Statistical tests in latent variable models - homework

December 31, 2019

[104]:
```
library(Biobase)
library(GEOquery)
library(data.table)
library(tidyverse)
library(RColorBrewer)
# creating heatmaps
library(gplots)
# for the jackstraw tests
library(jackstraw)
# calcuating q-values and FDRs
library(qvalue)
# combining ggplots
library(sva)
library(broom)
library(patchwork)
library(limma)
library(edge)
library(gridExtra)
```

```
Attaching package: ‘gridExtra’

The following object is masked from ‘package:dplyr’:

    combine

The following object is masked from ‘package:Biobase’:

    combine

The following object is masked from ‘package:BiocGenerics’:

    combine
```

0.1 Epigenetic profiling in CD4+ and CD8+ T cells from Graves' disease patients reveals changes in genes associated with T cell receptor signaling.

Limbach et al. Journal of Autoimmunity (2016)

Graves' disease (GD) is an autoimmune disease affecting the thyroid. The disorder results from an antibody, called thyroid-stimulating immunoglobulin (TSI), that has a similar effect to thyroid stimulating hormone (TSH). These TSI antibodies cause the thyroid gland to produce excess thyroid hormones.

The ethymology of GD is complex; a combination of genetic, epigenetic and environmental factors contribute to the development of the disease. T cells play an important role in Graves' disease as they influence the production of thyroid-stimulating antibodies. Circulating T-cells, in particular CD4+ and CD8+ cells, infiltrate the thyroid and recognize epitopes on the thyroid stimulation hormone receptor (TSHR). The disruption of T cell signaling and activation could contribute to GD onset and progression.

It has been suggested that enviromental factors could be influencing the performance of T cells through epigenetic modulations, in particular DNA hypermethylation.

In order to identify the epigenetic changes involved in GD, the authors performed DNA methylation analysis, gene expression measurements and ChIP-seq assays in sorted CD4+ and CD8+ T cells.

0.1.1 Describe "your data" from the final project. What is the data about and how is the data obtained experimentally and computationally? At the minimum, follow the steps taken in this notebook to show various attributes, dimensions, and meta data.

Although the study relies heavily on the integration of three experiments, the gene expression dataset for this analysis is suficient (and computationally undemanding).

As GD is more prevalent in woman, the study was carried out among 69 female participant: 38 with past diagnosis of GD and 31 healthy individuals. The CD4+ and CD8+ cells were purified from blood and gene expression was measured with Illumina Expression BeadChips.

[3]:
```
#Loading a GDS file with GEOquery
dat <- getGEO('GSE71956', destdir=".", GSEMatrix= TRUE)
```

```
Found 1 file(s)
GSE71956_series_matrix.txt.gz
Using locally cached version: ./GSE71956_series_matrix.txt.gz
Parsed with column specification:
cols(
  .default = col_double(),
  ID_REF = col_character()
)
See spec(…) for full column specifications.
Using locally cached version of GPL10558 found here:
./GPL10558.soft
Warning message:
```

```
"3270 parsing failures.
  row        col           expected    actual         file
29537 Unigene_ID 1/0/T/F/TRUE/FALSE Hs.571610 literal data
29538 Unigene_ID 1/0/T/F/TRUE/FALSE Hs.545780 literal data
29539 Unigene_ID 1/0/T/F/TRUE/FALSE Hs.554603 literal data
29540 Unigene_ID 1/0/T/F/TRUE/FALSE Hs.437179 literal data
29541 Unigene_ID 1/0/T/F/TRUE/FALSE Hs.128234 literal data
… … … … …
See problems(…) for more details.
"
```

[4]:
```
eset <- dat[[1]]
edata <- exprs(eset)
pheno <- pData(eset)
```

There are only 49 columns corresponding to 49 samples, which means that the low quality samples have been removed during the preprocessing.

[5]:
```
class(eset)
mode(eset)

dim(edata)
dim(pheno)
```

'ExpressionSet'

'S4'

1. 47323 2. 49

1. 49 2. 40

[6]:
```
edata[1:5,1:10]
```

	GSM1848135	GSM1848136	GSM1848137	GSM1848138	GSM1848139	GSM1848140
ILMN_1343291	13.758933	13.517861	13.450794	13.572097	13.875380	13.724038
ILMN_1343295	9.009108	9.344112	9.483680	9.302660	8.994604	8.600564
ILMN_1651199	4.424273	4.212721	4.200543	4.362133	4.218618	4.267591
ILMN_1651209	4.517634	4.740555	4.273515	4.199127	5.074448	4.302497
ILMN_1651210	4.381600	4.217415	4.637432	4.297888	4.294360	4.340287

[7]:
```
colnames(pheno)
```

1. 'title' 2. 'geo_accession' 3. 'status' 4. 'submission_date' 5. 'last_update_date' 6. 'type' 7. 'channel_count' 8. 'source_name_ch1' 9. 'organism_ch1' 10. 'characteristics_ch1' 11. 'characteristics_ch1.1' 12. 'characteristics_ch1.2' 13. 'characteristics_ch1.3' 14. 'characteristics_ch1.4' 15. 'molecule_ch1' 16. 'extract_protocol_ch1' 17. 'label_ch1' 18. 'label_protocol_ch1' 19. 'taxid_ch1' 20. 'hyb_protocol' 21. 'scan_protocol' 22. 'data_processing' 23. 'platform_id' 24. 'contact_name' 25. 'contact_email' 26. 'contact_phone' 27. 'contact_laboratory' 28. 'contact_department' 29. 'contact_institute' 30. 'contact_address' 31. 'contact_city' 32. 'con-

tact_zip/postal_code' 33. 'contact_country' 34. 'supplementary_file' 35. 'data_row_count' 36. 'age:ch1' 37. 'cell type:ch1' 38. 'diagonsis:ch1' 39. 'gender:ch1' 40. 'tissue:ch1'

[8]:
```
head(pheno)
```

	title	geo_accession	status	submission_date	last_u
GSM1848135	RNA_CD4_Control_rep1	GSM1848135	Public on Nov 06 2015	Aug 11 2015	Nov 0
GSM1848136	RNA_CD4_Control_rep2	GSM1848136	Public on Nov 06 2015	Aug 11 2015	Nov 0
GSM1848137	RNA_CD4_Control_rep3	GSM1848137	Public on Nov 06 2015	Aug 11 2015	Nov 0
GSM1848138	RNA_CD4_Control_rep4	GSM1848138	Public on Nov 06 2015	Aug 11 2015	Nov 0
GSM1848139	RNA_CD4_Control_rep5	GSM1848139	Public on Nov 06 2015	Aug 11 2015	Nov 0
GSM1848140	RNA_CD4_Control_rep6	GSM1848140	Public on Nov 06 2015	Aug 11 2015	Nov 0

0.1.2 Cleaning up pheno data

Columns: *status, submission_date, last_update_date, type, channel_count, organism_ch1, characteristics_ch1, characteristics_ch1.2, molecule_ch1, extract_protocol_ch1, label_ch1, label_protocol_ch1, taxid_ch1, hyb_protocol, scan_protocol, data_processing, platform_id, contact_name, contact_email, contact_phone, contact_laboratory, contact_department, contact_institute, contact_address, contact_city, contact_zip/postal_code, contact_country, data_row_count, gender:ch1, tissue:ch1* contain a single value for all the samples, therefore they can be ommited.

The *supplementary_file* column contains information irrelevant to this analysis (although the files contain important information on hormones concetrations, which could be relevant to the analysis.

characteristics_ch1.3 and *age:ch1* contain exactly the same information, so the former one was dropped. The same applies to *characteristics_ch1.4* and *diagnosis:ch1*, *characteristics_ch1.1* and *cell type:ch1*. The column *source_name_ch1* repeats the information from other columns and can also be labeled also dispensable.

Changing the names of the columns for convinience reasons.

There are remarkably few informations on technical variables such as expression bead array chip or array row.

[9]:
```
# head(pheno, 5)
# pheno %>% distinct(`tissue:ch1`)

pheno <- pheno %>% select("title" = title, "age" = `age:ch1`,
                  "cell_type" = `cell type:ch1`, "diagnosis" = `diagonsis:ch1`)
head(pheno)
```

	title	age	cell_type	diagnosis
GSM1848135	RNA_CD4_Control_rep1	50	CD4 T cells	Healthy
GSM1848136	RNA_CD4_Control_rep2	51	CD4 T cells	Healthy
GSM1848137	RNA_CD4_Control_rep3	50	CD4 T cells	Healthy
GSM1848138	RNA_CD4_Control_rep4	54	CD4 T cells	Healthy
GSM1848139	RNA_CD4_Control_rep5	52	CD4 T cells	Healthy
GSM1848140	RNA_CD4_Control_rep6	59	CD4 T cells	Healthy

Removing rows with missing values is unnecessary.

[10]:
```
# remove the rows with missing values
length(edata)
rows.na <- apply(edata ,1,function(x) sum(is.na(x)))
length(edata[rows.na != 0,])
```

2318827

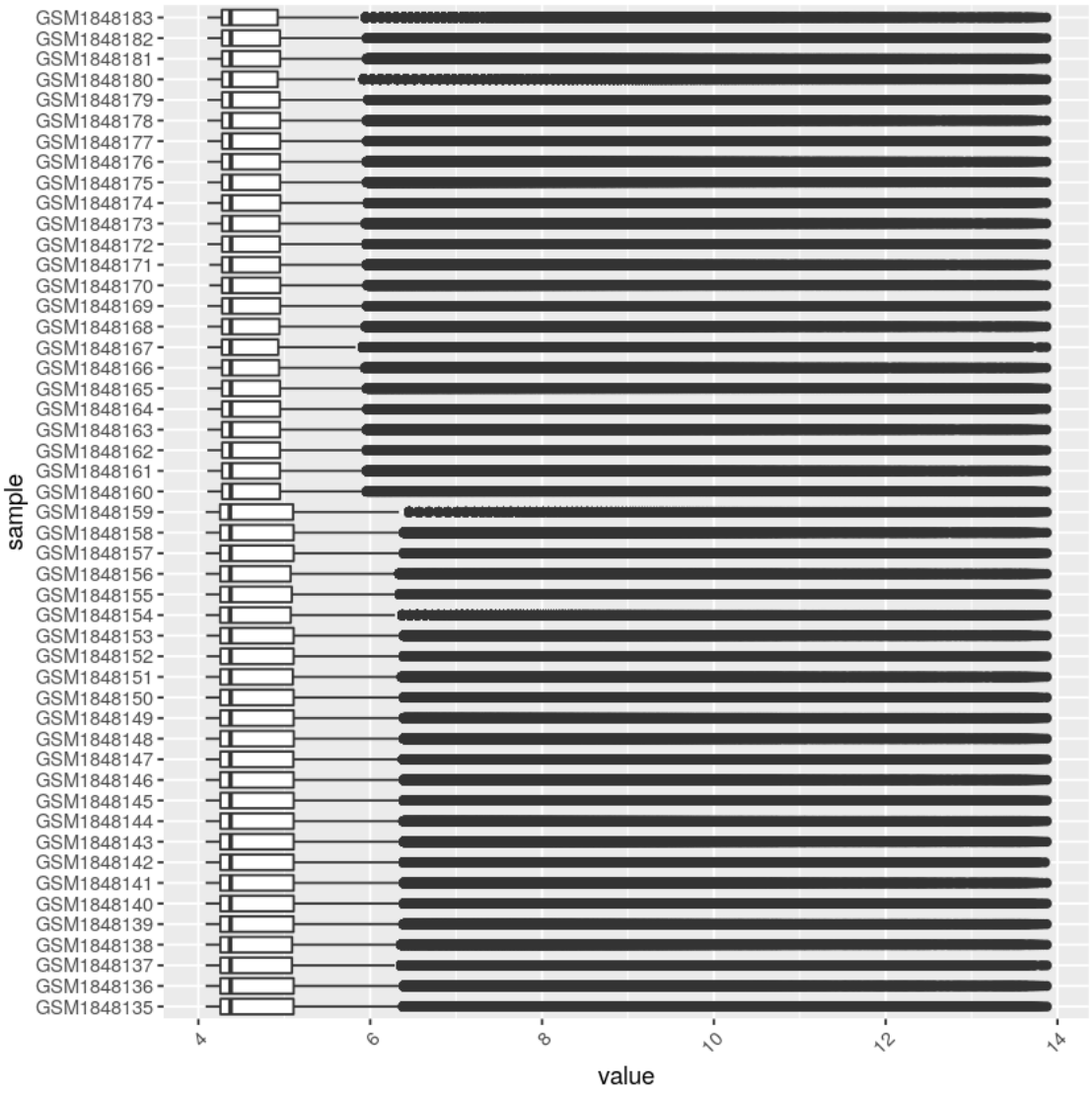
0

0.2 Data visualization

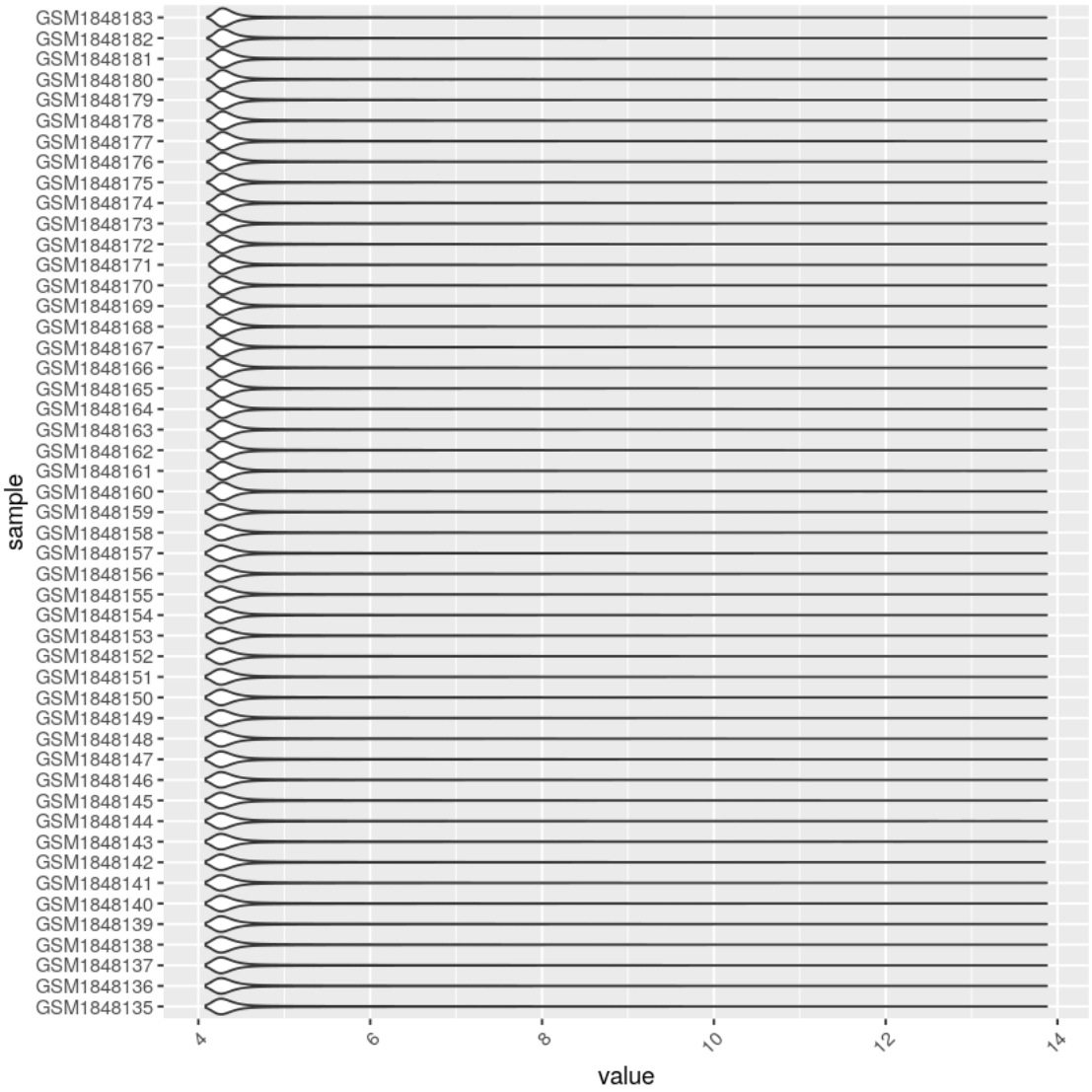
The plots shown below suggest that some kind of transormation was applied to the data, as shown on the boxplot and violin plot (all the samples are similar). However, the data are not scaled.

