RUVSeq: Remove Unwanted Variation from RNA-Seq Data

Davide Risso

Modified: April 13, 2014. Compiled: October 27, 2021.

Contents

1 Overview

In this document, we show how to conduct a differential expression (DE) analysis that controls for "unwanted variation", e.g., batch, library preparation, and other nuisance effects, using the between-sample normalization methods proposed in [1]. We call this approach *RUVSeq* for *remove unwanted variation from RNA-Seq data*.

Briefly, *RUVSeq* works as follows. For n samples and J genes, consider the following generalized linear model (GLM), where the RNA-Seq read counts are regressed on both the known covariates of interest and unknown factors of unwanted variation,

$$\log E[Y|W, X, O] = W\alpha + X\beta + O. \quad (1)$$

Here, Y is the $n \times J$ matrix of observed gene-level read counts, W is an $n \times k$ matrix corresponding to the factors of "unwanted variation" and α its associated $k \times J$ matrix of nuisance parameters, X is an $n \times p$ matrix corresponding to the p covariates of

interest/factors of "wanted variation" (e.g., treatment effect) and β its associated $p \times J$ matrix of parameters of interest, and O is an $n \times J$ matrix of offsets that can either be set to zero or estimated with some other normalization procedure (such as upper-quartile normalization).

The matrix X is a random variable, assumed to be known a priori. For instance, in the usual two-class comparison setting (e.g., treated vs. control samples), X is an $n \times 2$ design matrix with a column of ones corresponding to an intercept and a column of indicator variables for the class of each sample (e.g., 0 for control and 1 for treated) [2]. The matrix W is an unobserved random variable and α, β, and k are unknown parameters.

The simultaneous estimation of W, α, β, and k is infeasible. For a given k, we consider instead the following three approaches to estimate the factors of unwanted variation W:

- `RUVg` uses negative control genes, assumed to have constant expression across samples;
- `RUVs` uses centered (technical) replicate/negative control samples for which the covariates of interest are constant;
- `RUVr` uses residuals, e.g., from a first-pass GLM regression of the counts on the covariates of interest.

The resulting estimate of W can then be plugged into Equation 1, for the full set of genes and samples, and α and β estimated by GLM regression. Normalized read counts can be obtained as residuals from ordinary least squares (OLS) regression of $\log Y - O$ on the estimated W.

Note that although here we illustrate the RUV approach using the GLM implementation of *edgeR* and *DESeq2*, all three RUV versions can be readily adapted to work with any DE method formulated within a GLM framework.

See [1] for full details and algorithms for each of the three RUV procedures.

2 A typical differential expression analysis workflow

In this section, we consider the `RUVg` function to estimate the factors of unwanted variation using control genes. See Sections 3 and 4, respectively, for examples using the `RUVs` and `RUVr` approaches.

We consider the zebrafish dataset of [3], available through the *Bioconductor* package *zebrafishRNASeq*. The data correspond to RNA libraries for three pairs of gallein-treated and control embryonic zebrafish cell pools. For each of the 6 samples, we have RNA-Seq read counts for $32{,}469$ Ensembl genes and 92 ERCC spike-in sequences. See [1] and the *zebrafishRNASeq* package vignette for details.

```
library(RUVSeq)
library(zebrafishRNASeq)
data(zfGenes)
head(zfGenes)
```

```
##                     Ctl1 Ctl3 Ctl5 Trt9 Trt11 Trt13
## ENSDARG00000000001  304  129  339  102    16   617
## ENSDARG00000000002  605  637  406   82   230  1245
```

```
## ENSDARG00000000018  391  235  217  554   451   565
## ENSDARG00000000019 2979 4729 7002 7309  9395  3349
## ENSDARG00000000068   89  356   41  149    45    44
## ENSDARG00000000069  312  184  844  269   513   243
```

```
tail(zfGenes)
```

```
##             Ctl1 Ctl3 Ctl5  Trt9 Trt11 Trt13
## ERCC-00163   204   59  183   152   104    59
## ERCC-00164     6    1   74    11   206    21
## ERCC-00165   140  119   93   331    52    38
## ERCC-00168     0    0    0     0     2     0
## ERCC-00170   216  145  111   456   196   552
## ERCC-00171 12869 6682 7675 47488 24322 26112
```

