UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Analysis of RNA-Seq data with linear mixed models

Author: **Madeline VAST**
Supervisor: **Céline BUGLI**
Readers: **Bernadette GOVAERTS, Jérôme AMBROISE**
Academic year 2023–2024
Master [120] en statistique, orientation biostatistiques

Abstract

RNA sequencing is a technology used in transcriptomics. It allows to survey the entire transcriptome without prior knowledge of the sequences. This technology is commonly used in experiments to detect genes that are differentially expressed between different conditions. However, the most popular R packages used to this end do not use models appropriate for experimental design with dependence between the observations.

Several approaches have been proposed to perform differential expression analysis on RNA-Seq data with dependence between the observations. Most of them use either generalized linear mixed models or linear mixed models on transformed data.

This thesis aims to compare the package available in R for the differential expression analysis of RNA-Seq data using linear mixed models, *lmerSeq* and *variancePartition*.

This Master's thesis has highlighted that the main difference between the two packages is their approaches to take into account the mean-variance relationship in RNA-Seq data. In addition, although *lmerSeq* performs better for estimating model parameters, the two packages perform similarly for contrast testing of experiments with more than 20 samples. However, the false discovery rate is above the target value when *variancePartition* is used for a sample size below 20.

Acknowledgements

Firstly, I'd like to thank Professor Emeritus Bernadette Govaerts for advising me on this second master's degree and for supervising me on a second Master's thesis.

I would also like to thank PhD Jérôme Ambroise for the subject of this thesis and Professor Céline Bugli for agreeing to be my promoter.

I want to thank all three of them for all the help they gave me during this thesis.

Finally, I'd like to thank all the people who helped in any way in the completion of this thesis, whether they be my close relations or my colleagues at LSBA.

Contents

List of Figures

List of Tables

Introduction

RNA sequencing (RNA-Seq) is a technology used in transcriptomics, the study of the transcriptome. Compared to previous technologies used in this field, RNA-Seq has a higher coverage and a higher definition. It also allows to survey the entire transcriptome without prior knowledge of the sequences. It thus allows to both discover new transcripts and quantify gene expression (Kukurba and Montgomery, 2015). Consequently, RNA-Seq can be used for a wide range of applications. The methods used to analyse the data obtained must thus be adapted to the study and its goals.

A common goal for RNA-Seq studies is the identification of genes expressed differently between different conditions. The most popular R packages to perform this analysis are *edgeR* and *DESeq2*. However, the models they use are not appropriate for experimental designs with dependence between the observations. Several methods have been proposed to perform differential expression analysis with this type of design most of which use either generalized linear mixed models or linear mixed models on transformed data (Vestal et al., 2022).

In 2022, Vestal et al. proposed an R package, *lmerSeq*, to perform differential expression analysis of RNA-Seq data with linear mixed models and compared its performance to other available packages including *variancePartition* proposed by Hoffman and Roussos (2021).

This thesis aims to document the methods used by *lmerSeq* and *variancePartition* for the differential expression analysis of RNA-Seq data with linear mixed models. Furthermore, this thesis aims to compare their performances through a simulation study.

The first chapter of this thesis includes an introduction to RNA sequencing and the workflows for data acquisition and analysis. Furthermore, it includes a summary of the different methods used for the differential expression analysis of RNA-Seq data.

The second chapter of this thesis is dedicated to the comparison of the methods used by the two packages for the differential expression analysis of RNA-Seq data using linear mixed models. In this chapter, the complete workflows used are thus detailed and their implementation in R is discussed.

The last chapter is dedicated to the comparison of the performance of the two packages. To this end, a simulation study was carried out. This chapter detailed how the datasets were simulated and analysed and the results of the simulation study.

Chapter 1

RNA sequencing

This thesis is focused on the differential expression analysis of RNA-Seq data using linear mixed models. This chapter introduces RNA-Seq technology and the RNA-Seq data analysis workflow.

1.1 RNA-Seq technology

RNA sequencing (RNA-Seq) is a technology used in transcriptomics. This technology allows a survey of the entire transcriptome without prior knowledge of the sequences (Wang et al., 2009). Compared to other technologies used in transcriptomics, RNA-Seq has a higher coverage and a higher definition. This allows RNA-Seq to be used to both discover new transcripts and quantify gene expression. The details of a protocol for an RNA-Seq experiment depend on the organism studied and the goals of the researchers. However, a typical experiment includes five steps shown in Figure 1.1 (Kukurba and Montgomery, 2015):

1. The extraction of the RNA from the biological samples
2. The isolation of the desired RNA species
3. The conversion of the RNA to complementary DNA
4. The preparation of the sequencing library
5. The sequencing of the RNA on a next-generation sequencing platform

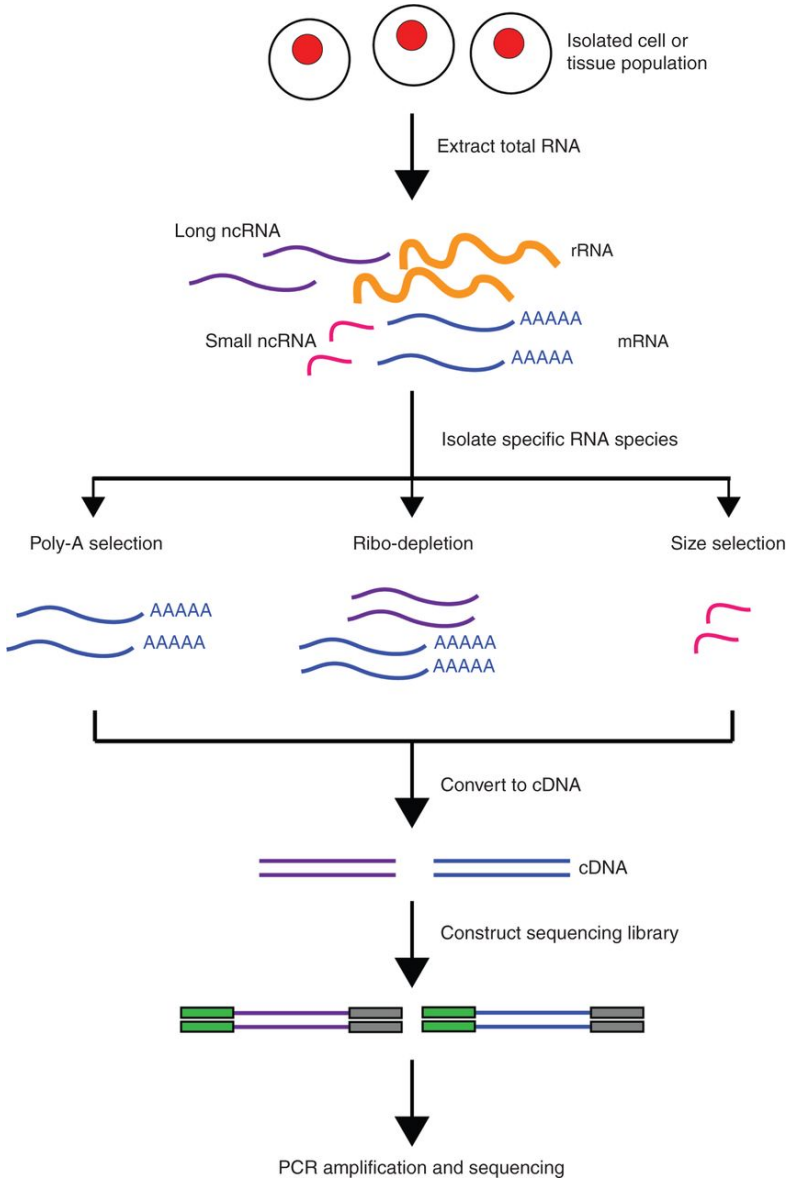


Figure 1.1: Overview of an RNA-Seq experiment from Kukurba and Montgomery (2015)

1.2 RNA-Seq data analysis

Starting from the raw data obtained after the sequencing, the workflow of an RNA-Seq experiment aimed at analyzing differential expression includes 3 main steps (Conesa et al., 2016):

1. Transcript identification
2. Transcript quantification
3. Differential expression analysis

These steps are accompanied by quality controls (Conesa et al., 2016). This section introduces the different steps of the RNA-Seq data analysis.

1.2.1 Quality control

The first quality check is performed on the raw reads. It includes the analysis of (Conesa et al., 2016):

- The sequence quality
- The guanine-cytosine content
- The presence of adaptors
- The overrepresentation of subsequences
- The number of duplicated reads

The goal of this first quality check is to control for sequencing errors and artefacts due to the PCR amplification (Conesa et al., 2016).

The second quality check is performed once the transcripts have been identified. At this point, the percentage of reads which have been mapped to a transcript is analysed to control the quality of the mapping (Conesa et al., 2016).

The third quality check is performed after the quantification of the transcripts. At this step, GC content and gene length biases are checked (Conesa et al., 2016).

1.2.2 Transcript identification and quantification

The first step of the RNA-Seq data analysis is the transcript identification. If a reference transcriptome or genome is available, reads are mapped to it to infer which transcripts are expressed (Conesa et al., 2016). If there is no reference transcriptome or it is incomplete, reads are assembled into transcripts using de novo reconstruction (Kukurba and Montgomery, 2015).

A common application of RNA-Seq is the estimation of the expression levels of the transcripts. Most methods to estimate these expression levels are based on counting the number of reads mapped to each transcript (Conesa et al., 2016). Alternative methods exist but will not be discussed in this thesis.

1.2.3 Differential gene expression

Once the transcripts have been identified and quantified, the next step is the differential gene expression analysis. Before performing the differential expression analysis, low-expression transcripts are filtered out and data are normalized to allow comparison between samples (Conesa et al., 2016).

Since the quantification is based on the number of reads mapped to a transcript, the data follows a discreet distribution. More precisely, the negative binomial distribution has been shown to best fit the RNA-Seq data distribution (Kukurba and Montgomery, 2015). Among the wide variety of negative binomial models, the most common implementation and the one used for RNA-Seq data is the NB2 model which has the following density (Cameron and Trivedi, 2013):

$$y \sim NB(\mu, \alpha)$$

$$f(y; \mu, \alpha) = \frac{\Gamma(y + \alpha^{-1})}{\Gamma(y+1)\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu}\right)^{\alpha^{-1}} \left(\frac{\mu}{\alpha^{-1} + \mu}\right)^{y}$$

With,

- $y = 0, 1, 2, ...$
- $\alpha \geq 0$ the dispersion parameter
- $E[Y] = \mu$
- $V[Y] = \mu + \alpha\mu^2$

Since RNA-Seq data follow a discreet distribution, it must either be modelled using generalized linear models or transformed to be modelled using linear models which are based on the normal distribution (Conesa et al., 2016).

1.2.3.1 Linear models

The first type of model that can be used for the differential expression analysis is the linear model. As a reminder, a linear model can be written as:

$$y = X\beta + \varepsilon$$

With,

- y the vector of observed responses for one transcript and n samples
- X the model matrix of the experiment
- β the vector of regression coefficients
- ε the vector of errors assumed to follow a multivariate normal distribution $N(0, \sigma^2 I)$

However, the RNA-Seq data do not follow the assumptions of this model. Firstly, they do not follow a normal distribution. The errors are also not homoskedastic, since the variance depends on the mean. The data must therefore be transformed and the variance-mean relationship taken into account in order to use general linear models (Conesa et al., 2016). Several methods exist for this purpose. For example, the package *limma* proposes to use a $\log_2$ transformation and precision weights (Law et al., 2014) and the package *DESeq2* proposes to use a variance-stabilizing transformation (Anders and Huber, 2010).

1.2.3.2 Linear mixed models

One of the assumptions of classical linear models is that the observed responses are independent. However, this is not necessarily the case in RNA-Seq experiments. RNA-Seq can be used for longitudinal studies, for example. In this case, dependence can be taken into account using covariance pattern models or linear mixed models (Verbeke and Molenberghs, 2000). The latter type of model is the one of interest in this thesis. Linear mixed models can be written as follows:

$$y = X\beta + Zb + \varepsilon$$

With,

- y the vector of observed responses
- X the model matrix for the fixed effects
- β the vector of regression coefficients
- Z the model matrix for the random effects
- b the vector of random effects which are assumed to be drawn from a multivariate normal distribution $N(0, \Sigma)$
- ε the vector of errors assume to follow a multivariate normal distribution $N(0, \sigma^2 I)$

Linear mixed model parameters can be estimated using maximum likelihood (ML) estimation, where variance parameters and regression coefficients are estimated simultaneously. However, the estimations of the variance parameters obtained with this method are biased (Verbeke and Molenberghs, 2000).

An alternative method has therefore been proposed in which variance parameters and fixed effects are estimated in two stages. The first is the estimation of variance parameters using restricted maximum likelihood (REML). The second step is to estimate the fixed parameters. The advantage of this method is that the variance estimators are unbiased (Verbeke and Molenberghs, 2000).

Independently of the estimation method used, conditionally on the estimates of the variance parameters, the $\hat{\beta}$ vector follows a multivariate normal of mean β. Hypothesis testing on the fixed effects can be expressed in contrast $L\beta$. These hypotheses can be tested using an approximate Wald test since $L\hat{\beta} \sim N(L\beta, Lvar(\hat{\beta})L')$. However, the tests must take into account that $var(\hat{\beta})$ depends on the estimated values of the variance parameters. The test statistic is thus (Verbeke and Molenberghs, 2000):

$$F = \frac{(\hat{\beta} - \beta)'L'[Lvar(\hat{\beta})^{-1}L']^{-1}L(\hat{\beta} - \beta)}{rank(L)}$$

The degrees of freedom of the numerator under the null hypothesis are equal to $rank(L)$. However, the degrees of freedom of the denominator must be estimated from the data (Verbeke and Molenberghs, 2000). Several methods have been proposed to estimate the degrees of freedom. The ones used in this thesis are the Kenward-Roger and the Satterthwaite approximations.

1.2.3.3 Generalized linear models

Another method to perform the differential expression analysis is to use generalized linear models (GLMs). GLMs are an extension of the linear models to other distributions of the exponential family. They are composed of three elements (McCullagh and Nelder, 1989):

1. y a vector of independent observations of the random variable Y which follows a distribution from the exponential family of mean μ
2. η the linear predictor which is given by $\eta = X\beta$
3. $g(.)$ the link function between η and μ such that $\eta = g(\mu)$

The maximum likelihood estimators of the parameters β can be obtained by iterative weighted least square (IWLS) as proposed by McCullagh and Nelder (1989). In this method, the regression is performed on an adjusted dependent variable z defined as:

$$z = g(\hat{\mu}) + (y - \hat{\mu})g'(\hat{\mu})$$

Where $\hat{\mu}$ are the fitted values.

The weights also depend on the fitted values and are defined as (McCullagh and Nelder, 1989):

$$W = g''(\hat{\mu})V[Y]|_{\mu=\hat{\mu}}$$

Since both z and W depend on the fitted values, they are recomputed at each iteration. The estimation of the parameters is repeated until the changes in the values are smaller than a set threshold (McCullagh and Nelder, 1989).

The negative binomial distribution can be used to define a GLM if the dispersion parameter α is known or has been previously estimated. Moreover, the link function generally used for a negative binomial GLM is the logarithm function (Dunn and Smyth, 2018).

Chapter 2

R packages

The goal of this Master's thesis is to compare the performance of two packages for the differential expression analysis of RNA-Seq data with linear mixed models. These packages are *variancePartition* proposed by Hoffman and Schadt (2016) and *lmerSeq* proposed by Vestal et al. (2022).

In this chapter, the two packages will be introduced and their specific workflows will be detailed. Both workflows start with the filtration of genes with low expression levels. This step will therefore not be included in the description of the specific workflows for the two methods.

The workflows will be illustrated with an example dataset. This dataset contains the expression of 200 genes for 8 samples. The experiment for the example dataset is composed of two groups with two subjects per group and two observation times (see Table 2.1). The models fitted to the data are composed of fixed effects for the time, the group and their interaction and a random intercept for the subject.

Table 2.1: Design matrix of the example dataset

Subject ID	Time	Group
1	0	0
2	0	0
3	0	1
4	0	1
1	1	0
2	1	0
3	1	1
4	1	1

2.1 variancePartition

The first package tested is the package *variancePartition* developed by Hoffman and Schadt (2016). More precisely, the first method tested is the *dream* method proposed by Hoffman and Roussos (2021) and implemented in the *variancePartition* package. This package is available on Bioconductor and uses linear mixed models with multiple random effects to perform the differential expression analysis of RNA-Seq data.

2.2 variancePartiton - Workflow

As can be seen in Figure 2.1, the workflow of the *dream* method is composed of four steps:

1. The normalization of the raw counts using the trimmed mean of M-values normalization method
2. The transformation of the data and computation of precision weights using voom
3. The fitting of the linear mixed models
4. The testing of the contrasts of interest

2.2.1 Trimmed mean of M-values normalization method

After the filtering, the first step of the workflow with the *dream* method is the normalization of the count matrix. The normalization method included in the documentation for the package is the trimmed mean of M-values normalization method proposed by Robinson and Oshlack (2010) as implemented in the package *edgeR*.

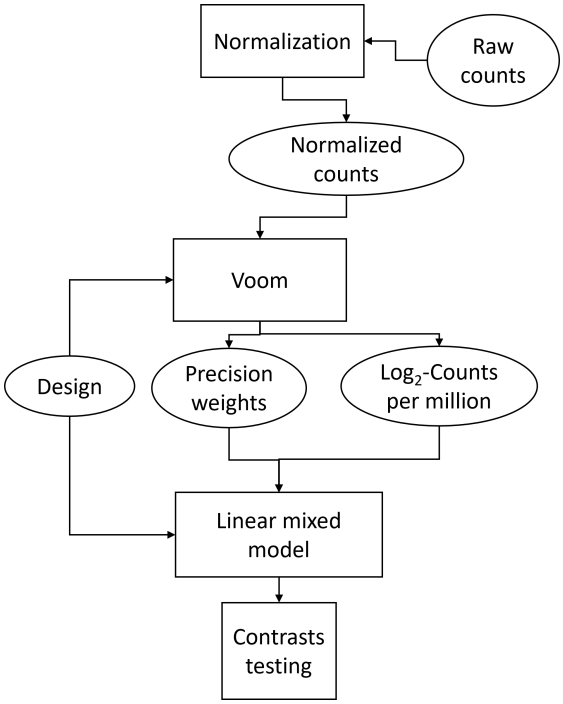


Figure 2.1: Workflow of the *dream* method

Robinson and Oshlack (2010) start with the framework that the expected values of the raw read counts can be modelled as:

$$E[K_{gj}] = \frac{\lambda_{gj} L_g N_j}{S_j}$$

With,

- K_{gj} the raw read count for gene g in sample j
- λ_{gj} the true expression level of gene g in sample j
- L_g the length of the gene g
- N_j the number of reads in the library of sample j
- S_j the total RNA output of sample j

In a differential expression analysis, the variable of interest is the true expression level. But, while the number of reads is known and the length of the gene is often absorbed in the true expression level, the total RNA output of a sample is unknown and cannot be estimated. However, the relative RNA output of two samples $f_j = \frac{S_j}{S_{j'}}$ can be estimated and used as a normalization factor (Robinson and Oshlack, 2010).