[11]:
```
edata.gathered <- as.data.frame(edata) %>% gather("sample", "value")
```

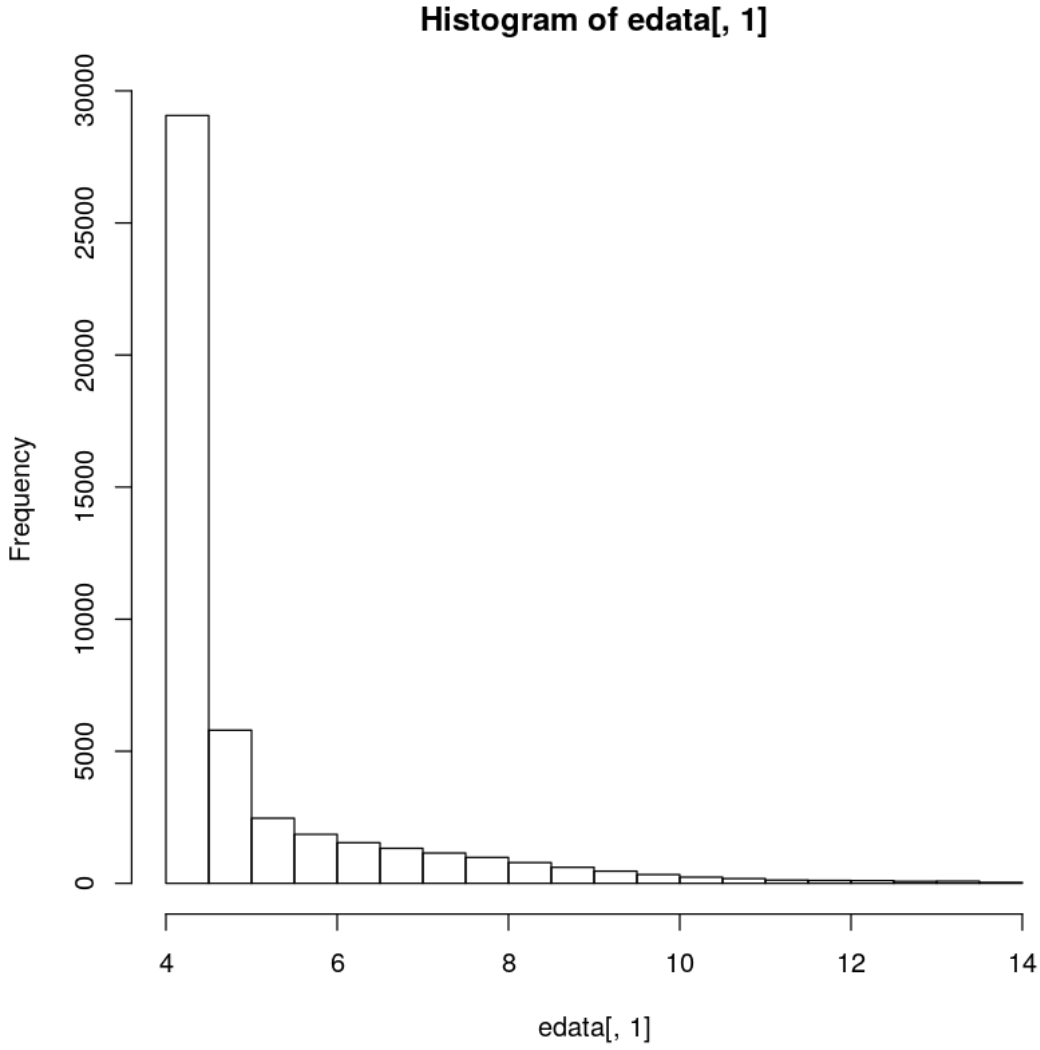
[12]:
```
ggplot(edata.gathered, aes(x=sample, y=value)) +
    geom_boxplot() +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    coord_flip()
```



[13]:
```
ggplot(edata.gathered, aes(x=sample, y=value)) +
    geom_violin() +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    coord_flip()
```



[177]:
```
hist(edata[,1])
```



0.2.1 Linear regression model with limma

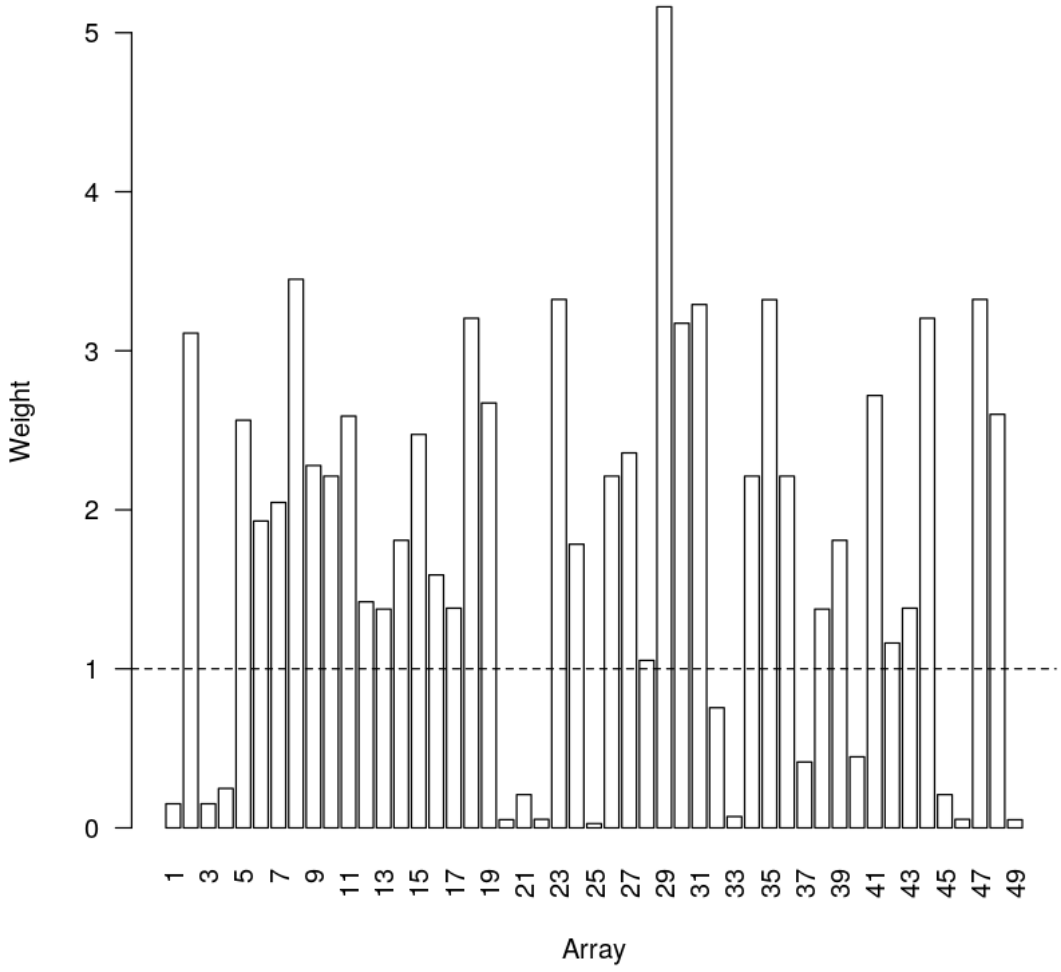
In the paper the expression dataset was processed with *limma* package.

> The remaining expression arrays were given weights as described in Ref. [27] and implemented in "limma". Age and the expression bead array chip were considered as covariates and the sample group was the variable of interest in a linear regression model that was used in the arrayWeights function.

The reason for using arrayWeights seems to be the use of human *in vivo* clinical patients data. In this case the RNA samples tend to be variable in quality.

[14]:
```
design = model.matrix(~pheno$age)
arrayw <- arrayWeights(edata, design)
barplot(arrayw, xlab="Array", ylab="Weight", col="white", las=2)
abline(h=1, lwd=1, lty=2)
fitw <- lmFit(edata, design, weights=arrayw)
fitw <- eBayes(fitw)
```

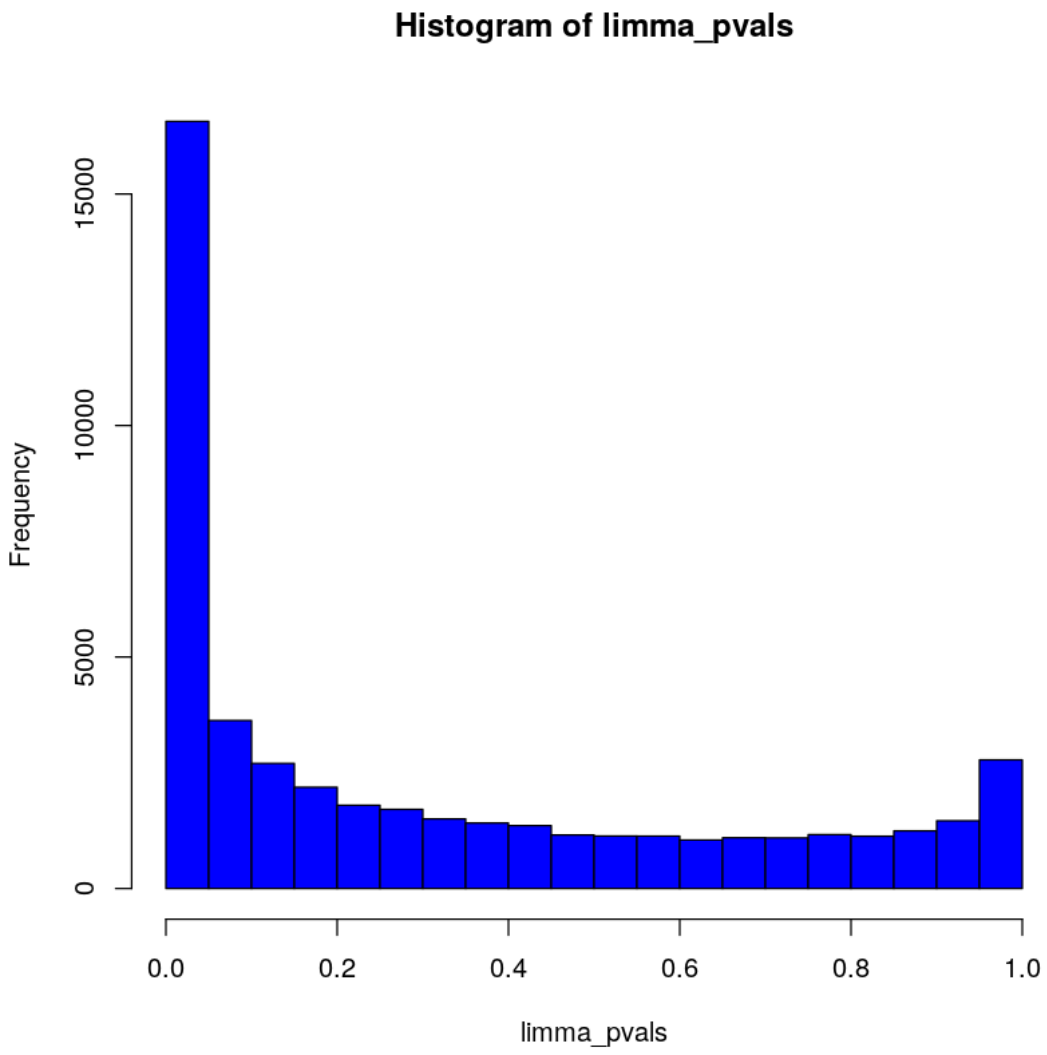
```
Warning message in .arrayWeightsREML(E, design = design, var.design = Z2,
prior.n = prior.n, :
"convergence tolerance not achievable, stopping prematurely"
```

