

More limma & multiple testing

Keegan Korthauer

7 February 2022

with slide contributions from Gabriela Cohen Freue, Jenny Bryan, and Sara Mostafavi



Announcements

- Issues with the livestream?: [Post here](#)
- Midterm survey: Link posted in [Discussion repo](#)
 - Help us improve your learning experience!
 - Please fill out before Feb 14 at 11:59pm
- Projects: feedback from your project mentors is posted on your initial project proposals (see the issue in your team's repository)
 - reminder: finalized proposals due Friday
- Reminder that the [paper critique](#) is due today

Learning objectives

- How to use `limma` to perform genome-wide differential expression testing on microarray data
- Understand the key differences between `limma` and `lm`
- Explain why multiple testing increases the number of errors we make by chance
- Be able to adjust for multiple comparisons by controlling the False Discovery Rate
 - e.g. using Benjamini-Hochberg or Storey's q-value

Recall: The hybrid estimator in limma

$$\tilde{s}_g^2 = \frac{d_0 s_0^2 + d s_g^2}{d_0 + d}$$

- recall that (s_0, d_0) are the *prior* parameters for σ_g^2 (random variable):

$$\frac{1}{\sigma_g^2} \sim \frac{1}{d_0 s_0^2} \chi_{d_0}^2$$

Recall: The hybrid estimator in limma

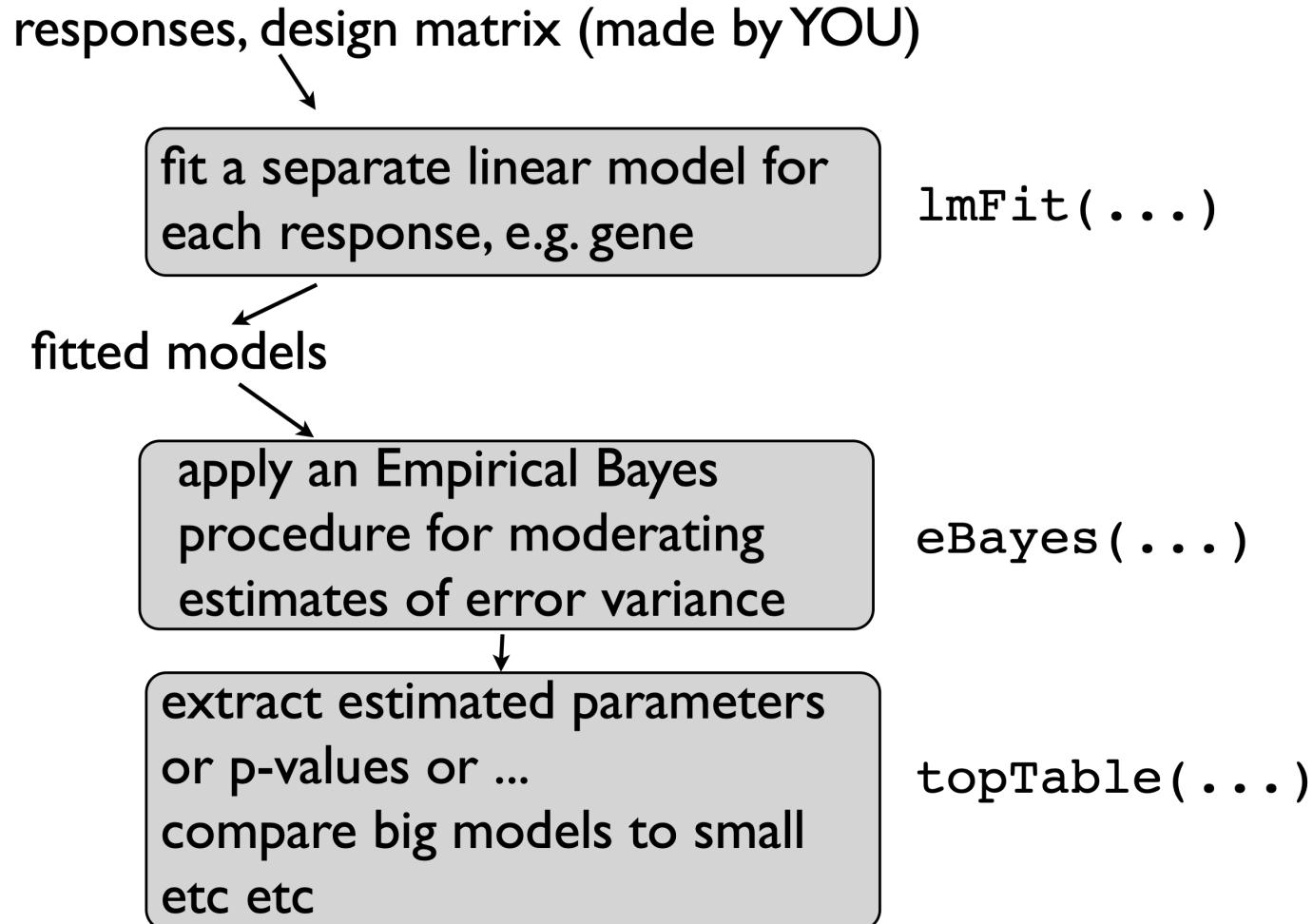
$$\tilde{s}_g^2 = \frac{d_0 s_0^2 + d s_g^2}{d_0 + d}$$

- recall that (s_0, d_0) are the *prior* parameters for σ_g^2 (random variable):

$$\frac{1}{\sigma_g^2} \sim \frac{1}{d_0 s_0^2} \chi_{d_0}^2$$

- the prior parameters incorporate information from all genes which allows us to **shrink/nudge the gene-specific variances toward a common consensus**
 - they are estimated from the data - the formulas for (s_0, d_0) and their derivations are beyond the scope of the course, but `limma` takes care of the details for us
 - note that (s_0, d_0) do not depend on g

Preview: limma workflow



Functions that make your life easier

Function	Description
<code>model.matrix</code>	Takes in your data frame and makes a design matrix
<code>limma::lmFit</code>	Fits the linear model to each gene separately - replace gene with "feature" depending on your data ('industrial scale' <code>lm</code>)
<code>limma::makeContrasts</code>	Create the contrast matrix C that you desire
<code>limma::contrast.fit</code>	Apply a contrast to your estimates
<code>limma::eBayes</code>	Use output of linear regression to compute moderated <i>t</i> statistics
<code>limma::topTable</code>	Query your results; sort your p-values; sort genes; Adjust for multiple comparisons
<code>limma::decideTests</code>	Identify which genes are significantly differentially expressed for each contrast

Getting help

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("limma")
```

[PDF](#) Limma One Page Introduction

[PDF](#) usersguide.pdf

[PDF](#) Reference Manual

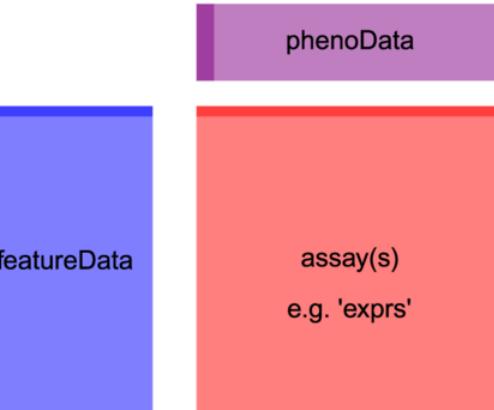
[Text](#) NEWS

- [Bioconductor homepage for limma](#)
- Bring up help pages for specific functions in RStudio, e.g. `?limma::topTable`

limma step one: lmFit

$$\mathbf{Y}_g = \mathbf{X}\boldsymbol{\alpha}_g + \boldsymbol{\varepsilon}_g,$$

- Within each gene observations are iid / constant variance
- `lmFit()` carries out multiple linear regression on each gene
- Usage: `lmFit(myDat, desMat)`
 - `myDat` is a `data.frame` or matrix with features in rows and samples in columns (G genes by N samples), or `ExpressionSet` object
 - `desMat` is a design matrix (output of `model.matrix(y ~ x)`; N samples by p parameters)



Bioconductor's ExpressionSet class

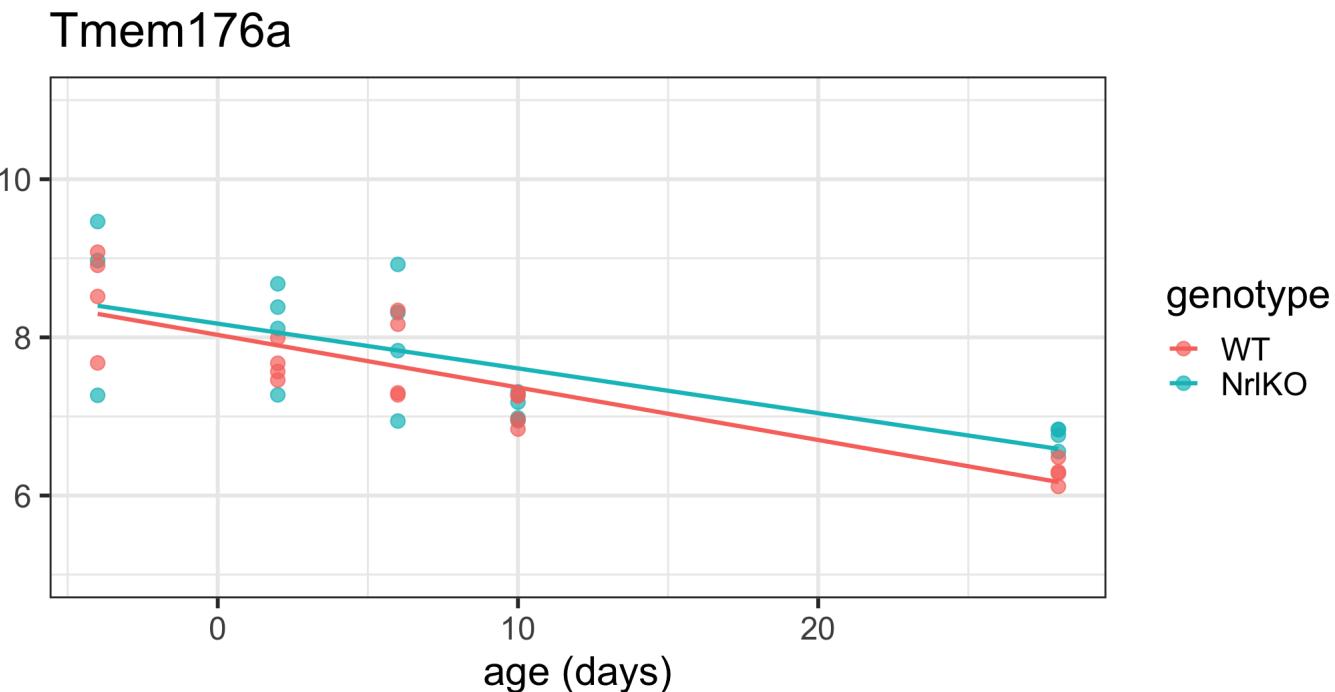
Let's run limma for the interactive model with age (continuous) and genotype (factor)

$$y_{ig} = \theta + \tau_{KO}x_{ig,KO} + \tau_{Age}x_{ig,Age} + \tau_{KO:Age}x_{ig,KO}x_{ig,Age}$$

- i indexes mouse, g indexes genes
- $x_{ig,KO}$ is the indicator variable for the NrlKO group
- $x_{ig,Age}$ is the continuous age variable

Interactive model with age and genotype

Example gene (but we want to fit this model on all genes):



Arranging input for lmFit: Bioconductor way

eset

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 45101 features, 39 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM92610 GSM92611 ... GSM92648 (39 total)
##   varLabels: sample_id dev_stage genotype age
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
##   pubMedIds: 16505381
## Annotation: GPL1261
```

