C. Retel, BSc

Simulation-based power calculation for Next-Generation Sequence data

Master thesis
Statistical Science for the Life and Behavioural Science

Supervised by:
prof. dr. M.A. van de Wiel (VUmc)
prof. dr. A.W. van der Vaart (MI Leiden)

Date master exam: 3 july 2015

Mathematisch Instituut,
Universiteit Leiden

VUmc, Amsterdam

Abstract

Next-Generation Sequence (NGS) technologies provide promising new opportunities for the quantitative comparison of genomic expression profiles. Analysis of NGS datasets is made challenging by their high-dimensionality and count-based nature. Modelling frameworks are based on negative binomial GLM's and involve multiple testing. Analytical formula's to express power are not available in this setting. During this investigation, functions to do simulation-based power calculations for NGS data, based on small pilot datasets, were created. The task sequence is as follows: First, empirical Bayesian estimation methods are applied to a pilot dataset to recover distributional parameters that reflect data structure and signal in a population. These parameters are then employed within a data-generative framework to simulate datasets of increasing sample size. Finally, tests of differential expression on these simulations yield a prediction of average power and number of rejections associated with each value of sample size.

To assess the performance of our proposed power calculation algorithm, we used publicly available, comparatively large datasets, sampled "pilot" subsets from these, and compared predictions based on pilots to results obtained with the full-sized datasets. Our functions are useful to any researcher confident about the homogeneity of his data. Our results however also indicate that stable estimation of the proportion of differential expression p_1 is difficult when sample size is small, which sometimes leads to inaccurate power calculations. The observed variation was such, that we suspect it also influences standard differential expression analysis in an undesired manner. We therefore argue that general care should be taken in NGS research, because currently accepted sample sizes may not always be large enough to yield a representative image of differential expression between populations.

Contents

1 Introduction

In the area of genomic research, technology is moving forward at a fast pace. Complete genomes are now routinely sequenced to investigate the precise role of genomics in biological characteristics. A cell's genome, or DNA, is retained in the cell nucleus, a closed compartment within the cell. Genomic features perform their functions by being transcribed into RNA molecules which can then be transported elsewhere in the cell.

Determining the complete chromosomal sequence of an organism is called whole-genome sequencing. Once a species' genome is available, it is possible to build microarrays, plates spotted with "probes", onto which specific known features attach. With the use of microarrays, researchers are able to detect and quantify the presence of DNA or RNA in a biological sample, for features with known sequence. The introduction of microarrays [Schena et al., 1995] has made it possible to compare expression levels between samples, and hence provide insight into transcription rate, on top of genomic structure.

1.1 RNAseq techniques

The next large technological advance in genomic research is the use of Next-Generation Sequencing (NGS) techniques; also referred to as RNAseq, direct high-throughput sequencing or deep sequencing [Mortazavi et al., 2008]. In a typical RNAseq analysis, an organism's complete transcriptome is digested into short sequences of 25 up to 200 base pairs. The collection of these short reads is sequenced multiple times. The exact sequence of every detected fragment is stored, and every unique read is mapped to a distinct genomic feature. Mapping can be based on the overlap between reads [Trapnell et al., 2010] or, when information on genomic structure is available, to places on a reference genome [Trapnell et al., 2009, Langmead et al., 2009].

RNAseq analysis methods offer some explicit advantages over microarrays. Pre-existing information on the sequence of the genome under investigation is no longer necessary. RNAseq methods use smaller quantities of biological material, reducing costs of data gathering and biases induced by artificial multiplication. Measuring provides a digital signal in the form of the number of observations, whereas in microarray analyses, the intensity of an (analog) fluorescence signal has to be processed by image analysis software and then quantified to reflect the amount of DNA present in a sample. Finally, samples can ben passed through the detector multiple times, greatly enhancing sensitivity for lowly expressed features [Wang et al., 2009, Malone and Oliver, 2011].

As in all bioinformatic research, the output files generated by RNA-seq methods have considerable size and require some skills in handling. There is usually a comparatively small number of

subjects (5-20), as opposed to a much larger number of features (10k - 50k) per subject. Modelling discreet count data is in itself less straightforward than analysis of a continuous response variable, and the observed variability in RNAseq data is typically higher than (and not in accordance with) what can be captured by "conventional" count models, using for example Poisson or negative binomial distributions. It is commonly accepted that available methods for preprocessing and analysis of RNAseq data are still under construction, and for a lot of basic problems, there are no undisputed solutions available yet [Pepke et al., 2009, Oshlack et al., 2010, Guo et al., 2013] .

1.2 Goal of this project

The aim of this research thesis is to create a sample size calculation tool for RNAseq data. Our proposed procedure involves the combination of two R software packages created for the analysis of RNAseq data. Of these two packages, edgeR [Robinson et al., 2010] is one of the (or arguably, the) most well-known and widespread procedures used in analysis of RNAseq datasets. It provides the user with exact p-value estimates and has computationally very efficient solutions to the problems mentioned above, making it attractive for the average biological investigator.

The second package, ShrinkBayes [Van de Wiel et al., 2014], uses Bayesian methods for RNAseq inference. It implements empirical Bayes principles of borrowing information across features to estimate prior distributions, and yields probability distributions of effect size. A more elaborate explanation of the mechanisms implemented in these software packages and the settings used for sample size calculation will follow in the Methods section. ShrinkBayes is less widespread in use than edgeR but, in exchange for longer computation times, has been shown to yield more reproducible results, especially on small samples.

The idea behind our sample size calculation mechanism is to use ShrinkBayes' flexible modelling approach *on small pilot samples* to find distribution parameters that accurately represent the RNAseq data structure. We then have a data-generative model: datasets of increasing size can be generated, after which edgeR's higher computational efficiency is used to "analyse" the simulations. This will give us a prediction of the results of analysing similar datasets of larger (or smaller) size. Because the values of the effect coefficients used for simulation are known, it is possible to calculate the power of detecting effects of a predefined size. The used Bayesian techniques are particularly fit to employ such a data-generative model. Since we want to represent data variability by sampling, having probability distributions of distributional parameters is preferable over point estimates.

This thesis will begin with an outline of the mathematical procedures used by the two software packages. This is followed by a definition of the concepts of sample size and power, which in the

high-dimensional setting differ somewhat from the concepts in a more classical sense. The Results section starts with a direct comparison of results obtained with both of the methods. After this, various data-generative settings are discussed, and the conditions resulting in optimal predictions are presented. Functions/software created to carry out RNAseq sample size calculation are given/described. All research was done using publicly available RNAseq datasets, but is also applicable to other (genomic) data-acquiring techniques that generate counts, such as miRNA, ChIP-Seq or SAGE. Functions were constructed and tested, and analyses were performed within the R statistical programming environment [R Core Team, 2014], using R version 3.1.0.

2 Methodology

2.1 `edgeR`: Differential expression analysis via exact tests

A popular RNAseq analysis pipeline is the R software package edgeR [Robinson et al., 2010]. The default setting, suggested by its creators, negative binomial framework, with overdispersion parameter ϕ to allow modelling of variance not captured by a Poisson distribution, and a link function on the mean expression:

$$
\begin{aligned}
Y_{ij} &\sim \mathrm{NB}(\mu_{ij}, \phi_i) \\
\mu_{ij} &= exp(\beta_{0i} + \beta_{1i} X_j) \\
H_{0i} &: \beta_{1i} = 0
\end{aligned}
$$

With the pdf of the NB distribution parameterised as follows:

$$
\mathrm{P}(Y_{ij} = y) = \frac{\Gamma(y + \phi_i^{-1})}{\Gamma(\phi_i^{-1})\Gamma(y+1)} \cdot \left(\frac{\phi_i^{-1}}{\phi_i^{-1} + \mu_{ij}}\right)^{\phi_i^{-1}} \cdot \left(\frac{\mu_{ij}}{\phi_i^{-1} + \mu_{ij}}\right)^{y}
$$

Y_{ij} is the response variable, i.e. the number of times a genomic feature appears in a sample. μ_{ij} is the per-feature average expression, consisting of an intercept β_{0i}, and of a group effect β_{1i}. For this thesis the focus is on the two-group setting, thus X_j takes on values 0 or 1, dependent on the group subject j belongs. In the notation above, the i represents the feature-wise index. The total number of subjects (sometimes called libraries, samples or individuals) will be denoted n, and the number of features p.

Parameter estimation

Estimation of per-feature mean expression is done by assuming that the counts per feature are i. i. d. (i. e. that H_{0i} is true). Then μ_{ij} no longer depends on j, and $\mu_{ij} = \frac{1}{n}\sum_{j=1}^{n} Y_{ij}$ is used as estimator.

Calculation of dispersion ϕ_i involves more steps, starting with calculation of a per-feature dispersion $\hat{\phi}_i$. The per-feature sum $Z_i = \sum_{j=1}^{n} Y_{ij} \sim \text{NB}\left(n \cdot \mu_i, \frac{\phi_i}{n}\right)$ is calculated. By conditioning on Z_i, the μ_i-parameter is removed from the log likelihood function, which allows straightforward maximization for $\hat{\phi}_i$ [Robinson and Smyth, 2008]:

$$l_{Y_{ij}|Z_i}(\hat{\phi}_i) = \sum_{j=1}^{n} log\Gamma(Y_{ij} + \hat{\phi}_i^{-1}) - log\Gamma(Z_i + n\hat{\phi}_i^{-1}) - n \cdot log\Gamma(\hat{\phi}_i^{-1}) + log\Gamma(n\hat{\phi}_i^{-1})$$

However, because it was established that for RNAseq data these conditional dispersion estimators often perform suboptimally [Robinson and Smyth, 2007], they are not directly used as dispersion values. Instead, a common dispersion value ϕ_C is calculated by maximizing the sum of loglikelihoods $\sum_{i=1}^{p} l_{Y_{ij}|Zi}(\hat{\phi}_i)$

Lastly, used dispersions ϕ_i are calculated by shrinking the observed values towards the common likelihood value, arriving at a weighted conditional maximum likelihood value [Robinson and Smyth, 2007]:

$$\text{WL}(\phi_i) = l_{Y_{ij}|Z_i}(\hat{\phi}_i) + \alpha \cdot \sum_{i=1}^{p} l_{Y_{ij}|Zi}(\hat{\phi}_i) \tag{1}$$

α is calculated using empirical Bayesian techniques: by assuming the "observed" conditional dispersion values are distributed according to a hierarchical Gaussian model with mean ϕ_i, it is possible to define and approximate the Bayes posterior mean estimator of ϕ_i:

$$\begin{aligned} \hat{\phi}_i|\phi_i &\sim N(\phi_i, \tau^2_{\phi_i}) \\ \phi_i &\sim N(\phi_0, \tau^2_{\phi_0}) \\ E(\phi_i|\hat{\phi}_i) &= \frac{\hat{\phi}_i/\tau^2_{\phi_i} + \phi_0/\tau^2_{\phi_0}}{1/\tau^2_{\phi_i} + 1/\tau^2_{\phi_0}} \end{aligned}$$

Then, α is chosen such that equation (1) matches this expression. For a full description, we refer to [Robinson and Smyth, 2007]. Practically, this has the effect that there is a lot of weight on the common likelihood when dispersion values are similar, and that α takes on larger values when the variation between $\hat{\phi}_i$'s is high. This has a very beneficial stabilising effect on estimation of ϕ_i, but dispersion estimates are no longer estimated independently.

Hypothesis tests

To test for differential expression between two groups, first the observations per group are added. We will call these partial sums Z_1 and Z_2. Under the null hypothesis of no differential expression, $\beta_1 = 0$, they come from the same negative binomial distribution, only differing by the number of observations in both groups, with n_1 and n_2 the number of subjects in resp. group 1 and 2.

The probability of observing exactly the values Z_1 and Z_2 under H_0, given μ and ϕ as calculated earlier, is calculated. Then, probabilities of all possible observation pairs with this or smaller probabilities are summed, and divided by the total probability of all possible observation pairs. This proportion can be interpreted as a classical p-value to assess significance, and is a version of Fisher's exact test, regularly used for analysis of contingency tables [Fisher, 1992]. Below, the i-index is dropped, but calculations are done per feature:

$$Z_1 = \sum_{j=1}^{n1} Y_j \qquad\qquad Z_2 = \sum_{j=n1+1}^{n} Y_j$$
$$Z_1 \sim \mathrm{NB}(n_1 \cdot \mu, \phi/n_1) \qquad\qquad Z_2 \sim \mathrm{NB}(n_2 \cdot \mu, \phi/n_2)$$

Let $Z_1 + Z_2 = Z$, and let (z_1, z_2) be any integer pair for which $z_1 + z_2 = Z$. Then

$$P = \frac{\sum_{(z_1,z_2);z_1+z_2=Z} \Pr(z_1)\Pr(z_2) \cdot I_{\Pr(z_1)\Pr(z_2)\leq\Pr(Z_1)\Pr(Z_2)}}{\sum_{z_1,z_2;z_1+z_2=Z} \Pr(z_1)\Pr(z_2)}$$

Besides p-values to assess statistical significance of the observed expression difference, edgeR provides the user with an effect size, in the form of a fold change on the $\log_2$-scale. Computation time, from normalization to testing significance on a 10.000×100 dataset takes roughly 12 seconds on a one-year old Intel i7 processor. This, together with the fact that its setup is often recognizable and easily understood for people familiar with the well-known Limma R package for the analysis of microarray data [Smyth, 2005], has made it the package of choice for many biologists carrying out RNAseq experiments.

2.2 `ShrinkBayes`: empirical Bayesian inference using INLA

ShrinkBayes is another R software package created for the analysis of RNAseq data, using a Bayesian approach [Van de Wiel et al., 2014]. The model setting used during this analysis is again build around a negative binomial distribution, but with an (optional) additional point mass on 0, to account for excess zeroes, often observed in RNAseq data. Also, Bayesian techniques are employed for parameter estimation, so prior distributions are assigned.

$$\begin{aligned}
Y_{ij} &\sim \mathrm{ZI} - \mathrm{NB}(q_{0i}, \mu_{ij}, \phi_i) \\
\mu_{ij} &= exp(\beta_{0i} + \beta_{1i} X_{ij}) \\
\beta_{0i} &\sim \mathrm{N}(\mu_{\beta_0}, \tau^2_{\beta_0}) \\
\log(\phi_i) &\sim \mathrm{N}(\mu_\phi, \tau^2_\phi) \\
\beta_{1i} &\sim F_{\beta_1} = p_0 \cdot \delta(x) + (1 - p_0) \cdot F_{\beta_1, nonzero}(x)
\end{aligned}$$

The ZI-NB probability density function is given by

$$f_{\mathrm{ZI-NB}}(x) = q_0 \cdot \delta(x) + (1 - q_0) \cdot f_{\mathrm{NB}}(x)$$

$F_{\beta_1,nonzero}$ is as of yet unspecified, $\delta(x)$ denotes the Dirac delta function, p_0 effect null probability, q_0 count null probability, and $f_{\mathrm{NB}}(x)$ the negative binomial pdf.

In the notation above, two assumptions that are the default in ShrinkBayes (but not a necessity) are already set: Gaussian distributions for intercept and dispersion. Usually there is one central parameter of interest, on which attention is focused. This is also the case for our two-group setting, where accurately modelling effect size is the most important, and reflected in the notation of F_{β_1}. This function can take a variety of forms and is not yet determined. We can view the set of observed counts from feature i as samples from a (zero-inflated) negative binomial density with mean $exp(\beta_{0i} + \beta_{1i}X_j)$ and dispersion ϕ_i. These coefficients are draws from their respective hyper distributions.

Parameter estimation: Iterative joint procedure
Estimation of coefficient priors is done iteratively [Van de Wiel et al., 2012]:

1. Choose initial Gaussian distributions for all prior parameters (also β_1), such that there is little information in the prior distribution
2. Update prior distribution, one parameter at the time: Given priors and observations Y_{ij}, approximate marginal posterior distributions $\tilde{\pi}(\beta_{0i}|Y_{ij})$ using INLA[1]
3. Take a very large sample (e.g. 100.000 values) from the set of p posteriors
4. Calculate Gaussian maximum likelihood estimators, replace μ_{β_0} and τ_{β_0}
5. Repeat from 2 for next parameter, until convergence

The EM-like algorithm described above is called the *iterative joint procedure* within ShrinkBayes. Information from all features is used to update estimates for the hyper parameters μ and τ. Results of the iterative joint procedure depend solely on marginal distributions, not on joint posteriors, which is an important benefit when the number of parameters is large.

[1]Calculating the Bayesian posterior from prior and data requires integration over the marginal likelihood. Doing this for every feature in a reasonably sized RNAseq dataset is computationally impossible. For the same reason, and because of the inevitable correlations between regression parameters, MCMC sampling is not a viable alternative. By assigning all coefficients a Gaussian prior, we have created a latent Gaussian model. For this class of models, accurate and comparatively very fast statistical inference can be acquired via integrated nested Laplace approximation. Integrated nested Laplace approximation makes use of the fact that marginal posterior distributions of any parameter can be estimated by Laplace approximation, provided that all other parameters have a Gaussian form. Integrated nested Laplace approximation is implemented in R within the INLA R package [Rue et al., 2009] and ShrinkBayes.

Parameter estimation: Iterative marginal procedure

At this stage we have a model where every regression parameter has a Gaussian form. As mentioned, in the two-group setting we would like to focus on the coefficient reflecting differential expression between the groups, β_1, and have some additional flexibility for modelling this parameter. This is achieved by again taking a large sample from the set of final posteriors (calculated under Gaussian priors) and use this to approximate the mixture of posteriors. This time, instead of calculating Gaussian ML estimators, a distribution with more flexible shape is fitted.

Because we now have a prior of non-Gaussian form, INLA can't be used to approximate posteriors. Instead, nonparametric posteriors $\pi_{\mathrm{NP}}(\beta_{1i}|Y_i)$ are calculated by multiplying the parametric estimates by the ratio of their prior probabilities, of either Gaussian or more complicated shape:

$$\pi_{\mathrm{NP}}(\beta_{1i}|Y_i) = C \cdot \pi_{\mathrm{P}}(\beta_1|Y_i) \cdot \frac{\mathrm{F}_{\mathrm{NP}}(\beta_{1i})}{\mathrm{F}_{\mathrm{P}}(\beta_1)}$$

Within ShrinkBayes documentation and this thesis, the updating of one parameter of interest via the algorithm described above is referred to as the *iterative marginal procedure*.