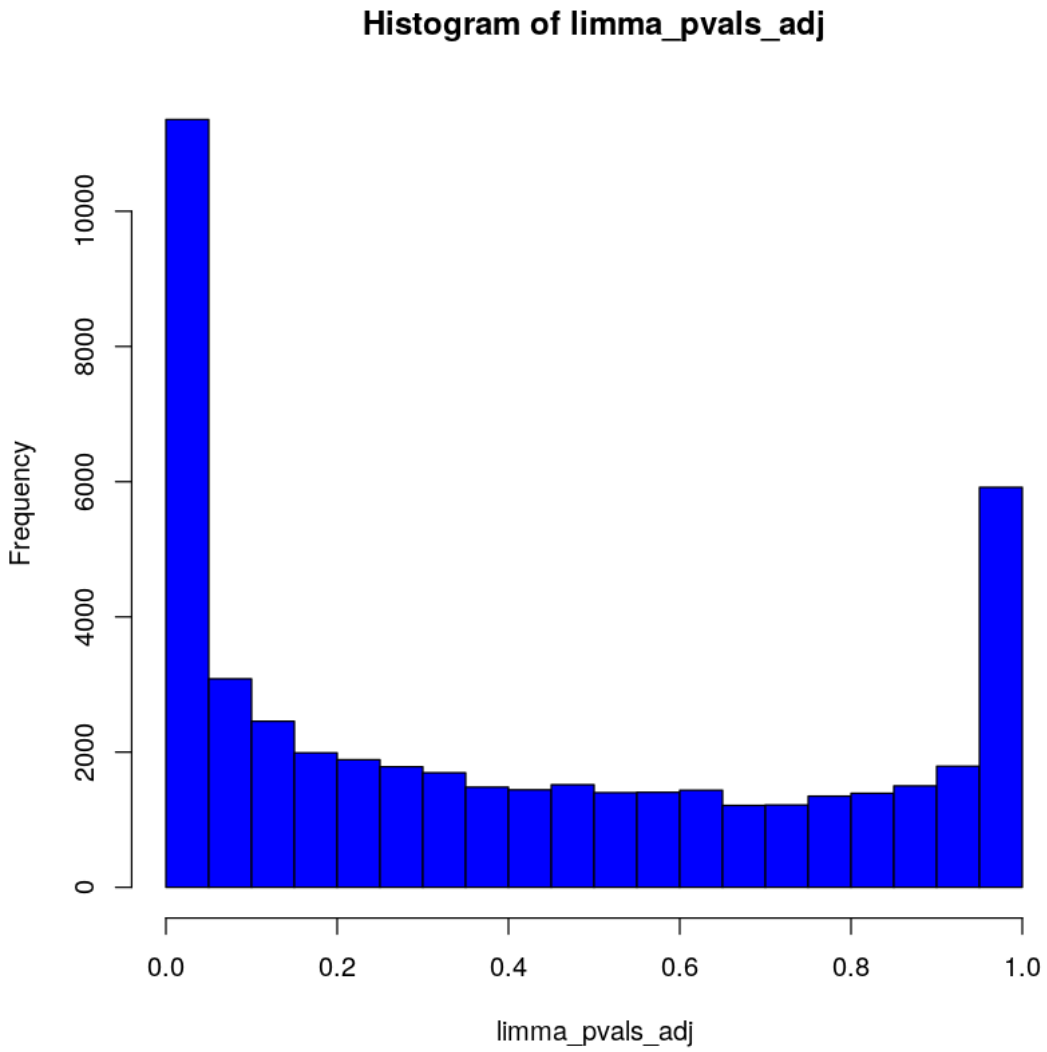


[15]:
```
limma_pvals = topTable(fitw,number=dim(edata)[1])$P.Value
limma_pvals_adj = topTable(fitw,number=dim(edata)[1])$adj.P.Val
```

```
hist(limma_pvals,col=4)
hist(limma_pvals_adj,col=4)
```

```
Removing intercept from test coefficients
Removing intercept from test coefficients
```





Because we do not know much about technical variables, it seems that applying SVA could reveal more.

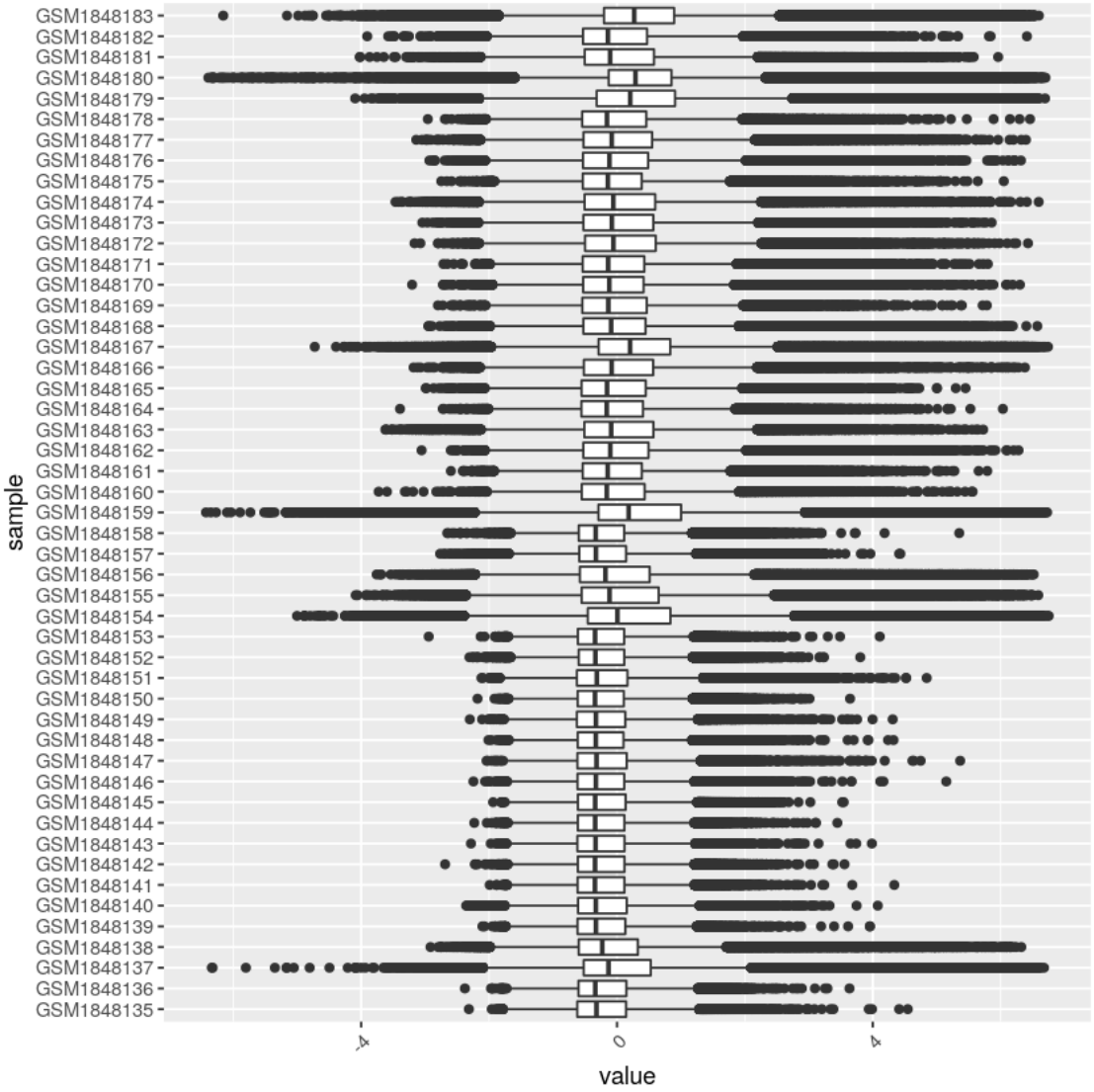
0.2.2 Normalize your data. Follow the prescribed normalization process in the original paper. Critically discuss (e.g., provide a comment) whether their normalization process can be improved with ComBat or SVA. Apply additional normalization steps. How do they affect your data (e.g., cleaned data from SVA)?

0.2.3 Normalization, scaling and centering

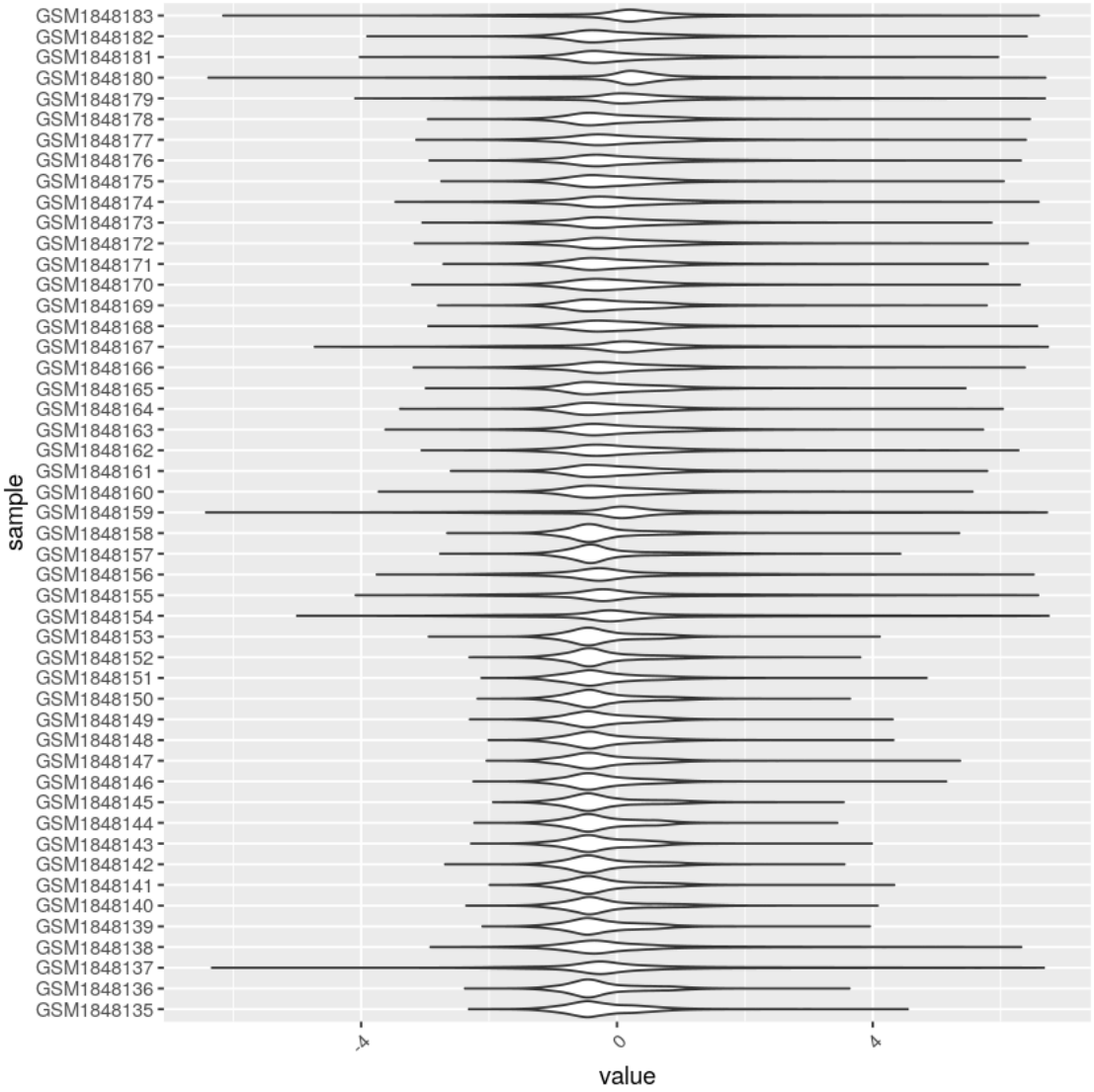
[16]:
```
# remove the first eigenmatrix from the raw data
raw.svd = svd(edata)
ec = edata - (raw.svd$d[1] * raw.svd$u[,1] %*% t(raw.svd$v[,1]))

# take the log transformation
elv = log(ec^2)
elv.svd = svd(elv)

# remove the first eigenmatrix from the log-transformed data
eclv = elv - (elv.svd$d[1] * elv.svd$u[,1] %*% t(elv.svd$v[,1]))
en = sign(ec) * sqrt(exp(eclv))

# center each row
edata.norm = t(scale(t(en), center=TRUE, scale=TRUE))
```

[17]:
```
edata.norm.gathered <- as.data.frame(edata.norm) %>% gather("sample", "value")
```

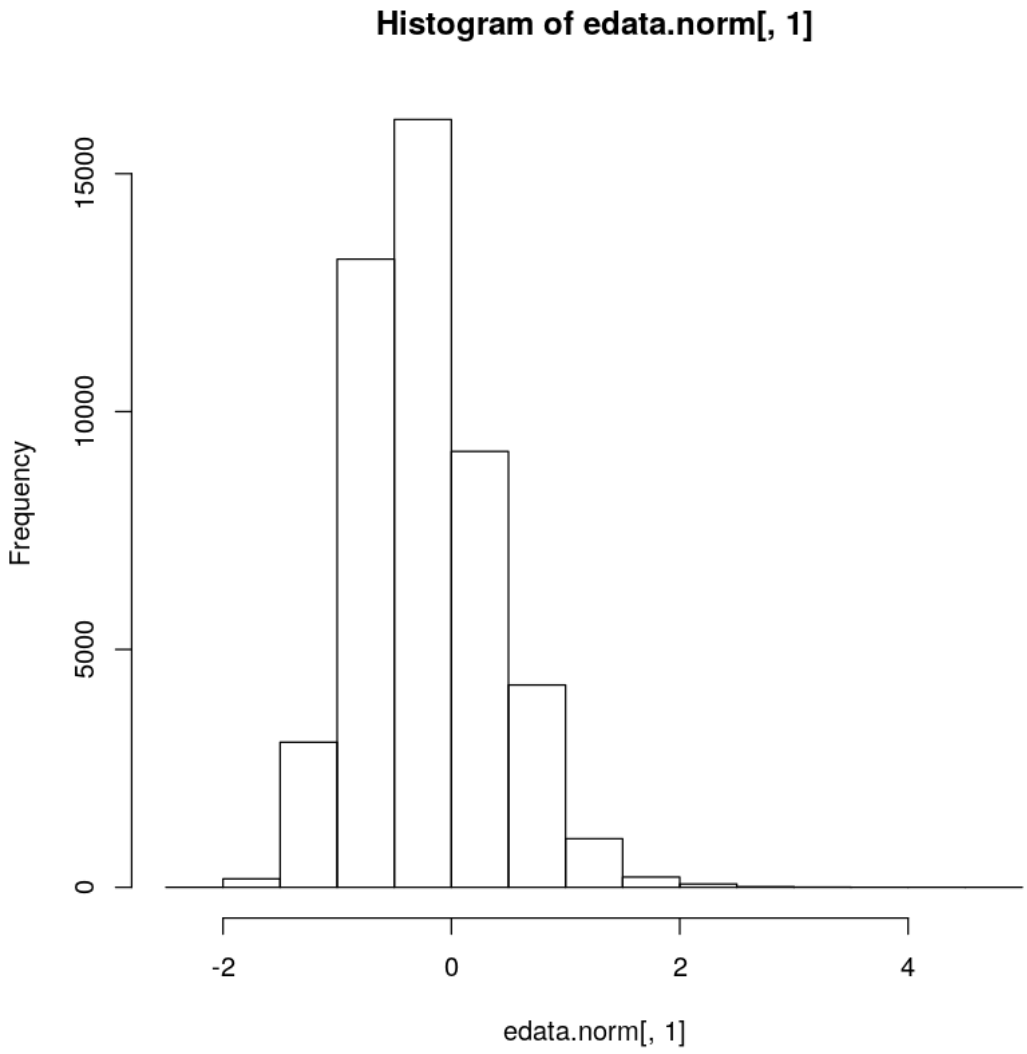
[18]:
```
ggplot(edata.norm.gathered, aes(x=sample, y=value)) +
    geom_boxplot() +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    coord_flip()
```



[19]:
```
ggplot(edata.norm.gathered, aes(x=sample, y=value)) +
    geom_violin() +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    coord_flip()
```



[20]:
```
hist(edata.norm[,1])
```



This process does not seem correct as the shape of histogram has changed abruptly. Perhaps some other normalization algoritm for exponentially distrubuted data would be more appropriate. In the following part of the analysis the non-standardized data will be used.