2.1 Filtering and exploratory data analysis

We filter out non-expressed genes, by requiring more than 5 reads in at least two samples for each gene.

```
filter <- apply(zfGenes, 1, function(x) length(x[x>5])>=2)
filtered <- zfGenes[filter,]
genes <- rownames(filtered)[grep("^ENS", rownames(filtered))]
spikes <- rownames(filtered)[grep("^ERCC", rownames(filtered))]
```

After the filtering, we are left with 20806 genes and 59 spike-ins.

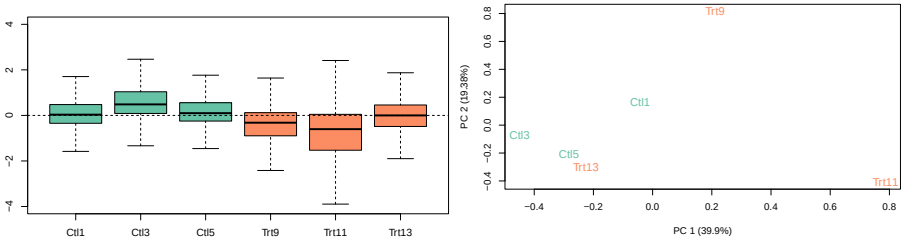
We store the data in an object of S4 class *SeqExpressionSet* from the *EDASeq* package. This allows us to make full use of the plotting and normalization functionality of *EDASeq*. Note, however, that all the methods in *RUVSeq* are implemented for both *SeqExpressionSet* and *matrix* objects. See the help pages for details.

```
x <- as.factor(rep(c("Ctl", "Trt"), each=3))
set <- newSeqExpressionSet(as.matrix(filtered),
                           phenoData = data.frame(x, row.names=colnames(filtered)))
set
```

```
## SeqExpressionSet (storageMode: lockedEnvironment)
## assayData: 20865 features, 6 samples
##   element names: counts, normalizedCounts, offset
## protocolData: none
## phenoData
##   sampleNames: Ctl1 Ctl3 ... Trt13 (6 total)
##   varLabels: x
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation:
```

The boxplots of relative log expression (RLE = log-ratio of read count to median read count across sample) and plots of principal components (PC) in Figure 1 reveal a clear need for betwen-sample normalization.

```
library(RColorBrewer)
colors <- brewer.pal(3, "Set2")
plotRLE(set, outline=FALSE, ylim=c(-4, 4), col=colors[x])
plotPCA(set, col=colors[x], cex=1.2)
```

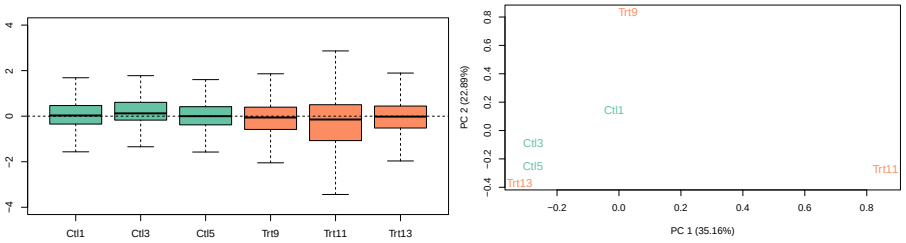


(a) RLE plot **(b)** PCA plot

Figure 1: No normalization

We can use the `betweenLaneNormalization` function of *EDASeq* to normalize the data using upper-quartile (UQ) normalization [4].

```
set <- betweenLaneNormalization(set, which="upper")
plotRLE(set, outline=FALSE, ylim=c(-4, 4), col=colors[x])
plotPCA(set, col=colors[x], cex=1.2)
```



(a) RLE plot **(b)** PCA plot

Figure 2: Upper-quartile normalization

After upper-quartile normalization, treated sample *Trt11* still shows extra variability when compared to the rest of the samples (Figure 2a). This is reflected by the first principal component (Figure 2b), that is driven by the difference between *Trt11* and the other samples.