Robinson and Oshlack (2010) proposed the trimmed mean of M-values normalization method to estimate this factor. This method is based on the assumption that the majority of genes are not deferentially expressed and is computed in three steps:

1. The selection of a reference sample in relation to which the relative RNA output of each sample will be estimated
2. The computation of the normalization factor which corresponds to an estimation of the relative RNA output of each sample
3. The adjustment of the normalization factors so their geometric mean is equal to 1.

The factors obtained are then multiplied by the library sizes to obtain the effective library sizes of the samples. These effective library sizes are used in the following steps of the workflow to transform the counts into counts per million.

2.2.1.1 Reference sample selection

The first step of the computation of the normalization factors is the selection of a reference sample. This will be the sample relatively to which all the factors will be computed. By default, *edgeR* selects the sample with the upper quartile of gene expression closest to the mean upper quartile. This selection is done in three steps:

1. The calculation of the upper quartile of the raw counts for each sample ($Q_{75;j}$)
2. The normalization of the upper quartile by the library size $\left(f_{75;j} = \frac{Q_{75;j}}{N_j}\right)$
3. The calculation of the deviation from the average over the samples ($|f_{75;j} - \bar{f}_{75}|$)

Example

The values obtained at each step for the example dataset can be seen in Table 2.2. The reference sample for the example is thus the sample number 3 which has the smallest deviation from the mean for its normalized upper quartile.

Table 2.2: Selection of a reference sample for an example with 8 samples. In red is the minimum value of $|f_{75;j} - \bar{f}_{75}|$.

Sample	$\mathbf{N_j}$	$\mathbf{Q_{75;j}}$	$\mathbf{f_{75;j}}$	$\mathbf{\lvert f_{75;j} - \bar{f}_{75} \rvert}$
1	301812	1206.50	0.00400	0.000716
2	292349	1131.00	0.00387	0.000587
3	382259	1184.75	0.00310	0.000182
4	302856	1214.25	0.00401	0.000728
5	341692	1196.25	0.00350	0.000219
6	276972	1120.00	0.00404	0.000762
7	648550	1162.25	0.00179	0.001490
8	554327	1077.00	0.00194	0.001339

2.2.1.2 Normalization factor computation

Once the reference sample is selected, the normalization factor is computed for each sample in relation to the reference sample. As a reminder, Robinson and Oshlack (2010) proposed to use an estimation of the relative RNA production of a sample compared to the reference sample as a normalization factor. This estimation is based on a trimmed mean of the gene-wise $\log_2$-fold-change.

The computation of the trimmed mean of the gene-wise $\log_2$-fold-change is done for each sample separately. It starts with the calculation, for sample j, of the gene-wise $\log_2$-fold-change (M_{gj}) and the absolute expression level (A_{gj}) which are defined by Robinson and Oshlack (2010) as:

$$M_{gj} = \log_2 \frac{K_{gj}/N_j}{K_{gr}/N_r}$$

$$A_{gj} = \frac{1}{2} \log_2 \left(\frac{K_{gj}}{N_j} \times \frac{K_{gr}}{N_r} \right)$$

Where,

- K_{gj} is the raw reads count for gene g in sample j
- K_{gr} is the raw reads count for gene g in the reference sample r
- N_j is the library size of sample j
- N_r is the library size of the reference sample r

The genes with non-finite values of M_{gj} and/or A_{gj} are then removed from the data for the rest of the computation of the normalization factor of the sample.

Afterwards, the remaining genes are trimmed based on their M_{gj} and A_{gj} values. By default, Robinson and Oshlack (2010) proposed to trim the genes with the:

- 30% higher and 30% lower values of M_{gj} for the sample
- 5% higher and 5% lower values of A_{gj} for the sample

Finally, the relative RNA output is estimated by taking the weighted mean of the remaining M_{gj}. The weights used by Robinson and Oshlack (2010) correspond to the inverse of the approximate asymptotic variances of the M_{gj}. By using the delta method, Robinson and Oshlack (2010) approximate the variances as :

$$\hat{\sigma}^2_{M_{gj}} = \frac{N_j - K_{gj}}{N_j K_{gj}} + \frac{N_r - K_{gr}}{N_r K_{gr}}$$

The trimmed mean of M_{gj} values (TMM_j) is thus equal to:

$$TMM_j = \frac{\sum_{g \in G^*} \frac{M_{gj}}{\hat{\sigma}^2_{M_{gj}}}}{\sum_{g \in G^*} \frac{1}{\hat{\sigma}^2_{M_{gj}}}}$$

With, G^* the ensemble of genes which were not trimmed.

Finally, the normalization factor (f_{TMM_j}) for the sample is given by:

$$f_{TMM_j} = 2^{TMM_j}$$

Example

As mentioned previously, in the example dataset the reference sample is the third sample. The relative RNA outputs of the other samples are thus in relation to the third sample.

For the first sample, the M_{g1} and A_{g1} values are represented in Figures 2.2 and 2.3.

From the 200 genes in the example dataset, 3 have a non-finite M_{g1} and/or A_{g1}. They are thus removed for the rest of the computation of the normalization factor for this sample. From the 197 remaining genes, 121 are trimmed because they correspond to the 30% higher and 30% lower values of M_{g1} and/or the 5% higher and 5% lower values of A_{g1}.

The weighted mean is thus calculated on the 76 remaining genes. For the first sample, TMM_1 is equal to 0.3172 and thus the normalization factor is equal to 1.246.

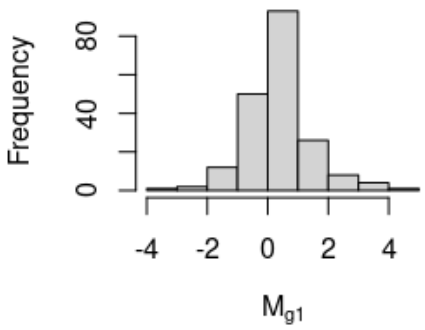


Figure 2.2: Histogram of the gene-wise $\log_2$-fold-changes for the first sample of the example dataset

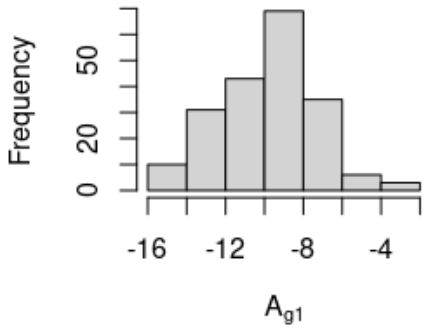


Figure 2.3: Histogram of the absolute expression levels for the first sample of the example dataset

These steps are applied separately to each sample. For the example dataset, the resulting normalization factors are shown in Table 2.3.

Table 2.3: Normalization factors for the example dataset

Sample	$\mathbf{f_{TMM_j}}$
1	1.25
2	1.36
3	1.00
4	1.27
5	1.13
6	1.39
7	0.59
8	0.68

2.2.1.3 Normalization factor adjustment

In the implementation in *edgeR* of the computation of the normalization factors, the factors are adjusted to multiply to 1. This allows a symmetry between the

library sizes and the effective library sizes, i.e. they have the same geometric mean:

$$\left(\prod_{j=1}^{n} N_j\right)^{\frac{1}{n}} = \left(\prod_{j=1}^{n} N_j \times f_j\right)^{\frac{1}{n}}$$

With,

- f_j the adjusted normalization factor for sample j
- n the number of samples

The adjusted normalization factors are thus obtained by dividing the previous factors (f_{TMM_gj}) by their geometric mean:

$$f_j = \frac{f_{TMM_j}}{\left(\prod_{j=1}^{n} f_{TMM_j}\right)^{\frac{1}{n}}}$$

The effective library size ($N_{eff;j}$) used for the rest of the analysis is then obtained by multiplying the adjusted normalization factor by the library size for each sample:

$$N_{eff;j} = N_j \times f_j$$

Example

In the example dataset, the geometric mean of the non-adjusted normalization factors is equal to 1.04. The resulting normalization factors and the effective library sizes for each sample are shown in Table 2.4.

Table 2.4: Results of the calculation of normalisation factors for the example dataset

Sample	$\mathbf{f_{TMM_j}}$	$\mathbf{f_j}$	$\mathbf{N_j}$	$\mathbf{N_{eff;j}}$
1	1.25	1.20	301 812	361 870
2	1.36	1.31	292 349	383 574
3	1.00	0.96	382 259	367 871
4	1.27	1.22	302 856	370 888
5	1.13	1.08	341 692	370 472
6	1.39	1.34	276 972	370 136
7	0.59	0.57	648 550	366 902
8	0.68	0.66	554 327	364 758

2.2.2 Voom

The next step in the *variancePartition* workflow is voom. Voom is a method proposed by Law et al. (2014) to take into account the mean-variance relationship during the differential expression analysis with linear models and adapted by Hoffman and Roussos (2021) to the case of linear mixed model.

As a reminder, the RNA-seq count data follow a negative binomial distribution for which there is a relationship between the mean and the variance. Indeed, larger counts have a larger variance. While $\log_2$-transformation reduces the variance for larger counts, it reduces it to the point where the variance of larger counts is smaller than the variance of smaller counts (Law et al., 2014). However, one of the assumptions of linear models and linear mixed-effects models is the homoskedasticity of the errors. Law et al. (2014) proposed to take into account the mean-variance relationship through precision weights for each observation.

The workflow used by Hoffman and Roussos (2021) to estimate these weights is shown in Figure 2.4 and includes three main steps:

1. The fitting of gene-wise linear mixed models to the $\log_2$-counts per million (cpm)
2. The computation of a trend between the square root of the residual standard deviations and the average $\log_2$-counts using a locally weighted regression (LOWESS)
3. The estimation of the precision weights

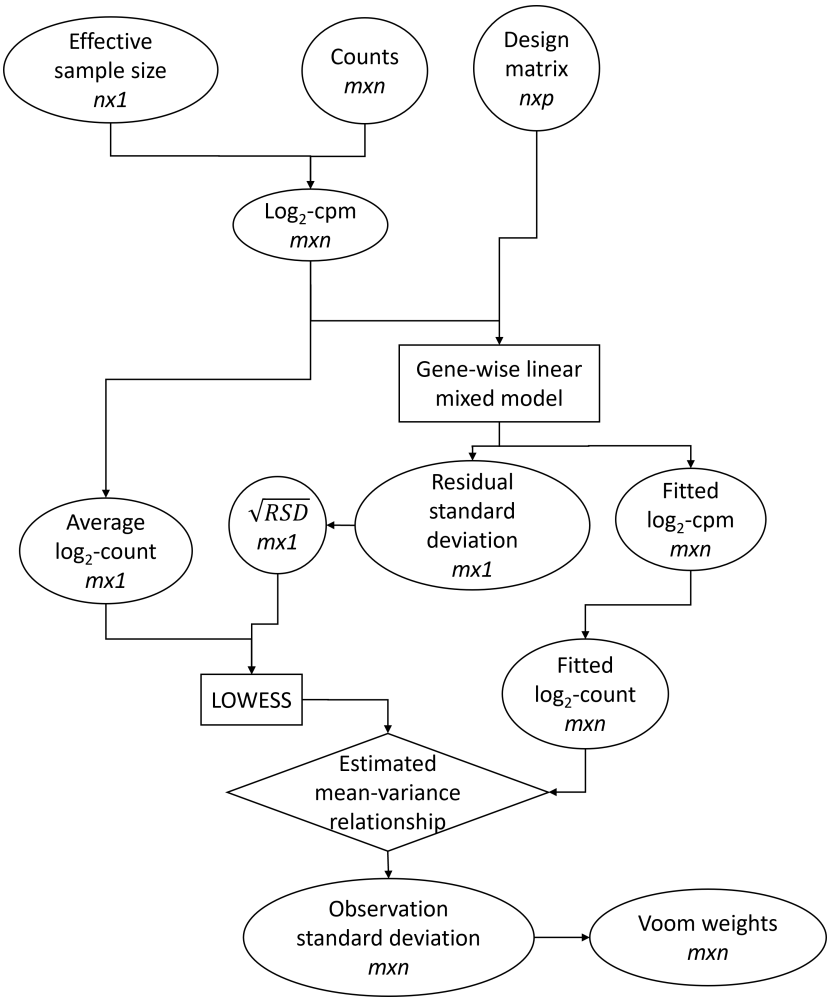


Figure 2.4: Workflow of voom. The ovals represent matrices, the rectangles regression steps and the diamonds estimated relationships. The dimensions of the matrices are given with n the number of samples, m the number of genes and p the number of model parameters.

2.2.2.1 Gene-wise linear mixed model

The first step of voom is the fitting of gene-wise linear mixed models to the $\log_2$-counts per million matrix to estimate a residual deviation for each gene and the fitted values of $\log_2$-counts per million.

The $\log_2$-counts per million are obtained based on the counts and effective sample size with the following formula:

$$Y_{gj} = \log_2 \left(\frac{K_{gj} + 0.5 \times N_{eff;j} / \bar{N}_{eff}}{N_{eff;j} + 1} \times 10^6 \right)$$

Where,

- Y_{gj} is the $\log_2$-count per million for gene g in sample j
- K_{gj} is the raw reads count for gene g in sample j
- $N_{eff;j}$ is the effective library size for sample j
- $\bar{N}_{eff}$ is the average effective library size

In this formula, the raw counts are augmented to avoid non-finite values. The term added to the counts is standardised using the effective library size to avoid introducing artificial variation between the sample (Hoffman et al., 2023).

Furthermore, the complete design matrix is taken into account for the models, i.e. the fixed and random effects. The model fitted to each gene is thus the following linear mixed model:

$$Y_g = \mathbf{X}\beta_g + \mathbf{Z}b_g + \varepsilon_g$$

With,

- Y_g the $\log_2$-counts per million for gene g
- $\mathbf{X}$ the model matrix for the fixed effects
- β_g the regression coefficients for gene g
- $\mathbf{Z}$ the model matrix for the random effects
- b_g the coefficients for the random effects where
  - b_{gki}, the coefficient for the i^{th} level of the k^{th} random effect for gene g which is assumed to be drawn from a normal distribution $N(0, \sigma^2_{gk})$
- ε_g the errors for gene g which follows a multivariate normal distribution $N(0, \sigma^2_g I)$

It is interesting to note that these models do not take into account the heteroskedasticity of the transformed counts.

Example

For the example dataset, the $\log_2$-counts per million are shown in Table 2.5 for the first six genes. As mentioned at the start of the chapter, the models fitted include fixed effects for the time, the group and their interaction and a random intercept for the subjects. The fitted values for the first six genes are shown in Table 2.6 and the residual standard deviations in Table 2.7.

Table 2.5: $\log_2$-cpm of the first six genes of the example dataset

Gene	**Sample**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	9.68	8.76	8.91	9.63	9.85	8.68	8.73	10.27
2	15.90	15.99	16.24	16.53	15.91	16.26	18.03	18.45
3	6.02	5.67	6.00	6.05	6.45	5.79	6.33	5.50
4	11.86	11.69	11.28	12.49	11.56	12.18	9.62	9.61
5	12.11	12.78	12.21	12.67	10.81	12.67	11.92	11.22
6	11.05	10.55	9.13	11.79	9.84	10.57	10.39	7.73

Table 2.6: Fitted values of the first six genes of the example dataset

Gene	**Sample**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	9.70	8.74	8.75	9.79	9.74	8.78	8.98	10.02
2	15.85	16.04	16.23	16.54	15.99	16.18	18.09	18.40
3	5.96	5.73	6.11	5.93	6.24	6.00	6.01	5.82
4	11.77	11.77	11.89	11.89	11.87	11.87	9.62	9.62
5	12.08	12.81	12.47	12.40	11.38	12.10	11.60	11.53
6	10.80	10.80	10.46	10.46	10.21	10.21	9.06	9.06

Table 2.7: Residual standard deviation of the first six genes of the example dataset

Gene	Residual standard deviation
1	0.21
2	0.07
3	0.23
4	0.34
5	0.42
6	0.97

2.2.2.2 LOWESS

The second step of voom is the computation of a trend between the square root of the residual standard deviation and the average $\log_2$-counts of the genes. Law et al. (2014) use a LOWESS curve to obtain a robust estimate of this trend. Hoffman and Roussos (2021) calculate the average $\log_2$-counts of the genes from $\log_2$-counts per million using the following formula:

$$\hat{\mu}_g = \frac{1}{n}\sum_{j=1}^{n} Y_{gj} + \frac{1}{n}\sum_{j=1}^{n} \log_2(N_{eff;j} + 1) - \log_2(10^6)$$

Where,

- $\hat{\mu}_g$ is the average $\log_2$-counts of the gene g
- Y_{gj} is the $\log_2$-count per million for gene g in sample j
- n is the number of samples

Example

The fitted curve for the example dataset is represented in Figure 2.5. As mentioned before, the figure shows that the residual standard deviation is larger for small $\log_2$-counts.

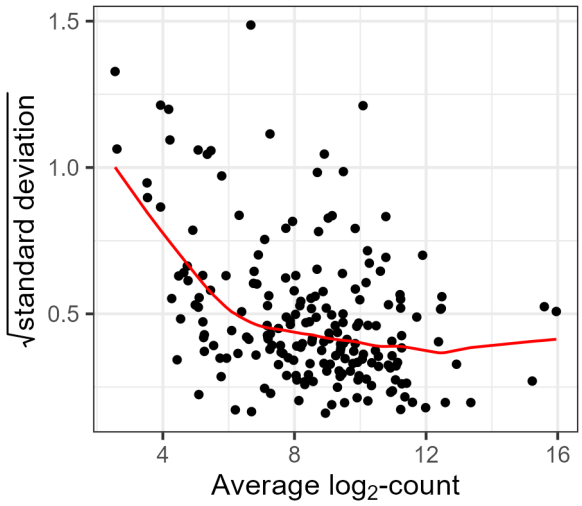


Figure 2.5: Fitted LOWESS curve between the square root of the residual standard deviation and average $\log_2$-count of the genes for the example dataset

2.2.2.3 Precision weights

The last step of voom is the computation of the precision weights or voom weights. These weights are obtained firstly by estimating the square root of the standard deviations at observation-level by interpolation using the trend obtained in the previous step and the fitted values of $\log_2$-counts. The fitted values of $\log_2$-counts are obtained from the fitted values of $\log_2$-counts per million with the following formula:

$$\hat{U}_{gj} = \hat{Y}_{gj} + \log_2(N_{eff;j} + 1) - \log_2(10^6)$$

Where,

- $\hat{U}_{gj}$ is the fitted value of $\log_2$-count for gene g in sample j
- $\hat{Y}_{gj}$ is the fitted value of $\log_2$-count per million for gene g in sample j

Afterwards, the precision weights correspond to the inverse of the variances at observation-level. It is thus given by:

$$w_{gj} = \frac{1}{\left(\widehat{\sqrt{\sigma_{gj}}}\right)^4}$$

Where,

- w_{gj} is the precision weight for the gene g and the sample j
- $\widehat{\sqrt{\sigma_{gj}}}$ is the estimated square root of the standard deviation at observation-level for the gene g and sample j

Example

For the example dataset, the observation-level standard deviation in Table 2.8 can be computed from the fitted values in Table 2.6 transformed in fitted $\log_2$-counts and the trend represented in Figure 2.5. Finally, the precision weights in Table 2.9 can be obtained based on the observation-level standard deviation.

