More `limma` & multiple testing

Keegan Korthauer

February 9, 2023

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`

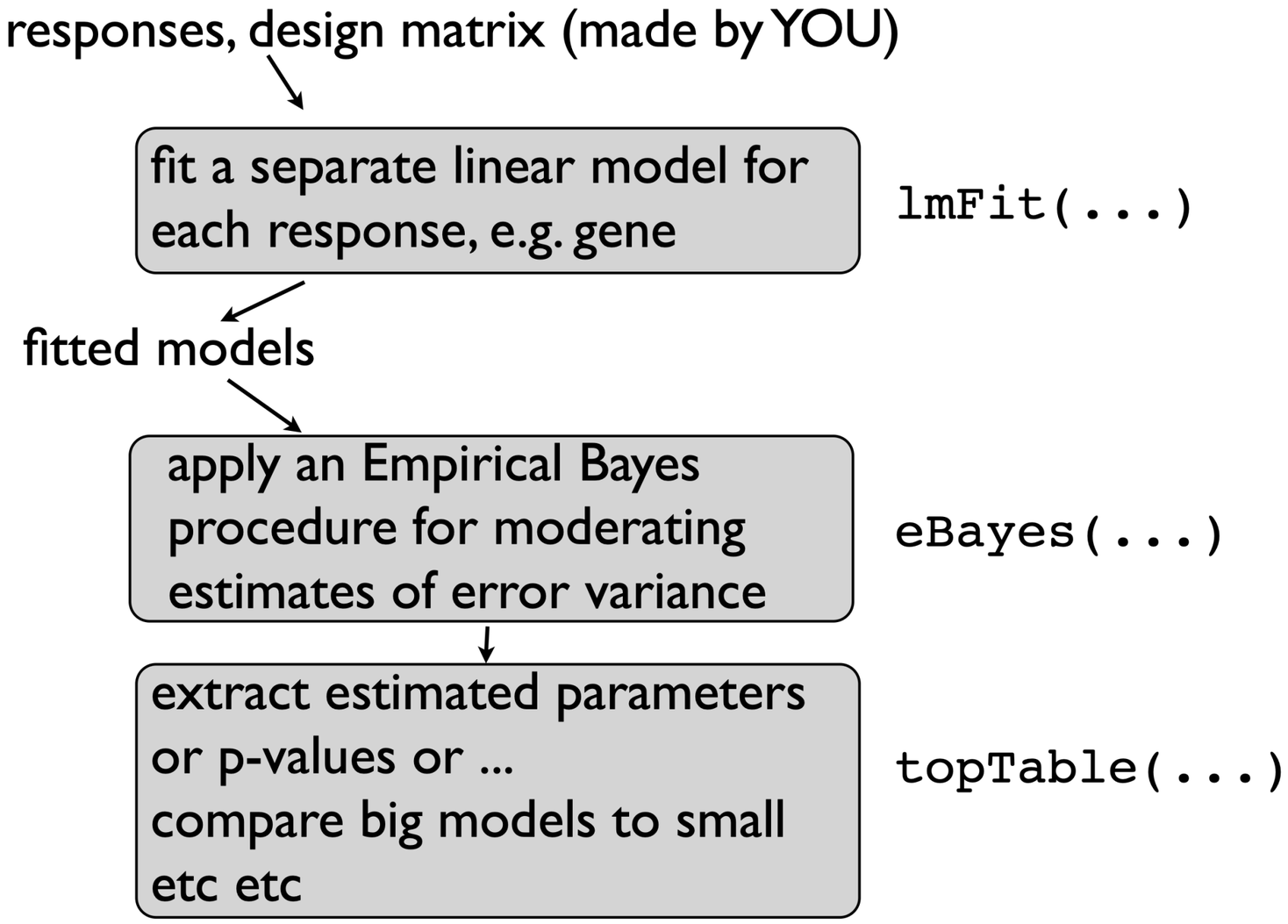
$$\tilde{s}_g^2 = \frac{d_0 s_0^2 + d s_g^2}{d_0 + d}$$

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

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

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

Getting help

Documentation

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

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

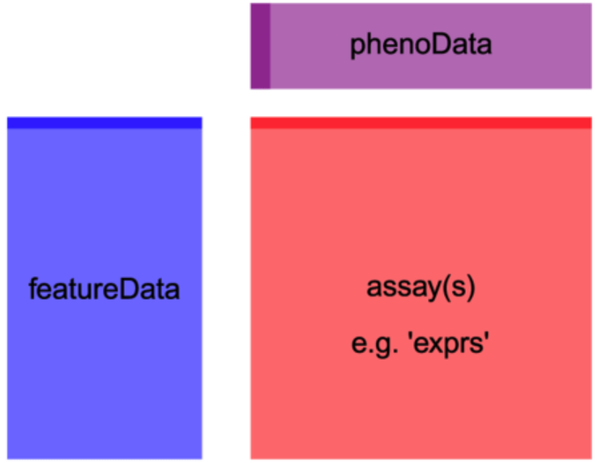
PDF		Limma One Page Introduction
PDF		usersguide.pdf
PDF		Reference Manual
Text		NEWS

- 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; image source

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

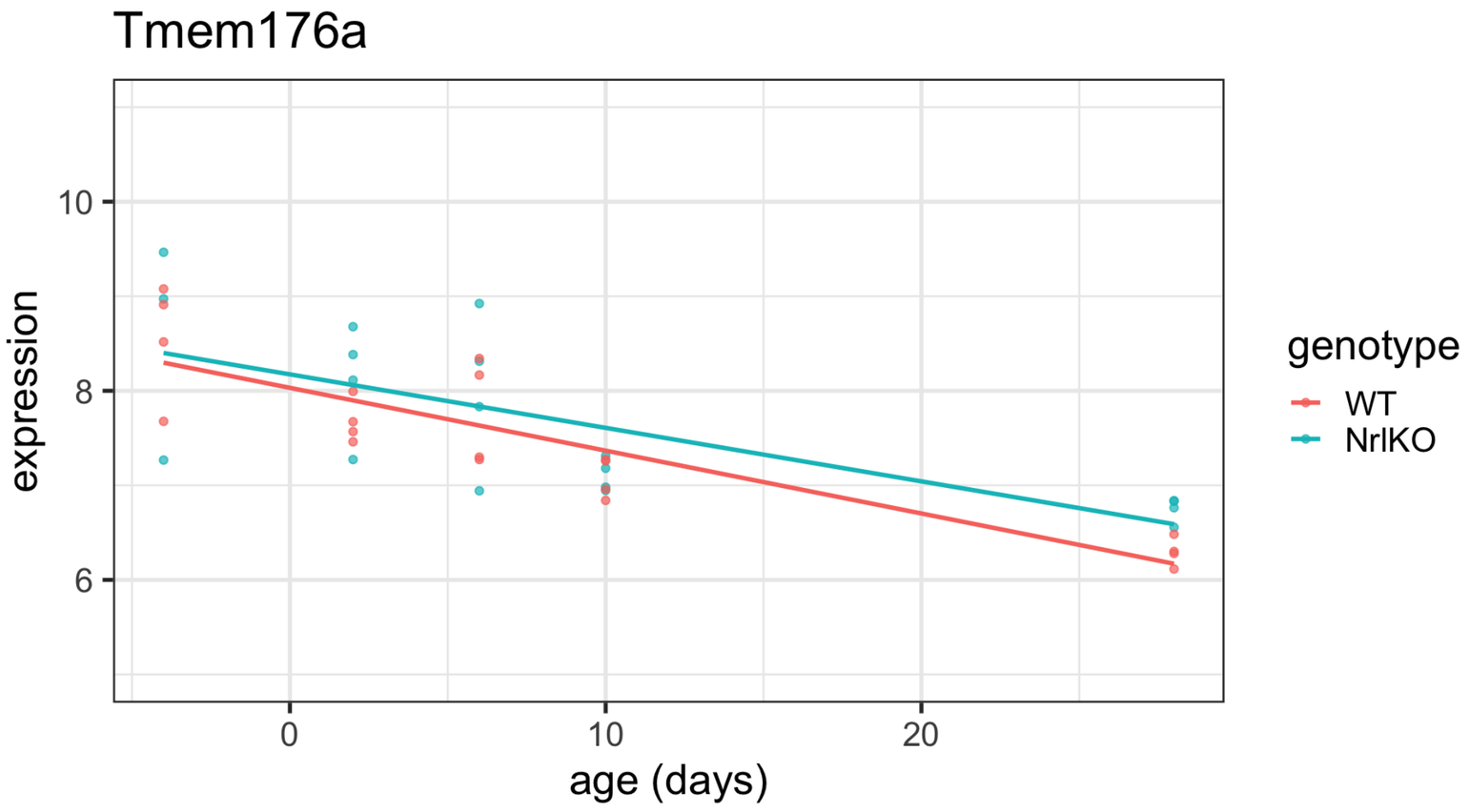
$$y_{ig} = \theta + \tau_{KO}x_{ig,KO} + \tau_{Age}x_{ig,Age} + \tau_{KO:Age}x_{ig,KO}x_{ig,Age}$$

