# OperaMate: data importing, processing and analysis for Opera High Content Screening System

Chenglin Liu

Shanghai Jiaotong University,
Shanghai, China
cliu@sjtu.edu.cn

Yixue Li

Shanghai Jiaotong Univeristy,
Shanghai, China
yxli@sibs.ac.cn

April 24, 2017

## Contents



## 1 Introduction

The *OperaMate* is intended to analyze protein intensity data derived from the images from PerkinElmer's Opera High Content Screening System (http://www.perkinelmer.com/pages/020/cellularimaging/products/opera.xhtml) and intepreted by Columbus Image Data Storage and Analysis System (http://www.perkinelmer.com/pages/020/cellularimaging/products/columbus.xhtml). PerkinElmer's Opera High Content Screening System is the state-of-the-art confocal microplate imaging solution for high throughput screening. The system works with complex disease models and offers various solutions like protein expression, RNAi screening. Its compatible tool Columbus image analysis system exports intensity data by analyzing the Opera images, but lacks further processing and analysis to uncover the biological meaning underlying

the image data. Hence, we develop an R [1] package `OperaMate` especially to process the intensity data exported by Columbus system, which can fulfills the procedure of data importing, processing and analysis in an easy and efficient way.

# 2 Getting Started

*OperaMate* integrates the entire process of data importing, processing and analysis into one function, which is easy to operate for the users who are new to the R language. However, it is also very convinient to customize each step according to the pipeline built by this function, checking the immediate results of each step, and re-performing a specific step with different parameters to achieve the desire of the users.

## 2.1 Configration

In order to process and analyze data by one function, the configuration file should be supplied. The templete can be obtained together with the package named param.txt. Users are required to fill the blanks after colons. Leave them as blank if NULL is expected. An example can be obtained by : Required column names of `genemap`:

- Barcode: character, the barcode of the plates.
- Well: character, the well ID.
- GeneSymbol: character, the annotated gene names of the well.

An example can be obtained by :

```
   Barcode Well PoolCatalogNumber GeneSymbol GENE.ID Accession.Number GI.Number
1 DSIMGA02  B03       M-058386-01       Rdh8  235033     NM_001030290  71892411
2 DSIMGA02  B04       M-043284-01      Nlrp9a  233001     NM_001048220 115299751
3 DSIMGA02  B05       M-162761-00     Polr2f   69833        NM_027231  58037180
4 DSIMGA02  B06       M-059494-01       MYOCD  214384        NM_146386 154240725
5 DSIMGA02  B07       M-066220-00        HPGD   15446        NM_008278 124486705
```

## 2.2 Running OperaMate

```
> library(OperaMate)
> configFile <- file.path(system.file("Test", package = "OperaMate"), "demoData", "demoParam.txt")
> operaReport <- operaMate(configFile, gDevice = "png")
> names(operaReport)
```

```
[1] "lstCells"  "report"    "funReport"
```

See the mannual for more details about this function.

## 2.3 Description of output

All the reports and figures of the demo data can be find at

```
> tempdir()
```

# 3 Class declaration

## 3.1 The expData class

Each `expData` object stores data of one imaging analysis report of the Opera system generatd by Columbus$^{TM}$ Image Data Storage and Analysis System. The report includes different types of data of one plate, distinguished by the paramters. The format of the report is set in the analysis system. *OperaMate* supports two most popular formats by now: the matrix and the table. The class requires the following information:

1. `name`, name of the plate
2. `path`, path of the importing file
3. `rep.id`, replicate ID
4. `exp.id`, barcode of the experiment
5. `format`, report format
   (a) `Matrix`: matrix format in the Columbus analysis system
   (b) `Tab`: table format in the Columbus analysis system

An example of creating a new expData object is as follows:

```
> onePlate <- expData(name = "DSIMGA02-s1",
+                     path = file.path(
+                       system.file("Test",package = "OperaMate"),
+                       "Matrix", "130504-s1-02.txt" ),
+                     rep.id = "s1", exp.id = "DSIMGA02",
+                     format = "Matrix")
```

However, it is highly recommended to import all files using the function `loadAll`. Details will be referred in Section 4.1.

