal), inverse the sign of the quantifications
333           # and restrict those to monotonic increasing
334           if(restricted.v[1] == restricted.v[length(restricted.v)]){
335
336             restricted.v <- restrictOrdinal(-vs[,2], d[[j]])
337           }
338
339           # Construct transformed variable from the ordinal quantifications
340           restricted.phi <- restricted.v[g[[j]]]
341         }
342
343
344         # Standardize PhiX: s.t. mean(PhiX) = 0, sigma(PhiX) = 1, ||PhiX||^2 =
    N
345         PhiX[,j] <- autoscale(restricted.phi)
346
347       }
348
349       ## Betas ##
350
351       betas[j] = sum(PhiX[,j] * u) / N
352     }
353
354
355     # Factor by Curve #
356     if(!is.null(fcmat)){
357
358       # Determine main effects
359       ME_terms <- lapply(1:P, function(k) return(as.matrix(betas[k]*PhiX[,k])))
360
361       for(l in 1:nrow(fcmat)){
362
363         # Sum partial regression terms and calculate the residual
364           # (without interaction effects in u)
365         u <- y - Reduce("+", ME_terms)
366
367         if(nrow(fcmat) > 1){
368           # Include interaction terms in partial residual,
369           # if more than one is modelled
370           exjIE_terms <- lapply((1:nrow(fcmat))[-l], function(k){
371             a <- intinds[k,1]; b <- intinds[k,2]
372             return(as.matrix(betasIE[a,b] * PhiX_FbC[,k]))
373           })
374
375           u <- u - Reduce("+", exjIE_terms)
```

```
376         }
377
378         fctr <- intinds[l,1] # Nr of variable which is the factor
379         crv <- intinds[l,2] # Nr of variable which is the continuous variable
380
381         PhiX_FbC_j <- u # unrestricted transformed variable
382
383         vs <- aggregate(PhiX_FbC_j, XNum_FbC[l], mean) # Nominal quantifications
384
385         nominal.phi <- vs[g_FbC[[l]],2] # Nominal transformed variable
386
387         restricted.phi <- nominal.phi
388
389         for(m in 1:nr.categories[[fctr]]){
390
391           # Specify the rows relevant for the given factor level
392           rows <- row.indices[[l]][[m]]
393
394
395           if(tolower(fcmat[l,5]) == 1){
396
397             ### Nonmonotone spline ###
398
399
400             # Restrict observations under given factor level using
401             # nonmonotone spline
402             splineFit <- fitNonMonotoneSpline(nominal.phi[rows],
403                                               splineBasesFbC[[l]][[m]])
404
405             restricted.phi[rows] <- splineFit$restrictedQuantifications
406           } else if(tolower(fcmat[l,5]) == 2){
407
408             ### Monotone spline ###
409
410             # Restrict observations under given factor level using monotone spline
411
412             if(i == 2){
413               # In the first iteration, use the splineCoefficients from the nominal
414               # spline solution as starting values for the ordinal spline fit.
415               # After the first iteration, continue with coefficients from previous
416               # ordinal updates.
417               splineFit <- fitNonMonotoneSpline(nominal.phi[rows],
418                                                 splineBasesFbC[[l]][[m]])
419
420               splineCoefficientsFbC[[l]][[m]] <- splineFit$splineCoefficients
421             }
422
423             splinefit <- fitMonotoneSpline(nominal.phi[rows],
424                                            splineBasesFbC[[l]][[m]],
425                                            splineCoefficientsFbC[[l]][[m]],
426                                            maxSplineIter = ifelse(i ==
    2, 200, 2),
427                                            FbC = TRUE)
```

```
428
429             # Save coefficients for next iteration
430             splineCoefficientsFbC[[l]][[m]] <- splinefit$splineCoefficients
431
432             restricted.phi[rows] <- splinefit$restrictedQuantifications
433           }
434         }
435
436         PhiX_FbC[,l] <- autoscale(restricted.phi) # Standardize PhiX_FbC as a whole
437
438         betasIE[fctr,crv] <- t(PhiX_FbC[,l])%*%u / N
439
440         if(fcmat[l,5] == 2 && sign(betasIE[fctr,crv]) == -1){
441           # If interaction beta is negative, the direction of the
442           # beta and all monotone functions should be changed. This is
443           # done by changing the sign of the corresponding beta, spline
444           # coefficients and transformed variable
445
446           betasIE[fctr,crv] <- -1 * betasIE[fctr,crv]
447
448           for(m in 1:nr.categories[[fctr]]){
449             rows <- row.indices[[l]][[m]]
450
451             splineCoefficientsFbC[[l]][[m]] <- -splineCoefficientsFbC[[l]][[m]]
452             PhiX_FbC[rows,l] <- -PhiX_FbC[rows,l]
453           }
454         }
455
456       }
457     }
458
459
460
461     # Calculate the loss: sum of squares between y and y.hat
462
463     # Determine y.hat
464     y.hat <- lapply(1:P, function(k) return(as.matrix(PhiX[,k]*betas[k])))
465     y.hat <- Reduce("+", y.hat)
466
467
468     #If interactions are specified, add those effects to the prediction
469     if(!is.null(fcmat)){
470       y.hat <- y.hat + Reduce("+",lapply(1:nrow(fcmat), function(k) {
471         a <- fcmat[k,1]; b <- fcmat[k,2]
472         return(as.matrix(betasIE[a,b] * PhiX_FbC[,k]))
473       }))
474     }
475
476     loss[i] <-  sum((y-y.hat)^2)
477
478     # Check if convergence criteria are met:
479     # Number of maximum iterations is met
480     # OR
```

```
481     # Difference in loss between previous and current solution is small enough
482
483     conv <- ((abs(loss[i-1]-loss[i]) < crititer) | i-1 >= maxiter)
484
485     # Increase the iteration number
486     i <- i + 1
487   }
488
489   # Do not store the first loss value (as we set it to Inf)
490   loss <- loss[2:(i-1)]
491   names(betas) <- pred.names
492   rownames(betasIE) <- pred.names
493   colnames(betasIE) <- pred.names
494
495   # Determining total sum of squares and APE
496   sstot <- N
497   ape <- loss[i-2]/sstot
498
499   # Create output of the quantifications
500   v.output <- lapply(1:P, function(k) {
501     vs <- aggregate(PhiX[,k], XNum[k], mean)
502     return(data.frame(categories=categories[[k]],
503                       quantifications = as.numeric(vs[,2])))
504   })
505
506   if(!is.null(fcmat)){
507     v.output_FbC <- lapply(1:(nrow(fcmat)), function(k){
508       vs <- aggregate(PhiX_FbC[,l], XNum_FbC[l], mean)
509       return(data.frame(categories=categories_FbC[[k]],
510                         quantifications = as.numeric(vs[,2])))
511     })
512
513     FbC.names <- sapply(1:nrow(fcmat), function(k) {
514       return(paste(pred.names[fcmat[k,1]],pred.names[fcmat[k,2]], sep =
    "."))
515     })
516
517   } else {
518     v.output_FbC <- NULL
519     FbC.names <- NULL
520     plot.output.FbC <- NULL
521     plot.output.total <- NULL
522   }
523
524
525   v.output <- append(v.output, v.output_FbC)
526
527   names(v.output) <- c(pred.names, FbC.names)
528
529   # Create some output helpful for plots of interaction effects
530   if(!is.null(fcmat)){
531
532     # Construct a matrix containing all observed category combinations
```