We can see by comparing Table 2.5 and 2.9 that the observation with smaller $\log_2$-cpm will be given less weight in the regression.

Table 2.8: Estimated observation-level standard deviation for the first six genes in the example dataset

Gene	**Sample**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0.19	0.20	0.20	0.19	0.19	0.20	0.20	0.18
2	0.16	0.16	0.16	0.16	0.16	0.16	0.17	0.17
3	0.50	0.53	0.46	0.50	0.44	0.48	0.48	0.52
4	0.16	0.15	0.15	0.15	0.15	0.15	0.19	0.19
5	0.15	0.15	0.15	0.15	0.16	0.15	0.16	0.16
6	0.17	0.17	0.17	0.17	0.18	0.18	0.20	0.20

Table 2.9: Precision weights for the first six genes in the example dataset

Gene	Sample							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	28.55	24.51	24.32	29.07	28.90	24.48	25.10	29.89
2	40.48	39.24	38.65	37.27	39.69	38.85	34.29	34.29
3	4.05	3.59	4.66	4.06	5.24	4.30	4.26	3.65
4	41.45	42.18	42.63	42.73	42.55	42.54	28.23	28.18
5	43.77	45.25	43.49	43.50	37.98	43.79	40.01	39.22
6	34.87	35.23	33.21	33.29	31.50	31.49	25.43	25.40

2.2.3 Model fitting

As mentioned previously, the *variancePartition* workflow used precision weights to take into account the mean-variance relationship. The model fitted to each gene is thus a weighted linear mixed effects model:

$$Y_g = \mathbf{X}\beta_g + \mathbf{Z}b_g + \varepsilon_g$$

With,

- Y_g the $\log_2$-counts per million for gene g
- $\mathbf{X}$ the model matrix for the fixed effects
- β_g the regression coefficients for gene g
- $\mathbf{Z}$ the model matrix for the random effects
- b_g the coefficients for the random effects where
  - b_{gki}, the coefficient for the i^{th} level of the k^{th} random effect for gene g which is assumed to be drawn from a normal distribution $N(0, \sigma^2_{gk})$
- ε_g the errors for gene g which follows a multivariate normal distribution $N(0, \text{diag}(w_g)^{-1}\sigma^2_g)$ with
  - w_g the vector of precision weights for gene g

By default, the models are fitted using restricted maximum likelihood (see Section 1.2.3.2) in the *dream* method. By analysing the code for the *dream* method, one can see that the model is refitted after the first optimization. According to Hoffman and Roussos (2021), this avoids edge cases where the approximate hessian of the first optimizer is negative. The second optimization starts from the values obtained with the first optimizer to converge faster.

2.2.4 Contrasts testing

In the *dream* method, the contrasts are tested using a Wald test. As explained in Section 1.2.3.2, the degrees of freedom must be approximated. Hoffman and Roussos (2021) recommend using the Kenward-Roger approximation when there are less than 20 samples and the Satterthwaite approximation otherwise.

Since the contrasts are tested for each gene, the p-values need to be adjusted for multiple testing. By default, Hoffman and Roussos (2021) propose to use the correction of Benjamini-Hochberg in the *variancePartition* workflow.

2.2.4.1 Empirical Bayes statistics

Since 2023, the package *variancePartition* includes an adaptation of the empirical Bayes method proposed by Smyth (2004) for differential expression analysis using linear models.

In differential expression analysis, the same model is fitted to each gene. Smyth (2004) propose to use this to borrow information from all the genes to improve the inference for each gene. More specifically, Smyth (2004) propose to shrink the residual variance of each gene toward a common value based on prior assumptions. Afterwards, the shrunken residual variances are used to compute moderated test statistics. This increases the degrees of freedom and thus has been shown to improve the statistical power and the control of the false discovery rate of the tests compared to the classic tests (Phipson et al., 2016).

Hoffman et al. (2023) propose an adaptation of this method for linear mixed models which is implemented in the *variancePartition* package and can be used in the workflow for the *dream* method.

2.3 variancePartition - Implementation in R

The implementation in R of the *variancePartition* workflow with moderated test statistics is done in six steps as represented in Figure 2.6:

1. The conversion of the counts matrix into a DGEList using the corresponding function in the *edgeR* package
2. The computation of the normalization factors using *calcNormFactors* from the *edgeR* package
3. The transformation of the count data and computation of the precision weights using *voomWithDreamWeigths* from the *variancePartition* package
4. The model fitting and contrasts testing using the function *dream* from the *variancePartition* package
5. The moderated test statistics computation using *eBayes* from the *variancePartition* package
6. The extraction of the results of contrast testing using *topTable* from the *variancePartition* package

It is interesting to note that the functions for the steps with gene-wise models fitting, i.e. *voomWithDreamWeigths* and *dream*, allow to use parallelization to speed up the computation.

A notable feature of this workflow is that model fitting and contrast testing are carried out by the same function. To use less memory, the adjusted model is thus not entirely included in the output. For example, random effects are not included in the *dream* outputs of the function. To extract the random effects coefficients, one can fit the model with the function *fitVarPartModel* from the *variancePartition* package which gives the complete model as output. However, the model adjusted with the two functions is not the same since the weights are scaled in the code for *fitVarPartModel*.

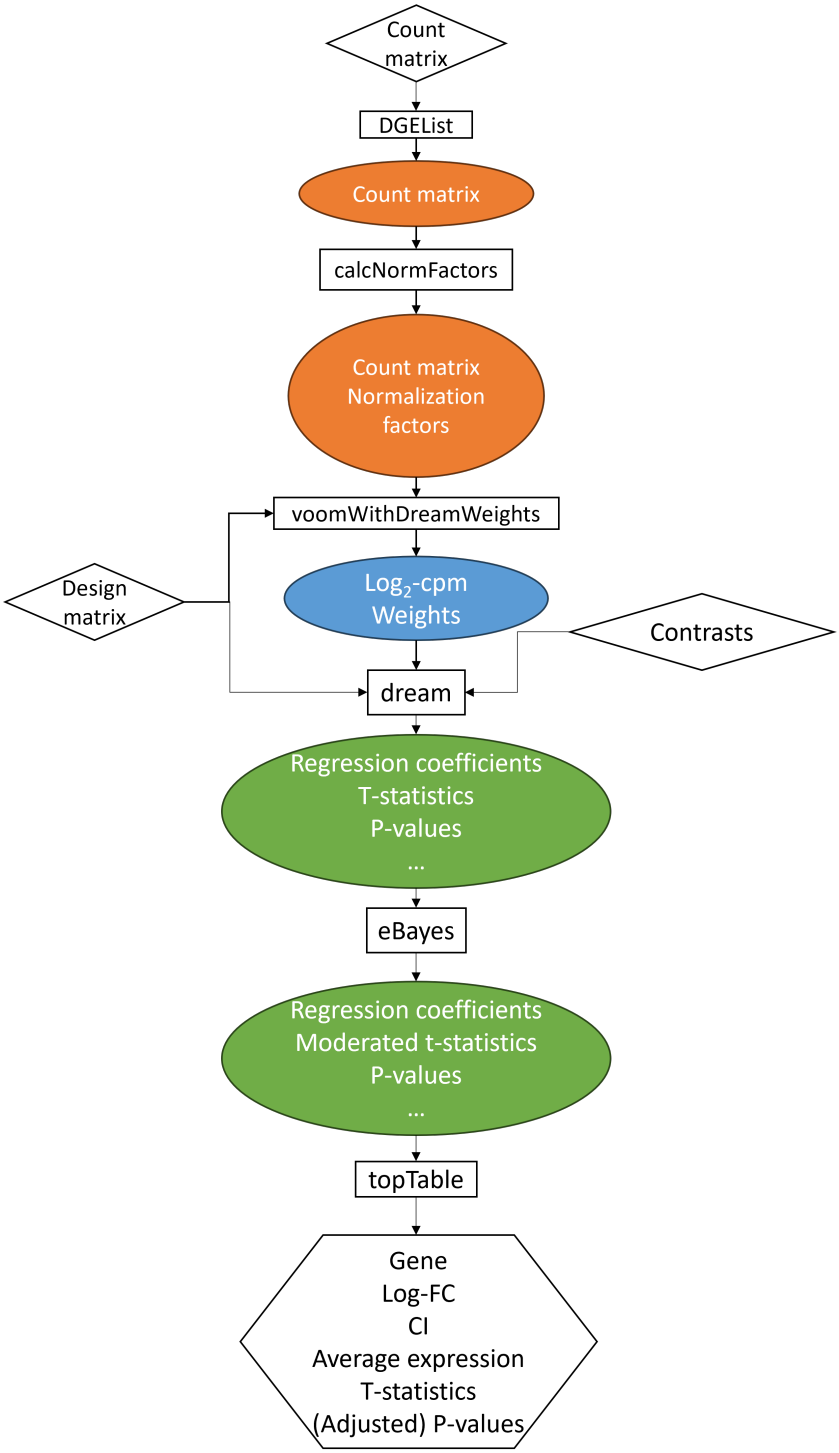


Figure 2.6: R workflow for *dream* analysis. The diamonds represent matrices, the ovals lists, the hexagon dataframe and the rectangles functions. The orange ovals represent DGE-lists, the blue ELists and the green MArrayLM2

2.4 lmerSeq

The second package tested is the package *lmerSeq* proposed by Vestal et al. (2022). This package has been available on Git Hub since 2022. It was created to offer users more flexibility for the analysis of RNA-Seq data using linear mixed models. To this end, *lmerSeq* is compatible with several transformation methods and includes functions to fit both linear mixed models with multiple random effects and models with correlation structure (Vestal et al., 2022). However, at the time of this thesis, the function for models with a covariance structure is no longer available due to updates to the package on which it relies.

2.5 lmerSeq - Workflow

Like the workflow for *variancePartition*, the recommended workflow for *lmerSeq* begins by filtering low-expressed genes. The rest of the workflow consists of three steps as shown in Figure 2.7:

1. The transformation of the count data using a variance-stabilizing transformation
2. The fitting of the gene-wise linear mixed models
3. The testing of the contrasts of interest

2.5.1 Variance-stabilizing transformation

After filtering, the first step of the workflow recommended with the *lmerSeq* package is the transformation of the count data using the variance-stabilizing transformation (VST) proposed by Anders and Huber (2010) and implemented in the package *DESeq2*.

As mentioned before, there is a mean-variance relationship in RNA-Seq data since it follows a negative binomial distribution. Moreover, one of the assumptions of the linear mixed model is the homoskedasticity of the errors. The mean-variance relationship must thus either be removed or taken into account with weights like in the *dream* method. The variance-stabilizing transformation recommended for the *lmerSeq* package by Vestal et al. (2022) removes approximately this relationship and in addition, normalizes the data (Anders and Huber, 2010). Moreover, the transformation is created to behave like a $\log_2$ transformation for large values.

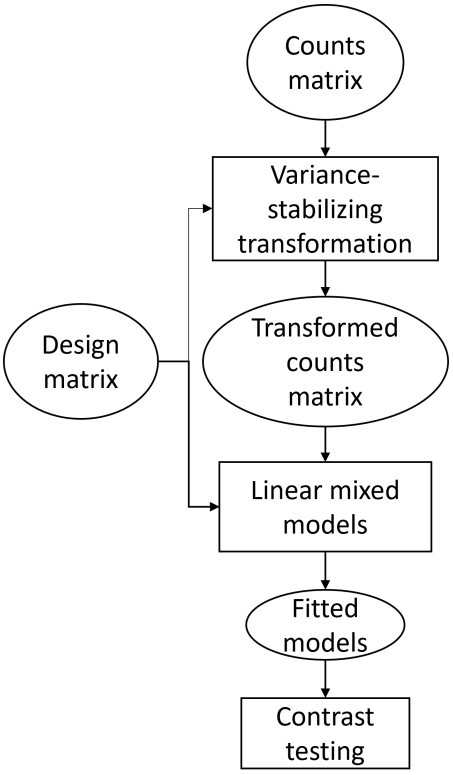


Figure 2.7: Workflow of the *lmerSeq* package

The variance-stabilizing transformation of the data is computed in four steps (see Figure 2.8):

1. The normalization of the counts
2. The estimation of the dispersion for each gene
3. The estimation of the global mean-dispersion relationship
4. The computation of the variance-stabilizing transformation and its application to the data

The variance-stabilizing transformation of Anders and Huber (2010) was not created for data with dependence between the observations and was not adapted by Vestal et al. (2022). This dependence is thus not taken into account during its computation.

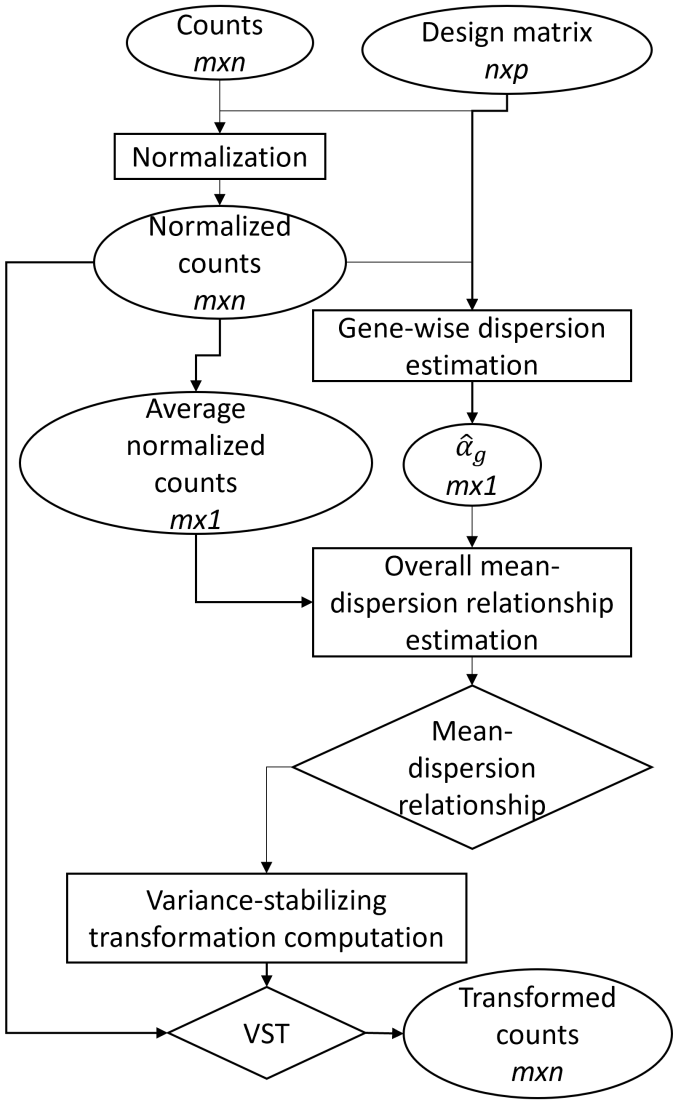


Figure 2.8: Workflow of the variance-stabilizing transformation of the count data. The ovals represent matrices, the rectangles represent steps and the diamonds represent equations. The dimensions of the matrices are given with m the number of genes, n the number of samples and p the number of model parameters.

2.5.1.1 Normalization

The first step of the transformation of the data is the normalization of the raw counts. To do this, Love et al. (2014) assume that the raw reads counts follow a negative binomial distribution:

$$K_{gj} \sim NB(\mu_{gj}, \alpha_g)$$

With,

- K_{gj} the raw read count for gene g and sample j
- α_g the dispersion for gene g
- μ_{gj} the mean for gene g and sample j

Furthermore, according to Love et al. (2014), the mean can be expressed as:

$$\mu_{gj} = s_j \tau_{gj}$$

Where,

- s_j is the normalization constant for sample j
- τ_{gj} is proportional to the expected value of the true concentration of gene g in sample j

Love et al. (2014) estimate the normalization constant by taking the median of the ratio between the gene expression in sample j and the geometric mean of the gene expression in all the samples:

$$s_j = \underset{g;K_g^R \neq 0}{\text{median}} \left(\frac{K_{gj}}{K_g^R} \right) \text{ with } K_g^R = \left(\prod_{v=1}^{n} K_{gv} \right)^{\frac{1}{n}}$$

Where n is the number of samples.

Finally, the normalized counts are obtained as follows:

$$q_{gj} = \frac{K_{gj}}{s_j}$$

Where q_{gj} is the normalized count for gene g and sample j.

Example

Using the same example dataset as previously, the first step of the estimation of the normalization constant is the computation of the geometric mean of gene expression for each gene. The results for the first six genes are shown in Table 2.10.

Table 2.10: Geometric mean over the samples of the gene expression of the first six genes in the example dataset

Gene	$\mathbf{K_g^R}$
1	234.36
2	38352.25
3	22.75
4	921.84
5	1563.91
6	413.83

The second step is to calculate the ratio between the gene expression in sample j and the geometric mean of the gene expression in all the samples and to take the median as normalization constants. Figure 2.9 shows a histogram of the values of the ratio for the first sample with the median marked by a red line. The normalization constant of each sample is shown in Table 2.11.

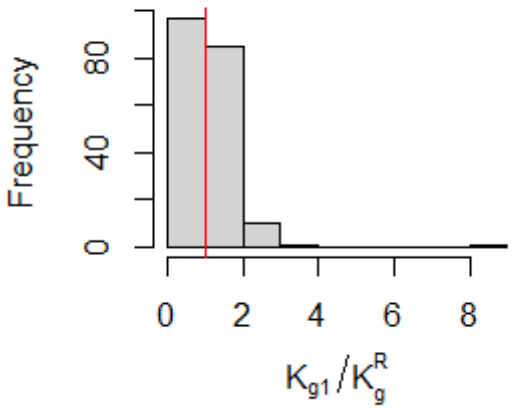


Figure 2.9: Histogram of the values of K_{gj}/K_g^R for the first sample in the example dataset. The red line corresponds to the median.

