Differential Expression Analysis using LIMMA

Naomi S. Altman
Department of Statistics
Penn State University
naomi@stat.psu.edu

November, 2013

1 Introduction

In this lab we will do differential expression analysis of a complex experiment using Affymetrix@ Genechip arrays.

2 The Data

Khaitovich et al (2004) considered gene expression in 7 homologous regions of human and chimpanzee brains. There were 3 human brains and 3 chimpanzee brains available for the study. Each brain was dissected to obtain tissue samples for each of the 7 regions. This is called a split plot design. Each brain is a "whole plot" yielding a tissue sample for each region. The brains are classified into species which is the whole plot factor. The dissected portions of the brain are called "subplots" and region is the subplot factor. The interaction between species and region is also considered a subplot effect.

The factors "species" and "region" are arranged in a balanced factorial design, because each combination of species and region was sampled with the same number of biological replicates. However, there is also a blocking factor "brain" with 6 levels representing the 6 individuals.

The samples were hybridized to a variety of Affymetrix@ Genechips and are available as experiment E-AFMX-2 at http://www.ebi.ac.uk/aerep/dataselection?expid=352682122. We will use only 4 of the brain regions: prefrontal cortex, caudate nucleus, cerebellum and Broca's region and only one of the Genechips, HG_U95B with one hybridization per sample.

The data have been compiled into a Bioconductor dataset called `ChimpHumanBrainData`.

To start, we need to load the data into R.

```
> library('ChimpHumanBrainData')
> library(affy)
> celfileDir = system.file('extdata',package='ChimpHumanBrainData')
> celfileNames = list.celfiles(celfileDir)
> brainBatch=ReadAffy(filenames=celfileNames,celfile.path=celfileDir,compress=TRUE)
>
```

The sample names for brainBatch are the cel file names, which are not informative. We will replace them with more informative names, and then extract the probewise raw expression values for quality assessment. The `paste` and `rep` command are very handy for creating names. The array names are coded "a_xny" where n is the replicate number, x is either "c" for chimpanzee or "h" for human, and the brain regions are a) prefrontal cortex, d) caudate nucleus e) cerebellum or f) Broca's region. First print the sampleNames to be sure the arrays are in the right order. Then replace the names with the more informative names.

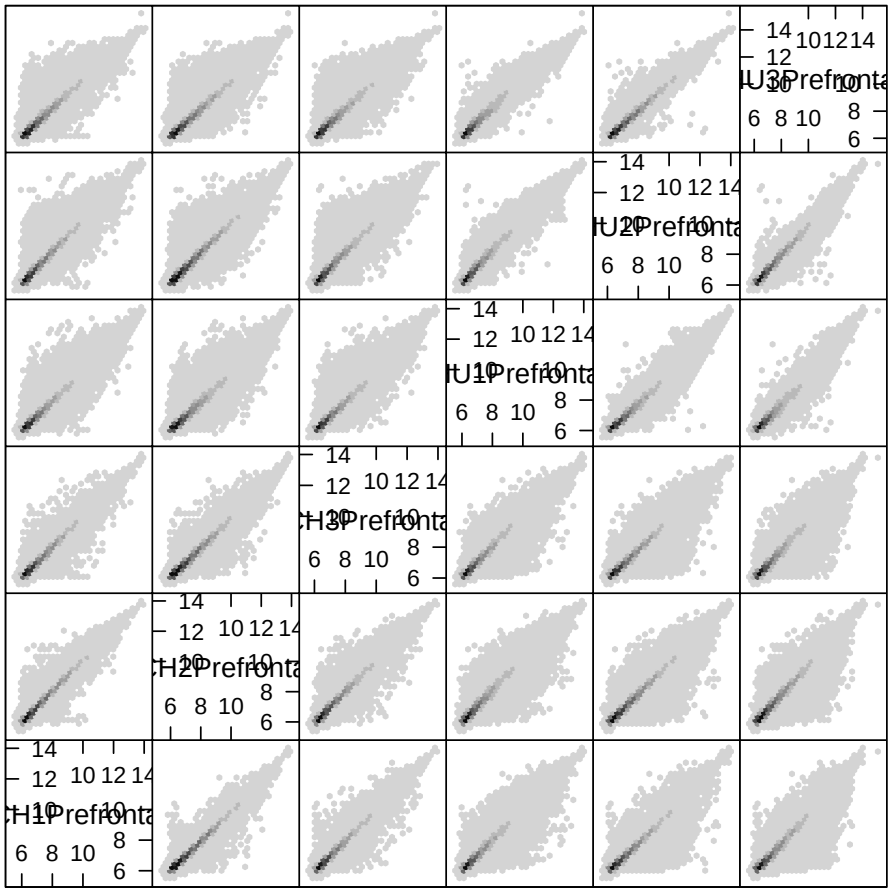
```
> sampleNames(brainBatch)
```

```
 [1] "a_c1a.cel.gz" "a_c1d.cel.gz" "a_c1e.cel.gz" "a_c1f.cel.gz" "a_c2a.cel.gz" "a_c2d.cel.gz"
 [7] "a_c2e.cel.gz" "a_c2f.cel.gz" "a_c3a.cel.gz" "a_c3d.cel.gz" "a_c3e.cel.gz" "a_c3f.cel.gz"
[13] "a_h1a.cel.gz" "a_h1d.cel.gz" "a_h1e.cel.gz" "a_h1f.cel.gz" "a_h2a.cel.gz" "a_h2d.cel.gz"
[19] "a_h2e.cel.gz" "a_h2f.cel.gz" "a_h3a.cel.gz" "a_h3d.cel.gz" "a_h3e.cel.gz" "a_h3f.cel.gz"
```

```
> sampleNames(brainBatch)=paste(rep(c("CH","HU"),each=12),rep(c(1:3,1:3),each=4),
+   rep(c("Prefrontal","Caudate","Cerebellum","Broca"),6),sep="")
```