- 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):

▶ Code



Arranging `lmFit` input: Bioconductor way

```
1 eset
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 45101 features, 39 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM92610 GSM92611 ... GSM92648 (39 total)
  varLabels: title geo_accession ... age (40 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 16505381
Annotation: GPL1261
```

Arranging `lmFit` input: Separate expression/metadata

Data matrix | Sample metadata

```
myDat <- assayData(eset)$exprs
dim(myDat)
```

```
[1] 45101    39
```

```
head(myDat)
```

```
             GSM92610 GSM92611  GSM92612 GSM92613  GSM92614  GSM92615 GSM92616
1415670_at   7.108863 7.322392  7.420947 7.351444  7.240428  7.336860 7.382548
1415671_at   9.714002 9.797742  9.831072 9.658442  9.708906 10.030904 7.636955
1415672_at   9.429030 9.846977 10.003092 9.914112 10.174548 10.242718 8.422627
1415673_at   8.426974 8.404206  8.594600 8.404206  8.835334  8.371316 8.362592
1415674_a_at 8.498338 8.458287  8.426651 8.372776  8.541722  8.892941 8.509461
1415675_at   9.739536 9.508494  9.598063 9.454508  9.672208  9.613500 9.655965
             GSM92617  GSM92618 GSM92619 GSM92620 GSM92621 GSM92622 GSM92623
1415670_at   7.215360  7.215360 7.118826 7.335319 7.087594 7.154789 7.076775
1415671_at   9.213571  9.488295 9.149957 9.330127 9.564756 9.554756 9.321551
1415672_at   9.590419 10.034838 9.842881 9.752964 9.881279 9.842790 9.241509
1415673_at   8.309017  8.546114 8.322816 8.430680 8.574776 8.329279 8.297757
1415674_a_at 8.309017  8.495662 8.209640 8.481508 8.586018 8.495662 8.480084
1415675_at   9.754453  9.806620 9.723386 9.700739 9.865178 9.970262 9.924127
             GSM92624  GSM92625 GSM92626  GSM92627 GSM92628 GSM92629 GSM92630
```

Formulating `lmFit` input: 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)	genotypeNrlKO	age	genotypeNrlKO: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

Computation is fast

- Bioconductor way:

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

```
   user  system elapsed
  0.111   0.031   0.144
```

Important

Using Bioconductor objects that store the measurements and metadata in one object reduces the risk of errors due to subsetting, reordering, etc.

- 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: data wrangling, selecting a model, choosing how to parameterize it, and interpreting it

Output of `lmFit`

```
1 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
```

```
1 nrow(eset)
```

```
Features
   45101
```

```
1 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

```
1 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
```

```
1 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
```

Components of the empirical Bayes estimators

math	plain english	`limma`	numerical result	also in `lm`?
s_g^2	gene-specific residual variance	`gFit$sigma^2`	45K numbers	✓
d	residual degrees of freedom $(n - p)$	`gFit$df.residual`	$39 - 4 = 35$[1]	✓
s_0^2	mean of inverse χ^2 prior for s_g^2	`ebFit$s2.prior`	0.048	
d_0	degrees of freedom for the prior	`ebFit$df.prior`	2.61	
$\tilde{s}_g^2$	posterior mean of s_g^2 (moderated resid. var.)	`ebFit$s2.post`	45K numbers	
$\tilde{t}_g$	moderated t statistics	`ebFit$t`	45K×p matrix	

1. 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()` extracts 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)
```

- Refer to the help page (`?topTable`) for full details of each argument
- Summary of key `topTable` arguments:
  - `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: `genotypeNrlKO`

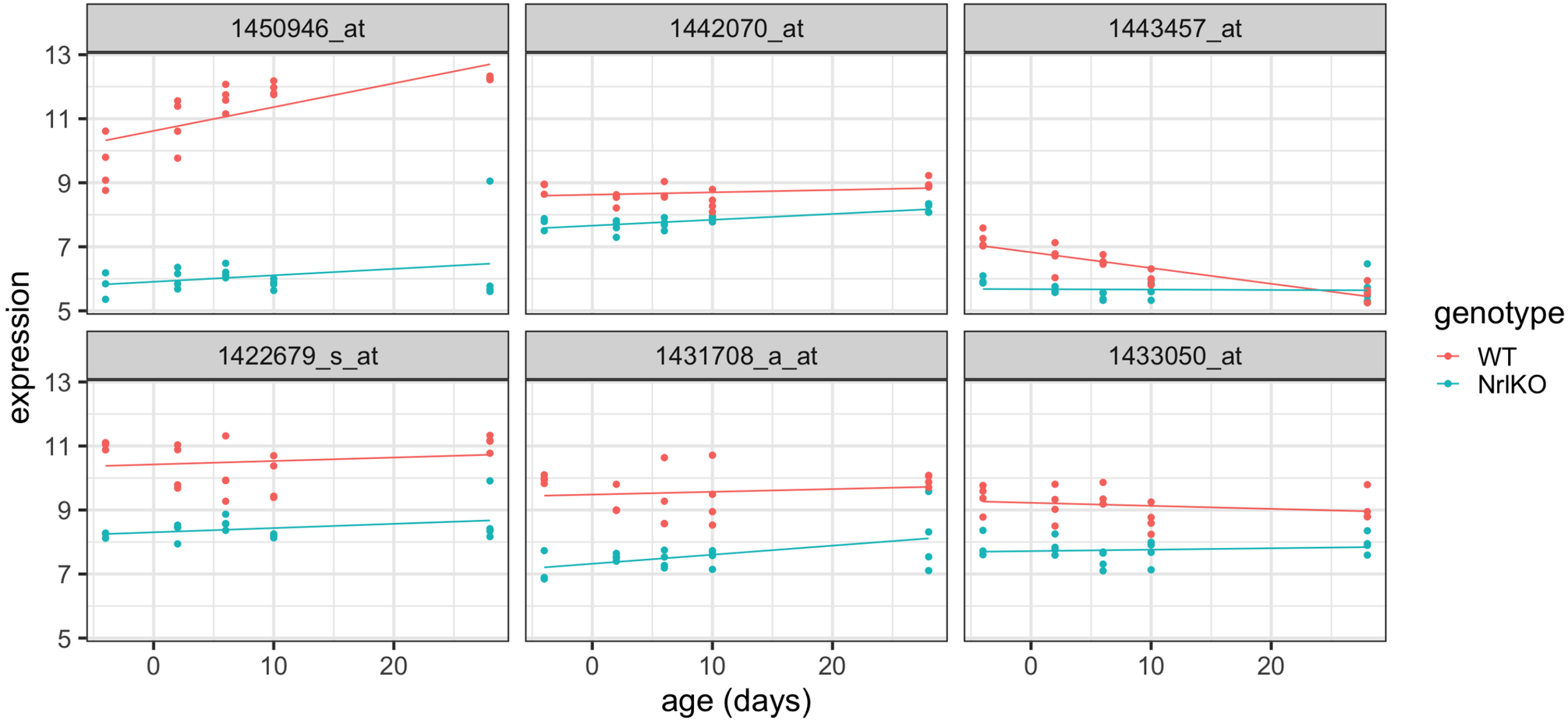
```
1 topTable(ebFit, coef = "genotypeNrlKO")
```

```
                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
```