## 3.2 The cellData class

An object of `expData` class stores all data corresponding to one parameter of the reports. It organizes the data as a matrix with rows are well IDs and columns the plate IDs. This class requires to provide the name of the object, which must be one parameter of the report. Additional information includes:

1. `posctrwell`, the well IDs of the positive controls, e.g. B05
2. `negctrwell`, the well IDs of the negative controls, e.g. B05
3. `expwell`, the well IDs of the samples, e.g. C12
4. `norm.method`, one method from "MP", "PMed", "Z", "Ctr", "None"
5. `QC.threshold`, the thresholds for quality control

The `expData` objects provide different places for different levels of the data: the raw data in the `origin.data` slot; the normalized data in the `norm.data` slot and data after quality control in the `qc.data` slot. In addition, the general quality of each plate is stored in the `plate.quality` slot.

An example of creating a new cellData object is as follows:

```
> oneCell <- cellData(name = "Average Intensity of Nuclei",
+                     positive.ctr = c("H02", "J02", "L02"),
+                     negative.ctr = c("C23", "E23", "G23"))
> oneCell
```

```
An object of cellData class.
Parameter:  Average.Intensity.of.Nuclei .
Raw Data: NULL.
Data Normalization: ToDo...;
     method: MP.
Quality Control: ToDo....
Significant hits:
      ToDo...
```

# 4 Data importing and processing

## 4.1 Raw data importing and re-organization

*OperaMate* imports all reports of the Columbus analysis system to `expData` objects using the function `loadAll`, and then re-organizes them to several `cellData` objects corresponding to different parameters of the reports. The function `loadAll` requires all reports are of the same data format and are placed in the same location. The structure of the file name can be specified by an example, e.g. egFilename = list(eg.filename = "0205-s2-01.txt", rep.id = "s2", exp.id = "01", sep = "-", barcode = "DSIMGA01"). If the file formats can meet these requirements, assign `cellformat` as "Matrix" or "Tab", and pass the location of the files to `datapath`. Otherwise, you need to specify the information of each file by a data.frame variable. An example can be referred by `data(platemap)`.

Then, you can import the data and construct `cellData` objects as follows:

```
> datapath <- file.path(system.file("Test", package = "OperaMate"), "Matrix")
> lstPlates <- loadAll(cellformat = "Matrix", datapath = datapath,
+                      egFilename <- list(eg.filename = "Tab.130504-s1-01.txt",
+                                         rep.id = "s1", exp.id = "01", sep = "-",
+                                         barcode = "DSIMGA01")
+                      )
> oneCell <- cellData(name = "Average Intensity of Nuclei")
> oneCell <- cellLoad(oneCell, lstPlates, neglect.well = c("*02", "*23"),
+                     positive.ctr = c("H02", "J02", "L02"),
+                     negative.ctr = c("C23", "E23", "G23"))
> str(oneCell["origin.data"])
```

```
 num [1:286, 1:9] 28.7 33.1 34.1 111 113.7 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:286] "H02" "J02" "L02" "C23" ...
  ..$ : chr [1:9] "DSIMGA02-s1" "DSIMGA03-s1" "DSIMGA02-s2" "DSIMGA03-s2" ...
```

If the reports are of other formats, you can redefine the function `parseTemplete`. See the manual for more information.

## 4.2 Data normalization

Data normalization is to reducing the systematical technical variance among the raw data. Technical sources of variation are unavoidable during experiments, and different amounts of variance occur in different rounds of experiments, which are referred as "batch effect". As to the plate-based assays, different techincal variations are added to different plates, and different locations of well are suffered with different amounts of noise, especially the edge wells. The latter phenomenon is called "edge effects".

Good normalization methods help to attenuate the batch effects. The *OperaMate* package provides several normalization methods including "MP", "PMed", "Z"" "Ctr". Ctr method divides data by the mean of the plate controls. This approach is often favored by biologists. However, as to the large sample screening, it is usually not as accurate as methods which take all samples into consideration. All of the other methods consider all samples, and use the majority of samples as a negative reference. The "PMed" method divides data by the median of their corresponding plates. "MP" employs the Tukey's median polish procedure, and divides data by the median of their corresponding plates and wells recursively. "Z" is the robust z-score method which firstly substracts data by the median of their plates, and then divides them by the median absolute deviation of the plates. The default normalization method is "MP". However, if no normalization methods are expected, you can assign the `norm.method` as "None".

