PH525.5x Section 5: Inference for genomics with Bioconductor

Lauren Kemperman

4/15/2021

- How to perform inference on ExpressionSet and SummarizedExperiment in Bioconductor
- Simple gene-wie t-tests with **genefilter** package
- Robust moderated t-tests with **limma** package
- use ROAST method for gene set enrichment analysis of single and multiple gene sets

Biological vs. technical variability

- Inference on high-throughput data by analyzing experiment comparing technical and biological variability
- t-test takes into account within population variability
- explaining the distinction between technical and biological variability
- Example from genomics of central limit theorem and t-distribution
- Experiment performed on two strains of mice - gene expression experiment done on microarrays
- 12 mice of each strain
- Population = all mice from first strain and all mice from other strain
- For each strain, there is some pop average
- Took RNA from each mice and saved it
- Created pool of all RNA for each strain
- Created 4 technical replicates - same RNA divided into 4
- Measurements for four and four samples - variability for this data is technical variability
- Varaibility from technical and biological replicates
- Population is different - four sub-samples from one sample
- Statistical inference is going to be about the sub-samples
- T-statistic assuming gene expression is normally distributed in log scale
- Small variability between groups -t-statistic is huge
- 12 biological replicated - individuals
- For each gene, compute STD across four technical replicates
- Do same for 12 individual mice
- Variability in biological replicates is much bigger
- Microarray technology is precise but biological replicates include natural variability which has nothing to do with technology
- Must measure this variability to perform statistical inference on the strains
- Within-group variability
- Inference about strains as a population
- Which gene is more likely to reproduce as differentially expressed
- What to do for high throughput data? T-test for each gene, decide which genes are statistically significant

T-tests in genomics

- Multiple comparison problem
- Determine if a gene is differentially expressed in two strains

- Compute t-statistic for each gene and using approximations (central limit theorem or t-distribution approximation) and obtain p-value
- P-value for each gene
- Decide which genes to report
- Permute the samples to eliminate the strain differences completely
- Null distribution must be true after permutation - random split into same groups
- When you are looking at thousands of genes, 5% of so p-values will be less than .05 by chance (called multiple comparison problem)
- Histogram of p-values from experiment and shuffled data
- Density of shuffled data is close to flat - iff null hypothesis is true of every gene, the distribution of p-values has to follow uniform distribution
- Higher number of genes before shuffling on the left - smaller p-values
- Volcano plot - observed effect size (numerator of t-statistic, log ratio of average gene expression values), y axis is the p-values on scale where smaller p-values go up (-log10)
- This tells you fold change at different p-values
- See many points with very small p-values with small effect sizes. May not be worth reporting.
- Improved estimates of standard deviation using t-statistic
- T-statistic not good approach to finding differentially expressed genes because denominator is very variable. Can get large t-statistics driven by fact small denominator by chance. Not very powerful

Moderated t-tests with the limma package

- Moderated t-tests - hierarchical models create test statistics that improve the power of the t-test
- More robust than basic t-testing performed by rowttests()

First, simple t-tests

In this unit, we will show the difference between using the simple t-test and doing differential expression with the `limma` hierarchical model. The reference is Smyth 2004, listed in the footnotes.

Here we also show the basic steps for performing a `limma` analysis. Note that the `limma` package is very powerful, and has hundreds of pages of documentation which we cannot cover in this course, however we recommend that users wanting to explore further should check out this guide.

We start by loading the spike-in data which was introduced in lecture, which has already been normalized.

```
# BiocManager::install("SpikeInSubset")
library(SpikeInSubset)
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: affy
```

```
data(rma95)
fac <- factor(rep(1:2,each=3))
```

We can now perform simple t-tests using the `rowttests` function in the `genefilter` package:

