University Of Zurich

STA 426 Project

An analysis of the riborex package performance

Authors
Meili Peter
Stutz Sascha

Supervisors
Prof. Dr. Mark D. Robinson
Dr. Hubert Rehrauer

January 9, 2018

Contents

1 Introduction

The aim of this report is to (re-)produce the genome-wide mapping differences of ribosome profiling (short Ribo-Seq) data with the use of the Riborex (abbreviation for analyzing Ribo-seq ratios with expression) package (Li et al., 2017) while measuring the computation time. The authors claim that the Riborex package "[...] is a method that is dramatically faster than available methods without sacrificing accuracy" (Li et al., 2017).
The analysis will be done with the same three so-called engines as in the paper, namely DESeq2 (Love et al., 2014), edgeR (Robinson et al., 2010) and Voom (Law et al., 2014). Additionally, the Xtail (Xiao et al., 2016) and Babel (Olshen et al., 2013) package will be used in the same fashion, while it is refrained here from using RiboDiff, since this is not an R package nor can it be run in R. Hence, the focus lies on computing speed linked with precision.
The speed will be assessed as computation time the different methods take to identify differentially translated genes across two conditions by pairwise comparison of ribosome profiling data while the precision is compared with diagrams of the F1 score, accuracy and predictive values.

2 Riborex package

The main goal of the Riborex package is to limit the trade-off between speed and accuracy which already established options do not convince in. Due to a simplified implementation of the mathematical structure Riborex should outperform already existing methods.:
"Riborex directly leverages established RNA-seq analysis frameworks for all parameter estimation, providing users with a choice among robust engines for these computations.(Li et al., 2017)".
This states that existing engines (DESeq2, edgeR and Voom) for analyzing RNA-seq data are used in a way that a small rash in the base leads to an extensive one in the result. The natural dependence on mRNA levels are modelled as a generalized linear model. This is nothing new but what is different in this package, however, is that all parameters of the model are estimated simultaneously. Hence, the claimed improvement in speed without the loss of accuracy. Mathematically this looks as follows.

$$log(E(y_{gi})) = x_i^T \beta_g + log(N_i) \tag{1}$$

$$y_{gi} : Read\ count\ of\ gene\ g\ in\ sample\ i$$
$$N_i : Total\ counted\ reads$$
$$x_j : Covariate\ vector\ of\ treatment\ condition\ i$$
$$\beta_j : Vector\ of\ gene\ specific\ coefficients\ i$$

Equation (1) is how the aforementioned engines model the RNA-seq read count in a negative binomial distribution. The Riborex package on the other hand uses a slightly different approach to detect differentially translated genes from Ribo-seq as visible in equation (2) (not considering α_g, all other variables stay the same for interpretation as before).

$$log(E(r_{gi})) = x_i^T(\alpha_g + \beta_g) + log(R_i) \tag{2}$$

$$\alpha_g : Vector\ of\ how\ the\ translations\ differ\ from\ background\ expression\ level$$

In the end both read counts are combined by constructing a design matrix with which α and β can be fit simultaneously.

2.1 Practical usage

The package contains only the one function `riborex()` which has the option to specify a total of seven arguments (four without default).
`riborex(rnaCntTable, riboCntTable, rnaCond, riboCond, engine = "DESeq2", contrast = NULL, minMeanCount = 1)`
The abbreviations are mostly self explanatory (count tables and condition vectors must be specified) and will not be further discussed here (see the R help file). In a later stage of this paper this command will be used with the three different engines; DESeq2, edgeR and Voom. Furthermore, these three engines estimate the dispersion across all RNA- and Ribo-seq samples. On the other hand, edgeR estimates the dispersion for RNA- and Ribo-seq data separately and is not considered in this paper.

