RNA-Seq data analysis using mulitple statistical algorithms with metaseqR

Panagiotis Moulos

October 27, 2020

During the past few years, a lot of R/Bioconductor packages have been developped for the analysis of RNA-Seq data, introducing several approaches. For example, packages using the negative binomial distribution to model the null hypotheses (*DESeq*, *edgeR*, *NBPSeq*) or packages using Bayesian statistics (*baySeq*, *EBSeq*). In addition, packages specialized to RNA-Seq data normalization have also been developed (*EDASeq*). The package *metaseqR* (pronounced meta-seek-er) tries to provide an interface to several algorithms, similar to the recent *TCC* package. However, it is much simpler to use than *TCC*, incoporates more algorithms for normalization and statistical analysis and builds a full report with several interactive and non-interactive diagnostic plots so that the users can easily explore the results and have whatever they need for this part of their research in one place. The metaseqR report is one of its most strong points as it provides an automatically generated summary, based on the pipeline inputs and the results, which can be used directly as a draft in methods paragraph in scientific publications. It also provides a lot of diagnostic figures and each figure is accompanied by a small explanatory text, and a list of references accroding to the algorithms used in the pipeline, which can also be used in a scientific article. All the report components are grouped in a comprehensive way, with a table of contents. Most importantly, metaseqR provides an interface for RNA-Seq data meta-analysis by providing the ability to use different algorithms for the statistical testing part and combining the p-values using popular published methods (e.g. Fisher's method, Whitlock's method) and two package-specific methods (intersection, union of statistically significant results). In the future, more algorithms will be incorporated in the package, with more diagnostic plots and more examples. This initial vignette contains just this introductory text and reference to some examples in the package documentation, which we believe that at this point contains sufficient explanation to run the metaseqr pipeline. Throughout the rest of this document, *metaseqr* refers to the name of the analysis pipeline while *metaseqR* refers to the name of the package.

1 Getting started

Detailed instructions on how to run the metaseqr pipeline can be found under the main documentation of the metaseqR package:

```
library(metaseqR)
help(metaseqr) # or
help(metaseqr.main)
```

Briefly, to run metaseqr you need:

- A text tab delimited file in a spreadsheet like format containing at least unique gene identifiers (corresponding to one of metaseqR's supported formats, for the time being Ensembl)
- A list of statistical contrasts for which you wish to check differential expression
- An internet connection so that the interactive parts of the report can be properly rendered, as the report template points to external Content Delivery Networks (CDNs) distributing the appropriate JavaScript

Everything else (e.g. genomic regions annotation etc.) can be handled by the metaseqr pipeline. Some example data are included in the package. See the related help pages:

```
help(hg18.exon.data)
help(mm9.gene.data)
```

2 Running the metaseqr pipeline

Running a metaseqr pipeline instance is quite straightforward. Again, see the examples in the main help page. An example and the command window output follow:

```
data("mm9.gene.data",package="metaseqR")
```

```
head(mm9.gene.counts)
```

```
##                     chromosome   start     end            gene_id gc_content
## ENSMUSG00000090699      chr12 3023883 3025753 ENSMUSG00000090699     0.4441
## ENSMUSG00000092168      chr12 3082667 3087041 ENSMUSG00000092168     0.4649
## ENSMUSG00000079179      chr12 3236611 3249999 ENSMUSG00000079179     0.4178
## ENSMUSG00000020671      chr12 3247430 3309969 ENSMUSG00000020671     0.4023
## ENSMUSG00000069181      chr12 3343816 3357153 ENSMUSG00000069181     0.4508
## ENSMUSG00000020668      chr12 3365132 3406494 ENSMUSG00000020668     0.4756
##                     strand transcripts     gene_name e14.5_1 e14.5_2 a8w_1 a8w_2
## ENSMUSG00000090699      +           1         Gm9071       2       5    11    25
## ENSMUSG00000092168      +           1         Gm9075       0       0     1     0
## ENSMUSG00000079179      +           1 1700012B15Rik     838     718   366   261
## ENSMUSG00000020671      -           1          Rab10    1868    2004   977   807
## ENSMUSG00000069181      -           1        Gm17681      15      18    14    28
## ENSMUSG00000020668      +           3          Kif3c     162     155   126   122
```

```
sample.list.mm9
```

```
## $e14.5
## [1] "e14.5_1" "e14.5_2"
##
## $adult_8_weeks
## [1] "a8w_1" "a8w_2"
```

```
libsize.list.mm9
```

```
## $e14.5_1
## [1] 3102907
##
## $e14.5_2
## [1] 2067905
##
## $a8w_1
## [1] 3742059
##
## $a8w_2
## [1] 4403954
```