0.2.4 Apply the jackstraw tests for PCA or related methods. Make sure how you decide to choose the number of significant PCs (r in the algorithm). Discuss the results. Create the histograms of p-values, q-values, null statistics, and observed statistics.

[32]:
```
mod = model.matrix(~as.factor(diagnosis), data=pheno)
num.sv(edata, mod, method="be")
num.sv(edata, mod, method="leek")
```

10

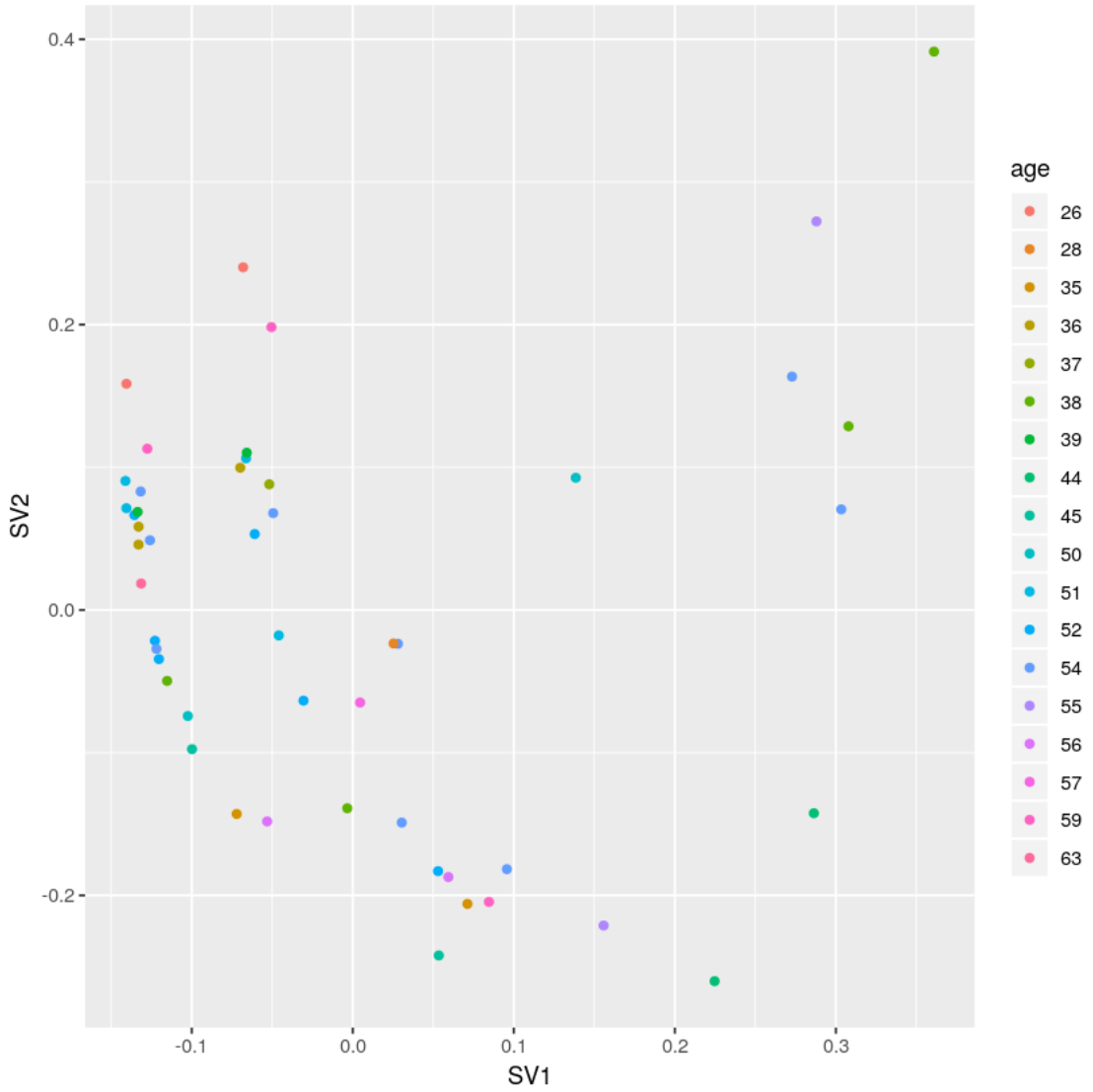
1

[33]:
```
mod = model.matrix(~as.factor(diagnosis), data=pheno)
mod0 = model.matrix(~1, data=pheno)
sva_output = sva(edata.norm, mod, mod0, n.sv=10)
```

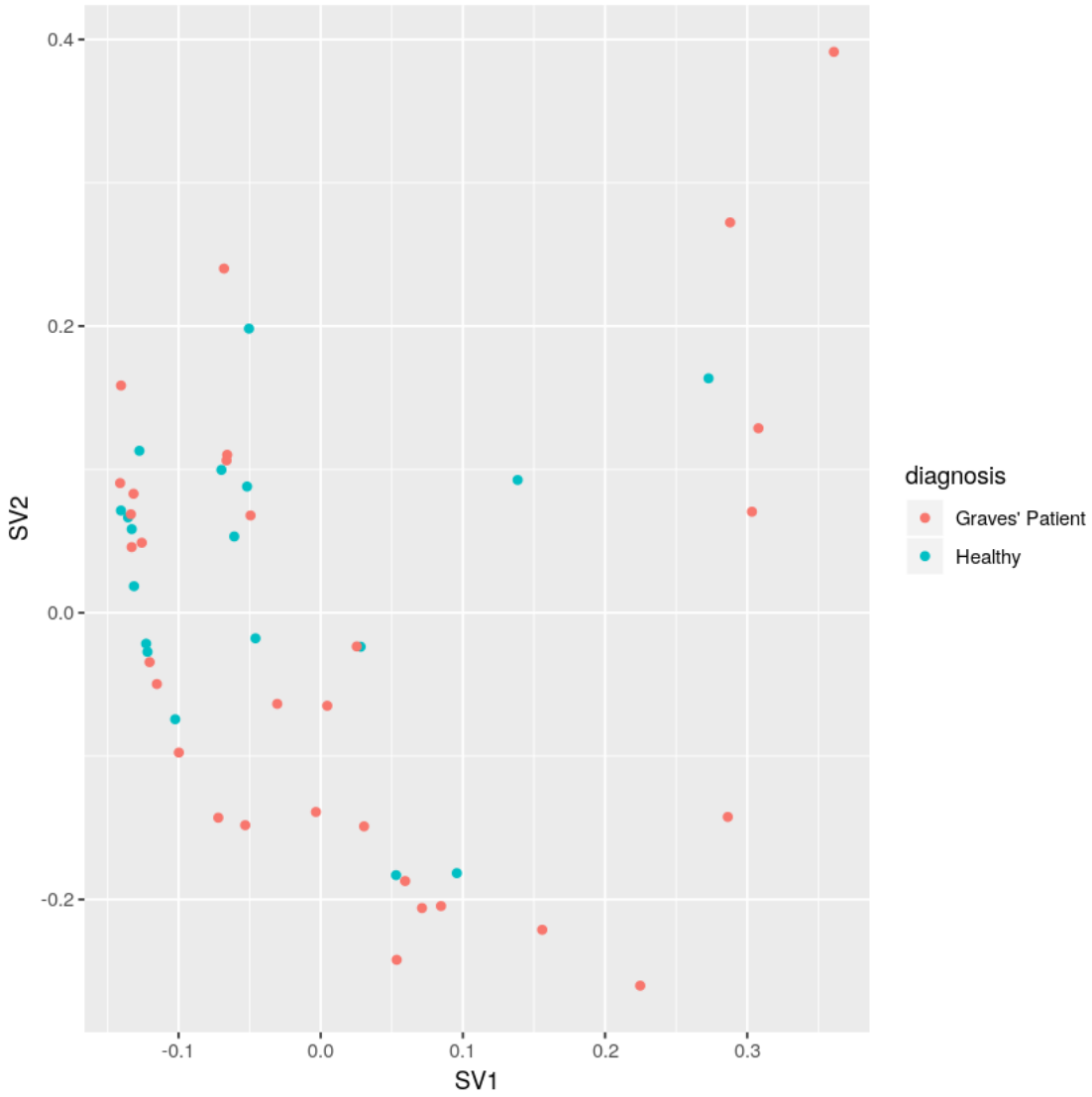
```
Number of significant surrogate variables is:  10
Iteration (out of 5 ):1  2  3  4  5
```

[34]:
```
sva_batch <- tibble(SV1=sva_output$sv[,1],
                    SV2=sva_output$sv[,2],
                    age=as.factor(pheno$age),
                    diagnosis=as.factor(pheno$diagnosis),
                    cell_type=as.factor(pheno$cell_type))
```

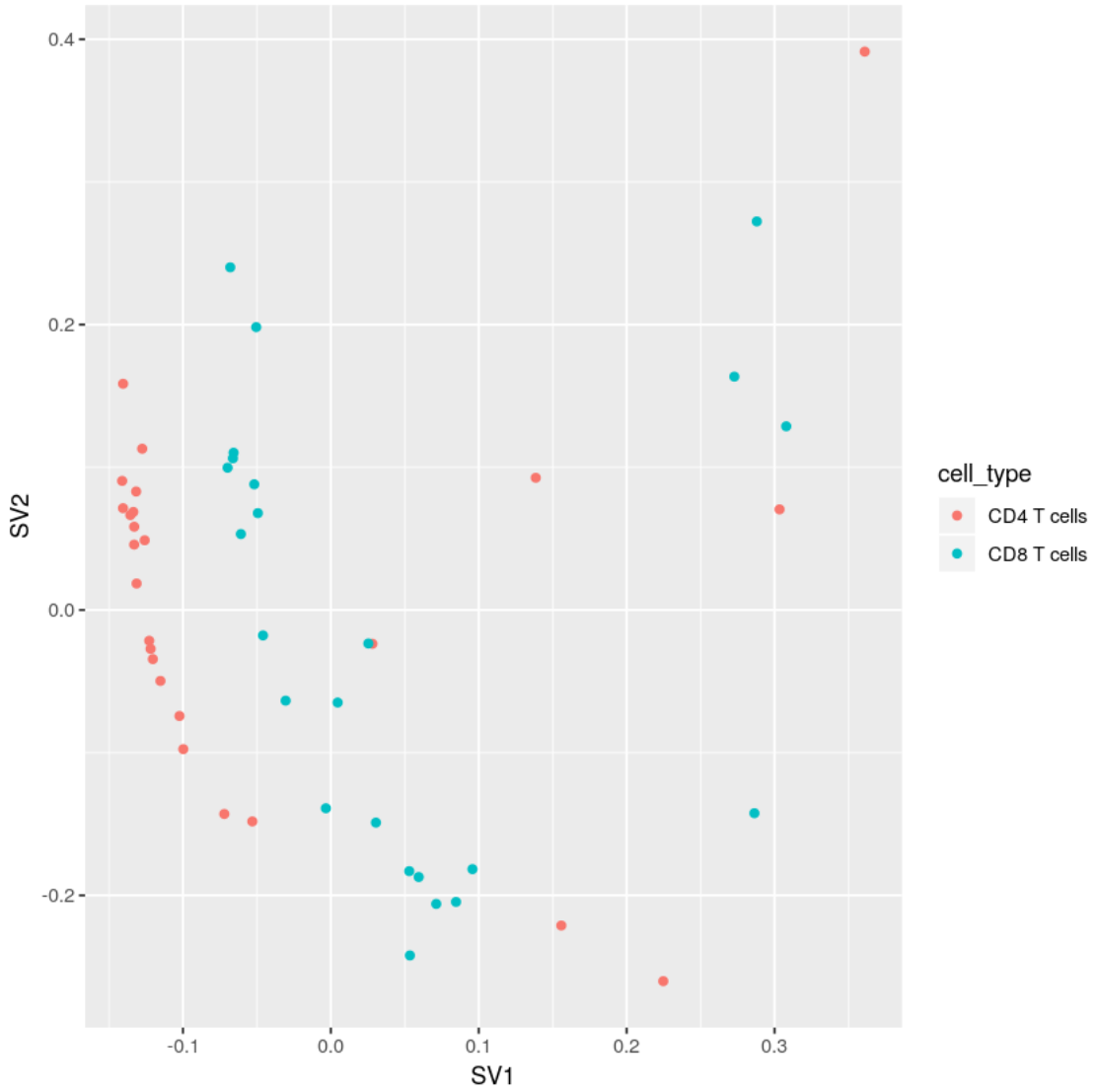
[35]:
```
ggplot(sva_batch) + geom_point(aes(x=SV1,y=SV2, col=age))
```