```
533     # of factor and continuous variable
534
535     plotmatlist <- list()
536     for(l in 1:nrow(fcmat)){
537       fctr <- fcmat[l,1]; crv <- fcmat[l,2]
538
539       cat.names <- categories_FbC[[l]]
540
541       lst <- lapply(strsplit(cat.names, "@"), function(k) matrix(k, nrow =
    1))
542
543       plotmatlist[[l]] <- Reduce(rbind, lst)
544     }
545
546     plot.output.FbC <- list()
547     plot.output.total <- list()
548
549
550     for(l in 1:nrow(fcmat)){
551       fctr <- fcmat[l,1]
552       crv <- fcmat[l,2]
553
554       fctrname <- pred.names[fcmat[l,1]]
555       crvname <- pred.names[fcmat[l,2]]
556
557       fctr.x <- factor(plotmatlist[[l]][,1], levels = categories[[fctr]])
558       crv.x <- as.numeric(plotmatlist[[l]][,2])
559
560       rows <- lapply(1:nr.categories[[fctr]],
561                      function(k) which(as.numeric(fctr.x) == k))
562
563       plot.output.FbC[[FbC.names[l]]] <-
564         list(x.axis = crv.x,
565              y.axis = v.output[[FbC.names[l]]]$quantifications * betasIE[fctr,crv],
566              x = X[,crv],
567              y = y,
568              lvls = categories[[fctr]],
569              rows = rows
570         )
571
572       adjusted.x.axis <- list()
573
574       quant.beta.FbC <- v.output[[FbC.names[l]]]$quantifications * betasIE[fctr,crv]
575       quant.beta.Crv <- v.output[[crvname]]$quantifications * betas[crv]
576       quant.beta.Fctr <- v.output[[fctrname]]$quantifications * betas[fctr]
577
578       xs <- lapply(1:nr.categories[[fctr]], function(k){
579         return(which(categoriesnum[[crv]] %in% plotmatlist[[l]][rows[[k]],2]))
580       })
581
582       quant.beta.total <- lapply(1:nr.categories[[fctr]], function(k){
583         return(quant.beta.FbC[rows[[k]]] +
584                  quant.beta.Crv[xs[[k]]] +
```

```
585                   rep(quant.beta.Fctr[k], length(xs[[k]]))
586           )
587         })
588
589         adjusted.x.axis <- lapply(xs, function(k) categoriesnum[[crv]][k])
590
591         plot.output.total[[FbC.names[l]]] <- list(x.axis = adjusted.x.axis,
592                                                   x.axis.tot = categoriesnum[[crv]],
593                                                   y.axis = quant.beta.total,
594                                                   x = XNum[[crv]],
595                                                   y = y)
596
597       }
598     }
599
600
601   return(list(betas = betas,
602               betasIE = betasIE,
603               sstot = N,
604               ssreg = sum(y.hat^2),
605               ape = ape,
606               loss = loss,
607               v = v.output,
608               plot.output.FbC = plot.output.FbC,
609               plot.output.total = plot.output.total))
610 }
```

A.1.2.2 Combined effect FbC-OS-regression model

```
FbC.OS.CE <- function(formula, fcmat = NULL, data,
                      SL = rep("Nominal", length(all.vars(formula))-1),
                      nDegrees, nInteriorKnots, crititer=0.00001, maxiter=1000) {
  # Function to perform Optimal Scaling on independent variables, including the
  #   possibility for Factor-by-Curve interactions.
  # Optimal scaling level of the outcome is restricted to "numeric".
  #
  # Arguments:
  #   formula: an object of class "formula": a symbolic description of the model
  #            to be fitted.
  #   fcmat: a matrix with, for each row, five values, the first number indicating
  #          the factor variable and the second indicating the curve variable. Third
  #          and fourth are spline degree and interior knots for the curve. The
  #          fifth value indicates whether to fit nonmonotone (1) or monotone (2)
  #          interaction curves.
  #   data: a data frame containing the variables in the model.
  #   SL: a vector containing the scaling levels, all nominal if not specified
  #   nDegrees: a vector containing the degrees of the splines for the main effects
  #   nInteriorKnots: a vector containing the number of interior knots of the
  #                   splines for the main effects
  #   crititer: critical value for the decrease in loss between two iterations
  #   maxiter: maximum number of iterations
  #
  # Returns:
  #   a list object with: a) The final regression coefficients for any main effects
  #                       b) The final regression coefficients for the interactions
  #                       c) Total sum of squares
  #                       d) Regression sum of squares
  #                       e) Apparent prediction error (APE)
  #                       f) Loss for each iteration
  #                       g) The category quantifications
  #                       h) Some output that can be used for plots

  # Perform some preliminary tests, checking argument validity
  if (class(formula)!="formula"){
    stop("Argument formula is not of the right class.")
  } else if (!is.data.frame(data)){
    stop("Argument data is not a dataframe.")
  } else if (!is.vector(SL) | (length(SL) == 0)){
    stop("Argument SL is not a vector or has a length of zero.")
  } else if (SLcheck(SL)){
    stop("Argument SL contains unrecognised scaling levels.
         Use only the following (case insensitive):
         FbC, Nominal, Ordinal, Numeric, NonMonotoneSpline or MonotoneSpline")
  }

  # Construct a variable for the outcome and a matrix with the predictors
  outcome.name <- all.vars(formula)[1]
  pred.names <- all.vars(formula)[-1]

  N <- nrow(data)

```

```
53  y <- data[, outcome.name]
54  y <- autoscale(y) # Scaled outcome, to be called y in the rest of the code
55
56  X <- as.data.frame(data[, pred.names]) # Independent variables, not scaled
57  colnames(X) <- pred.names
58
59  P <- length(pred.names) # Number of independent variables
60
61  SLnum <- # Vector of which variables are to be scaled numerically
62    sapply(SL, function(k) tolower(k) == "numeric")
63
64  categories <- # Category labels
65    lapply(pred.names, function(k) sort(unique(X[, k])))
66  nr.categories <- # Number of categories for each variable
67    lapply(pred.names, function(k) length(unique(X[, k])))
68
69  # If interaction is included, specify indices for variables
70  # to include as interactions
71  if(!is.null(fcmat)) {
72    fcmat <- matrix(fcmat, ncol = 5)
73    # Store indices of variables to be included as interactions separately
74    intinds <- matrix(fcmat[,1:2], ncol = 2)
75  } else {
76    intinds <- matrix(NA, ncol = 2, nrow = 1)
77  }
78
79
80  # Numeric values for each category for all observations
81  XNum <- lapply(1:P, function(k) {
82    if(is.numeric(X[, pred.names[k]])){
83      return(X[,pred.names[k]])
84    } else {
85      return(as.numeric(as.factor(X[,pred.names[k]])))
86      # As.factor also sorts the order, so first column of the indicator matrix
87      # (or its first index) are both indicated as one.
88    }
89  })
90
91  # Numeric values for each combination of category and numeric value
92  # for all observations for each Factor-by-Curve
93  XNum_FbC <- lapply(1:nrow(fcmat), function(k) {
94    a <- intinds[k,1]; b <- intinds[k,2]
95    comb.factor <- interaction(X[,a],X[,b], drop = TRUE,
96                               sep = "@", lex.order = TRUE)
97    return(as.numeric(comb.factor))
98  })
99
100 # Unique numerical values for categories
101 categoriesnum <- lapply(1:P, function(k) {
102   if(is.numeric(X[, pred.names[k]])){
103     return(sort(unique(X[,pred.names[k]])))
104     # Actual values of variable if it is numeric
105   } else {
```