Following is a full example with the informative messages that are printed in the command window:

```
library(metaseqR)
data("mm9.gene.data",package="metaseqR")
out.dir <- tempdir()
print(out.dir)

## [1] "/tmp/RtmpREyzeR"

result <- metaseqr(
       counts=mm9.gene.counts,
       sample.list=sample.list.mm9,
       contrast=c("e14.5_vs_adult_8_weeks"),
```

```
       libsize.list=libsize.list.mm9,
       annotation="download",
       org="mm9",
       count.type="gene",
       normalization="edger",
       statistics="edger",
       qc.plots=c(
           "mds","filtered","correl","pairwise","boxplot","gcbias",
           "lengthbias","meandiff","meanvar","deheatmap","volcano"
       ),
       pcut=0.05,
       fig.format=c("png","pdf"),
       export.what=c("annotation","p.value","meta.p.value",
          "adj.meta.p.value","fold.change"),
       export.scale=c("natural","log2"),
       export.values="normalized",
       export.stats=c("mean","sd","cv"),
       export.where=out.dir,
       restrict.cores=0.1,
       gene.filters=list(
             length=list(
                    length=500
             ),
             avg.reads=list(
                    average.per.bp=100,
                    quantile=0.25
             ),
             expression=list(
                    median=TRUE,
                    mean=FALSE,
                    quantile=NA,
                    known=NA,
                    custom=NA
             ),
             biotype=get.defaults("biotype.filter","mm9")
       ),
       out.list=TRUE
)
```

```
##
## 2020-10-27 21:32:35:  Data processing started...
##
## Read counts file:  imported custom data frame
## Conditions:  e14.5, adult_8_weeks
## Samples to include:  e14.5_1, e14.5_2, a8w_1, a8w_2
## Samples to exclude:  none
## Requested contrasts:  e14.5_vs_adult_8_weeks
## Library sizes:
##  e14.5_1:  3102907
##  e14.5_2:  2067905
##  a8w_1:  3742059
##  a8w_2:  4403954
## Annotation:  download
## Organism:  mm9
## Reference source:  ensembl
## Count type:  gene
## Transcriptional level:  gene
## Exon filters:  min.active.exons
##  min.active.exons:
##    exons.per.gene:  5
##    min.exons:  2
```

```
##    frac:  0.2
## Gene filters:  length, avg.reads, expression, biotype
##  length:
##    length:  500
##  avg.reads:
##    average.per.bp:  100
##    quantile:  0.25
##  expression:
##    median:  TRUE
##    mean:  FALSE
##    quantile:  NA
##    known:  NA
##    custom:  NA
##  biotype:
##    pseudogene:  FALSE
##    snRNA: FALSE
##    protein_coding:  FALSE
##    antisense:  FALSE
##    miRNA: FALSE
##    lincRNA: FALSE
##    snoRNA: FALSE
##    processed_transcript:  FALSE
##    misc_RNA: FALSE
##    rRNA: TRUE
##    sense_overlapping:  FALSE
##    sense_intronic:  FALSE
##    polymorphic_pseudogene:  FALSE
##    non_coding:  FALSE
##    three_prime_overlapping_ncrna:  FALSE
##    IG_C_gene:  FALSE
##    IG_J_gene:  FALSE
##    IG_D_gene:  FALSE
##    IG_V_gene:  FALSE
##    ncrna_host:  FALSE
## Filter application:  postnorm
## Normalization algorithm:  edger
## Normalization arguments:
##  method:  TMM
##  logratioTrim:  0.3
##  sumTrim:  0.05
##  doWeighting:  TRUE
##  Acutoff:  -1e+10
##  p:  0.75
## Statistical algorithm:  edger
## Statistical arguments:
##  edger:  classic, 5, 10, movingave, NULL, grid, 11, c(-6, 6), NULL, CoxReid, 10000, NULL, auto,
NULL, NULL, NULL, NULL, 0.125, NULL, auto, chisq, TRUE, FALSE, c(0.05, 0.1)
## Meta-analysis method:  none
## Multiple testing correction:  BH
## p-value threshold:  0.05
## Logarithmic transformation offset:  1
## Quality control plots:  mds, filtered, correl, pairwise, boxplot, gcbias, lengthbias, meandiff,
meanvar, deheatmap, volcano
## Figure format:  png, pdf
## Output directory:  /tmp/RtmpREyzeR
## Output data:  annotation, p.value, meta.p.value, adj.meta.p.value, fold.change
## Output scale(s):  natural, log2
## Output values:  normalized
## Downloading gene annotation for mm9...
## Saving gene model to /tmp/RtmpREyzeR/data/gene_model.RData
```

```
## Removing genes with zero counts in all samples...
## Normalizing with:  edger
## Applying gene filter length...
##  Threshold below which ignored:  500
## Applying gene filter avg.reads...
##  Threshold below which ignored:  0.0659670745106788
## Applying gene filter expression...
##  Threshold below which ignored:  68
## Applying gene filter biotype...
##  Biotypes ignored:  rRNA
## 2106 genes filtered out
## 1681 genes remain after filtering
## Running statistical tests with:  edger
##  Contrast:  e14.5_vs_adult_8_weeks
##  Contrast e14.5_vs_adult_8_weeks:  found 906 genes
## Building output files...
##  Contrast:  e14.5_vs_adult_8_weeks
##    Adding non-filtered data...
##      binding annotation...
##      binding p-values...
##      binding natural normalized fold changes...
##      binding log2 normalized fold changes...
##    Writing output...
##    Adding filtered data...
##      binding annotation...
##      binding p-values...
##      binding natural normalized fold changes...
##      binding log2 normalized fold changes...
##    Writing output...
## Creating quality control graphs...
## Plotting in png format...
##  Plotting mds...
##  Plotting correl...
##  Plotting pairwise...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting deheatmap...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting volcano...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting filtered...
## Plotting in pdf format...
##  Plotting mds...
##  Plotting correl...
##  Plotting pairwise...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting boxplot...
##  Plotting gcbias...
```

```
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting deheatmap...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting volcano...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting filtered...
## Creating HTML report...
## Compressing figures...
##
## 2020-10-27 21:32:54:  Data processing finished!
##
##
## Total processing time:  19 seconds
```

To get a glimpse on the results, run:

```
head(result[["data"]][["e14.5_vs_adult_8_weeks"]])
```

```
##                     chromosome     start       end            gene_id gc_content
## ENSMUSG00000058207      chr12 105576696 105583951 ENSMUSG00000058207      47.53
## ENSMUSG00000079012      chr12 105625374 105632467 ENSMUSG00000079012      47.77
## ENSMUSG00000066361      chr12 105384592 105392080 ENSMUSG00000066361      48.06
## ENSMUSG00000091553      chr12 105427727 105431136 ENSMUSG00000091553      47.65
## ENSMUSG00000074207       chr3 137923955 137953662 ENSMUSG00000074207      41.02
## ENSMUSG00000089635       chr3 137944372 137969986 ENSMUSG00000089635      42.05
##                     strand    gene_name        biotype p-value_edger
## ENSMUSG00000058207      +    Serpina3k protein_coding  3.286503e-64
## ENSMUSG00000079012      +    Serpina3m protein_coding  4.278067e-46
## ENSMUSG00000066361      -    Serpina3c protein_coding  2.989298e-44
## ENSMUSG00000091553      - Serpina3e-ps     pseudogene  3.114273e-43
## ENSMUSG00000074207      +         Adh1 protein_coding  5.271837e-41
## ENSMUSG00000089635      -      Gm16559      antisense  8.042133e-41
##                     natural_normalized_fold_change_e14.5_vs_adult_8_weeks
## ENSMUSG00000058207                                             104862.5000
## ENSMUSG00000079012                                               2951.7500
## ENSMUSG00000066361                                              16047.5000
## ENSMUSG00000091553                                               1055.0000
## ENSMUSG00000074207                                                625.5600
## ENSMUSG00000089635                                                652.8095
##                     log2_normalized_fold_change_e14.5_vs_adult_8_weeks
## ENSMUSG00000058207                                          16.678139
## ENSMUSG00000079012                                          11.527355
## ENSMUSG00000066361                                          13.970061
## ENSMUSG00000091553                                          10.043027
## ENSMUSG00000074207                                           9.289004
## ENSMUSG00000089635                                           9.350518
```

Check the HTML report generated in the output directory defined by the export.where argument above (out.dir variable).