Via 1) An empirical Bayesian approach, 2) Zero-inflation on the count distributions and 3) Nonparametric modelling of parameters of interest, ShrinkBayes has been shown to produce more reproducible results than other RNAseq analysis pipelines, especially for small sample sizes, as will be the case during sample size calculations. This comes at a cost: Even with the help of INLA and parallelization of the computationally more intensive steps, a typical ShrinkBayes analysis, from estimating priors to calculating updated posteriors, on a 10.000x100-sized dataset will take between 36 up to 48 hours on 6 cores with xxx GB RAM memory of a Linux cluster (longer under Microsoft Windows). Pilot samples will not have quite this size, but computational efficiency is a nontrivial factor in the implementation of our proposed sample size calculation approach.

Empirical Bayes approach

The strategy of borrowing information from the complete set of features to help stabilize results per feature, is commonly used in genomic analysis methods. Suppose that the number of features measured is comparatively large (e.g. $p \geq 1000$), and that the majority of these features is not differentially expressed. Then, the combined set of observations from all features is large enough to accurately represent the response variance under the null hypothesis of no differential expression, and can provide the researcher with powerful distributional information about his data.

In a Bayesian framework, a priori information about the data can be expressed in the prior distribution, which is exactly what is done in the iterative joint procedure. Estimating prior distribution from observed data is called empirical Bayes statistics. In genomic research,

using information from the complete set of features to do per-feature inference is not limited to Bayesian approaches, but implemented in various different ways: as we've seen in section 2.1, edgeR also utilizes a common dispersion to achieve more stable estimates for per-feature dispersions.

2.3 The goal of sample size calculation: Defining signal

Individual hypothesis testing

Consider a test statistic T. For individual hypothesis testing, a null hypothesis is rejected when the corresponding observed test statistic has a low probability of occurring under the null hypothesis, lower than a predefined confidence level α. A test statistic cutoff t_α is defined, such that

$$\mathrm{P}_0(T_n > t_\alpha) = \alpha$$

with T_n the observed test statistic. The probability distribution of T_n depends on the used test and on sample size n. The classical power concept in this scenario is

$$\mathrm{P}_\Delta(T_n > t_\alpha)$$

with $\mathrm{P}_\Delta(T_n)$ the the probability distribution of T at sample size n, given an effect size of exactly Δ. In the negative binomial setting, the distribution of a test statistic is dependent on average expression μ and dispersion ϕ. When these nuisance parameters are available (estimated) so that the probability distribution of the test statistics is known, this quantity varies with sample size, confidence level and effect size cutoff: $K(n, \alpha, \Delta)$.

Multiple testing

In high-dimensional research, hypothesis testing is not performed individually. Instead, the rejection criterion is adjusted to correct for multiple testing (see intermezzo below). Rejection of a hypothesis then becomes dependent on the set of hypotheses and therefore on the distribution of present effects. Power can no longer be calculated for any individual Δ, but involves integration over this distribution:

$$\int_{\Delta=\delta}^{\infty} \mathrm{P}_\Delta(T_n > \tilde{t}_\alpha \mid \Delta) f(\Delta) d\Delta \tag{2}$$

$$= \quad \mathrm{P}_\Delta(T_n > \tilde{t}_\alpha \mid \Delta \geq \delta) \tag{3}$$

where $f(\Delta)$ is the density function of effect sizes, and where the critical value $\tilde{t}_\alpha$ is now based on a multiple testing criterion, in our case FDR [Benjamini and Hochberg, 1995]. Integrating over all effects larger than a predefined cutoff value δ leads to an *average* detection rate. This

Table 1: Combinations of hypothesis state and rejection, with m the number of hypotheses tested. The symbol in brackets is used to indicate the number of hypotheses belonging in every category.

	H_0 accepted	H_0 rejected	
H_0 true	True Negatives	False Positives (V)	Actual Negatives ($m - m_1 = m_0$)
H_0 false	False Negatives	True Positives (U)	Actual Positives (m_1)
	$m - R$	Rejections (R)	Total (m)

average power is the quantity used in this project to determine required sample size, and will be denoted as $D(n, \alpha, \delta)$

Abundance

There is another difference between high-dimensional and individual sample size calculation: The absolute number of rejections is a very relevant factor. The arguments for assessing the total number of rejections are very practical. It would be undesirable to spend resources on finding hundreds of features with differential expression, when only a fraction can be followed up upon [2]. Hence we argue that in high-dimensional genomic research, to define the experimental goal for which sample size is calculated, we need to consider not just (average) power, but also a second component: the total number of rejections for given sample size and confidence level, which we call effect *abundance*, denoted by $A(n_{sample}, \alpha)$.

A simulation-based approach

A data-generative model is created via empirical Bayesian shrinkage, and generated datasets are analysed by exact testing of differential expression. Following the two-group comparison setting defined as in section 2.1, we define the null hypothesis

$$H_{0i} : \beta_{1i} = 0$$

The general Δ used above is in our two-group setting replaced by β_1. The exact testing procedure results directly in probabilities of encountering the observed values under H_0. These are transformed to produce adjusted p-values. Our test statistic cutoff α is equal to the desired maximum allowed FDR. Hence, the general formula for average power in a multiple testing setting as given

[2] To give an example of this, think of the creation of genetic screening tools, appearing regularly for diseases with known heritable factors such as Parkinson's, or to improve early detection of diseases where this is vital for complete recovery, like breast cancer. Screening tools use relatively low number of markers with high predictive value

in (3) is in our specific case equivalent to

$$\begin{aligned}
\mathrm{P}_{\Delta}(T_n > \tilde{t}_{\alpha} \mid \Delta \geq \delta) &= \mathrm{P}(P_{adj,i} \leq \alpha \mid \beta_{1i} \geq \delta) \\
&= \frac{\mathrm{P}(P_{adj,i} \leq \alpha,\ \beta_{1i} \geq \delta)}{\mathrm{P}(\beta_{1i} \geq \delta)} \\
&= \frac{\mathrm{E}(\sum_i I_{P_{adj,i} \leq \alpha} \cdot I_{\beta_{1i} \geq \delta}/m)}{\mathrm{E}(\sum_i I_{\beta_{1i} \geq \delta}/m)} \\
&= \frac{\mathrm{E}(U/m)}{\mathrm{E}(m_1/m)} \\
&= \frac{\mathrm{E}(U)}{\mathrm{E}(m_1)}
\end{aligned}$$

Because we restrict ourselves to sampling settings where m is never 0, conditioning on this is unnecessary. To calculate $\mathrm{E}(U)/\mathrm{E}(m_1)$, knowledge on the values of β_{1i} is required, which is never available for real data. During simulation, the values from which we sample are known, which makes it possible to arrive at an estimate of theoretical expected average power.

To summarize, the goal we define is: to predict, from a pilot dataset, for various sample sizes

- the average power for detecting effects of predefined magnitude, dependent of sample size and confidence level: $\frac{\mathrm{E}(U)}{\mathrm{E}(m_1)}$, denoted by $D(n_{sample}, \delta, \alpha)$
- the total number of rejections $\mathrm{E}\,(R)$ at a given sample size and confidence level, denoted by $A(n_{sample}, \alpha)$

We do this by creation of a representative data-generative model, iteratively generating and analysing datasets, and storing relevant results per iteration. A schematic summary of the steps involved can be found in figure 1.

Alternative approaches

Several analytical methods of power calculation while controlling for FDR have been proposed. Most of these are designed specifically for microarray experiments, where responses are continuous and effect sizes can be approximated by Gaussian distributions. The most widespread is the method described in [Ferreira and Zwinderman, 2006], where explicit equations for both average power and sample size are given. This framework was expanded by Van Iterson et al. [2013], to include other than Gaussian distributions of test statistics including the χ^2. This makes it applicable to several GLM-based RNAseq count analysis methods, but not to the exact testing procedure. Li et al. [2013] use the Weighted Maximum Likelihood estimates from section 2.1 and then via the score statistic arrive at a closed-form formula to estimate power. Another way of estimating needed sample size in RNAseq experiments, again for user-specified fold change,

power and type I error level is given in [Hart et al., 2013]. Here, a quite simple solution for estimating sample size is derived via the derivative of the negative binomial likelihood. The brevity of their solution is due to the fact that multiple testing is not considered.

All methods directly including FDR in power calculations suffer from two drawbacks. First, controlling FDR directly when computing power or sample size is very reliant on accurate estimation of p_1. There are several ways to do this based on a set of p-values [Langaas and Lindqvist, 2005] (and references therein), but these methods often perform suboptimal, illustrations of which will also be shown later. Second, the variation in RNAseq data is difficult to capture. It consists of a technical and a biological part, of which only the technical part can be reduced by increasing the number of replicates (n_{sample}), but also by increasing sequencing depth, i.e. the reads per subject. The biological variation is highly correlated to, but not determined by the mean expression level, and the ratio of biological to technical variation finally determines what the effect of sample size is on the ability to reject a null hypothesis when the alternative is true. This complicates and hampers the approximation of RNAseq test statistics by theoretical sampling distributions.

The drawbacks of the few analytical power calculation tools available for RNAseq data are the primary motivation behind the creation of our simulation-based sample size calculation tool. The empirical Bayesian techniques we employ to estimate data-generative settings have been shown to outperform the alternative available RNAseq modelling approaches in terms of reproducibility, particularly when the number of available subjects is low (e.g. in pilot experiments) [Van de Wiel et al., 2012]. Intrinsic to the Bayesian approach is that it puts more emphasis on probability distributions of parameters than on optimal point estimates. This makes it naturally suitable for the data-generative mechanism we create. We aim to overcome the challenges in power calculation posed by the complex nature of RNAseq data by approximating data structure instead of test statistics.

Very recently, a simulation-based approach quite comparable to our proposed method was published by [Shyr and Li, 2014]. The main difference is in the parameter estimates used for generating counts: they use the estimates from edgeR's procedures, while we estimate negative binomial parameter coefficients via iterative Bayesian updating. These authors have only tested their method using information of extremely large ($n \geq 600$) datasets, and thus bypass the main challenge in sample size calculation, of having limited information at hand. Another negative binomial model-based simulation scheme is provided in [Wu et al., 2014](advance access). The authors, like us, advocate the necessity of defining multiple criteria in RNAseq sample size calculation, and state that *prospective* power should be evaluated for various combinations of these criteria. The number of factors influencing sample size determination makes simulation-based power calculation a logical approach. Interestingly, to assess performance of their method, they use two ReCount datasets with 41 and 21 number of subjects, and only compare relative sample

size calculation results. An objective performance assessment of their method is lacking.

False Discovery Rate

When simultaneously testing multiple hypotheses and rejecting them whenever an individual type I error probability is $\leq \alpha$, the expected probability of not making any type I error multiplies with a factor $(1 - \alpha)$ per test [Tukey, 1977]. This is not trivial: for instance, at a standard confidence level of 0,05 and the testing of ten hypotheses, the probability of at least one false positive has already increased to more than 40%.

To prevent an unacceptable increase in the number of false positives in high-dimensional investigations, the observed p-values are transformed. In this investigation, expected False Discovery Rate [Benjamini and Hochberg, 1995] is used as a rejection criterium. FDR is defined as $\mathrm{E}\left(\frac{V}{R} \mid R > 0\right)$ (see Table 1). It has a nice statistical interpretation: When we reject all null hypotheses corresponding to an expected FDR $\leq 0,1$, we expect that 10% of these rejections are false positives. It is important to note that this is a statement on the set: the above conclusion is not equivalent to stating that any one of these rejections has a 10% chance of being a false positive.

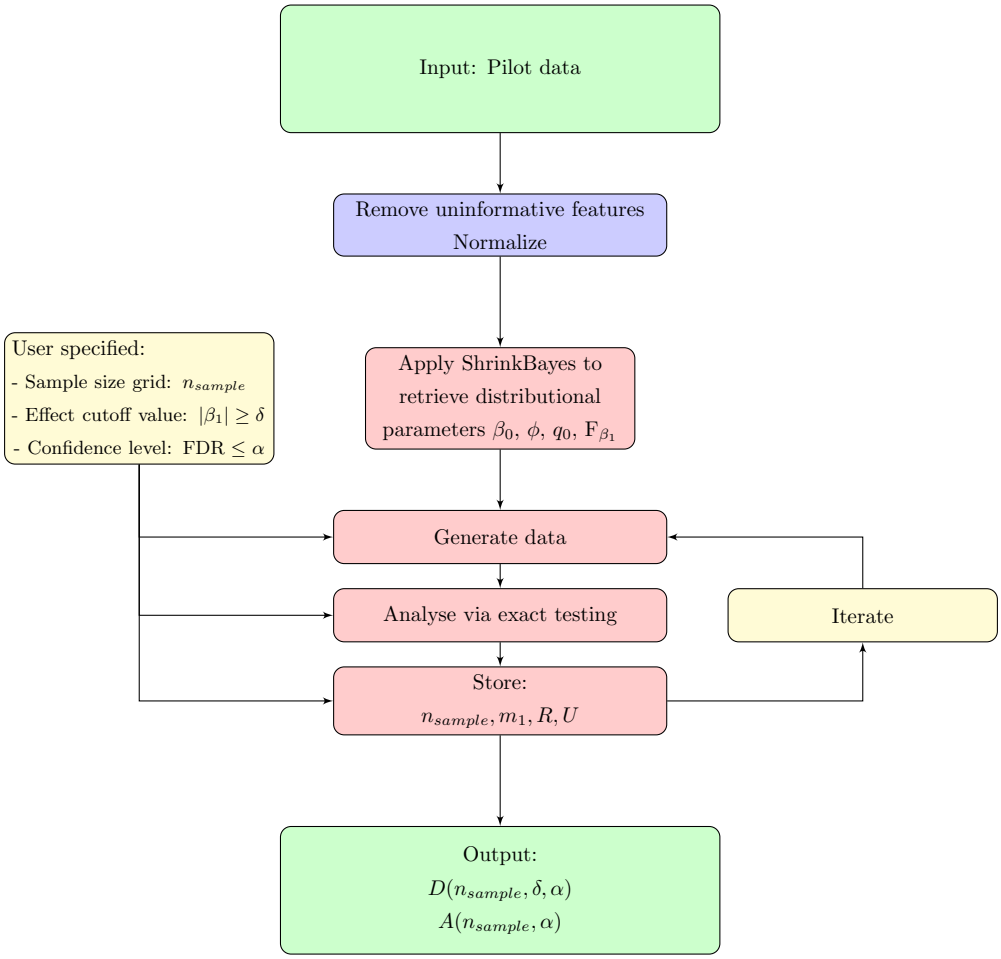

Figure 1: Schematic representation of proposed algorithm

2.4 Data-generative settings

In the next section we start by comparing the output generated via the two packages. This gives an overview of parameter distributions, and can then be used to assess various data-generative settings. The first step is to find data-generative settings that give the highest reproducibility on small sample sizes compared to larger analyses, while correctly preserving information about differential expression. To mimic the setting of a pilot experiment, we take equal sized subsamples from a larger RNAseq dataset. These subsamples, of for example 5+5 or 10+10 subjects, act as pilot studies, which contain all the information available for sample size calculation. The performance of our algorithm with various prior distributions is then assessed by comparing predicted abundances to true abundance (found by running exact tests on the complete datasets), and estimated prior parameters (p_0) to the results obtained via iterative empirical Bayesian analysis of the complete datasets.

To improve inference, rows with more than 80% zero entries in the complete dataset were removed (as advised by EdgeR's creators). In the datasets used, the preprocessing steps of aligning reads were already carried out, according to manuscripts available from repositories. Normalization between subjects was carried out according to the "trimmed mean of M-values"-method [Robinson and Oshlack, 2010], after subsampling from the complete set.

Parameter distributions

Primary interest is on the β_1 parameter, reflecting differential expression difference between the two groups. It is most important that the signal used for count generation resembles the actual signal in the data. Therefore we want more flexibility than a regular Gaussian distribution can provide while modelling this variable. As explained in section 2.2, ShrinkBayes offers a way of reshaping this prior into more pliable distributions with various constraints. We believe most of our features to be non-differentially expressed and are testing the null hypothesis of regression parameters equalling 0, hence only functions with a point mass on zero are considered.

Shape of the prior distribution is updated by taking a large sample from the set of posteriors, and using this to estimate distributional characteristics. After evaluation of several distributions, we discuss three distinct shapes: a mixture, log concave nonparametric and nonparametric distribution.

The mixture distribution is called so because it is a combination of two Gaussian distributions and a point mass, parameterised as follows:

$$f_{mixt}(x) = p_0 \cdot \delta(x) + (1 - p_0) \cdot \left\{ \quad p_{neg} f_N(-\mu, \sigma) \quad + \quad (1 - p_{neg}) f_N(\mu, \sigma) \quad \right\}$$

with $f_N(\mu, \sigma)$ the regular Gaussian pdf with mean μ and standard deviation σ. In [Van de Wiel et al., 2014], this distribution, called the "Spike-Gauss-Gauss" is generally recommended, for its "good performance in both FDR estimation and power". It is also noted that FDR estimates

obtained under this prior might be conservative. During high-dimensional genomic research this is generally not considered a large disadvantage, but the characteristics resulting in conservative FDR's may very well also give rise to underestimation of D and A.

A so-called nonparametric function can be fitted to the empirical sample of posteriors when extra flexibility is needed. The nonparametric prior shape is actually a mixture of Gaussian kernels (see also `?density` in R or [Venables and B.D.Ripley, 2002]). Such distributions can take on quite irregular forms, providing the user with as much freedom as possible in finding the prior shape best describing the observations. Van de Wiel et al. [2014] mention this might be especially useful in the situation when there is limited information available. Too much flexibility could however result in a "running wild" of the optimisation algorithm in the case of an unlucky pilot sample. This is exactly the problem we need to avoid in order to do stable power predictions.