Table 2.11: Normalization constants for the example dataset

Sample	s_j
1	0.999
2	1.048
3	1.029
4	1.023
5	1.019
6	0.983
7	1.033
8	0.986

2.5.1.2 Estimation of gene-wise dispersion

The second step of the computation of the variance-stabilizing transformation is the estimation of the dispersion parameter for each gene. To do this, Love et al. (2014) propose to use a maximum-likelihood estimation in three steps (see Figure 2.10):

1. A first estimation of the gene-wise dispersion ($\hat{\alpha}_{g_{MoM}}$) using a rough method of moments estimator
2. The estimation of fitted values for the counts ($\hat{\mu}^0_{gj}$) using a negative binomial generalized linear model
3. A final estimation of the gene-wise dispersion ($\hat{\alpha}_g$) obtained by maximizing an adjusted likelihood

Rough method of moments estimation

The first step of the estimation of the gene-wise dispersion is a rough method of moments estimation. As mentioned in Section 1.2.3, for negative binomial data the dispersion parameter is equal to (Love et al., 2014):

$$\alpha = \frac{\sigma^2 - \mu}{\mu^2}$$

For a sample $Y_1, ..., Y_n \overset{iid}{\sim} NB(\mu, \alpha)$, the method of moments estimator of the dispersion is thus given by:

$$\hat{\alpha} = \frac{\frac{1}{n}\sum_{j=1}^{n}\left(Y_j - \bar{Y}\right)^2 - \bar{Y}}{\bar{Y}^2} \text{ with } \bar{Y} = \frac{1}{n}\sum_{j=1}^{n} Y_j$$

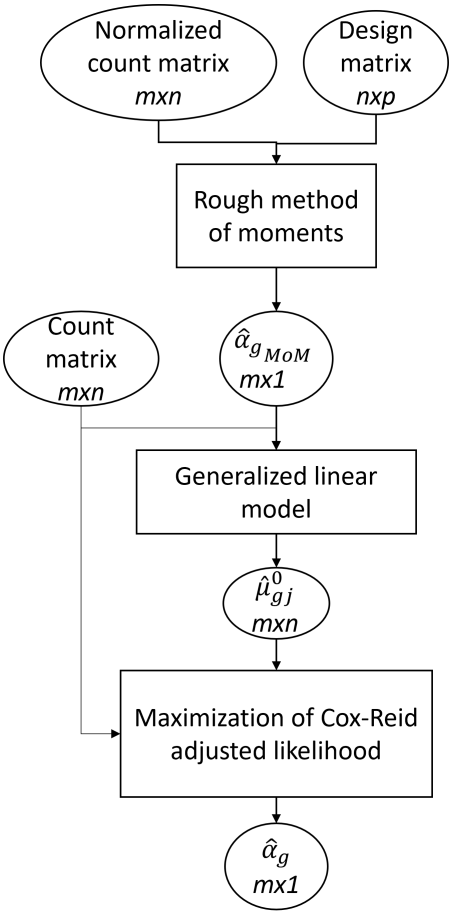


Figure 2.10: Workflow of the estimation of gene-wise dispersion. The ovals represent matrices and the rectangles represent the steps. The dimensions of the matrices are given with m the number of genes, n the number of samples and p the number of model parameters.

Love et al. (2014) compute the first estimates of gene-wise dispersions on a rough method of moments applied to the normalized counts. They roughly estimate this value with two methods developed empirically and then take the smallest estimated value.

The first estimator ($\hat{\alpha}_{g;r}$) proposed by Love et al. (2014) is based on the within-group variances and means and is given by:

$$\hat{\alpha}_{g;r} = \frac{1}{n-p} \sum_{j=1}^{n} \frac{(q_{gj} - \hat{\mu}_{gj})^2 - \hat{\mu}_{gj}}{\hat{\mu}_{gj}^2}$$

Where,

- n is the number of samples
- p is the number of model parameters
- q_{gj} is the normalized count for gene g in sample j
- $\hat{\mu}_{gj}$ is the estimated normalized count value of gene g for sample j

The estimated normalized count value ($\hat{\mu}_{gj}$) is the fitted value from the following linear model:

$$q_{gj} = X_j \beta_g + \varepsilon_{gj}$$

Where,

- X_j is the j^{th} row of the model matrix for the fixed effects
- β_g are the regression coefficients for gene g
- ε_{gj} is the error for sample j and gene g with $\varepsilon_{gj} \sim N(0, \sigma_g^2)$

The second estimator ($\hat{\alpha}_{g;m}$) proposed by Love et al. (2014) use the global means and variances and is given by:

$$\hat{\alpha}_{g;m} = \frac{\hat{\sigma}_g^2 - \bar{\mu}_g \times \overline{s^{-1}}}{\bar{\mu}_g^2}$$

Where,

- $\hat{\sigma}_g^2$ is the variance of the normalized counts for gene g
- $\bar{\mu}_g$ is the mean of normalized counts q_{gj} for gene g
- $\overline{s^{-1}}$ is the mean of the inverse of the normalization constants s_j

As mentioned before, the rough method of moments estimator of the gene-wise dispersion ($\hat{\alpha}_{g;MoM}$) proposed by Love et al. (2014) is then given by:

$$\hat{\alpha}_{g;MoM} = \min(\hat{\alpha}_{g;r}, \hat{\alpha}_{g;m})$$

Generalized linear model

The second step of the estimation of the gene-wise dispersion is the fitting of a generalized linear model (GLM) to obtain fitted count values ($\hat{\mu}^0_{gj}$). More specifically, Love et al. (2014) assume that the counts follow the negative binomial given below:

$$K_{gj} \sim NB(\mu_{gj}, \hat{\alpha}_{g;MoM})$$

With,

- K_{gj} the count for the gene g and sample j
- μ_{gj} the mean of the count for gene g and sample j
- $\hat{\alpha}_{g;MoM}$ the rough method of moments estimator of the dispersion of gene g

Love et al. (2014) also assume that the mean can be predicted with the following equation:

$$\log \mu_{gj} = X_j \beta_g + \varepsilon_{gj}$$

Where,

- X_j is the j^{th} row of the model matrix for fixed effects
- β_g are the regression coefficients for the gene g
- ε_{gj} is the error for sample j and gene g

This model is fitted by iterative weighted least-square (see Section 1.2.3.3).

Maximum-likelihood estimation

The final step of the estimation of gene-wise dispersion proposed by Love et al. (2014) is the maximum-likelihood estimation. More precisely, the final estimates ($\hat{\alpha}_g$) are obtained by maximizing the Cox-Reid adjusted likelihood conditioned on the fitted values ($\hat{\mu}^0_{gj}$). The estimates are thus given by,

$$\hat{\alpha}_g = \arg\max_{\alpha} \ \ell_{CR}(\alpha; \hat{\mu}^0_{gj}, K_{gj})$$

Where, $\ell_{CR}(\alpha; \hat{\mu}^0_{gj}, K_{gj})$ is the adjusted Cox-Reid log-likelihood which is given by

$$\ell_{CR}(\alpha; \hat{\mu}^0_{gj}, K_{gj}) = \ell(\alpha; \hat{\mu}^0_{gj}, K_{gj}) - \frac{1}{2}\log\left(|X^t W X|\right)$$

Where,

- $\ell(\alpha; \hat{\mu}^0_{gj}, K_{gj})$ is the log-likelihood of a negative binomial
- $\frac{1}{2}\log\left(|X^t W X|\right)$ is the penalty term proposed by Cox and Reid (1987) with,
  - X the model matrix
  - W the diagonal weights matrix of the iterative weighted least-squares algorithm

The penalty term is composed of the determinant Fisher information of the fitted values $\hat{\mu}^0_{gj}$, i.e. $|X^t W X|$. Its purpose is to adjust for the negative bias arising from the estimation of the coefficients to obtain the fitted values (McCarthy et al., 2012).

Since the data follow a negative binomial, the elements of the diagonal of the weights matrix are given by (Love et al., 2014):

$$w_{gj} = \frac{1}{1/\hat{\mu}^0_{gj} + \alpha_g}$$

Love et al. (2014) optimize the adjusted Cox-Reid log-likelihood using backtracking line search on the scale of $\log \alpha$.

Example

The different estimates of dispersion for the example dataset are represented in Figure 2.11. It shows that the values obtained with the rough method of moments based on global variances and means ($\hat{\alpha}_{g;m}$) have a wider range than the ones based

on within-group variances and means ($\hat{\alpha}_{g;r}$). However, the latter is selected as the rough method of moments estimator ($\hat{\alpha}_{g;MoM}$) for 60% of the genes in the example dataset. Furthermore, the values of $\hat{\alpha}_{g;r}$ are in average closer to the final estimated values ($\hat{\alpha}_g$) than the values of $\hat{\alpha}_{g;MoM}$.

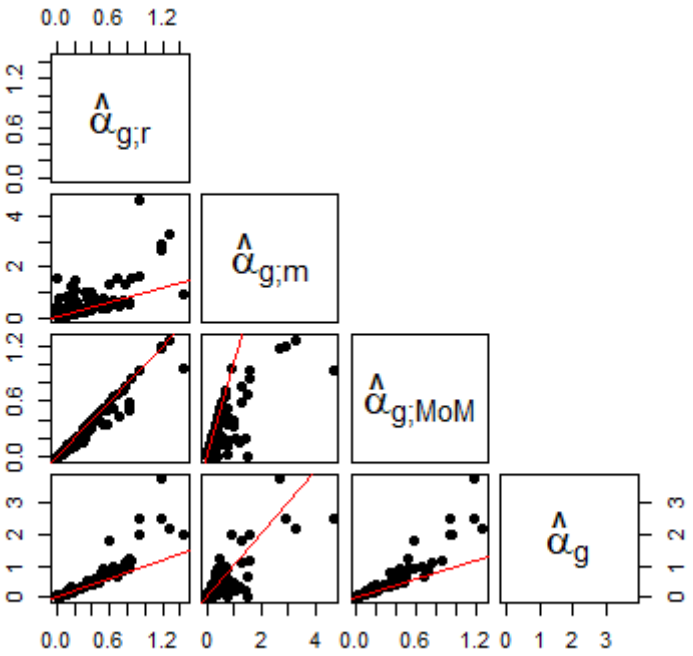


Figure 2.11: Relation between the estimated values of the dispersion for the example dataset. In the order, the rough method of moments estimators based on within-group and global variances and means, their minimum and the maximum-likelihood estimator. The red line represents the identity.

2.5.1.3 Mean-dispersion relationship estimation

The third step of the computation of the variance-stabilizing transformation is the estimation of the mean-dispersion relationship. The parameterisation of this relationship proposed by Anders et al. (2012) is as follows:

$$\alpha_{tr}(\bar{\mu}_g) = \frac{a_1}{\bar{\mu}_g} + a_0$$

Where,

- $\bar{\mu}_g$ is the mean of normalized counts for gene g
- $\alpha_{tr}(\bar{\mu}_g)$ the trend function between the dispersion and the mean of normalized counts
- a_1 and a_0 are the parameters

Since the sampling distribution of the dispersion estimates around the true values can be skewed, Anders et al. (2012) obtained the parameters by iteratively fitting a gamma-family GLM regression. Furthermore, at each iteration Anders et al. (2012) leaves out the genes with a ratio of dispersion to fitted values of dispersion outside $[10^{-4}; 15]$.

Example

The trend obtained for the example dataset is shown in Figure 2.12. The values estimated for the parameter are 0.13 for a_0 and 17.24 for a_1.

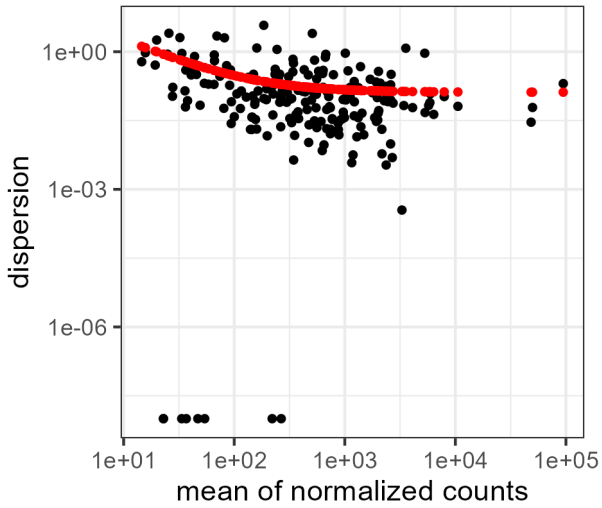


Figure 2.12: Per-gene dispersion estimate by mean of normalized counts (black) and fitted mean-dispersion relationship (red)

2.5.1.4 Variance-stabilizing transformation

The last step of the transformation of the data is the estimation of the variance-stabilizing transformation and its application to the normalized counts.

According to Anders and Huber (2010), a variance-stabilizing transformation is obtained with the following equation:

$$u(x) = \int_0^x \frac{d\bar{\mu}_g}{\sqrt{v(\bar{\mu}_g)}}$$

Where,

- $\bar{\mu}_g$ is the mean normalized count for gene g
- $v(\bar{\mu}_g)$ is the trend function between the variance and mean normalized count

The latter can be found from the mean-dispersion relationship. Indeed, for a negative binomial,

$$V(Y) = \mu + \alpha\mu^2$$

And thus,

$$v(\bar{\mu}_g) = \bar{\mu}_g + \alpha_{tr}(\bar{\mu}_g)\bar{\mu}_g^2 = \bar{\mu}_g + \bar{\mu}_g^2 a_0 + \bar{\mu}_g a_1$$

By computing the integral, Anders and Huber (2010) obtained the following expression for a general variance-stabilizing transformation:

$$u_0(x) = \frac{\log \frac{1+2a_0x+a_1+2\sqrt{a_0x(1+a_0x+a_1)}}{1+a_1}}{\sqrt{a_0}}$$

Furthermore, Anders and Huber (2010) states that if $u_0(x)$ is a variance-stabilizing transformation then $u(x) = \eta u_0(x) + \zeta$ is also a variance-stabilizing transformation.

Finally, Anders and Huber (2010) chose the parameters η and ζ so that the variance-stabilizing transformation $u(x)$ behaves like a $\log_2$ transformation for large values. The final expression for $u(x)$ is then,

$$u(x) = \frac{\log \frac{1+2a_0x+a_1+2\sqrt{a_0x(1+a_0x+a_1)}}{4a_0}}{\log 2}$$

This transformation is applied to the normalized counts to obtain the transformed counts.

Example

Figure 2.13 shows the relation between the raw reads counts and the counts obtained with the variance-stabilizing transformation of Anders and Huber (2010) for the example data. It shows that the variance-stabilizing transformation compresses more the low counts than the $\log_2$ transformation.

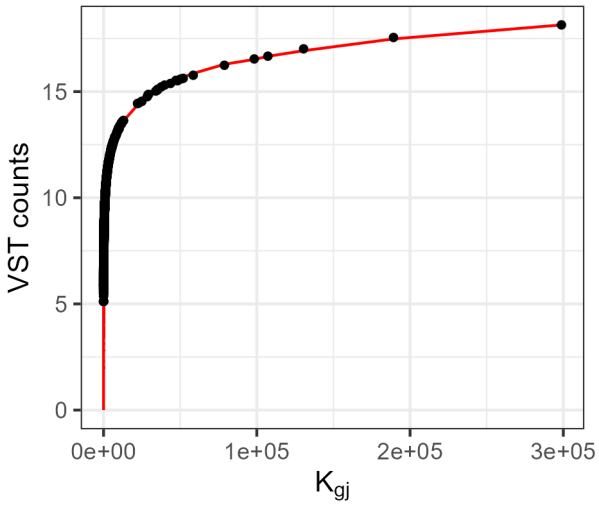


Figure 2.13: Counts transformed by a variance-stabilizing transformation (VST) in relation to raw counts. The red line corresponds to the $\log_2$-counts.

2.5.2 Model fitting

The model fitted for each gene in the *lmerSeq* workflow is a linear model with mixed effects:

$$Y_g = X\beta_g + Zb_g + \varepsilon_g$$

Where,

- Y_g is the vector of transformed counts for gene g
- X is the model matrix for fixed effects
- β_g is the vector of regression coefficients for gene g
- Z is the model matrix for random effects
- b_g is vector of random effects for gene g with
  - b_{gki} the values of the i^{th} level of the k^{th} random effect for gene g which follows a $N(0, \sigma^2_{gk})$
- ε_g is the vector of errors for gene g which follow $N(0, I\sigma^2_g)$

Vestal et al. (2022) proposed by default to fit these models using REML.

2.5.3 Contrasts testing

The contrasts are tested using the same method in *lmerSeq* and *variancePartition*, i.e. using Wald tests. Unlike for *variancePartition*, the default method for the approximation of the degrees of freedom for *lmerSeq* does not depend on the sample size and is the Satterthwaite method.

Finally, by default, the p-values are adjusted for multiple testing using the Benjamini-Hochberg method.

2.6 lmerSeq - Implementation in R

The recommended workflow for differential expression analysis using *lmerSeq* is implemented in R in four steps (see Figure 2.14):

1. Combining the count and design matrices into a DESeqDataSet using the function *DESeqDataSetFromMatrix* from the *DESeq2*
2. The transformation of the data using the function *varianceStabilizingTransformation* from the *DESeq2*
3. The fitting of the gene-wise linear mixed models using *lmerSeq.fit* from *lmerSeq*
4. The contrasts testing using *lmerSeq.contrast* from *lmerSeq*

As mentioned at the start of Section 2.4, the package was created to offer more flexibility to the users (Vestal et al., 2022). The workflow before the fitting of the models can thus be modified. For example, it is possible to use different normalization and transformation methods.

Similarly to *dream*, *lmerSeq.fit* which computes the gene-wise linear mixed model allows the use of parallelization to speed up the analysis.

Furthermore, unlike the workflow with *variancePartition*, models fitting and contrasts testing are done by two distinct functions. Moreover, the complete fitted models are in the outputs of *lmerSeq.fit* including the random effects which were not included in *dream* outputs.

In addition, if a contrast matrix is given to *lmerSeq.contrast* the F-test is automatically computed. To obtain the t-test results, the function must be called once by contrast.

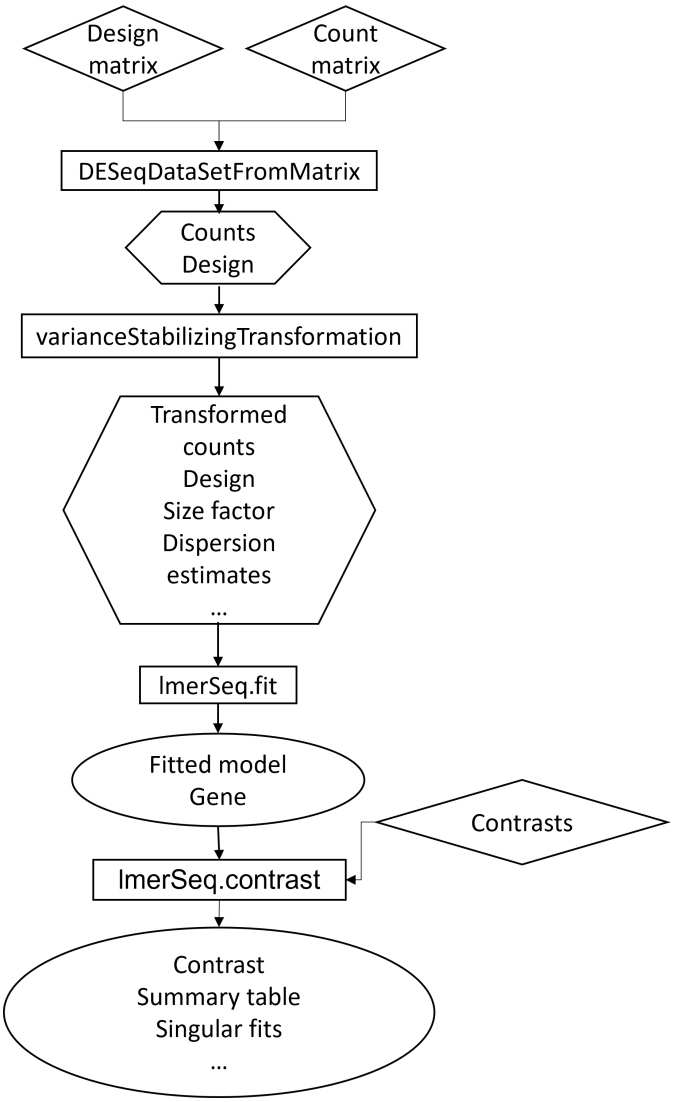


Figure 2.14: Recommended workflow in R for *lmerSeq*. The rectangles represent functions, the diamonds matrices, the hexagon DESeqDataSet objects and the ovals lists

Chapter 3

Simulation study

In this chapter, the performances of the two packages are compared using a simulation study.