We should at minimum check quality by doing some scatterplot matrices of the log2(expression) values. We could do a hexplom plot of each tissue.

```
> brain.expr=exprs(brainBatch)
> library(hexbin)
> plot(hexplom(log2(brain.expr[,paste(rep(c("CH","HU"),each=3),
+   c(1:3,1:3),rep("Prefrontal",6),sep="")])))
```



Notice that the most dense data are along the diagonal, and that the arrays of the same species are more correlated than the arrays from different species.

Exercise 1
Draw hexplom plots for the other 3 brain regions. Do any of the arrays appear to be different than the others?

We should set up the treatment names and blocks. This is readily done using `paste` and `rep`. The treatment names are the same as the sample names, but the replicate numbers are dropped. There is one block label for each brain.

```
> trts=factor(paste(rep(c("CH","HU"),each=12),
+   rep(c("Prefrontal","Caudate","Cerebellum","Broca"),6),sep=""))
> blocks=factor(rep(1:6,each=4))
```

Finally, we should normalize the expression values and combine into probeset summaries using a method such as RMA.

```
> brain.rma=rma(brainBatch)
```

```
Background correcting
Normalizing
Calculating Expression
```

We might also want to do some quality checks after normalization.

3 LIMMA analysis

We are now ready to perform analysis in LIMMA. The steps are:

1. Compute S_p^2 the pooled variance. To do this, we need to compute within region variance for each gene, and the correlation among regions from the same brain (averaged across all the genes).
2. Create the coefficient matrix for the contrasts.
3. Compute the estimated contrasts.
4. Compute the moderated contrast t-test for each gene.
5. Plot the histogram of p-values for each contrast for each gene.
6. Create the list of significant genes based on the p-values, adjusted p-values or FDR estimates.

3.1 Compute S_p^2

There are 3 steps to computing the pooled variance.

1. Create a design matrix for the treatment effects.
2. If there are blocks, compute the within block correlation for each gene.
3. Fit the model for the treatment effects to obtain the pooled variance.

A design matrix is a matrix whose columns give the coefficients of the linear model. There is one row for each sample. The simplest way to set up the matrix is with an incidence matrix that has value 1 if the column belongs to the treatment and 0 otherwise. Since we have already created the factor `trts` which connects the columns of the expression matrix to the treatment names, the design matrix is readily created.

```
> library(limma)
> design.trt=model.matrix(~0+trts)
```