An example is as follows:

```
> oneCell <- cellNorm(oneCell, norm.method = "MP")
> str(oneCell["norm.data"])
```

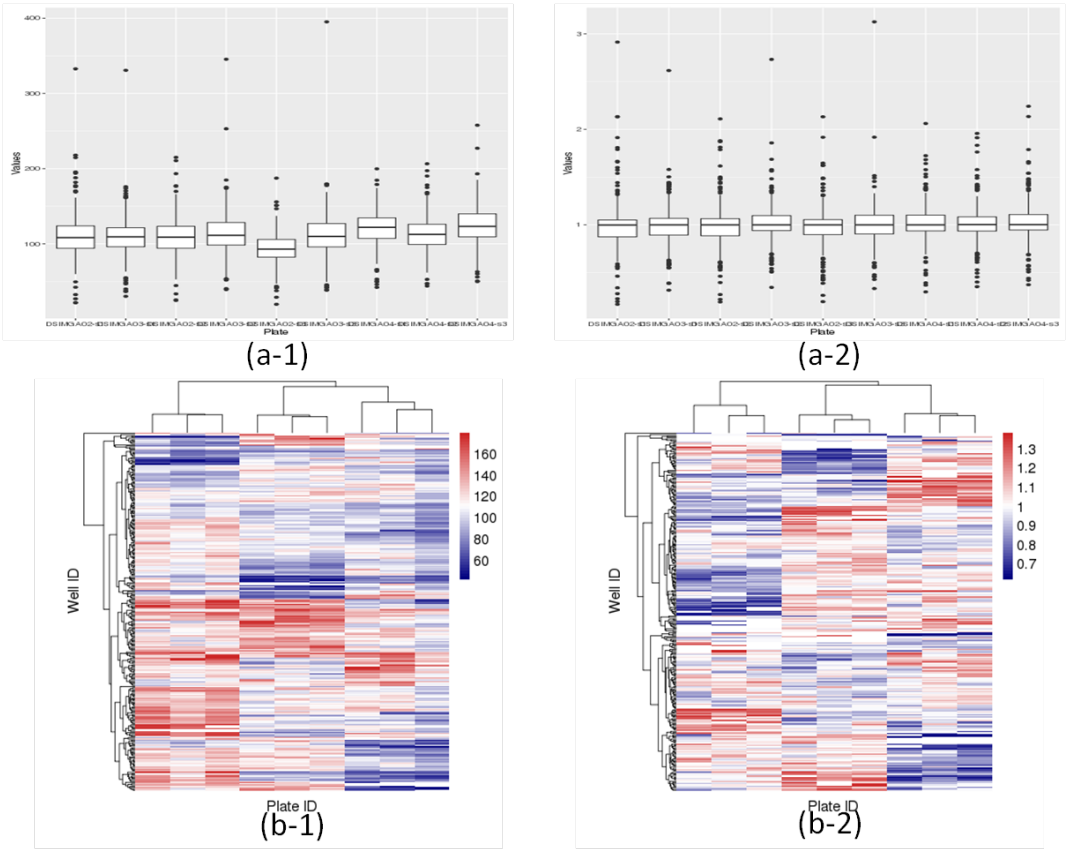


Figure 1: (a-1) Boxplot of raw data.; (a-2) boxplot of norm data; (b-1) heatmap of raw data; (b-2) heatmap of norm data.

```
 num [1:286, 1:9] 1.06 1.06 1.22 1.03 1.01 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:286] "H02" "J02" "L02" "C23" ...
  ..$ : chr [1:9] "DSIMGA02-s1" "DSIMGA03-s1" "DSIMGA02-s2" "DSIMGA03-s2" ...
```

Contrasting colors are very useful to visualize the batch effects. The *OperaMate* package provides two ways, hierarchical clustering (heatmap), the boxplot of each plate data. Heatmap method performs hieratical clustering to data matrix, and a large region of distinguishing color indicates the corresponding data with different technical variations. Boxplots visualize the distribution of the data in each plate. Examples are shown as shown in Figure 1.

```
> cellViz(oneCell, data.type = c("raw", "norm"), plateID = 1:6, outpath = tempdir())
```

In addition, you can check a specific plate.

```
> cellViz(oneCell, data.type = c("raw", "norm"), plateID = 1, outpath = tempdir())
```

