Visualization and assessment of RNA-Seq data normalization - the Normalization Visualization Tool

Thomas Eder[1,2], Florian Grebien[1], Thomas Rattei[2]

[1] Ludwig Boltzmann Institute for Cancer Research,
Währinger Straße 13A, 1090 Vienna, Austria.
[2] CUBE Division of Computational Systems Biology,
Department of Microbiology and Ecosystem Science,
University of Vienna, Althanstraße 14, 1090 Vienna, Austria.

February 12, 2016

Abstract

Measuring differential gene expression is a common task in the analysis of RNA-Seq data. To identify differentially expressed genes between two samples, it is crucial to normalize the datasets. While multiple normalization methods are available, all of them are based on certain assumptions that may or may not be suitable for the type of data they are applied on. Researchers therefore need to select an adequate normalization strategy for each RNA-Seq experiment. This selection includes exploration of different normalization methods as well as their comparison. Methods that agree with each other most likely represent realistic assumptions under the particular experimental conditions. We developed the *NVT* package, which provides a fast and simple way to analyze and evaluate multiple normalization methods via visualization and representation of correlation values, based on a user-defined set of uniformly expressed genes. This vignette explains the use of the package and demonstrates a typical work flow.

NVT version: 1.0

If you use *NVT* in published research, please cite:

T. Eder, F. Grebien, T. Rattei: **NVT: a fast and simple tool for the assessment of RNA-Seq normalization strategies**.
JournalXXX 2016, **XX**:XXX.
http://dx.doi.org/xxx/xxx

Contents

1 Installation

The latest NVT package can be downloaded from: https://github.com/Edert/NVT/releases. The installation of the package can be done directly in R.

```
install.packages(file.path("/home/user/Downloads/","NVT_1.0.tar.gz"),
repos=NULL, type="source")
```

2 Standard work flow

Initially the library has to be loaded, which provides access to the data examples 2.1.1 and the *NVT* functions.

```
library("NVT")
```

2.1 Load data

For the demonstration of the *NVT* functions we need gene expression data, so we load the example data provided with the package. We use the two human expression data sets *GSM1275862* and *GSM1275863* from the airway[1] package.

2.1.1 Load expression data

We simply load the provided data sets *myexp1* from *GSM1275862*, *myexp2* from *GSM1275863* and the length data per gene as *mylen*.

```
data(mylen)
data(myexp1)
data(myexp2)

#show just the first four elements of the loaded data
head(mylen,4)

##                 Length
## ENSG00000000003   3688
## ENSG00000000005   1881
## ENSG00000000419   4032
## ENSG00000000457  10784

head(myexp1,4)

##                 GSM1275862
## ENSG00000000003        679
## ENSG00000000005          0
## ENSG00000000419        467
## ENSG00000000457        260
```

In order to evaluate the expression between the two RNA-seq expression samples, we have to define a list of human housekeeping genes *GAPDH, ALB, ACTB, QARS, PGK2, HPRT1, ADA* and *POLR2L*.

```
mylist1<-c("ENSG00000111640","ENSG00000163631","ENSG00000075624",
           "ENSG00000172053","ENSG00000170950","ENSG00000165704",
           "ENSG00000196839","ENSG00000177700")
```

2.1.2 Load gene length data

Instead of using the length data generated directly in R or using a simple flat file, it is also possible to load the gene or exon length data directly from an annotation file in gtf or gff format. This function utilizes the GenomicRanges[2] and rtracklayer[3] packages.

- gff-version: version of the provided gff file $[gff1, gff2, gff3, gtf]$
- gff-feature: feature to use $[default : exon]$
- gff-name: name to use $[default : gene\_id]$

```
#this line gets the path to the gff file provided in the NVT package
#(annotation from Chlamydia trachomatis, ACCESSION: NC_000117 )
mygffpath <- system.file("extdata", "Ctr-D-UW3CX.gff", package = "NVT")

#this function loads the gff file from the gffpath
mylen1 <- NVTloadgff(mygffpath,"gff3","gene","locus_tag")

head(mylen1)

##       Length
## CT001    273
## CT002    303
## CT003   1476
## CT004   1467
## CT005   1092
## CT006    570
```

2.2 Generate NVTdata objects

In the first step you need to generate an *NVTdata* object with the *NVTinit* function. Here you have to provide the list of housekeeping genes, the two gene expression samples and the normalization method. Optionally you can also add the gene or exon length data.