3.1 Data simulation

3.1.1 Model

The model used for the simulation of the data is the same as the first scenario in Vestal et al. (2022). This first scenario corresponds to a study with 2 groups (control and treated) and paired observation for each patient (baseline and follow-up). Vestal et al. (2022) generate the simulated counts from the following negative binomial generalized linear mixed model with a random intercept:

$$K_{gij} \sim NB(\mu_{gij}, \alpha_g)$$

$$\log(\mu_{gij}) = \beta_{g0} + \beta_{g1} I_{T_i} + \beta_{g2} t_{ij} + \beta_{g3} I_{T_i} t_{ij} + b_{gi}$$

$$b_{gi} \sim N(0, \sigma^2_{bg})$$

Where,

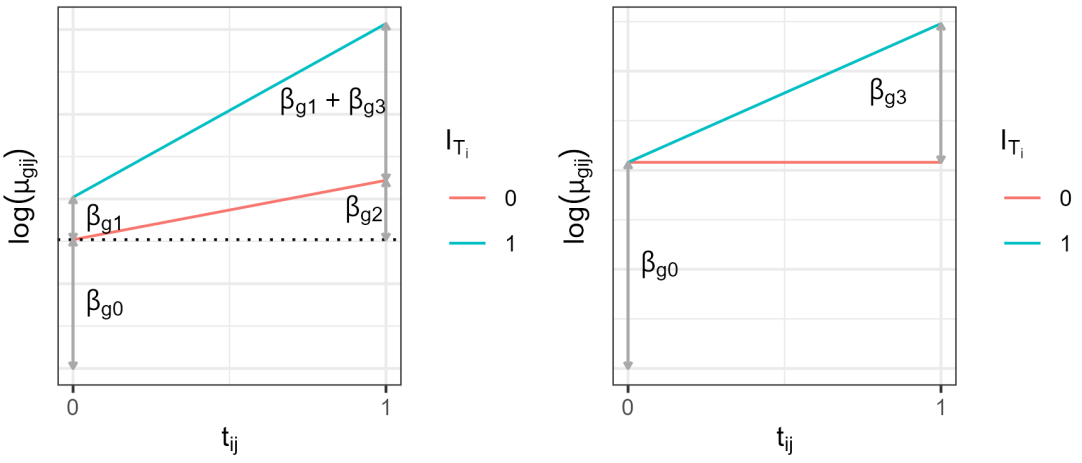
- K_{gij} is the observed count of gene g for patient i at observation j
- μ_{gij} is the expected count of gene g for patient i at observation j
- α_g is the dispersion of gene g
- $\beta_{g0}, \beta_{g1}, \beta_{g2}$ and β_{g3} are the parameters of the fixed effects for gene g

- I_{T_i} is a group indicator for patient i which is equal to 1 when the subject is in the treated group
- t_{ij} is the observation time for subject i at observation j which is equal to 0 at baseline and 1 at follow up
- b_{gi} is the random intercept for subject i and gene g
- σ^2_{bg} is the random intercept variance for gene g

The interaction plot for differentially expressed genes in this model is shown in Figure 3.1a.

3.1.2 Parameters

The values for the parameters are, as for the model, those proposed by Vestal et al. (2022). Therefore, β_{g1} and β_{g2} are set to 0. Vestal et al. (2022) estimated the triplets of β_{g0}, α_g and σ^2_{bg} from available human RNA-Seq datasets. These triplets are thus taken from the available simulated datasets of Vestal et al. (2022). Finally, β_{g3} is not equal to 0 for 20% of the genes selected randomly. The values of the non-zero β_{g3} are drawn from a gamma distribution with a mode equal to $\log 2$ and a standard deviation equal to 0.5 and the sign is randomly assigned. The interaction plot for differentially expressed genes with these parameters is shown in Figure 3.1b.



(a) Complete model

(b) Model with $\beta_{g1} = \beta_{g2} = 0$

Figure 3.1: Interaction plots for differentially expressed genes in the simulation model.

3.1.3 Datasets generation

Datasets are generated in R for 4 different sample sizes, i.e. 12, 20, 40 and 80 samples. For each sample size, 100 datasets are generated. Furthermore, each dataset included simulated raw read counts for 14 759 genes of which 2 951 are differentially expressed. The R code for the generation of the datasets is available in Appendix A.1. The simulated values of non-zero β_{g3} go from 0.01582 to 5.099 in absolute value and their distribution is shown in Figure 3.2.

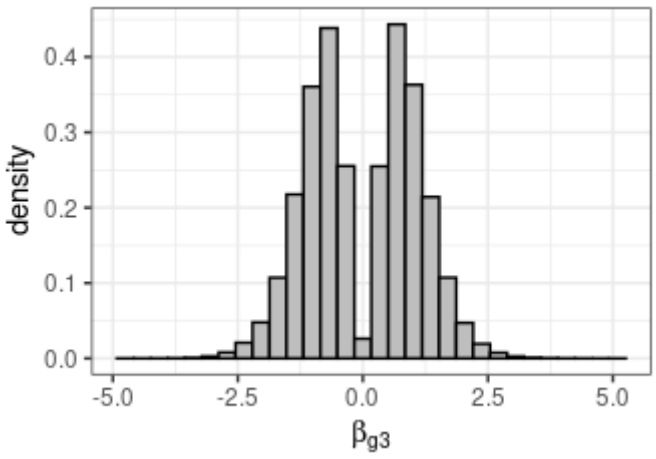


Figure 3.2: Histogram of the non-zero β_{g3} simulated

3.2 Analysis workflows

Four different workflows are used to perform the differential expression analysis on the datasets:

1. The recommended workflow for the *dream* method (dream)
2. The recommended workflow for the *dream* method with empirical Bayes statistics (dream with eBayes)
3. The recommended workflow for the *lmerSeq* package (lmerSeq)
4. The *lmerSeq* workflow with voom (lmerSeq with voom), i.e. with $\log_2$-counts per million and precision weights instead of the variance stabilizing transformation

The four workflows start with the filtering of lowly expressed genes. This filtering is done with the function *filterByExpr* from the package *edgeR*. The code for the analysis is included in Appendix A.2.

The formula of the gene-wise models fitted is the same for all workflow, i.e. :

$$Y_{gij} = \beta'_{g0} + \beta'_{g1} I_{T_i} + \beta'_{g2} t_{ij} + \beta'_{g3} I_{T_i} t_{ij} + b'_{gi} + \varepsilon_{gij}$$

Where,

- Y_{gij} is the normalized and transformed count for gene g in subject i at observation j
- $\beta'_{g0}, \beta'_{g1}, \beta'_{g2}$ and β'_{g3} are the regression coefficients of the fixed effects for gene g
- I_{T_i} is a group indicator for patient i which is equal to 1 when the subject is in the treated group
- t_{ij} is the observation time for subject i at observation j which is equal to 0 at baseline and 1 at follow-up
- b'_{gi} is the random intercept for subject i and gene g with,
  - $b'_{gi} \sim N(0, \sigma^2_{bg})$ where σ^2_{bg} is the variance of the random intercept for gene g
- ε_{gij} is the error term for gene g and subject i at observation j

In the *lmerSeq* workflow, $\varepsilon_g \sim N(0, \sigma^2_g)$ where σ^2_g is the residual variance for gene g. However, for the three other workflows which use precision weights, $\varepsilon_g \sim N(0, \text{diag}(w_g)^{-1}\sigma^2_g)$ where w_g is the vector of weights for gene g.

3.2.1 Contrasts

Three contrasts are tested separately during the simulation study. These are the same contrasts as the one tested in Vestal et al. (2022) in order to compare the results obtained with those of the article. The null hypothesis of the contrasts tested are:

1. $\beta'_{g3} = 0$ which will be referred to in the rest as interaction
2. $\beta'_{g1} + \beta'_{g3} = 0$ which is a between-subject contrast and will be referred to in the rest as between
3. $\beta'_{g2} + \beta'_{g3} = 0$ which is a within-subject contrast and will be referred to in the rest as within

3.3 Performance indicators

The performance of the four workflows will be assessed mainly using three indicators:

1. The intraclass correlation
2. The false discovery rate
3. The sensitivity

The intraclass correlation will be used to evaluate the estimation of the parameters while the other indicators will be used to compare the performances regarding contrasts testing.

3.3.1 Intraclass correlation

The aim of the first indicator is to measure the correlation between the simulated and estimated interaction parameters. Since the aim of the models is to estimate the exact values of the parameters and not simply correlated values, the Pearson correlation coefficient is not appropriate to measure the link between the values. An alternative correlation coefficient proposed in the literature is the intraclass correlation coefficient for absolute agreement. This coefficient is used to evaluate if several measurement classes give the same values to subjects (McGraw and Wong, 1996). In our case, there are two measurement classes, the simulated and estimated parameters, and 14 759 subjects, the genes.

The intraclass correlation is obtained by dividing the covariance between the values of the parameter for the same gene by the total variance of a value. Its estimation is based on an analysis of variance. In the case used in this thesis, it can be estimated with the following formula (McGraw and Wong, 1996):

$$ICC = \frac{MSS - MSE}{MSS + (k-1)MSE + \frac{k}{n}(MSC - MSE)}$$

Where,

- MSS is the mean square between subject
- MSE is the mean square error
- MSC is the mean square between classes

- k is the number of classes
- n is the number of subjects

3.3.2 False discovery rate

The second performance indicator is the false discovery rate (FDR). The FDR corresponds to the proportion of false discoveries in the tests where the null hypothesis was rejected. It's thus an estimation of the probability that a feature considered significant is truly null:

$$FDR = \frac{FP}{FP + TP} \approx P(H_0|RH_0)$$

Where,

- FP is the number of false positive
- TP is the number of true positive
- RH_0 indicates a rejection of the null hypothesis

3.3.3 Sensitivity

The third performance indicator is the sensitivity. The sensitivity is the proportion of truly differentially expressed genes that are detected. It is also an estimation of the power of a test:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \approx P(RH_0|H_1) = \text{power}$$

3.4 Results

This section presents the results of the simulation study. Unless explicitly stated, the results presented are the results of simulations with 40 samples.

3.4.1 Data transformation

The first point of interest is the comparison of the normalized and transformed counts in the two recommended workflows.

As a reminder, in the *dream* method, the raw counts are normalized using a trimmed mean of M-values normalization method and then transformed in $\log_2$ count-per-million. In the *lmerSeq* workflow, the data is normalized using the median of the ratio of the gene expression in the sample and the geometric mean of the gene expression. Afterwards, the normalized counts are transformed using a variance-stabilizing transformation. Moreover, the transformation is created so it behaves like a $\log_2$ transformation for large count values.

Figure 3.3 shows the normalized and transformed counts obtained with the *lmerSeq* workflow compared to the ones obtained with the *dream* workflow. The first noticeable feature is that for large values, the relationship between the values of the two methods is parallel to the identity. This seems to confirm that the variance-stabilizing transformation behaves like a $\log_2$ transformation for large values. There is however a difference of behavior for small counts. Indeed, the *lmerSeq* workflow transformation gives more compressed values for the low counts.

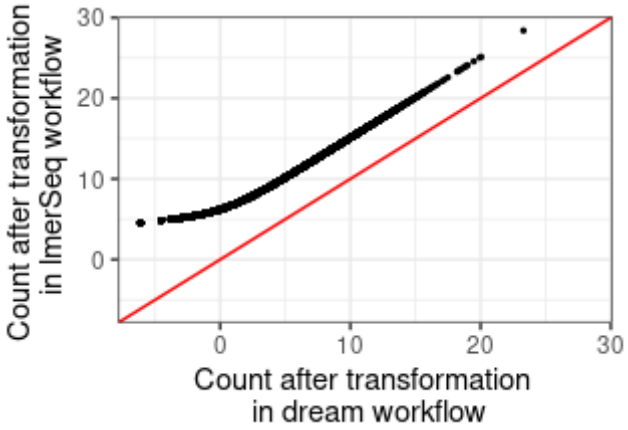


Figure 3.3: Normalized and transformed count data in the *lmerSeq* workflow by the values with the *dream* method. The red line represents the identity

Similarly, Figure 3.4 shows the boxplots of the transformed counts for four samples of the first simulation. For this figure, the raw counts of sample 11 were multiplied by 100 to simulate a manipulation error. Figure 3.4 shows, as in Figure 3.3, that the low values are more compressed with *lmerSeq* and that there is an approximately constant difference between the transformed counts of the two methods for large values. Figure 3.4 also shows that the transformed counts are closer to the values of the $\log_2$ raw counts with *lmerSeq*. The *dream* method thus has transformed counts smaller than the $\log_2$ raw counts and the transformed counts from *lmerSeq*. This comes from the fact that dream transforms counts per million, whereas the other two values are obtained by transforming counts.

Furthermore, if we look at the boxplots for sample 11 in Figure 3.4, we can see that both methods have eliminated the effect of the simulated manipulation error. Indeed, the normalised and transformed counts for this sample have similar levels to the other samples.

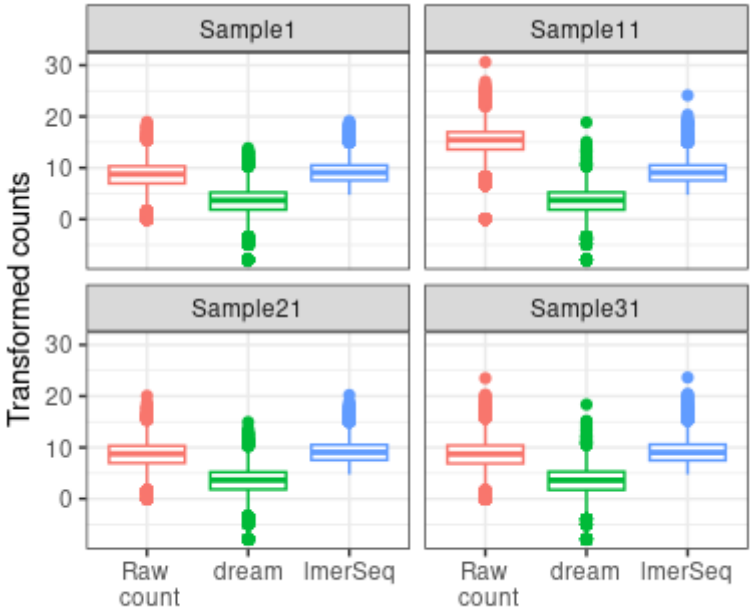


Figure 3.4: Boxplots of transformed counts for four samples of the first simulation. The raw counts are transformed in $\log_2$ counts. The sample raw counts of sample 11 are multiplied by 100 to simulated a manipulation error.

3.4.2 Parameters estimation

The second point of interest is the estimation of the parameters of the gene-wise models.

3.4.2.1 Fixed effects

First, the estimates of the fixed effects parameters will be examined, i.e. $\hat{\beta}_{g1}, \hat{\beta}_{g2}$ and $\hat{\beta}_{g3}$. Figure 3.5 shows the density of $\hat{\beta}_{g1}, \hat{\beta}_{g2}$ and $\hat{\beta}_{g3}$ centred on the values simulated for the two recommended workflow. It shows that the variance is higher for the parameters estimated with the *dream* method. Figure 3.5 also shows that the variance is different depending on the parameter. Indeed, after centring, the estimate of the interaction parameter $\hat{\beta}_{g3}$, which is the only parameter with some non-zero simulated values, varies more than the others.

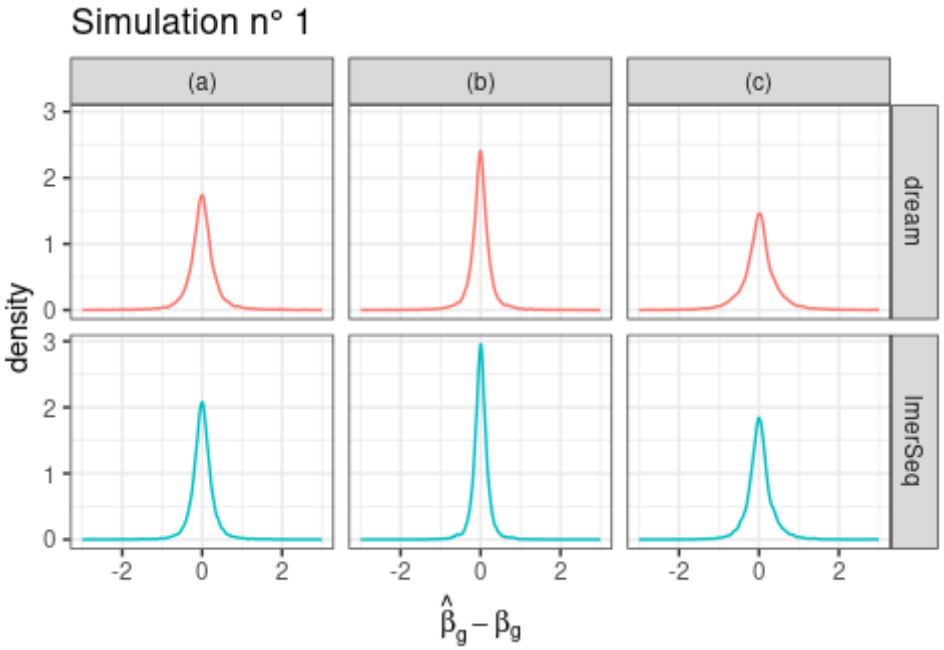


Figure 3.5: Density of the centred estimated parameters obtained with the two recommended workflows in the first simulation for (a) $\hat{\beta}_{g1}$ (b) $\hat{\beta}_{g2}$ (c) $\hat{\beta}_{g3}$

Interaction parameter

Since the interaction parameter is the only one with simulated values other than zero, we will analyse the results of these estimates in more detail.