3 Datasets

As mentioned before, a lack of available datasets along with the needed information of the truly differentially translated genes was missing which therefore

would not allow to compute and compare precision. As usual, data was simulated trying to replicate nature as precise as anyhow possible. The simulated data is equally distributed between up- and down-regulated in group two compared to group one.
The count data is simulated from a negative binomial distribution while dispersion parameters are estimated from two sample datasets (one RNA and one Ribo count table) with the `estimateDisp()` command from the package `edgeR()` and then given over to the function.
No outlier groups are introduced and genes with zero counts across all samples are excluded. These settings approximately correspond to the ones denoted B_{625}^{625} in (Soneson, 2014).

```
> for(i in 1:length(diffExpr)){
+   for(j in 1:length(nsamples)){
+     for(v in 1:length(RnaOrRibo)){
+       myData <- generateSyntheticData(dataset = "B_625_625",
+                                       n.vars = ngenes,
+                                       samples.per.cond = nsamples[j],
+                                       n.diffexp = ngenes*diffExpr[i],
+                                       relmeans = means[[v]],
+                                       dispersions = disp[[v]])
+
+       type <- ifelse(RnaOrRibo[v] == 1, "RNA", "Ribo")
+
+       assign(paste0(type, ".nsamples", nsamples[j],".percDiffExpre"
+                     , diffExpr[i], sep = ""), myData)
+     }
+   }
+ }
>
```

The code junk above shows the standard use of the `generateSyntheticData()` function in this report. The for-loops are necessary to generate the different datasets for the different settings in number of samples, percentage of differentially expressed genes as well as if a RNA or Ribosome dataset is generated. While the `assign()` command is used to give a unique name to the datasets. Four synthetic count datasets are generated with different settings as seen in table 1.

Shortname	Genes	Samples	Differentially expressed [%]
10000/2/0.01	10000	2	0.01
10000/2/0.1	10000	2	0.1
10000/4/0.01	10000	4	0.01
10000/4/0.1	10000	4	0.1

Table 1: The four different generated datasets

These will be the settings used until subsection 4.5 where the number of genes for a fixed setting of the other two variables will be altered to 1’250, 2’500, 5’000 and 10’000. It is clear, that 1’250 up to 10’000 are low numbers for gene counts. But as stated above, the aim is to prove that Riborex is less time consuming without missing precision in identifying differentially expressed genes which should be measurable with low numbers of counts.

4 Analysis

In the main part of this report the F1 scores will be compared at first followed by a comparison of the accuracy with the corresponding 95% confidence intervals. To find out, whether the functions / engines are particularly designed for finding differentially translated genes an analysis of the positive predictive value and the negative predictive value is done.
This first part should answer the claim of (Li et al., 2017) that the Riborex package is as accurate as other existing methods such as Xtail and Babel. In the second part, a comparison of computing time with fixed numbers of genes is conducted as well as a computation time assessment as a function of the number of genes. This should answer the claim that the package is faster than any other having the same purpose.
In a first step the `riborex()`, Xtail and Babel function is given the four different datasets.

```
> # Set the start time
> start.time <- Sys.time()
> # Actual computing: identifying differentially expressed genes
> # by co-analyzing both RNA- and Ribo-seq data
> res <- riborex(RNACntTable, RiboCntTable, condition, condition,
+                engine = "edgeR")
> # Stop the time save it in seconds
> time <- as.numeric(difftime(Sys.time(), start.time,
+                             units = "secs"))
> # Save the time along with the used method, the number of
> # samples and the percent of differentiall expressed
> # genes in a matrix
> times <- rbind(times,
+                c(time, "edgeR", nsamples[j], diffExpr[i]))
> # give the result list a setting specific name
> assign(paste0("res.riborex.edgeR.nsamples", nsamples[j],
+               ".percDiffExpre", diffExpr[i], sep = ""), res)
```

The code junk for the Riborex function with the edgeR engine is used here exemplarily. Although there are some differences, variables that must be given over to the functions do not vary much compared to other engines and Xtail as well as Babel. A RNA and Ribosome count table must be specified as well as the conditions - treated or control.