Arranging input for lmFit: Separate expression & metadata

```
myDat <- assayData(eset)$exprs  
myDat[1:5,1:5]
```

```
##                 GSM92610  GSM92611  GSM92612  GSM92613  GSM92614  
## 1415670_at    7.108863  7.322392  7.420947  7.351444  7.240428  
## 1415671_at    9.714002  9.797742  9.831072  9.658442  9.708906  
## 1415672_at    9.429030  9.846977  10.003092 9.914112  10.174548  
## 1415673_at    8.426974  8.404206  8.594600  8.404206  8.835334  
## 1415674_a_at  8.498338  8.458287  8.426651  8.372776  8.541722
```

```
myMeta <- pData(eset)  
myMeta[1:5, ]
```

```
##           sample_id dev_stage genotype age  
## GSM92610  GSM92610        4W   NrlKO  28  
## GSM92611  GSM92611        4W   NrlKO  28  
## GSM92612  GSM92612        4W   NrlKO  28  
## GSM92613  GSM92613        4W   NrlKO  28  
## GSM92614  GSM92614       E16   NrlKO  -4
```

Formulating input for lmFit: Design Matrix

Bioconductor way:

```
desMat <- model.matrix(~ genotype*age,  
                      data = pData(eset))
```

Equivalently, if using the separate way:

```
desMat <- model.matrix(~ genotype*age,  
                      data = myMeta)
```

desMat

	(Intercept)	genotypeNr1K0	age	genotypeNr1K0:age
## GSM92610	1	1	28	28
## GSM92611	1	1	28	28
## GSM92612	1	1	28	28
## GSM92613	1	1	28	28
## GSM92614	1	1	-4	-4
## GSM92615	1	1	-4	-4
## GSM92616	1	1	-4	-4
## GSM92617	1	1	10	10
## GSM92618	1	1	10	10
## GSM92619	1	1	10	10
## GSM92620	1	1	10	10
## GSM92621	1	1	2	2
## GSM92622	1	1	2	2
## GSM92623	1	1	2	2
## GSM92624	1	1	2	2
## GSM92625	1	1	6	6
## GSM92626	1	1	6	6
## GSM92627	1	1	6	6
## GSM92628	1	1	6	6
## GSM92629	1	0	28	0
## GSM92630	1	0	28	0
## GSM92631	1	0	28	0
## GSM92632	1	0	28	0
## GSM92633	1	0	-4	0
## GSM92634	1	0	-4	0
## GSM92635	1	0	-4	0
## GSM92636	1	0	-4	0
## GSM92637	1	0	10	0
## GSM92638	1	0	10	0
## GSM92639	1	0	10	0

Computation is fast

```
system.time(gFit <- lmFit(eset,
                           model.matrix(~ genotype*age, data = pData(eset))))
```

```
##    user  system elapsed
##  0.115   0.026   0.141
```

- Equivalently, using the 'separate' way:

```
gFit <- lmFit(myDat,
               model.matrix(~ genotype*age, data = myMeta))
```

- Using `lmFit` to fit an interactive model on 45K probesets takes a fraction of a second

Computation is fast

```
system.time(gFit <- lmFit(eset,
                           model.matrix(~ genotype*age, data = pData(eset))))
```

```
##    user  system elapsed
##  0.115   0.026   0.141
```

- Equivalently, using the 'separate' way:

```
gFit <- lmFit(myDat,
               model.matrix(~ genotype*age, data = myMeta))
```

- Using `lmFit` to fit an interactive model on 45K probesets takes a fraction of a second
- The time-intensive parts of an analysis lie in selecting the model and covariates, choosing how to parameterize it, and interpreting the output

Output of lmFit

```
summary(gFit)
```

```
##                                     Length Class  Mode
## coefficients                  180404 -none- numeric
## rank                           1 -none- numeric
## assign                          4 -none- numeric
## qr                             5 qr    list
## df.residual                   45101 -none- numeric
## sigma                          45101 -none- numeric
## cov.coefficients               16 -none- numeric
## stdev.unscaled                 180404 -none- numeric
## pivot                           4 -none- numeric
## Amean                          45101 -none- numeric
## method                          1 -none- character
## design                         156 -none- numeric
```

```
nrow(eset)
```

```
## Features
##      45101
```

```
nrow(eset)*4
```

```
## Features
##      180404
```

Output of lmFit

```
summary(gFit)
```

```
##                                     Length Class  Mode
## coefficients                  180404 -none- numeric
## rank                           1 -none- numeric
## assign                          4 -none- numeric
## qr                             5 qr    list
## df.residual                   45101 -none- numeric
## sigma                          45101 -none- numeric
## cov.coefficients               16 -none- numeric
## stdev.unscaled                 180404 -none- numeric
## pivot                           4 -none- numeric
## Amean                          45101 -none- numeric
## method                          1 -none- character
## design                         156 -none- numeric
```

```
nrow(eset)
```

```
## Features
##      45101
```

```
nrow(eset)*4
```

```
## Features
##      180404
```

- OK... but where are the shrunken variable estimates?? How do I pull out p-values??
- Actually, we haven't carried out the empirical Bayesian computation yet -- need to run `eBayes()`!

Moderated t -tests using eBayes

```
summary(gFit)
```

```
##                                     Length Class  Mode
## coefficients                  180404 -none- numeric
## rank                           1 -none- numeric
## assign                          4 -none- numeric
## qr                             5 qr   list
## df.residual                   45101 -none- numeric
## sigma                          45101 -none- numeric
## cov.coefficients               16 -none- numeric
## stdev.unscaled                180404 -none- numeric
## pivot                           4 -none- numeric
## Amean                          45101 -none- numeric
## method                          1 -none- character
## design                         156 -none- numeric
```

```
summary(ebFit <- eBayes(gFit))
```

```
##                                     Length Class  Mode
## coefficients                  180404 -none- numeric
## rank                           1 -none- numeric
## assign                          4 -none- numeric
## qr                             5 qr   list
## df.residual                   45101 -none- numeric
## sigma                          45101 -none- numeric
## cov.coefficients               16 -none- numeric
## stdev.unscaled                180404 -none- numeric
## pivot                           4 -none- numeric
## Amean                          45101 -none- numeric
## method                          1 -none- character
## design                         156 -none- numeric
## df.prior                       1 -none- numeric
## s2.prior                        1 -none- numeric
## var.prior                      4 -none- numeric
## proportion                     1 -none- numeric
## s2.post                         45101 -none- numeric
## t                               180404 -none- numeric
## df.total                        45101 -none- numeric
## p.value                         180404 -none- numeric
## lods                            180404 -none- numeric
## F                               45101 -none- numeric
## F.p.value                       45101 -none- numeric
```

Components of the empirical Bayes estimators

math	plain english	limma	numerical result	also in <code>lm?</code>
s_g^2	gene-specific residual variance	<code>gFit\$sigma^2</code>	45K numbers	✓
d	residual degrees of freedom ($n - p$)	<code>gFit\$df.residual</code>	$39 - 4 = 35^*$	✓
s_0^2	mean of inverse χ^2 prior for s_g^2	<code>ebFit\$s2.prior</code>	0.048	
d_0	degrees of freedom for the prior	<code>ebFit\$df.prior</code>	2.61	
\tilde{s}_g^2	posterior mean of s_g^2 (i.e. moderated residual variance)	<code>ebFit\$s2.post</code>	45K numbers	

* limma can handle more complicated models where this is not the same for each gene, so this is actually a vector of 45K copies of the number 35

topTable() will help us extract relevant output in a convenient format!

```
topTable(fit, coef=NULL, number=10, genelist=fit$genes, adjust.method="BH",
         sort.by="B", resort.by=NULL, p.value=1, lfc=0, confint=FALSE)
```

fit

list containing a linear model fit produced by `lmFit`, `lm.series`, `gls.series` or `mrlm`. For `topTable`, `fit` should be an object of class `MArrayLM` as produced by `lmFit` and `eBayes`.

coef

column number or column name specifying which coefficient or contrast of the linear model is of interest. For `topTable`, can also be a vector of column subscripts, in which case the gene ranking is by F-statistic for that set of contrasts.

number

maximum number of genes to list

adjust.method

method used to adjust the p-values for multiple testing. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See `p.adjust` for the complete list of options. A `NULL` value will result in the default adjustment method, which is "BH".

sort.by

character string specifying statistic to rank genes by. Possible values for `topTable` and `toptable` are "logFC", "AveExpr", "t", "P", "p", "B" or "none". (Permitted synonyms are "M" for "logFC", "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P".) Possibilities for `topTableF` are "F" or "none". Possibilities for `topTreat` are as for `topTable` except for "B".

... (truncated - see `?topTable` for full listing)

Summary of topTable function

- `coef` is the argument where you specify the coefficient(s) you want to test for equality with zero (default is NULL; must be specified)
- `number` lets you control size of hit list (default is 10)
- `p.value` lets you specify a minimum adjusted p-value cutoff (default is 1)
- `lfc` lets you specify a minimum observed effect size - log2 fold change (default is 0)
- `sort.by` and `resort.by` give control over the ordering (default is by "B": log-odds that the gene is differentially expressed)
- `adjust.method` specifies how/if to adjust p-values for multiple testing (default is BH)

topTable in action: genotypeNr1KO

```
topTable(ebFit, coef = "genotypeNr1KO")
```

```
##          logFC AveExpr      t    P.Value adj.P.Val      B
## 1450946_at -4.7134891 8.733103 -15.583683 5.390931e-18 2.431364e-13 28.40938
## 1442070_at -0.9669400 8.268122 -9.787407 6.896220e-12 1.555132e-07 16.46086
## 1443457_at -1.1530057 6.049428 -9.091412 4.947029e-11 6.029428e-07 14.68843
## 1422679_s_at -2.1187394 9.495435 -9.002625 6.388954e-11 6.029428e-07 14.45703
## 1431708_a_at -2.1610625 8.591450 -8.967950 7.062007e-11 6.029428e-07 14.36634
## 1433050_at -1.5083887 8.468891 -8.923932 8.021234e-11 6.029428e-07 14.25096
## 1426288_at -4.2100657 9.323796 -8.772209 1.246491e-10 8.031144e-07 13.85106
## 1418108_at  0.8662091 8.260647  8.537984 2.475294e-10 1.395478e-06 13.22713
## 1450770_at  1.3103576 7.357033  8.128847 8.332369e-10 3.747976e-06 12.11870
## 1457802_at -0.8362030 7.778189 -8.117112 8.629984e-10 3.747976e-06 12.08657
```

- `topTable(ebFit, coef = 2)` is equivalent here, but much less informative!!
- What is the null hypothesis here?