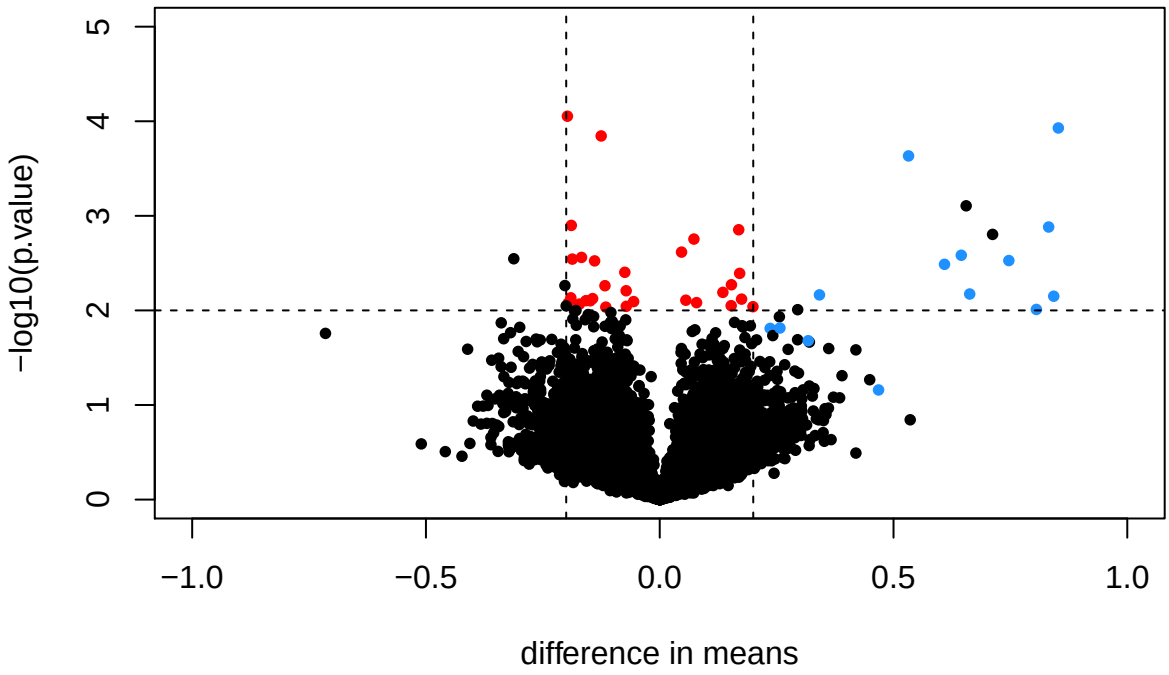
```
library(genefilter)
rtt <- rowttests(exprs(rma95),fac)
```

We will define colors depending on whether the p-value is small, the absolute difference in means is large, and whether the feature is a spike-in value.

```
mask <- with(rtt, abs(dm) < .2 & p.value < .01)
spike <- rownames(rma95) %in% colnames(pData(rma95))
cols <- ifelse(mask,"red",ifelse(spike,"dodgerblue","black"))
```