2.2 RUVg: Estimating the factors of unwanted variation using control genes

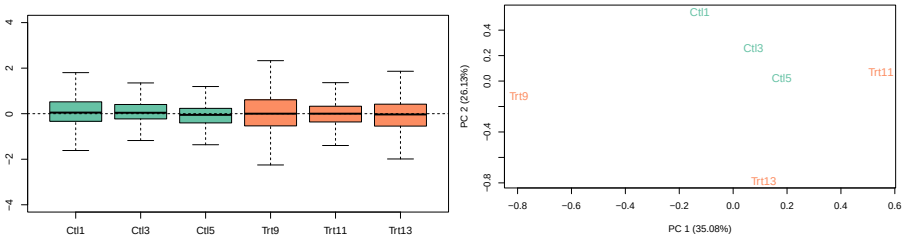
To estimate the factors of unwanted variation, we need a set of *negative control genes*, i.e., genes that can be assumed not to be influenced by the covariates of interest (in the case of the zebrafish dataset, the Gallein treatment). In many cases, such a set can be identified, e.g., housekeeping genes or spike-in controls. If a good set of negative controls is not readily available, one can define a set of "in-silico empirical" controls as in Section 2.4.

Here, we use the ERCC spike-ins as controls and we consider $k = 1$ factors of unwanted variation. See [1] and [5] for a discussion on the choice of k.

```
set1 <- RUVg(set, spikes, k=1)
pData(set1)
```

```
##          x         W_1
## Ctl1   Ctl -0.04539413
## Ctl3   Ctl  0.50347642
## Ctl5   Ctl  0.40575319
## Trt9   Trt -0.30773479
## Trt11  Trt -0.68455406
## Trt13  Trt  0.12845337
```

```
plotRLE(set1, outline=FALSE, ylim=c(-4, 4), col=colors[x])
plotPCA(set1, col=colors[x], cex=1.2)
```



(a) RLE plot **(b)** PCA plot

Figure 3: **RUVg normalization based on spike-in controls**

The `RUVg` function returns two pieces of information: the estimated factors of unwanted variation (added as columns to the `phenoData` slot of `set`) and the normalized counts obtained by regressing the original counts on the unwanted factors. The normalized values are stored in the `normalizedCounts` slot of `set` and can be accessed with the `normCounts` method. These counts should be used only for exploration. It is important that subsequent DE analysis be done on the *original counts* (accessible through the `counts` method), as removing the unwanted factors from the counts can also remove part of a factor of interest [6].

Note that one can relax the negative control gene assumption by requiring instead the identification of a set of positive or negative controls, with a priori known expression fold-changes between samples, i.e., known β. One can then use the centered counts for these genes ($\log Y - X\beta$) for normalization purposes.

2.3 Differential expression analysis

Now, we are ready to look for differentially expressed genes, using the negative binomial GLM approach implemented in *edgeR* (see the *edgeR* package vignette for details). This is done by considering a design matrix that includes both the covariates of interest (here, the treatment status) and the factors of unwanted variation.

```
design <- model.matrix(~x + W_1, data=pData(set1))
y <- DGEList(counts=counts(set1), group=x)
y <- calcNormFactors(y, method="upperquartile")
```

```
y <- estimateGLMCommonDisp(y, design)
y <- estimateGLMTagwiseDisp(y, design)

fit <- glmFit(y, design)
lrt <- glmLRT(fit, coef=2)

topTags(lrt)
```

2.4 Empirical control genes

If no genes are known *a priori* not to be influenced by the covariates of interest, one can obtain a set of "in-silico empirical" negative controls, e.g., least significantly DE genes based on a first-pass DE analysis performed prior to RUVg normalization.

```
design <- model.matrix(~x, data=pData(set))
y <- DGEList(counts=counts(set), group=x)
y <- calcNormFactors(y, method="upperquartile")
y <- estimateGLMCommonDisp(y, design)
y <- estimateGLMTagwiseDisp(y, design)

fit <- glmFit(y, design)
lrt <- glmLRT(fit, coef=2)

top <- topTags(lrt, n=nrow(set))$table
empirical <- rownames(set)[which(!(rownames(set) %in% rownames(top)[1:5000]))]
```