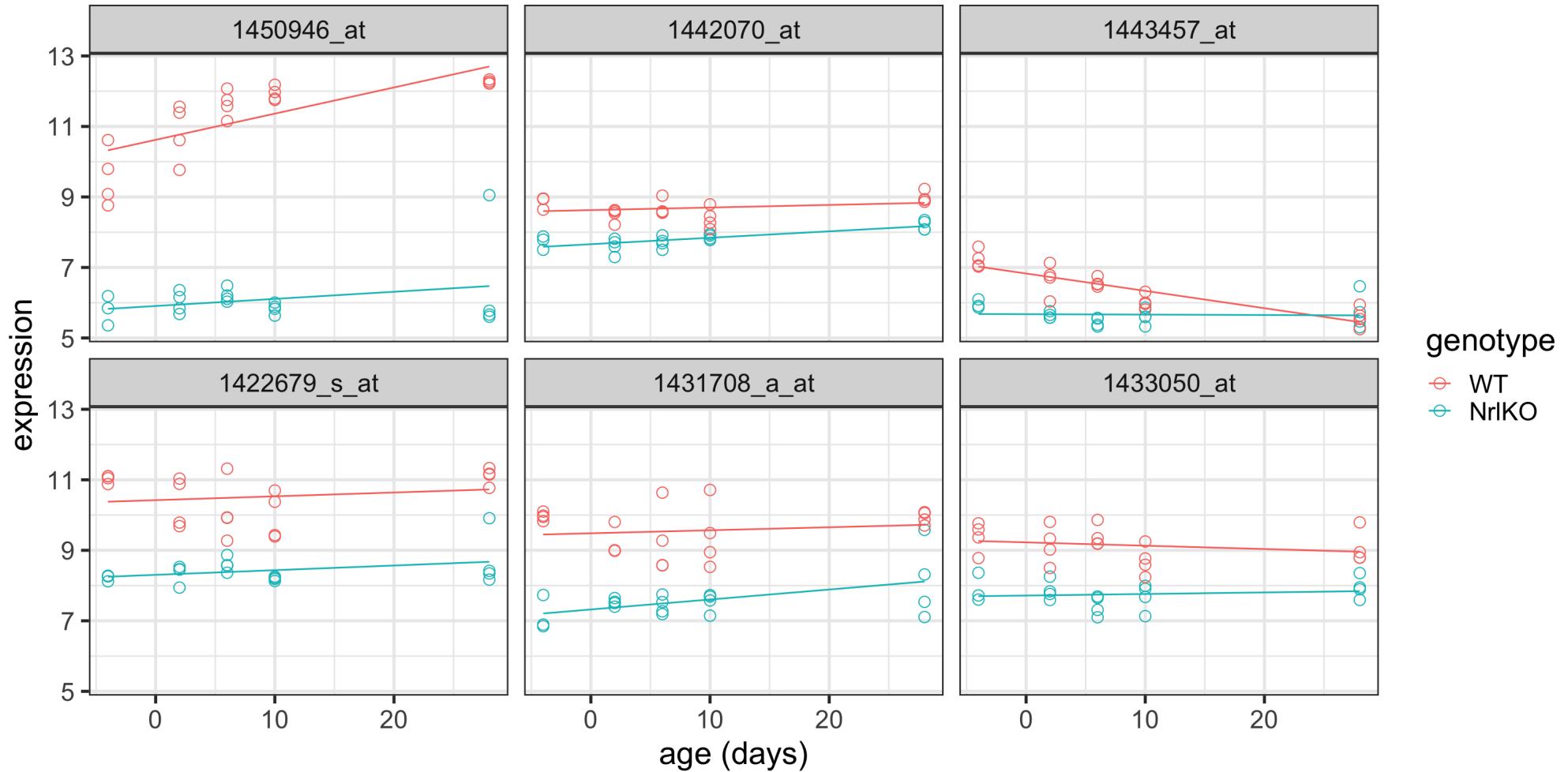
topTable in action: genotypeNr1KO

```
topTable(ebFit, coef = "genotypeNr1KO")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1450946_at	-4.7134891	8.733103	-15.583683	5.390931e-18	2.431364e-13	28.40938
## 1442070_at	-0.9669400	8.268122	-9.787407	6.896220e-12	1.555132e-07	16.46086
## 1443457_at	-1.1530057	6.049428	-9.091412	4.947029e-11	6.029428e-07	14.68843
## 1422679_s_at	-2.1187394	9.495435	-9.002625	6.388954e-11	6.029428e-07	14.45703
## 1431708_a_at	-2.1610625	8.591450	-8.967950	7.062007e-11	6.029428e-07	14.36634
## 1433050_at	-1.5083887	8.468891	-8.923932	8.021234e-11	6.029428e-07	14.25096
## 1426288_at	-4.2100657	9.323796	-8.772209	1.246491e-10	8.031144e-07	13.85106
## 1418108_at	0.8662091	8.260647	8.537984	2.475294e-10	1.395478e-06	13.22713
## 1450770_at	1.3103576	7.357033	8.128847	8.332369e-10	3.747976e-06	12.11870
## 1457802_at	-0.8362030	7.778189	-8.117112	8.629984e-10	3.747976e-06	12.08657

- `topTable(ebFit, coef = 2)` is equivalent here, but much less informative!!
- What is the null hypothesis here?
 - $H_0 : \tau_{KO} = 0$; this finds genes where the KO differs from WT *when age is zero*

Plotting the top 6 probes for genotypeNr1KO



topTable in action: age

```
topTable(ebFit, coef = "age")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1451635_at	0.19294061	7.791237	20.78935	3.204717e-22	1.445359e-17	40.60768
## 1425222_x_at	0.17604111	6.514475	19.72633	1.962954e-21	4.426561e-17	38.79502
## 1433699_at	0.14875031	6.939973	17.80467	6.442530e-20	7.957605e-16	35.29412
## 1452243_at	0.15889179	6.840975	17.75658	7.057586e-20	7.957605e-16	35.20253
## 1435436_at	0.13067119	8.274676	17.61808	9.187120e-20	8.286966e-16	34.93763
## 1458418_at	0.21277027	6.270187	16.43686	9.326171e-19	7.006477e-15	32.60718
## 1424977_at	0.07814988	6.505464	16.36079	1.087456e-18	7.006477e-15	32.45260
## 1431174_at	0.16158579	7.418000	16.01901	2.183055e-18	1.230725e-14	31.75106
## 1421818_at	0.11250919	7.982278	15.66666	4.531133e-18	2.270651e-14	31.01562
## 1419069_at	0.09325234	8.222096	15.54775	5.813459e-18	2.621928e-14	30.76457

- `topTable(ebFit, coef = 3)` is equivalent here, but much less informative!!
- What is the null hypothesis here?

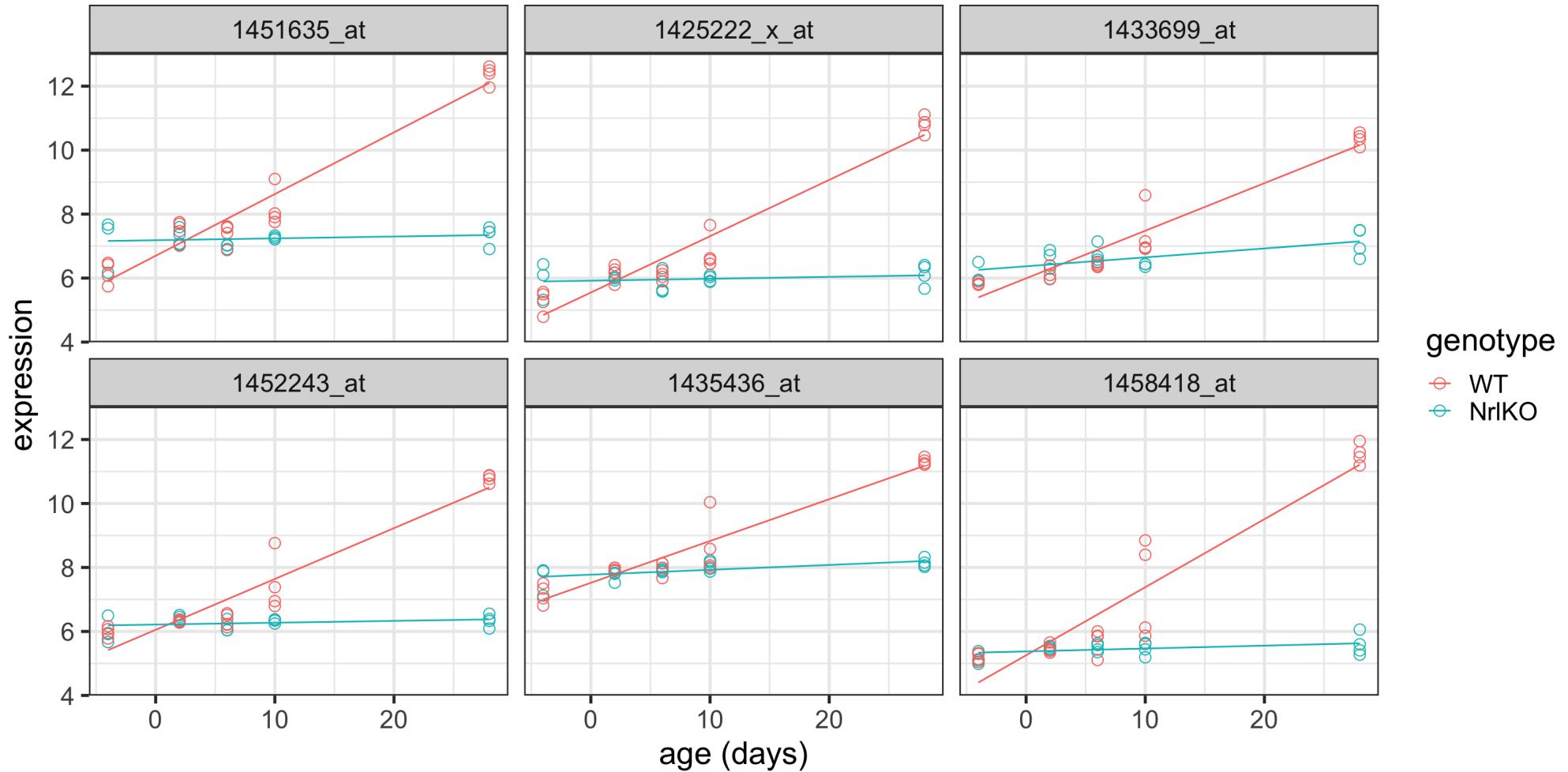
topTable in action: age

```
topTable(ebFit, coef = "age")
```

```
##                logFC AveExpr      t    P.Value adj.P.Val      B
## 1451635_at    0.19294061 7.791237 20.78935 3.204717e-22 1.445359e-17 40.60768
## 1425222_x_at  0.17604111 6.514475 19.72633 1.962954e-21 4.426561e-17 38.79502
## 1433699_at    0.14875031 6.939973 17.80467 6.442530e-20 7.957605e-16 35.29412
## 1452243_at    0.15889179 6.840975 17.75658 7.057586e-20 7.957605e-16 35.20253
## 1435436_at    0.13067119 8.274676 17.61808 9.187120e-20 8.286966e-16 34.93763
## 1458418_at    0.21277027 6.270187 16.43686 9.326171e-19 7.006477e-15 32.60718
## 1424977_at    0.07814988 6.505464 16.36079 1.087456e-18 7.006477e-15 32.45260
## 1431174_at    0.16158579 7.418000 16.01901 2.183055e-18 1.230725e-14 31.75106
## 1421818_at    0.11250919 7.982278 15.66666 4.531133e-18 2.270651e-14 31.01562
## 1419069_at    0.09325234 8.222096 15.54775 5.813459e-18 2.621928e-14 30.76457
```

- `topTable(ebFit, coef = 3)` is equivalent here, but much less informative!!
- What is the null hypothesis here?
 - $H_0 : \tau_{Age} = 0$; this finds genes where age significantly affects gene expression *for WT*