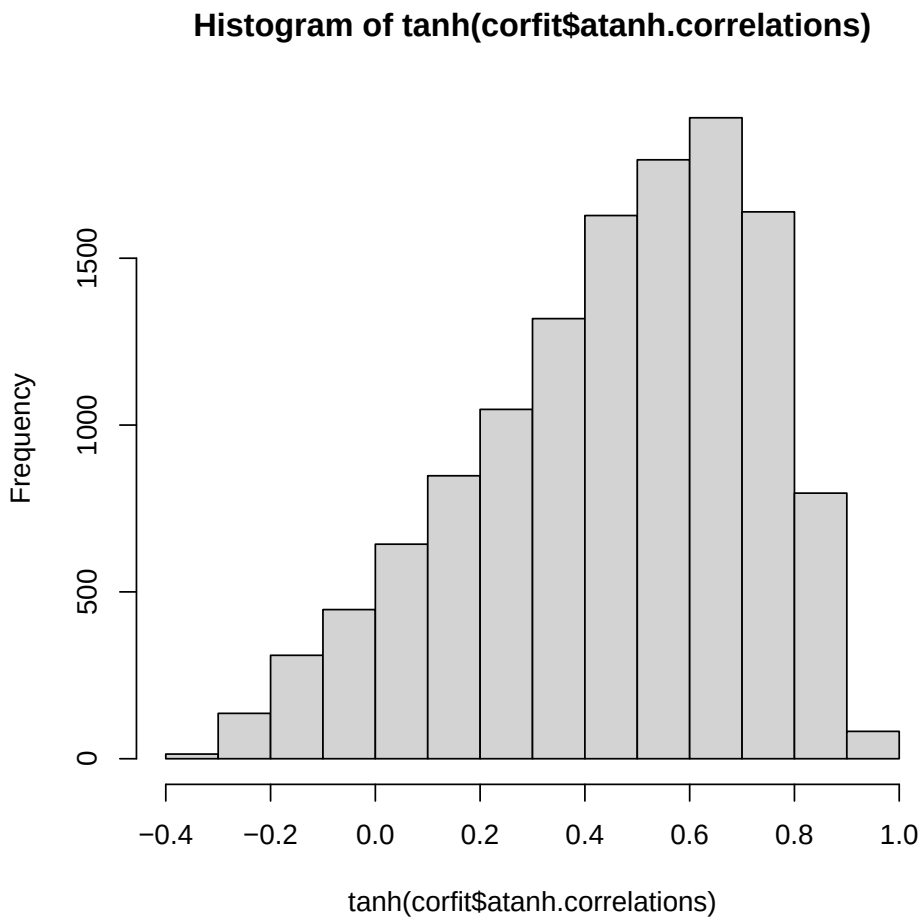
By default, `model.matrix` includes a column of all 1's representing μ in the ANOVA model $Y_{ij} = \mu + \alpha_i + \text{error}$. In this case, the fitted values will be estimates of μ and the α's. We prefer to eliminate this column ("0+") because then the fitted values will be the treatment mean expression value for each gene in each treatment, which are quantities we usually want to compute. You might want to print `design.trt` to see what it looks like.

If there are blocks or technical replicates the correlation of genes within the blocks need to be computed. This requires the design matrix and the blocking factor. In our case, the blocking factor is called `blocks`. The function `duplicateCorrelation` requires the package `statmod` which may not be automatically downloaded to your R directory with the Bioconductor routines. If R cannot find it, you will need to install it from a CRAN site. Note also that `duplicateCorrelation` is a time-consuming computation. Be patient.

```
> library(statmod)
> corfit <- duplicateCorrelation(brain.rma, design.trt, block = blocks)
```

The within-block correlation for each gene is stored on as hyperbolic arctan(correlation). So to obtain a histogram of the correlations, you need to use the `tanh` function:

```
> hist(tanh(corfit$atanh.correlations))
```



Notice that the correlations are mainly positive and have a mode around 0.6. A consensus correlation is computed by discarding the most extreme outliers, averaging the remainder on the hyperbolic arctan scale, and then transforming back to a correlation. This is stored in component `consensus.correlation`. `LIMMA` assumes that the correlation induced by the blocks is the same for all genes and uses the consensus.

We are now ready to compute the pooled sample variance for each gene. As a side effect, we also compute the sample mean expression of each gene in each treatment (remembering that after RMA normalization, the data are on the log2 scale).

```
> fitTrtMean <- lmFit(brain.rma, design.trt, block = blocks, cor = corfit$consensus.correlation)
```

The output `fitTrtMean` has several components, but only 2 of these are of interest. Component `coefficients` contains the mean expression for each gene in each treatment. Component `sigma` has the estimate of S_p. (Notice this the pooled SD, not the pooled variance.)

