RNAseq Differential Expression Analysis

Jana Schor and Jörg Hackermüller

November, 2017

RNA-Seq: Example dataset and download}

Dataset taken from Somel et al. 2012: (check publication for details)

- 3 human + 3 chimp brain transcriptome data
- Illumina RNA-Seq, paired-end 76bp
- Reads mapped to human and chimp genome and reads counted for each transcript

Download the following data files into a directory of your choice (e.g. data):

- Human: http://www.nowick-lab.info/wp-content/uploads/2013/12/segemehl.hg19.readCount.txt
- Chimp: http://www.nowick-lab.info/wp-content/uploads/2013/12/segemehl.panTro3.readCount.txt

Read dataset into R

```
# filenames
HreadCounts <- "data/segemehl.hg19.readCount.txt"
CreadCounts <- "data/segemehl.panTro3.readCount.txt"
# counts
Hcounts <- read.table(HreadCounts, head=T, sep="\t", quote="", stringsAsFactor=FALSE, row.names="id")
Ccounts <- read.table(CreadCounts, head=T, sep="\t", quote="", stringsAsFactor=FALSE, row.names="id")
```

Task 01:

- examine `Hcounts` and `Ccounts` to know what you got
- show the first 10 lines of `Hcounts`

Prepare dataset in R

What do we have in our datasets?

- **Columns:** samples **Colnames:** sample names
- **Rows:** genes **Rownames:** human RefSeq IDs for orthologous genes

Now we combine the 2 datasets into a single table and appropriately assign (short) columnnames. Before we do this we need to check if the rownames are the same and if they are in the same order.

```
length(rownames(Hcounts))
```

```
## [1] 38439
```

```
length(rownames(Hcounts)) == length(rownames(Ccounts))
```

```
## [1] TRUE
```

```
sum(rownames(Hcounts) != rownames(Ccounts))
```

```
## [1] 3780
```

```
# => same number of genes, but different names OR different order OR both
indH <- order(rownames(Hcounts))
indC <- order(rownames(Ccounts))
sum(rownames(Hcounts[indH,]) != rownames(Ccounts[indC,]))
```

```
## [1] 0
```

Now we found a way to sort both datasets such that we don't have any differences anymore in the rownames. This means we have the same names in both datasets, which are now in the same order! We can combine both datasets into a single table.

```
counts <- cbind(Hcounts[indH,], Ccounts[indC,])
colnames(counts) <- c("H1", "H2", "H3", "C1", "C2", "C3")
head(counts)
```

```
##             H1    H2    H3    C1   C2    C3
## NM_000014 1148  3261  3665  2608 4724  2016
## NM_000015    6     4     2    68   66    92
## NM_000016 4911  2254  1774  2199 2406  2235
## NM_000017  145   316   508   631  880   894
## NM_000018 5698 15003 15356 10701 7518 10458
## NM_000019 3776  1232  1486  2678 2470  2051
```

We define the attribute species as condition for this example.

```
condition <- c("h", "h", "h", "c", "c", "c")
colData <- data.frame(condition, row.names = colnames(counts))
```

Task 02:

- show the content of `colData`, what does it mean?

What do we have for now?

- a table containing the expression of `38439` genes in `6` samples
- a table assigning a condition to each of the samples

Install and load packages and workflow

All information about the package that we use can be found here: https://www.bioconductor.org/packages/release/bioc/html/DESeq2.html

An example workflow for such an analysis can be found here: https://www.bioconductor.org/help/workflows/rnaseqGene/

At the beginning we need to install and load the packages that we need for our analysis.