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

Plot the top 6 probes for `genotypeNrlKO`

▶ Code



`topTable` in action: `age`

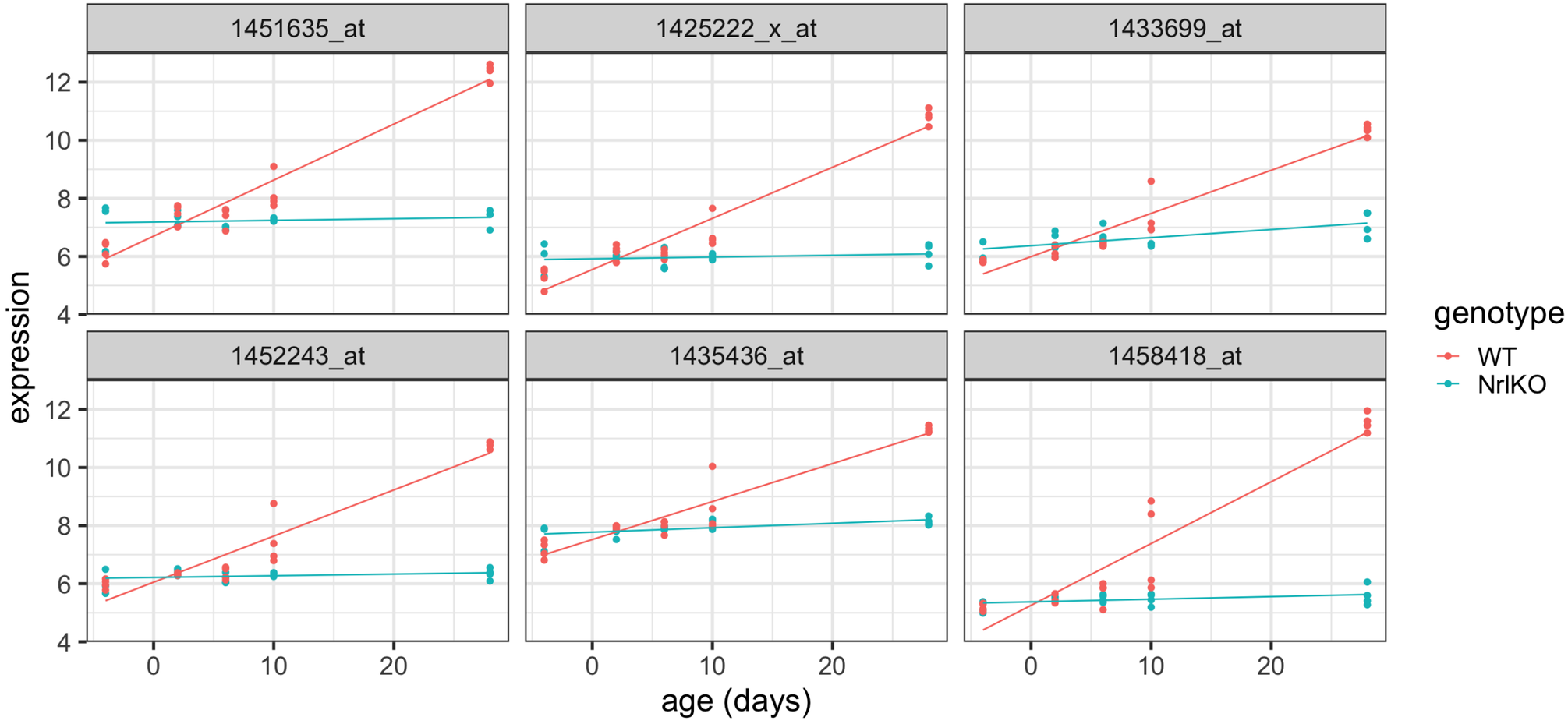
```
1 topTable(ebFit, coef = "age")
```

```
                  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?

Plot the top 6 probes for age

▶ Code



`topTable` in action: `genotypeNrlKO:age`

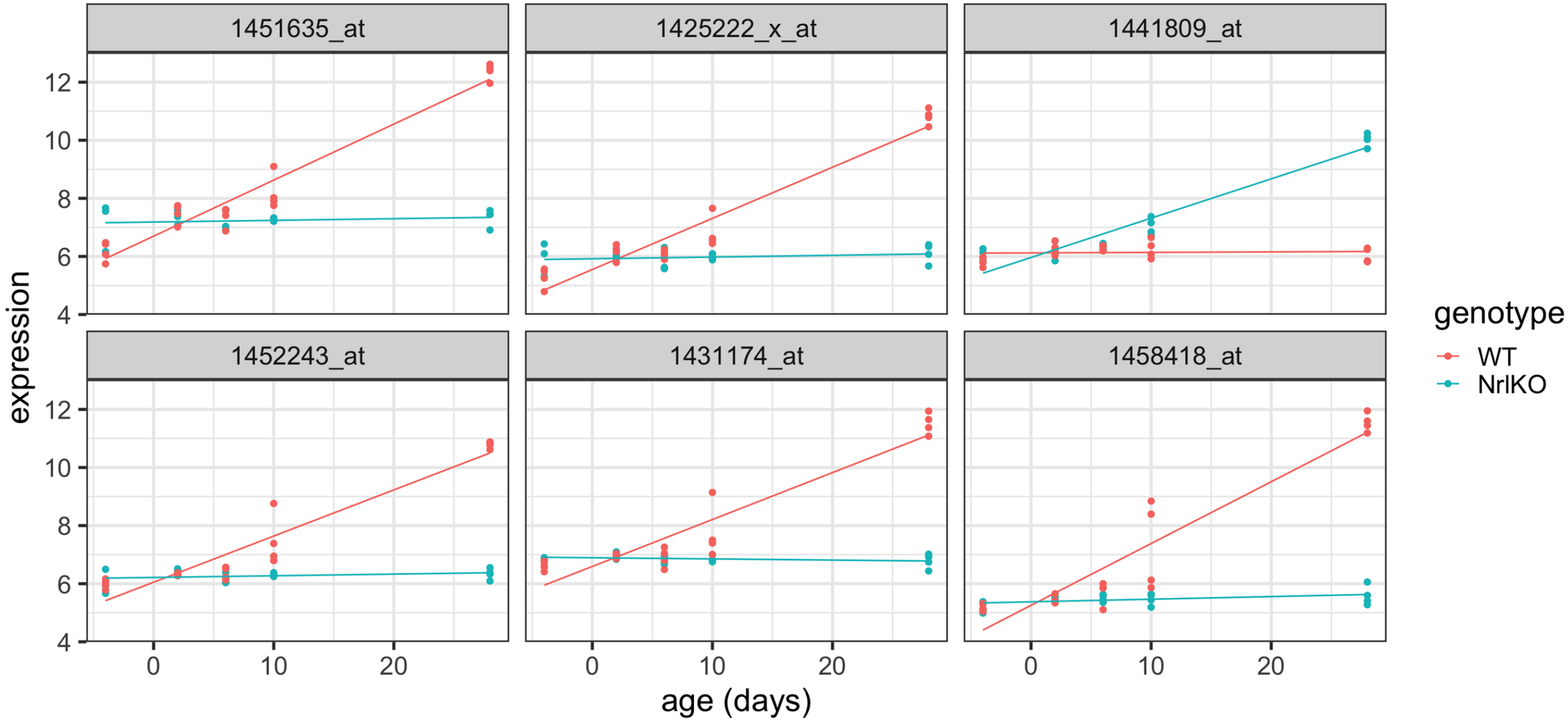
```
1 topTable(ebFit, coef = "genotypeNrlKO:age")
```