3.2 Create the coefficient matrix for the contrasts

We need to compute the coefficient matrix for any contrasts we want to do. We do not need to worry about the rank of this matrix, as we will obtain the pooled variances from `fitTrtMean`.

We need to decide what contrasts are interesting to us. For this lab, we will look at 6 contrasts:

1. Average chimpanzee versus average human
2. Chimpanzee versus human for each region
3. The interaction between species and the comparison of cerebellum to Broca's region.

Note that the treatment names are taken from the columns of the design matrix. To make more useful names for the final output, we will want to rename the columns of the contrast matrix.

```
> colnames(design.trt)

[1] "trtsCHBroca"      "trtsCHCaudate"    "trtsCHCerebellum" "trtsCHPrefrontal"
[5] "trtsHUBroca"      "trtsHUCaudate"    "trtsHUCerebellum" "trtsHUPrefrontal"

> contrast.matrix=makeContrasts(
+   (trtsCHBroca+trtsCHCaudate+trtsCHCerebellum+trtsCHPrefrontal)/4
+      -(trtsHUBroca+trtsHUCaudate+trtsHUCerebellum+trtsHUPrefrontal)/4,
+   trtsCHBroca-trtsHUBroca,
+   trtsCHCaudate-trtsHUCaudate,
+   trtsCHCerebellum-trtsHUCerebellum,
+   trtsCHPrefrontal-trtsHUPrefrontal,
+   (trtsCHCerebellum-trtsHUCerebellum)-(trtsCHBroca-trtsHUBroca),
+  levels=design.trt)
> colnames(contrast.matrix)=
+   c("ChVsHu","Broca","Caudate","Cerebellum","Prefrontal","Interact")
```

The resulting contrast coefficient matrix has one row for each treatment and one column for each contrast.

3.3 Compute the estimated contrasts.

We simply fit the contrast matrix to the previous fitted model:

```
> fit.contrast=contrasts.fit(fitTrtMean,contrast.matrix)
```

3.4 Compute the moderated contrast t-test.

The `eBayes` command will compute the consensus pooled variance, and then use it to compute the empirical Bayes (moderated) pooled variance for each gene. This also adjusts the degrees of freedom for the contrast t-tests. The command also computes the t-tests and associated p-values.

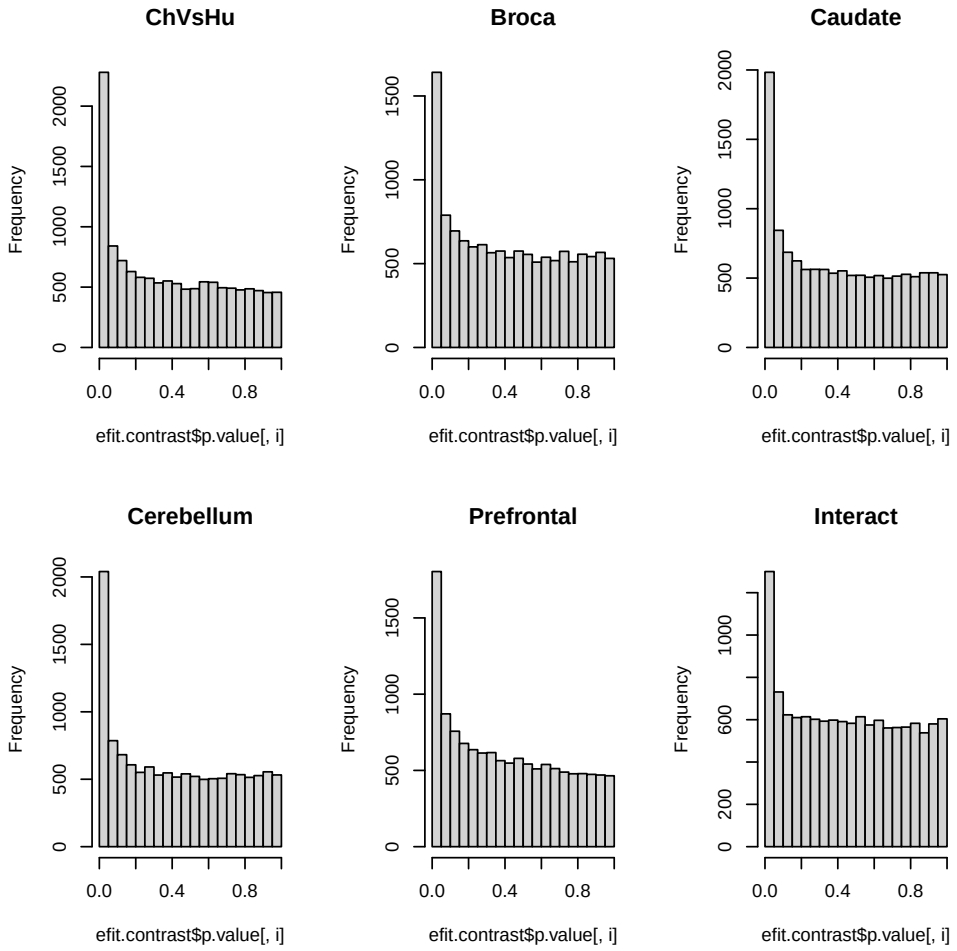
```
> efit.contrast=eBayes(fit.contrast)
```