```
    return(1:length(categories[[k]]))
    # {1, ..., C_k} if variable is nonnumeric
  }
})

# Unique numerical values for each combination of category and numeric value
# for Factor-by-Curve
categoriesnum_FbC <- lapply(1:nrow(fcmat), function(k){
  a <- intinds[k,1]; b <- intinds[k,2]
  comb.factor <- unique(interaction(X[,a],X[,b], drop = TRUE,
                                    sep = "@", lex.order = TRUE))
  return(comb.factor)
})




# Initialize transformed variables as autoscaled numeric variables
# Variables included in Factor-by-Curve are implemented as one combined
# transformation for both factor and curve
PhiX <- matrix(NA, nrow = N, ncol = P)
Fctrs <- Crvs <- MEs <- NULL

for(k in 1:P){
  if(k %in% intinds[,1]){
    # If variable is included as factor for interaction, then this column
    # is not used, NA column is used for ease of further coding
    PhiX[,k] <- rep(NA, N)
    Fctrs <- c(Fctrs, k)
  } else if(k %in% intinds[,2]){
    ind <- which(k == intinds[,2])
    PhiX[,k] <- autoscale(XNum_FbC[[ind]])
    Crvs <- c(Crvs, k)
  } else {
    PhiX[,k] <- autoscale(XNum[[k]])
    MEs <- c(MEs, k)
  }
}


# Construct the indicator matrices, as vectors containing indices
# to reduce redundancy
g <- lapply(1:P, function(k){
  if(k %in% Crvs){
    # Indicator matrix for continuous variable in interaction is for
    # unique combinations of categories between interacting variables
    ind <- which(k == Crvs)
    return(as.numeric(as.factor(XNum_FbC[[ind]])))
  } else if(!SLnum[k] | (k %in% Fctrs)){
    return(as.numeric(as.factor(XNum[[k]])))
  } else {
    return(NULL)
  }
```

```
159   })
160
161   # And the univariate marginals
162   d <- lapply(1:P, function(k){
163     ifelse(!SLnum[k], return(as.numeric(table(XNum[[k]]))), return(NULL))
164     })
165
166   # Initialize the coefficients:
167   betas <- numeric(P) + 1
168   betasIE <- matrix(0, nrow = P, ncol = P)
169
170   if(!is.null(fcmat)){
171     for(i in split(intinds, seq(nrow(intinds)))){
172       betasIE[i[1],i[2]] <- 1
173     }
174   }
175
176
177
178   # Initialize splines
179   splineBases <- list()
180   splineCoefficients <- vector("list", P)
181
182   for(k in 1:P){
183     if(tolower(SL[k]) %in% c("nonmonotonespline","monotonespline")){
184       # Use quantiles for selecting the knot placement
185         # Using unique categories, with unweighted knot placement as a result
186
187       interiorKnots_k <- quantile(categoriesnum[[k]],
188                                   probs =
189                                     seq(0,1, length.out = nInteriorKnots[k] +
    2),
190                                   type = 6)[-c(1, nInteriorKnots[k] +2)]
191                                           # +2 to exclude the boundaries
192
193       splineBases[[k]] <- iSpline(XNum[[k]],
194                                   knots = interiorKnots_k,
195                                   degree = nDegrees[k]-1,
196                                     # Degree -1, due to usage of I-splines
197                                   intercept = T)
198     } else {
199       splineBases[[k]] <- NULL
200     }
201   }
202
203   # Initialize splines for interaction effects if included in the model
204   if(!is.null(fcmat)){
205     splineBasesFbC <- list()
206     splineCoefficientsFbC <- list()
207     row.indices <- list()
208
209     for(l in 1:nrow(intinds)){
210       a <- intinds[l,1]; b <- intinds[l,2]
```

```
211       row.indices[[l]] <- list()
212       splineBasesFbC[[l]] <- list()
213       splineCoefficientsFbC[[l]] <- list()
214
215       for(m in 1:nr.categories[[a]]){
216         row.indices[[l]][[m]] <- rows <- which(g[[a]]==m)
217
218         interiorKnots_k <- quantile(categoriesnum[[b]],
219                                     probs = seq(0,1, length.out = fcmat[l,4] +
    2),
220                                     type = 6)[-c(1, fcmat[l,4] +2)]
221
222         splineBasesFbC[[l]][[m]] <- iSpline(XNum[[b]][rows],
223                                             knots = interiorKnots_k,
224                                             degree = fcmat[l,3]-1,
225                                             intercept = T)
226       }
227     }
228   }
229
230
231   # Initialize a vector to save the loss (=squared euclidean norm)
232   # after each iteration
233   loss <- numeric(maxiter+1)
234   # Use as starting value for the loss:
235   loss[1] <- Inf
236
237   # We set our index i at 2, because we already used the first
238   # element of the loss vector
239   i <- 2 # Note that the actual iteration number of our algorithm is i -
    1
240
241   # Convergence variable to indicate convergence (due to either reaching
242   # max number of iterations or reaching the loss criterion)
243   conv <- FALSE
244
245
246   # Start of iterative algorithm
247   while(!conv) { # I.e. while not converged
248
249     # Main effects #
250
251     # Skipped if only interacting variables are included in the model
252     for (j in 1:P) {
253       if(!(j %in% MEs)) next()
254
255       if(P == 1){
256         u <- y
257       } else {
258         # Calculate partial residuals
259         if(length(MEs) == 1){
260           exj_terms <- 0
261         } else {
```

```
262           exj <- MEs[MEs != j]
263
264           exj_terms <- lapply(exj, function(k) return(as.matrix(betas[k]*PhiX[,k])))
265         }
266
267
268         if(!is.null(fcmat)){
269           exjIE_terms <- lapply(1:nrow(fcmat), function(k) {
270             a <- fcmat[k,1]; b <- fcmat[k,2]
271             return(as.matrix(betasIE[a,b] * PhiX[,b]))
272           })
273           # Sum partial regression terms and calculate the residual
274             # (including interaction effects in u)
275           u <- y - Reduce("+", exj_terms) - Reduce("+",exjIE_terms)
276         } else {
277           # Sum partial regression terms and calculate the residual
278             # (no interaction effects in u)
279           u <- y - Reduce("+", exj_terms)
280         }
281       }
282
283       # Calculate quantifications and/or betas for given scaling level of variable j
284       if(!SLnum[j]){
285         ## Quantifications ##
286
287         ### Nominal ###
288
289         # Calculation of updated transformed variable:
290
291         Phix_j <- u # Fully unrestricted phi_j(x_j)
292
293         # Update quantifications
294         vs <- aggregate(Phix_j, XNum[j], mean)
295
296         # Construct new phi_j(x_j), which is now restricted to nominal
297         nominal.phi <- as.numeric(vs[g[[j]],2])
298
299         restricted.phi <- nominal.phi
300
301         if(tolower(SL[j]) == "nonmonotonespline"){
302           ### NonMonotone Spline ###
303
304           # Restrict transformed variable using nonmonotone spline
305           restricted.phi <- fitNonMonotoneSpline(nominal.phi,
306                                                  splineBases[[j]]
307                                                  )$restrictedQuantifications
308
309         } else if(tolower(SL[j]) == "monotonespline"){
310           ### Monotone Spline ###
311
312           # Restrict transformed variable using monotone spline
313           if(i == 2){
314             # In the first iteration, use the splineCoefficients from the nominal
```