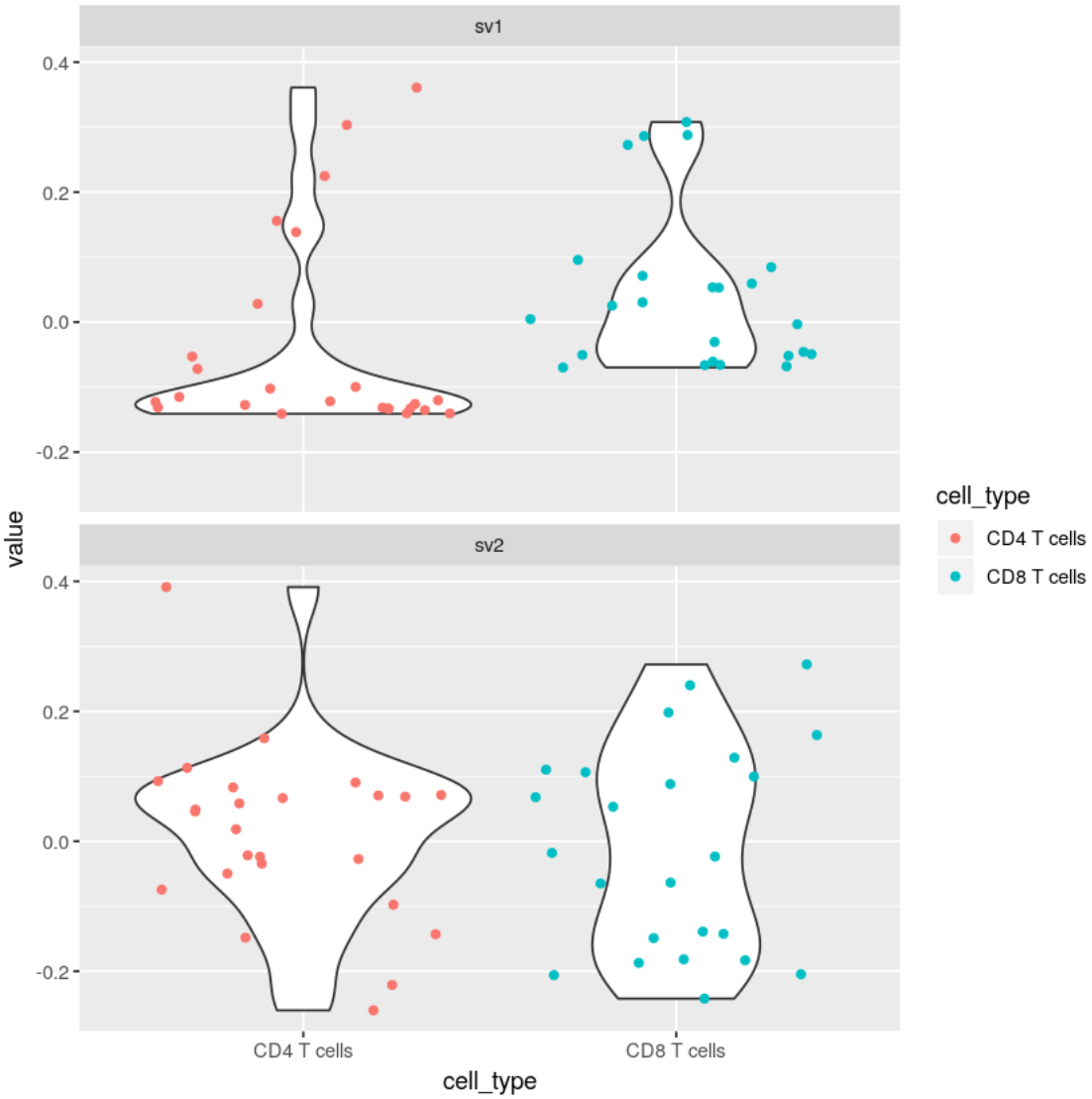
[36]:
```
ggplot(sva_batch) + geom_point(aes(x=SV1,y=SV2, col = diagnosis))
```



[37]:
```
ggplot(sva_batch) + geom_point(aes(x=SV1,y=SV2, col=cell_type))
```



[41]:
```
sva_batch <- tibble(sv1 = sva_output$sv[,1], sv2 = sva_output$sv[,2], cell_type␣
↪= as.factor(pheno$cell_type))
sva_batch <- gather(sva_batch,"sv","value",-cell_type)
ggplot(sva_batch) + geom_violin(aes(x=cell_type,y=value)) +␣
↪geom_jitter(aes(x=cell_type,y=value,col=cell_type)) + facet_wrap(~sv, ncol =␣
↪1)
```

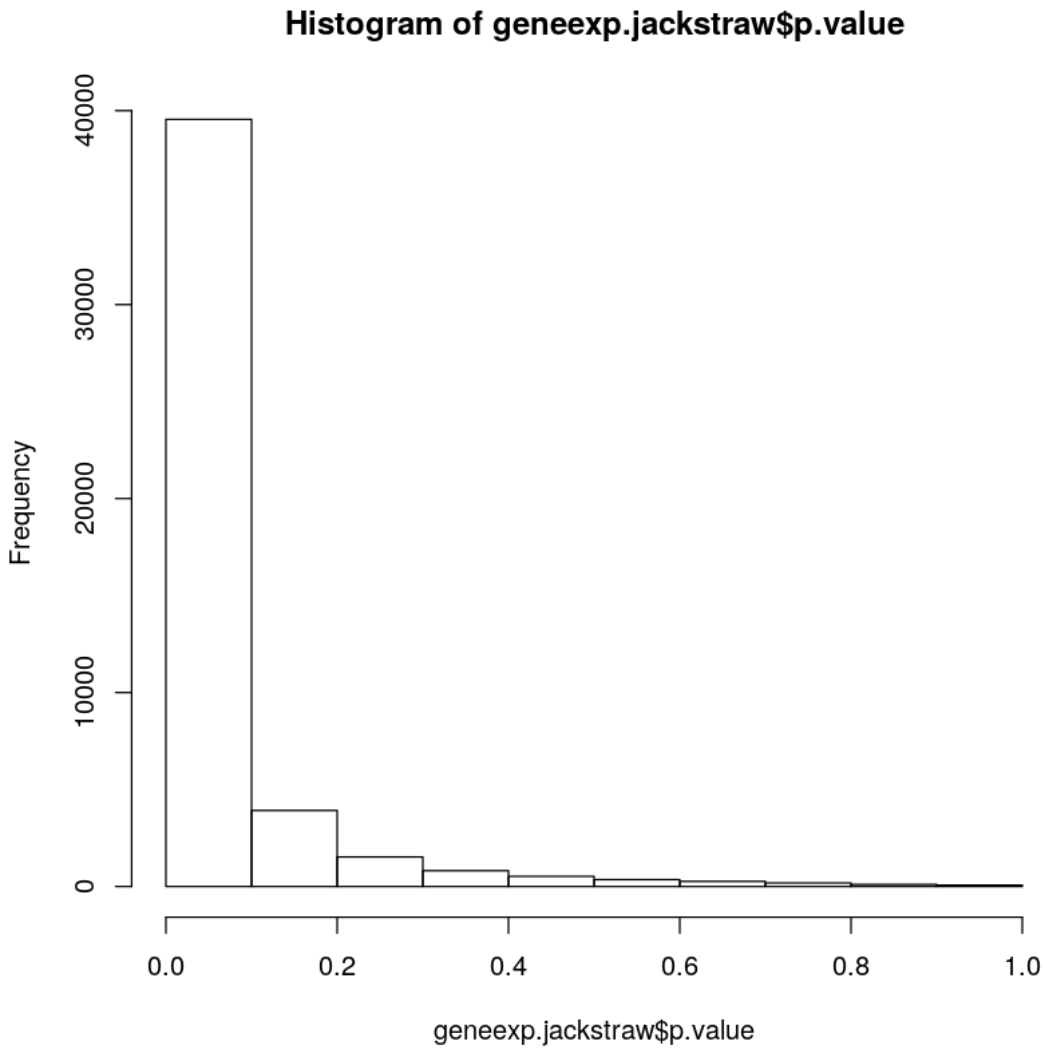


The first two surrogate variables are nicely clustered by the cell type. It is a good sign as the main purpose of this analysis was to discover genes differentially expressed in CD4+ and CD8+. It means that the data is free from technical noise.

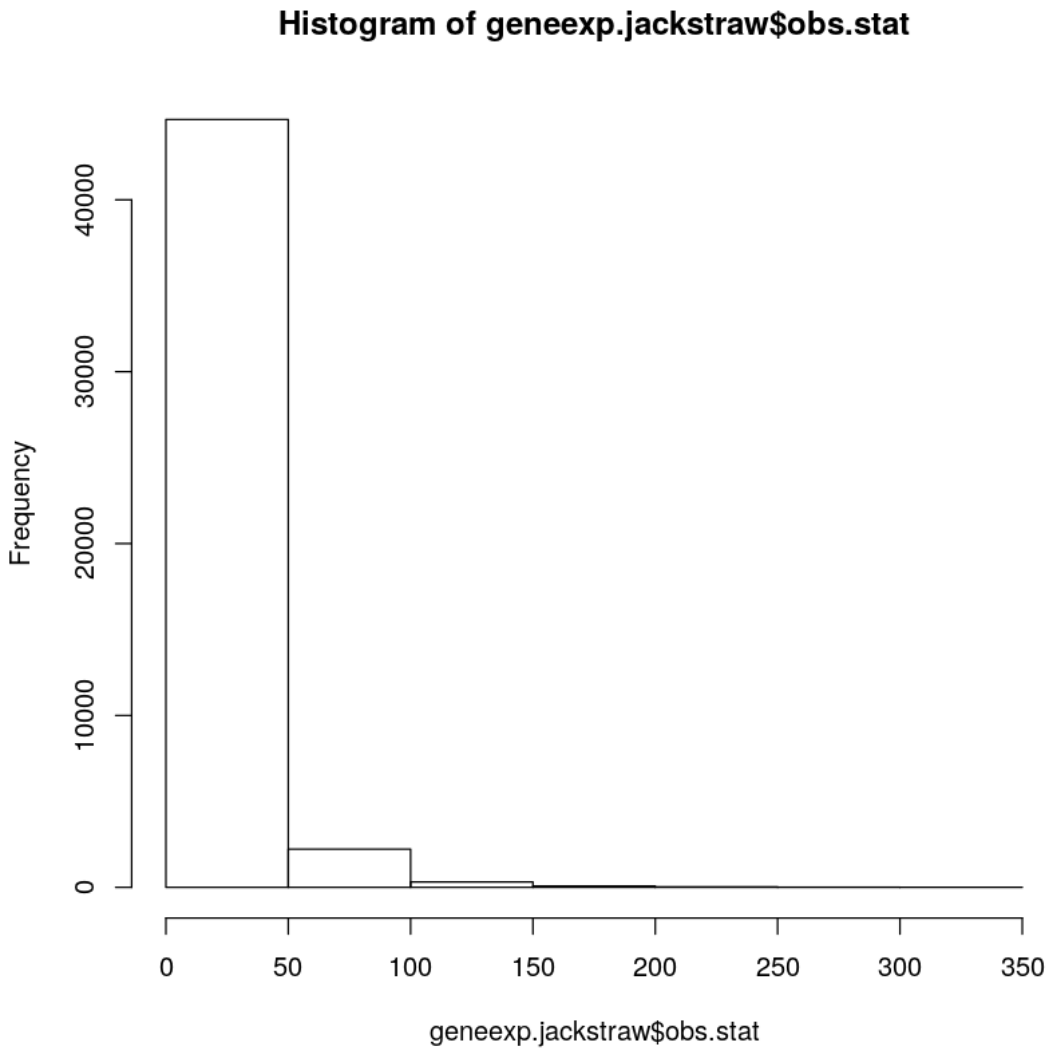
[65]:
```
m = dim(edata)[1]
n = dim(edata)[2]

geneexp.jackstraw = jackstraw_pca(edata.norm, r=10,
                                  s=round(m*.1), B=10)
```

```
Computating null statistics (10 total iterations): 1  2  3  4  5  6  7  8  9  10
```

[66]:
```
# the m p-values of association tests between variables
# and their principal components
hist(geneexp.jackstraw$p.value,10)
```

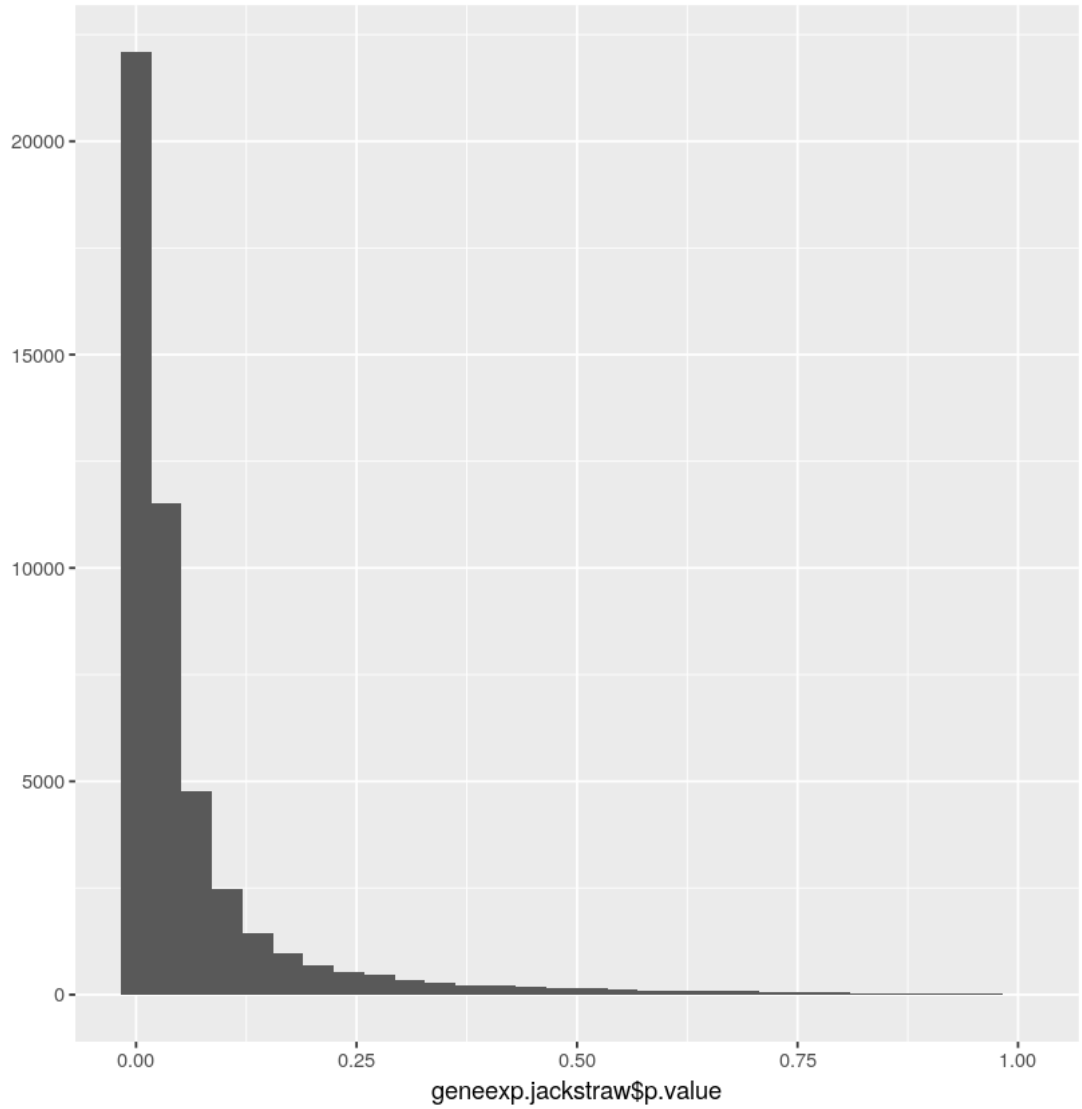


[67]:
```
# the observed F-test statistics
hist(geneexp.jackstraw$obs.stat,10)
```