The interesting components of this output are the estimated contrasts, which are stored in the component `coefficient` and the contrast p-values, which are stored in component `p.value`.

3.5 Plot the p-values.

It is important to remember that when the null hypothesis is true for every comparison, the p-values should be uniformly distributed and we expect to have false detections. In the more usual situation that some of the genes differentially express, we will have both false detections and false nondetections. As simple way of visualizing what to expect is to plot a histogram of p-values for each contrast. Below we put all 6 histograms on a single plot and use the contrast names as labels.

```
> par(mfrow=c(2,3))
> for (i in 1:ncol(efit.contrast$p.value)) {
+ hist(efit.contrast$p.value[,i],main=colnames(efit.contrast$p.value)[i])
+ }
```



Notice that the overall species contrast has the most differentially expressing genes (small p-values). All of the comparisons have a large percentage of differentially expressing genes. We will want to use a multiple comparisons procedure that adapts to having a large number of non-null hypotheses such as the Benjamini and Yuketiel or Storey methods. All of the plots show sharp peaks of small p-values. This indicates that we have good detection power in this study. When the power is poor, the histogram drops slowly towards the flat area.

3.6 Compute the gene list

The most statistically significant genes for each contrast can be assembled into spreadsheets. There are several ways to do this. `LIMMA` provides 2 functions, `topTable` and `decideTests` to assemble gene lists. I prefer to compute FDR or q-value estimates or adjusted p-values for each gene and output the treatment means and estimated contrasts, p-values and FDR or q-values to a comma separated text file which I can import to a spreadsheet.

To use `topTable`, select a contrast and one of the adjustment methods. Of those available, Benjamini and Yuketiel (2001) ("BY") is a good general purpose choice. You also need probeset ids, which can either be extracted from the original data or from the row names of the p-value matrix.

To limit the output to the most statistically significant genes, set the input parameter `p.value` to the maximum adjusted p-value or estimated FDR that you want to consider and the input parameter `n` to the maximum number of genes you want on the list. If you want a complete list, set `p.value=1.0` and `n=X` where X is bigger than the total number of probesets on the array.

For example to get the top 10 genes with $p<10^{-5}$ for the overall comparison and for the interaction contrast:

```
> genes=geneNames(brainBatch)
> topTable(efit.contrast,coef=1,adjust.method="BY",n=10,p.value=1e-5,genelist=genes)

                   ID     logFC   AveExpr         t      P.Value    adj.P.Val        B
41155_at     41155_at -3.476775  7.240403 -28.90310 2.318240e-18 2.932833e-13 29.94653
33364_at     33364_at -1.645199  5.082405 -22.69074 3.165648e-16 2.002449e-11 26.14110
39758_f_at 39758_f_at -2.158371  8.912186 -20.16317 3.385050e-15 1.427489e-10 24.15538
40421_at     40421_at -2.844953  7.269491 -18.31548 2.284563e-14 7.225570e-10 22.49517
40725_at     40725_at -2.251058  6.830048 -17.96030 3.362940e-14 8.508991e-10 22.15308
326_i_at     326_i_at -1.954376  7.773746 -17.56547 5.210262e-14 1.098594e-09 21.76346
1323_at       1323_at -1.680757 10.521953 -16.39710 2.005788e-13 3.172924e-09 20.54961
34440_at     34440_at  1.832690  7.486743  16.39684 2.006414e-13 3.172924e-09 20.54933
34526_s_at 34526_s_at -2.025578  7.511111 -16.16352 2.652420e-13 3.728454e-09 20.29543
39370_at     39370_at -1.868039  8.229215 -15.98748 3.281734e-13 4.151762e-09 20.10120

> topTable(efit.contrast,coef=6,adjust.method="BY",n=10,p.value=1e-5,genelist=genes)

               ID     logFC  AveExpr         t      P.Value    adj.P.Val        B
36271_at 36271_at -1.640468 6.589687 -13.53282 8.012140e-12 1.013626e-06 16.07697
35669_at 35669_at -2.203430 7.830890 -11.61954 1.360944e-10 8.608735e-06 13.74181
```