Plotting the top 6 genes for age



topTable in action: genotypeNr1K0:age

```
topTable(ebFit, coef = "genotypeNr1K0:age")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1451635_at	-0.1871800	7.791237	-14.00463	1.684302e-16	7.596369e-12	27.52010
## 1425222_x_at	-0.1700908	6.514475	-13.23451	9.961393e-16	2.246344e-11	25.74877
## 1441809_at	0.1336356	6.649440	12.55702	5.037221e-15	7.572790e-11	24.13110
## 1452243_at	-0.1531326	6.840975	-11.88284	2.670693e-14	3.011273e-10	22.46414
## 1431174_at	-0.1655894	7.418000	-11.39882	9.161231e-14	8.263614e-10	21.23123
## 1458418_at	-0.2037314	6.270187	-10.92850	3.122990e-13	2.347500e-09	20.00384
## 1435436_at	-0.1154500	8.274676	-10.80855	4.289363e-13	2.763637e-09	19.68613
## 1446715_at	-0.1409774	5.613122	-10.57909	7.912260e-13	4.325169e-09	19.07307
## 1442190_at	-0.1323038	5.896077	-10.54672	8.630965e-13	4.325169e-09	18.98601
## 1416306_at	0.1702059	6.560050	10.41285	1.238250e-12	5.584630e-09	18.62456

- `topTable(ebFit, coef = 4)` is equivalent here, but much less informative!!
- What is the null hypothesis here?

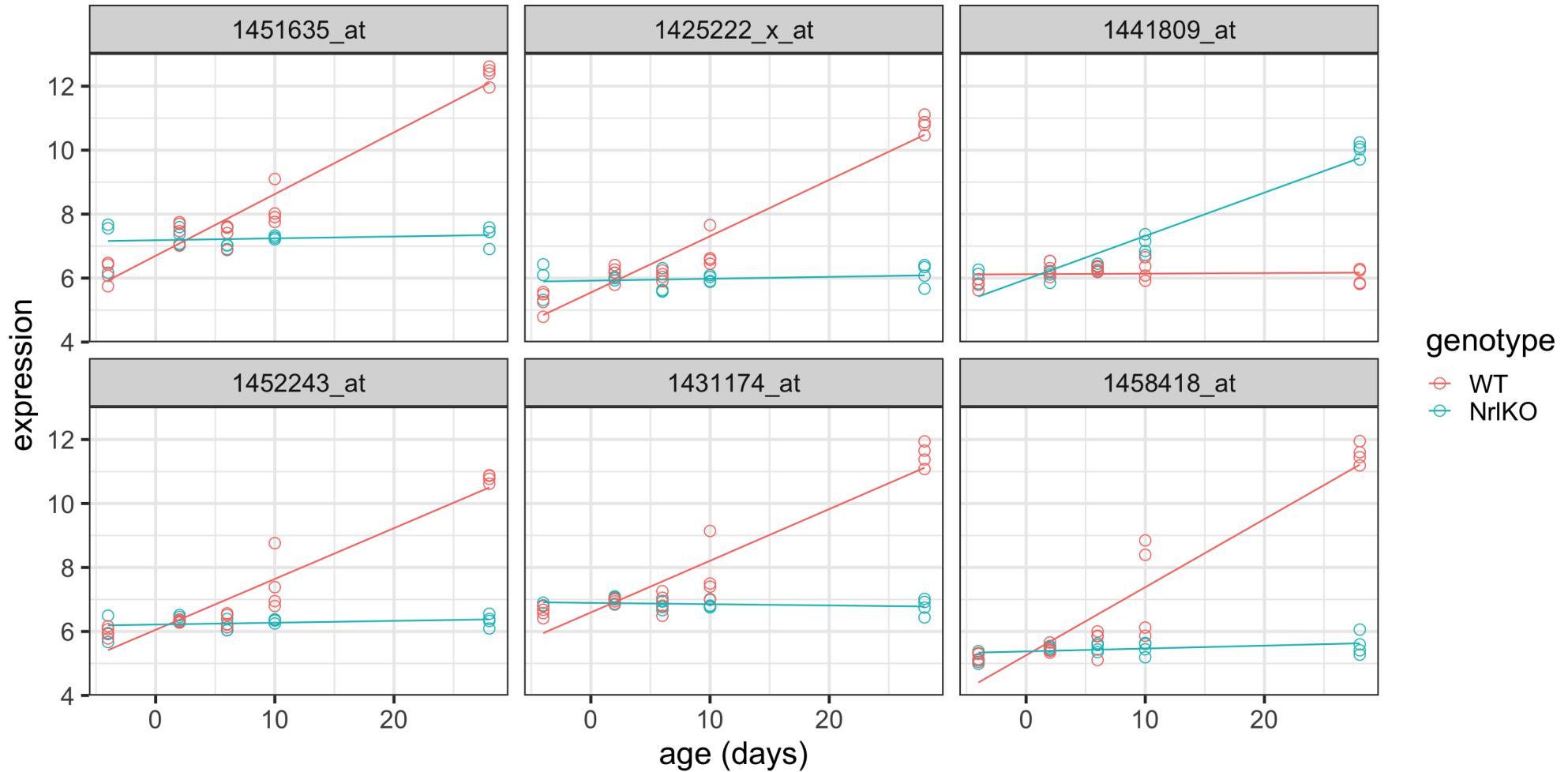
topTable in action: genotypeNr1KO:age

```
topTable(ebFit, coef = "genotypeNr1KO:age")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1451635_at	-0.1871800	7.791237	-14.00463	1.684302e-16	7.596369e-12	27.52010
## 1425222_x_at	-0.1700908	6.514475	-13.23451	9.961393e-16	2.246344e-11	25.74877
## 1441809_at	0.1336356	6.649440	12.55702	5.037221e-15	7.572790e-11	24.13110
## 1452243_at	-0.1531326	6.840975	-11.88284	2.670693e-14	3.011273e-10	22.46414
## 1431174_at	-0.1655894	7.418000	-11.39882	9.161231e-14	8.263614e-10	21.23123
## 1458418_at	-0.2037314	6.270187	-10.92850	3.122990e-13	2.347500e-09	20.00384
## 1435436_at	-0.1154500	8.274676	-10.80855	4.289363e-13	2.763637e-09	19.68613
## 1446715_at	-0.1409774	5.613122	-10.57909	7.912260e-13	4.325169e-09	19.07307
## 1442190_at	-0.1323038	5.896077	-10.54672	8.630965e-13	4.325169e-09	18.98601
## 1416306_at	0.1702059	6.560050	10.41285	1.238250e-12	5.584630e-09	18.62456

- `topTable(ebFit, coef = 4)` is equivalent here, but much less informative!!
- What is the null hypothesis here?
 - $H_0 : \tau_{KO:Age} = 0$; this finds genes where the effect of age is significantly different in each genotype

Plotting the top 6 genes for genotypeNr1KO:age



topTable in action: any effect of genotype

```
topTable(ebFit, coef = c("genotypeNrlKO", "genotypeNrlKO:age")) %>% as_tibble()
```

```
## # A tibble: 10 × 6
##   genotypeNrlKO genotypeNrlKO.age AveExpr      F  P.Value adj.P.Val
##   <dbl>           <dbl>    <dbl> <dbl>    <dbl>       <dbl>
## 1 -4.71            -0.0541     8.73 247. 2.40e-22  1.08e-17
## 2  0.487           -0.187      7.79 130. 1.31e-17  2.18e-13
## 3 -0.989           -0.223      7.42 129. 1.45e-17  2.18e-13
## 4  0.370           -0.170      6.51 120. 4.72e-17  5.32e-13
## 5 -0.158            0.134      6.65 118. 6.59e-17  5.95e-13
## 6  0.428            0.170      6.56 113. 1.21e-16  9.11e-13
## 7  0.657            0.0617     5.86 111. 1.62e-16  1.00e-12
## 8  0.994            0.0423     7.37 111. 1.78e-16  1.00e-12
## 9  0.164            -0.153      6.84 106. 3.34e-16  1.68e-12
## 10 -0.682           -0.140      8.16 103. 5.77e-16  2.44e-12
```

- `topTable(ebFit, coef = c(2,4))` is equivalent here, but much less informative!!
- What is the null hypothesis here?

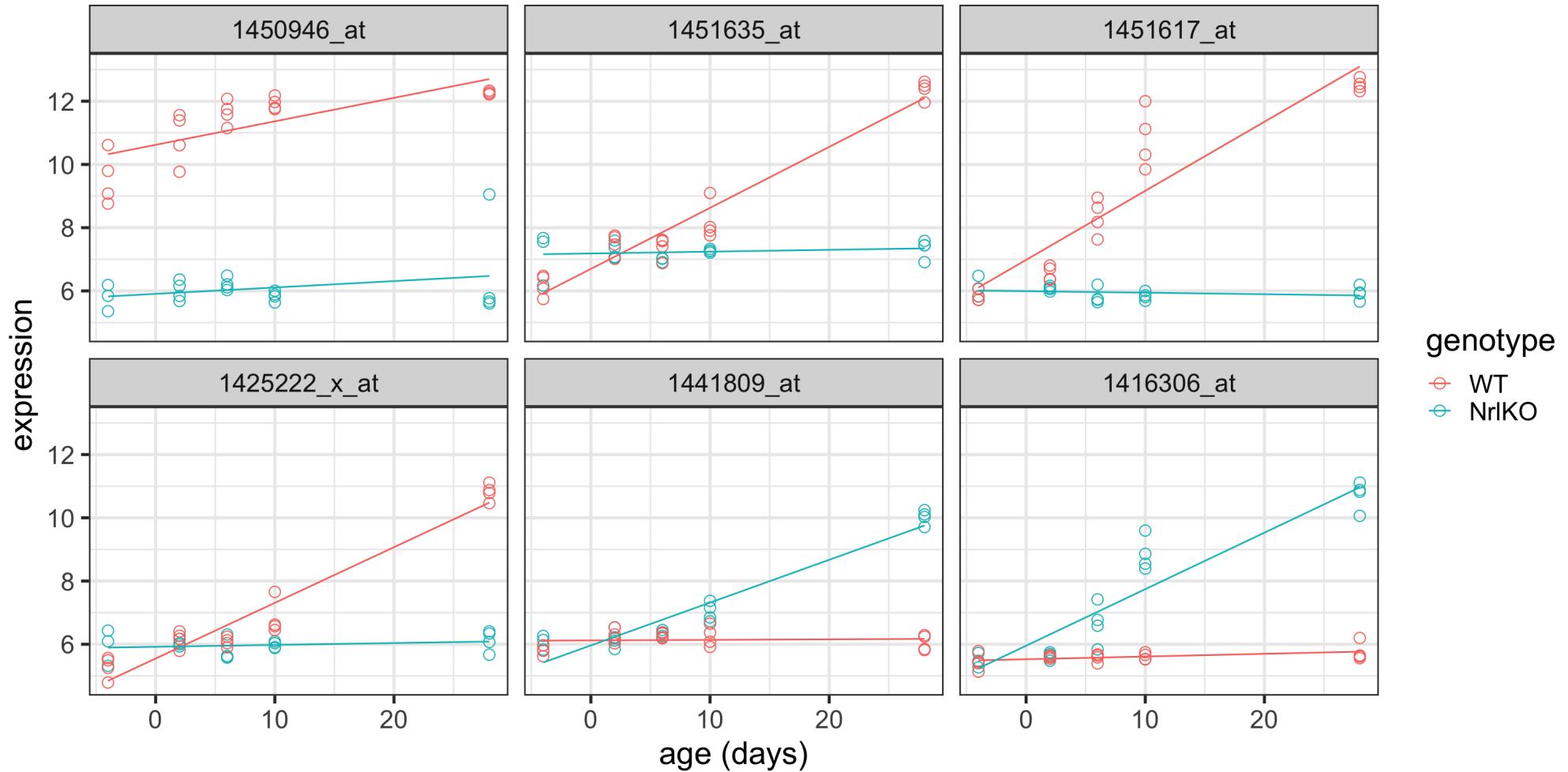
topTable in action: any effect of genotype

```
topTable(ebFit, coef = c("genotypeNrlKO", "genotypeNrlKO:age")) %>% as_tibble()
```

```
## # A tibble: 10 × 6
##   genotypeNrlKO genotypeNrlKO.age AveExpr      F  P.Value adj.P.Val
##   <dbl>           <dbl>    <dbl> <dbl>    <dbl>       <dbl>
## 1 -4.71            -0.0541     8.73 247. 2.40e-22  1.08e-17
## 2  0.487           -0.187      7.79 130. 1.31e-17  2.18e-13
## 3 -0.989           -0.223      7.42 129. 1.45e-17  2.18e-13
## 4  0.370           -0.170      6.51 120. 4.72e-17  5.32e-13
## 5 -0.158            0.134      6.65 118. 6.59e-17  5.95e-13
## 6  0.428            0.170      6.56 113. 1.21e-16  9.11e-13
## 7  0.657            0.0617     5.86 111. 1.62e-16  1.00e-12
## 8  0.994            0.0423     7.37 111. 1.78e-16  1.00e-12
## 9  0.164            -0.153      6.84 106. 3.34e-16  1.68e-12
## 10 -0.682           -0.140      8.16 103. 5.77e-16  2.44e-12
```