Now, the same example but with more than one statistical selection algorithms, a different normalization, an analysis preset and filtering applied prior to normalization:

```
library(metaseqR)
data("mm9.gene.data",package="metaseqR")
out.dir2 <- tempdir()
print(out.dir2)
```

```
## [1] "/tmp/RtmpREyzeR"
```

```
result <- metaseqr(
       counts=mm9.gene.counts,
       sample.list=sample.list.mm9,
       contrast=c("e14.5_vs_adult_8_weeks"),
       libsize.list=libsize.list.mm9,
       annotation="download",
       org="mm9",
       count.type="gene",
       when.apply.filter="prenorm",
       normalization="edaseq",
       statistics=c("deseq","edger"),
       qc.plots=c(
           "mds","filtered","correl","pairwise","boxplot","gcbias",
           "lengthbias","meandiff","meanvar","deheatmap","volcano"
       ),
       meta.p="fisher",
       fig.format=c("png","pdf"),
       preset="medium.normal",
       export.where=out.dir2,
       restrict.cores=0.1,
       out.list=TRUE
)
```

```
##
## 2020-10-27 21:32:55:  Data processing started...
##
## Read counts file:  imported custom data frame
## Conditions:  e14.5, adult_8_weeks
## Samples to include:  e14.5_1, e14.5_2, a8w_1, a8w_2
## Samples to exclude:  none
## Requested contrasts:  e14.5_vs_adult_8_weeks
## Library sizes:
##  e14.5_1:  3102907
##  e14.5_2:  2067905
##  a8w_1:  3742059
##  a8w_2:  4403954
## Annotation:  download
## Organism:  mm9
## Reference source:  ensembl
## Count type:  gene
## Analysis preset:  medium.normal
## Transcriptional level:  gene
## Exon filters:  min.active.exons
##  min.active.exons:
##    exons.per.gene:  5
##    min.exons:  2
##    frac:  0.2
## Gene filters:  length, avg.reads, expression, biotype
##  length:
##    length:  500
##  avg.reads:
##    average.per.bp:  100
##    quantile:  0.25
##  expression:
##    median:  TRUE
##    mean:  FALSE
##    quantile:  NA
##    known:  NA
##    custom:  NA
##  biotype:
##    pseudogene:  FALSE
```

```
##    snRNA: FALSE
##    protein_coding:  FALSE
##    antisense:  FALSE
##    miRNA: FALSE
##    lincRNA: FALSE
##    snoRNA: FALSE
##    processed_transcript:  FALSE
##    misc_RNA: FALSE
##    rRNA: TRUE
##    sense_overlapping:  FALSE
##    sense_intronic:  FALSE
##    polymorphic_pseudogene:  FALSE
##    non_coding:  FALSE
##    three_prime_overlapping_ncrna:  FALSE
##    IG_C_gene:  FALSE
##    IG_J_gene:  FALSE
##    IG_D_gene:  FALSE
##    IG_V_gene:  FALSE
##    ncrna_host:  FALSE
## Filter application:  prenorm
## Normalization algorithm:  edaseq
## Normalization arguments:
##  within.which:  loess
##  between.which:  full
## Statistical algorithm:  deseq, edger
## Statistical arguments:
##  deseq:  blind, fit-only, local
##  edger:  classic, 5, 10, movingave, NULL, grid, 11, c(-6, 6), NULL, CoxReid, 10000, NULL, auto,
NULL, NULL, NULL, NULL, 0.125, NULL, auto, chisq, TRUE, FALSE, c(0.05, 0.1)
## Meta-analysis method:  fisher
## Multiple testing correction:  BH
## p-value threshold:  0.05
## Logarithmic transformation offset:  1
## Analysis preset:  medium.normal
## Quality control plots:  mds, filtered, correl, pairwise, boxplot, gcbias, lengthbias, meandiff,
meanvar, deheatmap, volcano
## Figure format:  png, pdf
## Output directory:  /tmp/RtmpREyzeR
## Output data:  annotation, p.value, adj.p.value, meta.p.value, adj.meta.p.value, fold.change,
stats, counts
## Output scale(s):  natural, log2
## Output values:  normalized
## Output statistics:  mean, sd, cv
##
## Downloading gene annotation for mm9...
## Saving gene model to /tmp/RtmpREyzeR/data/gene_model.RData
## Removing genes with zero counts in all samples...
## Prefiltering normalization with:  edaseq
## Applying gene filter length...
##  Threshold below which ignored:  500
## Applying gene filter avg.reads...
##  Threshold below which ignored:  0.0983646994748505
## Applying gene filter expression...
##  Threshold below which ignored:  101.5
## Applying gene filter biotype...
##  Biotypes ignored:  rRNA
## Normalizing with:  edaseq
## 2110 genes filtered out
## 1677 genes remain after filtering
## Running statistical tests with:  deseq
```

```
##  Contrast:  e14.5_vs_adult_8_weeks
##  Contrast e14.5_vs_adult_8_weeks:  found 333 genes
## Running statistical tests with:  edger
##  Contrast:  e14.5_vs_adult_8_weeks
##  Contrast e14.5_vs_adult_8_weeks:  found 925 genes
## Performing meta-analysis with fisher
## Building output files...
##  Contrast:  e14.5_vs_adult_8_weeks
##    Adding non-filtered data...
##      binding annotation...
##      binding p-values...
##      binding FDRs...
##      binding meta p-values...
##      binding adjusted meta p-values...
##      binding natural normalized fold changes...
##      binding log2 normalized fold changes...
##      binding normalized mean counts...
##      binding normalized count sds...
##      binding normalized count CVs...
##      binding normalized mean counts...
##      binding normalized count sds...
##      binding normalized count CVs...
##      binding all normalized counts for e14.5...
##      binding all normalized counts for adult_8_weeks...
##    Writing output...
##    Adding filtered data...
##      binding annotation...
##      binding p-values...
##      binding FDRs...
##      binding meta p-values...
##      binding adjusted meta p-values...
##      binding natural normalized fold changes...
##      binding log2 normalized fold changes...
##      binding normalized mean counts...
##      binding normalized count sds...
##      binding normalized count CVs...
##      binding normalized mean counts...
##      binding normalized count sds...
##      binding normalized count CVs...
##      binding all normalized counts for e14.5...
##      binding all normalized counts for adult_8_weeks...
##    Writing output...
## Creating quality control graphs...
## Plotting in png format...
##  Plotting mds...
##  Plotting correl...
##  Plotting pairwise...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting deheatmap...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting volcano...
```

```
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting filtered...
## Plotting in pdf format...
##  Plotting mds...
##  Plotting correl...
##  Plotting pairwise...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting boxplot...
##  Plotting gcbias...
##  Plotting lengthbias...
##  Plotting meandiff...
##  Plotting meanvar...
##  Plotting deheatmap...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting volcano...
##  Contrast:  e14.5_vs_adult_8_weeks
##  Plotting filtered...
## Creating HTML report...
## Compressing figures...
##
## 2020-10-27 21:33:19:  Data processing finished!
##
##
## Total processing time:  24 seconds
```