```
315             # spline solution as startingvalues for the ordinal spline fit.
316             # After the first iteration, continue with coefficients from previous
317             # ordinal updates.
318
319             splinefit <- fitNonMonotoneSpline(nominal.phi[,j],
320                                               splineBases[[j]])
321
322             splineCoefficients[[j]] <- splinefit$splineCoefficients
323           }
324
325           splinefit <- fitMonotoneSpline(nominal.phi[,j],
326                                          splineBases[[j]],
327                                          splineCoefficients[[j]],
328                                          maxSplineIter = ifelse(i == 2, 200, 2))
329
330           splineCoefficients[[j]] <- splinefit$splineCoefficients
331
332           if(all(splineCoefficients[[j]] == 0)){
333             # If all spline coefficients are zero, the direction of the
334             # monotone function should be changed. This is done by changing
335             # the sign of the corresponding beta and then fitting the spline
336             # on the negative versions of the quantifications.
337             betas[j] <- -1 * betas[j]
338             splinefit <- fitMonotoneSpline(-PhiX[,j],
339                                            splineBases[[j]],
340                                            -splineCoefficients[[j]])
341
342             splineCoefficients[[j]] <- splinefit$splineCoefficients
343           }
344           restricted.phi <- splinefit$restrictedQuantifications
345
346         } else if (tolower(SL[j]) == "ordinal"){
347           ### Ordinal ###
348
349           # Restrict quantifications to monotonic increasing
350           restricted.v <- restrictOrdinal(vs[,2], d[[j]])
351
352           # If first and last quantifications are equal (i.e. all quantifications
353           # are equal), inverse the sign of the quantifications and restrict those
354           # to monotonic increasing
355           if(restricted.v[1] == restricted.v[nrow(restricted.v)]){
356
357             restricted.v <- restrictOrdinal(-vs[,2], d[[j]])
358           }
359
360           # Construct transformed variable from the ordinal quantifications
361           restricted.phi <- restricted.v[g[[j]]]
362         }
363
364
365         # Standardize PhiX: s.t. mean(PhiX) = 0, sigma(PhiX) = 1, ||PhiX||^2 =
    N
366         PhiX[,j] <- autoscale(restricted.phi)
```

```
367
368       }
369
370       ## Betas ##
371
372       betas[j] = (matrix(PhiX[,j], nrow = 1)%*%u)/N
373     }
374
375
376     # Factor by Curve #
377     if(!is.null(fcmat)){
378
379       # Determine main effects if included
380       if(is.null(MEs)){
381         ME_terms <- list(0)
382       } else {
383         ME_terms <- lapply((1:P)[MEs],
384                            function(k) return(as.matrix(betas[k]*PhiX[,k])))
385       }
386
387       for(l in 1:nrow(fcmat)){
388         # Sum partial regression terms and calculate the residual
389           # (no interaction effects in u)
390         u <- y - Reduce("+", ME_terms)
391
392         if(nrow(fcmat) != 1){
393           # Include interaction terms in partial residual if
394           # more than one is modelled
395           exjIE_terms <- apply(matrix(intinds[-l,],ncol = 2), 1, function(k) return(as.matrix(b
    PhiX[,k[2]])))
396           u <- u - rowSums(exjIE_terms)
397         }
398
399         fctr <- intinds[l,1] # Include interaction terms in partial residual
400         crv <- intinds[l,2] # Nr of variable which is the continuous variable
401
402
403         vs <- aggregate(u, XNum_FbC[l], mean) # Nominal quantifications
404
405         nominal.phi <- vs[g[[crv]],2] # Nominal transformed variable
406
407         restricted.phi <- nominal.phi
408
409         for(m in 1:nr.categories[[fcmat[l,1]]]){
410
411           # Specify the rows relevant for the given factor level
412           rows <- row.indices[[l]][[m]]
413
414
415           if(tolower(fcmat[l,5]) == 1){
416
417             ### Nonmonotone spline ###
418
```

```
419             # Restrict observations under given factor level using
420             # nonmonotone spline
421             splineFit <- fitNonMonotoneSpline(nominal.phi[rows],
422                                               splineBasesFbC[[l]][[m]])
423
424             restricted.phi[rows] <- splineFit$restrictedQuantifications
425           } else if(tolower(fcmat[l,5]) == 2){
426
427             ### Monotone spline ###
428
429             # Restrict observations under given factor level using monotone spline
430
431             if(i == 2){
432               # In the first iteration, use the splineCoefficients from the nominal
433               # spline solution as starting values for the ordinal spline fit.
434               # After the first iteration, continue with coefficients from previous
435               # ordinal updates.
436               splineFit <- fitNonMonotoneSpline(nominal.phi[rows],
437                                                 splineBasesFbC[[l]][[m]])
438
439               splineCoefficientsFbC[[l]][[m]] <- splineFit$splineCoefficients
440             }
441
442             splinefit <- fitMonotoneSpline(nominal.phi[rows],
443                                            splineBasesFbC[[l]][[m]],
444                                            splineCoefficientsFbC[[l]][[m]],
445                                            maxSplineIter = ifelse(i ==
    2, 200, 2),
446                                            FbC = TRUE)
447
448             # Save coefficients for next iteration
449             splineCoefficientsFbC[[l]][[m]] <- splinefit$splineCoefficients
450
451             restricted.phi[rows] <- splinefit$restrictedQuantifications
452           }
453
454
455         }
456
457         PhiX[,crv] <- autoscale(restricted.phi) # Standardize PhiX_FbC as a whole
458
459         betasIE[fcmat[l,1],fcmat[l,2]] <- t(PhiX[,crv])%*%u / N
460
461         if(fcmat[l,5] == 2 && sign(betasIE[fcmat[l,1],fcmat[l,2]]) ==
    -1){
462           # If interaction beta is negative, the direction of the beta and all
463           # monotone functions should be changed. This is done by changing the
464           # sign of the corresponding beta, spline coefficients and quantifications
465
466           betasIE[fcmat[l,1],fcmat[l,2]] <- -1 * betasIE[fcmat[l,1],fcmat[l,2]]
467
468           for(m in 1:nr.categories[[fcmat[l,1]]]){
469             rows <- row.indices[[l]][[m]]
```