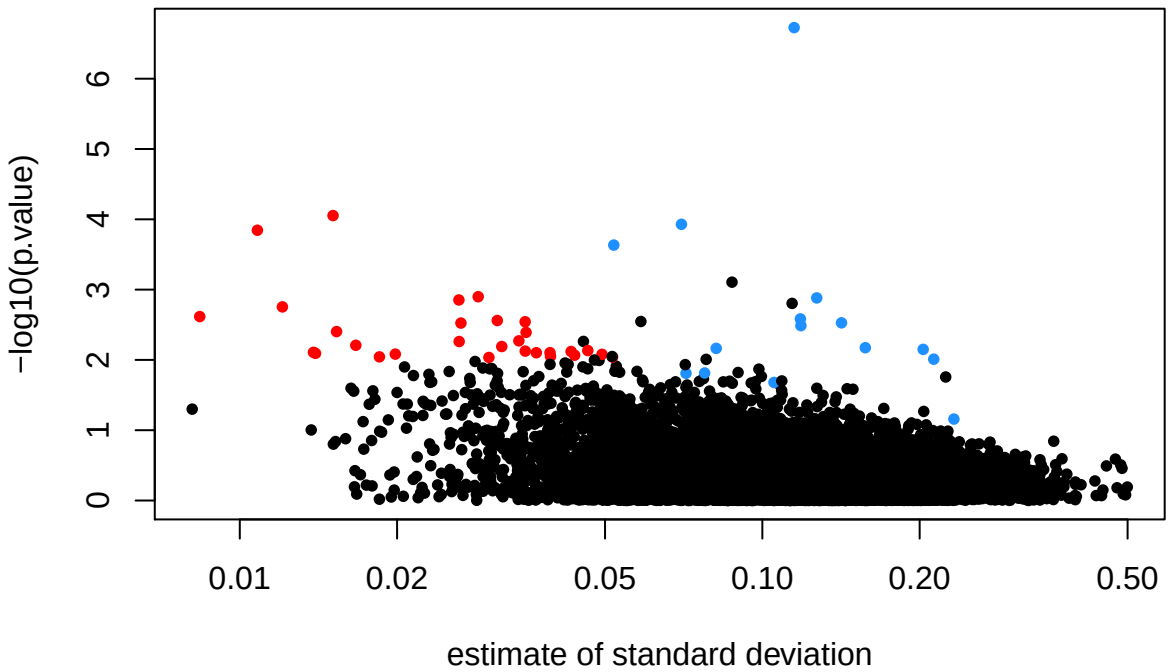
We now plot the results, using the colors defined above. We multiply the `dm` by -1, because we are interested in the difference from the second group to the first (this is the difference used by `lm` and the `limma` package by default). The spike-in genes are in blue, which have mostly small p-value and large difference in means. The red points indicate genes which have small p-values but also small differences in means. We will see how these points change after using `limma`.

```
with(rtt, plot(-dm, -log10(p.value), cex=.8, pch=16,
     xlim=c(-1,1), ylim=c(0,5),
     xlab="difference in means",
     col=cols))
abline(h=2,v=c(-.2,.2), lty=2)
```



Note that the red genes have mostly low estimates of standard deviation.

```
rtt$s <- apply(exprs(rma95), 1, function(row) sqrt(.5 * (var(row[1:3]) + var(row[4:6]))))
with(rtt, plot(s, -log10(p.value), cex=.8, pch=16,
               log="x",xlab="estimate of standard deviation",
               col=cols))
```



limma steps

The following three steps perform the basic `limma` analysis. We specify `coef=2` because we are interested in the difference between groups, not the intercept.

```
library(limma)
```

```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```
fit <- lmFit(rma95, design=model.matrix(~ fac))
colnames(coef(fit))
```

```
## [1] "(Intercept)" "fac2"
```

```
fit <- eBayes(fit)
tt <- topTable(fit, coef=2)
tt
```

```
##               logFC   AveExpr          t      P.Value    adj.P.Val        B
## 1708_at  -7.0610613  7.945276 -73.529269 7.816370e-17 9.868948e-13 8.646866
## 36202_at  0.8525527  9.373033   9.975114 4.935683e-07 3.115897e-03 4.587736
## 36311_at  0.8318298  8.564315   8.363252 3.017008e-06 1.269758e-02 3.567790
## 33264_at  0.7118997  4.918953   7.434888 9.666328e-06 2.706595e-02 2.835849
## 32660_at  0.6554022  8.680132   7.356180 1.071834e-05 2.706595e-02 2.768151
## 38734_at  0.7467142  6.255772   7.185131 1.345115e-05 2.830571e-02 2.617789
## 1024_at   0.8426550  9.697281   6.730664 2.503461e-05 4.400123e-02 2.195944
## 36085_at  0.6449402 12.193130   6.653830 2.787976e-05 4.400123e-02 2.121308
## 33818_at  0.5321749 12.285643   6.454504 3.699480e-05 5.189960e-02 1.923063
## 39058_at  0.6090625  7.534532   6.278815 4.767986e-05 5.687699e-02 1.742696
```

`topTable` will return the top genes ranked by whichever value you define. You can also ask topTable to return all the values, sorted by `"none"`. Note that a column automatically is included which gives the *adjusted p-values* for each gene. By default the method of Benjamini-Hochberg is used, by calling the `p.adjust` function.