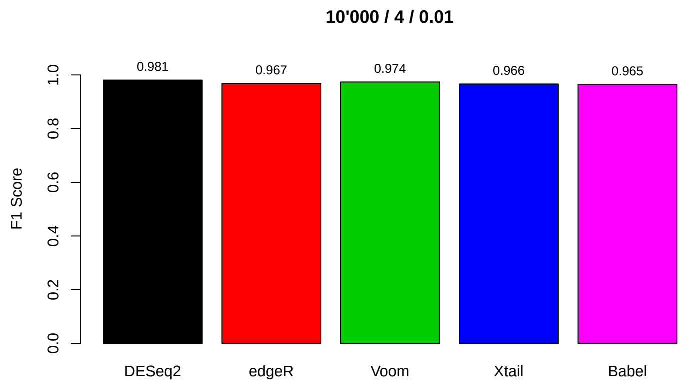

Figure 1: F1 scores for the different methods for 10000/4/0.01

The rest of the code does the time measurement (the difference between start and end time is measured).

4.1 F1 scores

Like in (Li et al., 2017) the F1 scores are displayed in figure 1 (the color code with black, red, green, blue and violet for the different functions will be used in the same fashion for all figures throughout the rest of this report). There is almost no difference between the F1 scores when applying a differetent dataset, so 10'000/4/0.01 is chosen at random.
The scores look very similar, all methods arrive at a very high score with some minor differences between the engines / functions. The relatively high F1 scores can be explained by the fact that simulated data was used instead of real data. This stands in comparison to (Li et al., 2017) where they arrived at values around 0.8 based on the fact that they randomly assigned fold changes. This was was not conducted in this very report. Additionally, it has to be noted that the F1 score here is based on just one run of the different functions / engines while in (Li et al., 2017) up to a 100 runs were conducted.

4.2 Accuracy

Since the F1 scores did not reveal too much information about differences, the accuracy with its 95% confidence interval as seen in figure 2 is used for further assessment. Here, a clear difference between the engines is visible.
While the accuracy of DESeq2 scores the highest, the other four are all below. Only calculations with one fixed dataset are displayed because, as before, the

```
> plotCI(1:5, Accuracy[,1], ui=Accuracy[,3], li=Accuracy[,2],
+        col =c(1,2,3,4,6), xlab = "", ylab = "Accuracy [%]",
+        xaxt = 'n', main = "Accuracy of 10'000 / 2 / 0.01")
> axis(1, at = 1:5, labels = method)
```

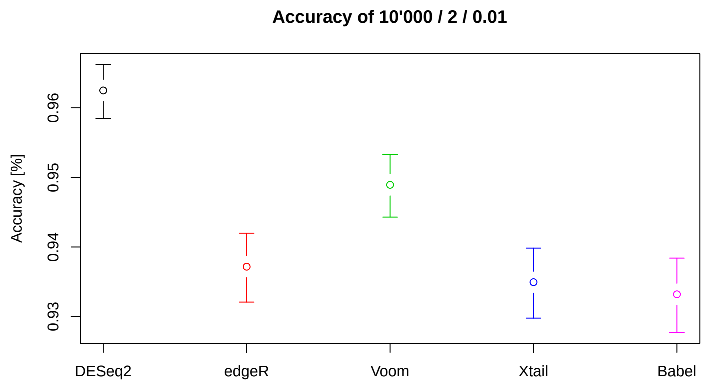

Figure 2: Accuracy with 95% confidence interval of the different methods

figure does not differ significantly for the four different datasets. The set 10'000 / 2 / 0.01 was chosen again for consistency reasons.
The `plotCI()` function is used here to visualize the accuracy estimates along with their confidence intervals is from the package `gplots()`. The three values for lower bound, upper bound and point estimate are from an object created by the `confusionMatrix()` function.