```
470
471             splineCoefficientsFbC[[l]][[m]] <- -splineCoefficientsFbC[[l]][[m]]
472             PhiX[rows,crv] <- -PhiX[rows,crv]
473           }
474         }
475
476       }
477     }
478
479
480
481     # Calculate the loss: sum of squares between y and y.hat
482
483     # Determine y.hat
484     if(is.null(MEs)){
485       y.hat <- 0
486     } else {
487       y.hat <- lapply(MEs, function(k) return(as.matrix(PhiX[,k]*betas[k])))
488       y.hat <- Reduce("+", y.hat)
489     }
490
491     #If interactions are specified, add those to y.hat
492     if(!is.null(fcmat)){
493       y.hat <- y.hat + Reduce("+",lapply(1:nrow(fcmat), function(k) {
494         a <- fcmat[k,1]; b <- fcmat[k,2]
495         return(as.matrix(betasIE[a,b] * PhiX[,b]))
496       }))
497     }
498
499
500
501
502     loss[i] <-  sum((y-y.hat)^2)
503
504     # Check if convergence criteria are met:
505     # Number of maximum iterations is met
506     # OR
507     # Difference in loss between previous and current solution is small enough
508
509     conv <- ((abs(loss[i-1]-loss[i]) < crititer) | i-1 >= maxiter)
510
511     # Increase the iteration number
512     i <- i + 1
513   }
514
515   # Do not store the first loss value (as we set it to Inf)
516   loss <- loss[2:(i-1)]
517   names(betas) <- pred.names
518   rownames(betasIE) <- pred.names
519   colnames(betasIE) <- pred.names
520
521   # Determining total sum of squares and APE
522   sstot <- N
```

```
523  ape <- loss[i-2]/sstot
524
525  # Create output of the quantifications
526  if(is.null(MEs)){
527    v.output <- list()
528    output.names <- pred.names[Crvs]
529  } else {
530    v.output <- lapply(1:P, function(k) {
531      if(!SLnum[k] & !(k %in% c(Fctrs,Crvs))){
532        vs <- aggregate(PhiX[,k], XNum[k], mean)
533        return(data.frame(categories=categories[[k]],
534                          quantifications = as.numeric(vs[,2])))
535      } else {
536        return(NULL)
537      }
538    })
539    output.names <- pred.names
540  }
541
542
543
544  if(!is.null(fcmat)){
545    # Construct a matrix containing all observed category combinations
546    # of factor and continuous variable
547    # Additionally, save the quantifications for the interaction
548    plotmatlist <- list()
549    for(l in 1:nrow(fcmat)){
550      fctr <- fcmat[l,1]; crv <- fcmat[l,2]
551
552      output.names[crv] <- paste(pred.names[fctr], pred.names[crv], sep =
    ".")
553      vs <- aggregate(PhiX[,crv], XNum_FbC[l], mean)
554
555      cat.names <- as.character(categoriesnum_FbC[[l]]
556                                )[order(as.numeric(categoriesnum_FbC[[l]]))]
557
558      lst <- lapply(strsplit(cat.names, "@"), function(k) matrix(k, nrow =
    1))
559
560      plotmatlist[[l]] <- Reduce(rbind, lst)
561
562      v.output[[crv]] <-
563        data.frame(categories = cat.names, quantifications = vs[,2])
564    }
565  }
566
567  names(v.output) <- output.names
568
569
570  if(!is.null(fcmat)){
571
572    plot.output.FbC <- list()
573
```

```
574     for(l in 1:nrow(fcmat)){
575       fctr <- fcmat[l,1]
576       crv <- fcmat[l,2]
577       fctrname <- pred.names[fcmat[l,1]]
578       crvname <- pred.names[fcmat[l,2]]
579 
580       y.hat.FbC <- as.matrix(PhiX[,crv] * betasIE[fctr,crv])
581 
582       fctr.x <- factor(plotmatlist[[l]][,1], levels = categories[[fctr]])
583       crv.x <- as.numeric(plotmatlist[[l]][,2])
584 
585       rows <- lapply(1:nr.categories[[fctr]],
586                      function(k) which(as.numeric(fctr.x) == k))
587 
588       plot.output.FbC[[output.names[crv]]] <-
589         list(x.axis = crv.x,
590              y.axis = v.output[[crv]]$quantifications * betasIE[fctr,crv],
591              lvls = categories[[fctr]],
592              rows = rows)
593     }
594   }
595 
596   if(is.null(MEs)){
597     betas.output <- NULL
598   } else {
599     betas.output <- betas[MEs]
600   }
601   return(list(betas = betas.output,
602               betasIE = betasIE,
603               sstot = N,
604               ssreg = sum(y.hat^2),
605               ape = ape,
606               loss = loss,
607               v = v.output,
608               plot.output.FbC = plot.output.FbC))
609 }
```

A.2 Data

A.2.1 Loading and estimating densities of the BCRP data

```
library(quint)

# Select relevant variables and create data frame
data.bcrp <- data.frame(physt.delta = autoscale(bcrp$physt3 - bcrp$physt1),
                        cesdt.delta = autoscale(bcrp$cesdt3 - bcrp$cesdt1),
                        uncomt = bcrp$uncomt1,
                        cond = as.factor(bcrp$cond),
                        comorbid = bcrp$comorbid)

# Drop observations with missing values
data.bcrp <- data.bcrp[complete.cases(data.bcrp),]

bcrp.names <- c("physt.delta", "cesdt.delta", "uncomt", "cond", "comorbid")

levels(data.bcrp$cond) <- c("Nutritional", "Educational", "Control")



# Estimating densities for plots and standardising outcome
densities <- list()

for(i in bcrp.names){

  if(i == "physt.delta" | i == "cesdt.delta"){
    data.bcrp[,i] <- as.numeric(autoscale(data.bcrp[,i]))

    densities[[i]] <- density(data.bcrp[,i])
    densities[[i]] <- cbind(densities[[i]]$x, densities[[i]]$y)
  } else if(!is.factor(data.bcrp[,i])){
    densities[[i]] <- density(data.bcrp[,i])
    densities[[i]] <- cbind(densities[[i]]$x, densities[[i]]$y)
  }
}
```

A.2.2 Data simulation

```r
set.seed(12)

# Simulate data
x1 <- c(1,2,3)
x2 <- -24.5:24.5

y_true <- as.numeric(outer(x1, x2, function(a, b) {
  .2 * autoscale(a) + .8 * autoscale(sin(b/10 -1)) +
    .4 * autoscale(a) * autoscale(sin(b/10))
  }))

y_obs <- y_true + rnorm(150, 0, .3)


# Save simulated data as data frame
x1 <- rep(c(1,2,3), 50)
x2 <- as.vector(t(replicate(3, as.numeric(x2))))

## Add a variable containing bin labels
binvar <- sort(rep(1:10, 15))
matnames <- cbind("(", apply(cbind(seq(-25, 20, 5), seq(-20, 25, 5)), 1,
                             function(x) paste(x, sep = "", collapse =
  ",")), ")")
binnames <- apply(matnames, 1, function(x) paste(x, collapse = ""))
x3 <- binvar


dat <- data.frame(x1 = as.factor(x1), x2 = x2, y = y_obs, x3 = x3)
```

A.3 Plotting

A.3.1 Driver's age and number of involved crashes simulation