## 4.3 Quality control

Before using the data for biological analysis, you should verify that your data passes quality control checks. `OperaMate` provides several ways to do quality control. "PlateCorrelation" checks if the duplicated plates have similar distributions. Pearson correlations are performed between every pair of duplicated plates, and the low correlation means bad replicates (default: 0.8). "Zfactor" checks the z factors of the wells of each plate. The data from the bad plates/wells are replaced by the mean of their duplicates if replace.badPlateData is TRUE. The qualities of the plates are stored in the `plate.quality` slot. Next, it performs the quality control well by well. It detects abnormal standardard deviations (sds) of replicated

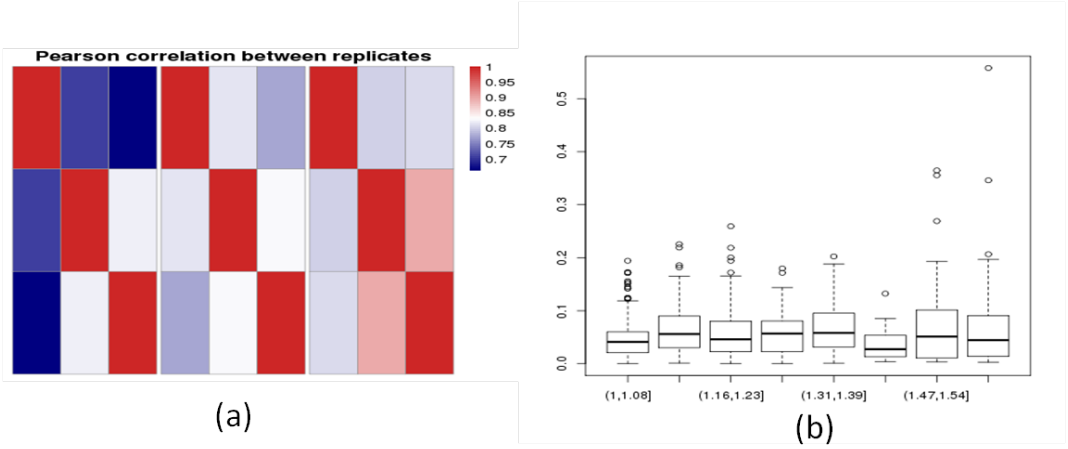


Figure 2: (a) Pearson correlation of replicated plates; (b) standard deviations of replicated wells binned by mean values.

wells considering the mean values. Boxplot shows the abnormal sds as the outliers (black points outside the box), as shown in Figure 2.

The cell numbers are loaded by the following function:

```
> cell.cellNum <- cellData(name = "Cells.Analyzed")
> cell.cellNum <- cellLoad(cell.cellNum, lstPlates, neglect.well = c("*02", "*23"),
+                     positive.ctr = c("H02", "J02", "L02"),
+                     negative.ctr = c("C23", "E23", "G23"))
> oneCell <- cellNumLoad(oneCell, cell.cellNum)
```

Quality control is performed by the following function:

```
> oneCell <- cellQC(oneCell, qcType = c("plateCorrelation", "wellSd", "cellNumber"),
+                   qc.threshold = c(correlation = 0.7), outpath = tempdir())
> str(oneCell["qc.data"])
```

```
'data.frame':        858 obs. of  8 variables:
 $ s1             : num  1.06 1.06 1.22 1.03 1.01 ...
 $ s2             : num  1.401 1.357 1.138 0.997 0.944 ...
 $ s3             : num  1 0.919 0.957 1 0.934 ...
 $ mean           : num  1.2 1.138 1.047 0.998 0.939 ...
 $ pass.plateQC.s1: logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ pass.plateQC.s2: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ pass.plateQC.s3: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ pass.wellQC    : logi  FALSE FALSE TRUE TRUE TRUE TRUE ...
```

```
> head(oneCell["plate.quality"])
```

```
         plateQC.s1 plateQC.s2 plateQC.s3
DSIMGA02      FALSE       TRUE       TRUE
DSIMGA03       TRUE       TRUE       TRUE
DSIMGA04       TRUE       TRUE       TRUE
```

## 5 Hit identification and biological analysis

The "hits" are the samples which are meaningfully different from the negative controls. The compulsory method is the t-test between the samples and the controls. The hits are the samples those significantly differ from the negative controls

based on t-test ( default: p-value less than 0.05). This method is combined with the following methods to reduce a high false positives:

1. `ksd`, $mean \pm k\ standard\,deviation$. The hits are the samples those surpass k (default: 3) standard deviation relative to the mean. This approach is often used with z-score normalization.
2. `kmsd`, $median \pm k\ median\,absolute\,deviation$. The hits are the samples those surpass k (default: 3) median absolute deviation relative to median. It is more robust than `ksd` as to a nonnormal distribution.
3. `stable`, stable distribution method. You can check the fitness to the stable distribution by the QQ plot.