There are ways of restricting the non-parametric flexibility somewhat: by enforcing symmetry, unimodality and/or log-concavity on $F_{\beta_{1,nonzero}}$. These constraints all yield less irregular distributions than the unconstrained nonparametric one, and we expect them to result in more similarity between pilot analyses. Several (combinations of) options were tried. Most notable differences with default nonparametric distribution were obtained by enforcing log-concavity. The algorithm used for finding the optimal log-concave density is described in [Dümbgen and Rufibach, 2011] and implemented in the `logcondens` R package.

Sets of coefficients

After completing an empirical Bayesian analysis, done in order to estimate data-generative settings, there are multiple ways of sampling new counts. For intercept and dispersion, parameter distributions are not updated to a nonparametric form. Thus, fresh coefficient vectors can be generated from prior distributions (e.g. $f_N(\mu_{\beta_o}, \tau^2_{\beta_0})$), or by using characteristics of the marginal posteriors per feature (mean and variance or expected value). Sampling from priors is done independently. The use of marginal posteriors on the other hand will preserve data structure, which as we will see in Section 3.2 is present.

As for the differential effect parameter, these, but also updated priors and sets of posterior distributions are available, both in the form of INLA density objects. Using the unupdated Gaussian information is unadvisable, because there is not yet a point mass on zero. Making every feature in the data-generative model differentially expressed results in conditions not at all resemblant of what we're trying to simulate.

The largest gain in computation time can be accomplished by sampling β_1-values from the updated prior distribution instead of from the set of posteriors, because this makes the updating of p posteriors obsolete and saves a lot of memory used for storing these large objects. Possible correlation between effect size and other coefficients in the data-generative model is then lost. Via the resampling of negative binomial parameters, we will discover whether there are correlations

between data-generative settings that have an effect on estimated power and abundance.

2.5 Datasets used

Three different RNAseq datasets were used for calibration and testing of our proposed algorithm.

Pickrell

An extensively studied RNAseq dataset available from the ReCount RNAseq data archive was used for initial exploration [Frazee et al., 2011, Pickrell et al., 2010]. Being one of the first large-scale RNAseq experiments ever conducted, this set has become somewhat of a benchmark dataset, often used as an example dataset in modelling efforts. Its homogeneity makes it a relatively "easy" target for new modelling approaches.

The Pickrell set contains read counts of 52.580 features of 58 Nigerian individuals. Rows with less than 20% nonzero entries were removed from the data, which leaves 9482 features. Gender was taken as the factor of interest. Of these, standard exact testing analysis yields 37 rejections at $\alpha = 0,1$, hence observed $p_0 = \frac{9482 - 37}{9482} = 0,996$.

Due to privacy issues, the amount of phenotypical information is very limited. Prediction of A and D between males and females in the Pickrell dataset is quite challenging, because of the high p_0. The smaller the order of magnitude of signal in a dataset, the larger the impact of extremes caused by random count generation will be. To make life somewhat easier on ourselves, a lot of the calculations were done on a subset of the Pickrell dataset where the proportion of differential expression between males and females is larger than over the full genome: features located on the human sex chromosomes. This has the added benefit of reducing computational costs in initial exploration of data-generative settings.

Out of the 296 features located on X-Y chromosomes, a standard edgeR analysis on the same 58-sized dataset that was used earlier results in 12 rejections, making the observed proportion of differentially expressed features just over 4,5%.

VUmc miRNAseq

A second dataset, containing counts of microRNA strands, obtained from an associated research group at the VUmc was used to validate the results found with Pickrell data. After removal of rows with excess zeroes, this set contains $1241 \cdot 52$ observations, on 26 individuals. Primary tumor versus metastatic tissue is compared, with individual included as a fixed parameter.

Though we are confident about conditions being comparable during data collection, the fact that samples are paired, and from different organs makes this dataset more challenging. To remain in the two-group comparison setting, information on the time of sampling and previous

chemotherapy was ignored. The organ from which metastatic resections were taken was available. This was not included in the model, but during sampling of pilot samples, we ensured that the ratio of organs in pilot sets remained comparable to that in the complete set. Pilots of sample size 10 are all non-overlapping subsamples, as are respectively pilot 1 versus 2 and 3 versus 4 of the 20-sized subsets.

TCGA Lung Squamous Cell Carcinoma

Finally, a large set of RNAseq data for Lunch Squamous Cell Carcinoma tumor samples was downloaded from The Cancer Genome Atlas. In 2005 this data portal was set up to enhance investigation into the genetic basis of tumor growth [http://cancergenome.nih.gov/]. The goal was to make oncological data globally accessible to any researcher with the desire to spend resources on genomic cancer research. Counts were generated with identical machines and aligned to a UCSC reference transcriptome fixed in december 2009, via the SeqWare analysis software developed for TCGA RNAseq analysis. Data from TCGA was collected in research centers throughout the United States.

For numerical reasons in iteratively estimating data-generative settings, the largest available batches adding up to $\geq$ 150 subjects were downloaded. Removing technical replicates and metastatic tumor samples left just over 120 subjects, of which forty were non-recurrent. After removal of non-informative features a dataset of $19755 \cdot 80$ features remains. Exact testing of differential expression between recurrent and non-recurrent tumours gives 977 rejections at $\alpha = 0,1$, hence an observed p_0 of $0,95$.

Cues for cell division lie in multiple signalling pathways, and mutations in all parts of these pathways can be a cause for erratic division. For these reasons, this dataset is likely to be more heterogeneous than the first. Hence we have chosen this as a final test for our sample size calculation algorithm.

3 Results

3.1 Functions

The process of simulation-based power calculation, as summarised in figure 1 was automated in two functions. The first, `predictPower()`, generates a dataset of user-specified size from input negative binomial coefficient vectors and returns the empirical power and abundance values of an exact testing analysis on this simulated dataset. The second function, `powerCurve()`, iteratively applies the `predictPower()`-function for a vector of sample sizes, hence returning an estimate of average power and abundance, for a range of sample sizes. For more direct compatibility with the empirical Bayesian data-generative parameter estimation we suggest, it requires as input not

vectors of negative binomial parameters, but ShrinkBayes-objects, from which these parameters are extracted. On the pages below, documentation for these functions is provided in the style of a standard R help file. The actual R code is provided in appendix A.

Besides the two functions described above, a lot of functions useful to the more in-depth ShrinkBayes-user were created. The extraction of negative binomial coefficients (i.e. expected values of the marginal distributions), plotting of and sampling from Gaussian prior, updated prior and updated posterior distributions are now conveniently standardized within functions. Code for these can be found in appendix B. They were annotated in such a way, that they should be understandable to anyone with some experience in handling high-dimensional count datasets.

R documentation

of 'predictPower.Rd' etc.

December 8, 2014

---

predictPower — *Simulation-based power calculation for negative binomial count data*

Description

Computes power and effect abundance for the detection of differentially expressed features in a simulated dataset. Requires (possibly zero-inflated) negative binomial coefficients as input, generates a dataset and performs exact tests of differential expression based on (Robinson, 2007). Assumes an equalsized two-group comparison. For more info, see (Retel, 2014).

Usage

```
predictPower(nsample=100, b0=b0vect, b1=b1vect, size=sizevect, q0=q0vect,
delta=0.5, alpha=0.1)
```

Arguments

nsample	Number of subjects/columns to generate
b0	Numeric vector of intercept (β_0) values
b1	Numeric vector of effect (β_1) values
size	Numeric vector of log(size) (n) values
q0	Numeric vector of count zero probability (q_0) values. If left unspecified, count generation is from a negative binomial distribution
delta	Numeric. Effect size cutoff: average power is calculated for all $\|\beta_1\| \geq \delta$
alpha	Numeric. Confidence level, maximum estimated False Discovery Rate for which hypotheses are rejected

Details

The data-generative model is specified as follows:

$$Y_{ij} \sim \text{ZI} - \text{NB}(q_{0i}, \mu_{ij}, n_i)$$

$$\mu_{ij} = exp(\beta_{0i} + \beta_{1i} X_{ij})$$

With the ZI-NB probability density function given by

$$f_{\text{ZI}-\text{NB}}(x) = q_0 \cdot \delta(x) + (1 - q_0) \cdot \frac{\Gamma(y+n)}{\Gamma(n)\Gamma(y+1)} \cdot \left(\frac{n}{n+\mu}\right)^n \cdot \left(\frac{\mu}{n+\mu}\right)^y$$

Value

An object of class `data.frame()` of dimension `1 * 5` with columns the number of Actual positives, Rejections, True positives, Actual positives with $\beta_1 \geq \delta$ and True positives with $\beta_1 \geq \delta$.

Author(s)

Cas Retel

See Also

powerCurve

Examples

```
## randomly generate negative binomial parameter vectors
set.seed(345); p <- 2000  # number of genomic features to simulate

b0 <- rnorm(p, 2, 2)
size <- rnorm(p, 2, 2)
b1 <- c(sign(runif(.05*p, -1, 1)) * rnorm(p1, 2, 1), rep(0, .95*p))
q0vect <- rexp(p, rate=20)

## Calculate power to detect effects $\geq 0.5$ on a set of 100 subjects
power <- predictPower(nsample=100, b0=b0vect, b1=b1vect, size=sizevect, q0=q0vect,
delta=0.5, alpha=0.1)
```

`powerCurve` *Iterative simulation-based power calculation for negative binomial count data using parallel computing*

Description

Computes power and effect abundance for the detection of #differentially expressed features in a simulated dataset of negative binomial counts. Requires (possibly #zero-inflated) negative binomial coefficients as input, generates a dataset and #performs exact tests of differential expression based on (Robinson, 2007). #Assumes an equalsized two-group comparison. For more info, see (Retel, 2014).

Description

Computes power and effect abundance for a grid of sample sizes, for the detection of differentially expressed features in a simulated dataset of negative binomial counts. Negative binomial coefficients are extracted from input `ShrinkBayes::FitAllShrink` and `ShrinkBayes::UpdatePrior`-objects .

Usage

```
powerCurve(fasobj, uprobj, nsample=c(10, 20, 40, 60, 80, 100),
delta=0.5, alpha=0.1, niter=50, ncpus=4)
```

Arguments

fasobj	A list containing shrunken marginal Gaussian distributions of negative binomial parameters for every feature, as returned by ShrinkBayes::FitAllShrink
uprobj	A list containing updated prior probability of effect parameter (difference between groups), as returned by ShrinkBayes::MixtureUpdatePrior or ShrinkBayes::NonParaUpdatePrior
nsample	Numeric vector. Sample sizes for which power and abundance are calculated
delta	Numeric. Effect size cutoff: Power is calculated for all $\|\beta_1\| \geq \delta$. It is advisable to check updated prior probabilities and base the threshold both on biological arguments and the range of this distribution
alpha	Numeric. Confidence level, maximum estimated False Discovery Rate for which hypotheses are rejected
niter	Numeric. Number of datasets to generate and analyse per sample size
ncpus	Numeric. Number of computing cores to use

Details

Iteratively applies predictPower-function niter times per value of nsample, implementing parallel computing via snowfall package for more efficient computing.

Value

An object of class data.frame of dimension length(nsample) * 6 with columns Sample size, the number of Actual positives, Rejections, True positives, Actual positives with $\beta_1 \geq \delta$ and True positives with $\beta_1 \geq \delta$, averaged per sample size value.

Author(s)

Cas Retel

See Also

predictPower, ShrinkBayes::FitAllShrink, ShrinkBayes::MixtureUpdatePrior, ShrinkBayes::NonParaUpdatePrior

Examples

```
## Run iterative empirical Bayesian analysis
abc.ss <- ShrinkSeq(formula, data, fams="zinb", ...)
abc.fas <- FitAllShrink(formula, data, abc.ss, ...)
abc.fas0 <- FitAllShrink(formula0, data, abc.ss, ...)
abc.npupr <- NonParaUpdatePrior(abc.fas, abc.fas0, modus="fixed", shrinkpara="groupfactor")

## Calculate power to detect effects $\geq 0.5$
abc.power <- powerCurve(abc.fas, abc.npupr, delta=0.5, nsample=20*(1:5))
```

3.2 Empirical Bayes and Weighted Maximum Likelihood: Comparing output

In both approaches, model settings are largely comparable. We now proceed with a comparison of the results they produce, to give the reader an overview of produced values in a standard RNAseq analysis and to assess where the methods differ. These areas are likely candidates to produce incongruence in prediction, and will be monitored more closely.

Though both packages use the same model (with or without zero-inflation), the negative binomial distributions are parameterised in a different way. Marginal distributions from ShrinkBayes are obtained via INLA's maximum likelihood, which uses a parameter called size (n) as dispersion parameter. The prior is defined on the scale of $\log(n)$. In edgeR, the dispersion parameter is denoted by $\phi = n^{-1}$. Larger ϕ's correspond to less overdispersion, with the NB pdf equalling a Poisson as $\phi \to \infty$.

Figure 2 up to 9 show coefficients of effect size and dispersion, either obtained via iterative Bayesian updating or conditional weighted maximum likelihood. Analyses were performed on a 10+10-sized sample from the complete Pickrell dataset. Unless stated otherwise, estimates were obtained within a negative binomial model, without zero-inflation. Conditional weighted maximum likelihood gives point estimates of average count, dispersion and group difference. Iterative Bayesian updating, however, returns the estimated Gaussian parameters of marginal posterior distributions. Below, expected values of these posteriors are used to compare point estimates to. Since $^1/_{\mathrm{E}\left[\ \log(n)\ \right]} \neq \mathrm{E}\left[\ \log\left(\frac{1}{n}\right)\ \right]$, dispersion estimates are compared on the scale of the prior, i. e. log(size).

Dispersion

Figure 2 shows that there is some correlation between dispersion and expression. This is a known phenomenon in RNAseq data, and using a zero-inflated model (like we will later) should reduce this correlation [Van de Wiel et al., 2012]. For moderate to highly expressed features the two methods result in similar dispersion estimates, with those from ShrinkBayes on average being slightly smaller. For features with low average expression, there is a group of features with a distinctly lower $\log(n)$ value than $\log\left(\frac{1}{\phi}\right)$.

Effect size

Effect sizes can be viewed in figure 4, 5 and 7. On the left, only β_1 values from ShrinkBayes are plotted. Log fold changes show an entirely similar pattern, which can be deduced from figure 7. The difference in magnitude is due to shrinkage: logFC's are estimated using only the information per feature, while β_1-values are estimated under an empirically estimated prior. Compared to the situation where the prior is uninformative, this causes a shift in parameter values towards zero. There is however a very strong correlation between logFC's and β_1's, which

is useful information: When setting a threshold effect size value δ, it does not matter whether effects are selected via β_1 or logFC.

Count null probability

Coefficient estimates discussed above were all generated within a NB binomial setting. Figure 2 indicates that dispersion is not independent of intercept. Addition of a point null probability on count values Y_{ij} should remove most of this dependence. A ZI-NB model was fitted, and estimated q_0-values are shown in figure 6. Addition of a point mass on zero has an effect on estimation of other coefficients, but only when a feature (data row) has zero values. Then, the q_0 effectively removes some weight from these zero-observations in estimating β_0, $\log(n)$ and β_1. This leads to higher estimated intercept estimations, see figure 8 and increased variance estimates seen in figure 9.

3.3 Data-generative settings: β_1 prior

Iterative empirical Bayesian analysis was applied to equal-sized subsamples of the Pickrell XY-dataset of size 5+5 and 10+10, acting as pilots. Resulting priors can be viewed in figure 10. No direct conclusions were based on these pictures, but they are included to give an image of the differences in shape and scale. Note that these are the nonzero parts of the distributions: to represent the prior densities, they should be scaled to corresponding p_0 estimates, which are shown in table 2. Results of iterative Bayesian analysis on the full 58-sized set are also shown, to compare the pilot results to. The signal abundance found by conditional weighted maximum likelihood on the full 58-sized set is 12, which corresponds to a p_0 of $\frac{296-12}{296} = 96\%$.

Mixture

Mixture updating on small samples (5+5) sometimes results in extremely low p_0 estimates. This is accompanied by nonzero parts of the density densely concentrated around zero. These are situations in which the iterative marginal procedure has performed sub optimally: The point mass in the effect size probability function was designed specifically to counter estimation of such a prior shape, with an unrealistically large proportion of differential expression, with most group differences of trivial size.

Nonparametric

Nonparametric updating results in priors with the least weight on zero, as was expected. Priors all have similar shapes, with the greater part of the nonzero probability on the interval of $[-1, 0]$, and some mass further away from 0 and more spread out on the positive plane. This indicates that, to obtain an accurate image of the shape of the β_1-distributions, a sample size of 10 should be enough. Note that most of the functions actually have stretches on the parameter space between the modes on the negative and positive plane, where the probability is exactly 0.

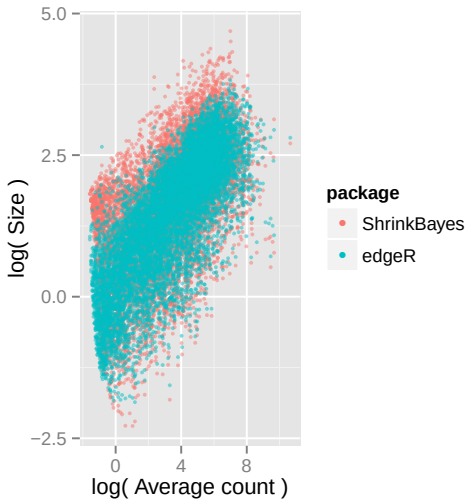

Figure 2: Dispersion coefficients calculated via ShrinkBayes and edgeR plotted versus average feature count, within a negative binomial framework.