- `topTable(ebFit, coef = c(2,4))` is equivalent here, but much less informative!!
- What is the null hypothesis here?
 - $H_0 : \tau_{KO} = \tau_{KO:Age} = 0$; this finds genes where any (additive and/or interaction) effect of genotype is significant

Plotting the top 6 genes for any effect of genotype

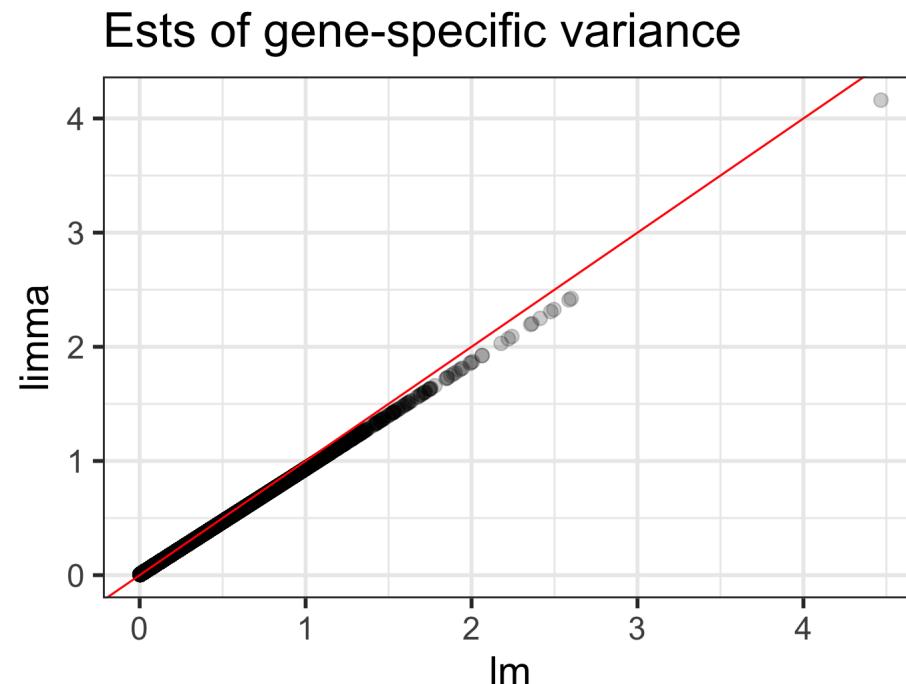


Comparison of s_g^2 and \tilde{s}_g^2 (shrinkage!)

Fill in the blank:

For **large** variances,
limma _____
the estimates.

- (a) INCREASES
- (b) DECREASES

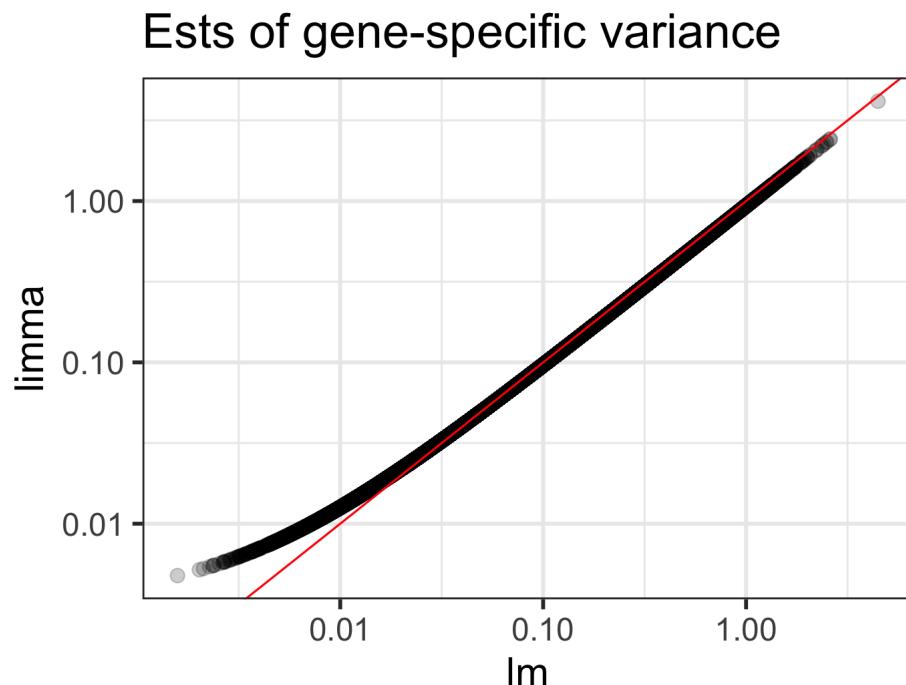


Comparison of s_g^2 and \tilde{s}_g^2 (shrinkage!)

Fill in the blank:

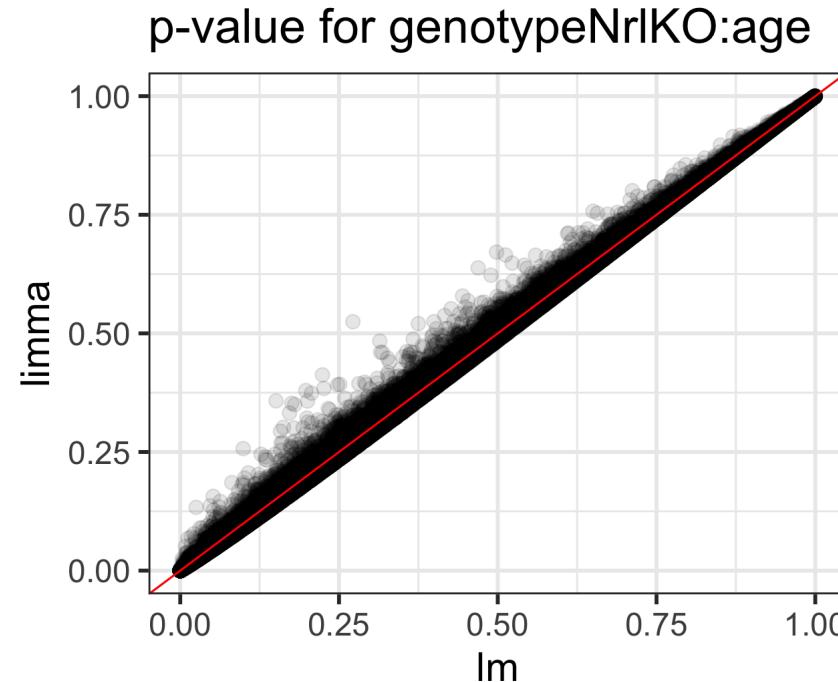
For **small** variances,
limma _____
the estimates.

- (a) INCREASES
- (b) DECREASES



This plot is on the log-scale to 'zoom in' on the low range

Comparison of interaction coefficient p-values

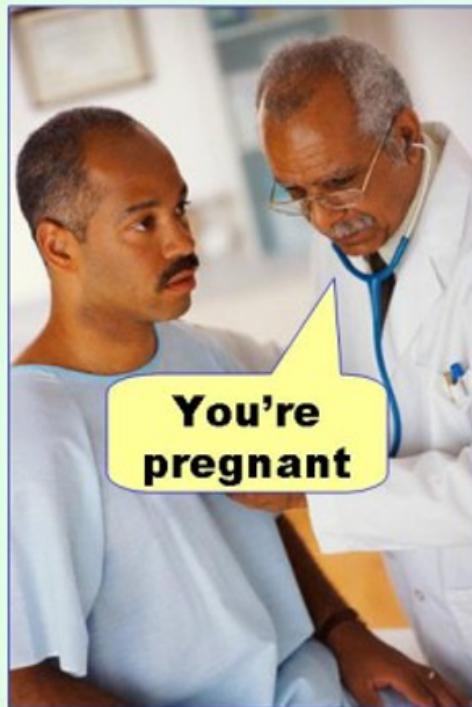


- 17479 genes where limma p-value is *larger* than lm
- 27622 genes where limma p-value is *smaller* than lm

Multiple testing

Recall the two types of errors

Type I error
(false positive)



Type II error
(false negative)



Error rates

		Actual Situation “Truth”	
		H_0 True	H_0 False
Decision	Do Not Reject H_0	Correct Decision $1-\alpha$	Incorrect Decision Type II Error β
	Reject H_0	Incorrect Decision Type I Error α	Correct Decision $1-\beta$

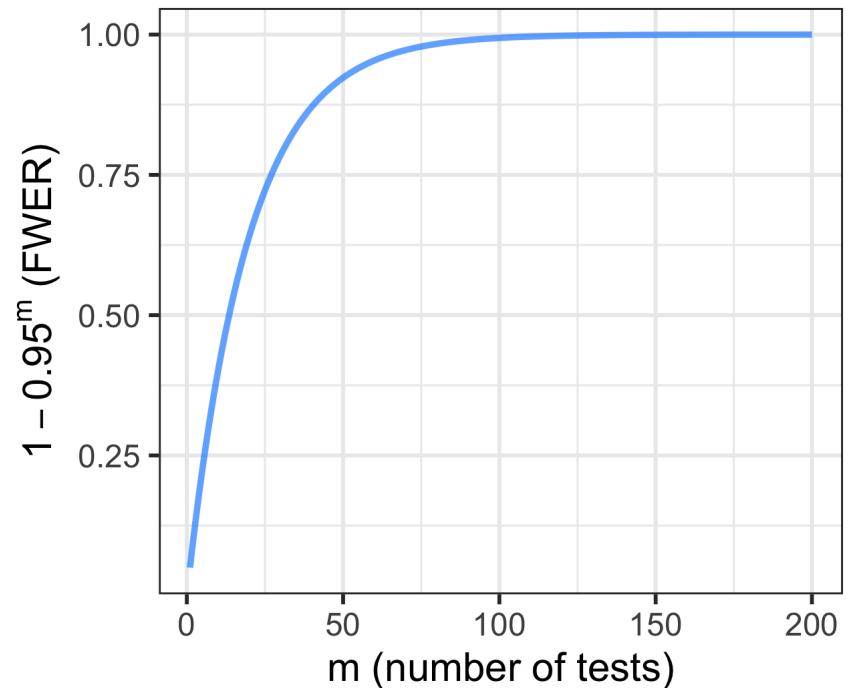
$$\alpha = P(\text{Type I Error}), \beta = P(\text{Type II Error}), \text{Power} = 1 - \beta$$

Type I Error rate for m tests

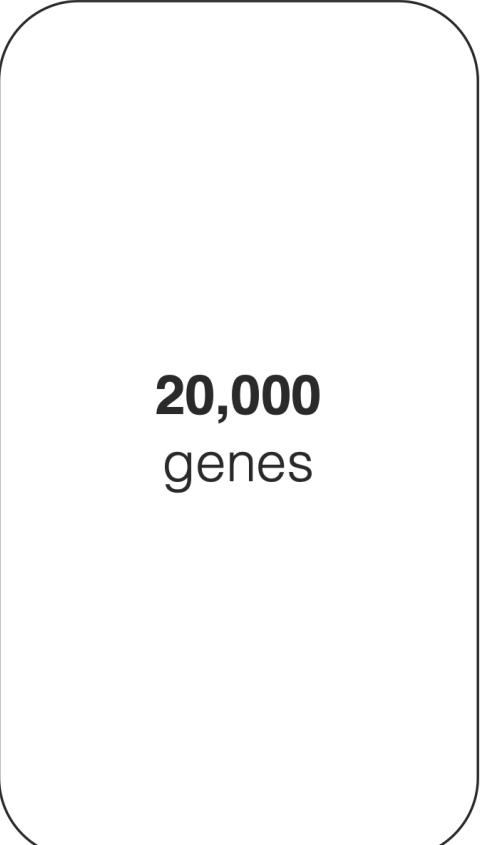
- $P(\text{incorrect decision} | H_0) = \alpha$
 - let $\alpha = 0.05$
- $P(\text{correct decision} | H_0) = 1 - \alpha = 0.95$