An example is as follows:

```
> oneCell <- cellSig(oneCell, method = "stable", th = c(0.05, 0.05),
+                    outpath = tempdir())
> names(oneCell["Sig"])
```

```
[1] "SigMat"    "threshold" "stats"     "pvalue"
```

The fitness of stable distribution to the data can be visualized, as shown in Figure 3(a).

In addition, the signicance of the hits can be exihibited by a volvano plot,as shown in Figure 3(b).

```
> labels <- c("Axin1")
> names(labels) <- c("DSIMGA04:C07")
> cellSigPlot(oneCell, highlight.label = labels, outpath = tempdir())
```

Moreover, the potential biological meanings can be interpreted with the help of gProfileR package. You are required to provide the well-gene specification table. The description of `genemap` is referred in Section 2.1. The p-values barplot can be visualized, as shown in Figure 3(d)

```
> genemap <- read.csv(file.path(system.file("Test", package = "OperaMate"),
+                               "demoData", "genemap.csv"), stringsAsFactors = FALSE)
> chart <- cellSigAnalysis(oneCell, genemap, organism = "mmusculus")
> head(chart, n = 5)
```

```
  query.number significant  p.value term.size query.size overlap.size recall precision
1            1        TRUE 0.017400         5         21            2  0.095      0.40
2            1        TRUE 0.000336        12         21            3  0.143      0.25
3            1        TRUE 0.050000         2          1            1  1.000      0.50
4            1        TRUE 0.023700         5          7            2  0.286      0.40
5            1        TRUE 0.023700         5          7            2  0.286      0.40
     term.id domain subgraph.number                                        term.name
1 GO:0071877     BP               6 regulation of adrenergic receptor signaling pathway
2 GO:0030877     CC               2                    beta-catenin destruction complex
3 CORUM:5459    cor               8                            Axin2-Ctnnb1-Apc complex
4 HP:0030255     hp               5                       Large intestinal polyposis
5 HP:0005227     hp               5                     Adenomatous colonic polyposis
  relative.depth    intersection
1              1      RGS2,GSK3A
2              1 AXIN2,APC,AXIN1
3              1           AXIN2
4              1       AXIN2,APC
5              1       AXIN2,APC
```

```
> cellSigAnalysisPlot(chart, prefix = oneCell@name, outpath = tempdir())
```