The normalization methods are:

- N = No normalization
- TC = Total count normalization
- Med = Median normalization
- TMM = Trimmed Mean of M-values normalization
- UQ = Upper Quartile normalization
- UQ2 = Upper Quartile normalization (from NOISeq[4])
- Q = Quantile normalization implemented in limma[5]

- RPKM = Reads Per Kilobase per Million mapped reads normalization
- RPM = Reads Per Million mapped reads normalization
- TPM = Transcripts Per Million normalization
- DEQ = Relative log expression method included in the DESeq[6] package
- G = Use the provided genes to normalize

For the most methods no length information is required.

```
mynvt <- NVTinit(mylist1,myexp1,myexp2,"TMM")
```

But for RPKM and TPM normalization the length data has to be provided.

```
mynvt <- NVTinit(mylist1,myexp1,myexp2,"RPKM",mylen)
```

2.2.1 Normalize the NVTdata

The now initialized *NVTdata* object can be normalized in the next step.

```
mynorm <- NVTnormalize(mynvt)

## [1] "RPKM normalization!"
```

If required the now normalized data can be retrieved easily.

```
mynvalues <- show(mynorm)

head(mynvalues)

##                 GSM1275862 GSM1275863
## ENSG00000000003  8.9209656  5.8859979
## ENSG00000000005  0.0000000  0.0000000
## ENSG00000000419  5.6121512  6.1889890
## ENSG00000000457  1.1682249  0.9480595
## ENSG00000000460  0.2395964  0.2196300
## ENSG00000000938  0.0000000  0.0000000
```

2.3 Visualize expression data

One of the key features of *NVT* is the plotting of the XY-Scatter-Plots. This can be done with two functions: *NVTplot* and *NVTadvancedplot*.

2.3.1 Simple plot

The normalized *NVTdata* object can be visualized with the plot function. With the second parameter you can modify the size-ratio of the text and the data points. All expression values are shown as grey dots and the linear model as red line, it is calculated with the data from the housekeeping genes only. The dashed grey line highlights the diagonal, so the perfect correlation between the two data sets.

```
NVTplot(mynorm,0.6)
```