Type I Error rate for m tests

- $P(\text{incorrect decision} | H_0) = \alpha$
 - let $\alpha = 0.05$
- $P(\text{correct decision} | H_0) = 1 - \alpha = 0.95$
- $P(\text{correct decision on } m \text{ tests} | H_0) =$
$$(1 - \alpha)^m = 0.95^m$$
- $P(\text{at least one error on } m \text{ tests} | H_0) =$
$$1 - (1 - \alpha)^m =$$
$$1 - 0.95^m = \alpha_{FWER}$$

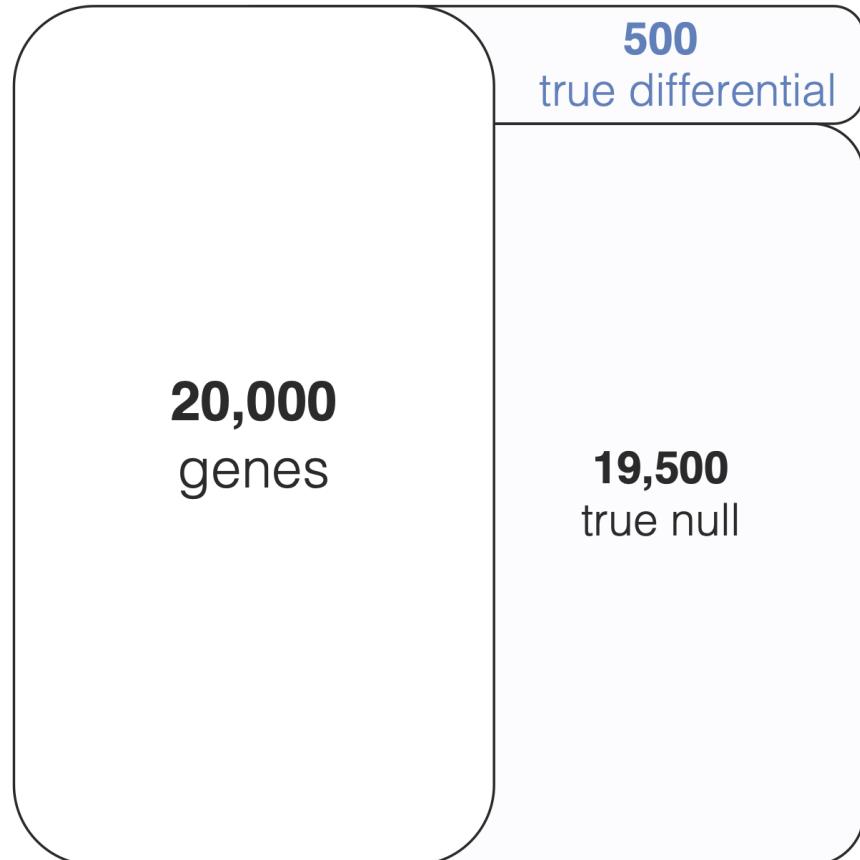


Multiple comparisons in genomics

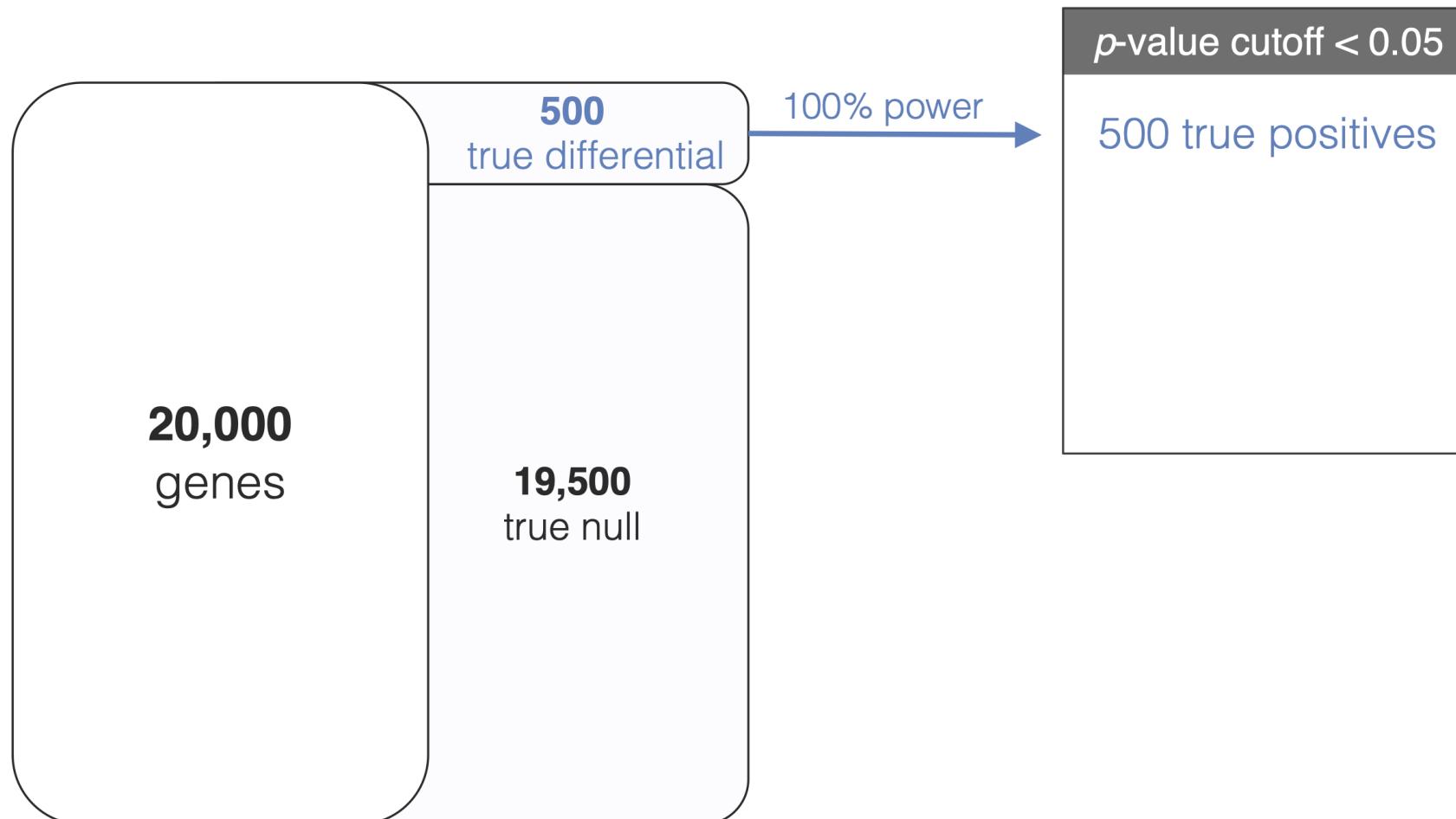


20,000
genes

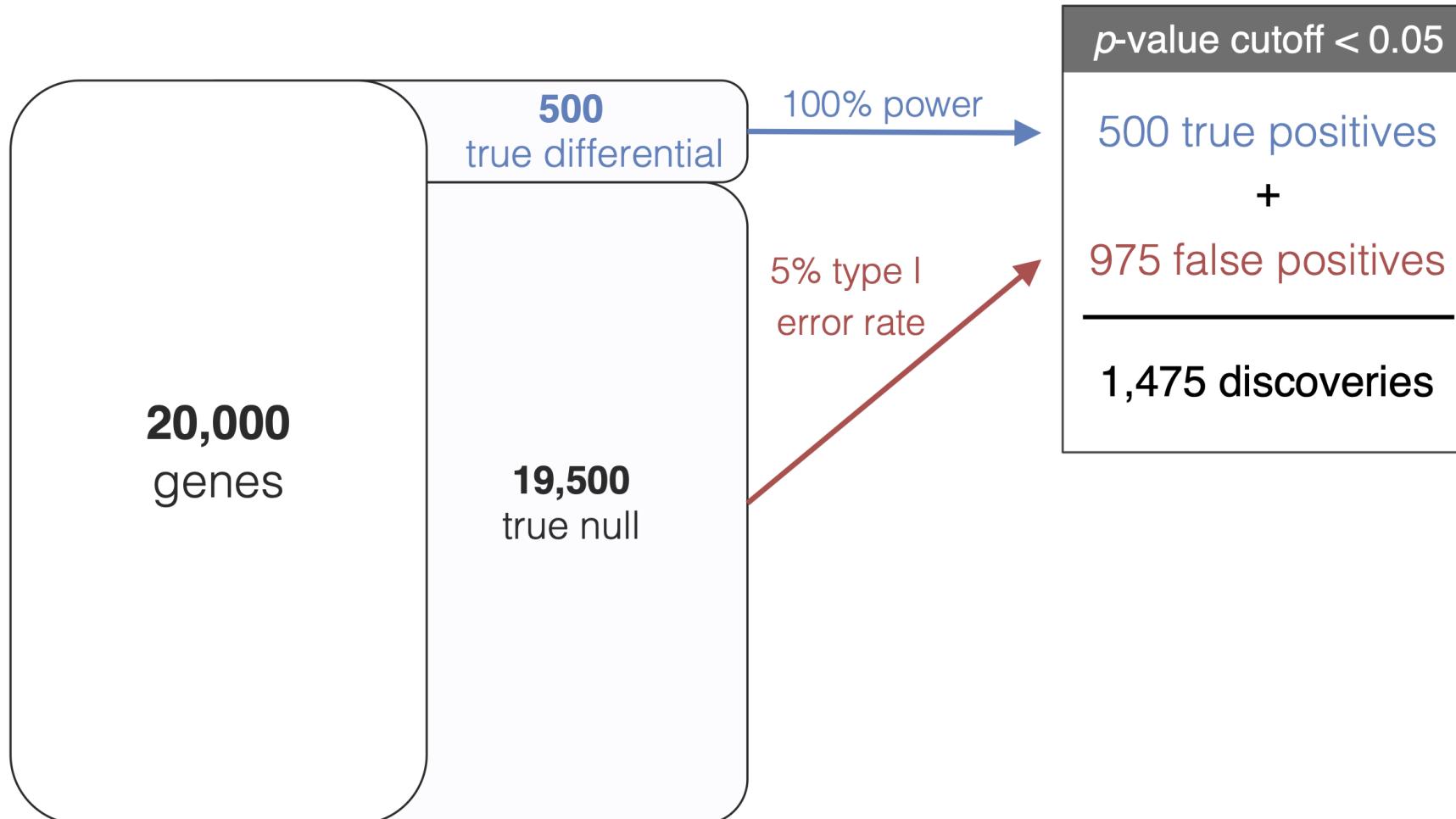
Multiple comparisons in genomics



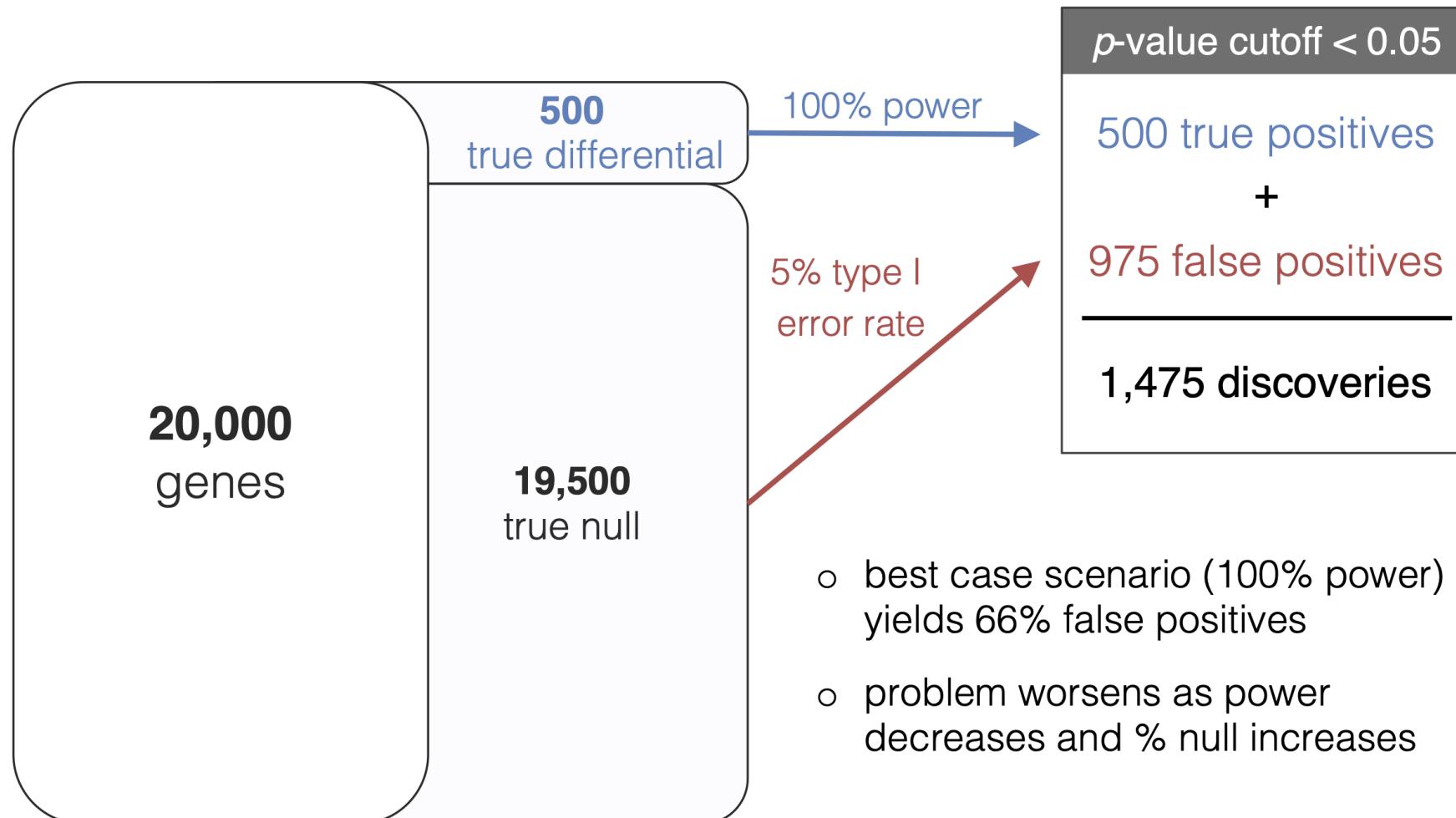
Multiple comparisons in genomics



Multiple comparisons in genomics



Multiple comparisons in genomics



- best case scenario (100% power) yields 66% false positives
- problem worsens as power decreases and % null increases

Family-Wise Error Rate (FWER)

- FWER is the probability of making at least one error when testing m tests

Family-Wise Error Rate (FWER)

- FWER is the probability of making at least one error when testing m tests
- Control the FWER: limit the probability of making at least one incorrect decision
- One example: the Bonferroni correction for $\alpha = 0.05$:

If $P(\text{at least one error on } m \text{ tests}) < \alpha$

$$\Rightarrow P(\text{at least one error on } m \text{ tests}) < \sum_{i=1}^m P(\text{error on test } i)$$

$$\sum_{i=1}^m P(\text{error on test } i) = m\alpha_{Bon}$$

$$\alpha_{Bon} = \frac{\alpha}{m} = \frac{0.05}{m}$$

Bonferroni correction: controlling the FWER

Can think of controlling the probability of at least one false positive in two ways:

1. Adjust the p-values; keep same α :

$$p_{Bon,i} = mp_i$$

(to be more precise: $p_{Bon,i} = \min(mp_i, 1)$)

Then, threshold $p_{Bon,i}$ at α

2. Adjust the α threshold; keep same p-values:

$$\alpha_{Bon} = \frac{\alpha}{m}$$

Then, threshold p_i at α_{Bon}

Can we do better?

- Bonferroni correction is very **conservative** (i.e. controls the FWER even lower than α in many settings)
- Several other options are better
- For example, the Holm procedure: multiplier for p-value correction is not the same for all genes; more powerful

$$p_{Holm,1} = mp_1$$

$$p_{Holm,2} = (m - 1)p_2$$

$$p_{Holm,3} = (m - 2)p_3$$

⋮

$$\Rightarrow FWER \leq \alpha$$

How practical is the FWER in high-throughput biology?

- Why do we care so much about making one single error??

How practical is the FWER in high-throughput biology?

- Why do we care so much about making one single error??
- One easy way to ensure no Type I errors: reject no hypotheses! 😊
 - However, then our power is zero... 😭

How practical is the FWER in high-throughput biology?

- Why do we care so much about making one single error??
- One easy way to ensure no Type I errors: reject no hypotheses! 😊
 - However, then our power is zero... 😭

Being overly strict about Type I error leads to greater Type II error (loss of power)

Radical idea: it's OK to make multiple errors, as long as you also have some true positives!

Enter: the False Discovery Rate (FDR)

J. R. Statist. Soc. B (1995)
57, No. 1, pp. 289-300

Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing

By YOAV BENJAMINI† and YOSEF HOCHBERG

Tel Aviv University, Israel

[Received January 1993. Revised March 1994]

SUMMARY

The common approach to the multiplicity problem calls for controlling the familywise error rate (FWER). This approach, though, has faults, and we point out a few. A different approach to problems of multiple significance testing is presented. It calls for controlling the expected proportion of falsely rejected hypotheses – the false discovery rate. This error rate is equivalent to the FWER when all hypotheses are true but is smaller otherwise. Therefore, in problems where the control of the false discovery rate rather than that of the FWER is desired, there is potential for a gain in power. A simple sequential Bonferroni-type procedure is proved to control the false discovery rate for independent test statistics, and a simulation study shows that the gain in power is substantial. The use of the new procedure and the appropriateness of the criterion are illustrated with examples.

Benjamini Y, Hochberg Y.
"Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society: Series B (Methodological)*. 1995 Jan;57(1): 289-300.

Over 80K citations!!

False Discovery Rate

	Null True	Alternative True	Total
Not Called Significant	U	T	$m - R$
Called Significant	V	S	R
	m_0	$m - m_0$	m

V = # Type I errors [false positives]

FDR is designed to control the expected proportion of false positives (V) among all hypotheses where the null has been rejected (R)

False Discovery Rate

	Null True	Alternative True	Total
Not Called Significant	U	T	$m - R$