The first measure of interest is the intraclass correlation between the simulated and estimated parameters. Figure 3.6 shows the boxplots of the values of the intraclass correlation for the different simulations. It shows that the values estimated using the recommended workflow for *lmerSeq* have a higher intraclass correlation with the values simulated than the ones obtained with the *dream* workflow.

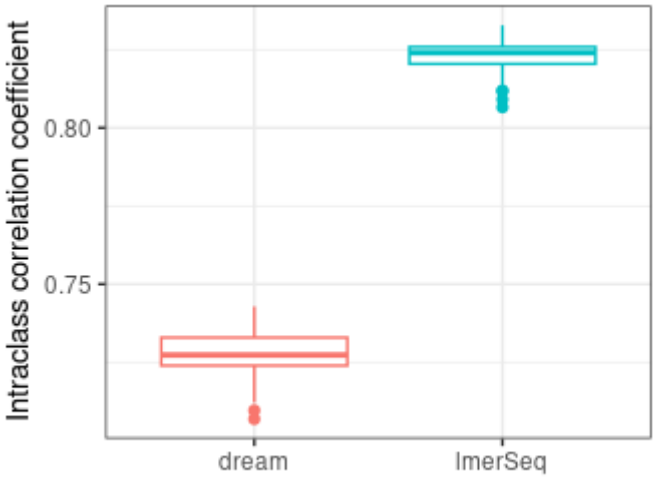


Figure 3.6: Boxplot of the intraclass correlation between the parameters estimated and simulated in the simulations

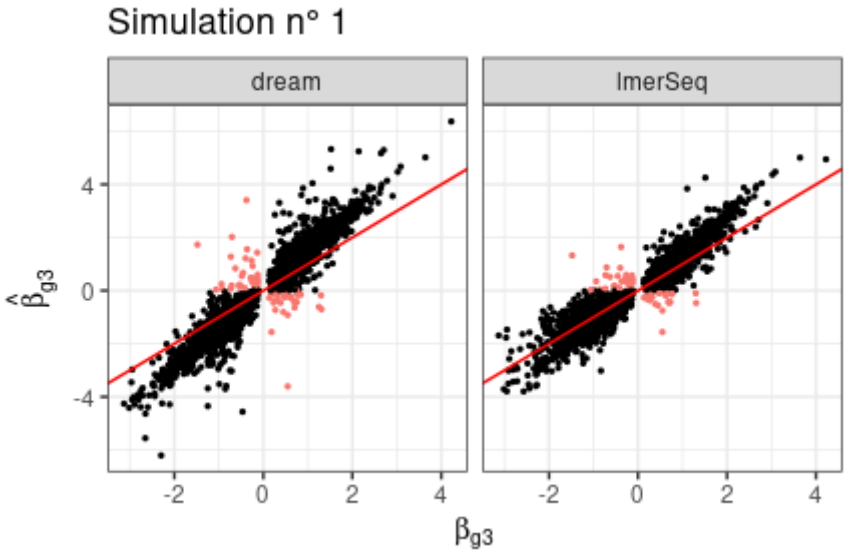
Similarly, plotting the estimated values of the interaction parameter by the simulated values (Figure 3.7a) shows that the relation is closer to the identity for the *lmerSeq* workflow. It also shows that for both recommended workflows, the estimated values move away from the simulated values as the latter increases.

However, when simulating the datasets, the log of the mean was equal to a linear function of the parameters. Whereas, the data used for modelling is transformed with a $\log_2$ function for *dream* and a function that behaves like a $\log_2$ for large values for *lmerSeq*. This means that the values simulated (β_{g3}) are not equal to the true values of the parameter in the models (β'_{g3}). While the goal of the complete workflow is to estimate β_{g3} and find for which genes this parameter is significantly different from zero, the parameters estimated in the gene-wise models are the β'_{g3}. Figure 3.7b shows the estimated values of the interaction parameters by a

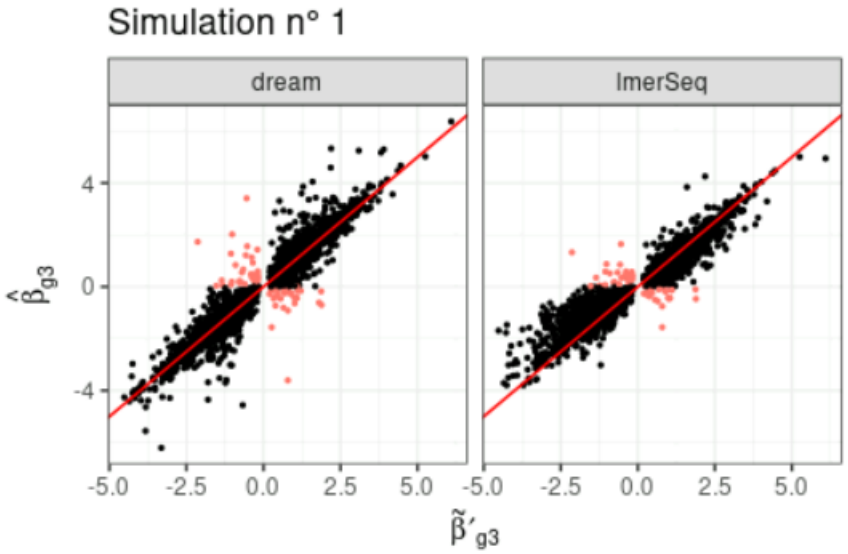
rough approximation of the corresponding β'_{g3} obtained by dividing β_{g3} by log(2). This approximation is better for the *dream* method than for *lmerSeq* since the transformation used by the latter behaves like a $\log_2$ transformation only for large values. Nevertheless, we can see that the trend observed in Figure 3.7a, of moving away from the true value as it increases, is no longer present in Figure 3.7b. This trend therefore seems to be caused simply by the change in scale.

Another point of interest is the evolution of the estimation error in relation to average gene expression. Since the aim of the analysis is to estimate the true values β_{g3} and not the transformed values β'_{g3}, the estimation error is calculated in relation to the former. Figure 3.8 shows the estimation error by average expression of the gene for each recommended workflow. It can be seen that the range of estimation error is wider for the *dream* method. Figure 3.8 therefore shows, similarly to Figure 3.5, that the variance is higher for the *dream* method.

Finally, Figure 3.9 shows the absolute value of the estimation error by average expression of the gene and the trend between the two values. It shows that for both methods, the absolute errors tend to increase when the average expression of the gene increases. However, it also shows that with the *dream* method, the absolute error is also higher for genes with low average expression.



(a) Parameter simulated (β_{g3})



(b) Parameter simulated in $\log_2$ scale ($\tilde{\beta}'_{g3}$)

Figure 3.7: Parameter estimated in function of the parameter simulated by workflow for differentially expressed genes in simulation 1. The colored points correspond to genes where the sign of the estimated parameter does not match the sign of the simulated parameter and the red line represents the identity.

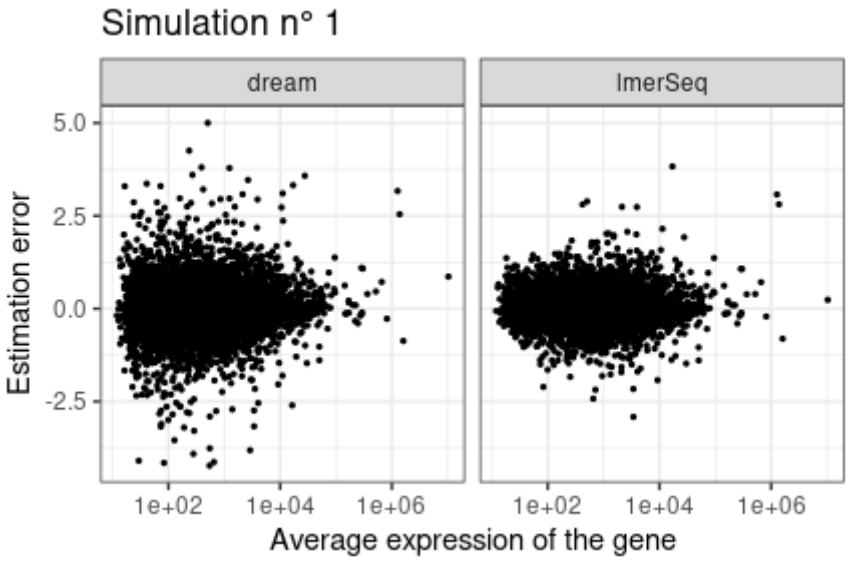


Figure 3.8: Estimation error by average expression of the gene with the two recommended workflows in simulation 1

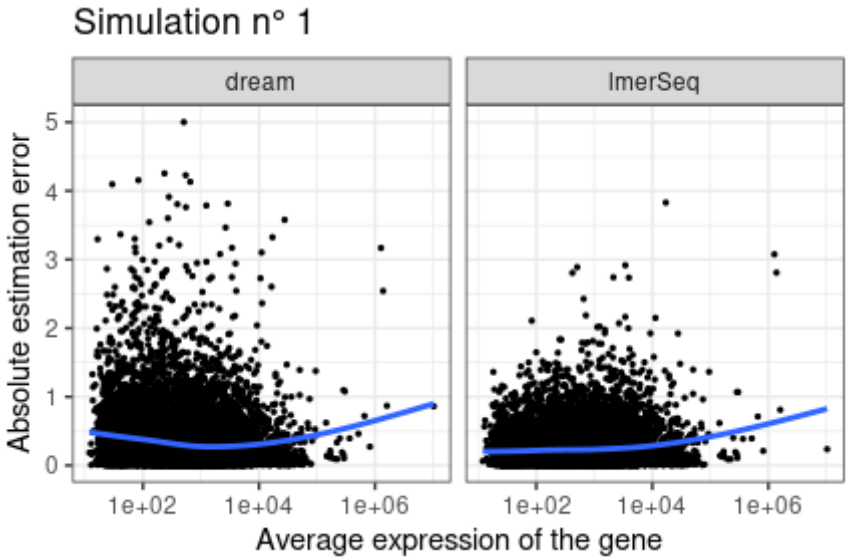


Figure 3.9: Absolute estimation error by average gene expression with the two recommended workflows in simulation 1. The blue line represents the trend

3.4.2.2 Random effect

The second category of parameters estimated is the variance of the random effects. In the model used for the simulation study, there is only one random effect: the random intercept. The estimated and simulated values of its variance are represented in Figure 3.10. It shows that the *dream* method tends to overestimate the variance of the random intercept.

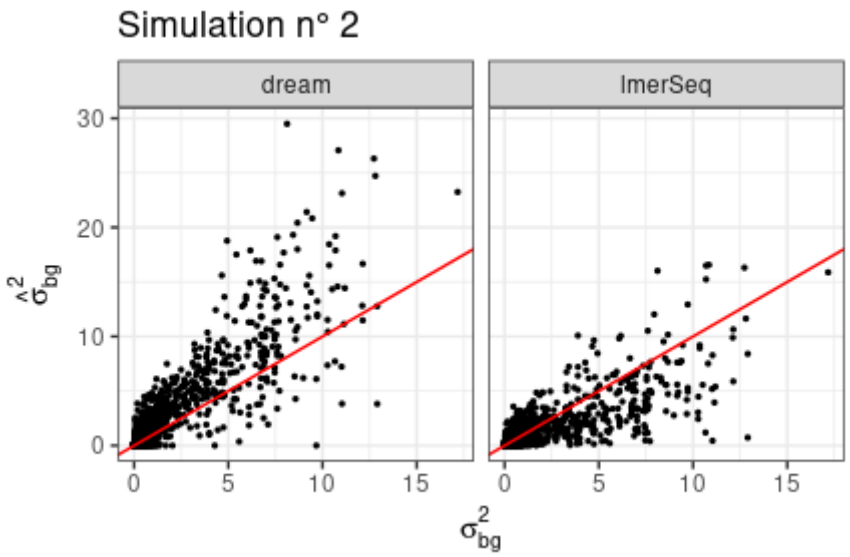


Figure 3.10: Random intercept variance estimated vs simulated for the second simulation with the two recommended methods. The red line represents the identity

3.4.3 Contrasts testing

The third point of interest is the results of the tests of the contrasts.

3.4.3.1 P-values

Firstly, Figure 3.11 shows the $-\log_{10}$ of the adjusted p-values as a function of the simulated values for the parameters for the test on the interaction. It can be seen that the plot is similar for the four different workflows. Indeed, the same points stand out, i.e. the points labelled but also the three points below the red line but with simulated values between -3 and -1.5.

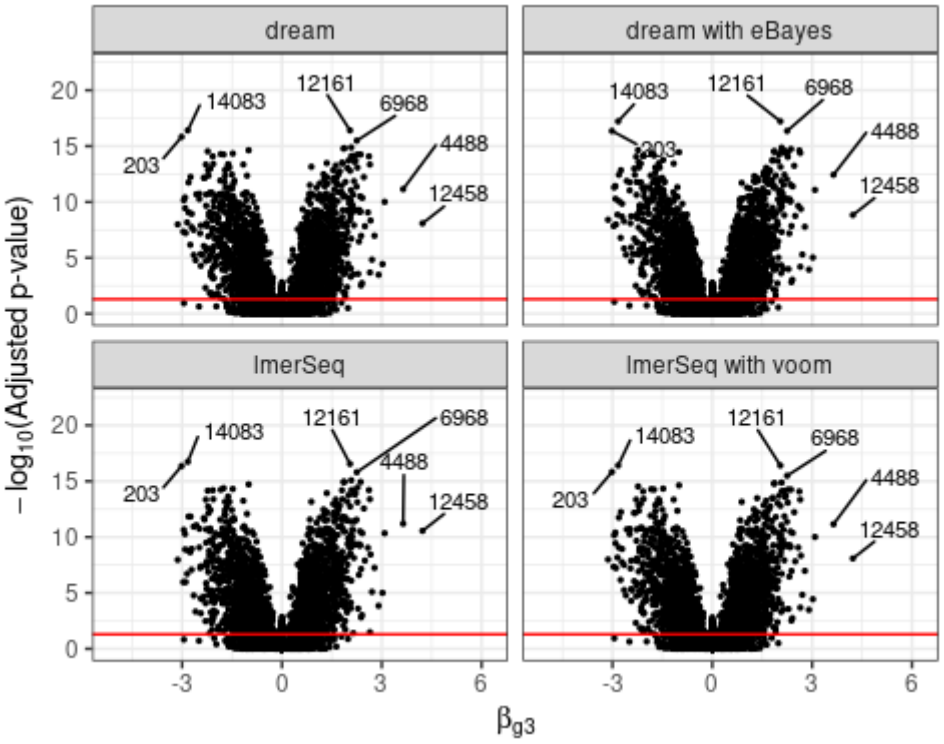


Figure 3.11: $-\log_{10}$(adjusted p-values) as a function of the simulated parameter for each workflow. The red line corresponds to a p-value of 0.05

3.4.3.2 False discovery rate

Secondly, the focus is on the false discovery rate (FDR). In this thesis, the nominal value of the FDR is fixed at 0.05. Figure 3.12 represents the boxplots of the FDR observed in the simulations with 40 samples for the three contrasts and the four workflows tested. It shows that for the majority of the simulation the FDR observed is below the nominal value for the four workflows. However, all four methods have observed values at and above the nominal value for the test on the interaction parameter and the "within" contrast.

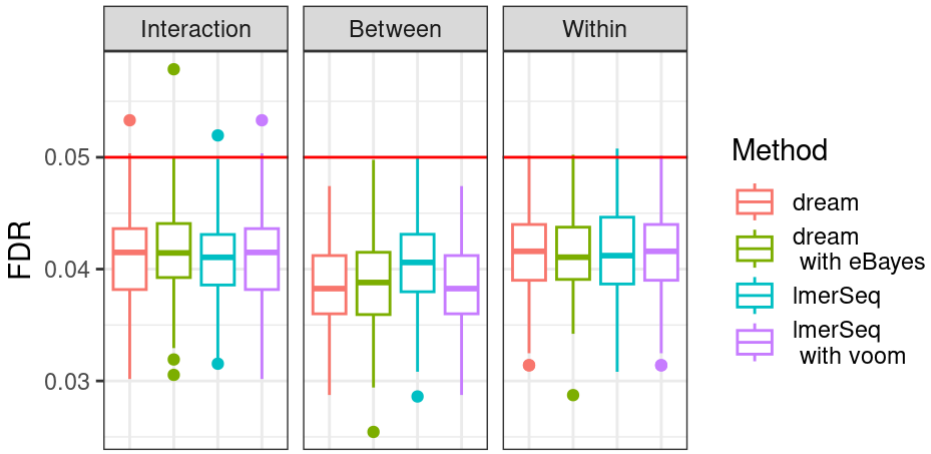


Figure 3.12: Boxplots of the FDR for the 4 tested workflows. The red line represents the nominal FDR (0.05)

3.4.3.3 Sensitivity

Thirdly, we analyse the sensitivity achieved by the different methods. The sensitivity is represented for each contrast tested in Figure 3.13. It shows that the sensitivity is higher for the "within" contrast. It also shows that the *dream* method with empirical Bayes statistics gives the highest sensitivity of the four methods tested as expected.

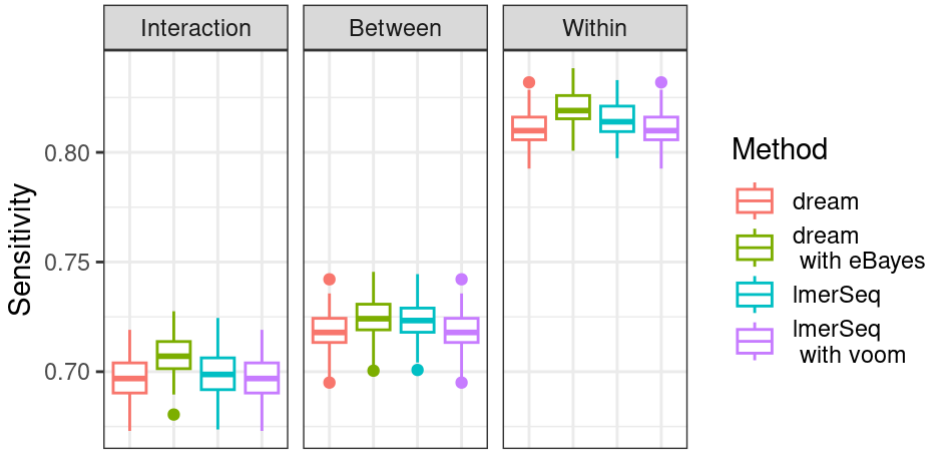


Figure 3.13: Boxplots of the sensitivity for the 4 tested workflows

3.4.3.4 Genes with errors

Fourthly, we focus on genes whose test leads to a decision error for the interaction. Figure 3.14 shows the distribution of the simulated parameters for those genes in the first simulation. It shows that the majority of the errors are made on genes with a small difference in expression. However, as shown in Figure 3.11, genes with a higher simulated parameter are sometimes not detected as differentially expressed. In this simulation, some genes with simulated values between -3 and -2 were not detected as differentially expressed.