```r
# Simulate data for this example
set.seed(1371878)
num <- 500
hor <- scale(20:80, TRUE, TRUE)
vert <- hor^2 + rnorm(length(hor), 0, .4)

lmod <- lm(vert ~ hor)
lmline <- function(x){lmod$coef[1] + lmod$coef[2] * x}

par(mar = c(3,3,1,1), mgp = c(.5,.5,0))

# Plot points and linear prediction
plot(hor, vert, xlab = "", ylab = "", xaxt = "n", yaxt = "n")
curve(lmline, min(hor), max(hor), 100, add = TRUE, lwd = 2, col = 2)
legend("top", legend = c("Observations", "Linear prediction"),
       col =  1:2, pch = c(1, -1), lty = c(0, 1), lwd = c(1, 2))
```

A.3.2 Scaling levels

```r
# Fit model for each scaling level

modnom <- FbC.OS.SE(y ~ x3, NULL, dat, "nominal")
modord <- FbC.OS.SE(y ~ x3, NULL, dat, "ordinal")
modnum <- FbC.OS.SE(y ~ x3, NULL, dat, "numeric")
modnmspl <- FbC.OS.SE(y ~ x3, NULL, dat, "nonmonotonespline", 2,2)
modmspl <- FbC.OS.SE(y ~ x3, NULL, dat, "monotonespline", 2,2)

modlst <- list(modnom, modord, modnum, modnmspl, modmspl)

# Plot quantifications for each scaling level
for(i in 1:5){
  plot(1:10, modlst[[i]]$v$x3$quantifications, type = "b", cex = 0.9,
       lwd = 2, axes = FALSE, xlab = "", ylab = "", ylim = c(-1.75, 2.25))
  axis(side = 1, at = 1:10, labels = FALSE)
  text(x = 1:10, par("usr")[3]+par("usr")[3]/8, labels = unique(binnames),
       srt = 45, pos = 1, xpd = TRUE)
  axis(side = 2)
  box()
}
```

A.3.3 Simulated Factor-by-Curve example

```
sim.FbC <- FbC.OS.SE(y ~ x1 * x2, matrix(c(1,2,2,1,1), nrow = 1),
                     dat, c("nominal", "monotonespline"), c(0,2), c(0,1))
plot.output <- sim.FbC$plot.output.total$x1.x2

xlims <- c(min(unlist(plot.output$x.axis)),max(unlist(plot.output$x.axis)))
ylims <- c(min(unlist(plot.output$y.axis)),max(unlist(plot.output$y.axis)))

# Construct empty plot
par(mar=c(4, 4, 1, 2) + 0.1)
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Continous variable x",
     ylab = "Outcome y")

# Plot observed outcome values
for(i in 1:3){
  xax <- -24.5:24.5
  yax <- y[seq(i,150, by = 3)]
  points(xax,yax,
         lwd = 1,
         col = c("#990000", "#009900", "#000099")[i],
         cex = .4)
}

# Plot each curve separately
for(i in 1:3){
  xax <- plot.output$x.axis[[i]]
  yax <- plot.output$y.axis[[i]]
  points(xax,yax, type = "b", lwd = 2, cex = 0.9, col = i+1)
}

# Add legend
legend("topleft",
       legend = LETTERS[1:3],
       pch = 1,
       col = 2:4,
       pt.lwd = 2,
       cex = .9)
```

A.3.4 Descriptives BCRP data

```r
xname <- c("Change in physical functioning", "Change in depressive symptoms",
           "Unmitigated communion", "Number of comorbidities")
xvar <- list(data.bcrp$physt.delta, data.bcrp$cesdt.delta,
             data.bcrp$uncomt, data.bcrp$comorbid)

# Plot stacked histogram for both outcomes and both predictors
for(i in 1:4){
  histplot <- ggplot() +
    geom_histogram(mapping = aes(x = xvar[[i]], fill = cond),
                   data = data.bcrp, binwidth = c(.5, .5, 2, 1)[i], col = 1) +
    labs(fill = "Experimental condition", x = xname[i], y = "Frequency") +
    guides(fill = FALSE) +
    theme_bw() +
    theme(axis.text=element_text(size=12),
          axis.title=element_text(size=13,face="bold"),
          axis.title.y = element_text(margin = margin(t = 0, r = 10, b = 0, l = 0)),
          axis.title.x = element_text(margin = margin(t = 10, r = 0, b = 0, l = 0)),
          plot.margin = unit(c(.5,.5,.5,.5), "cm"))

  grid.draw(histplot)
}

# Extract legend for separate plotting
lgd_hist <- cowplot::get_legend(
  ggplot() +
    geom_histogram(mapping = aes(x = comorbid, fill = cond),
                   data = data.bcrp, col = 1, binwidth = 1) +
    labs(fill = "Experimental condition",
         x = "Number of comorbidities",
         y = "Frequency") +
    theme_bw() +
    theme(legend.title = element_text(size=13,face="bold"),
          legend.text = element_text(size=12))
)

# Plot legend
grid.newpage()
grid.draw(lgd_hist)
```

A.3.5 Depressive symptoms models

A.3.5.1 Model fitting

```
# Fit model with separate main effects
dep.uncomt <- FbC.OS.SE(formula(cesdt.delta ~ cond * uncomt),
                        matrix(c(1,2,2,2,2), nrow = 1),
                        data.bcrp, c("nominal", "monotonespline"),
                        c(0,2), c(0,2))

# Fit model without separate main effects
dep.uncomt2 <- FbC.OS.CE(formula(cesdt.delta ~ cond * uncomt),
                         matrix(c(1,2,2,2,2), nrow = 1),
                         data.bcrp, c("fbc", "fbc"))
```

A.3.5.2 Quantification plots: Separate effects model

```
# Create italic subscripts for the plots
txt1.dep <- c(substitute(paste(italic("Coefficient value: 0.176"))),
              substitute(paste(italic("Coefficient value: 0.007"))),
              substitute(paste(italic("Coefficient value: 0.222")))
         )

# Quantification plot for experimental condition
plot(1:3, dep.uncomt$v$cond[,2],
     ylab = "Quantification", xlab = "Experimental Condition",
     sub = txt1.dep[1], cex.sub = .85,
     type = "b", cex = 0.9, lwd = 2,
     axes = FALSE)
axis(side = 1, at = 1:3, labels = FALSE)
text(x = 1:3, par("usr")[3] - 0.1,
     labels = c("Nutritional", "Educational", "Standard Care"),
     srt = 0, pos = 1, xpd = TRUE)
axis(side = 2)
box()


# Quantification plot for unmitigated communion
plot(dep.uncomt$v$uncomt,
     ylab = "Quantification", xlab = "Unmitigated Communion",
     sub = txt1.dep[2], cex.sub = .85,
     type = "b", cex = 0.9, lwd = 2)