```
                 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?

Plot the top 6 probes for `genotypeNrlKO:age`

▶ Code



`topTable` in action: any effect of genotype

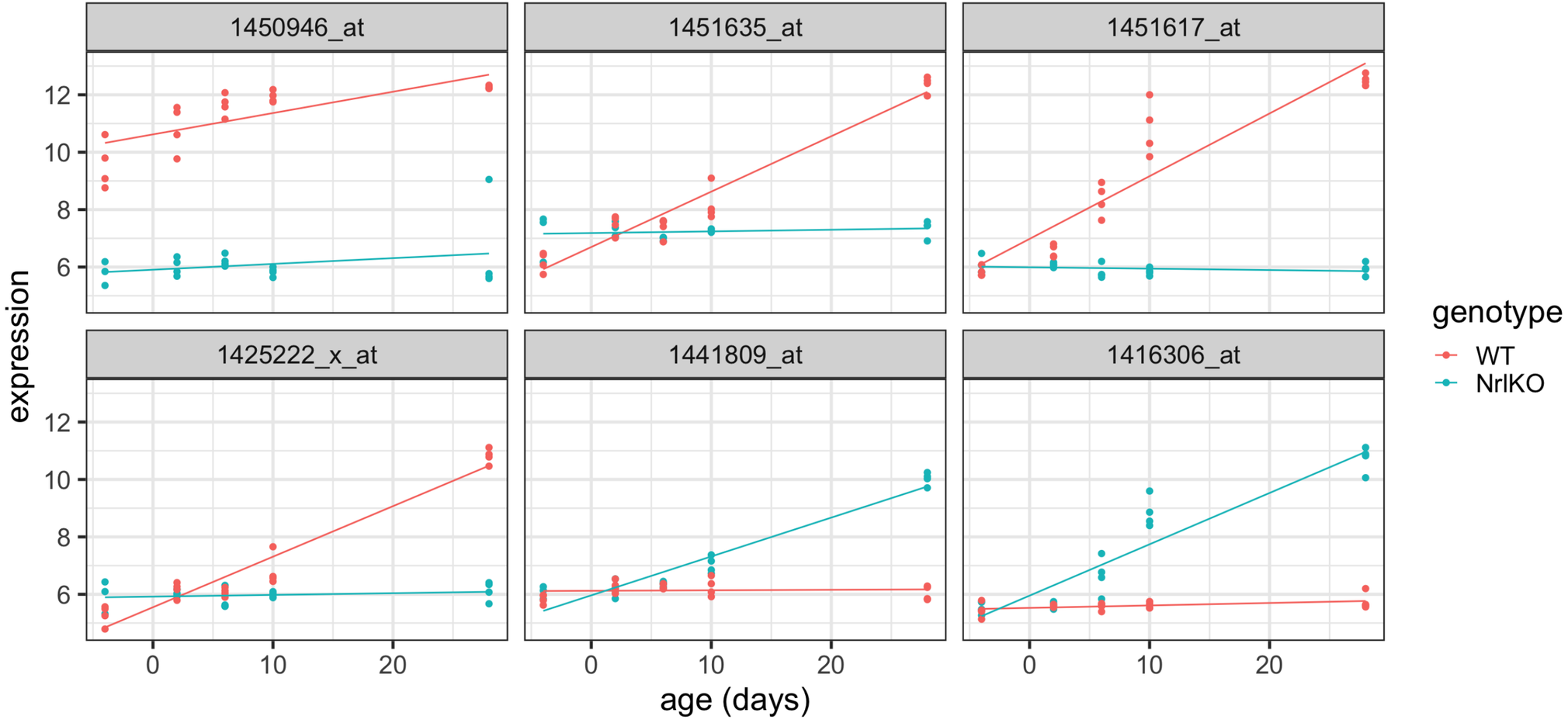
```
1 topTable(ebFit, coef = c("genotypeNrlKO", "genotypeNrlKO:age"))
```

```
             genotypeNrlKO genotypeNrlKO.age  AveExpr        F      P.Value
1450946_at      -4.7134891       -0.05411298 8.733103 246.5762 2.403638e-22
1451635_at       0.4868705       -0.18718005 7.791237 129.8112 1.307185e-17
1451617_at      -0.9885684       -0.22329233 7.419244 128.9935 1.450111e-17
1425222_x_at     0.3698846       -0.17009077 6.514475 120.0041 4.719613e-17
1441809_at      -0.1576064        0.13363563 6.649440 117.5571 6.594308e-17
1416306_at       0.4278776        0.17020589 6.560050 113.2125 1.212332e-16
1439143_at       0.6570269        0.06170447 5.856507 111.2102 1.616023e-16
1449526_a_at     0.9935801        0.04232958 7.372946 110.5598 1.775851e-16
1452243_at       0.1639373       -0.15313265 6.840975 106.2784 3.344140e-16
1451763_at      -0.6819899       -0.14010359 8.160065 102.7010 5.771358e-16
                adj.P.Val
1450946_at   1.084065e-17
1451635_at   2.180049e-13
1451617_at   2.180049e-13
```