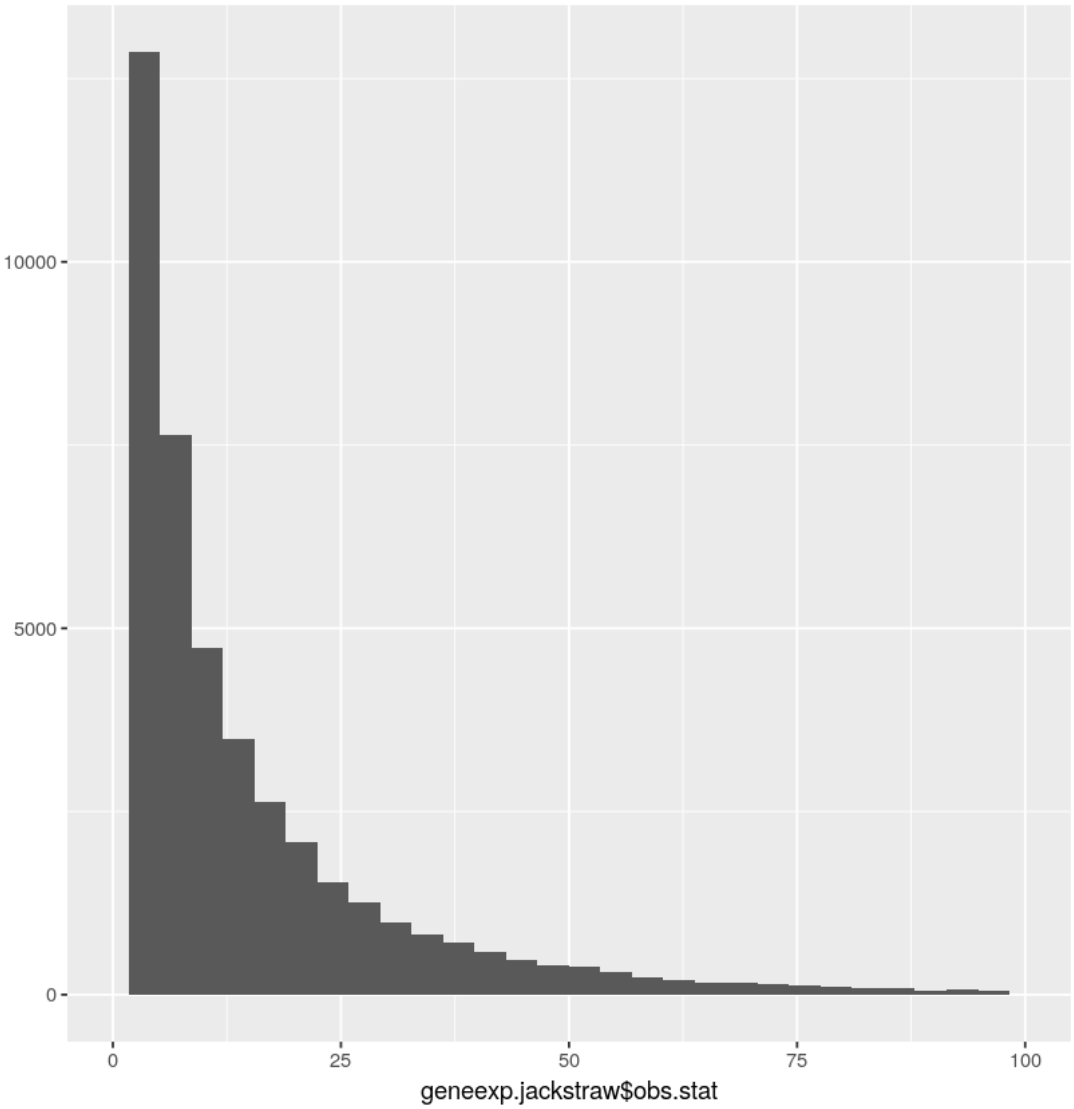
[68]:
```
qplot(geneexp.jackstraw$p.value, geom="histogram")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



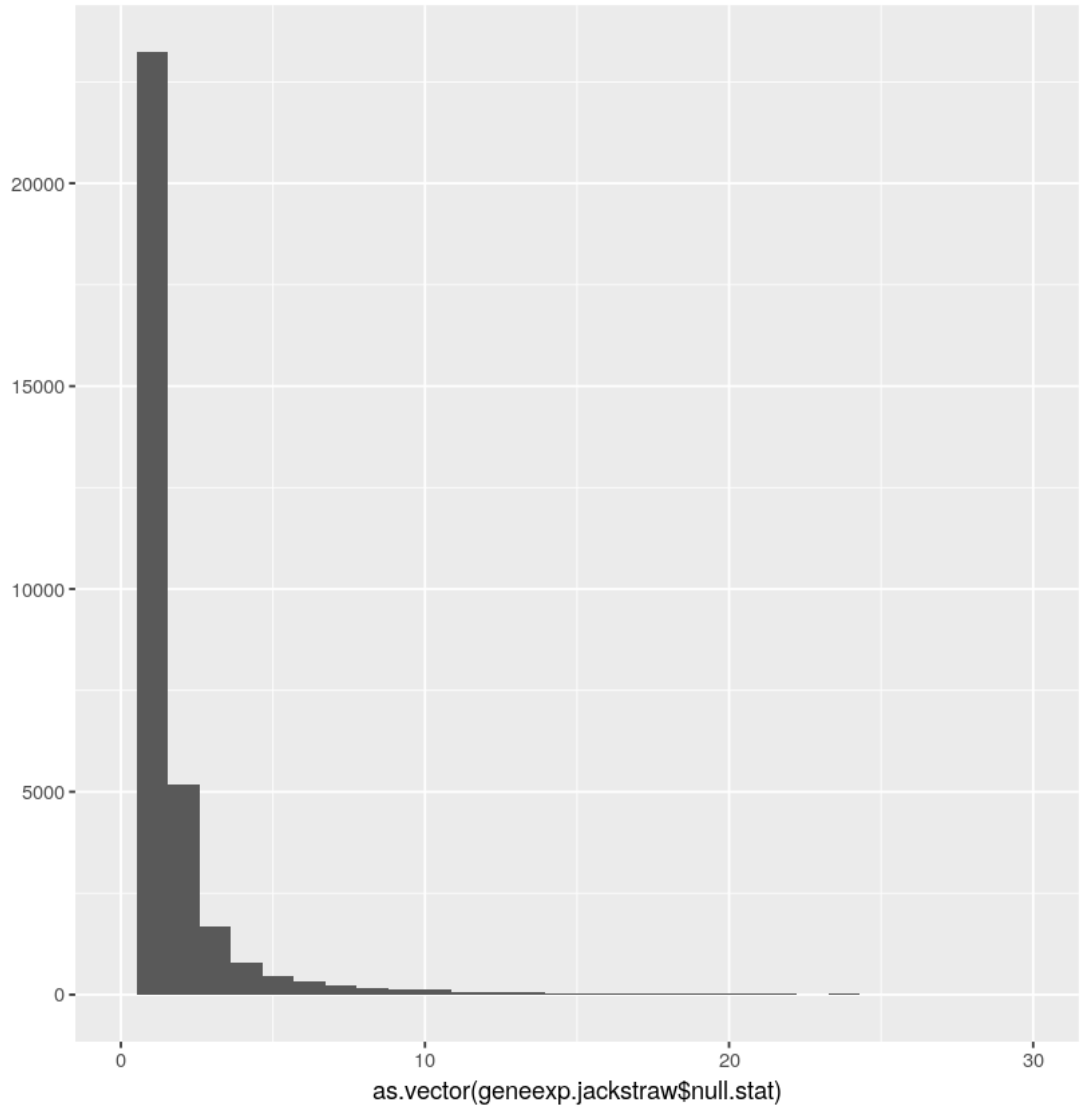
[69]:
```
qplot(geneexp.jackstraw$obs.stat, geom="histogram", xlim=c(0,100))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Warning message:
"Removed 426 rows containing non-finite values (stat_bin)."Warning message:
"Removed 2 rows containing missing values (geom_bar)."
```



[70]:
```
qplot(as.vector(geneexp.jackstraw$null.stat), geom="histogram", xlim=c(0,30))
```

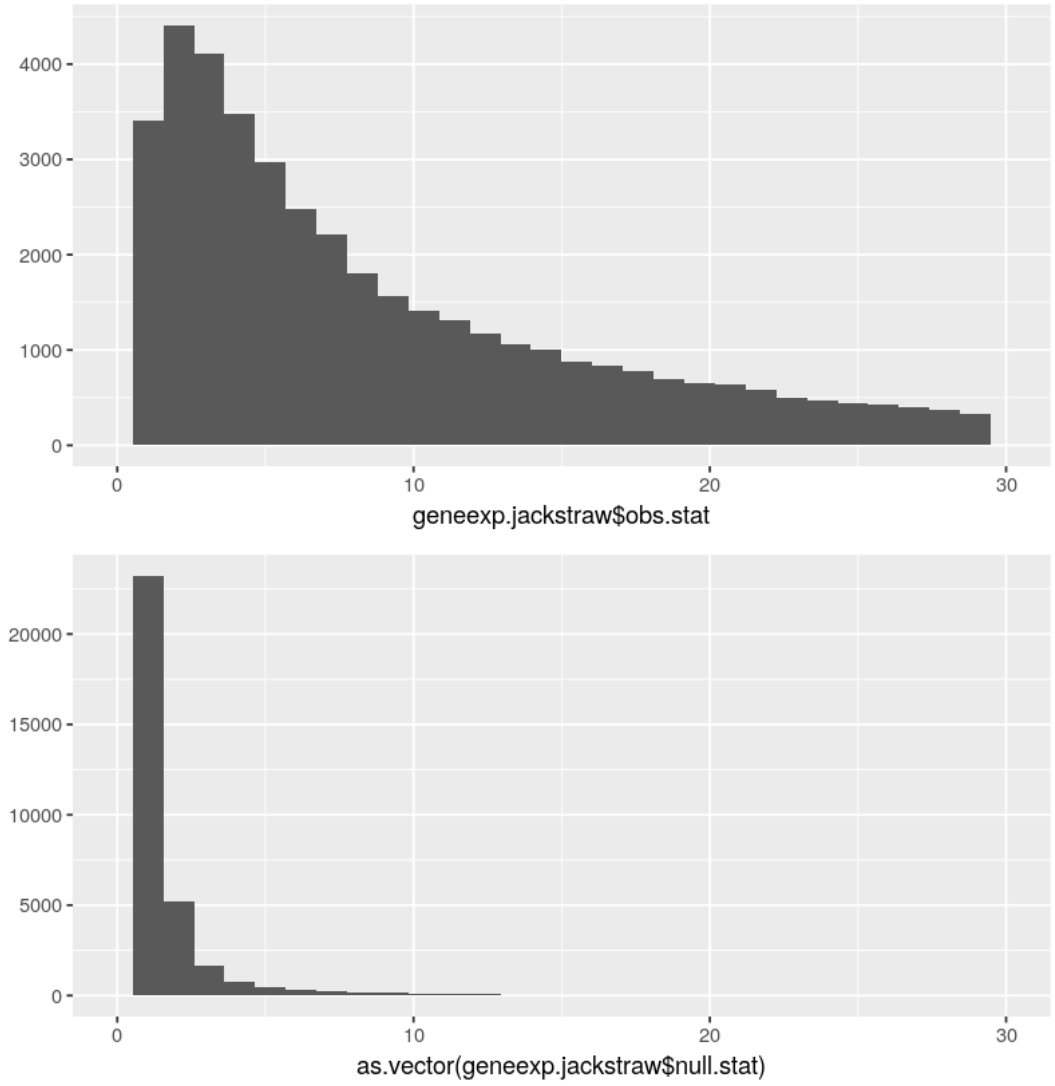
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Warning message:
"Removed 61 rows containing non-finite values (stat_bin)."Warning message:
"Removed 2 rows containing missing values (geom_bar)."
```



[71]:
```
# lets combine 2 histograms, severly limiting the x-axis
obs.hist <- qplot(geneexp.jackstraw$obs.stat, geom="histogram", xlim=c(0,30))
null.hist <- qplot(as.vector(geneexp.jackstraw$null.stat), geom="histogram",␣
 ↪xlim=c(0,30))
print(obs.hist / null.hist)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Warning message:
"Removed 6408 rows containing non-finite values (stat_bin)."Warning message:
"Removed 2 rows containing missing values (geom_bar)."`stat_bin()` using `bins =
30`. Pick better value with `binwidth`.
Warning message:
"Removed 61 rows containing non-finite values (stat_bin)."Warning message:
```

```
"Removed 2 rows containing missing values (geom_bar)."
```



[72]:
```
sum(qvalue(geneexp.jackstraw$p.value)$qvalue < .01)
```

47323

[73]:
```
pval.pc1 = jackstraw_pca(edata, r1=1, r=2, s=round(m*.1), B=10)$p.value
```