# Quantification plot for interaction effect
xlims <- c(min(data.bcrp$uncomt),max(data.bcrp$uncomt))
ylims <- c(min(dep.uncomt$v$cond.uncomt$quantifications),
           max(dep.uncomt$v$cond.uncomt$quantifications))

plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
```

```
      xlab = "Unmitigated Communion",
      ylab = "Quantification",
      sub = txt1.dep[3], cex.sub = .85)

fact.rows <- dep.uncomt$plot.output.FbC$cond.uncomt$rows
xax.tot <- dep.uncomt$plot.output.FbC$cond.uncomt$x.axis

for(i in 1:3){
  xax <- xax.tot[fact.rows[[i]]]
  yax <- dep.uncomt$v$cond.uncomt$quantifications[fact.rows[[i]]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("bottomleft",
       legend = c("Nutritional", "Educational", "Control"),
       lty = 0,
       pch = 1,
       lwd = 2,
       col = 2:4,
       pt.lwd = 2,
       cex = .9)
```

A.3.5.3 Quantification plot: Combined effect model

```
xlims <- c(min(data.bcrp$uncomt),max(data.bcrp$uncomt))
ylims <- c(min(dep.uncomt2$v$cond.uncomt$quantifications),
           max(dep.uncomt2$v$cond.uncomt$quantifications))

# Create italic subscripts for the plot
txt2.dep <- substitute(paste(italic("Coefficient value: 0.264")))

# Quantification plot for interaction effect
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Unmitigated Communion",
     ylab = "Quantification",
     sub = txt2.dep, cex.sub = .85)

fact.rows <- dep.uncomt2$plot.output.FbC$cond.uncomt$rows
xax.tot <- dep.uncomt2$plot.output.FbC$cond.uncomt$x.axis

for(i in 1:3){
  xax <- xax.tot[fact.rows[[i]]]
  yax <- dep.uncomt2$v$cond.uncomt$quantifications[fact.rows[[i]]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("bottomleft",
       legend = c("Nutritional", "Educational", "Control"),
       lty = 0,
```

```
       pch = 1,
       lwd = 2,
       col = 2:4,
       pt.lwd = 2,
       cex = .9)
```

A.3.5.4 Fitted value plots

```
# FbC With ME
plot.output <- dep.uncomt$plot.output.total$cond.uncomt

xlims <- c(min(unlist(plot.output$x.axis)),max(unlist(plot.output$x.axis)))
ylims <- c(min(unlist(plot.output$y.axis)),max(unlist(plot.output$y.axis))+.6)

par(mar=c(5, 5, 4, 6) + 0.1)
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Unmitigated Communion",
     ylab = "Predicted scaled change \n in depressive symptoms",
     main = "FbC model with separate Main Effects")

for(i in 1:3){
  xax <- plot.output$x.axis[[i]]
  yax <- plot.output$y.axis[[i]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("topleft",
       legend = c("Nutritional", "Educational", "Control", "Density"),
       lty = c(0,0,0,3),
       pch = c(1,1,1,-1),
       lwd = c(2,2,2,2),
       col = c(2:4,"#828282"),
       pt.lwd = c(2,2,2,0),
       cex = .9)

par(new = TRUE)
plot(densities[["uncomt"]][,1], densities[["uncomt"]][,2], type = "l",
     col = "#828282", lty = 1, lwd = 2, axes = FALSE,
     xlab = "", ylab = "", xlim = xlims)

mtext("Density of unmitigated communion", side = 4, col = "#828282", line =
 4)
axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")

segments(43.075,-0.008,43.075,0.088, col = "#828282")


# FbC Without ME
plot.output <- dep.uncomt2$plot.output.FbC$cond.uncomt
```

```r

par(mar=c(5, 5, 4, 6) + 0.1)
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Unmitigated Communion",
     ylab = "Predicted scaled change \n in depressive symptoms",
     main = "FbC model without separate Main Effects")


for(i in 1:3){
  xax <- plot.output$x.axis[plot.output$rows[[i]]]
  yax <- plot.output$y.axis[plot.output$rows[[i]]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}


legend("topleft",
       legend = c("Nutritional", "Educational", "Control", "Density"),
       lty = c(0,0,0,3),
       pch = c(1,1,1,-1),
       lwd = c(2,2,2,2),
       col = c(2:4,"#828282"),
       pt.lwd = c(2,2,2,0),
       cex = .9)

par(new = TRUE)
plot(densities[["uncomt"]][,1], densities[["uncomt"]][,2], type = "l",
     col = "#828282", lty = 1, lwd = 2, axes = FALSE,
     xlab = "", ylab = "", xlim = xlims)

mtext("Density of unmitigated communion", side = 4, col = "#828282", line =
 4)
axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")

segments(43.075,-0.008,43.075,0.088, col = "#828282")



# LM

dep.lm <- lm(cesdt.delta ~ 0 + uncomt*cond, data.bcrp)

plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Unmitigated Communion",
     ylab = "Predicted scaled change \n in depressive symptoms",
     main = "LM")

for(i in 1:3){
  ind <- which(as.numeric(data.bcrp$cond) == i)
  xax <- data.bcrp$uncomt[ind]
```

```
  yax <- predict(dep.lm)[ind]

  ord <- order(xax)

  points(xax[ord], yax[ord], type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("topleft",
       legend = c("Nutritional", "Educational", "Control", "Density"),
       lty = c(0,0,0,3),
       pch = c(1,1,1,-1),
       lwd = c(2,2,2,2),
       col = c(2:4,"#828282"),
       pt.lwd = c(2,2,2,0),
       cex = .9)


par(new = TRUE)
plot(densities[["uncomt"]][,1], densities[["uncomt"]][,2], type = "l",
     col = "#828282", lty = 1, lwd = 2, axes = FALSE,
     xlab = "", ylab = "", xlim = xlims)

mtext("Density of Unmitigated Communion", side = 4, col = "#828282", line =
 4)
axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")

segments(43.075,-0.008,43.075,0.088, col = "#828282")
```

A.3.6 Physical functioning models

A.3.6.1 Model fitting

```
# Fit model with separate main effects
phy.comorbid <- FbC.OS.SE(formula(physt.delta ~ cond * comorbid),
                          matrix(c(1,2,2,2,2), nrow = 1), data.bcrp,
                          c("nominal", "monotonespline"), c(0,2), c(0,2))

# Fit model without separate main effects
phy.comorbid2 <- FbC.OS.CE(formula(physt.delta ~ cond * comorbid),
                           matrix(c(1,2,2,2,2), nrow = 1),
                           data.bcrp, c("fbc", "fbc"))
```

A.3.6.2 Quantification plots: Separate effects model

```
# Create italic subscripts for the plots
txt1.phy <- c(substitute(paste(italic("Coefficient value: 0.307"))),
          substitute(paste(italic("Coefficient value: 0.062"))),
          substitute(paste(italic("Coefficient value: 0.260")))
          )

# Quantification plot for experimental condition
plot(1:3, phy.comorbid$v$cond[,2],
     ylab = "Quantification", xlab = "Experimental Condition",
     sub = txt1.phy[1], cex.sub = .85,
     type = "b", cex = 0.9, lwd = 2,
     axes = FALSE)
axis(side = 1, at = 1:3, labels = FALSE)
text(x = 1:3, par("usr")[3] - 0.1,
     labels = c("Nutritional", "Educational", "Standard Care"),
     srt = 0, pos = 1, xpd = TRUE)
axis(side = 2)
box()