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

Plot the top 6 probes for any effect of genotype

▶ Code



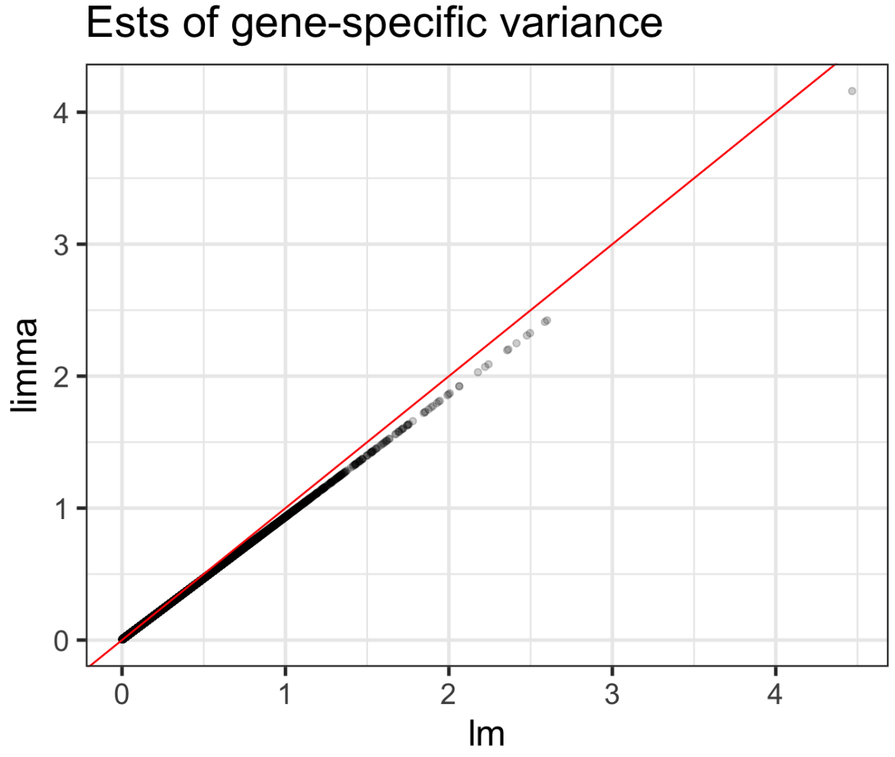
Comparison of s_g^2 and $\tilde{s}_g^2$ (shrinkage!)

Fill in the blank:

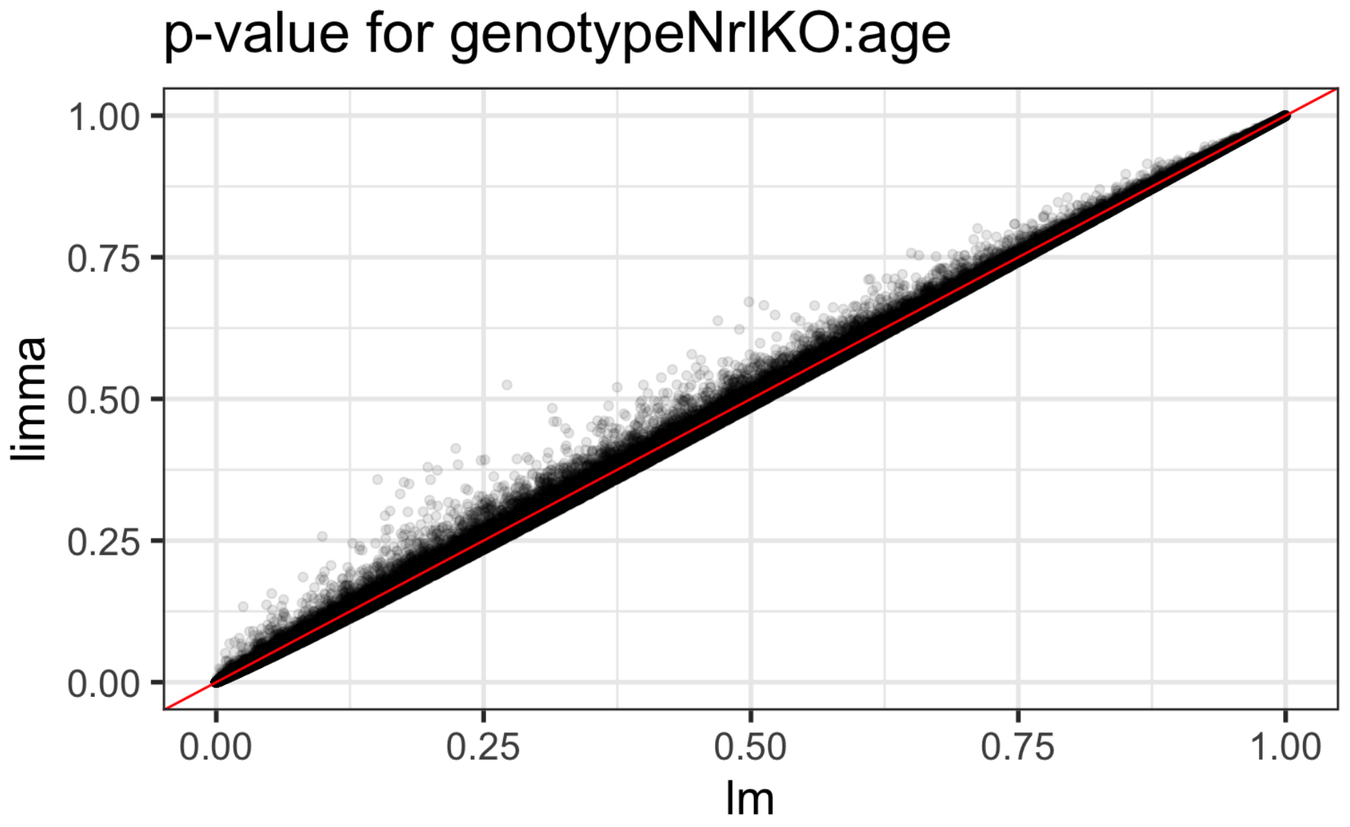
1. For **large** variances, limma _____ the gene-specific variance estimates.
   a. Increases
   b. Decreases
2. For **small** variances, limma _____ the gene-specific variance estimates.
   a. Increases
   b. Decreases

original scale | log-scale

▶ Code



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

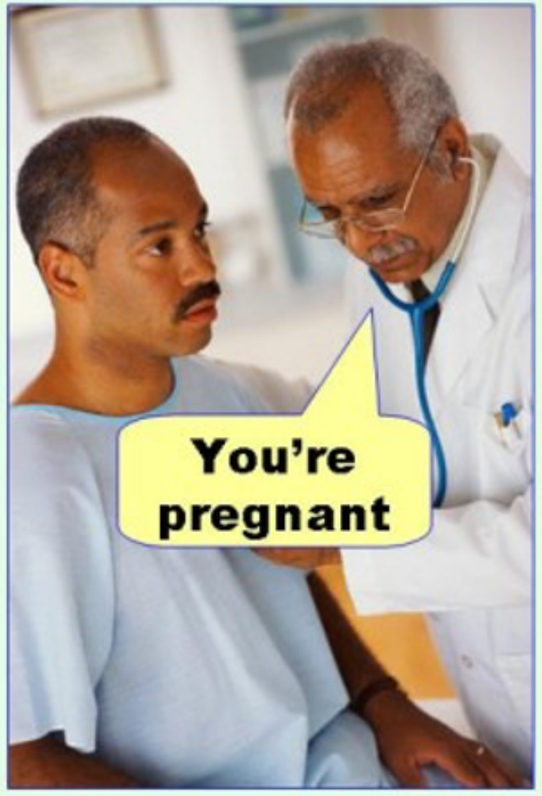
Multiple testing

What do we do with our lists of thousands of p-values?