A similar example with no filtering applied and no Venn diagram generation (not evaluated here):

```
library(metaseqR)
data("mm9.gene.data",package="metaseqR")
out.dir <- tempdir()
print(out.dir)
result <- metaseqr(
       counts=mm9.gene.counts,
       sample.list=sample.list.mm9,
       contrast=c("e14.5_vs_adult_8_weeks"),
       libsize.list=libsize.list.mm9,
       annotation="download",
       org="mm9",
       count.type="gene",
       normalization="edaseq",
       statistics=c("deseq","edger"),
       meta.p="fisher",
       fig.format=c("png","pdf"),
       preset="medium.normal",
       out.list=TRUE,
       export.where=out.dir
)
```

An additional example with human exon data (if you have a multiple core system, be very careful on how you are using the restrict.cores option and generally how many cores you are using with scripts purely written in R. The analysis with exon read data can very easily cause memory problems, so unless you have more than 64Gb of RAM available, consider setting restrict.cores to something like 0.2):

```
# A full example pipeline with exon counts
data("hg19.exon.data",package="metaseqR")
out.dir <- tempdir()
print(out.dir)
```

```
metaseqr(
       counts=hg19.exon.counts,
       sample.list=sample.list.hg19,
       contrast=c("normal_vs_paracancerous","normal_vs_cancerous",
          "normal_vs_paracancerous_vs_cancerous"),
       libsize.list=libsize.list.hg19,
       id.col=4,
       annotation="download",
       org="hg19",
       count.type="exon",
       normalization="edaseq",
       statistics="deseq",
       pcut=0.05,
       qc.plots=c(
             "mds","biodetection","countsbio","saturation","rnacomp","pairwise",
             "boxplot","gcbias","lengthbias","meandiff","meanvar","correl",
             "deheatmap","volcano","biodist","filtered"
       ),
       fig.format=c("png","pdf"),
       export.what=c("annotation","p.value","adj.p.value","fold.change","stats","counts"),
       export.scale=c("natural","log2","log10","vst"),
       export.values=c("raw","normalized"),
       export.stats=c("mean","median","sd","mad","cv","rcv"),
       restrict.cores=0.8,
       gene.filters=list(
             length=list(
                    length=500
             ),
             avg.reads=list(
                    average.per.bp=100,
                    quantile=0.25
             ),
             expression=list(
                    median=TRUE,
                    mean=FALSE
             ),
             biotype=get.defaults("biotype.filter","hg19")
       ),
       export.where=out.dir
)
```

or in a more simplified version

```
# A full example pipeline with exon counts
data("hg19.exon.data",package="metaseqR")
out.dir <- tempdir()
print(out.dir)
metaseqr(
       counts=hg19.exon.counts,
       sample.list=sample.list.hg19,
       contrast=c("normal_vs_paracancerous","normal_vs_cancerous",
          "normal_vs_paracancerous_vs_cancerous"),
       libsize.list=libsize.list.hg19,
       id.col=4,
       annotation="download",
       org="hg19",
       count.type="exon",
       normalization="edaseq",
       statistics="deseq",
       preset="medium.normal",
       restrict.cores=0.8,
```

```
       export.where=out.dir
)
```