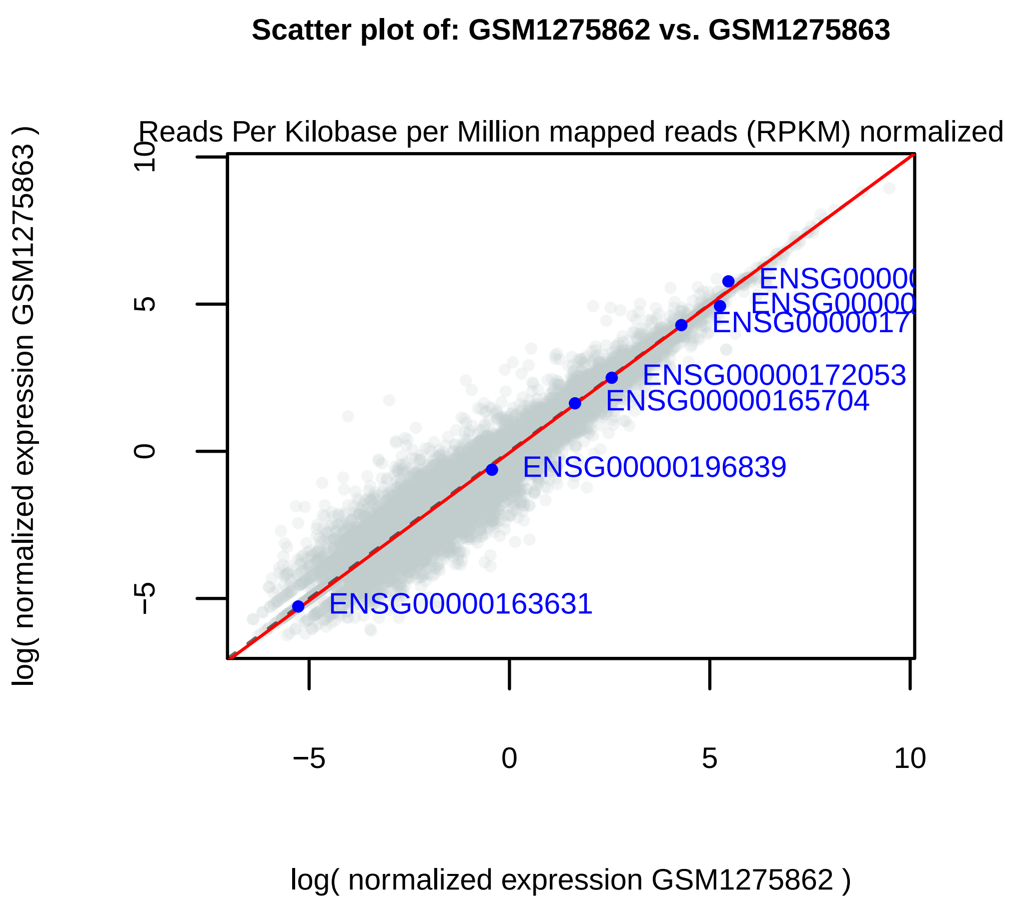

Figure 1: **XY-Scatter-Plot.** The resulting plot from the simple NVTplot function

2.3.2 Advanced plot

The *NVTadvancedplot* plots via ggplot2[7] and the size parameters control data points, text and labels separately. Here we use the default values of 1 for the data points and the text and increase the labels of the housekeeping genes to 2. Again the grey dots represent the expression data and their density is visualized by the rug in dark red for x- and y-axis. The linear model of the housekeeping gene data is shown in red and the diagonal is highlighted as grey dashed line. As we want to compare different methods we now normalize with the TMM method.

```
mynvt <- NVTinit(mylist1,myexp1,myexp2,"TMM",mylen)
mynorm <- NVTnormalize(mynvt)

## [1] "Trimmed Mean of M-values normalization!"

NVTadvancedplot(mynorm,1,1,2)
```