# Quantification plot for number of comorbidities
plot(phy.comorbid$v$comorbid,
     ylab = "Quantification", xlab = "Number of Comorbidities",
     sub = txt1.phy[2], cex.sub = .85,
     type = "b", cex = 0.9, lwd = 2)


# Quantification plot for interaction effect
xlims <- c(min(data.bcrp$comorbid),max(data.bcrp$comorbid))
ylims <- c(min(phy.comorbid$v$cond.comorbid$quantifications),
           max(phy.comorbid$v$cond.comorbid$quantifications))

plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Number of Comorbidities",
```

```
      ylab = "Quantification",
      sub = txt1.phy[3], cex.sub = .85)


fact.rows <- phy.comorbid$plot.output.FbC$cond.comorbid$rows
xax.tot <- phy.comorbid$plot.output.FbC$cond.comorbid$x.axis

for(i in 1:3){
  xax <- xax.tot[fact.rows[[i]]]
  yax <- phy.comorbid$v$cond.comorbid$quantifications[fact.rows[[i]]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("topright",
       legend = c("Nutritional", "Educational", "Control"),
       lty = 0,
       pch = 1,
       lwd = 2,
       col = 2:4,
       pt.lwd = 2,
       cex = .9)
```

A.3.6.3 Quantification plot: Combined effect model

```
xlims <- c(min(data.bcrp$comorbid),max(data.bcrp$comorbid))
ylims <- c(min(phy.comorbid2$v$cond.comorbid$quantifications),
           max(phy.comorbid2$v$cond.comorbid$quantifications))

# Create italic subscripts for the plot
txt2.phy <- substitute(paste(italic("Coefficient value: 0.300")))

# Quantification plot for interaction effect
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Number of Comorbidities",
     ylab = "Quantification",
     sub = txt2.phy, cex.sub = .85)


fact.rows <- phy.comorbid2$plot.output.FbC$cond.comorbid$rows
xax.tot <- phy.comorbid2$plot.output.FbC$cond.comorbid$x.axis

for(i in 1:3){
  xax <- xax.tot[fact.rows[[i]]]
  yax <- phy.comorbid2$v$cond.comorbid$quantifications[fact.rows[[i]]]
  points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("bottomleft",
       legend = c("Nutritional", "Educational", "Control"),
```

```
       lty = 0,
       pch = 1,
       lwd = 2,
       col = 2:4,
       pt.lwd = 2,
       cex = .9)
```

A.3.6.4 Fitted value plots

```
# FbC With ME
plot.output <- phy.comorbid$plot.output.total$cond.comorbid


xlims <- c(min(unlist(plot.output$x.axis)),max(unlist(plot.output$x.axis)))
ylims <- c(min(unlist(plot.output$y.axis)),max(unlist(plot.output$y.axis)) + .1)

par(mar=c(5, 5, 4, 6) + 0.1)
plot(NULL, NULL,
     xlim = xlims,
     ylim = ylims,
     xlab = "Number of comorbidities",
     ylab = "Predicted scaled change \n in physical functioning",
     main = "FbC model with separate Main Effects")

for(i in 1:3){
  x <- plot.output$x.axis[[i]]
  y <- plot.output$y.axis[[i]]
  points(x,y, type = "b", lwd = 2, col = i + 1, cex = .9)
}

legend("bottomleft",
       legend = c("Nutritional", "Educational", "Control", "Density"),
       lty = c(0,0,0,3),
       pch = c(1,1,1,-1),
       lwd = c(2,2,2,2),
       col = c(2:4,"#828282"),
       pt.lwd = c(2,2,2,0),
       cex = .9)

par(new = TRUE)
plot(densities[["comorbid"]][,1], densities[["comorbid"]][,2], type = "l",
     col = "#828282", lty = 1, lwd = 2, axes = FALSE,
     xlab = "", ylab = "", xlim = xlims)

mtext("Density of number of comorbidities", side = 4, col = "#828282", line = 4)
axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")

segments(13.51,-.01,13.51,0.2, col = "#828282")
```

```
40
41
42
43 # FbC Without ME
44 plot.output <- phy.comorbid2$plot.output.FbC$cond.comorbid
45
46 par(mar=c(5, 5, 4, 6) + 0.1)
47 plot(NULL, NULL,
48      xlim = xlims,
49      ylim = ylims,
50      xlab = "Number of comorbidities",
51      ylab = "Predicted scaled change \n in physical functioning",
52      main = "FbC model without separate Main Effects")
53
54
55 for(i in 1:3){
56   xax <- plot.output$x.axis[plot.output$rows[[i]]]
57   yax <- plot.output$y.axis[plot.output$rows[[i]]]
58   points(xax,yax, type = "b", lwd = 2, col = i + 1, cex = .9)
59 }
60
61
62 legend("bottomleft",
63       legend = c("Nutritional", "Educational", "Control", "Density"),
64       lty = c(0,0,0,3),
65       pch = c(1,1,1,-1),
66       lwd = c(2,2,2,2),
67       col = c(2:4,"#828282"),
68       pt.lwd = c(2,2,2,0),
69       cex = .9)
70
71 par(new = TRUE)
72 plot(densities[["comorbid"]][,1], densities[["comorbid"]][,2], type =
    "l",
73      col = "#828282", lty = 1, lwd = 2, axes = FALSE,
74      xlab = "", ylab = "", xlim = xlims)
75
76 mtext("Density of number of comorbidities", side = 4, col = "#828282", line =
    4)
77 axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")
78
79 segments(13.51,-.01,13.51,0.2, col = "#828282")
80
81
82
83 # LM
84
85 phy.lm <- lm(physt.delta ~ 0 + comorbid*cond, data.bcrp)
86
87 plot(NULL, NULL,
88      xlim = xlims,
89      ylim = ylims,
90      xlab = "Number of comorbidities",
```

```
91      ylab = "Predicted scaled change \n in physical functioning",
92      main = "LM")
93
94 for(i in 1:3){
95   ind <- which(as.numeric(data.bcrp$cond) == i)
96   xax <- data.bcrp$comorbid[ind]
97   yax <- predict(phy.lm)[ind]
98
99   ord <- order(xax)
100
101  points(xax[ord], yax[ord], type = "b", lwd = 2, col = i + 1, cex = .9)
102 }
103
104 legend("bottomleft",
105       legend = c("Nutritional", "Educational", "Control", "Density"),
106       lty = c(0,0,0,3),
107       pch = c(1,1,1,-1),
108       lwd = c(2,2,2,2),
109       col = c(2:4,"#828282"),
110       pt.lwd = c(2,2,2,0),
111       cex = .9)
112
113 par(new = TRUE)
114 plot(densities[["comorbid"]][,1], densities[["comorbid"]][,2], type =
    "l",
115      col = "#828282", lty = 1, lwd = 2, axes = FALSE,
116      xlab = "", ylab = "", xlim = xlims)
117
118 mtext("Density of number of comorbidities", side = 4, col = "#828282", line =
    4)
119 axis(4, col = 1, col.axis = "#828282", las = 1, col.ticks = "#828282")
120
121 segments(13.51,-.01,13.51,0.2, col = "#828282")
```