One of the main strong points of metaseqR is the use of the area under False Discovery Curves to assess the performance of each statistical test with simulated datasets created from true datasets (e.g. your dataset). Then, the performance assessment can be used to construct p-value weights for each test and use these weights to supply the "weight" parameter of metaseqr when "meta.p" is "weight" or "whitlock" (see the next sections for p-value combination methods). The following example shows how to create such weights (depending on the size of the dataset, it might take some time to run):

```
data("mm9.gene.data",package="metaseqR")
weights <- estimate.aufc.weights(
    counts=as.matrix(mm9.gene.counts[,9:12]),
    normalization="edaseq",
    statistics=c("edger","limma"),
    nsim=1,N=10,ndeg=c(2,2),top=4,model.org="mm9",
    seed=42,multic=FALSE,libsize.gt=1e+5
)

## Loading required package:  zoo
##
## Attaching package:  'zoo'
## The following objects are masked from 'package:Rsamtools':
##
##    index, index<-
## The following objects are masked from 'package:base':
##
##    as.Date, as.Date.numeric
##
## Downsampling counts...
## Estimating initial dispersion population...
## Estimating dispersions using log-likelihood...
##
## Running simulations...  This procedure requires time...  Please wait...
## 2020-10-27 21:33:25:  Data processing started...
##
## Read counts file:  imported custom data frame
## Conditions:  G1, G2
## Samples to include:  G1_rep1, G1_rep2, G1_rep3, G2_rep1, G2_rep2, G2_rep3
## Samples to exclude:  none
## Requested contrasts:  G1_vs_G2
## Annotation:  embedded
## Organism:  mm9
## Reference source:  ensembl
## Count type:  gene
## Analysis preset:  all.basic
## Transcriptional level:  gene
## Exon filters:  none applied
## Gene filters:  none applied
## Filter application:  postnorm
## Normalization algorithm:  edaseq
## Normalization arguments:
##  within.which:  loess
##  between.which:  full
## Statistical algorithm:  edger, limma
## Statistical arguments:
##  edger:  classic, 5, 10, movingave, NULL, grid, 11, c(-6, 6), NULL, CoxReid, 10000, NULL, auto,
NULL, NULL, NULL, NULL, 0.125, NULL, auto, chisq, TRUE, FALSE, c(0.05, 0.1)
##  limma:  none
## Meta-analysis method:  simes
## Multiple testing correction:  BH
```

```
## Logarithmic transformation offset:  1
## Analysis preset:  all.basic
## Quality control plots:
## Figure format:  png
## Output directory:  /tmp/RtmpREyzeR
## Output data:  annotation, p.value, adj.p.value, meta.p.value, adj.meta.p.value, fold.change
## Output scale(s):  natural, log2
## Output values:  normalized
## Saving gene model to /tmp/RtmpREyzeR/data/gene_model.RData
## Removing genes with zero counts in all samples...
## Normalizing with:  edaseq
## Warning:  'exprs' is deprecated.
## Use 'counts' instead.
## See help("Deprecated")
##
## Running statistical tests with:  edger
##  Contrast:  G1_vs_G2
## Running statistical tests with:  limma
##  Contrast:  G1_vs_G2
## Performing meta-analysis with simes
## Building output files...
##  Contrast:  G1_vs_G2
##    Adding non-filtered data...
##      binding annotation...
##      binding p-values...
##      binding FDRs...
##      binding meta p-values...
##      binding adjusted meta p-values...
##      binding natural normalized fold changes...
##      binding log2 normalized fold changes...
##    Writing output...
##
## 2020-10-27 21:33:25:  Data processing finished!
##
##
## Total processing time:  00 seconds
##
##
## Estimating AUFC weights...  Please wait...
## Processing edger
## Processing limma
##
## Retrieving edger
## Retrieving limma
```

...and the weights...

```
weights

## edger limma
##   0.5   0.5
```

3 metaseqR components

The metaseqR package includes several functions which are responsible for running each part of the pipeline (data reading and summarization, filtering, normalization, statistical analysis and meta-analysis and reporting). Although metaseqR is designed to run as a pipeline, where all the parameters for each individual part can be passed in the main function, several of the individual functions can be run separately so that the more experienced user can build custom pipelines. All the HTML help pages contain analytical documentation on how to run these functions, their inputs and outputs and contain basic examples. For example, runnning

```
help(stat.edgeR)
```

will open the help page of the wrapper function over the edgeR statistical testing algorithm which contains an example of data generation, processing, up to statistical selection.

Most of the diagnostic plots, work with simple matrices as inputs, so they can be easily used outside the main pipeline, as long as all the necessary arguments are given. It should be noted that a report can be generated only when running the whole metaseqr pipeline and in the current version there is no support for generating custom reports.

A very detailed documentation on how to run metaseqr and explanation for all its parameters can be obtained by

```
help(metaseqr)
```

4 Details on metaseqR's components

4.1 Data input

The metaseqR package currently supports three methods of data input:

1. Aligned reads in SAM/BAM or BED format. In this case, the input files are passed to the metaseqr pipeline through a simply structured tab-delimited text file. This file contains in its first column unique names that are used to identify each sample, the SAM/BAM/ BED file names (preferably with their full path) in the second column, and the biological conditions/groups for each sample/file in the third column. The columns may or may not be named, as this information is not used by metaseqR. The above order (sample names, file names, biological condition) is used instead. This is the preferred method of data input as in this way, the analysis is streamlined within R from the beginning until the end, ensuring data integrity (e.g. the compatibility of the genome annotations used, something that can sometimes be broken when using external read counting software). SAM/BAM files are imported through the Bioconductor packages Rsamtools and GenomicRanges and BED files are imported through the Bioconductor package rtracklayer. The aligned reads are converted to a read counts table through facilities provided in the Bioconductor package GenomicRanges. The Ensembl/UCSC/RefSeq gene or exon regions which are used to summarize the read alignments and create the final read counts table for each gene or exon can be obtained either through downloading at the time of usage using the Bioconductor packages biomaRt/RMySQL, or from a user-specified file.

2. Summarized numbers of reads for each genomic feature (gene or exon) by providing a file containing the read counts table. This file should contain at least: a) a column with a unique identifier for each gene/exon (currently, Ensembl, UCSC and RefSeq identifiers are supported unless the required annotation elements for each genomic region are embedded in the file), b) as many columns as the number of samples in the experiment. Each column with sample read counts should be named with a unique sample name. Optionally, the read counts file can contain annotation elements for each genomic feature (gene or exon) corresponding to the unique identifier column. In this case, the pipeline may be executed with the annotation="embedded" argument and it is very useful when the user does not wish to use supported annotations or the organisms under investigation is not supported by metaseqR. If the user chooses to use embedded to the read counts file annotation, then at least the following elements should be provided for each genomic region, apart from its unique name should be provided (in parenthesis next to each element, the required column name): chromosome where the genomic feature is located (chromosome), starting base pair in the chromosome (start), ending base pair in the chromosome (end). For best performance (e.g. availability of all quality control and diagnostic plots), the following annotation elements should also be provided: GC content (gc_content) for genomic features of type "gene", the gene model name (gene_name) for genomic features of type "exon", the transcribing strand of each feature, denoted as "+" or "-" (strand), HUGO (or other alias) gene name (gene_name) and each genomic features biotype/biological categorization, for example Ensembl categorizations like "protein_coding", "ncRNA", "pseudogene", etc. (biotype). In any case, if the user needs are satisfied with Ensembl annotation, it is better practice not to use embedded annotation in the counts file but download it or use the embedded in the package annotation using the respective options.