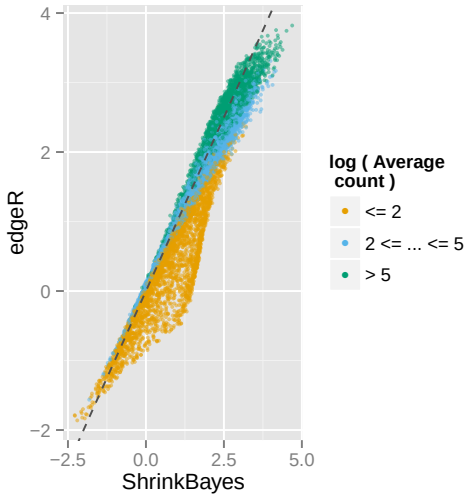

Figure 3: Dispersion coefficients calculated via ShrinkBayes or edgeR. ϕ_i's on the y-axis were transformed to log(size) scale.

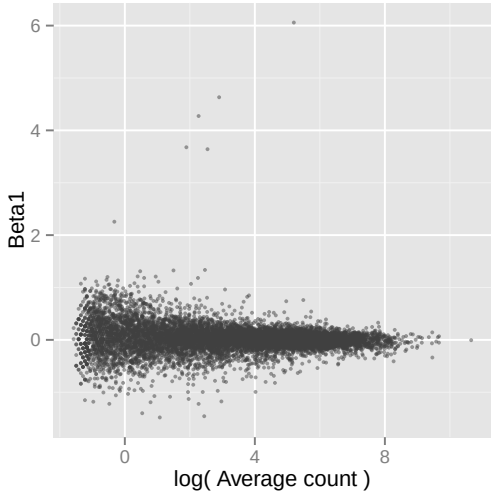

Figure 4: Effect size versus average gene expression. Only coefficients from ShrinkBayes are shown, but edgeR's logFC's produce a similar plot.

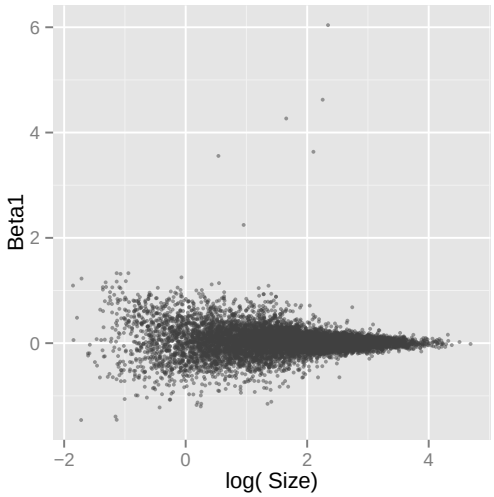

Figure 5: Effect size versus dispersion estimates, obtained via iterative Bayesian updating of latent Gaussian model

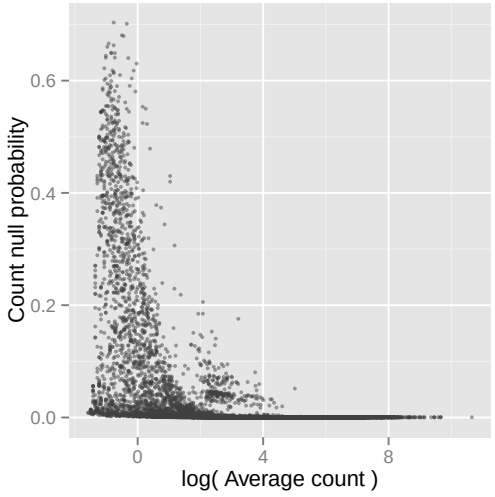

Figure 6: Count null probability q_0 from ZI-NB model plotted against average gene expression

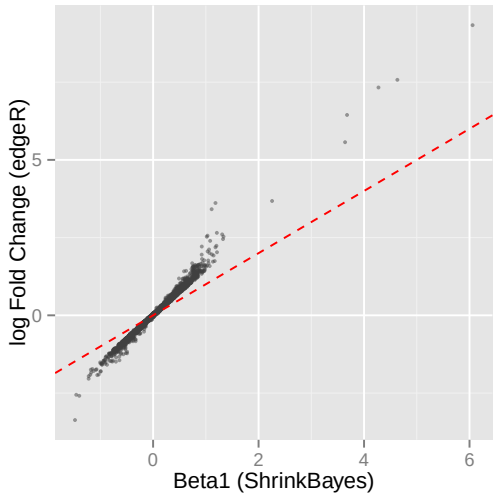

Figure 7: Effect sizes estimated via ShrinkBayes or edgeR.

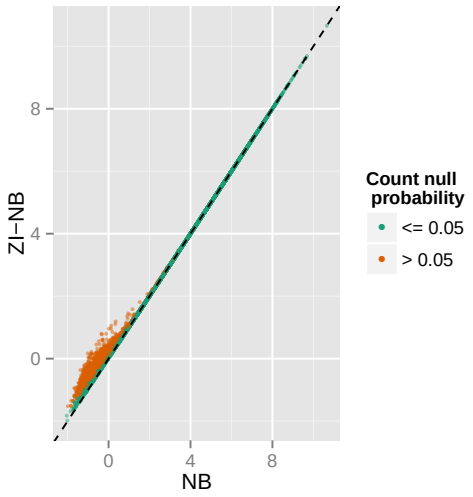

Figure 8: Intercepts obtained when analysing a 20-sized Pickrell sample with or without zero-inflation

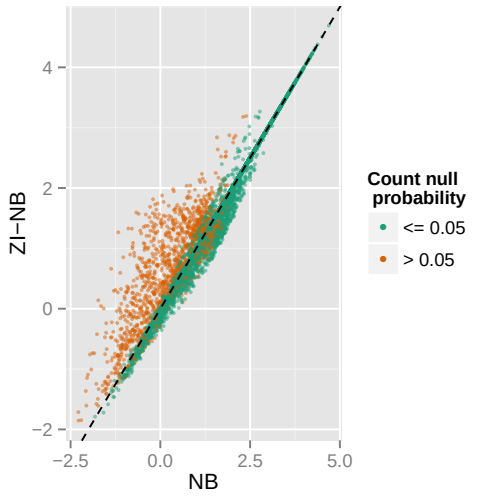

Figure 9: Dispersion values obtained when analysing a 20-sized Pickrell sample with or without zero-inflation

Marginal posteriors can never take values other than 0 on these stretches.

The variation between p_0 estimates on different priors is highest of the three shapes. Estimated effect nonzero probabilities are sometimes twice or more the magnitude of the actual found abundance. The calculation of individual posterior distributions and BFDR's might lessen this discrepancy, but we would prefer to generate counts with effects from the estimated prior distribution. This saves some computation time during estimation of data-generative settings, but more importantly, exporting one instead of p distributions significantly decreases the memory space needed for the generation of β_1-vectors.

Constrained nonparametric

Of constrained nonparametric distributions, the combination of log-concavity and unimodality (over both the negative and positive plain) was the one resulting a) in a shape noticeably different from the unconstrained nonparametric estimates, and b) in most accurate p_0 estimates, when compared with the full size estimates. In contrast with this, nonzero parts of the distributions shown in figure 10 (bottom) are not entirely similar over the various pilot samples. It is quite likely that these smooth distributions do not bear close resemblance to the actual effect size distribution, but because of the consistency in p_0-estimation the log-concave unimodal nonparametric distribution is included in further assessment of prediction performance.

Table 2: Effect null probabilities p_0 estimated by fitting various prior shapes to equal sized subsamples of Pickrell XY-features dataset.

Pilot size	ShrinkBayes distribution shape		
	Mixture	Nonparametric	Constrained NP
58	0,962	0,840	0,973
10	0,500	0,922	0,931
10	0,500	0,878	0,932
10	0,981	0,974	0,979
10	0,939	0,771	0,972
20	0,962	0,903	0,968
20	0,962	0,943	0,978
20	0,971	0,979	0,965
20	0,977	0,934	0,972

Table 3: Proportion of zero hypotheses (p_0) in Pickrell XY-dataset, estimated via empirical Bayesian (with constrained nonparametric prior) or `convest()`-method.

Pilot size	Empirical Bayesian	`convest()`
58	0.973	0.835
10	0.931	0.973
10	0.932	0.988
10	0.979	0.990
10	0.972	0.985
20	0.968	0.962
20	0.978	0.965
20	0.965	0.973
20	0.972	0.816

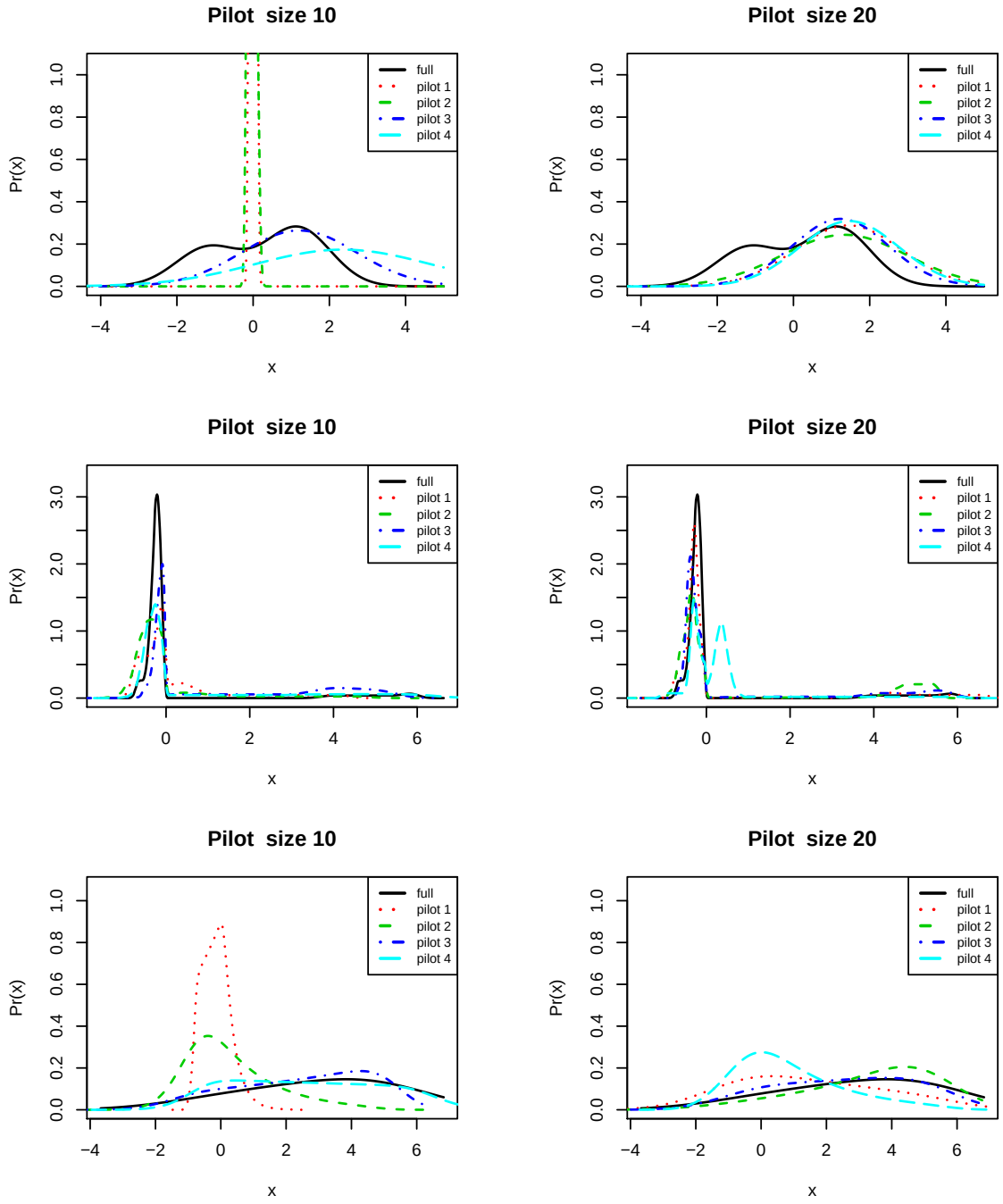

Figure 10: Prior distribution estimates on equal-sized subsamples of Pickrell XY-dataset, fitting a mixture (top), unconstrained nonparametric (middle) or log concave unimodal nonparametric (bottom) distribution. Note the scale differences of the y-axes.

Power and abundance per β_1-prior shape

Let us now consider what kind of detected signal abundance the three distributions yield. We have multiple possible negative binomial coefficient values (Gaussian or updated marginals, priors or posteriors, draws or expected values) to use during count generation, and have not decided what kind of estimates will be used for count generation. For now, expectations of marginal Gaussian posteriors for intercept, dispersion and q_0 values are used. This way, all possible correlations between these parameters are left intact and we have as close a resemblance to the original data as possible within our data-generative framework.

As for effect parameters, we can't use the shrunken β_1-expectations directly, because p_0 is not yet taken into consideration here. In the figures below, β_1 coefficients are sampled from updated prior probabilities. Every iteration of generating data to exact testing, a new vector of effects was drawn. The number of nonzero effects was fixed, based on estimated p_0.

From these coefficients, negative binomial observations, half male, half female, were generated for a grid of sample sizes and analysed by exact testing 50 times. The average detection rate of $|\beta_1| \geq 0,5$ and the average total number of rejections per iteration are plotted in figures 12 up to 14.

For all combinations of sample size $\geq$ 40 and distributional shape, observed false discovery proportions were on average $< 0,1$, except for the predictions with log concave unimodal effects on large simulation datasets. Here, observed average FDR increases to 0,12 and 0,13 for sample sizes 150 and 200, respectively. Generally, though sometimes lacking in power, the exact testing procedure is adequate for controlling the number of false positives.

The larger circles reflect predictions obtained when the full 58-sized dataset is used for the estimation of data-generative settings. For the nonparametric prior, the results on abundance concur with what is actually observed with a direct analysis of the complete dataset. Predicted abundance based on data generated with a mixture prior for differential effect is consistently lower than that found on the complete dataset. The two 10-sized pilot analyses mentioned earlier gave median predicted DE features of respectively 1 and 3 for sample size 60. The standard nonparametric form performs best in terms of abundance prediction: The lower p_0, combined with the fact that not all effects are detected, results in values that resemble observed abundance on the 58-sized dataset best. Still, having an "unlucky" pilot dataset can result in a complete over- (pilot 20.4) or underestimation (pilot 10.3) of the signal present. For log-concave unimodal distributions holds that estimated p_0's are less variable than their more flexible counterparts. Sampling effects from them consistently results in an underestimation of the number of rejections. Per pilot, the estimated feature abundance has lower variance, indicating that less iterations of generation-analysis are needed to do a sample size calculation.

As for power, effects sampled from mixture distributions are very constantly detected over the

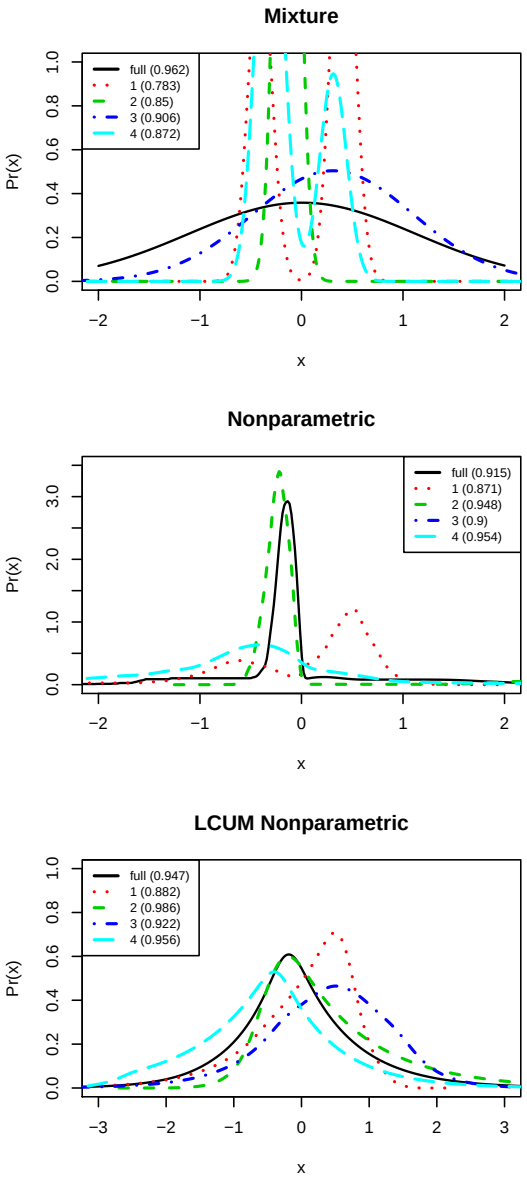

Figure 11: Prior distribution estimates on equal-sized subsamples of Lung Squamous Cell Carcinoma dataset. Only results obtained with pilots of size 5+5 are shown, corresponding p_0-values are given between brackets in the legend.

various iterations. For pilots 10.1 and 10.2, there is simply no $|\beta_1| \geq 0,5$, therefore the detection rate equals 0. Nonparametric effect distributions give more variability in power estimations over the pilot analyses. The probability of detecting $|\beta_1| \geq 0,5$ stabilizes quickest for the constrained effect distributions, as sample size is increased. The figures on the right suggest that in order to obtain representative estimates of at least power (effect abundance is more troublesome), a pilot sample of at least 20 subjects is needed.