4.3 Predictive values

A third value of interest after F1 scores and accuracy are predictive values. Displayed in figure 3 they do not show a diverse picture of the situation. While the positive predictive values (PPV) are all very high and close to 1 with a difference under 1%, the negative predictive values (NPV) show values of $<$ 10 %. The best performing tool is, like before for accuracy, DESeq2 with the highest PPV and NPV.
It seems, that the packages and engines are well designed to find the differentially expressed genes and take a heavy toll on the negative side. It seems that the cost for false negative are higher than for false positive. There is to add that it does not matter which dataset is used for the analysis as they lead to approximately the same numbers.

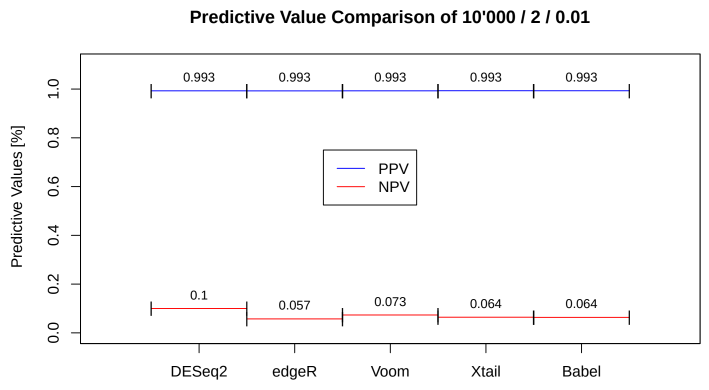

Figure 3: Predictive values for 10000/2/0.01 of the different functions / engines

4.4 Time comparison

The major claim of (Li et al., 2017) was that the Riborex package is much faster than other existing methods. While the focus in this subsection lies on the raw computation time without alterations, the next subsection (see subsection 4.5) deals with the growth of computation time linked to the number of genes in the dataset.

Execution time		Riborex		Xtail	Babel
datasets	DESeq2	edgeR	Voom		
10000/2/0.01	12.416 s	4.152 s	1.419 s	66.735 s	251.973 s
10000/2/0.1	10.744 s	4.457 s	1.452 s	66.002 s	272.965 s
10000/4/0.01	15.81 s	6.601 s	1.561 s	81.945 s	550.231 s
10000/4/0.1	15.019 s	6.566 s	1.96 s	82.847 s	500.2 s

Table 2: Running time of Riborex with its three engines, Xtail and Babel on four datasets

As visible in table 2, Xtail as Babel take considerably more time to compute (from up to over 6 fold more between DESeq2 and Xtail, to 360 fold between Voom and Babel). Riborex with its different engines runs much faster by a wide margin compared to the other two. The percentage of differentially expressed genes does not seem to have an effect on the computing time at first sight, however, the number of samples does. With doubled amount of samples per condition the time is higher as well, the factor the computation time is multiplied with underlies great variation across used methods.

4.5 Computation time dependent on the number of genes

The differences between the functions in the computation time almost certainly has a root in the number of genes which need to be analyzed. For that purpose four different gene counts (1'250, 2'500, 5'000 and 10'000) are used to display the behavior of the computation time.
These gene counts are set up in a way that the next count is always by factor 2 bigger than the previous one. As of that one would expect intuitionally, a doubled number of genes in each dataset proportionally might lead to a doubled amount of computation time.
Figure 4 displays the computation times in two plots, which is necessary because the computation time of the fast Riborex engines would not be visible due to immense time differences between the methods used. Furthermore the transformation to a log scale would complicate the interpretation, and was not conducted hence. While edgeR and Voom only show modest increase in computation time, DESeq2 displays an almost linear increase with a slope of two. The computation time of the Xtail package increases only marginally, starting, however, at a high point. Babel shows a almost log like curve. One may assume, that increase in computational time per gene drops with the number of genes rising.