3. Like case (2) but all the data mentioned in (2) are stored in an R data frame object (for example, derived after a user-defined or otherwise customized preprocessing).

In any case of the above, some of the main input arguments to the pipeline can become mutually exclusive. For example the user cannot supply both an input read counts table and a file with targets including sample filenames. Details on such issues can be found in the package documentation.

4.2 Data filtering

Two optional data filter types are implemented in the metaseqR package, operating at the exon (when exon counts are requested or provided to/from the pipeline) or the gene level (applied after summarizing exon counts to gene counts when exon counts are provided, or applied on gene counts if only a gene read counts table is provided). It should also be noted that, like the metaseqr pipeline, these filters are created with only the detection of differential expression at the gene level and not at the exon level or the detection of differential splicing.

Exon filters

Currently only one exon filter is implemented in metaseqr. This filter excludes genes that do not have enough reads presence in a fraction of their exons. This fraction should not be too high so that to avoid excluding genes that are possibly simply differentially spliced (although metaseqR is not intended to detect differential splicing), nor too low so as to be able to exclude artifacts. This filtered was inspired by the fact that certain genes contain "spikes" of read data in their UTR regions or a couple of their exons. These spikes are usually artifacts that can affect the subsequent differential expression analysis. The exon filter has three parameters: exons.per.gene, min.exons and frac and is applied as follows: if a gene has up to exons.per.gene exons, then read presence is required in at least min.exons of them, else read presence is required in a frac fraction of the total exons in the gene mode. With the default values, the filter instructs that if a gene has up to 5 exons, read presence is required in at least 2, else in at least 20% of its exons, in order to be included in further analysis. More filters will be implemented in future versions and users are encouraged to propose exon filter ideas.

After the determination of the genes that will be filtered from further processing (normalization and/or statistical analysis), a gene model expression value is constructed based on the sum of all exons of an annotated Ensembl gene. It should be noted that while this particular way of summarizing a gene expression value is not recommended in applications where for example, one studies differential splicing, differential isoform expression or differential exon usage, it is sufficient for most applications where only expression of a gene as a total is the goal of the study. Thus, it should be sufficient for a majority of related projects, where the researchers are interested in summarized gene expression values.

Gene filters

The gene filters are a set of filters applied to gene expression as this is manifested through the read presence on each gene and can be applied before or after normalization. While for some categories this is not important (e.g. gene length filter), for others (e.g. expression filters), the application prior to or post normalization is important as any expression filter must be applied to normalized data. In that case, metaseqr performs two rounds of normalization. The first round serves as a temporary normalization in order to get normalized expression values. The expression filters are then applied there and genes not passing the filters are excluded from the second and final round of normalization. The gene filters can be applied both when the pipeline input consists of exon read counts and gene read counts. Such filter can for example be verbalized to "accept all genes above a certain count threshold" or "accept all genes with expression above the median of the normalized counts distribution" or "accept all genes with length above a certain threshold in kb" or "exclude the 'pseudogene' biotype from further analysis". Currently, there are four categories of gene filters. The first category is a qualitative filter, specifically a gene length filter where genes are accepted for further analysis if they are above a certain (the filter parameter) kb. The second category consists of a combined qualitative/quantitative filter where a gene is accepted for further analysis if it has more average reads per x bp than the quantile of the average normalized count distribution per x bp in the gene body. This filter avg.reads has two parameters: average.per.bp expressing the number of base pairs for which reads are summarized and quantile for the quantile of the averaged normalized count distribution. The latter quantiles are calculated for each normalized sample and genes passing the filter should have an average read count larger than the maximum of the quantiles vector calculated above. The third category consists of a set of expression filters which can be applied altogether or only a subset of them. The expression filters are the following:

1. a global median filter, where genes below the median of the global normalized count distribution in all samples are not accepted for further analysis (this filter has been used to distinguish between "expressed" and "not expressed" genes in several cases, e.g. (Mokry et al., NAR, 2011). The value of this filter is a boolean, TRUE for applying this filter and FALSE for not applying.

2. a global mean filter, similar as the global median filter but using the global mean.

3. a global quantile filter, which is the same as the previous two but using a specific quantile of the total counts distribution

4. a filter based on the expression of genes known to be specifically expressed (e.g. not expressed) under the system under investigation. In this case, a set of known not-expressed genes is used to estimate an expression cutoff. This can be quite useful, as the genes are filtered based on *a true biological cutoff* instead of a statistical cutoff. The value of this filter is a character vector of HUGO gene symbols (which must be contained in the annotation, see previous section). Thus, if the user intends to use this filter, it is better to instruct metaseqr to download annotation on the fly or use the annotations embedded in the package. Then, these genes are used to build a "null" expression distribution. The 90^{th} quantile of this distribution is then the expression cutoff. This filter can be combined with any other filter. The user should be careful with gene names as they are case sensitive and must match exactly ("Pten" is different from "PTEN").

The fourth filter category is a qualitative filter based on the biological categorization of each gene (for example using Ensembl biotypes). In this case, genes with a certain biotype (which must be contained in the annotation) are excluded from the analysis.

4.3 Data normalization

The metaseqR package currently supports five count data normalization algorithms from five different RNA-Seq analysis Bioconductor packages. Each package may provide additional options for normalization (e.g. more than one normalization algorithm present in a package, for example the NOISeq package), which can be controlled through normalization options (norm.opts parameter) passed to the pipeline call. The initial normalization parameters are the default parameters provided by the authors of each package. Specifically, metaseqR supports normalization with the EDASeq package which is the default option, with the edgeR package, with the NOISeq package and the NBPSeq package. There is also an option to not normalize the data (not recommended). Popular normalization methods (e.g. RPKM normalization) are not directly supported as they are coded within the current package and they can be used by changing the normalization parameters passed to the pipeline. For example, RPKM normalization can be performed with the NOISeq package, although not currently recommended due to RPKM limitations discussed in several articles.