- **DESeq2** for differential expression analysis of RNAseq data.
- **pheatmap** for drawing nice heatmaps
- **RColorBrewer** for using nice color palettes

To install the package use:

```
# Bioconductor packages
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
```

```
if(! "DESeq2" %in% installed.packages()) { biocLite("DESeq2") }
# R packages
if(! "pheatmap" %in% installed.packages()) { install.packages("pheatmamp") }
if(! "RColorBrewer" %in% installed.packages()) { install.packages("RColorBrewer") }
if(! "ggplot2" %in% installed.packages()) { install.packages("ggplot2") }
```

Now we can load these packages:

```
library(DESeq2)
library(pheatmap)
library(RColorBrewer)
library(ggplot2)
```

Based on our - count matrix - the phenotype information (sample to condition assignment), and - the experimental design **condition** we can create a DESeq2 dataset:

```
dds0 <- DESeqDataSetFromMatrix(countData = counts,
                               colData = colData,
                               design = ~ condition)
dds0
```

```
## class: DESeqDataSet
## dim: 38439 6
## metadata(1): version
## assays(1): counts
## rownames(38439): NM_000014 NM_000015 ... NR_046426 NR_046427
## rowData names(0):
## colnames(6): H1 H2 ... C2 C3
## colData names(1): condition
```

Task 03:

- What is a `DESeqDataSet`?
- How can we access the raw count matrix from this data set?

Pre-filtering

In general, it is not necessary to filter the data for the analysis. However, it will - reducde the memory size - speed up the transformations and calculations

```
dds <- dds0[ rowSums(counts(dds0)) > 1, ]
dds
```

```
## class: DESeqDataSet
## dim: 37286 6
## metadata(1): version
## assays(1): counts
## rownames(37286): NM_000014 NM_000015 ... NR_046426 NR_046427
## rowData names(0):
## colnames(6): H1 H2 ... C2 C3
## colData names(1): condition
```

Task 04:

- How many genes have been removed?

Data quality assessment

(by sample clustering and visualization)

Our purpose is the detection of differentially expressed genes between the two species.

Count data transformation

In order to test for differential expression, we operate on raw counts and use discrete distributions. However for other downstream analyses e.g. for visualization or clustering it might be useful to work with transformed versions of the count data.

Here we use the regularized logarithm or rlog that incorporates a prior on the sample differences. It produces transformed data on the log2 scale which has been normalized with respect to library size. The point of these two transformation is to remove the dependence of the variance on the mean, particularly the high variance of the logarithm of count data when the mean is low.
Note that we do not require or desire that all the genes have exactly the same variance after transformation. After the transformations the genes with the same mean do not have exactly the same standard deviations, but the experiment-wide trend has flattened. It is those genes with row variance above the trend which will allow us to cluster samples into interesting groups.

```
rld <- rlog(dds, blind=FALSE)
```

Task 05:

- Examine the difference in the assays of `dds` and `rld`?

Heatmap of sample-to-sample distances

We use the transformed data for sample clustering. We apply the dist function to the transpose of the transformed count matrix to get sample-to-sample distances.

```
sampleDists <- dist(t(assay(rld)))
```