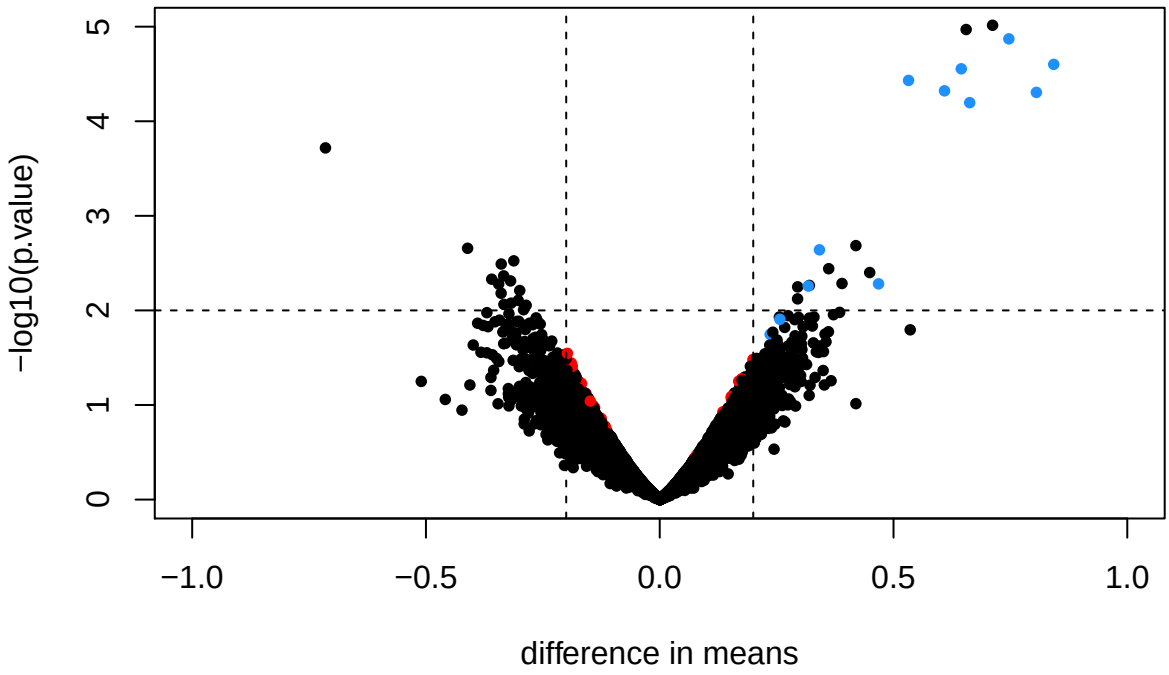
```
# ?topTable
dim(topTable(fit, coef=2, number=Inf, sort.by="none"))
```

```
## [1] 12626     6
```

```
# ?p.adjust
```

Here we will compare the previous volcano plot with the `limma` results. Note that the red points are now all under the line where `-log10(p.value)` is equal to 2. Also, the blue points which represent real differences have p-values which are even higher than before.

```
limmares <- data.frame(dm=coef(fit)[,"fac2"], p.value=fit$p.value[,"fac2"])
with(limmares, plot(dm, -log10(p.value),cex=.8, pch=16,
     col=cols,xlab="difference in means",
     xlim=c(-1,1), ylim=c(0,5)))
abline(h=2,v=c(-.2,.2), lty=2)
```