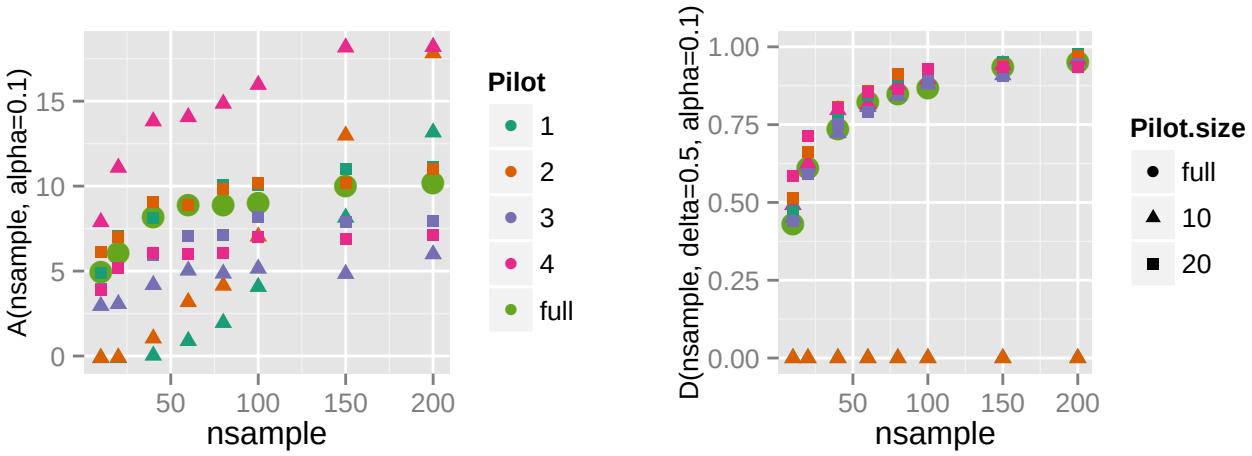

Figure 12: Predicted effect abundance $A(n_{sample};\ \alpha = 0, 1)$ and detection rate $D(n_{sample};\ \delta = 0.5;\ \alpha = 0.1)$ for Pickrell XY dataset. Predictions were based on information from 10- or 20-sized pilot datasets, effects were sampled from updated mixture distributions. In the left figure, points were jittered in vertical direction, $\pm 0, 2$

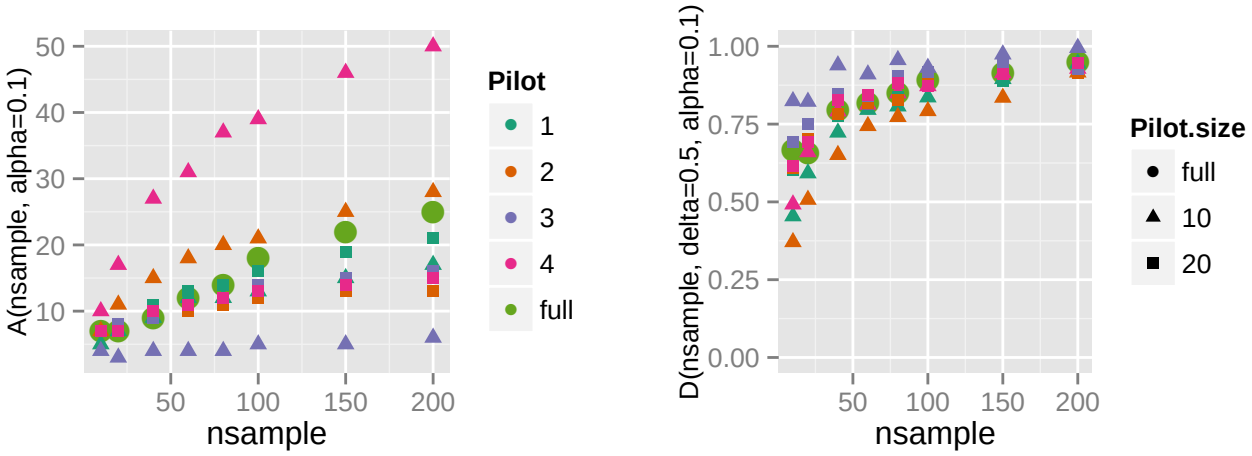

Figure 13: Predicted effect abundance $A(n_{sample};\ \alpha = 0, 1)$ and detection rate $D(n_{sample};\ \delta = 0.5;\ \alpha = 0.1)$ with effects sampled from an updated nonparametric distribution

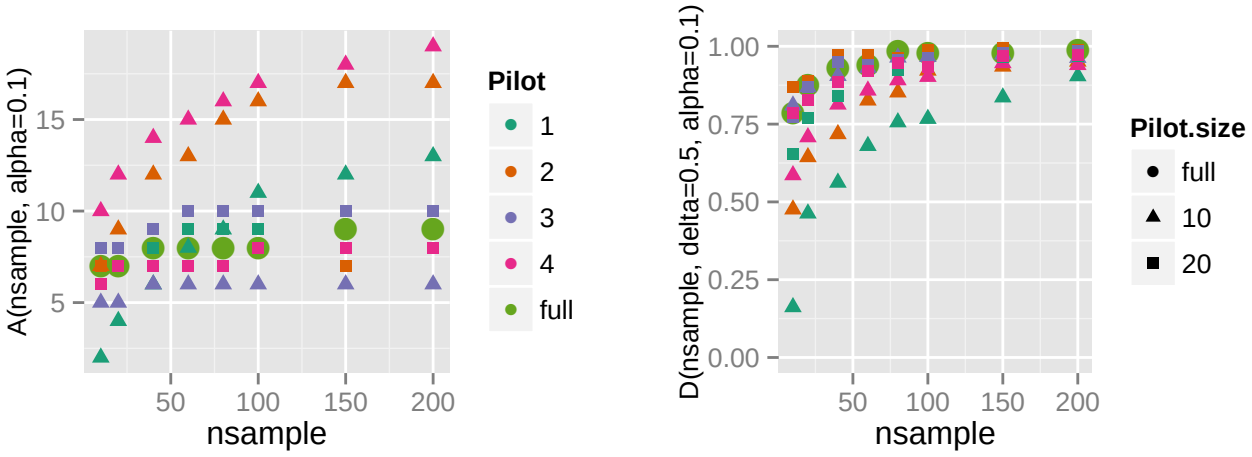

Figure 14: Predicted effect abundance $A(n_{sample};\ \alpha = 0, 1)$ and detection rate $D(n_{sample};\ \delta = 0.5;\ \alpha = 0.1)$ with effects sampled from an updated log-concave unimodal nonparametric distribution

3.4 Data-generative settings: sets of coefficients

Based on tryouts of various settings, we have concluded that data-generation with marginal posterior expectations for intercept, dispersion and count zero probability parameters provides best resemblance to the full-sized results. The effect of disrupting data structure was tested by sampling from Gaussian priors, and by random resampling without replacement ("shuffling") of the marginal expectation vectors. This always results in a dramatic decrease in both power and abundance.

The shapes of updated posterior distributions $\tilde{\pi}(\beta_{1i} \mid Y_{ij})$ turned out to be quite sensitive to pilot sampling. We compared large samples from sets of $\tilde{\pi}(\beta_{1i} \mid Y_{ij})$ of different pilot samples to assess whether this variety in posterior distribution shape averages out over the set of features, but this was not the case. Updated prior shapes $F(\beta_1)$ were more stable (see fig. 11). More importantly, β_1's drawn from the updated prior, thus uncorrelated to β_0, ϕ and q_0, resulted in higher observed abundances, comparable to those observed on full-sized datasets.

3.5 MiRNAseq

The same strategy of drawing "pilot" samples from the complete dataset used earlier was employed to the MiRNAseq dataset, available from the VUMC. A log-concave unimodal nonparametric prior shape was fitted to differential effect parameter. Not all pilot sample analyses resulted in useful parameter estimates. Several pilots had an updated prior β_1-distribution with a nonzero part in the range of values of ≥ 50 (concurrent estimated β_0's in the range of $[-3, 6]$). Observations of this magnitude are physically very unlikely, and sampling and analysing negative binomial draws results in numerical errors, due to the magnitude of generated response observations. For those pilots resulting in useable β_1-values, predicted effect abundance and power are shown in figure 15.

Here, A and D tell a different story. Simulation-based estimates of total abundance are more often than not very far from the truth, and of little use. This variation is mainly caused by differences in estimated p_0. Because D reflects average power, this variation is largely mitigated and predictions are more stable. This is reflected in the shape of estimated $F_{\beta_1, nonzero}()$'s, found in section 3.3 in figure 11C: They are comparatively similar, only effect null probabilities vary. By increasing δ a researcher is able to further stabilize inference over the various pilot datasets, at the cost of a decrease in the proportion of effects this inference applies to.

Shown are observed power values for $|\beta_1| \geq 0,5$. For stable predictions, a pilot dataset of 10+10 is needed, and even then, one of the four pilot experiments results in a noticeable underestimation of power at sample size 200.

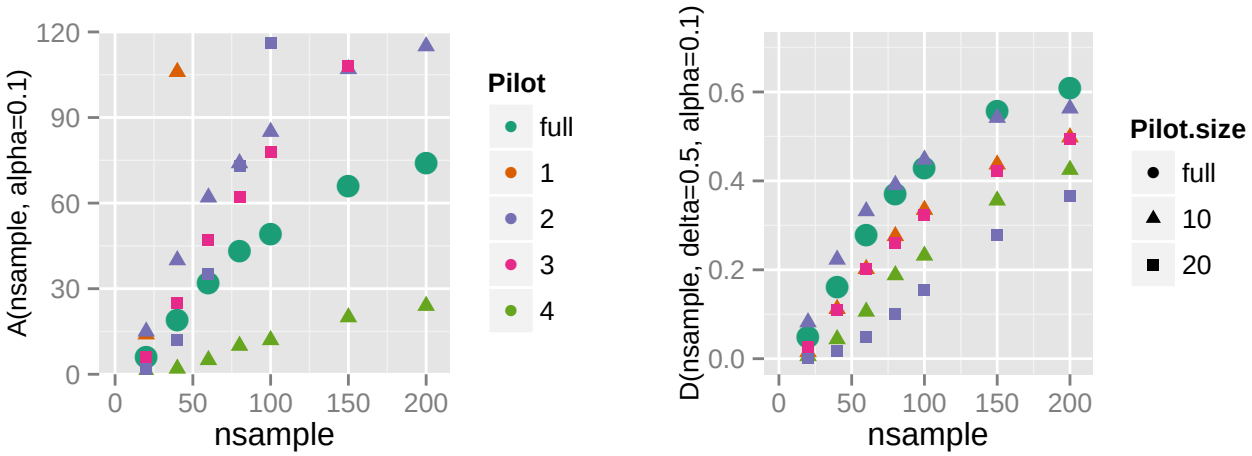

Figure 15: Predicted effect abundance $A(n_{sample};\ \alpha = 0.1)$ and detection rate $D(n_{sample};\ \delta = 0.5;\ \alpha = 0.1)$ for MiRNAseq dataset. Predictions were based on information from 10- or 20-sized pilot datasets, effects were sampled from updated log concave unimodal nonparametric distributions.

3.6 Lung Squamous Cell Carcinoma

Figure 16 shows predicted average power and abundance. Again, predictions on abundance are not really useful. This is due to the variability in estimated prior p_0 values, also shown in table 4 below. Results on power are more stable, like with the MiRNAseq data. The remarks made on this in Section 3.5 on also apply here.

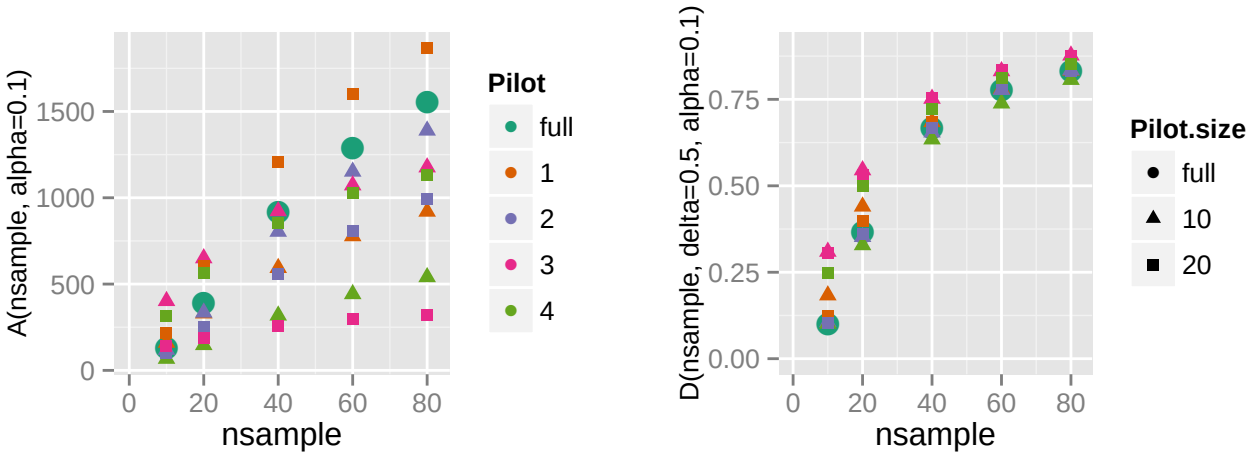

Figure 16: Predicted effect abundance $A(n_{sample};\ \alpha = 0, 1)$ and detection rate $D(n_{sample};\ \delta = 0, 5;\ \alpha = 0, 1)$

Table 4: Effect null probabilities p_0 estimated by fitting log-concave unimodal distributions to two groups of patients suffering from lung squamous cell carcinoma, either with or without recurrence. Equal sized subsamples of size 10+10 and 20+20 were taken by random independent sampling without replacement from the two groups. Below, estimates of the proportion of true null hypotheses, based on a convex decreasing density estimate [Langaas and Lindqvist, 2005] of the observed p-values returned by exact testing.

	Empirical Bayesian p_0		
Sample size	20	40	80
1	0,813	0,926	0,906
2	0,895	0,796	
3	0,833	0,854	
4	0,908	0,976	
	`convest`-method π_0		
Sample size	20	40	80
1	0,899	0,933	0,823
2	0,926	0,776	
3	0,919	0,860	
4	0,905	0,932	

4 Implementation

Functions for simulation-based RNAseq power calculation via empirical Bayesian shrinkage methods were created. Because of the variety of possible parameter settings, creation of the data-generative model with functions from the `ShrinkBayes` R Package was not fully automated. The (zero-inflated) negative binomial parameter estimates thus obtained can be passed into our `calcPower()`-function. Via simulation and analysis of counts based on these parameters, this will render a prediction of both the average power $D(n_{sample}, \delta, \alpha)$ for detection of effects with predefined boundaries ($|\beta_1| \geq \delta$), and the total number of hypothesis rejections $A(n_{sample}, \alpha)$, both at confidence level α. For increased user-convenience, a wrapper was created in `powerCurve()`, that directly accepts `ShrinkBayes` output objects and a vector of sample sizes as input, and iteratively applies `calcPower()` for each of these values, returning average abundance and true positives for each of the input sample size values.

Based on extensive tryouts with various parameter settings, we recommend using a log concave unimodal nonparametric prior distribution to approximate the effect size distribution, and sampling effect sizes from this prior $F_{NP}(\beta_1)$. During generation of counts, the dependency structure between intercept, dispersion and count zero probability should be left intact, which is why we recommend using estimated values of the (Gaussian) marginal posteriors. They can readily be extracted from `ShrinkBayes`-objects, using functions created during this investigation. This renders most stable p_0 estimates and most accurate abundance, compared to "real" analysis results. We advise never to use less than 10 subjects for stable power assessment of RNAseq data.

5 Discussion

The sample size calculation functions are especially useful when the researcher is confident about the homogeneity in his dataset, and library sizes are comparable. The dataset collected by Pickrell [Pickrell et al., 2010] is such a set. When data homogeneity is questionable, for example due to its being collected at more than one place via varying protocols, the observed variability between subsets of size < 20 was such, that it was not possible to reliably predict signal abundance.

While we state that we are not able to generate stable *predictions* of a certain outcome, the actual observed outcomes are more variable still. Analysis on 10-20 sized (which is considered intermediate in the field of RNAseq analysis) subsets of large datasets on Lung Squamous Cell Carcinoma, obtained from The Cancer Genome Atlas resulted in observed differences in signal abundance of a magnitude of more than ten. Standard normalization techniques were not enough to nullify the effect of varying sequencing depth per subject. The empirical Bayesian techniques employed via ShrinkBayes do stabilise data-generative settings, but in this case simply not enough.

This property of current RNAseq technologies and analysis methods is something to be aware of. In current biological research it is quite common to do RNAseq experiments with $n \leq 10$. Researchers should be aware that the correlation between per-feature average expression and dispersion, the total proportion of differentially expressed features and the rejection criterion in high-dimensional count-based models are complicating factors in estimation of parameters. This is enhanced by the fact that the high-dimensionality of the data has led to forms of "stabilize-over-the-set" methods of parameter estimation in all widespread analysis methods. This creates dependency between parameter estimates and increases the effect a column of outliers or sub-optimal measurements has on the model. With current available inferential methods, RNAseq experiments with ≤ 10 subjects will often not yield a representative image of differential expression. Experience during this investigation was, that the difficulties lie in the area of sensitivity: the exact testing procedure is adequate for controlling FDR, i. e. not rejecting "true" null hypotheses, but is likely not to reject all (most) false null hypotheses.

Finding an objective criterium to assess whether predicted power equals true detection probability is not entirely straightforward in a non-simulation setting. The absolute truth on differential expression is not available for the set of features, hence we relied on exact testing results based on large datasets and treated this as the truth. Comparable investigations use other definitions of differentially expressed such as "rejected by all test procedures" [Ching et al., 2015], only compare relative power by altering data-generative settings [Wu et al., 2014] or even lack any assessment of the performance of their algorithm [Shyr and Li, 2014].

No surprising trends in dispersion estimation were found when comparing results from the it-

erative Bayesian and the exact testing approach for *intermediate* sample sizes. Referring back to figures 2 and 3, the observed differences can for a large part be traced back to the fact that estimation was done on an inverse scale. If obtaining unbiased power and abundance predictions based on too small pilot samples is problematic regardless of the estimation procedure, and `edgeR`'s tagwise dispersion estimation gives comparable dispersion estimates to `ShrinkBayes`' for intermediate sample sizes, the gain in computational speed might favour estimation of the data-generative model via this method, such as provided in [Wu et al., 2014].

Obtaining useable real RNAseq data was more problematic than anticipated. After initial calibration on the Pickrell dataset, the two datasets used for further assessment turned out to be quite a lot more heterogeneous than the first. An average pilot experiment will be carried out in a small time period by a single research group (or even researcher), leading to more carefully controlled circumstances. This should always result in noticeably smaller technical variation than between TCGA subjects collected from different institutes. Deleterious growth in lung cancer cells can be caused by multiple cell signalling pathways, which makes the biological variation between subjects larger than average and difficult to capture. The MiRNAseq dataset used was carried out under controlled circumstances, but the nature of the experimental setup was not ideally suited to test a two-group comparison setting.