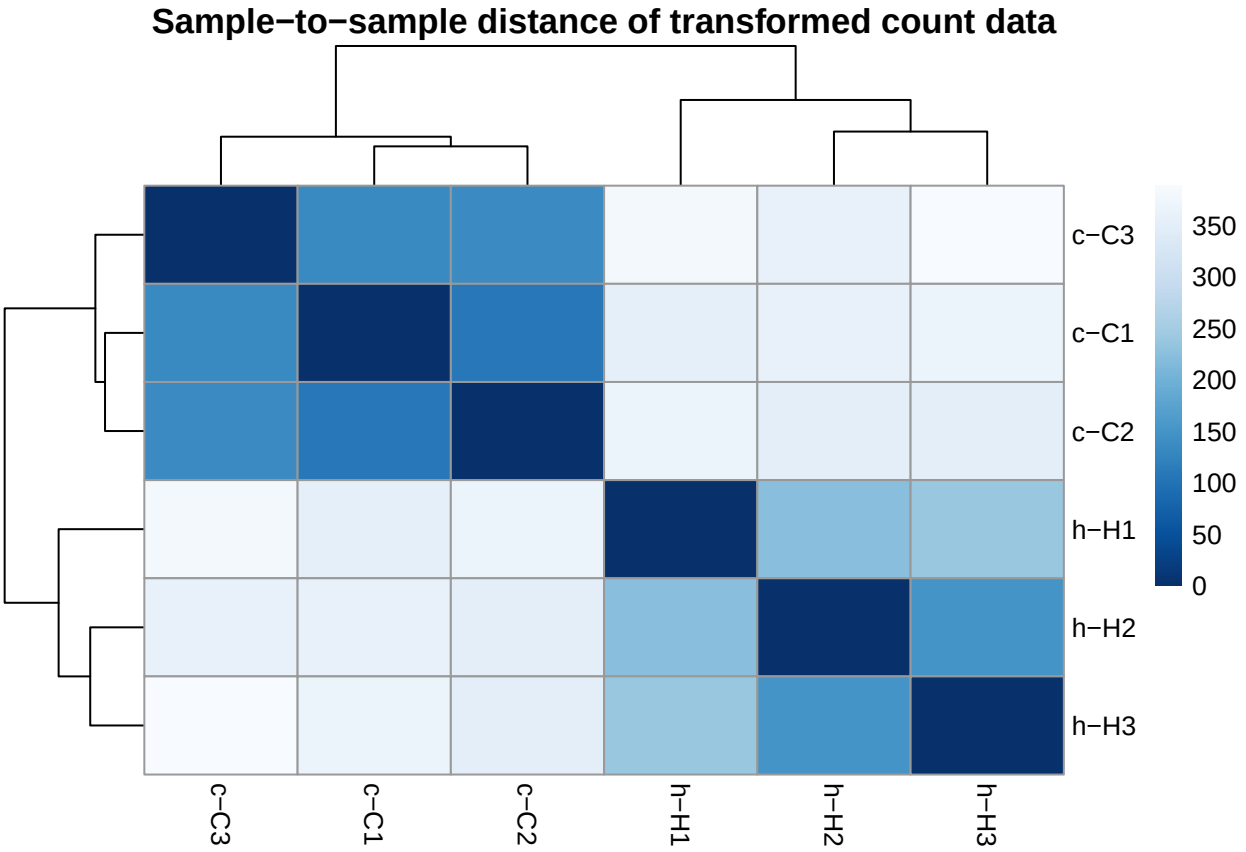
A heatmap of this distance matrix gives us an overview over similarities and dissimilarities between samples

```
sampleDistMatrix <- as.matrix(sampleDists)

rownames(sampleDistMatrix) <- paste(rld$condition, colnames(rld), sep="-")
colnames(sampleDistMatrix) <- paste(rld$condition, colnames(rld), sep="-")

colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)

pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors,
         legend=T,
         main="Sample-to-sample distance of transformed count data")
```



Task 06:

- What do you see?

Principle component analysis of samples

To further investigate the drivers of this clustering we can use a principal component analysis of the transformed data.

```
data <- plotPCA(rld, intgroup=c("condition"), returnData=TRUE)
data
```