5 Conclusion

It has been shown that the F1 scores for all methods are very high, even higher than in (Li et al., 2017), which is due to data simulation. Additionally the accuracy and positive predictive values are high.
All the different methods used are therefore very precise at identifying differentially translated genes by co-analyzing both RNA- and Ribo-seq data. The negative predictive values hence are not taken into account in this report in detail. Precision for datasets altering number of samples and amount of differentially translated genes did not change substantially. The Ribroex methods took considerably less computational time than Xtail and Babel in the range of up to 10'000 genes. That the methods differentially behave concerning the computational time by changing the number of genes is revealed in figure 4. It might be interesting in a next step to examine the packages with >>10'000 genes.
In conclusion it could be proved that the Riborex package with its three engines - namely Voom, DESeq2 and edgeR - do not lack precision in identifying differentially translated genes compared to Xtail and Babel, do, however, need considerably less computational time.

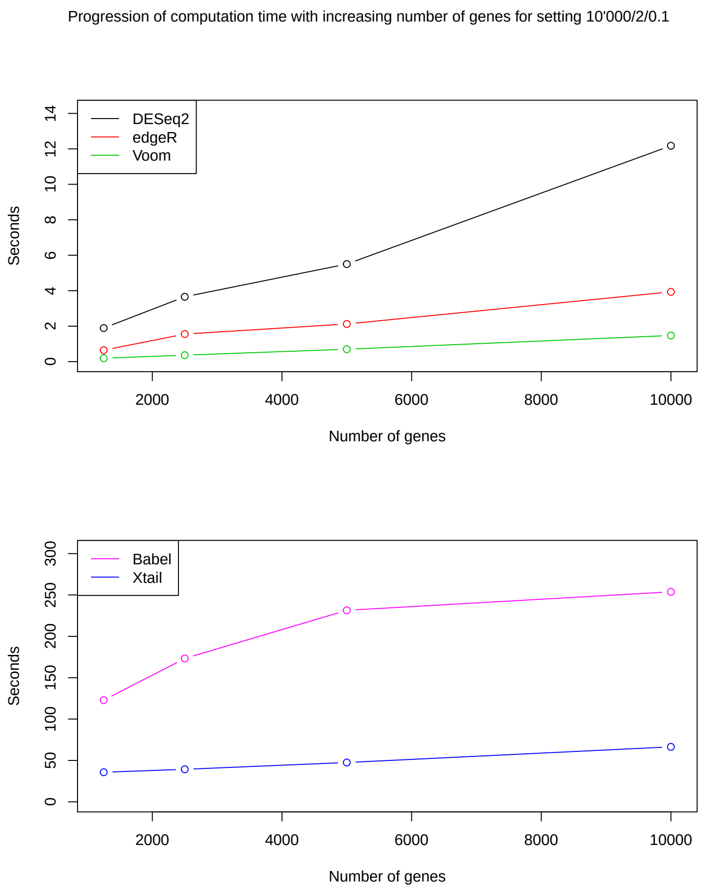

Figure 4: Computation time for four different ammounts of genes for the different engines / functions

References

Law, C. et al. (2014), ‘Voom: precision weights unlock linear model analysis tools for rna-seq read counts’, *Genome Biol.* **15**, R29.

Li, W., et al. (2017), ‘Riborex: fast and flexible identification of differential translation from ribo-seq data’, *Bioinformatics* **33**, 3472.

Love, M. et al. (2014), ‘Moderated estimation of fold change and dispersion for rna-seq data with deseq2’, *Genome Biol.* **15**, 1–21.

Olshen, A. et al. (2013), ‘Assessing gene-level translational control from ribosome profiling’, *Bioinformatics* **29**, 2995–3002.

Robinson, M. et al. (2010), ‘edger: a bioconductor package for differential expression analysis of digital gene expression data’, *Bioinformatics* **26**, 139–140.

Soneson, C. (2014), ‘compcoder - an r package for benchmarking differential expression methods for rna-seq data’, *Bioinformatics* **30**(17), 2517–2518.

Xiao, Z. et al. (2016), ‘Genome-wide assessment of differential translations with ribosome profiling data’, *Nature Communications* **7**, 11194.