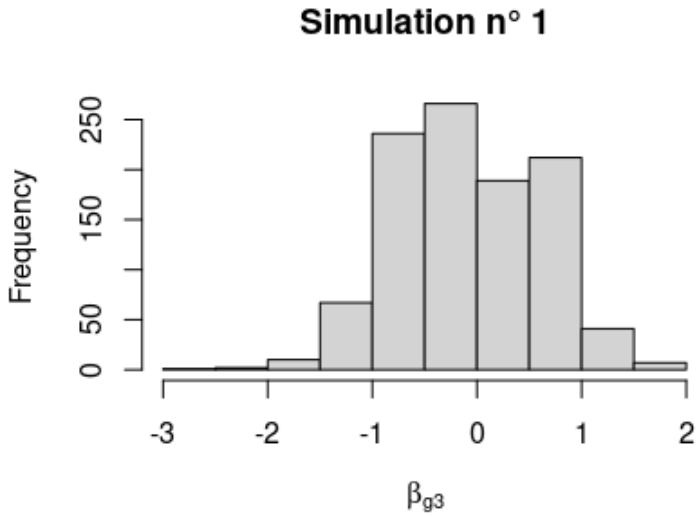
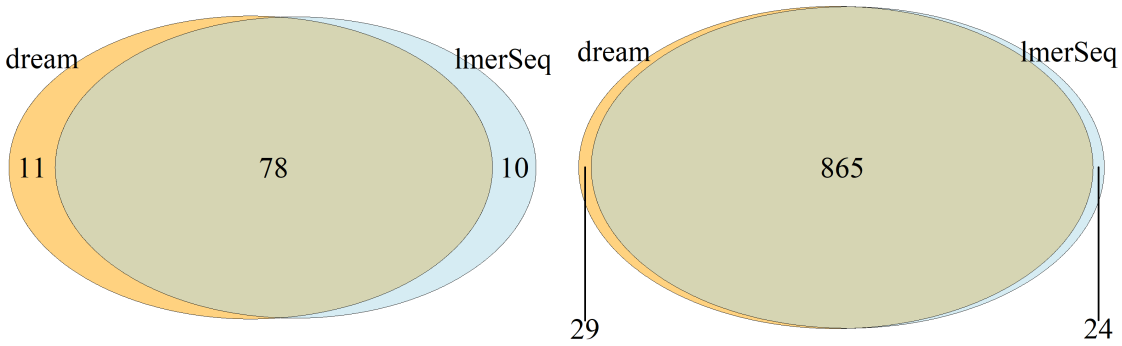


Figure 3.14: Histogram of the simulated value of the interaction for the genes with an error in the test conclusion for the first simulation

Figure 3.15 shows the average number of genes with decision error for the test on the interaction parameter for the two recommended workflows. It shows that there are more false negatives than false positives. This is to be expected since the FDR is below 5% and sensitivity is around 70%. Figure 3.15 also shows that the two methods result in the same decision for the majority of genes.



(a) False positive (b) False negative

Figure 3.15: Average number of decision errors of the test on the interaction for the two recommended workflows

3.4.3.5 Effect size impact

Another point of interest is the impact of the effect size on the sensitivity. The relation between the two is represented in Figure 3.16 for the four workflows and the three contrasts tested. It shows that the sensitivity increases as expected when the effect size increases and that the four workflows have similar results for sensitivity.

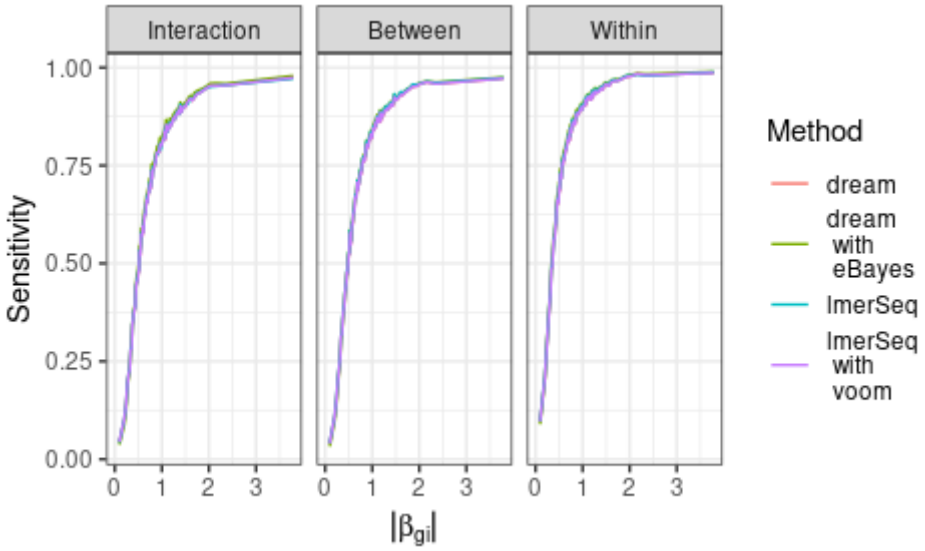


Figure 3.16: Sensitivity as a function of the absolute value of the simulated parameter.

3.4.3.6 Sample size effect

The last point of interest is the impact of the sample size on the FDR and the sensitivity. Figure 3.17 shows their values by sample size and workflow. It can be seen in Figure 3.17a that for the smallest sample size, the two workflows using the *dream* method obtained an average FDR much higher than the target value while the workflows with *lmerSeq* produced an FDR below the target and also closer.

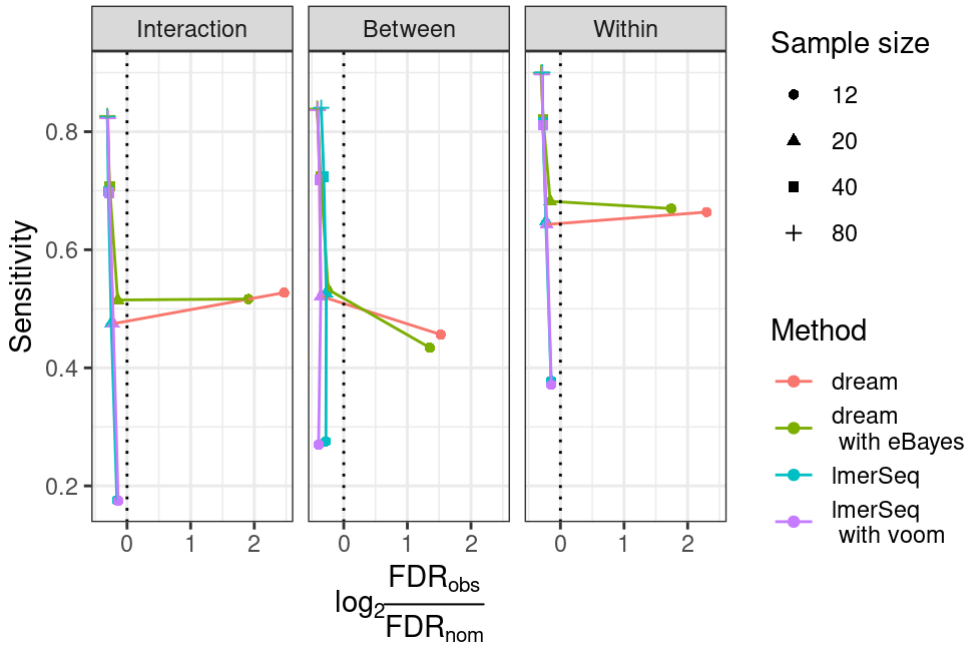
Figure 3.17b shows the results without the smallest sample size to allow a closer analysis of the difference between the workflows for the other sample sizes. The first point to note is that, for these sample sizes, the sensibility and FDR ratio are identical for the recommended *dream* workflow and the *lmerSeq* workflow with voom. The difference between the two recommended workflows seems thus to come from their different approaches to taking into account the variance-mean relationship.

A second point to note is that, for all the workflows, the sensibility increases with the sample size and that the *dream* method with empirical Bayes statistics has a higher sensitivity than the other workflows for the smallest of the remaining sample sizes. In contrast, the FDR slightly decreases with the sample size for all methods and thus diverges from the target value of 0.05.

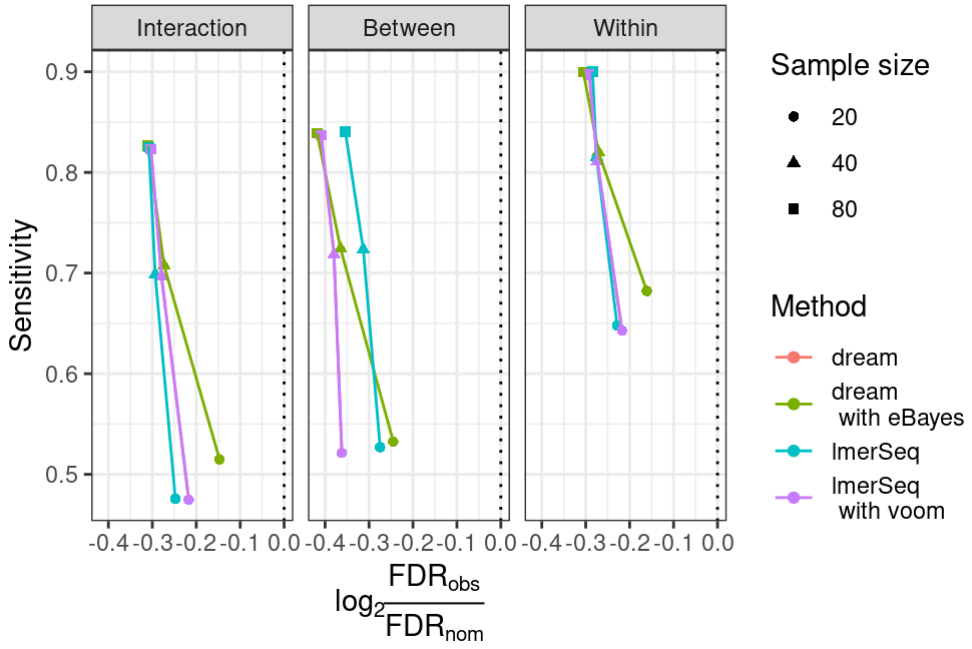
Comparing the two plots in Figure 3.17, we can make a hypothesis as to the cause of the high FDR values for the two *dream* methods at the smallest sample size. Indeed, there is no difference between the recommended *dream* workflow and *lmerSeq* with voom workflow except for the smallest sample size. The smallest sample size is also the only sample size where the two workflows do not use the same method to approximate the degrees of freedom during the contrasts testing. As a reminder, the default function parameters have been used during the analysis. This means that for *lmerSeq* the Satterthwaite approximation is always used but for dream, the approximation used depends on the sample size. More precisely, *dream* uses the Kenward-Roger approximation when the sample size is smaller than 20 which is only the case for the smallest sample size tested. The high FDR values could thus be due to the use of the Kenward-Roger approximation for the smaller sample size.

A closer look at the effect of sample size on the FDR (see Figure 3.18) reveals that the majority of observed values are below the nominal value. However, all methods have some observed values above the target value for all sample sizes. It can also be seen that as in Figure 3.17b, the results for *dream* and *lmerSeq* with voom are identical.

Finally, Figure 3.19 shows that the sensitivity increases with the sample size for the four workflows. Furthermore, it shows that the sensitivity is higher for the "within" contrast. Figure 3.19 also shows that the four workflows achieved a similar sensitivity with 40 and 80 samples. However, the *dream* method with empirical Bayes statistics has a higher sensitivity for 20 samples.



(a) With $n = 12$



(b) Without $n = 12$

Figure 3.17: Sensitivity and $\log_2$ of the ratio between the FDR observed and nominal by sample size and workflow. The dotted line corresponds to a FDR observed equal to the FDR nominal (0.05)

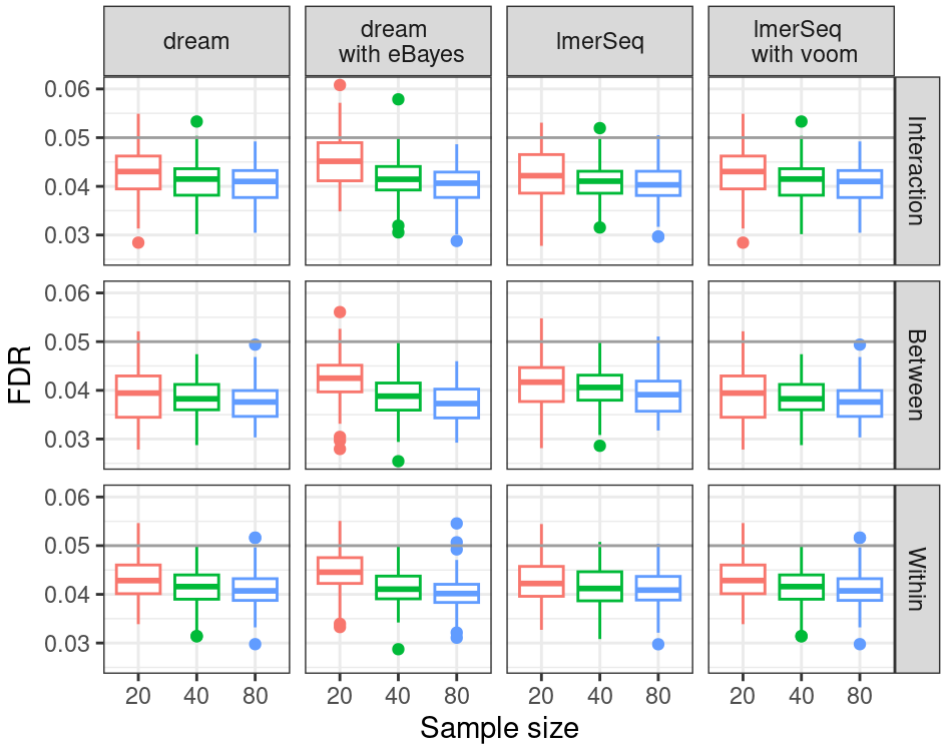


Figure 3.18: Boxplots of the FDR for each workflow and method and the three higher sample sizes tested. The line represents the nominal value

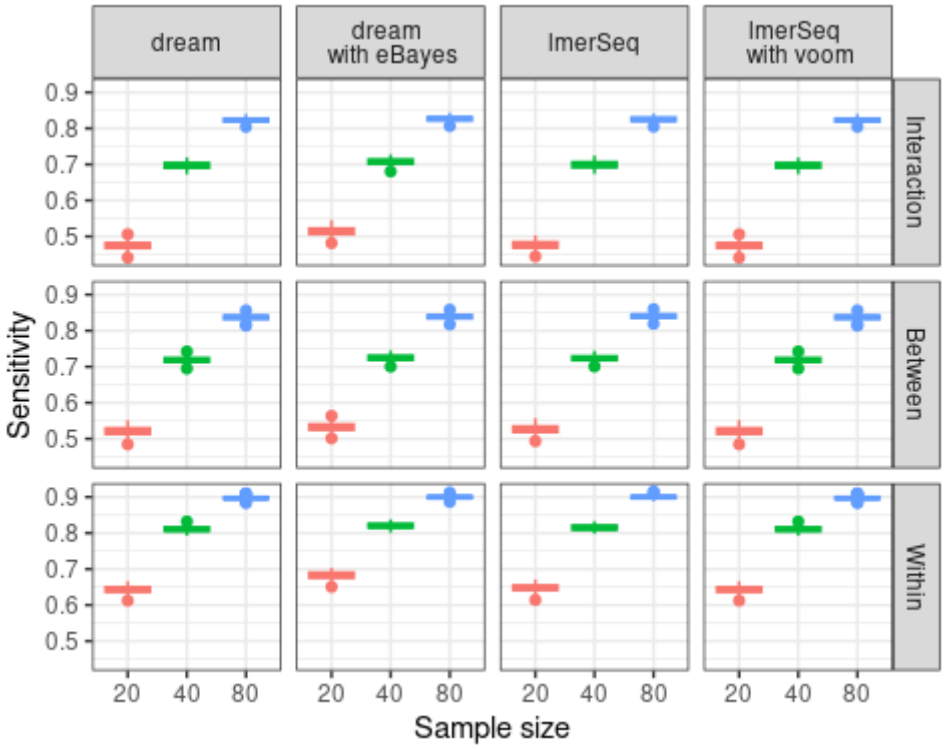


Figure 3.19: Boxplots of the sensitivity for each workflow and method and the three higher sample sizes tested. The line represents the nominal value

3.5 Conclusion

The results of the simulation study show that the difference in performance between the two recommended workflows of the packages is due to their different solutions to the problem of the relation between the mean and the variance in the original data.

The first difference in performance concerns the estimation of model parameters. Simulations have shown that estimated parameter values have a higher variance with the *dream* method. In addition, the *dream* method tends to overestimate the variance of the random intercept in the simulations.

Secondly, in the article from Vestal et al. (2022) on which this thesis is based, there was a significant difference in performance between *lmerSeq* and *dream*. In particular, *dream* had an FDR above the nominal value for all sample sizes, and this was as much as 5 times higher for the largest sample size. This trend was not observed in this thesis. This may be because normalization was not performed for the analysis with *dream* in the article but was performed in this thesis. The results in this thesis show that the two methods have similar performances when they use the same method of approximation of the degrees of freedom for the statistical tests.

Finally, the two recommended workflows have similar performance in terms of sensitivity. However, the *dream* method with empirical Bayesian statistics has a higher sensitivity, although the difference with the recommended methods decreases with increasing sample size.

Conclusions and perspectives

RNA sequencing is a technology used in transcriptomics. It allows to survey the entire transcriptome without prior knowledge of the sequences. A common use of this technology is to identify genes which are differentially expressed between different conditions (Kukurba and Montgomery, 2015). However, experimental designs with dependence between observations are increasingly used and the most popular R packages for differential expression analysis rely on models that are not appropriate for these data. Generally, the approaches proposed for this type of experiment are either using generalized linear mixed models or linear mixed models on transformed data (Vestal et al., 2022).

The aim of this thesis is the comparison of two packages available on R for the differential analysis of RNA-Seq data with linear mixed models, *lmerSeq* and *variancePartition*.

The first step was to document the workflows used by the two packages to perform this analysis. This documentation revealed that the main difference was how the link between variance and mean in the data was handled. Indeed, the package *variancePartition* uses precision weights to take it into account while *lmerSeq* removes the relationship by using a variance-stabilizing transformation.

These two methods have been developed and applied for the analysis of RNA-Seq data with linear models. The precision weight method was subsequently adapted for linear mixed models by Hoffman and Roussos (2021) for the *variancePartition* package. However, the method used in *lmerSeq* was not adapted for linear mixed models. As a result, the observations are considered independent when calculating the transformation, although they are not.

In the implementation of the methods, the package *lmerSeq* outputs the models estimated in their entirety which is not the case of *variancePartition*. This allows among other things to have the estimated values for the variance of random effects if they are of interest. However, this consumes memory which is the reason why *variancePartition* does not offer this opportunity.