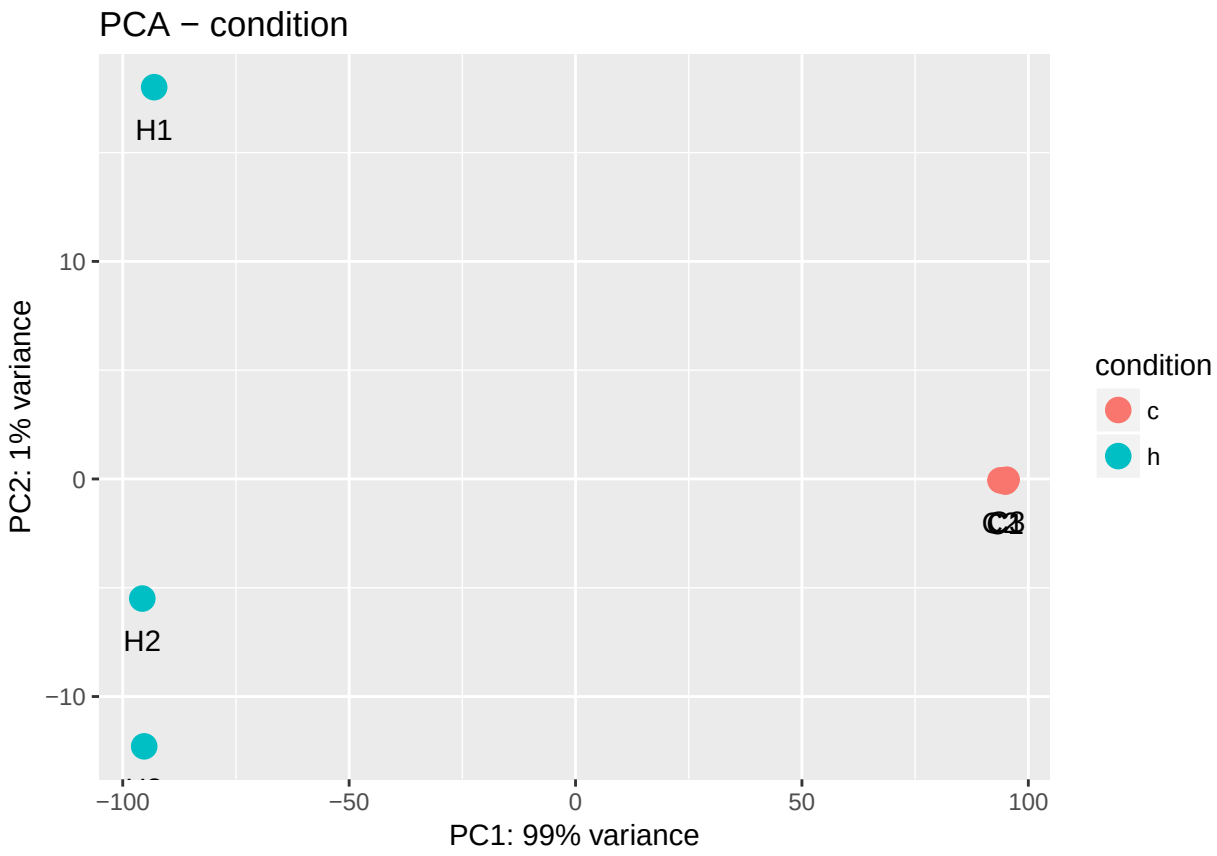
```
##          PC1          PC2 group condition name
## H1 -93.05333  18.00811223     h         h   H1
## H2 -95.67847  -5.49666158     h         h   H2
## H3 -95.26084 -12.29005510     h         h   H3
## C1  94.93182  -0.12344889     c         c   C1
## C2  93.85284  -0.06357473     c         c   C2
## C3  95.20798  -0.03437194     c         c   C3
```

```
percentVar <- round(100*attr(data, "percentVar"))
percentVar
```

```
## [1] 99  1
```

```
g <- ggplot(data, aes(x=PC1, y=PC2, color=condition))
g <- g + labs(color="condition")
```

```
g <- g + geom_point(size=4)
g <- g + geom_text(aes(label=colnames(dds)), vjust=2.5, size=4, colour="black")
g <- g + xlab(paste0("PC1: ", percentVar[1], "% variance"))
g <- g + ylab(paste0("PC2: ", percentVar[2], "% variance"))
g <- g + ggtitle("PCA - condition")
print(g)
```



Task 06:

- What do you see? Try to explain

Conclusion

The attribute condition is perfect and is good to use as design of our model.

Identify differentially expressed genes (DE genes)

Now we call the function `DESeq` to identify the differentially expressed genes.

```
dds1 <- DESeq(dds, parallel=T)
res <- results(dds1, parallel=T)
summary(res)
```

```
##
```

```
## out of 37286 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 4946, 13%
## LFC < 0 (down)   : 4720, 13%
## outliers [1]     : 317, 0.85%
## low counts [2]   : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
head(res)
```

```
## log2 fold change (MLE): condition h vs c
## Wald test p-value: condition h vs c
## DataFrame with 6 rows and 6 columns
##              baseMean log2FoldChange     lfcSE       stat       pvalue
##             <numeric>      <numeric> <numeric>  <numeric>    <numeric>
## NM_000014  2965.82419      0.6726647 0.5995806  1.1218920 2.619084e-01
## NM_000015    29.96077     -3.3316017 0.7501711 -4.4411223 8.949093e-06
## NM_000016  2848.44559      1.2828548 0.5051844  2.5393794 1.110493e-02
## NM_000017   509.89263     -0.4089217 0.5525244 -0.7400971 4.592411e-01
## NM_000018 11604.06134      1.2302219 0.5055531  2.4334176 1.495704e-02
## NM_000019  2343.86057      0.7454728 0.5489810  1.3579210 1.744888e-01
##                   padj
##              <numeric>
## NM_000014 5.068305e-01
## NM_000015 8.577626e-05
## NM_000016 5.013741e-02
## NM_000017 6.955786e-01
## NM_000018 6.392448e-02
## NM_000019 3.916117e-01
```