The columns of the table are the row number of the gene, the gene id, the estimated contrast, the expression mean over all arrays, contrast t-value, contrast p-value, contrast adjusted p-value or estimated FDR and the estimated log-odds probability ratio that the gene is differentially expressed.

The `decideTests` function can be used to create indicator variables for significance of contrasts with a variety of options.

As an alternative, `write.table` can be used to create a comma separated text file, using `cbind` to concatenate matrices.

```
> write.table(file="fits.txt",
+   cbind(genes,fitTrtMean$coefficients,efit.contrast$coefficients,efit.contrast$p.value),
+   row.names=F,
+   col.names=c("GeneID",colnames(fitTrtMean$coefficients),colnames(efit.contrast$p.value),
+   paste("p",colnames(efit.contrast$coefficients))),sep=",")
```

Exercise 2
Append adjusted p-values to the table above using either `p.adjust` *or* `qvalue`*. Note that to use* `qvalue` *with* `apply` *you need to write a wrapper function. For example*

```
> library(qvalue)
> q.values=apply(efit.contrast$p.value,2, function(x) qvalue(x)$qvalues)
```

References

[1] Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*,**29**: 1165?-1188.

[2] Khaitovich, P., Muetzel, B., She, X., Lachmann, M., Hellmann, I., Dietzsch, J., Steigele, S., Do, H. H., Weiss, G., Enard, W., Heissig, F., Arendt, T., Nieselt-Struwe, K., Eichler, E. E., Pääbo, S. (2004) Regional patterns of gene expression in human and chimpanzee brains. *Genome research*, **14** (8) :1462–73.

[3] Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, **3**, Article 3. http://www.bepress.com/sagmb/vol3/iss1/art3.

[4] Storey JD. (2003) The positive false discovery rate: A Bayesian interpretation and the q-value. *Annals of Statistics*, **31**: 2013–2035.

SessionInfo

- R version 4.1.1 (2021-08-10), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.54.0, BiocGenerics 0.40.0, ChimpHumanBrainData 1.32.0, affy 1.72.0, hexbin 1.28.2, hgu95av2cdf 2.18.0, limma 3.50.0, qvalue 2.26.0, statmod 1.4.36
- Loaded via a namespace (and not attached): AnnotationDbi 1.56.1, BiocManager 1.30.16, Biostrings 2.62.0, DBI 1.1.1, GenomeInfoDb 1.30.0, GenomeInfoDbData 1.2.7, IRanges 2.28.0, KEGGREST 1.34.0, R6 2.5.1, RCurl 1.98-1.5, RSQLite 2.2.8, Rcpp 1.0.7, S4Vectors 0.32.0, XVector 0.34.0, affyio 1.64.0, assertthat 0.2.1, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.2, cachem 1.0.6, colorspace 2.0-2, compiler 4.1.1, crayon 1.4.1, dplyr 1.0.7, ellipsis 0.3.2, fansi 0.5.0, fastmap 1.1.0, generics 0.1.1, ggplot2 3.3.5, glue 1.4.2, grid 4.1.1, gtable 0.3.0, httr 1.4.2, lattice 0.20-45, lifecycle 1.0.1, magrittr 2.0.1, memoise 2.0.0, munsell 0.5.0, pillar 1.6.4, pkgconfig 2.0.3, plyr 1.8.6, png 0.1-7, preprocessCore 1.56.0, purrr 0.3.4, reshape2 1.4.4, rlang 0.4.12, rstudioapi 0.13, scales 1.1.1, splines 4.1.1, stats4 4.1.1, stringi 1.7.5, stringr 1.4.0, tibble 3.1.5, tidyselect 1.1.1, tools 4.1.1, utf8 1.2.2, vctrs 0.3.8, zlibbioc 1.40.0

```
> print(gc())
```

```
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells  4197054 224.2    8270621 441.7  8270621 441.7
Vcells 25183250 192.2   41232102 314.6 35550707 271.3
```