Here, we consider all but the top $5{,}000$ genes as ranked by *edgeR* p-values.

```
set2 <- RUVg(set, empirical, k=1)
pData(set2)
```

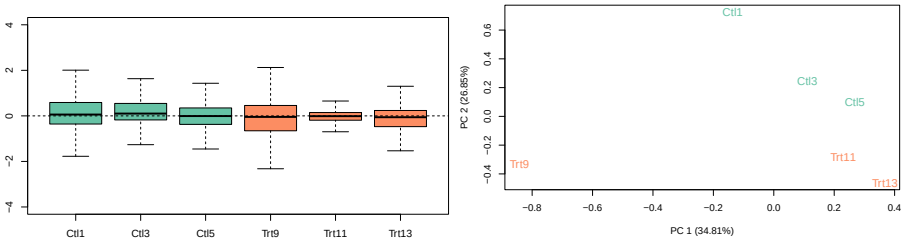
```
##          x         W_1
## Ctl1   Ctl -0.10879677
## Ctl3   Ctl  0.23066424
## Ctl5   Ctl  0.19926266
## Trt9   Trt  0.07672121
## Trt11  Trt -0.83540924
## Trt13  Trt  0.43755790
```

```
plotRLE(set2, outline=FALSE, ylim=c(-4, 4), col=colors[x])
plotPCA(set2, col=colors[x], cex=1.2)
```

2.5 Differential expression analysis with *DESeq2*

In alternative to *edgeR*, one can perform differential expression analysis with *DESeq2*. The approach is very similar, namely, we will use the same design matrix specified in Section 2.3, but we need to specify it within the *DESeqDataSet* object.

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(countData = counts(set1),
                              colData = pData(set1),
```



(a) RLE plot **(b)** PCA plot

Figure 4: RUVg normalization based on empirical controls

```
                                    design = ~ W_1 + x)
dds <- DESeq(dds)
res <- results(dds)
res
```

Note that this will perform by default a Wald test of significance of the last variable in the design formula, in this case x. If one wants to perform a likelihood ratio test, she needs to specify a reduced model that includes W (see the *DESeq2* vignette for more details on the test statistics).

```
dds <- DESeq(dds, test="LRT", reduced=as.formula("~ W_1"))
res <- results(dds)
res
```

3 RUVs: Estimating the factors of unwanted variation using replicate samples

As an alternative approach, one can use the `RUVs` method to estimate the factors of unwanted variation using replicate/negative control samples for which the covariates of interest are constant.

First, we need to construct a matrix specifying the replicates. In the case of the zebrafish dataset, we can consider the three treated and the three control samples as replicate groups. The function `makeGroups` can be used.

```
differences <- makeGroups(x)
differences
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Although in principle one still needs control genes for the estimation of the factors of unwanted variation, we found that `RUVs` is robust to that choice and that using all the genes works well in practice [1].

```
set3 <- RUVs(set, genes, k=1, differences)
pData(set3)
```

4 RUVr: Estimating the factors of unwanted variation using residuals

Finally, a third approach is to consider the residuals (e.g., deviance residuals) from a first-pass GLM regression of the counts on the covariates of interest. This can be achieved with the `RUVr` method.

First, we need to compute the residuals from the GLM fit, without RUVg normalization, but possibly after normalization using a method such as upper-quartile normalization.

```
design <- model.matrix(~x, data=pData(set))
y <- DGEList(counts=counts(set), group=x)
y <- calcNormFactors(y, method="upperquartile")
y <- estimateGLMCommonDisp(y, design)
y <- estimateGLMTagwiseDisp(y, design)

fit <- glmFit(y, design)
res <- residuals(fit, type="deviance")
```

Again, we can use all the genes to estimate the factors of unwanted variation.

```
set4 <- RUVr(set, genes, k=1, res)
pData(set4)
```