Data normalization can be performed before or after data filtering. The issue of applying normalization to the total dataset or to a filtered instance of the dataset is not sufficiently treated in the relative literature and there is not a definite answer. Our own experience suggests that normalization is smoother and subsequently, the statistical analysis more accurate, when normalization is applied after filtering. In the case of pre-normalization filtering, a first round of normalization is applied as certain thresholds (e.g. gene expression thresholds) must be defined based on the global distributions of normalized data, otherwise, biases present in individual samples will cause confusion. For example, if the user filters data below a specific quantile of the normalized count distribution, if that quantile is not determined based on normalized data, it will not be representative of the global data distribution but it will comprise a biased estimation depended on the initial count distribution of the un-normalized samples. After the first normalization round, filters are applied and the filtered un-normalized data are normalized again. In the case of post-normalization filtering, data are firstly normalized and then filtered. In any case (pre- or post-normalization filtering), genes with zero counts across all samples are removed prior to normalization (as also suggested by most package authors).

4.4 Statistical testing

The metaseqR package currently supports six statistical testing algorithms developed for RNA-Seq data. The algorithms supported are the testing procedures in the Bioconductor packages DESeq, edgeR, NOISeq, baySeq, limma and NBPSeq. The arguments required for each statistical testing algorithm are passed by the metaseqr pipeline through the argument stat.opts (please refer to the package documentation for instructions on how to use the argument). The default options for each algorithm are the same as the corresponding default arguments used by the authors of each package. The default algorithm used by metaseqR is DESeq.

4.5 Meta analysis

When analyzing data with metaseqR, the user may use more than one statistical testing method (any combination of the six currently supported). In this case, metaseqR will perform meta-analysis on the p-values that

will be returned from each of the applied statistical tests and will also report, apart from the p-value from each method, a combined p-value using one of the following methods:

1. The Simes p-value combination method. This method uses the minimum ordered p-value from all methods divided by the inverse order of the p-values. The ordering is performed across the number of statistics used. This is the default method.

2. the Fisher p-value combination meta-analysis method, implemented in the R package MADAM. This is the default method.

3. same as (2) but using permutations, as implemented in the R package MADAM. This option is quite computationally intensive and requires additional running time.

4. the Whitlock p-value combination method, implemented in the Bioconductor package survcomp. This method has the advantage of allowing weighting for each methodology. In the current version of metaseqR, the weights are equal for all statistical tests. For example, the user may estimate weights for his/her dataset using metaseqR facilities for this purpose.

5. The maximum p-value returned by a set of statistical tests for the same gene. This is equivalent to the "intersection" of the results derived from each statistical test, returning genes which have been found as statistically significant by all the statistical tests applied. The maximum p-value ensures that the false positives are minimized at a (usually high) cost on the true positives (statistical power).

6. The minimum p-value returned by a set of statistical tests for the same gene. This is equivalent to the "union" of the results derived from each statistical test, returning genes which have been found as statistically significant by at least one of the statistical tests applied. The minimum p-value ensures that the true positives are maximized at a (usually high) cost on the false positives (type I error).

7. A weighted p-value where the weights can be either fixed (e.g. equal or set by the user according to performance evidence from the literature), or estimated using metaseqR's facilities for this purpose (simulation based on the user data and weighting according to performance measurement on simulated data). The weights must sum to 1.

8. A method based on permutations. This method has three variants:

   (a) In the first variant (dperm.min), an initial p-value vector is constructed for each gene, containing the minimum p-value resulted from the applied statistical tests (more lose, see above).

   (b) In the second variant (dperm.max), an initial p-value vector is constructed for each gene, containing the maximum p-value resulted from the applied statistical tests (more strict, see above).

   (c) In the third variant (dperm.weight), an initial p-vale vector is constructed for each gene, containing the convex linear combination of the p-values resulted from all the statistical tests applied for each gene. To construct the convex linear combination, a vector of weights is used, one for each statistical test, and the sum of all weights must be 1.

   After the construction of the original combined p-value with one of the aforementioned variants a permutation procedure is initialed, where nperm permutations are performed across the samples of the normalized counts matrix, producing nperm permuted instances of the initial dataset. Then, all the chosen statistical tests are re-executed for each permutation. The final p-value is the number of times that the p-value of the permuted datasets is smaller than the original dataset. The p-value of the original dataset is created based on the choice of one. Generally, the permutation procedure usually requires a lot of time in order to yield accurate results (at least 10000 iterations especially in smaller datasets). Additionally, this method will NOT work when there are no replicated samples across biological conditions. In that case, the Simes method or one of the others should be used.

It should be noted that in the case of NOISeq, the significance value returned is not similar to the "classic" t-test like statistics, thus its inclusion to a meta-analysis should be handled and interpreted with caution. It should also be noted that the meta-analysis feature provided by metaseqR is currently experimental and does not satisfy the strict definition of "meta-analysis", which is the combination of multiple similar datasets under the same statistical methodology. Instead it is the use of multiple statistical tests applied to the same data so the results at this point are not guaranteed and should be interpreted appropriately. We are working on a more solid methodology for combining multiple statistical tests based on multiple testing correction and Monte Carlo methods.

5 References

1. Anders, S., and Huber, W. (2010). Differential expression analysis for sequence count data. Genome Biol 11, R106.

2. Benjamini, Y., and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society Series B (Methodological) 57, 289-300.

3. Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. Annals of Statistics 26, 1165-1188.

4. Chen, H., and Boutros, P.C. (2011). VennDiagram: a package for the generation of highly-customizable Venn and Euler diagrams in R. BMC Bioinformatics 12, 35.

5. Di, Y, Schafer, D. (2012): NBPSeq: Negative Binomial Models for RNA-Sequencing Data. R package version 0.1.8, http://CRAN.R-project.org/package=NBPSeq.