The original goal of this project was, to create software that would calculate sample size for a predefined experimental goal. The functions provided do not yet do this: needed sample size has to be inferred manually, based on estimated abundance and power. In the author's opinion, for follow-up research the most logical thing to do is more extensively investigate and visualize the heterogeneity that was observed within several available large RNAseq datasets. It may then be possible to determine when small-sample RNAseq differential expression analyses yield reproducible results and when they don't. For this, a large-scale database such as the Cancer Genome Atlas would provide an excellent starting point. For now, it is likely that to investigate the behaviour of tens of thousands of genes, a straightforward investment in collecting more than a handful of replicates is worthwhile.

References

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate - a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society - Series B*, 1995.

T. Ching, S.J. Huang, and L.X. Garmire. Power analysis and sample size estimation for RNA-Seq differential expression. *RNA*, 2015.

L. Dümbgen and K. Rufibach. logcondens: Computations related to univariate log-concave density estimation. *Journal of Statistical Software*, 2011.

J. A. Ferreira and A. H. Zwinderman. On the Benjamini-Hochberg method. *Annals of Statistics*, 2006.

R.A. Fisher. On the interpretation of χ^2 from contingency table, and the calculation of P. *Journal of the Royal Statistical Society*, 1992.

Alyssa C. Frazee, Ben Landmead, and Jeffrey T. Leek. ReCount: A multi-experiment resource of analysis-ready RNA-seq gene count datasets. *BMC Bioinformatics*, 2011.

Yan Guo, Quanhu Sheng, Jiang Li, Fei Ye, David C. Samuels, and Yu Shyr. Large scale comparison of gene expression levels by microarrays and RNAseq using TCGA data. *PLoS ONE*, 2013.

S.N. Hart, T.M. Therneau, Y. Zhang, G.A. Poland, and J-P. Kocher. Calculating sample size estimates for RNA sequencing data. *Journal of Computational Biology*, 2013.

Mette Langaas and Bo Henry Lindqvist. Estimating the proportion of true null hypotheses, with application to DNA microarray data. *Journal of the Royal Statistical Society, Series B*, 2005.

B Langmead, C Trapnell, M Pop, and S.L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 2009.

Cl Li, PF Su, and Y Shyr. Sample size calculation based on exact test for assessing differential expression analysis in RNA-seq data. *BMC Bioinformatics*, 2013.

John H. Malone and Brian Oliver. Microarrays, deep sequencing and the true measure of the transcriptome. *BMC Biology*, 2011.

Ali Mortazavi, Brian A. Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transciptomes by RNA-seq. *Nature Methods*, 2008.

Alicia Oshlack, Mark D. Robinson, and Matthew D. Young. From RNA-seq reads to differential expression results. *Genome Biology*, 2010.

Shirley Pepke, Barbara Wold, and Ali Mortazavi. Computation for ChIP-seq and RNA-seq studies. *Nature Methods*, 2009.

Joseph K. Pickrell, John Marioni, Athma Pai, Jacob Degner, Barbara Engelhardt, Everlyne Nkadori, Jean-Baptiste Veyrieras, Matthew Stephens, Yoav Gilad, and Jonathan Pritchard. Understanding mechanisms underlying human gene expression variation with RNA sequencing. *Nature*, 2010.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL `http://www.R-project.org/`.

Mark D. Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 2010.

Mark D. Robinson and Godon K. Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 2007.

Mark D. Robinson and Godon K. Smyth. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 2008.

Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 2010.

Havard Rue, Sara Martino, and Nicholas Chopin. Approximate Bayesian inference for latent Gaussian models using Integrated Nested Laplace Approximation (with discussion). *Journal of the Royal Statistical Society, Series B*, 2009.

Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA Microarray. *Science*, 1995.

Derek Shyr and Chung-I Li. Sample size calculation of RNA-sequencing experiment - a simulation-based approach of TCGA data. *Journal of Biometrics & Biostatistics*, 2014.

Gordon K Smyth. Limma: linear models for microarray data. In *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005.

Cole Trapnell, Lior Pachter, and Steven Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 2009.

Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 2010.

John W Tukey. Some thoughts on clinical trials, especially problems of multiplicity. *Science*, 1977.

Mark Van de Wiel, Gwenael G. R. Leday, Lubo Pardo, Havard Rue, Aad W. van der Vaart, and Wessel N. van Wieringen. Bayesian analysis of RNA sequencing data by estimating multiple shrinkage priors. *Biostatistics*, 2012.

Mark Van de Wiel, Maarten Neerincx, Tineke E. Buffart, Daoud Sie, and Henk M.W. Verheul. ShrinkBayes: a versatile R-package for analysis of count-based sequencing data in complex study designs. *BMC Bioinformatics*, 2014.

Maarten Van Iterson, Mark Van de Wiel, Judith M. Boer, and Renée X. De Menezes. General power and sample size calculations for high-dimensional genomic data. *Statistical applications in Genetics and Molecular Biology*, 2013.

W.N. Venables and B.D.Ripley. *Modern Applied Statistics with S*. Springer, 4th edition, 2002.

Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-seq: A revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 2009.

Hao Wu, Chi Wang, and Zhijin Wu. PROPER: Comprehensive power evaluation for differential expression using RNA-seq. *Bioinformatics*, 2014.

Appendices

A `predictPower()`- and `powerCurve()`-code

RNAseq_functions.R

cas

Tue Dec 23 21:04:58 2014

```
#########################################################
### Cas Retel
### Functions used during RNAseq research
### VUMC, 02-2014 until 12-2014
#########################################################

#############################
### Main predictive functions, as discussed in thesis
predictPower <- function(nsample, b0, b1, size, q0, delta=0, alpha=0.1,
                         fixedseed=FALSE){
  # Computes power and effect abundance for the detection of differentially
  # expressed features in a simulated dataset. Requires
  # (possibly zero-inflated) negative binomial coefficients as input,
  # generates a dataset and performs exact tests of differential expression
  # based on (Robinson, 2007). Assumes an equalsized two-group comparison.
  if(0 %in% sapply(list(nsample, b0, b1, size, q0, delta, alpha),
                   function(x) length(x))){
    stop("Not all input arguments are specified")
  }
  if(any(is.na(c(nsample, b0, b1, size, q0)))) stop("Input contains NA values")
  if(!all.equal(length(b0), length(b1), length(size)) |
       (!is.null(q0) & length(b0)!=length(q0))){
    stop("Coefficient vectors are not of the same length")
  }
  require(edgeR)
  p <- length(b0)

  # round to nearest smaller even number:
  while(nsample%%2) nsample <- ceiling(nsample - 1)
  if(is.null(q0)) q0 <- numeric(p)  # when likelihood is nb

  ### Generate count matrix:
  if(fixedseed){
    if(!exists(".Random.seed", mode="integer", envir=globalenv())) sample(NA)
    oldSeed <- .Random.seed
    set.seed(2909)
  }
  sd <- matrix(0, p, nsample)  # matrix of counts
  sd[, 1:(nsample/2)] <- t(sapply(1:p, function(i)
    rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i]))))
  sd[, ((nsample/2)+1):nsample] <- t(sapply(1:p, function(i)
    rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i]+b1[i]))))
  for(i in 1:p) sd[i, sample(1:nsample, size=round(nsample*q0[i]))] <- 0
  set.seed(oldSeed)

  ### Exact testing:
  res <- DGEList(sd, norm.factors=calcNormFactors(sd, method="TMM"),
```

```
                group=factor(rep(1:2, each=ncol(sd)/2)))
  res <- estimateCommonDisp(res)
  res <- estimateTagwiseDisp(res)

  res <- exactTest(res)
  res <- topTags(res, n=nrow(sd), sort.by="none")$table

  ### Output
  out <- data.frame(ActualPos=sum(b1!=0), Rejections=sum(res$FDR <= alpha),
                    TruePos=sum(which(b1!=0) %in% which(res$FDR <= alpha)))
  #   out <- data.frame(DE.feats=sum(b1!=0), detected.feats=sum(res$FDR <= alpha),
  #                     matching.feats=sum(is.element(which(res$FDR <= alpha),
  #                                                   which(b1!=0))))
  if(delta!=0){
    out$ActualPos.delta=sum(abs(b1)>=delta)
    out$TruePos.delta=
      sum(which(res$FDR <= alpha) %in% which(abs(b1)>=delta))
  }
  rm(sd, res, p)
  return(out)
}


powerCurve <- function(fasobj, muprobj, nsample=c(10, 20, 40, 60, 80, 100),
                       delta=0, alpha=0.1, niter=50, ncpus=2){
  # Computes power and effect abundance for the detection of
  # differentially expressed features in a simulated dataset of
  # negative binomial counts. Requires (possibly zero-inflated)
  # negative binomial coefficients as input, generates a dataset and
  # performs exact tests of differential expression based on (Robinson, 2007).
  # Assumes an equalsized two-group comparison.
  require(edgeR); require(INLA); require(snowfall)
  b1s <- function(p, obj, fixednumbernon0=TRUE){
    # draw p samples from updated prior distribution
    # fixednumbernon0 = should the number of nonzero effects be fixed
    #                   or determined stochastically?

    if(!any(is.element(c("p0est", "best"), names(obj)))){
      stop("input is not an UpdatePrior-object")
    }
    effect <- numeric(p)
    p0 <- effectindex <- numeric()

    p0 <- ifelse(is.element("p0est", names(obj)), obj$p0est, obj$best["p0"])

    if(fixednumbernon0){
      effectindex <- sample(1:p, size=round((1-p0)*p))
    }else{
      effectindex <- runif(p) > p0
    }

    # for NonPara-objects:
    if("p0est" %in% names(obj)){    # for NonParaUpdate-objects:
```

```
    effect[effectindex] <- inla.rmarginal(n=sum(effectindex != 0),
                                          obj$priornew)
  }else{    # for MixtureUpdate-objects:
    effect[effectindex] <- rnorm(sum(effectindex!=0), mean=obj$best["mu"],
                                 sd=obj$best["stdev"])
    effect[effect!=0] <- effect[effect!=0] +
      (-2 * (runif(sum(effectindex!=0)) > obj$best["pmin"]) *
         effect[effect!=0])
    # because x + (-2*x) = -x
  }
  return(effect)
}

predictPower <- function(nsample, b0, b1, size, q0, delta=0, alpha=0.1,
                         fixedseed=fixedseed){

  if(0 %in% unlist(lapply(list(nsample, b0, b1, size, q0, delta, alpha),
                          function(x) length(x)))){
    stop("Not all input arguments are specified")
  }
  if(any(is.na(c(nsample, b0, b1, size, q0)))) stop("Input contains NA values")
  if(!all.equal(length(b0), length(b1), length(size)) |
       (!is.null(q0) & length(b0)!=length(q0))){
    stop("Coefficient vectors are not of the same length")
  }
  require(edgeR)
  p <- length(b0)

  # round to nearest smaller even number:
  while(nsample%%2) nsample <- ceiling(nsample - 1)
  if(is.null(q0)) q0 <- numeric(p)  # when likelihood is nb

  ### Generate count matrix:
  if(fixedseed){
    if(!exists(".Random.seed", mode="integer", envir=globalenv())) sample(NA)
    oldSeed <- .Random.seed
    set.seed(2909)
  }
  sd <- matrix(0, p, nsample)  # matrix of counts
  sd[, 1:(nsample/2)] <- t(sapply(1:p, function(i)
    rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i]))))
  sd[, ((nsample/2)+1):nsample] <- t(sapply(1:p, function(i)
    rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i]+b1[i]))))
  for(i in 1:p) sd[i, sample(1:nsample, size=round(nsample*q0[i]))] <- 0
  set.seed(oldSeed)

  ### Exact testing:
  res <- DGEList(sd, norm.factors=calcNormFactors(sd, method="TMM"),
                 group=factor(rep(1:2, each=ncol(sd)/2)))
  res <- estimateCommonDisp(res)
  res <- estimateTagwiseDisp(res)

  res <- exactTest(res)
```

```
    res <- topTags(res, n=nrow(sd), sort.by="none")$table

    ### Output
    out <- data.frame(ActualPos=sum(b1!=0), Rejections=sum(res$FDR <= alpha),
                      TruePos=sum(which(b1!=0) %in% which(res$FDR <= alpha)))
    #   out <- data.frame(DE.feats=sum(b1!=0), detected.feats=sum(res$FDR <= alpha),
    #                     matching.feats=sum(is.element(which(res$FDR <= alpha),
    #                                                   which(b1!=0))))
    if(delta!=0){
      out$ActualPos.delta=sum(abs(b1)>=delta)
      out$Truepos.delta=
        sum(which(res$FDR <= alpha) %in% which(abs(b1)>=delta))
    }
    rm(sd, res, p)
    return(out)
  }

  p <- length(fasobj$res)
  b0 <- extB0s(fasobj)
  size <- extSizes(fasobj)
  q0 <- try(extCountp0s(fasobj))

  sfInit(cpus=ncpus, parallel=TRUE)
  sfLibrary(edgeR); sfLibrary(INLA)

  sfExport("predictPower", "b1s", "p","nsample", "niter",
           "alpha", "b0", "size", "q0", "muprobj", "delta")

  abundance <- lapply(seq_along(nsample), function(i) matrix(NA, niter, 5))
  abundance <- lapply(nsample, function(obj)
    t(sfSapply(1:niter, function(i)
      predictPower(obj, b0, b1s(p, muprobj), size, q0, delta=delta, alpha=alpha))))

  sfStop()

  out <- t(sapply(abundance, function(x)
    apply(x, 2, function(y) mean(as.numeric(y)))))
  out <- round(out)
  out <- cbind(nsample, out)

  colnames(out) <- c("nsample","ActualPos", "Rejections", "TruePos",
                     "ActualPos.delta", "TruePos.delta")[1:ncol(out)]

  rm(abundance, p, b0, size, q0)
  if(b1thresh!=0) return(out) else return(out[, 1:4])
}
```