Statistical variables for filtering the results

```
p.adjusted <- 0.1
baseMean <- 10
```

and subsequent filtering of the results.

```
use <- res$baseMean >= baseMean & !is.na(res$pvalue)
res.f <- res[use,]
rownames(res.f) <- gsub("\\.\\d+", "", rownames(res.f))
res.f$padj <- p.adjust(res.f$pvalue, method="BH")
summary(res.f)
```

```
##
## out of 34792 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 4817, 14%
## LFC < 0 (down)   : 4675, 13%
## outliers [1]     : 0, 0%
## low counts [2]   : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

We can now do different things with this data:

- write the results to a csv file
- annotate each DE gene and check whether you see something that you had expected from this experiment
- do term enrichment analysis to check whether there are genes differentially expressed that belong to certain biological/functional terms
- draw heatmap(s):

```
DEgenes <- subset(res.f, padj <= p.adjusted)
dim(DEgenes)
```

```
## [1] 9492    6
```

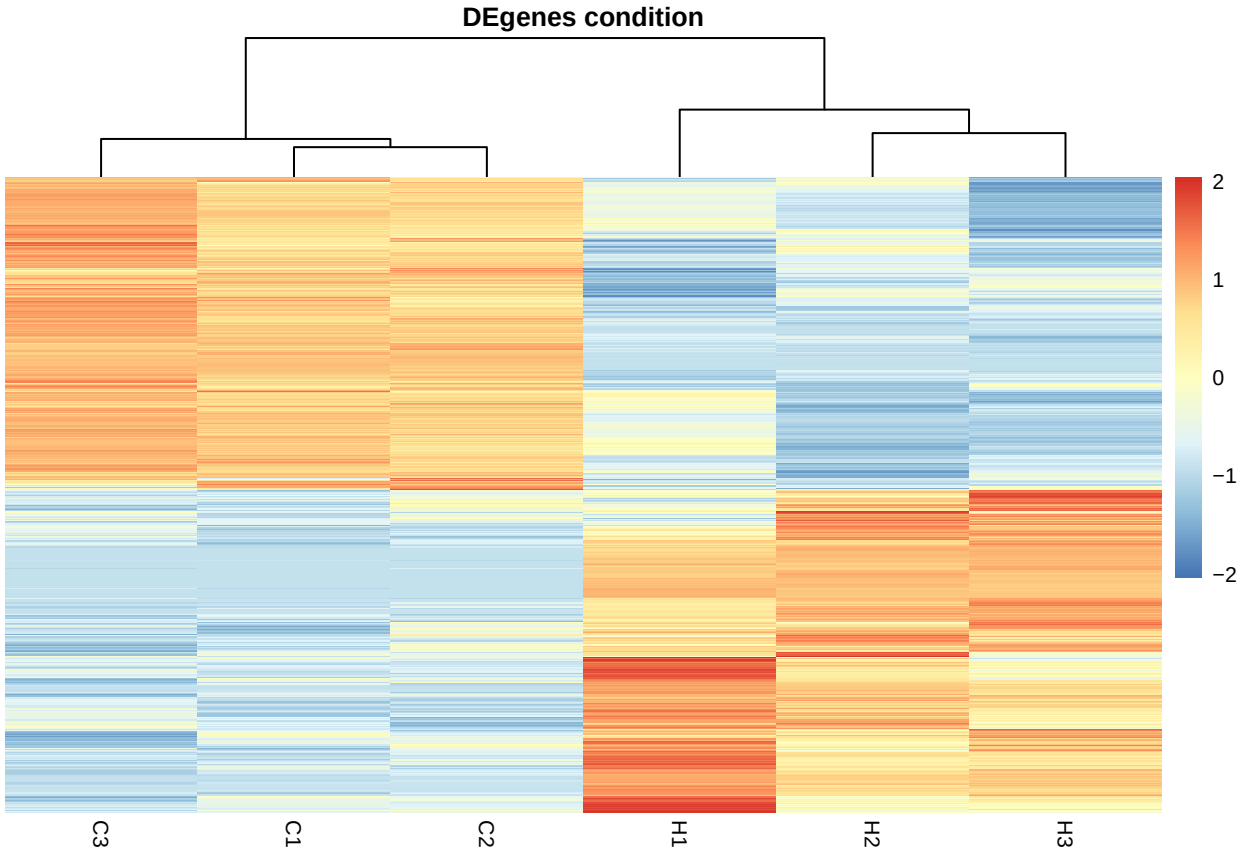
```
ID_DEgenes <- rownames(DEgenes)
rld_DEgenes = rld[rownames(rld) %in% ID_DEgenes, ]
dim(rld_DEgenes)
```