Recall the two types of errors

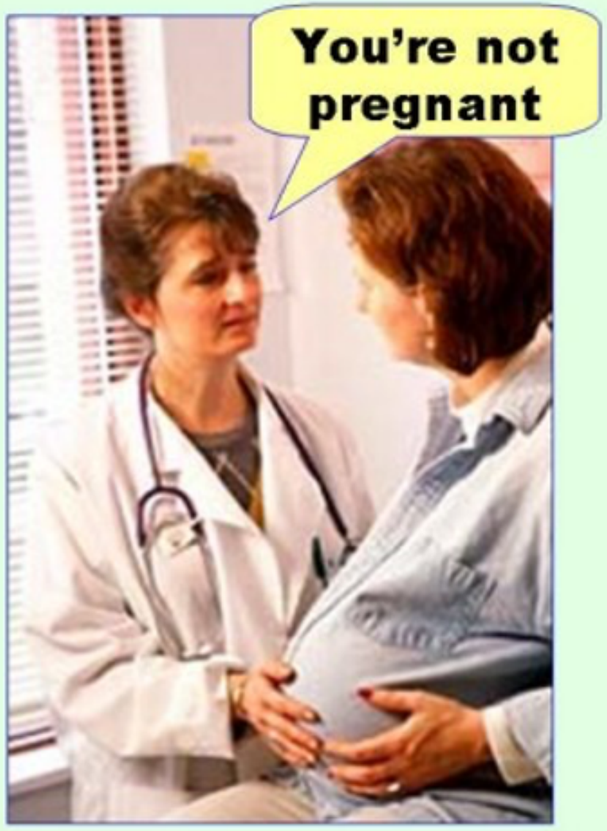
Type I error

(false positive)



Type II error

(false negative)



Error rates

Decision	Actual Situation "Truth": H_0 True	Actual Situation "Truth": H_0 False
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 $\alpha = 0.05$
- $P(\text{correct decision}|H_0) = 1 - \alpha = 0.95$
- $P(\text{correct decision on } m \text{ tests}|H_0) =$

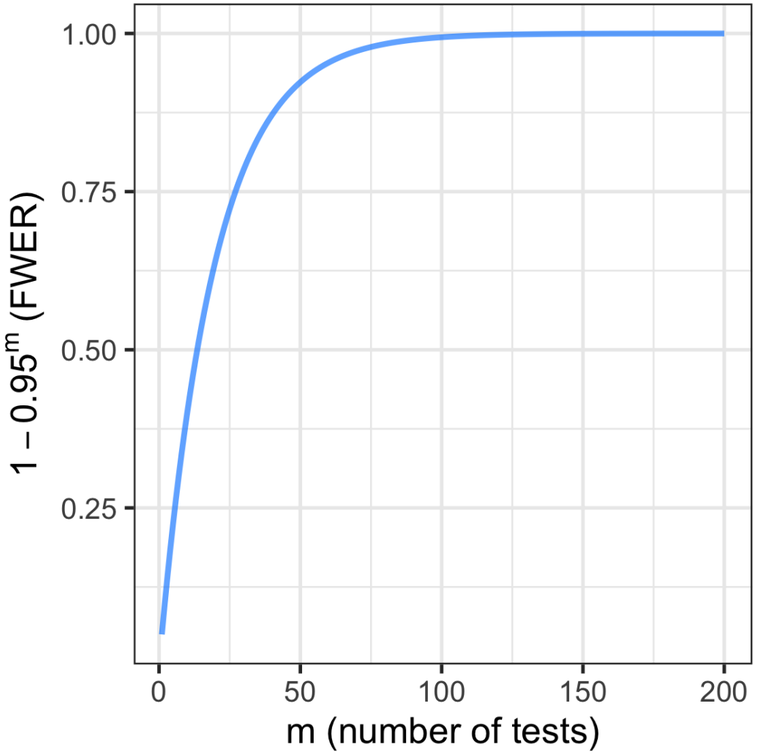
$$(1 - \alpha)^m = 0.95^m$$

- $P(\text{at least one error on } m \text{ tests}|H_0) =$

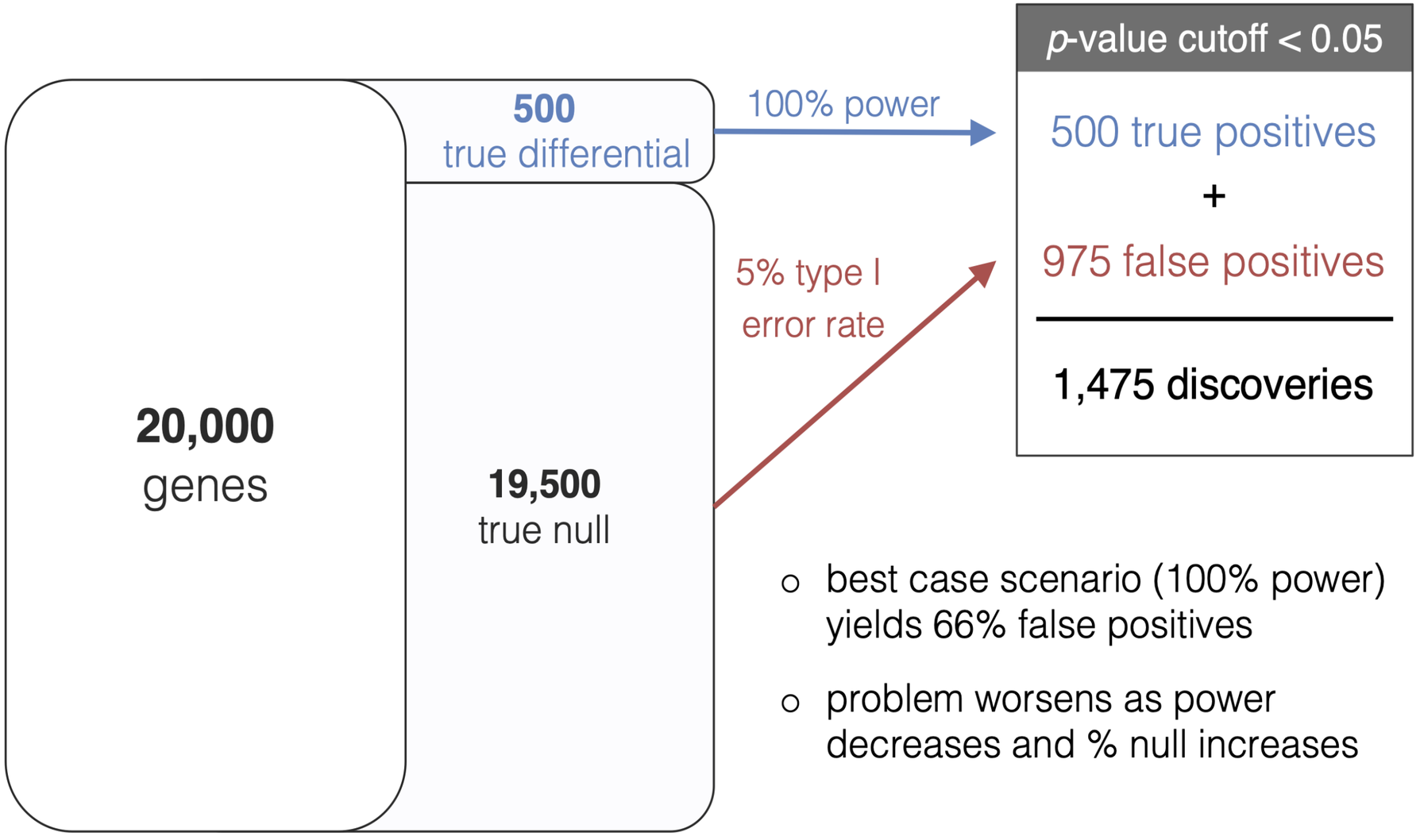
$$1 - (1 - \alpha)^m =$$

$$1 - 0.95^m = \alpha_{FWER}$$

▶ Code



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

- **Control the FWER**: limit the probability of making at least one incorrect decision
- One example: the **Bonferroni** correction for $\alpha = 0.05$:

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

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

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

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

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**:

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

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

$$\text{Then, threshold } p_{Bon,i} \text{ at } \alpha$$

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

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

$$\text{Then, threshold } p_i \text{ at } \alpha_{Bon}$$

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

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

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

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

$$\vdots$$

$$\Rightarrow FWER \leq \alpha$$

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… 😭

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

Radical idea

FWER is not that relevant in high-throughput studies

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 90K 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)

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

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 $p_1 \leq p_2 \leq \ldots \leq p_m$
  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-val	BH-thr: jq/m	Reject H_0?	$FWER < 0.05$?
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

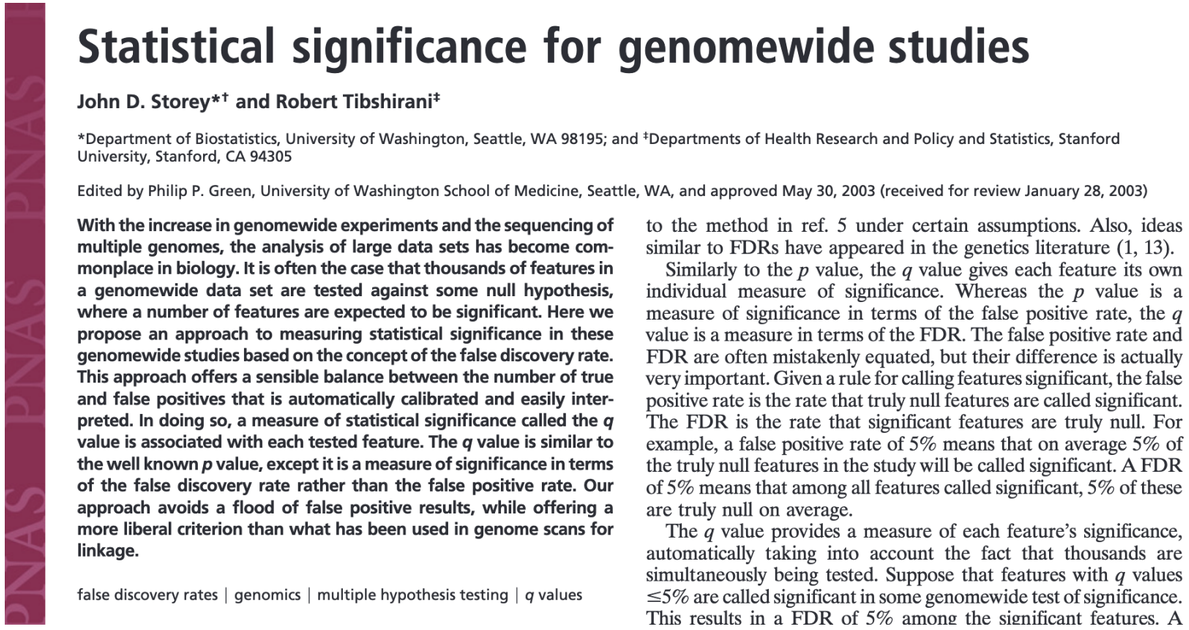
Or, obtain them yourself for any vector of p-values `p` with `p.adjust(p, method="BH")`:

```
pvals <- topTable(ebFit, coef = "genotypeNrlKO", number = Inf)$P.Value
p.adjust(pvals, method = "BH") %>% head()
```

```
[1] 2.431364e-13 1.555132e-07 6.029428e-07 6.029428e-07 6.029428e-07
[6] 6.029428e-07
```

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



PNAS PNAS PNAS

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

to the method in ref. 5 under certain assumptions. Also, ideas similar to FDRs have appeared in the genetics literature (1, 13).

Similarly to the *p* value, the *q* value gives each feature its own individual measure of significance. Whereas the *p* value is a measure of significance in terms of the false positive rate, the *q* value is a measure in terms of the FDR. The false positive rate and FDR are often mistakenly equated, but their difference is actually very important. Given a rule for calling features significant, the false positive rate is the rate that truly null features are called significant. The FDR is the rate that significant features are truly null. For example, a false positive rate of 5% means that on average 5% of the truly null features in the study will be called significant. A FDR of 5% means that among all features called significant, 5% of these are truly null on average.

The *q* value provides a measure of each feature's significance, automatically taking into account the fact that thousands are simultaneously being tested. Suppose that features with *q* values ≤5% are called significant in some genomewide test of significance. This results in a FDR of 5% among the significant features. A

- `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:

[Math Processing Error]

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

$$\hat{p}_{BH}(p_i) = \min_{i \leq j} \left(\frac{m}{j} p_{(j)} \right)$$

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

BH vs q-value in toy example

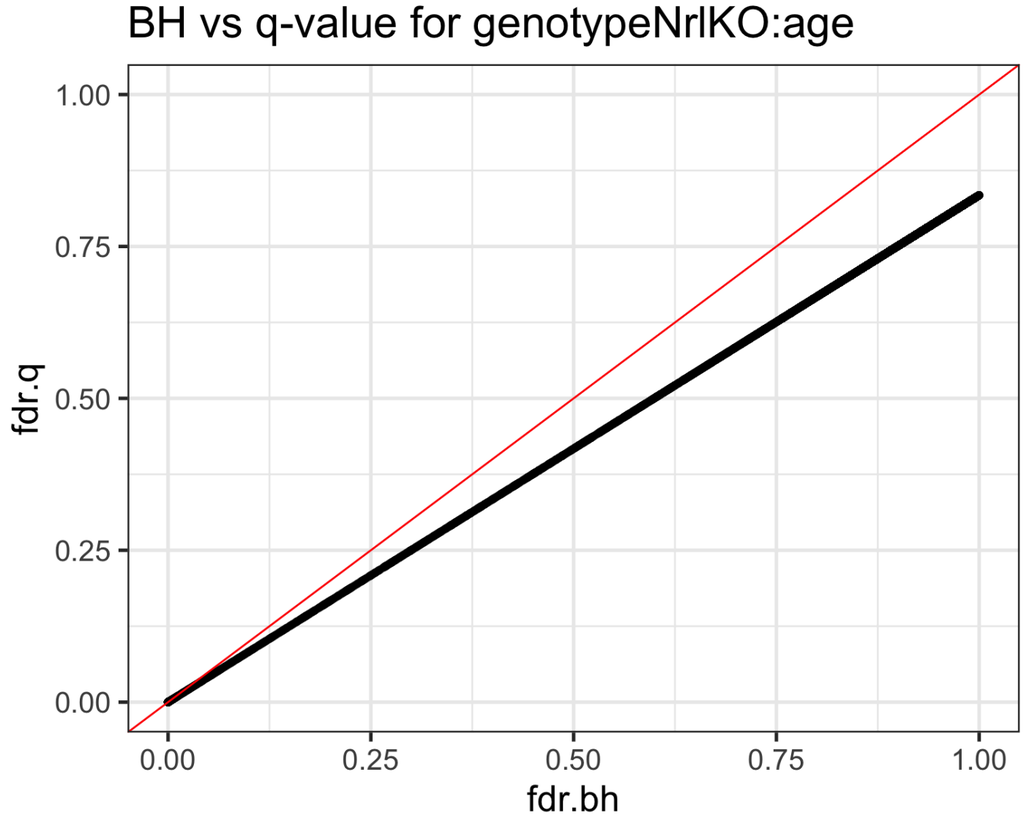
Rank (j)	P-value	$\hat{p}_{BH}(p_i)$	$\hat{q}(p_i)$[1]
1	0.0008	0.045	0.045
2	0.009	0.045	0.045
3	0.127	0.423	0.423
4	0.205	0.513	0.513
5	0.396	0.750	0.750
6	0.450	0.750	0.750
7	0.641	0.916	0.916
8	0.781	0.976	0.976
9	0.900	0.993	0.993
10	0.993	0.993	0.993