B Other R code

```
### ShrinkBayes convenience functions
extB0s <- function(...){
  # ExtB0s() returns estimates beta0 (intercept) coefficients
  # from input FitAllShrink-objects
  # Returns na when when a $res-list is NULL
  fasall <- list(...)
  if(any(!sapply(fasall, function(obj)
    ("res" %in% names(obj) | "priors" %in% names(obj))))){
    stop("Input are not all FitAllShrink-objects")
  }
#   if(any(!sapply(seq_along(fasall), function(i)
#     ("res" %in% names(fasall[[i]]) | "priors" %in% names(fasall[[i]])))){
#     stop("Input are not all FitAllShrink-objects")
#   }
  as.numeric(sapply(fasall, function(obj)
    sapply(1:length(obj$res), function(i) as.numeric(
      obj$res[[i]]$summary.fixed[1, 1]))))
}

extB1s <- function(...){
  # ExtB1s() returns estimated beta1 (group effect) coefficients
  # from FitAllShrink-objects
  # Returns na when when a $res-list is NULL
  fasall <- list(...)
  if(any(!sapply(fasall, function(obj)
    ("res" %in% names(obj) | "priors" %in% names(obj))))){
    stop("Input are not all FitAllShrink-objects")
  }
  as.numeric(sapply(fasall, function(obj)
    sapply(1:length(obj$res), function(i) as.numeric(
      obj$res[[i]]$summary.fixed[2, 1]))))
}
extSizes <- function(..., logscale=T){
  # ExtSizes() returns estimated log(size) coefficients
  # from FitAllShrink-objects.
  # set logscale = F to return the exponent of this estimate
  # Returns na when when a $res-list is NULL
  fasall <- list(...)
  if(any(!sapply(fasall, function(obj)
    ("res" %in% names(obj) | "priors" %in% names(obj))))){
    stop("Input are not all FitAllShrink-objects")
  }
  out <- as.numeric(sapply(fasall, function(obj)
    sapply(1:length(obj$res), function(i) as.numeric(
      obj$res[[i]]$internal.summary.hyper[1, 1]))))
  if(!logscale) out <- exp(out)
  return(out)
}
extCountp0s <- function(...){
  # extCountp0s() returns count null probabilities (q0)
  # from FitAllShrink-objects
  # Returns na when when a $res-list is NULL
  fasall <- list(...)
```

```
  if(any(!sapply(fasall, function(obj)
    ("res" %in% names(obj) | "priors" %in% names(obj))))){
    stop("Input are not all FitAllShrink-objects")
  }
  as.numeric(sapply(fasall, function(obj)
    sapply(1:length(obj$res), function(i) as.numeric(
      obj$res[[i]]$summary.hyperpar[2, 1]))))
}
extPriorEffp0s <- function(..., digits=NA){
  # Extracts effect null probabilities from UpdatePrior-objects
  # (either Mixture or NonPara). Returns a numeric vector.
  # digits is an optional parameter, specifying number of digits to round to
  objall <- list(...)

  if(any(!sapply(objall, function(obj)
    ("p0est" %in% names(obj) | "best" %in% names(obj))))){
      stop("input is not an UpdatePrior-object")
  }

  out <- numeric(length(objall))
  out <- sapply(seq_along(objall), function(i)
    if(any(names(objall[[i]])=="best")){  # Mixture prior
      out <- objall[[i]]$best["p0"]
    } else out <- objall[[i]]$p0est)   # Nonparametric prior
  if(!is.na(digits)) out <- round(out, digits=digits)
  as.numeric(out)
}

extEffp0s <- function(obj){
  # Extracts effect null probabilities from an UpdatePosterior-object
  # (either Mixture or NonPara). Returns a numeric vector.
  if(!identical(names(obj[[1]]), c("postbetanon0", "postbeta0", "loglik"))){
    stop("input is not an UpdatePosterior-object")
  }

  as.numeric(sapply(obj, function(x) x$postbeta0))
}

sampleB1s <- function(obj, expectation=FALSE, numbernon0=NA){
  # sampleB1s() returns a draw from posterior distributions in the
  # input UpdatePosterior-object (either Mixture or NonPara).
  # expectation = Boolean. Return expected value instead of draw?
  # numbernon0 = Numeric. When specified, those numbernon0 distributions
  #              with smallest posterior p0 are used to draw from
  if(is.na(numbernon0)){   # zero or draw determined stochastically
    effectind <- (runif(length(obj)) >
                    sapply(obj, function(x) x$postbeta0))
  }else{   # take smallest p0's
    effectind <- is.element(
      seq_along(obj), order(unlist(
        lapply(obj, function(x) x$postbeta0)))[1:numbernon0])
  }
```

```
  effect <- numeric(length(obj))
  if(!expectation){   # random draw
    effect[effectind] <- sapply(which(effectind), function(i)
      inla.rmarginal(1, obj[[i]]$postbetanon0[[1]]))
  }else{   # expected value
    effect[effectind] <- sapply(which(effectind), function(i)
      inla.emarginal(function(x) x, obj[[i]]$postbetanon0[[1]]))
  }
  return(effect)
}

expB1s <- function(obj, ignorep0=FALSE, stochasticp0=FALSE){
  # expB1s() returns expected values of (posterior distribution times the
  # corresponding null probability), for every feature, from an
  # UpdatePosterior-object.
  # Set ignorep0 = TRUE to return expectation of nonzero part only
  # Set stochasticp0 = TRUE to compare effect p0 to a random draw
  # (with fixed seed) from U(0, 1) distr., and use this to determine
  # whether 0 or E(nonzero dist) is returned.
  if(!is.list(obj) |
       !identical(names(obj[[1]]), c("postbetanon0", "postbeta0", "loglik"))){
    stop("input is not an UpdatePosterior-object")
  }
  p <- length(obj)
  effect <- numeric(p)

  # return 0 for posteriors with zero probability of exactly 1:
  # for these posteriors, marginal INLA functions will result in errors
  nonzeroind <- which(sapply(obj, function(x) (x$postbeta0!=1)))

  if(ignorep0){  # return E(nonzero part)
    effect[nonzeroind] <- sapply(nonzeroind, function(i)
      inla.emarginal(function(x) x, obj[[i]]$postbetanon0[[1]]))
  }else{
    if(!stochasticp0){  # effectp0*E(nonzero part)
      effect[nonzeroind] <- sapply(nonzeroind, function(i)
        (1-obj[[i]]$postbeta0) *
          inla.emarginal(function(x) x, obj[[i]]$postbetanon0[[1]]))
    }else{  # return (pseudo)random sample from E(nonzero)'s
      #if(!exists(".Random.seed", mode="integer", envir=globalenv())) sample(NA)
      #oldSeed <- .Random.seed
      #set.seed(1605)

      nonzeroind <- runif(p) > sapply(1:p, function(i) obj[[i]]$postbeta0)
      effect[nonzeroind] <- sapply(which(nonzeroind), function(i)
        inla.emarginal(function(x) x, obj[[i]]$postbetanon0[[1]]))

      #assign(".Random.seed", oldSeed, envir=globalenv())
    }
  }
  return(effect)
}
```

```
samplePriorSizes <- function(p, obj, logscale=T){
  # draws n samples from prior size distribution, as stored in ShrinkSeq-object
  if(logscale){
    return(rnorm(p, mean=obj$pmlist$mudisp, sd=1/sqrt(obj$pmlist$precdisp)))
  }else{
    return(exp(rnorm(p, mean=obj$pmlist$mudisp,
                     sd=1/sqrt(obj$pmlist$precdisp))))
  }
}

samplePriorB1s <- function(p, obj, fixednumbernon0=TRUE){
  # draw p samples from updated prior distribution
  # fixednumbernon0 = should the number of nonzero effects be fixed
  #                   or determined stochastically?

  if(!any(is.element(c("p0est", "best"), names(obj)))){
    stop("input is not an UpdatePrior-object")
  }
  effect <- numeric(p)
  p0 <- effectindex <- numeric()

  p0 <- ifelse(is.element("p0est", names(obj)), obj$p0est, obj$best["p0"])

  if(fixednumbernon0){
    effectindex <- sample(1:p, size=round((1-p0)*p))
  }else{
    effectindex <- runif(p) > p0
  }

  # for NonPara-objects:
  if("p0est" %in% names(obj)){    # for NonParaUpdate-objects:
    effect[effectindex] <- inla.rmarginal(n=sum(effectindex != 0),
                                          obj$priornew)
  }else{    # for MixtureUpdate-objects:
    effect[effectindex] <- rnorm(sum(effectindex!=0), mean=obj$best["mu"],
                                 sd=obj$best["stdev"])
    effect[effect!=0] <- effect[effect!=0] +
      (-2 * (runif(sum(effectindex!=0)) > obj$best["pmin"]) *
         effect[effect!=0])
    # because x + (-2*x) = -x
  }
  return(effect)
}

##############################
### Plotting of ShrinkBayes output
plotMixturePrior <- function(mixtureupdate, scaletodens=TRUE, plotp0=TRUE,
                             colour=1, ltype=1, lwdth=1, xlim=c(-5, 5),
                             ylim=NA, main=NULL){
  # scaletodens = logical indicating whether the nonzero part of the
  # distribution (that integrates to 1) should be scaled, so that
  # [ p0*diracdelta(0) + (nonzero part) ] integrates to 1
  # plotp0 = logical indicating whether the point mass on zero is plotted
```

```
  p0 <- pneg <- ppos <- mu <- sd <- numeric(1)
  x <- y <- numeric(1000)

  p0 <- mixtureupdate$best["p0"]
  pneg <- mixtureupdate$best["pmin"]
  ppos <- (1-mixtureupdate$best["pmin"])
  if(scaletodens){
    pneg <- (1-p0)*pneg
    ppos <- (1-p0)*ppos
  }

  mu <- mixtureupdate$best["mu"]
  sd <- mixtureupdate$best["stdev"]

  x <- seq(xlim[1], xlim[2], length.out=1000)
  y <- pneg*dnorm(x, -mu, sd) + ppos*dnorm(x, mu, sd)
  suppressWarnings(if(is.na(ylim)) ylim <- c(0, 1.1*max(y, p0)))
  plot(x, y, type='l', col=colour, lty=ltype, lwd=lwdth, ylim=ylim,
       main=main, ylab="Pr(x)")
  if(plotp0) points(0, p0, pch=(15+ltype), col=colour)
}


linesMixturePrior <- function(mixtureupdate, scaletodens=TRUE, plotp0=TRUE,
                              colour=1, ltype=1, lwdth=2, xlim=c(-5, 5)){
  # linesMixturePrior is meant to be used in conjunction with plotMixturePrior()
  p0 <- pneg <- ppos <- mu <- sd <- numeric(1)
  x <- y <- numeric(1000)

  p0 <- mixtureupdate$best["p0"]
  pneg <- mixtureupdate$best["pmin"]
  ppos <- (1-mixtureupdate$best["pmin"])
  if(scaletodens){
    pneg <- (1-p0)*pneg
    ppos <- (1-p0)*ppos
  }

  mu <- mixtureupdate$best["mu"]
  sd <- mixtureupdate$best["stdev"]

  x <- seq(xlim[1], xlim[2], length.out=1000)
  y <- pneg*dnorm(x, -mu, sd) + ppos*dnorm(x, mu, sd)
  lines(x, y, type='l', lty=ltype, col=colour, lwd=lwdth)
  if(plotp0) points(0, p0, pch=(15+ltype), col=colour)
}


plotNonParaPrior <- function(nonparaupdate, scaletodens=TRUE, plotp0=TRUE,
                             colour=1, ltype=1, lwdth=1, xlim=c(NA, NA),
                             ylim=NA, main=NULL){
  # scaletodens = logical indicating whether the nonzero part of the
  # distribution (that integrates to 1) should be scaled, so that
  # [ p0*diracdelta(0) + (nonzero part) ] integrates to 1
```

```
  # plotp0 = logical indicating whether the point mass on zero is plotted
  p0 <- x <- y <- numeric()

  p0 <- nonparaupdate$p0est
  x <- nonparaupdate$priornew[, "nx"]
  y <- nonparaupdate$priornew[, "ny"]
  if(scaletodens) y <- (1-p0)*y

  xmax <- ifelse(is.na(xlim[2]), max(x), xlim[2])
  xmin <- ifelse(is.na(xlim[1]), min(x), xlim[1])

  suppressWarnings(if(is.na(ylim)) ylim <- c(0, 1.1*max(y, p0)))
  plot(x, y, type='l', col=colour, lty=ltype, lwd=lwdth, xlim=c(xmin, xmax),
       ylim=ylim, main=main, ylab="Pr(x)")
  if(plotp0) points(0, p0, pch=(15+ltype), col=colour)
}


linesNonParaPrior <- function(nonparaupdate, scaletodens=TRUE, plotp0=TRUE,
                              colour=1, ltype=1, lwdth=2){
  # linesNonParaPrior is meant to be used in conjunction with plotNonParaPrior()
  p0 <- x <- y <- numeric()

  p0 <- nonparaupdate$p0est
  x <- nonparaupdate$priornew[, "nx"]
  y <- nonparaupdate$priornew[, "ny"]
  if(scaletodens) y <- (1-p0)*y

  lines(x, y, type='l', col=colour, lty=ltype, lwd=lwdth)
  if(plotp0) points(0, p0, pch=(15+ltype), col=colour)
}


plotFASPrior <- function(fitallshrink, xlimit=NA, ylimit=NA, colour=1, ltype=1){
  mu <- fitallshrink$priors$mufixed
  sd <- sqrt(1/fitallshrink$priors$precfixed)
  suppressWarnings(if(is.na(xlimit)) xlimit <- c(mu-(4*sd), mu+(4*sd)))
  x <- seq(xlimit[1], xlimit[2], length.out=1000)
  y <- dnorm(x, mu, sd)
  suppressWarnings(if(is.na(ylimit)) ylimit <- c(0, 1.1*max(y)))
  plot(x, y, type='l', col=colour, lty=ltype, ylim=ylimit)
}


linesFASPrior <- function(fitallshrink, colour=1, ltype=1, xlimit=NA){
  # linesFASPrior is meant to be used in conjunction with plotFASprior()
  mu <- fitallshrink$priors$mufixed
  sd <- sqrt(1/fitallshrink$priors$precfixed)
  suppressWarnings(if(is.na(xlimit)) xlimit <- c(mu-(6*sd), mu+(6*sd)))
  x <- seq(xlimit[1], xlimit[2], length.out=1000)
  y <- dnorm(x, mu, sd)
  lines(x, y, type='l', col=colour, lty=ltype, ylim=ylim)
}
```

```
# auc (= area under curve) is used to normalize the marginals obtained
# from NonParaUpdatePosterior(), since these are not densities
auc <- function(dens){
  x <- dens[, 1]
  y <- dens[, 2]
  dx <- sapply(1:(length(x)-1), function(i) x[i+1]-x[i])
  sum(y[-1]*dx)
}

plotPosterior <- function(upobj, index=1, xlim=c(NA, NA), ymax=1.1){
  # index = numeric vector indicating which marginals to draw
  p <- length(index)
  xmax.data <- max(sapply(1:p, function(i)
    max(upobj[[index[i]]]$postbetanon0[[1]][, "x"])))
  xmin.data <- min(sapply(1:p, function(i)
    min(upobj[[index[i]]]$postbetanon0[[1]][, "x"])))
  xmax <- ifelse(is.na(xlim[2]), xmax.data, xlim[2])
  xmin <- ifelse(is.na(xlim[1]), xmin.data, xlim[1])
  with(upobj[[index[1]]], plot(
    x=postbetanon0[[1]][, "x"],
    y=((1-postbeta0)/auc(postbetanon0[[1]]))*postbetanon0[[1]][, "y"],
    type='l', xlim=c(xmin, xmax), ylim=c(0, ymax), xlab="beta", ylab="pr(beta)"
  ))
  with(upobj[[index[1]]], points(0, postbeta0, pch=16))
  if(p > 1){
    cols <- rainbow(p-1)
    for(i in 2:p){
      with(upobj[[index[i]]], lines(
        x=postbetanon0[[1]][, "x"],
        y=((1-postbeta0)/auc(postbetanon0[[1]]))*postbetanon0[[1]][, "y"],
        col=cols[i-1]
      ))
      with(upobj[[index[i]]], points(0, postbeta0, pch=16, col=cols[i-1]))
    }
  }
}

linesPosterior <- function(upobj, index=1, lty=2, col=NA){
  # linesPosterior() is meant to be used in conjunction with plotPosterior();
  # same index will have same color, but different linetype.
  # convenient to compare marginals calculated under different constraints
  p <- length(index)
  if(is.na(col)) col <- 1
  with(upobj[[index[1]]], lines(
    x=postbetanon0[[1]][, "x"],
    y=((1-postbeta0)/auc(postbetanon0[[1]]))*postbetanon0[[1]][, "y"],
    col=col, lty=lty
  ))
  with(upobj[[index[1]]], points(0, postbeta0, pch=16, col=col))
  if(p > 1){
    if(is.na(col)) cols <- rainbow(p-1) else cols=col
    for(i in 2:p){
```

```
      with(upobj[[index[i]]], lines(
        x=postbetanon0[[1]][, "x"],
        y=((1-postbeta0)/auc(postbetanon0[[1]]))*postbetanon0[[1]][, "y"],
        col=cols[i-1], lty=lty
      ))
      with(upobj[[index[i]]], points(0, postbeta0, pch=16, col=cols[i-1]))
    }
  }
}

##############################
### Data manipulation
edgeRnormalization <- function(counts){
  # edgeRnormalization normalizes a table of counts according to TMM-method,
  # implemented in as is usually done when creating a DGEList-object
  require(edgeR)
  rellibsize <- colSums(counts)/exp(mean(log(colSums(counts))))
  nf = calcNormFactors(counts, "TMM")*rellibsize
  normcounts = round(sweep(counts, 2, nf, "/"))
  normcounts
}

samp <- function(factor, npergroup=10, replace=FALSE){
  # samp() is used to take a random sample from a factor, with an equal
  # number of subjects from every factor level. output is an index vector.
  # npergroup = size per factor level
  # replace = allow sampling with replacement?
  l <- levels(factor)
  ind <- matrix(NA, npergroup, length(l))

  if(any(sapply(seq_along(l), function(i)
    length(which(factor==l[i])) < npergroup))){
    replace <- TRUE
    warning("factor has less entries than specified npergroup,
            subjects will be recycled")
  }
  ind <- sapply(seq_along(l), function(i)
    sample(which(factor==l[i]), size=npergroup, replace=replace))
  return(sort(as.numeric(ind)))
}

sampByFraction <- function(factor, n=20){
  # sampByFraction() is used to take a random sample from a factor,
  # keeping the original ratio's of subjects per level intact
  # Output is an index vector
  # n = total sample size
  l <- levels(factor)
  ntotal <- sapply(seq_along(l), function(i) sum(factor==l[i]))

  npergroup <- ntotal*(n/length(factor))

  # make sure the returned index is of length n:
  if(sum(round(npergroup)) < n) npergroup[which.max(npergroup%%1)] <-
```

```
    npergroup[which.max(npergroup%%1)] + 1
  if(sum(round(npergroup)) > n) npergroup[which.min(npergroup%%1)] <-
    npergroup[which.min(npergroup%%1)] - 1

  npergroup <- round(npergroup)

  ind <- sapply(seq_along(l), function(i)
    sample(which(factor==l[i]), size=npergroup[i]))
  return(sort(as.numeric(unlist(ind))))
}

removeZeroes <- function(counts, propnon0=0.2, report=TRUE){
  # removeZeroes() removes all rows from a dataset that have less than
  # propnon0 nonzero entries
  remove <- apply(counts, 1, function(row)
    sum(row!=0) < propnon0*ncol(counts))
  if(report) cat("removeZeroes():", sum(as.numeric(remove)),
                 "out of", nrow(counts), "rows removed \n")
  return(counts[!remove, ])
}

createZeroes <- function(x, nzero){
  # assigns nzero elements of vector x the value 0
  # created for own convenience, slower than
  #   x[sample(seq_along(x), size=nzero)] <- 0
  if(!is.numeric(nzero) | nzero < 0 | nzero > length(x)){
    stop("nzero must be positive and no larger than the length of x")
  }

  out <- numeric(length(x))
  index <- sample(seq_along(x), size=(length(x)-nzero))
  out[index] <- x[index]
  return(out)
}

calcMode <- function(x, multiplemodes=T){
  # calcMode() returns the mode(s) of input vector x
  # set multiplemodes=F to return only the first element of the set of modes
  out <- as.numeric(names(table(x))[table(x) %in% max(table(x))])
  if(!multiplemodes) out <- out[1]
  cat("Largest occurence =", table(x)[table(x) %in% max(table(x))][1], "\n")
  return(out)
}

##############################
### Obsolete power calculation functions;
### used for exploration and analysis before calcPower() and powerCurve()
predictDEFeats <- function(nsample, b0, b1, size, q0, FDRcutoff=0.1){
  # predictDEFeats() generates count datasets based on input NB coefficients,
  # and analyzes these with edgeR. The number of differentially expressed genes,
  # the number of rejections and the number of true positives is returned

  if(any(is.na(c(nsample, b0, b1, size)))) stop("Input contains NA values")
```

```
  ### Check input:
  if(!all.equal(length(b0), length(b1), length(size)) |
       (!is.null(q0) & length(b0)!=length(q0))){
    stop("Coefficient vectors are not of the same length")
  }
  require(edgeR)
  p <- length(b0)

  # round to nearest smaller even number:
  while(nsample%%2) nsample <- ceiling(nsample - 1)
  if(is.null(q0)) q0 <- numeric(p)  # when likelihood is nb, not zi-nb

  ### Generate count matrix:
  counts <- matrix(0, p, nsample)  # matrix of counts
  counts[, 1:(nsample/2)] <- t(sapply(1:p, function(i)
    createZeroes(rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i])),
                 nzero=round(nsample*q0[i]/2))))
  counts[, ((nsample/2)+1):nsample] <- t(sapply(1:p, function(i)
    createZeroes(rnbinom(n=nsample/2, size=exp(size[i]), mu=exp(b0[i]+b1[i])),
                 nzero=round(nsample*q0[i]/2))))

  ### Exact testing:
  res <- DGEList(counts, norm.factors=calcNormFactors(counts, method="TMM"),
                 group=factor(rep(1:2, each=ncol(counts)/2)))
  res <- estimateCommonDisp(res)
  res <- estimateTagwiseDisp(res)

  res <- exactTest(res)
  res <- topTags(res, n=nrow(counts), sort.by="none")$table


  ### Output
  return(data.frame(DE.feats=sum(b1!=0),
                    detected.feats=sum(res$FDR <= FDRcutoff),
                    matching.feats=sum(is.element(which(res$FDR <= FDRcutoff),
                                                  which(b1!=0)))))
}

findTags3 <- function(nsample, b0, b1, size, q0=NULL, b1thres=0,
                      fixedcp0=TRUE, fixedseed=TRUE, alpha=0.1){
  # findTags3() returns the number of rejections detected by exact testing
  # on a simulated dataset with input (ZI-)NB parameters,
  # at FDR cutoff level alfa
  # nsample = number of subjects of dataset
  # b0, b1, size, q0 = ZI-NB parameters, as obtained by ShrinkBayes-analysis
  # b1thres = effect size threshold: smaller beta1s are
  #    set to zero before data generation
  # fixedseed = boolean. fix the random seed, so that output will be
  #             identical on consecutive calls
  if(any(is.na(c(nsample, b0, b1, size)))) stop("Input contains NA values")
  if(!all.equal(length(b0), length(b1), length(size)) |
       (!is.null(q0) & length(b0)!=length(q0))){
    stop("Coefficient vectors are not of the same length")
```

```
  }
  require(edgeR)
  p <- length(b0)
  b1[abs(b1)<= b1thres] <- 0

  # round to nearest smaller even number:
  while(nsample%%2) nsample <- ceiling(nsample - 1)

  fctr <- factor(rep(1:2, each=nsample/2))
  wl1 <- which(fctr=="1")  # extra overhead, but will work for factors with
  wl2 <- which(fctr=="2")  # different order

  if(fixedseed){
    if(!exists(".Random.seed", mode="integer", envir=globalenv())) sample(NA)
    oldSeed <- .Random.seed
    set.seed(2909)  # no visible effect on distribution
  }

  # Generate counts:
  sd <- matrix(0, p, nsample)
  sd[, wl1] <- t(sapply(1:p, function(i)
    rnbinom(n=length(wl1), size=exp(size[i]), mu=exp(b0[i]))))
  sd[, wl2] <- t(sapply(1:p, function(i)
    rnbinom(n=length(wl2), size=exp(size[i]), mu=exp(b0[i]+b1[i]))))

  # Create zeroes:
  if(!is.null(q0)){
    if(fixedcp0){
      for(i in 1:p) sd[i, sample(1:nsample, size=round(nsample*q0[i]))] <- 0
    }else{
      sd <- sapply(1:nsample, function(col) (runif(p) > q0)*sd[, col])
    }
  }

  if(fixedseed) assign(".Random.seed", oldSeed, envir=globalenv())

  ######## edgeR inference:
  sd <- DGEList(sd, norm.factors=calcNormFactors(sd, method="TMM"),
                group=fctr, genes=rownames(sd))
  sd <- estimateCommonDisp(sd)
  sd <- estimateTagwiseDisp(sd)

  sd <- exactTest(sd)
  sd <- topTags(sd, n=nrow(sd), sort.by="none")$table

  return(data.frame(ActualPos=sum(b1!=0), Rejections=sum(sd$FDR <= alpha),
                    TruePos=sum(which(b1!=0) %in% which(sd$FDR <= alpha))))

}

##############################
### processing data archives downloaded from TCGA data portal
TCGA.extFileNames <- function(TCGA_dir, mapping=c("gene", "exon", "spljxn"),
```

```
                              technique="IlluminaHiSeq_RNAseq"){
  # extFileNames returns a list of all gene, exon or splice junction
  # RNAseq filenames in a TCGA archive.
  # TCGA_dir = path to unzipped .tgz file downloaded from TCGA data portal;
  #   contains at least folders "METADATA" and "RNAseq"
  # mapping = type of expression signal
  # technique = technical device used to acquire data
  #   (name of the directory above Level_3)
  #   ! These directories may not have consistent names, therefore they
  #   require creative input to resemble structure of the downloaded folder

  strng <- l3dir <- filenames <- character(0)
  strng <- paste(".", mapping[1], ".quant", sep="")
  l3dir <- paste(as.character(TCGA_dir), "/RNASeq/UNC__",
                 technique, "/Level_3", sep="")
  filenames <- list.files(path=l3dir, pattern=strng)
  filenames
}

TCGA.filelistToBarcode <- function(filelist, form=c("UCSC", "complete")){
  # transforms names of tab-delimited files containing RNAseq data,
  # as downloaded from TCGA data matrix, into TCGA subject barcodes
  # set form = "UCSC" to abbreviate to the form "TCGA-xx-xxxx-xxA"
  #              (used to match to UCSC clinical data files)

  n <- length(filelist)
  out <- numeric(n)
  out <- strsplit(filelist, "[.]")
  out <- sapply(1:n, function(i) out[[i]][2])
  if(form[1]=="UCSC") out <- substr(out, 1, 15)
  return(as.character(out))
}

TCGA.createCountMat <- function(
  TCGA_dir, filelist, unit=c("raw_counts", "median_length_normalized", "RPKM"),
  technique="IlluminaHiSeq_RNAseq"
){
  # createCountMat() combines a set of tab delimited text files in a TCGA
  # data archive, containing RNAseq data, into one data frame.
  # Subjects are marked by their so-called TCGA-barcode (column names)
  # TCGA_dir = path to unzipped .tgz file downloaded from TCGA data portal;
  #   contains at least folders "METADATA" and "RNAseq"
  # filelist = a vector with names of text files, usually obtained from
  #   TCGA.extFileNames() function
  # unit = unit to extract

  n <- length(filelist)
  filenames <- character()
  out <- data.frame()

  filenames <- paste(as.character(TCGA_dir), "/RNASeq/UNC__",
                     technique, "/Level_3/", filelist, sep="")
  out <- sapply(1:n, function(i)
```

```
    as.numeric(read.delim(filenames[i], header=T)[, unit[1]]))
  rownames(out) <- read.delim(filenames[1], header=T)[, 1]
  colnames(out) <- TCGA.filelistToBarcode(filelist, "complete")
  return(out)
}

TCGA.checkFeatureNames <- function(
  TCGA_dir, filelist, technique="IlluminaHiSeq_RNAseq"
){
  # checkFeatureNames() returns TRUE when rownames of all files are identical,
  # i.e. when every text file has information on the same feature
  # Should be used prior to createCountMat, but is necessary only once
  #    (hence a separate function)
  # TCGA_dir = path to unzipped .tgz file downloaded from TCGA data portal;
  #   contains at least folders "METADATA" and "RNAseq"
  # filelist = a vector with names of text files, usually obtained from
  #   TCGA.extFileNames() function
  n <- length(filelist)
  filenames <- character()
  featurenames <- data.frame()

  filenames <- paste(as.character(TCGA_dir), "/RNASeq/UNC__",
                     technique, "/Level_3/", filelist, sep="")

  identical <- logical(0)
  featurenames <- sapply(1:n, function(i)
    as.character(read.delim(filenames[i], header=T)[, 1]))
  identical <- sapply(2:n, function(i)
    all.equal(featurenames[, 1], featurenames[, i]))
  return(!any(!identical))
}


##############################
### edgeR wrapper
callEdgeR <- function(countmat, groupfact=NA, designmat=NA, normalize=FALSE,
                      outputtable=TRUE, FDRcutoff=0.1){
  # callEdgeR() runs an edgeR analysis with tagwise dispersion estimation,
  # from input count matrix and
  # groupfact = two-level factor for comparison of two groups, or
  # designmat = model matrix of more complicated experimental design
  #   (using edgeR's likelihood ratio tests instead of exact tests)
  # set normalize = TRUE implements normalization by "TMM" method
  # set outputtable = FALSE to return (number of FDRs <= FDRcutoff)
  #   instead of output table as given by topTags()-function
  if(is.na(groupfact) && is.na(designmat)[1]){
    stop("Specify experimental design \n")
  }
  if(!is.na(groupfact) && !is.na(designmat)[1]){
    cat("Multiple experimental designs specified: Higher-level design
        from model matrix is used within GLM-producedure \n")
  }
  require(edgeR)
```

```
  if(normalize) countmat <- edgeRnormalization(countmat)

  if(!is.na(designmat[1])){  # when design is specified via model matrix
    ll <- DGEList(countmat, group=designmat[, ncol(designmat)])
    ll <- estimateGLMTrendedDisp(ll, design=designmat)
    ll <- estimateGLMTagwiseDisp(ll, design=designmat)

    ll <- glmFit(ll, design=designmat)
    ll <- glmLRT(ll)
    ll <- topTags(ll, n=nrow(countmat), sort.by="none")$table
  }else{  # when design is specified via factor
    ll <- DGEList(countmat, group=groupfact)

    ll <- estimateCommonDisp(ll)
    ll <- estimateTagwiseDisp(ll)

    ll <- exactTest(ll)
    ll <- topTags(ll, sort.by="none", n=nrow(countmat))$table
  }

  if(outputtable) return(ll) else return(sum(ll$FDR <= FDRcutoff))
}
```