Called Significant	V	S	R
	m_0	$m - m_0$	m

V = # Type I errors [false positives]

FDR is designed to control the expected proportion of false positives (V) among all hypotheses where the null has been rejected (R)

$$FDR = E\left[\frac{V}{R}\right]$$

FDR vs FPR vs FWER

- **False Discovery Rate (FDR)** is the rate that significant features (R) are truly null

$$FDR = E\left[\frac{V}{R}\right]$$

- **False Positive Rate (FPR)** is the rate that truly null features (m_0) are called significant

$$FPR = E\left[\frac{V}{m_0}\right]$$

- **Family-Wise Error Rate (FWER)** is the probability that the number of truly null features rejected (V) is at least 1

$$\text{FWER} = P(V \geq 1)$$

Benjamini-Hochberg FDR (BH procedure)

- Proposed the idea of controlling FDR instead of FWER
- Proposed a procedure for doing so
 - note that we know R , but we don't know V
- Procedure: control FDR at level q
 1. order the raw p-values $p_1 \leq p_2 \leq \dots \leq p_m$
 2. find test with highest rank j such that $p_j < \frac{jq}{m}$
 3. declare all smaller ranks up to j significant

Controlling FDR at level $q = 0.05$

Rank (j)	P-value
1	0.0008
2	0.009
3	0.127
4	0.205
5	0.396
6	0.450
7	0.641
8	0.781
9	0.900
10	0.993

Controlling FDR at level $q = 0.05$

Rank (j)	P-value	(j/m)*q
1	0.0008	0.005
2	0.009	0.010
3	0.127	0.015
4	0.205	0.020
5	0.396	0.025
6	0.450	0.030
7	0.641	0.035
8	0.781	0.040
9	0.900	0.045
10	0.993	0.050

Controlling FDR at level $q = 0.05$

Rank (j)	P-value	$(j/m)*q$	Reject H_0 ?
1	0.0008	0.005	✓
2	0.009	0.010	✓
3	0.127	0.015	
4	0.205	0.020	
5	0.396	0.025	
6	0.450	0.030	
7	0.641	0.035	
8	0.781	0.040	
9	0.900	0.045	
10	0.993	0.050	

Controlling FDR at level $q = 0.05$

Rank (j)	P-value	$(j/m)*q$	Reject H_0 ?	$FWER_{Bon} < 0.05$?
1	0.0008	0.005	✓	✓
2	0.009	0.010	✓	
3	0.127	0.015		
4	0.205	0.020		
5	0.396	0.025		
6	0.450	0.030		
7	0.641	0.035		
8	0.781	0.040		
9	0.900	0.045		
10	0.993	0.050		

Where $\alpha_{Bon} = 0.05/10 = 0.005$

BH FDR values given in limma by default

```
topTable(ebFit, coef = "genotypeNrlKO")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1450946_at	-4.7134891	8.733103	-15.583683	5.390931e-18	2.431364e-13	28.40938
## 1442070_at	-0.9669400	8.268122	-9.787407	6.896220e-12	1.555132e-07	16.46086
## 1443457_at	-1.1530057	6.049428	-9.091412	4.947029e-11	6.029428e-07	14.68843
## 1422679_s_at	-2.1187394	9.495435	-9.002625	6.388954e-11	6.029428e-07	14.45703
## 1431708_a_at	-2.1610625	8.591450	-8.967950	7.062007e-11	6.029428e-07	14.36634
## 1433050_at	-1.5083887	8.468891	-8.923932	8.021234e-11	6.029428e-07	14.25096
## 1426288_at	-4.2100657	9.323796	-8.772209	1.246491e-10	8.031144e-07	13.85106
## 1418108_at	0.8662091	8.260647	8.537984	2.475294e-10	1.395478e-06	13.22713
## 1450770_at	1.3103576	7.357033	8.128847	8.332369e-10	3.747976e-06	12.11870
## 1457802_at	-0.8362030	7.778189	-8.117112	8.629984e-10	3.747976e-06	12.08657

BH FDR values given in limma by default

```
topTable(ebFit, coef = "genotypeNrlKO")
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
## 1450946_at	-4.7134891	8.733103	-15.583683	5.390931e-18	2.431364e-13	28.40938
## 1442070_at	-0.9669400	8.268122	-9.787407	6.896220e-12	1.555132e-07	16.46086
## 1443457_at	-1.1530057	6.049428	-9.091412	4.947029e-11	6.029428e-07	14.68843
## 1422679_s_at	-2.1187394	9.495435	-9.002625	6.388954e-11	6.029428e-07	14.45703
## 1431708_a_at	-2.1610625	8.591450	-8.967950	7.062007e-11	6.029428e-07	14.36634
## 1433050_at	-1.5083887	8.468891	-8.923932	8.021234e-11	6.029428e-07	14.25096
## 1426288_at	-4.2100657	9.323796	-8.772209	1.246491e-10	8.031144e-07	13.85106
## 1418108_at	0.8662091	8.260647	8.537984	2.475294e-10	1.395478e-06	13.22713
## 1450770_at	1.3103576	7.357033	8.128847	8.332369e-10	3.747976e-06	12.11870
## 1457802_at	-0.8362030	7.778189	-8.117112	8.629984e-10	3.747976e-06	12.08657

Or, obtain them yourself for any vector of p-values `p` with `p.adjust(p, method="BH")`

Other ways to control FDR

- BH is just one (the first) method to control FDR
- Since the publication of the BH method, other methods have been proposed
- One of the most popular is Storey's q-value

The image shows a thumbnail of a scientific article from the Proceedings of the National Academy of Sciences (PNAS). The title of the article is "Statistical significance for genomewide studies". The authors listed are John D. Storey*† and Robert Tibshirani‡. The article was edited by Philip P. Green and approved on May 30, 2003. The abstract discusses the challenges of analyzing large data sets in genomics and introduces the concept of the q-value as a measure of statistical significance that accounts for the false discovery rate. The text is in a standard academic font, and the background of the thumbnail is a dark red color.