It is important to note, however, that while *variancePartition* is frequently updated, at the time of this thesis the *lmerSeq* package has not been updated since it was released. This prevents some functions offered by the package from being used due to incompatibility with updated versions of packages on which it depends.

The second step of the comparison was a simulation study. It firstly showed that the variance of the estimator of the fixed parameters is higher with *variancePartition*.

However, unlike what was shown in Vestal et al. (2022), the simulation study also shows that the performance of the two packages is similar for sample sizes greater than 20. The difference with the article could come from the addition of the normalization step of *variancePartition* in the workflows used in this thesis.

Furthermore, the simulations show that the adaptation of the empirical Bayes statistics to linear mixed models proposed Hoffman et al. (2023) improved the sensitivity.

However, the false discovery rate for small sample sizes is more than two times the target value when the recommended parameters are used for *variancePartition*. This seems to be because, for those sample sizes, the degrees of freedom for the tests are approximated with the Kenward-Rogers method. However, a deeper analysis of this problem should be done to understand the cause.

In conclusion, the packages used for the differential expression analysis of RNA-Seq data with linear mixed models should depend on the goals of the researchers.

If the exact values of the parameters of the models are of interest, the variance of the estimators is smaller with the recommended workflow of *lmerSeq* and the variance of the random effects is also directly available. However, it should be kept in mind that the assumptions made during the computation of the transformation are not true for the data.

If the main goal of the study is to identify differentially expressed genes, the *variancePartition* package with the recent addition of empirical Bayes statistics allows a higher sensibility while still controlling the FDR for sample sizes greater than 20.

Bibliography

Anders, S. and Huber, W. (2010), 'Differential expression analysis for sequence count data', *Genome Biology* **11**(10), R106.

Anders, S., Reyes, A. and Huber, W. (2012), 'Detecting differential usage of exons from RNA-seq data', *Genome Research* **22**(10), 2008.

Cameron, A. C. and Trivedi, P. K. (2013), Basic Count Regression, *in* 'Regression Analysis of Count Data', Econometric Society Monographs, Cambridge University Press, pp. 69–110.

Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., Szcześniak, M. W., Gaffney, D. J., Elo, L. L., Zhang, X. and Mortazavi, A. (2016), 'A survey of best practices for RNA-seq data analysis', *Genome Biology* **17**(1), 13.

Cox, D. R. and Reid, N. (1987), 'Parameter Orthogonality and Approximate Conditional Inference', *Journal of the Royal Statistical Society. Series B (Methodological)* **49**(1), 1–39.

Dunn, P. K. and Smyth, G. K. (2018), Chapter 10: Models for Counts: Poisson and Negative Binomial GLMs, *in* P. K. Dunn and G. K. Smyth, eds, 'Generalized Linear Models With Examples in R', Springer Texts in Statistics, Springer New York, New York, NY, pp. 371–424.

Hoffman, G. E., Lee, D., Bendl, J., Fnu, P., Hong, A., Casey, C., Alvia, M., Shao, Z., Argyriou, S., Therrien, K., Venkatesh, S., Voloudakis, G., Haroutunian, V., Fullard, J. F. and Roussos, P. (2023), 'Efficient differential expression analysis of large-scale single cell transcriptomics data using dreamlet', *bioRxiv* p. 2023.03.17.533005.

Hoffman, G. E. and Roussos, P. (2021), 'Dream: powerful differential expression analysis for repeated measures designs', *Bioinformatics* **37**(2), 192–201.

Hoffman, G. E. and Schadt, E. E. (2016), 'variancePartition: interpreting drivers of variation in complex gene expression studies', *BMC Bioinformatics* **17**(1), 483.

Kukurba, K. R. and Montgomery, S. B. (2015), 'RNA Sequencing and Analysis', *Cold Spring Harbor Protocols* **2015**(11), pdb.top084970. Publisher: Cold Spring Harbor Laboratory Press.

Law, C. W., Chen, Y., Shi, W. and Smyth, G. K. (2014), 'voom: precision weights unlock linear model analysis tools for RNA-seq read counts', *Genome Biology* **15**(2), R29.

Love, M. I., Huber, W. and Anders, S. (2014), 'Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2', *Genome Biology* **15**(12), 550.

McCarthy, D. J., Chen, Y. and Smyth, G. K. (2012), 'Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation', *Nucleic Acids Research* **40**(10), 4288–4297.

McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, Springer US, Boston, MA.

McGraw, K. O. and Wong, S. P. (1996), 'Forming inferences about some intraclass correlation coefficients', *Psychological Methods* **1**(1), 30–46. Place: US Publisher: American Psychological Association.

Phipson, B., Lee, S., Majewski, I. J., Alexander, W. S. and Smyth, G. K. (2016), 'Robust hyperparameter estimation protects against hypervariable genes and improves power to detect differential expression', *The annals of applied statistics* **10**(2), 946–963.

Robinson, M. D. and Oshlack, A. (2010), 'A scaling normalization method for differential expression analysis of RNA-seq data', *Genome Biology* **11**(3), R25.

Smyth, G. K. (2004), 'Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments', *Statistical Applications in Genetics and Molecular Biology* **3**(1).

Verbeke, G. and Molenberghs, G. (2000), *Linear Mixed Models for Longitudinal Data*, Springer Series in Statistics, Springer, New York, NY.

Vestal, B. E., Wynn, E. and Moore, C. M. (2022), 'lmerSeq: an R package for analyzing transformed RNA-Seq data with linear mixed effects models', *BMC Bioinformatics* **23**(1), 489.

Wang, Z., Gerstein, M. and Snyder, M. (2009), 'RNA-Seq: a revolutionary tool for transcriptomics', *Nature reviews. Genetics* **10**(1), 57–63.

Appendix A

R code

A.1 Dataset simulation

```
sim_data <- readRDS("~/Memoire/Data_article/sim_data_ROC_MIX_n_10_1.RDS") #Simulated data article to recover the parameters

for(n in c(6,10,20,40)){
  m <- nrow(sim_data$counts) #Number of gene

  l_t <- 2 #Number of level time factor
  l_g <- 2 #Number of group

  sample <- data.frame(id = as.factor(rep(1:n,l_t)),
                       time = as.factor(rep(0:(l_t-1),each=n)),
                       groups = as.factor(rep(rep(0:(l_g-1),each=n/l_g),l_t))) #Design matrix

  for(i in 1:100){
    set.seed(2023*i)

    #Random selection of the triplets of parameters
    tp_id <- sample(1:length(sim_data$param[,"beta0s"]),m,replace=TRUE)
    B0 <- sim_data$param[tp_id,"beta0s"]
    a <- sim_data$param[tp_id,"dispersion"]
    ri_var <- sim_data$param[tp_id, "ri_var"]
```

```
21
22      # Generation of Bg3
23      pos <- sample(1:m,0.2*m)
24
25      mod <- log(2)
26      sd <- 0.5
27
28      rate <- (mod + sqrt(mod^2 + 4 * sd^2)) / (2 * sd^2)
29      shape <- 1 + mod * rate
30
31      B3 <- rgamma(m,shape,rate)
32      B3 <- B3 * sample(c(-1,1),m,replace = TRUE)
33
34      B3[-pos] <- 0
35
36      param <- list(B0=B0,a=a,ri_var=ri_var,B3=B3)
37
38      # Random intercept values
39      bgi <- matrix(nrow = m, ncol = n)
40      for(p in 1:n){
41        bgi[,p] <- rnorm(m, 0, sqrt(ri_var))
42      }
43      bgi <- cbind(bgi,bgi)
44
45      # Raw counts
46      C <- matrix(nrow = m, ncol = 2*n)
47      for (j in 1:(2*n)) {
48        l_mu <- B0 + B3 * (as.numeric(sample$time[j])-1) * (as
      .numeric(sample$groups[j])-1) + bgi[,j]
49        C[, j] <- as.integer(rnbinom(m, mu = exp(l_mu), size =
      1/a))
50        C[is.na(C[,j]),j] <- .Machine$integer.max
51      }
52
53      sim <- list(count = C, sample = sample, pos = pos,
     parameter=param,bgi=bgi[,1:n])
54      saveRDS(sim, paste("Sim_data_n",n,"/sim100_", i, ".rds",
      sep = ""))
55    }
56 }
```

A.2 Data analysis

```
library(lmerSeq)
library(DESeq2)
library(dplyr)
library(BiocParallel)
library(openxlsx)
library(edgeR)
library(variancePartition)
require(lme4)

for (n in c(6, 10, 20, 40)) {
  # dream and dream with eBayes
  rm(list = ls(all = TRUE)[sapply(mget(ls(all = TRUE)),
   class) == "list"])

  res <- list()
  res_eb <- list()

  for (i in 1:100) { # i <- 5
    data <- readRDS(paste("Sim_data_n", n, "/sim100_", i, ".
   rds", sep = ""))
    C <- data$count
    sample <- data$sample

    start_time <- Sys.time()

    # Filtering

    isexpr <- filterByExpr(C, sample)
    C <- C[isexpr, ]

    #Normalization
    dge <- DGEList(C)
    dge <- calcNormFactors(dge)

    param <- SnowParam(4, "SOCK", progressbar = TRUE)

    #Voom
    expr_dream <- voomWithDreamWeights(
      counts = dge,
      formula = ~ groups * time + (1 | id),
      data = sample,
```

```
    BPPARAM = param
  )

  #Model fitting for random effects
  L_mat <- cbind(
    c(0, 0, 0, 1),
    c(0, 1, 0, 1),
    c(0, 0, 1, 1)
  )

  vpm_dream <- fitVarPartModel(expr_dream,
    formula = ~ groups * time + (1 | id),
    data = sample,
    REML = TRUE,
    BPPARAM = param
  )

  rd_u <- lapply(vpm_dream, function(x) x@u) #random
effects
  rd_u <- matrix(unlist(rd_u), ncol = n, byrow = TRUE)

  fit_idv <- lapply(vpm_dream, function(x) VarCorr(x)[["id
"]])#variance

  rm(vpm_dream)

  #Model fitting and contrasts testing
  res_dream_cont <- dream(
    exprObj = expr_dream,
    formula = ~ groups * time + (1 | id),
    L = L_mat,
    data = sample,
    BPPARAM = param
  )
  rm(expr_dream)

  dream_coef <- topTable(res_dream_cont,
    number = "Inf",
    sort.by = "none"
  )

  tt_int_dream <- topTable(
    fit = res_dream_cont,
```

```
81        coef = "L1",
82        number = Inf,
83        sort.by = "none"
84      )
85
86      tt_btw_dream <- topTable(
87        fit = res_dream_cont,
88        coef = "L2",
89        number = Inf,
90        sort.by = "none"
91      )
92
93      tt_wtn_dream <- topTable(
94        fit = res_dream_cont,
95        coef = "L3",
96        number = Inf,
97        sort.by = "none"
98      )
99
100     end_time <- Sys.time()
101
102     cont <- list(int = tt_int_dream, btw = tt_btw_dream, wtn
        = tt_wtn_dream)
103
104     res[[i]] <- list(
105       data = data, summary_int = dream_coef, cont = cont,
106       rd_u = rd_u, ri_v = unlist(fit_idv),
107       time = end_time - start_time, isexpr
108     )
109
110     rm(tt_int_dream)
111     rm(tt_btw_dream)
112     rm(tt_wtn_dream)
113     # with eBayes
114     res_dream_cont <- eBayes(res_dream_cont)
115
116     dream_coef <- topTable(res_dream_cont,
117       number = "Inf",
118       sort.by = "none"
119     )
120
121     tt_int_dream <- topTable(
122       fit = res_dream_cont,
```

```
123       coef = "L1",
124       number = Inf,
125       sort.by = "none"
126     )
127
128     tt_btw_dream <- topTable(
129       fit = res_dream_cont,
130       coef = "L2",
131       number = Inf,
132       sort.by = "none"
133     )
134
135     tt_wtn_dream <- topTable(
136       fit = res_dream_cont,
137       coef = "L3",
138       number = Inf,
139       sort.by = "none"
140     )
141
142     end_time <- Sys.time()
143
144     rm(res_dream_cont)
145
146     cont <- list(int = tt_int_dream, btw = tt_btw_dream, wtn
      = tt_wtn_dream)
147
148     res_eb[[i]] <- list(
149       data = data, summary_int = dream_coef, cont = cont,
150       rd_u = rd_u, ri_v = unlist(fit_idv),
151       time = end_time - start_time, isexpr
152     )
153
154     rm(tt_int_dream)
155     rm(tt_btw_dream)
156     rm(tt_wtn_dream)
157   }
158
159   saveRDS(res, paste("res_Sim100_dream_", n, ".rds", sep = "
      "))
160
161   saveRDS(res_eb, paste("res_Sim100_dream_", n, "eb.rds",
      sep = ""))
162
```

```
163
164   # lmerSeq
165
166   rm(list = ls(all = TRUE)[sapply(mget(ls(all = TRUE)),
       class) == "list"])
167
168   res <- list()
169
170   for (i in 1:100) {
171     data <- readRDS(paste("Sim_data_n", n, "/sim100_", i, ".
      rds", sep = ""))
172     C <- data$count
173     sample <- data$sample
174
175     start_time <- Sys.time()
176
177     #Filtering
178     isexpr <- filterByExpr(C, sample)
179     C <- C[isexpr, ]
180
181     #Transformation
182     dds <- DESeqDataSetFromMatrix(
183       countData = C,
184       colData = sample,
185       design = ~ groups * time
186     )
187
188     dds_vst <- varianceStabilizingTransformation(dds,
189       blind = F,
190       fitType = "parametric"
191     )
192
193     rm(dds)
194     expr_vst <- assay(dds_vst)
195     rm(dds_vst)
196
197     #Model fitting
198     res_vst <- lmerSeq.fit(
199       form = ~ groups * time + (1 | id),
200       expr_mat = expr_vst,
201       sample_data = sample,
202       parallel = T,
203       cores = 4
```

```
204     )
205
206     rd_u <- lapply(res_vst, function(x) x$fit@u) #random
    effects
207     rd_u <- matrix(unlist(rd_u), ncol = n, byrow = TRUE)
208
209     fit_idv <- lapply(res_vst, function(x) VarCorr(x$fit)[["
    id"]]) #random intercept variance
210
211     vst_sum <- lmerSeq.summary(res_vst, coefficient = 4,
    sort_results = F) #interaction parameter
212
213     rm(expr_vst)
214     tt_int_vst <- lmerSeq.contrast(
215       lmerSeq_results = res_vst,
216       contrast_mat = rbind(c(0, 0, 0, 1)),
217       p_adj_method = "BH",
218       sort_results = F,
219       include_singular = T
220     )
221
222     tt_btw_vst <- lmerSeq.contrast(
223       lmerSeq_results = res_vst,
224       contrast_mat = rbind(c(0, 1, 0, 1)),
225       p_adj_method = "BH",
226       sort_results = F,
227       include_singular = T
228     )
229
230     tt_wtn_vst <- lmerSeq.contrast(
231       lmerSeq_results = res_vst,
232       contrast_mat = rbind(c(0, 0, 1, 1)),
233       p_adj_method = "BH",
234       sort_results = F,
235       include_singular = T
236     )
237
238     end_time <- Sys.time()
239
240     rm(res_vst)
241
242     cont <- list(int = tt_int_vst, btw = tt_btw_vst, wtn =
    tt_wtn_vst)
```

```
243
244     res[[i]] <- list(
245       data = data, summary_int = vst_sum, cont = cont,
246       rd_u = rd_u, ri_v = unlist(fit_idv),
247       time = end_time - start_time, isexpr
248     )
249
250     rm(tt_int_vst)
251     rm(tt_btw_vst)
252     rm(tt_wtn_vst)
253     rm(vst_sum)
254   }
255
256   saveRDS(res, paste("res_Sim100_lmerSeq_RI_", n, ".rds",
       sep = ""))
257
258   # lmerSeq with voom
259
260   rm(list = ls(all = TRUE)[sapply(mget(ls(all = TRUE)),
       class) == "list"])
261
262   res <- list()
263
264   for (i in 1:100) {
265     data <- readRDS(paste("Sim_data_n", n, "/sim100_", i, ".
       rds", sep = ""))
266     C <- data$count
267     sample <- data$sample
268
269     start_time <- Sys.time()
270
271     isexpr <- filterByExpr(C, sample)
272     C <- C[isexpr, ]
273
274     dge <- DGEList(C)
275     dge <- calcNormFactors(dge)
276
277     param <- SnowParam(4, "SOCK", progressbar = TRUE)
278
279     expr_vst <- voomWithDreamWeights(
280       counts = dge,
281       formula = ~ groups * time + (1 | id),
282       data = sample,
```

```
283       BPPARAM = param
284     )
285
286     rm(dge)
287     res_vst <- lmerSeq.fit(
288       form = ~ groups * time + (1 | id),
289       expr_mat = expr_vst$E,
290       weights = expr_vst$weights,
291       sample_data = sample,
292       parallel = T,
293       cores = 4
294     )
295
296     rd_u <- lapply(res_vst, function(x) {
297       if (is.null(x$fit)) {
298         rep(NA, n)
299       } else {
300         x$fit@u
301       }
302     })
303     rd_u <- matrix(unlist(rd_u), ncol = n, byrow = TRUE)
304
305     fit_idv <- lapply(res_vst, function(x) {
306       if (is.null(x$fit)) {
307         NA
308       } else {
309         VarCorr(x$fit)[["id"]]
310       }
311     })
312
313     vst_sum <- lmerSeq.summary(res_vst, coefficient = 4,
      sort_results = F)
314
315     rm(expr_vst)
316     tt_int_vst <- lmerSeq.contrast(
317       lmerSeq_results = res_vst,
318       contrast_mat = rbind(c(0, 0, 0, 1)),
319       p_adj_method = "BH",
320       sort_results = F,
321       include_singular = T
322     )
323
324     tt_btw_vst <- lmerSeq.contrast(
```

```
325       lmerSeq_results = res_vst,
326       contrast_mat = rbind(c(0, 1, 0, 1)),
327       p_adj_method = "BH",
328       sort_results = F,
329       include_singular = T
330     )
331
332     tt_wtn_vst <- lmerSeq.contrast(
333       lmerSeq_results = res_vst,
334       contrast_mat = rbind(c(0, 0, 1, 1)),
335       p_adj_method = "BH",
336       sort_results = F,
337       include_singular = T
338     )
339
340     end_time <- Sys.time()
341
342     rm(res_vst)
343
344     cont <- list(int = tt_int_vst, btw = tt_btw_vst, wtn =
      tt_wtn_vst)
345
346     res[[i]] <- list(
347       data = data, summary_int = vst_sum, cont = cont,
348       rd_u = rd_u, ri_v = unlist(fit_idv),
349       time = end_time - start_time, isexpr
350     )
351
352     rm(tt_int_vst)
353     rm(tt_btw_vst)
354     rm(tt_wtn_vst)
355     rm(vst_sum)
356   }
357
358   saveRDS(res, paste("res_Sim100_lmerSeq_voom_", n, ".rds",
      sep = ""))
359 }
```

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des Sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/sc