6. Fisher, R.A. (1932). Statistical Methods for Research Workers (Edinburgh, Oliver and Boyd).

7. Hardcastle, T.J., and Kelly, K.A. (2010). baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. BMC Bioinformatics 11, 422.

8. Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. Biometrika 75, 800-803.

9. Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65-70.

10. Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. Biometrika 75, 383-386.

11. Leng, N., Dawson, J.A., Thomson, J.A., Ruotti, V., Rissman, A.I., Smits, B.M., Haag, J.D., Gould, M.N., Stewart, R.M., and Kendziorski, C. (2013). EBSeq: an empirical Bayes hierarchical model for inference in RNA-seq experiments. Bioinformatics 29, 1035-1043

12. Planet, E., Attolini, C.S., Reina, O., Flores, O., and Rossell, D. (2012). htSeqTools: high-throughput sequencing quality control, processing and visualization in R. Bioinformatics 28, 589-590.

13. Risso, D., Schwartz, K., Sherlock, G., and Dudoit, S. (2011). GC-content normalization for RNA-Seq data. BMC Bioinformatics 12, 480.

14. Robinson, M.D., McCarthy, D.J., and Smyth, G.K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139-140.

15. Schroder, M.S., Culhane, A.C., Quackenbush, J., and Haibe-Kains, B. (2011). survcomp: an R/Bioconductor package for performance assessment and comparison of survival models. Bioinformatics 27, 3206-3208.

16. Shaffer, J.P. (1995). Multiple hypothesis testing. Annual Review of Psychology 46, 561-576.

17. Simes, R. J. (1986). An improved Bonferroni procedure for multiple tests of significance. Biometrika 73 (3): 751-754.

18. Smyth, G. (2005). Limma: linear models for microarray data. In Bioinformatics and Computational Biology Solutions using R and Bioconductor, G. R., C. V., D. S., I. R., and H. W., eds. (New York, Springer), pp. 397-420.

19. Storey, J.D., and Tibshirani, R. (2003). Statistical significance for genomewide studies. Proc Natl Acad Sci U S A 100, 9440-9445.

20. Tarazona, S., Garcia-Alcalde, F., Dopazo, J., Ferrer, A., and Conesa, A. (2011). Differential expression in RNA-seq: a matter of depth. Genome Res 21, 2213-2223.

21. Whitlock, M.C. (2005). Combining probability from independent tests: the weighted Z-method is superior to Fisher's approach. J Evol Biol 18, 1368-1373.

6 R session information

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.5 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.12-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.12-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] zoo_1.8-8                   metaseqR_1.30.0
##  [3] qvalue_2.22.0               limma_3.46.0
##  [5] DESeq_1.42.0                lattice_0.20-41
##  [7] locfit_1.5-9.4              EDASeq_2.24.0
##  [9] ShortRead_1.48.0            GenomicAlignments_1.26.0
## [11] SummarizedExperiment_1.20.0 MatrixGenerics_1.2.0
## [13] matrixStats_0.57.0          Rsamtools_2.6.0
## [15] GenomicRanges_1.42.0        GenomeInfoDb_1.26.0
## [17] Biostrings_2.58.0           XVector_0.30.0
## [19] IRanges_2.24.0              S4Vectors_0.28.0
## [21] BiocParallel_1.24.0         Biobase_2.50.0
## [23] BiocGenerics_0.36.0
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1       rjson_0.2.20           hwriter_1.3.2
##  [4] ellipsis_0.3.1         affyio_1.60.0          bit64_4.0.5
##  [7] AnnotationDbi_1.52.0   xml2_1.3.2             splines_4.0.3
## [10] R.methodsS3_1.8.1      geneplotter_1.68.0     knitr_1.30
## [13] log4r_0.3.2            annotate_1.68.0        baySeq_2.24.0
## [16] vsn_3.58.0             dbplyr_1.4.4           png_0.1-7
## [19] R.oo_1.24.0            BiocManager_1.30.10    compiler_4.0.3
## [22] httr_1.4.2             assertthat_0.2.1       Matrix_1.2-18
## [25] prettyunits_1.1.1      tools_4.0.3            gtable_0.3.0
## [28] glue_1.4.2             GenomeInfoDbData_1.2.4 affy_1.68.0
## [31] reshape2_1.4.4         dplyr_1.0.2            rappdirs_0.3.1
## [34] Rcpp_1.0.5             vctrs_0.3.4            preprocessCore_1.52.0
## [37] rtracklayer_1.50.0     xfun_0.18              stringr_1.4.0
## [40] lifecycle_0.2.0        gtools_3.8.2           XML_3.99-0.5
## [43] edgeR_3.32.0           zlibbioc_1.36.0        scales_1.1.1
## [46] aroma.light_3.20.0     hms_0.5.3              RColorBrewer_1.1-2
## [49] curl_4.3               memoise_1.1.0          NBPSeq_0.3.0
## [52] ggplot2_3.3.2          biomaRt_2.46.0         latticeExtra_0.6-29
## [55] stringi_1.5.3          RSQLite_2.2.1          highr_0.8
## [58] genefilter_1.72.0      corrplot_0.84          GenomicFeatures_1.42.0
## [61] caTools_1.18.0         rlang_0.4.8            pkgconfig_2.0.3
## [64] bitops_1.0-6           evaluate_0.14          purrr_0.3.4
## [67] bit_4.0.4              tidyselect_1.1.0       plyr_1.8.6
```

```
## [70] magrittr_1.5           R6_2.4.1               gplots_3.1.0
## [73] generics_0.0.2         DelayedArray_0.16.0    DBI_1.1.0
## [76] pillar_1.4.6           survival_3.2-7         abind_1.4-5
## [79] RCurl_1.98-1.2         tibble_3.0.4           crayon_1.3.4
## [82] KernSmooth_2.23-17     BiocFileCache_1.14.0   jpeg_0.1-8.1
## [85] progress_1.2.2         grid_4.0.3             blob_1.2.1
## [88] digest_0.6.27          xtable_1.8-4           brew_1.0-6
## [91] R.utils_2.10.1         openssl_1.4.3          munsell_0.5.0
## [94] NOISeq_2.34.0          askpass_1.1
```