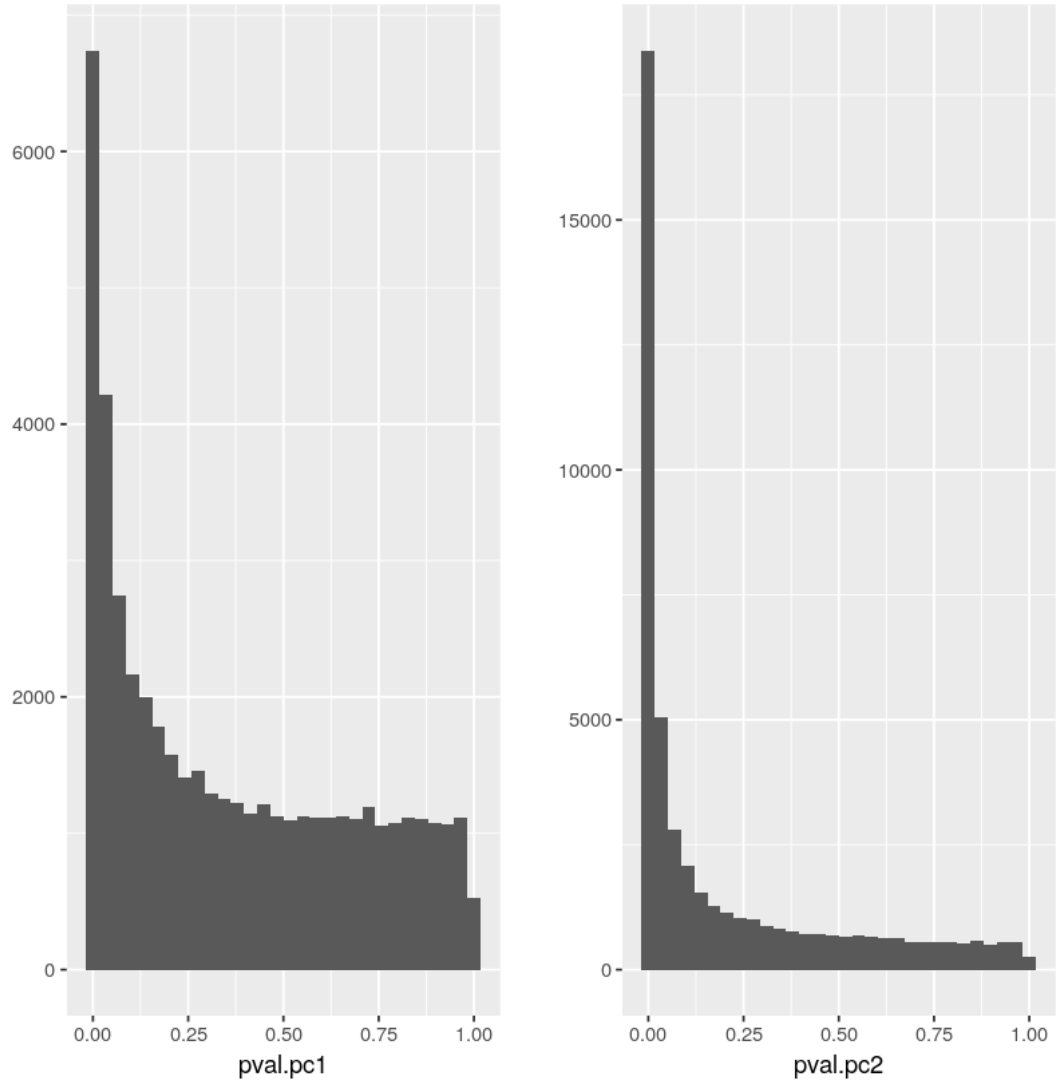
```
Computating null statistics (10 total iterations): 1  2  3  4  5  6  7  8  9  10
```

[74]:
```
pval.pc2 = jackstraw_pca(edata, r1=2, r=2, s=round(m*.1), B=10)$p.value
```

```
Computating null statistics (10 total iterations): 1  2  3  4  5  6  7  8  9  10
```

[76]:
```
# lets combine 2 histograms, severly limiting the x-axis
pc1.hist <- qplot(pval.pc1, geom="histogram")
pc2.hist <- qplot(pval.pc2, geom="histogram")
print(pc1.hist + pc2.hist)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



1 Testing for each of two PCs

As seen on the SVA plots, we wold be very interestes in identifying genes that are related to the 1st and 2nd PCs.

[128]:
```
q.pc1 <- qvalue(pval.pc1)
q.pc2 <- qvalue(pval.pc2)