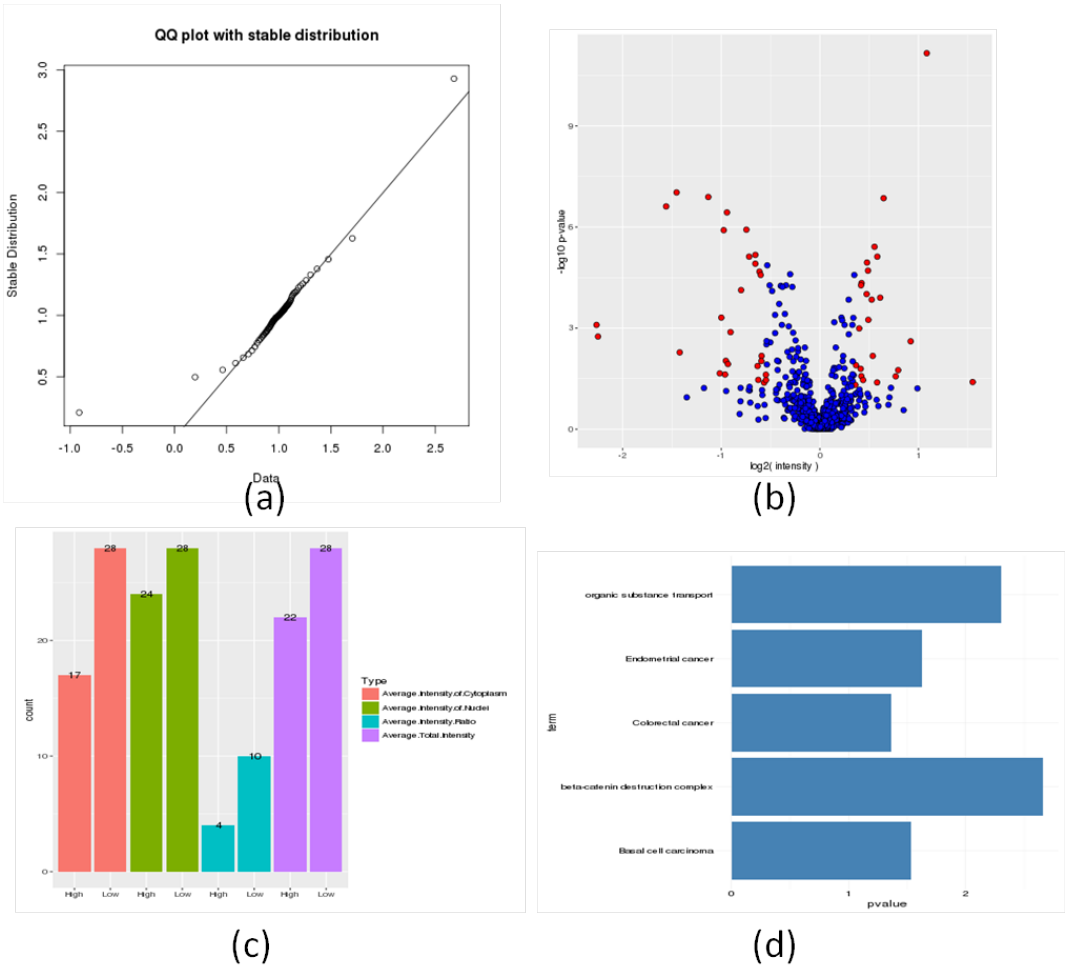


Figure 3: (a) QQ plot between data and stable distribution; (b) volcano plot between the log2 intensity and the log10 p-value in the multiple t-tests. The red points are hits statistically and quantitatively significantly different from the negative controls. (c) The counts of the detected hits. (d) The functions of the detected hits.

## 6 Summarization

At last, all processed data and the significant hits of all data type are summarized to a single report, which is easy to check using Microsoft Office Excel or other softwares. The numbers of hits are visualized by a histogram, as shown in Figure 3(c).

```
> report <- generateReport(list(oneCell), genemap, verbose = FALSE,
+                          plot = FALSE)
> head(report, n = 5)
```

```
   Barcode Well PoolCatalogNumber GeneSymbol GENE.ID Accession.Number GI.Number      mean
1 DSIMGA02  B03       M-058386-01       Rdh8  235033     NM_001030290  71892411 0.9984792
2 DSIMGA02  B04       M-043284-01     Nlrp9a  233001     NM_001048220 115299751 0.6131442
3 DSIMGA02  B05       M-162761-00     Polr2f   69833        NM_027231  58037180 0.9601916
4 DSIMGA02  B06       M-059494-01      MYOCD  214384        NM_146386 154240725 0.8990051
5 DSIMGA02  B07       M-066220-00       HPGD   15446        NM_008278 124486705 0.9863102
  pass.quality   Hits
1         Pass notHit
```

```
2         Pass notHit
3         Pass notHit
4         Pass notHit
5         Pass notHit
```

## 7 Acknowledgement

We thank Li Mao and his advisor Lin Li for providing the original screening data generated by the Columbus system. The example data in the package are synthesis data generated based on their providing data.

## 8 Session info

```
R version 3.4.0 (2017-04-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.2 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] OperaMate_1.9.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.10          knitr_1.15.1          magrittr_1.5
 [4] munsell_0.4.3         colorspace_1.3-2      gProfileR_0.6.1
 [7] stringr_1.2.0         plyr_1.8.4            timeSeries_3022.101.2
[10] tools_3.4.0           grid_3.4.0            timeDate_3012.100
[13] gtable_0.2.0          htmltools_0.3.5       yaml_2.1.14
[16] lazyeval_0.2.0        rprojroot_1.2         digest_0.6.12
[19] tibble_1.3.0          gridExtra_2.2.1       reshape2_1.4.2
[22] RColorBrewer_1.1-2    ggplot2_2.2.1         bitops_1.0-6
[25] RCurl_1.95-4.8        evaluate_0.10         rmarkdown_1.4
[28] labeling_0.3          pheatmap_1.0.8        stringi_1.1.5
[31] compiler_3.4.0        scales_0.4.1          backports_1.0.5
[34] fBasics_3011.87       stabledist_0.7-1      BiocStyle_2.5.0
```

## References

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