Statistical significance for genomewide studies

John D. Storey*† and Robert Tibshirani‡

*Department of Biostatistics, University of Washington, Seattle, WA 98195; and †Departments of Health Research and Policy and Statistics, Stanford University, Stanford, CA 94305

Edited by Philip P. Green, University of Washington School of Medicine, Seattle, WA, and approved May 30, 2003 (received for review January 28, 2003)

With the increase in genomewide experiments and the sequencing of multiple genomes, the analysis of large data sets has become commonplace in biology. It is often the case that thousands of features in a genomewide data set are tested against some null hypothesis, where a number of features are expected to be significant. Here we propose an approach to measuring statistical significance in these genomewide studies based on the concept of the false discovery rate. This approach offers a sensible balance between the number of true and false positives that is automatically calibrated and easily interpreted. In doing so, a measure of statistical significance called the *q* value is associated with each tested feature. The *q* value is similar to the well known *p* value, except it is a measure of significance in terms of the false discovery rate rather than the false positive rate. Our approach avoids a flood of false positive results, while offering a more liberal criterion than what has been used in genome scans for linkage.

false discovery rates | genomics | multiple hypothesis testing | *q* values

- `qvalue` package implementation: provides adjusted p-values

Storey's q-value vs BH (Conceptual)

- Just like BH, is focused on the proportion of discoveries that are false positives
- *Conceptual* difference between BH and Storey's q-value is:
 - BH **controls** the FDR
 - q-values give an unbiased **estimate** of the FDR (will control the FDR *on average*)

Storey's q-value vs BH (Mathematical)

- Mathematically, the difference between the two is in how m_0 is estimated
 - Or equivalently, how $\pi_0 = \frac{m_0}{m}$ is estimated (since m is known)
 - π_0 represents the overall proportion of tests that are truly null
- q-value:

$$\hat{q}(p_i) = \min_i \left(\frac{\hat{\pi}_0 m}{\text{rank}(p_i)} p_i, 1 \right)$$

- q-value and BH-adjusted p-values are equivalent when $\pi_0 = 1$

$$\hat{p}_{BH}(p_i) = \min_i \left(\frac{m}{\text{rank}(p_i)} p_i, 1 \right)$$

(BH conservatively assumes that $\pi_0 = 1$)

BH vs q-value in our example

Rank (j)	P-value	$\hat{p}_{BH}(p_i)$	$\hat{q}(p_i)$
1	0.0008	0.045	0.045
2	0.009	0.045	0.045
3	0.127	0.423	0.423
4	0.205	0.513	0.513
5	0.396	0.750	0.750
6	0.450	0.750	0.750
7	0.641	0.916	0.916
8	0.781	0.976	0.976
9	0.900	0.993	0.993
10	0.993	0.993	0.993

FDR control is an active area of research

The screenshot shows the 'Genome Biology' journal website. The article title is 'A practical guide to methods controlling false discoveries in computational biology'. It was published on 04 June 2019, is open access, and has 16k accesses and 12 citations. The authors listed are Keegan Korthauer, Patrick K. Kimes, Claire Duvallet, Alejandro Reyes, Ayshwarya Subramanian, Mingxiang Teng, Chinmay Shukla, Eric J. Alm & Stephanie C. Hicks.

Genome Biology

Home About Articles Submission Guidelines

Research | Open Access | Published: 04 June 2019

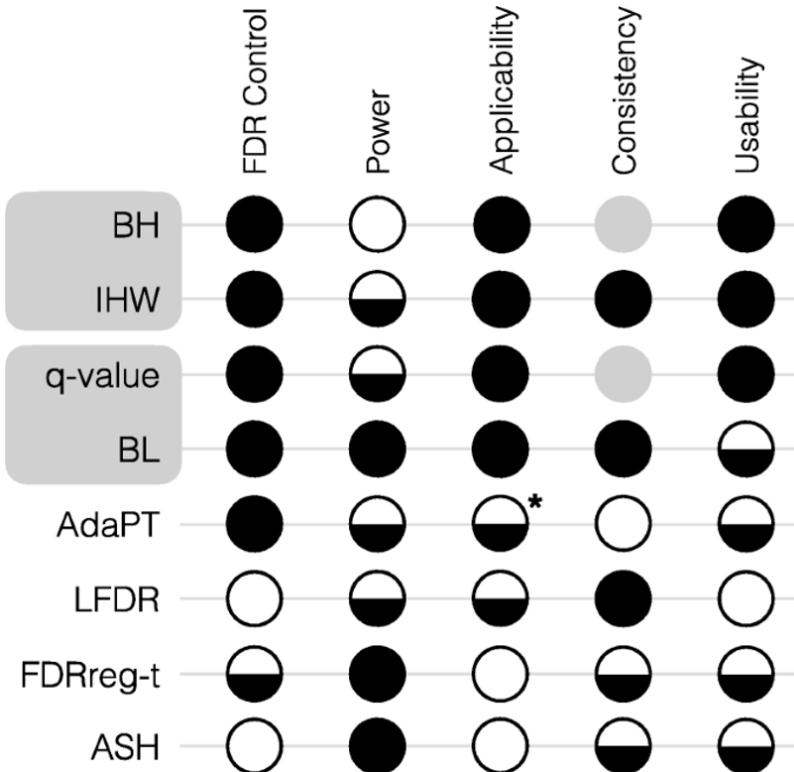
A practical guide to methods controlling false discoveries in computational biology

Keegan Korthauer, Patrick K. Kimes, Claire Duvallet, Alejandro Reyes, Ayshwarya Subramanian, Mingxiang Teng, Chinmay Shukla, Eric J. Alm & Stephanie C. Hicks

Genome Biology 20, Article number: 118 (2019) | [Cite this article](#)

16k Accesses | 12 Citations | 156 Altmetric | [Metrics](#)

Korthauer, Kimes et al. 2019



Compounding issues of multiple comparisons

- What if you're not only testing 20K genes, but also multiple tests per gene (e.g. multiple contrasts, such as several two-group comparisons)?
- Classical procedures for adjustment:
 - Tukey multiple comparison procedure
 - Scheffe multiple comparison procedure
 - Bonferroni or Holm FWER correction
- In our setting, we can also apply BH to all p-values globally
 - `limma::decideTests(pvals, method="global")` for a matrix of p-values or `eBayes` output (e.g. rows = genes, columns = contrasts)
 - p-values are combined, adjusted globally, then separated back out and sorted

Assumptions about p-values

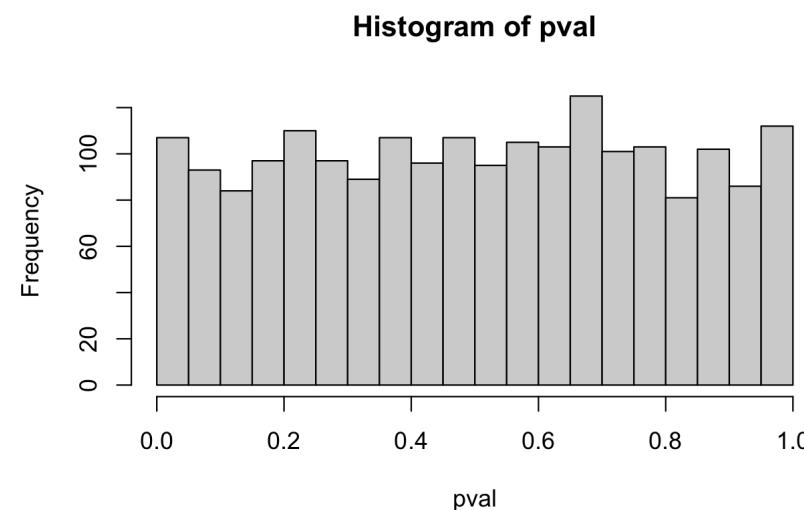
Implicit assumption for all multiple testing correction methods: p-value distribution is "well-behaved"

Assumptions about p-values

Implicit assumption for all multiple testing correction methods: p-value distribution is "well-behaved"

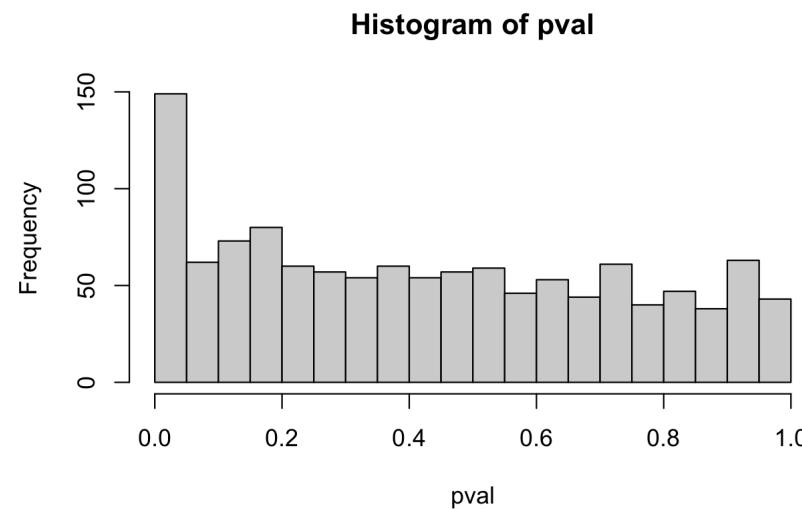
What does this mean?

Primarily, that the distribution of p-values *under the null* is **uniform** (i.e. flat)



p-value distributions

Spike of small p-values indicates non-null tests:



Great primer on how things can go wrong: <http://varianceexplained.org/statistics/interpreting-pvalue-histogram/>

What if p-values are *poorly behaved*?

- FDR estimates can be invalid (assumptions are violated)
- Possible solution: nonparametric test
- Alternative: estimate sampling distribution or p-values "empirically" using resampling techniques
 - **Bootstrap**: estimate sampling distribution of a statistic of interest (e.g. mean expression in each group) by taking repeated random samples *with replacement* from your dataset and recomputing. Use these to estimate confidence intervals without assuming data follows a particular distribution
 - **Permutation**: construct a simulated version of your dataset that satisfies the null hypothesis and compute statistic (e.g. shuffle group labels for a two-group comparison); repeat many times and use permutation statistics as your sampling distribution rather than a t, Normal, F, χ^2 , etc
- Downsides: often computationally intensive for genomics; not optimal for small sample sizes