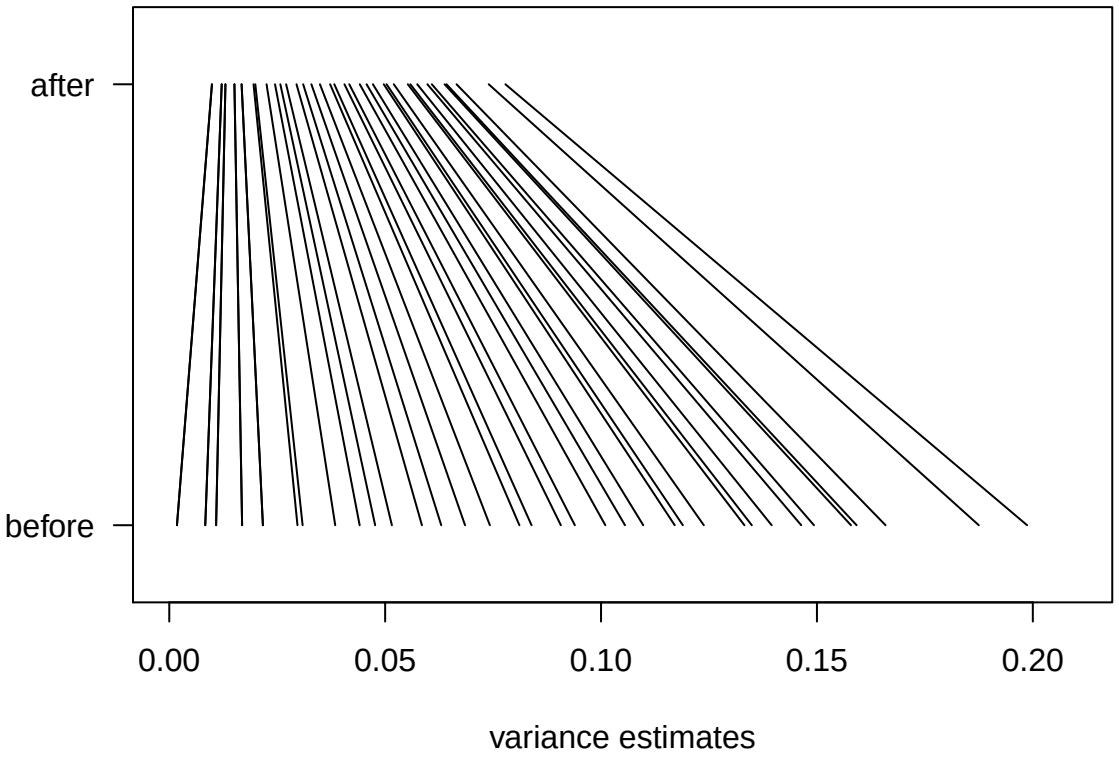
Finally, we will construct a plot which shows how `limma` shrinks the variance estimates towards a common value, eliminating false positives which might arise from too-low estimates of variance.

Here we pick, for each of 40 bins of different variance estimates, a single gene which falls in that bin. We remove bins which do not have any such genes.

```
n <- 40
qs <- seq(from=0,to=.2,length=n)
idx <- sapply(seq_len(n),function(i) which(as.integer(cut(rtt$s^2,qs)) == i)[1])
idx <- idx[!is.na(idx)]
```

Now we will plot a line, from the initial estimate of variance for these genes to the estimate after running `limma`.

```
par(mar=c(5,5,2,2))
plot(1,1,xlim=c(0,.21),ylim=c(0,1),type="n",
     xlab="variance estimates",ylab="",yaxt="n")
axis(2,at=c(.1,.9),c("before","after"),las=2)
segments((rtt$s^2)[idx],rep(.1,n),
         fit$s2.post[idx],rep(.9,n))
```



Assessment: Inference for differential expression

```
library(Biobase)
library(maPooling)
data(maPooling)
pd=pData(maPooling)
pooled=which(rowSums(pd)==12)
individuals=which(rowSums(pd)==1)
##remove replicates
individuals=individuals[-grep("tr",names(individuals))]
pool = exprs(maPooling)[,pooled]
indiv = exprs(maPooling)[,individuals]
strain = ifelse(grepl("a",rownames(pData(maPooling))),0,1)

g_pool = strain[pooled]
g_indiv = strain[individuals]

tech<-rowSds(pool[, g_pool == 1])
bio<-rowSds(indiv[, g_indiv == 1])

mean(bio>tech)
```

```
## [1] 0.7994725
```

```
library(genefilter)
pvals<-rowttests(pool,factor(g_pool))$p.value
library(qvalue)
qvals = qvalue(pvals)$qvalues
```

```
sum(qvals<0.05)
```

```
## [1] 2169
```

```
index=which(qvals<0.05)

pvals2<-rowttests(indiv[index,],factor(g_indiv))$p.value
mean(pvals2>0.05)
```

```
## [1] 0.2784693
```

```
library(limma)
X = model.matrix(~g_indiv)
fit = lmFit(indiv,design=X)
eb = eBayes(fit)
pvals2= eb$p.value[,2]
qvals2 = qvalue(pvals2)$qvalue
sum( qvals2<0.05 & qvals<0.05)/sum(qvals<0.05)
```

```
## [1] 0.4882434
```