q.pc1$pi0
q.pc2$pi0
```

0.660143010728853

0.314205093317101

[99]:
```
sum(q.pc1$qvalue < .01)
sum(q.pc2$qvalue < .01)
```

1738

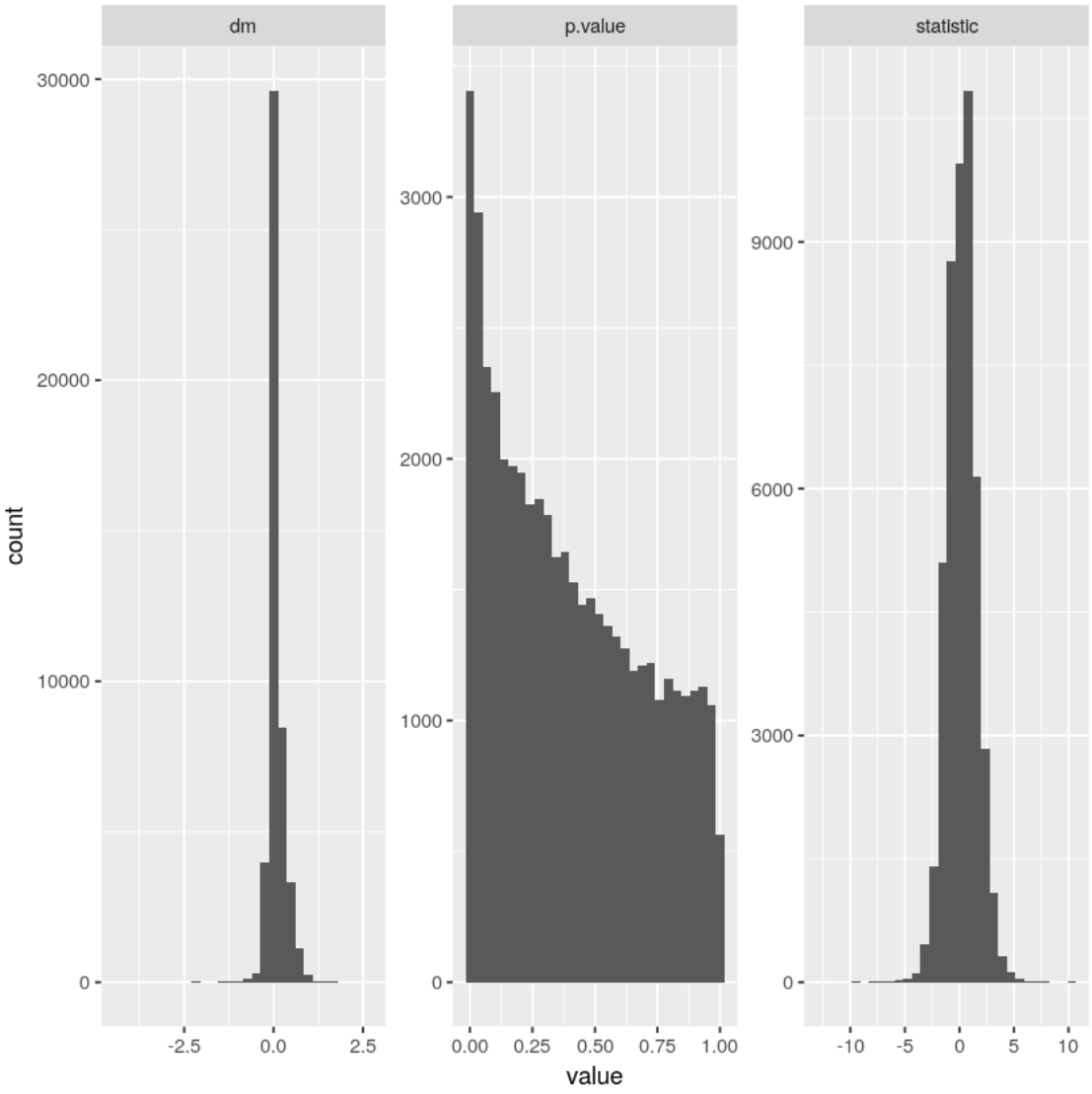
16597

1.1 Was the jackstraw really helpful?

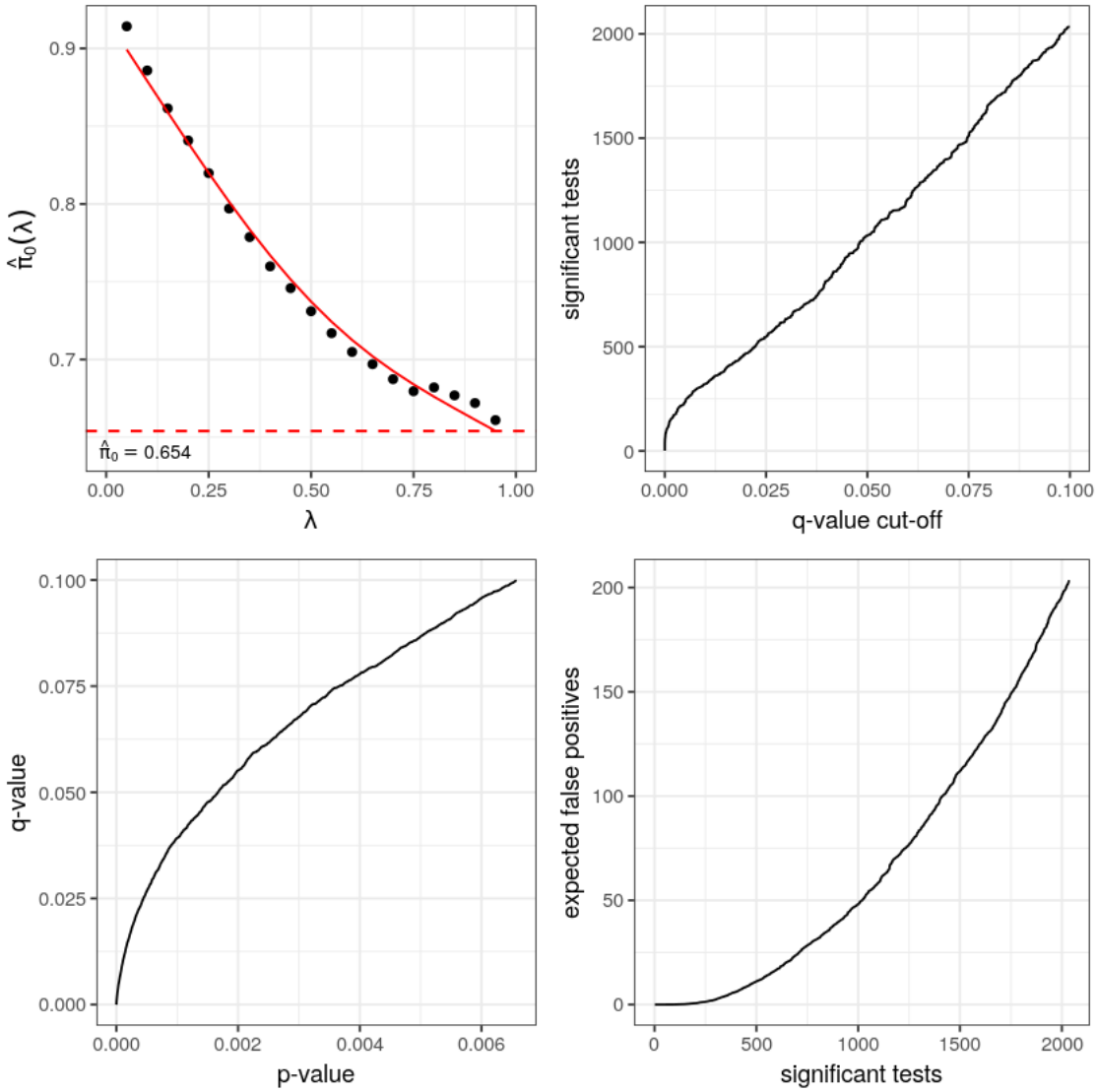
Let's calculate te p-values and q-values the conventional way as done in "Statistical tests and multiple hypothesis tests" lab.

[110]:
```
tout = rowttests(x = edata, fac = as.factor(pheno$cell_type))
```

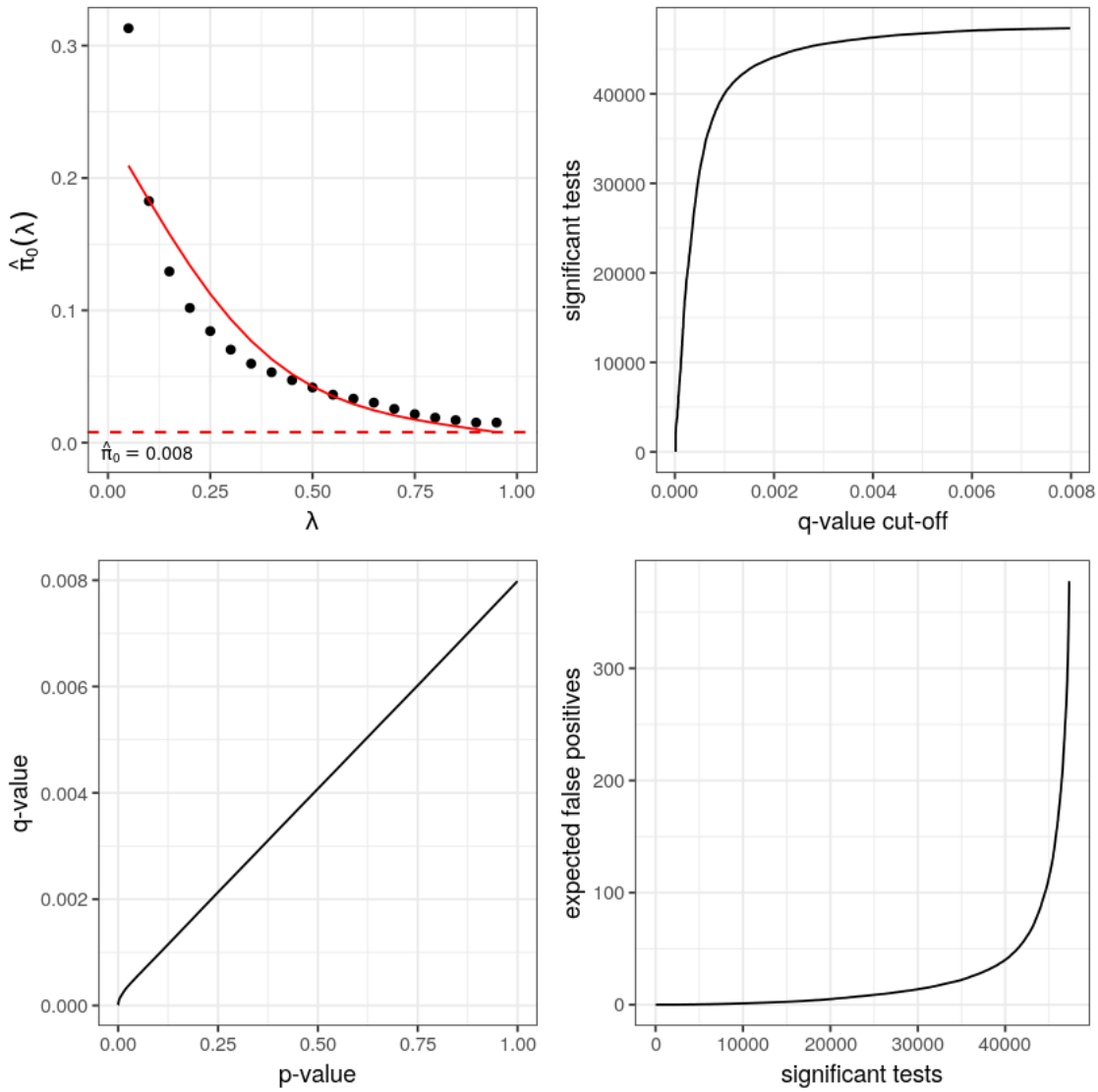
[111]:
```
ttidy <- gather(tout)
ggplot(ttidy) + geom_histogram(bins = 30,aes(x=value)) + facet_wrap(~ key,␣
↪scales="free")
```



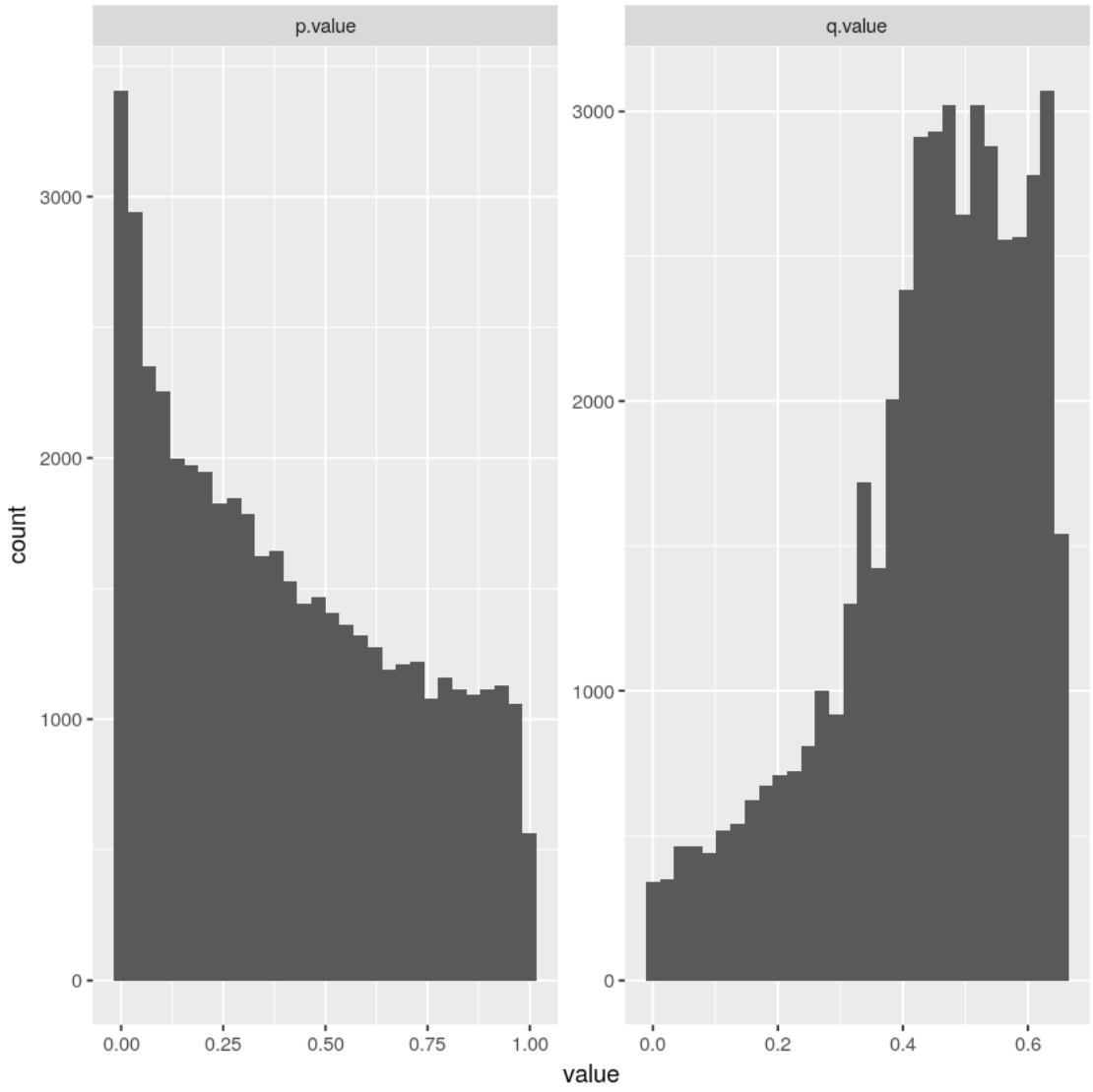
[112]:
```
q.obj <- qvalue(tout$p.value)
plot(q.obj)
```



[126]:
```
jackstraw.q <- qvalue(geneexp.jackstraw$p.value)
plot(jackstraw.q)
```



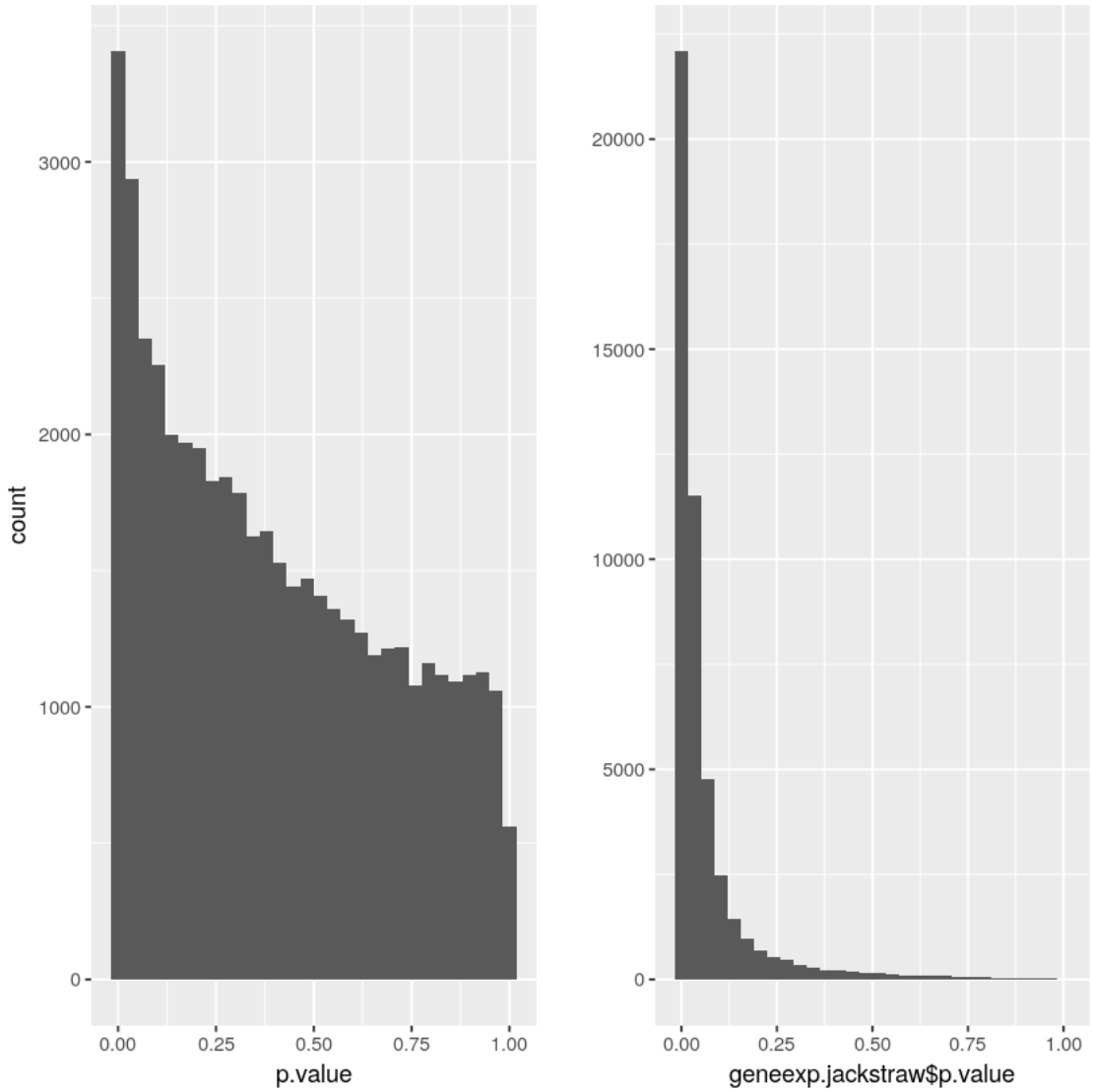
[123]:
```
tout$q.value <- q.obj$qvalues
ttidy <- gather(tout[,c("p.value","q.value")])
ggplot(ttidy) + geom_histogram(bins = 30,aes(x=value)) + facet_wrap(~ key,␣
↪scales="free")
```



1.2 Plots for the comparison

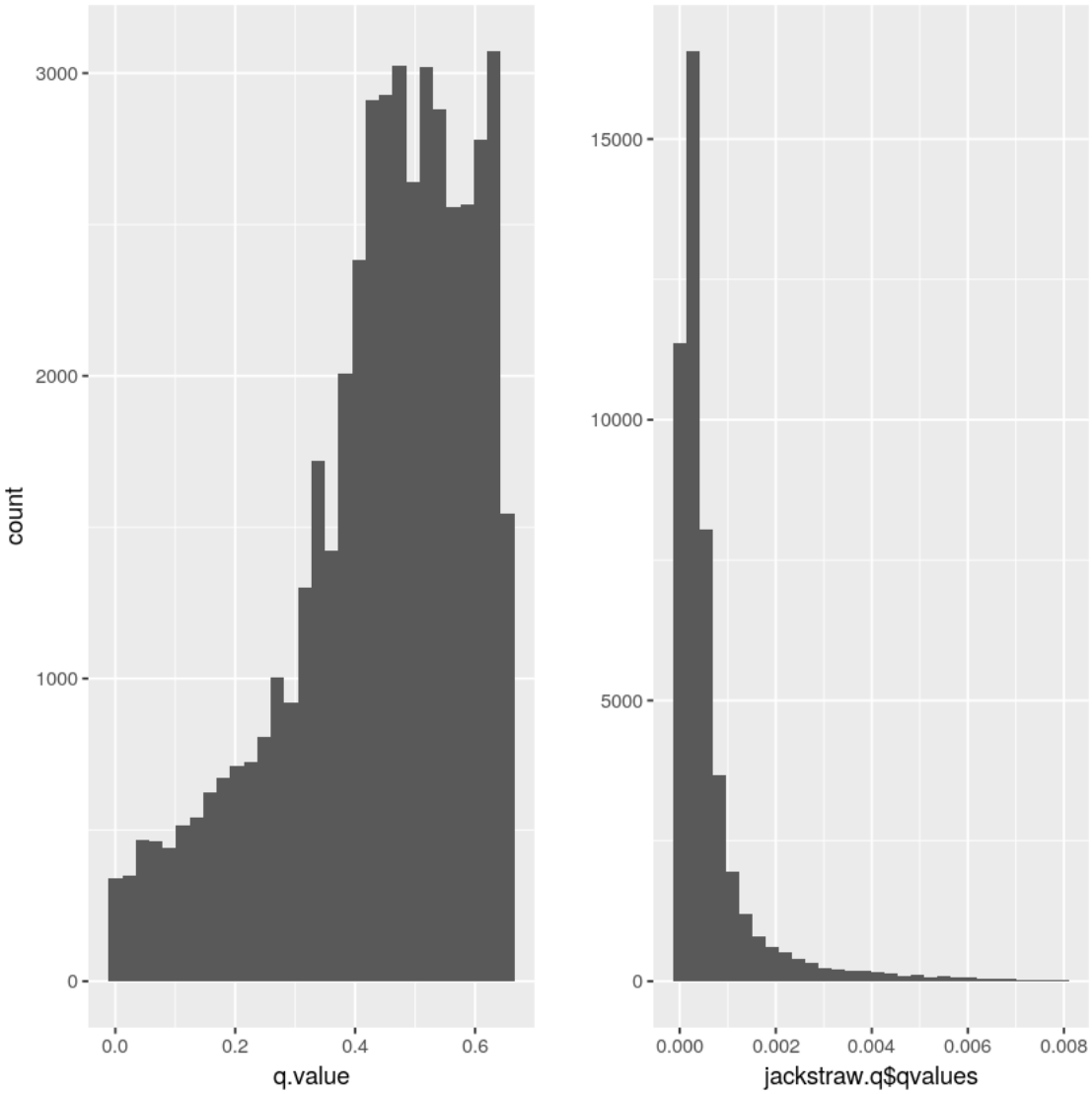
[124]:
```
plot1 <- ggplot(tout) + geom_histogram(bins = 30,aes(x=p.value))
plot2 <- qplot(geneexp.jackstraw$p.value, geom="histogram")
grid.arrange(plot1, plot2, nrow=1)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



[127]:
```
plot1 <- ggplot(tout) + geom_histogram(bins = 30,aes(x=q.value))
plot2 <- qplot(jackstraw.q$qvalues, geom="histogram")
grid.arrange(plot1, plot2, nrow=1)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As seen at the final sections, jackstraw helped us to eliminate systemic variation and control the effect of hidden variables. The jackstraw allowed us to find a number of genes differentially expressed between the cells of type CD4+ and CD8+.