C Data-generative settings: sets of coefficients

Figures 17 and 18 show what the effect of various data-generative settings is on average power and abundance. The data-generative model was based on the same 58-sized Pickrell dataset used in chapter 3. In figure 17, effects are sampled from the estimated prior distributions $F_{NP}(\beta_1)$. The number of nonzero effects m_1 can be set at exactly $p \cdot (1 - p_0)$, which is done in the most left combination (*"Intact, m1 fixed"*). In the second combination (*"Intact, m1 not fixed"*), every β_1 has a p_0 probability of being 0 and is otherwise drawn from $F(\beta_1)$. This is determined per effect, hence the actual number of nonzero effects can vary per simulation. Resampling the marginal prior expectations of intercept, count null probability and size was done without replacement, resulting only in a shuffling of the elements in the vectors.

These results were already discussed briefly in Section 3.4. Drawing effect size parameters from one prior distribution, hence removing all correlations with the rest of the data-generative settings, results in higher detection rate of differentially expressed features. Removing the stochasticity in p_0 from the model does not have an effect that is worth mentioning, on either power or abundance. Resampling either intercept, dispersion or count zero probability results in a drastic power decrease.

Random draws from $\pi(\beta_1)$ do not yield as high power as draws from $F(\beta_1)$, and abundances are lower than observed abundance. Drawing from the m_1 posteriors with lowest p_0 results in more rejections. Observed abundances are then comparable to those obtained with effects sampled from $F(\beta_1)$. Note that m_1 is based on $F(\beta_1)$, hence this makes inconsistent use of prior and posterior information. Again, resampling of β_1, n or q_0 dramatically decreases power and abundance.

D edgeR settings

When differential expression is tested in a NB count model, dispersion values can have a large impact on determining the statistical significance. During the edgeR analysis pipeline, multiple values for dispersion are calculated. The first and most straightforward one is the common value, estimated by conditioning on the total observed count. Trended dispersion fits a function through individual dispersion values, with mean expression level as independent variable. Tagwise dispersion is a combination of individual and common dispersion values, as explained in the Methodology section of this thesis.

For the figures below, data was generated from negative binomial distributions with coefficients obtained by a ShrinkBayes-analysis on Pickrell dataset. Before analysis, rows with over 80%

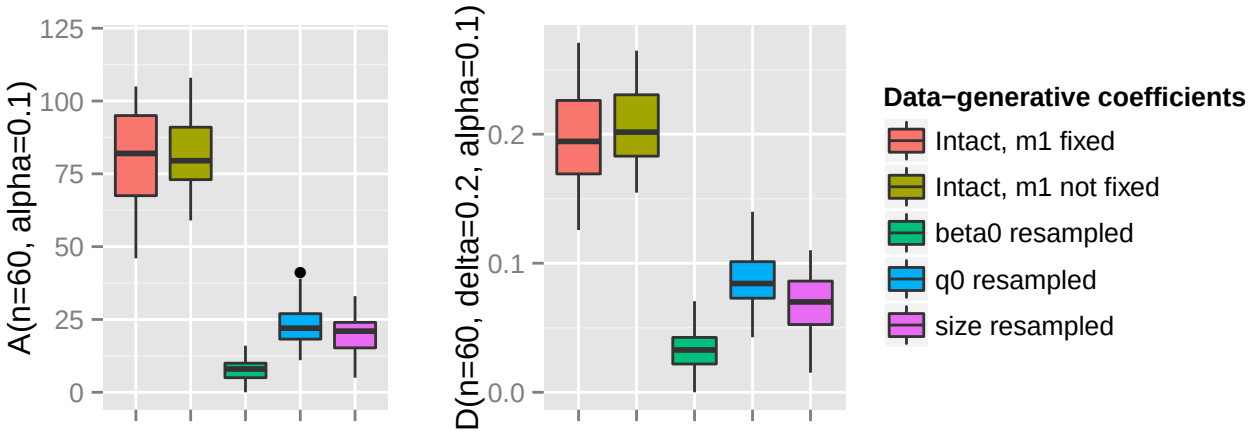

Figure 17: Abundance and power with β_1-values sampled from estimated prior distributions

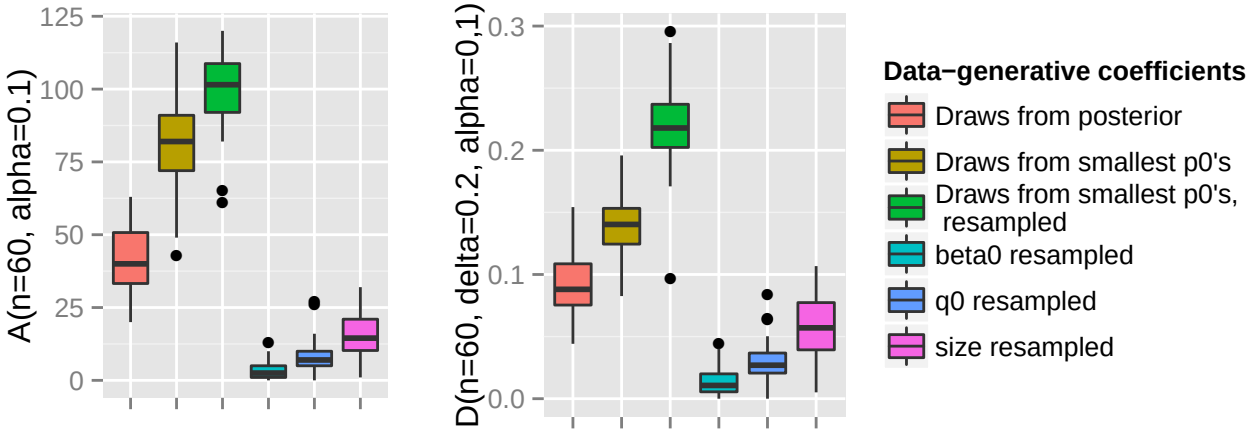

Figure 18: Abundance and power with β_1-values sampled from estimated posterior distributions

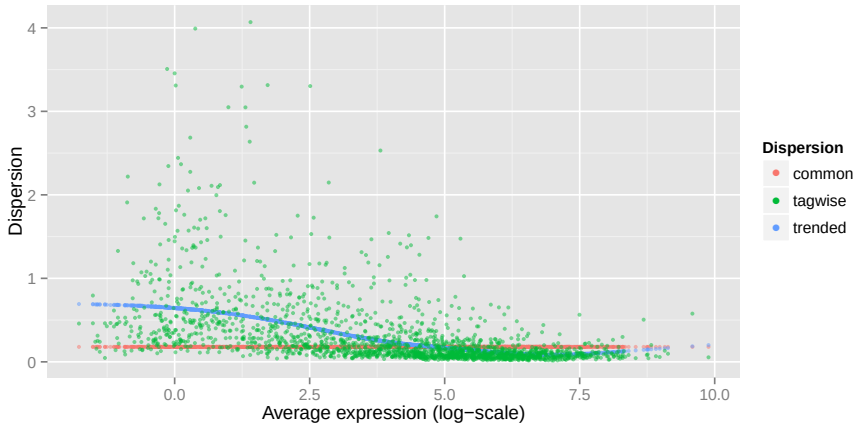

Figure 19: Dispersion values calculated from simulated dataset

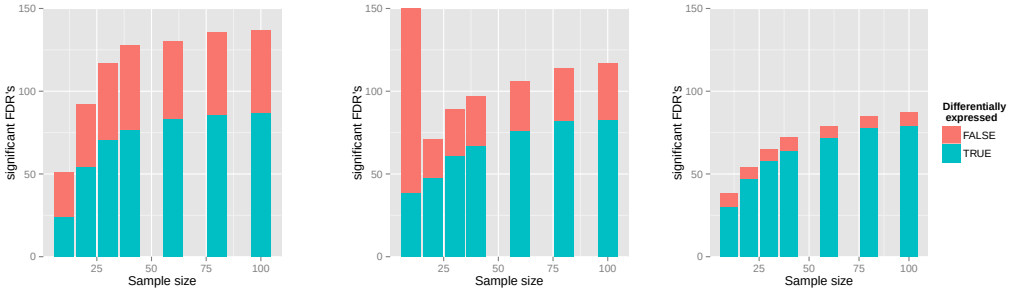

Figure 20: The number of rejections at confidence level 0,1, colored by their true differential expression status, when testing with common (A), trended (B) or tagwise (C) dispersion values. For every sample size, ten random samples were taken from the original dataset, and the average over these ten is shown.

zeroes were removed. Every column was divided by normalization factors calculated according to the TMM-procedure implemented in edgeR. Only the first 2000 features were used. From these, 100 β_1's that had at least a value of $0,5$ were included as a group effect ($\Rightarrow p_0 = 0,95$). Data for 100 subjects were generated independently.

With common and trended dispersion used during the exact testing procedure, observed FDR is higher than set confidence level. Exact testing with tagwise dispersion estimates yields average observed FDR's $\leq \alpha$.