5 Session info

```
toLatex(sessionInfo())
```

- R Under development (unstable) (2021-10-19 r81077), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.15-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.15-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: Biobase 2.55.0, BiocGenerics 0.41.0, BiocParallel 1.29.0, Biostrings 2.63.0, EDASeq 2.29.0, GenomeInfoDb 1.31.0, GenomicAlignments 1.31.0, GenomicRanges 1.47.1, IRanges 2.29.0, MatrixGenerics 1.7.0, RColorBrewer 1.1-2, RUVSeq 1.29.0, Rsamtools 2.11.0, S4Vectors 0.33.0, ShortRead 1.53.0, SummarizedExperiment 1.25.0, XVector 0.35.0, edgeR 3.37.0, knitr 1.36, limma 3.51.0, matrixStats 0.61.0, zebrafishRNASeq 1.13.0

- Loaded via a namespace (and not attached): AnnotationDbi 1.57.0, BiocFileCache 2.3.0, BiocIO 1.5.0, BiocManager 1.30.16, BiocStyle 2.23.0, DBI 1.1.1, DelayedArray 0.21.0, GenomeInfoDbData 1.2.7, GenomicFeatures 1.47.1, KEGGREST 1.35.0, MASS 7.3-54, Matrix 1.3-4, R.methodsS3 1.8.1, R.oo 1.24.0, R.utils 2.11.0, R6 2.5.1, RCurl 1.98-1.5, RSQLite 2.2.8, Rcpp 1.0.7, XML 3.99-0.8, aroma.light 3.25.0, assertthat 0.2.1, biomaRt 2.51.0, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.2, cachem 1.0.6, codetools 0.2-18, compiler 4.2.0, crayon 1.4.1, curl 4.3.2, dbplyr 2.1.1, digest 0.6.28, dplyr 1.0.7, ellipsis 0.3.2, evaluate 0.14, fansi 0.5.0, fastmap 1.1.0, filelock 1.0.2, generics 0.1.1, glue 1.4.2, grid 4.2.0, highr 0.9, hms 1.1.1, htmltools 0.5.2, httr 1.4.2, hwriter 1.3.2, jpeg 0.1-9, lattice 0.20-45, latticeExtra 0.6-29, lifecycle 1.0.1, locfit 1.5-9.4, magrittr 2.0.1, memoise 2.0.0, parallel 4.2.0, pillar 1.6.4, pkgconfig 2.0.3, png 0.1-7, prettyunits 1.1.1, progress 1.2.2, purrr 0.3.4, rappdirs 0.3.3, restfulr 0.0.13, rjson 0.2.20, rlang 0.4.12, rmarkdown 2.11, rtracklayer 1.55.0, stringi 1.7.5, stringr 1.4.0, tibble 3.1.5, tidyselect 1.1.1, tools 4.2.0, utf8 1.2.2, vctrs 0.3.8, xfun 0.27, xml2 1.3.2, yaml 2.2.1, zlibbioc 1.41.0

References

[1] D. Risso, J. Ngai, T.P. Speed, and S. Dudoit. Normalization of RNA-seq data using factor analysis of control genes or samples. *Nature Biotechnology*, 2014. In press.

[2] P McCullagh and JA Nelder. *Generalized Linear Models*. Chapman and Hall, New York, 1989.

[3] T. Ferreira, S. R. Wilson, Y. G. Choi, D. Risso, S. Dudoit, T. P. Speed, and J. Ngai. Silencing of odorant receptor genes by G Protein $\beta\gamma$ signaling ensures the expression of one odorant receptor per olfactory sensory neuron. *Neuron*, 81:847–859, 2014.

[4] J. Bullard, E. Purdom, K. Hansen, and S. Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 11(1):94, 2010.

[5] J. Gagnon-Bartsch and T.P. Speed. Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, 13(3):539–552, 2012.

[6] J. Gagnon-Bartsch, L. Jacob, and T. P. Speed. Removing unwanted variation from high dimensional data with negative controls. Technical Report 820, Department of Statistics, University of California, Berkeley, 2013.