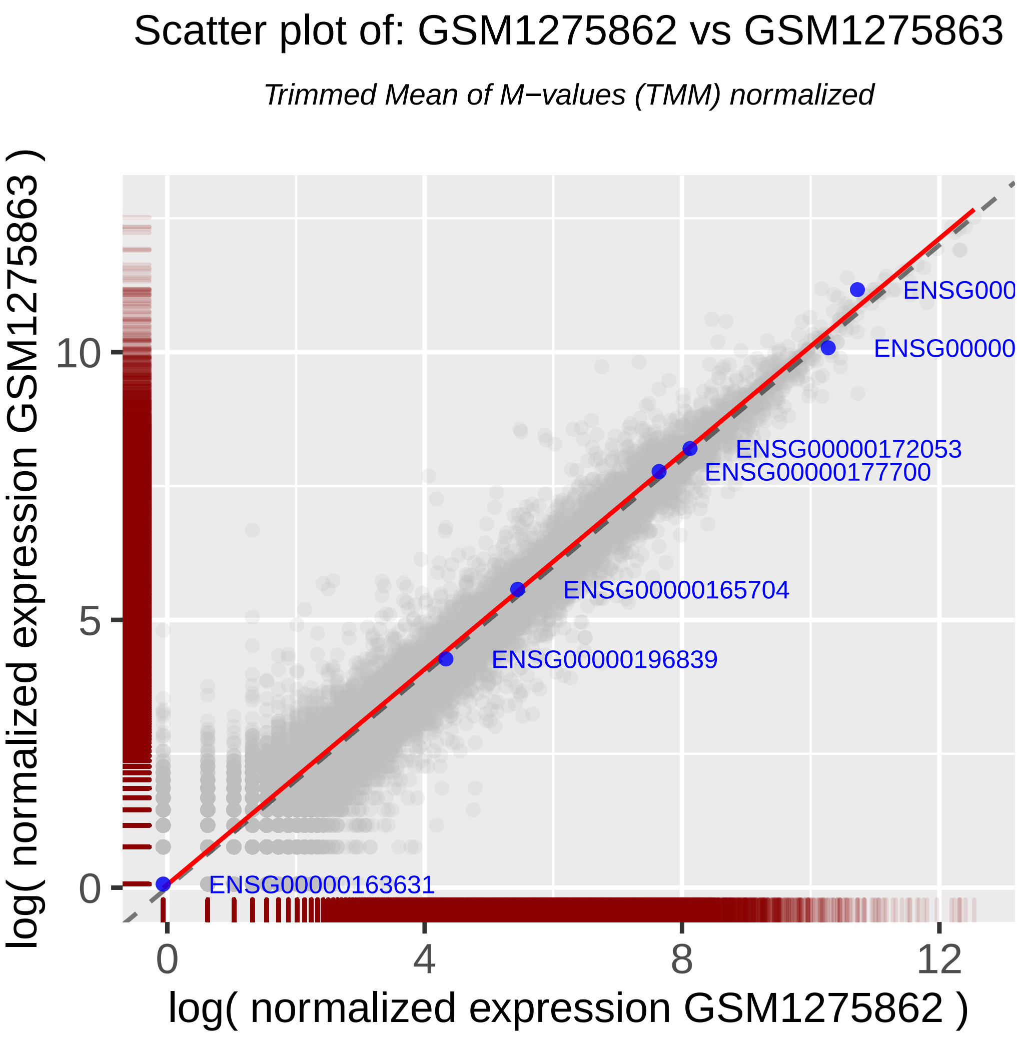

Figure 2: **Advanced XY-Scatter-Plot.** The NVTadvancedplot function results in a plot produced by ggplot2 with a rug on both axis, indicating the density of the data-points

2.3.3 Linear model

If required, the linear model plotted as red line can be retrieved with the *NVTlm* function.

```
mylm <- NVTlm(mynorm)

summary(mylm)

##
## Call:
## lm(formula = sample2 ~ sample1)
##
## Residuals:
## ENSG00000111640 ENSG00000163631 ENSG00000075624 ENSG00000172053 ENSG00000165704
##        -0.30369         0.07392         0.32555        -0.02492         0.03966
## ENSG00000196839 ENSG00000177700
##        -0.13993         0.02941
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.05662    0.17320   0.327    0.757
## sample1      1.00548    0.02310  43.522 1.21e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2128 on 5 degrees of freedom
## Multiple R-squared:  0.9974,Adjusted R-squared:  0.9968
## F-statistic:  1894 on 1 and 5 DF,  p-value: 1.209e-07
```

2.4 Correlation values

In addition to the graphical representation of the gene expression data, the correlation coefficients of the housekeeping genes of the two samples can be calculated with the functions *NVTpearson*, *NVTrmsd* and *NVTmae*.

2.4.1 Pearson correlation

The Pearson correlation coefficient of the normalized expression of the housekeeping genes is calculated with the following command, using an already normalized *NVTdata* object.

```
NVTpearson(mynorm)

##      pearson      p-value
## 0.9636246988 0.0001170673
```

2.4.2 RMSD and MEA correlation

The root mean square error (RMSE) also called the root mean square deviation (RMSD) is calculated with the *NVTrmsd* function.

```
NVTrmsd(mynorm)

##     RMSD
## 9097.041
```

And the mean absolute error (MAE) can be calculated and retrieved with the *NVTmae* function.

```
NVTmae(mynorm)

##      MAE
## 3855.185
```

2.5 Test all methods

To test more normalization methods on the provided data sets in one single step, the correlation coefficients of all implemented normalization methods can be calculated with the *NVTtestall* function. It requires a normalized *NVTdata* object and the correlation coefficient you are interested in. The method can be defined with:

- p = Pearson correlation
- rmsd = root mean square deviation
- mae = mean absolute error

```
NVTtestall(mynorm,"p")

## [1] "No normalization!"
## [1] "Total count normalization!"
## [1] "Median normalization!"
## [1] "Trimmed Mean of M-values normalization!"
## [1] "Upper Quartile normalization!"
## [1] "Upper Quartile normalization (from NOISeq)!"
## [1] "Quantile normalization!"
## [1] "RPKM normalization!"
## [1] "RPM normalization!"
## [1] "DESeq normalization!"
## [1] "Using DESeq"
## [1] "Input counts are not integer, converting them!"
## [1] "TPM normalization!"
## [1] "Normalization by given gene-set!"
## [1] "ENSG00000111640" "ENSG00000163631" "ENSG00000075624" "ENSG00000172053"
## [5] "ENSG00000170950" "ENSG00000165704" "ENSG00000196839" "ENSG00000177700"
##        pearson      p-value
## Q    0.9716940 5.550234e-05
## DEQ  0.9636247 1.170673e-04
## TMM  0.9636247 1.170673e-04
## G    0.9636247 1.170673e-04
## TC   0.9636247 1.170673e-04
## UQ   0.9636247 1.170673e-04
## N    0.9636247 1.170673e-04
## Med  0.9636247 1.170673e-04
## UQ2  0.9636247 1.170673e-04
## RPM  0.9636247 1.170673e-04
## RPKM 0.9489857 3.193364e-04
## TPM  0.9489857 3.193364e-04
```

3 Session Info

- R version 3.2.2 (2015-08-14), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=de_AT.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=de_AT.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=de_AT.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=de_AT.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: knitr 1.12, NVT 1.0
- Loaded via a namespace (and not attached): annotate 1.48.0, AnnotationDbi 1.32.3, Biobase 2.30.0, BiocGenerics 0.16.1, BiocParallel 1.4.3, BiocStyle 1.8.0, Biostrings 2.38.3, bitops 1.0-6, colorspace 1.2-6, DBI 0.3.1, DESeq 1.22.0, evaluate 0.8, formatR 1.2.1, futile.logger 1.4.1, futile.options 1.0.0, genefilter 1.52.0, geneplotter 1.48.0, GenomeInfoDb 1.6.1, GenomicAlignments 1.6.3, GenomicRanges 1.22.3, ggplot2 2.0.0, grid 3.2.2, gtable 0.1.2, highr 0.5.1, IRanges 2.4.6, labeling 0.3, lambda.r 1.1.7, lattice 0.20-33, limma 3.26.5, magrittr 1.5, Matrix 1.2-3, munsell 0.4.2, NOISeq 2.14.0, parallel 3.2.2, plyr 1.8.3, RColorBrewer 1.1-2, Rcpp 0.12.3, RCurl 1.95-4.7, Rsamtools 1.22.0, RSQLite 1.0.0, rtracklayer 1.30.1, S4Vectors 0.8.7, scales 0.3.0, splines 3.2.2, stats4 3.2.2, stringi 1.0-1, stringr 1.0.0, SummarizedExperiment 1.0.2, survival 2.38-3, tools 3.2.2, XML 3.98-1.3, xtable 1.8-0, XVector 0.10.0, zlibbioc 1.16.0

References

[1] Himes, B. E., Jiang, X., Wagner, P., Hu, R., Wang, Q., Klanderman, B., Whitaker, R. M., Duan, Q., Lasky-Su, J., Nikolos, C., Jester, W., Johnson, M., Panettieri, R. A., Tantisira, K. G., Weiss, S. T., Lu, and Q. RNA-Seq Transcriptome Profiling Identifies CRISPLD2 as a Glucocorticoid Responsive Gene that Modulates Cytokine Function in Airway Smooth Muscle Cells. *PLoS ONE*, 9(6):e99625, 2014. doi:10.1371/journal.pone.0099625.

[2] Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin Morgan, and Vincent Carey. Software for computing and annotating genomic ranges. *PLoS Computational Biology*, 9, 2013. doi:10.1371/journal.pcbi.1003118.

[3] Michael Lawrence, Robert Gentleman, and Vincent Carey. rtracklayer: an r package for interfacing with genome browsers. *Bioinformatics*, 25:1841–1842, 2009. doi:10.1093/bioinformatics/btp328.

[4] Tarazona S, Garcia-Alcalde F, Dopazo J, Ferrer A, and Conesa A. Differential expression in rna-seq: a matter of depth. *Genome research*, 21:4436, 2011. doi:10.1101/gr.124321.111.

[5] Matthew E Ritchie, Belinda Phipson, Di Wu, Yifang Hu, Charity W Law, Wei Shi, and Gordon K Smyth. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, 43(7):e47, 2015. doi:10.1093/nar/gkv007.

[6] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010. doi:10.1186/gb-2010-11-10-r106.

[7] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. URL: http://had.co.nz/ggplot2/book.