```
## [1] 9492    6
```

```
head(assay(rld_DEgenes))
```

```
##                  H1        H2        H3        C1        C2        C3
## NM_000015  3.627098  3.370841  3.060833  5.185764  5.160255  5.420443
## NM_000016 12.442267 11.503848 11.248912 10.795869 10.903324 10.734785
## NM_000018 12.995414 14.111920 14.163645 13.026665 12.628664 12.919934
## NM_000031 11.122759 12.030526 12.319637 10.147946 11.287888 10.686719
## NM_000033  7.103330  8.116499  7.905324  8.744742  8.731151  9.037026
## NM_000036  4.264207  4.584846  4.168484  6.877252  7.145766  7.654321
```

```
if(length(ID_DEgenes) >=2) {
  pheatmap(assay(rld_DEgenes),
           cluster_rows=TRUE,
           show_rownames=FALSE, # too many! more than 9000
           treeheight_row = 0,
           cluster_cols=TRUE,
           scale='row',
           main="DEgenes condition",
           fontsize = 8
  )
}
```




```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] ggplot2_2.2.1              RColorBrewer_1.1-2
##  [3] pheatmap_1.0.8             DESeq2_1.16.1
##  [5] SummarizedExperiment_1.4.0 Biobase_2.34.0
##  [7] GenomicRanges_1.26.4       GenomeInfoDb_1.10.3
##  [9] IRanges_2.8.2              S4Vectors_0.12.2
## [11] BiocGenerics_0.22.0
##
## loaded via a namespace (and not attached):
```

```
##  [1] genefilter_1.58.1    locfit_1.5-9.1       splines_3.3.2
##  [4] lattice_0.20-35      colorspace_1.3-2     htmltools_0.3.6
##  [7] yaml_2.1.14          blob_1.1.0           survival_2.41-3
## [10] XML_3.98-1.9         rlang_0.1.4          foreign_0.8-69
## [13] DBI_0.7              BiocParallel_1.8.2   bit64_0.9-7
## [16] plyr_1.8.4           stringr_1.2.0        zlibbioc_1.20.0
## [19] munsell_0.4.3        gtable_0.2.0         htmlwidgets_0.9
## [22] memoise_1.1.0        evaluate_0.10.1      labeling_0.3
## [25] latticeExtra_0.6-28  knitr_1.17           geneplotter_1.52.0
## [28] AnnotationDbi_1.38.0 htmlTable_1.9        Rcpp_0.12.13
## [31] acepack_1.4.1        xtable_1.8-2         scales_0.5.0
## [34] backports_1.1.1      checkmate_1.8.5      Hmisc_4.0-0
## [37] annotate_1.52.1      XVector_0.14.1       bit_1.1-12
## [40] gridExtra_2.3        digest_0.6.12        stringi_1.1.6
## [43] grid_3.3.2           rprojroot_1.2        tools_3.3.2
## [46] bitops_1.0-6         magrittr_1.5         lazyeval_0.2.1
## [49] RCurl_1.95-4.8       tibble_1.3.4         RSQLite_2.0
## [52] Formula_1.2-2        cluster_2.0.5        Matrix_1.2-12
## [55] data.table_1.10.4-3  rmarkdown_1.8        rpart_4.1-11
## [58] nnet_7.3-12
```