1. $\hat{\pi}_0 = 1$ in this case

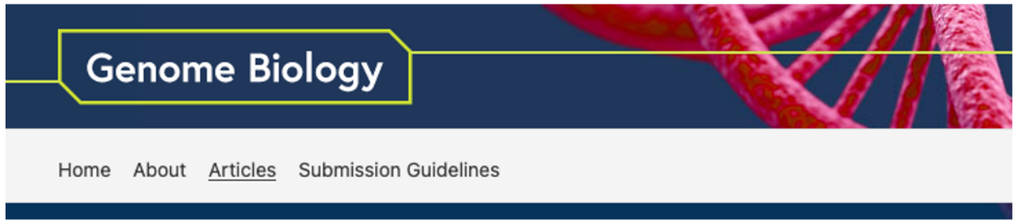
BH vs q-value in effect of `genotypeNrlKO:age`

Original scale | log-scale

▶ Code



FDR control is an active area of research



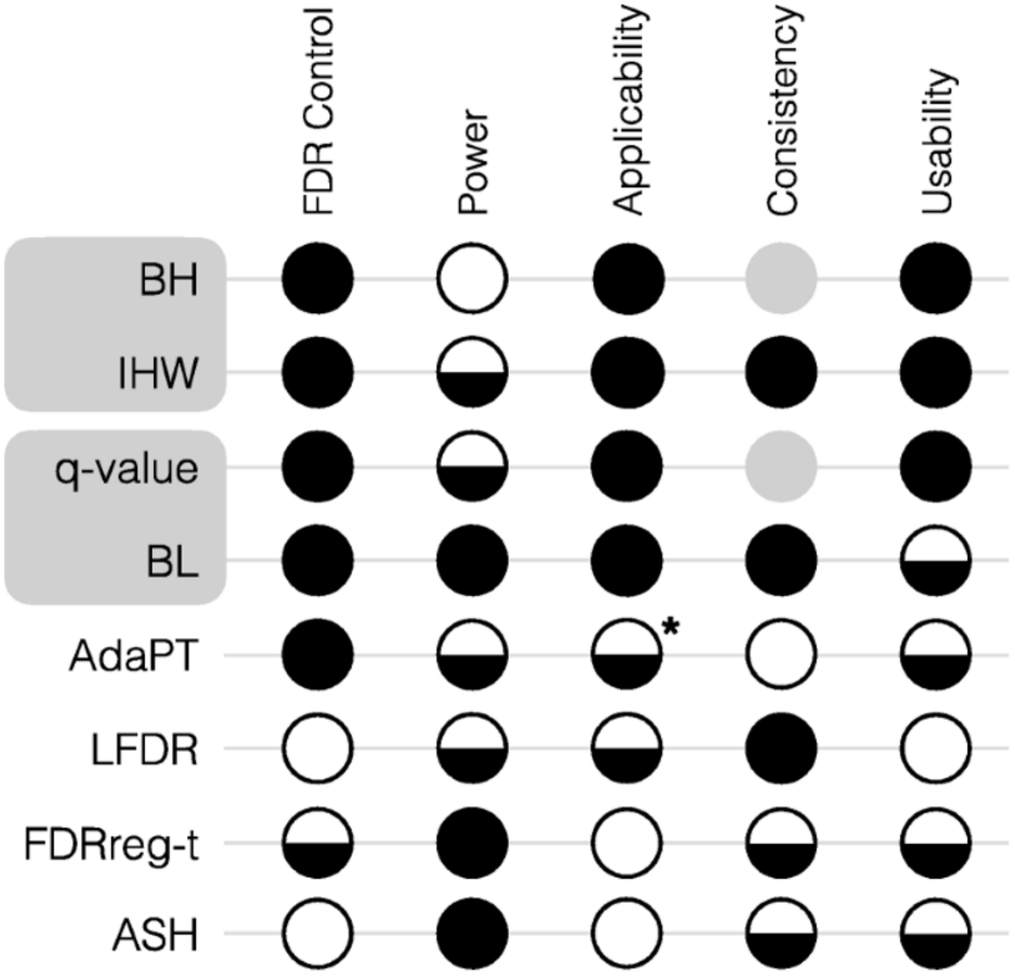
Research | Open Access | Published: 04 June 2019

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

Korthauer*, Kimes* et al. 2019



Compounding issues of multiple comparisons

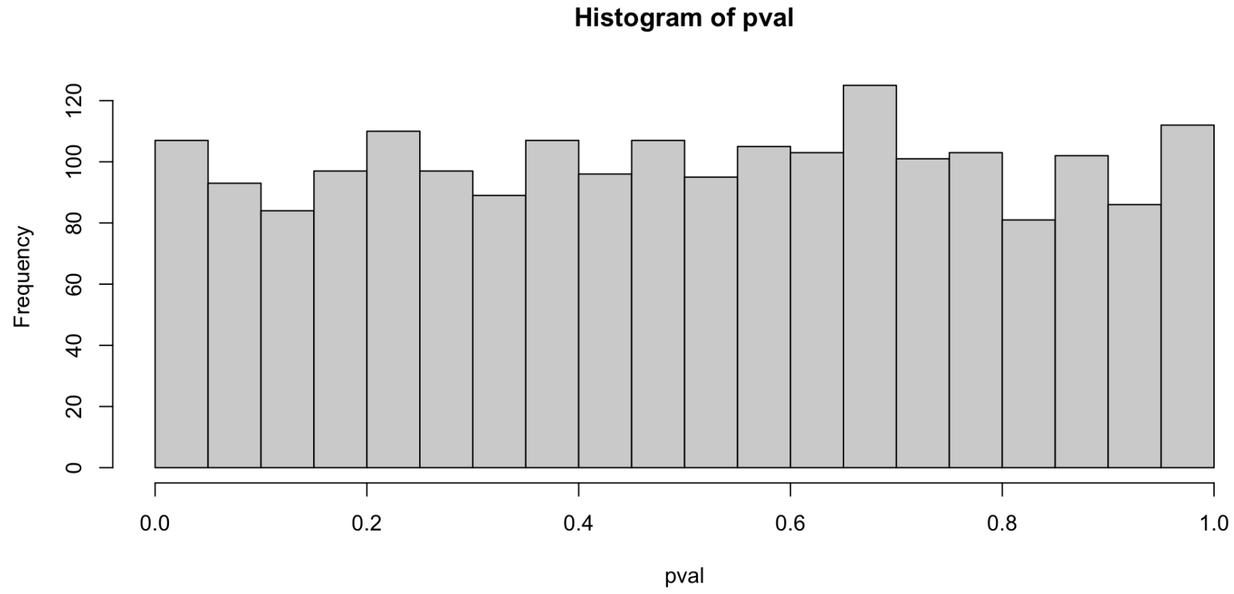
- What if you're not only testing 20K genes, but also multiple tests per gene (e.g. multiple contrasts, such as several two-group comparisons)?
- Classical procedures for adjustment:
  - Tukey multiple comparison procedure
  - Scheffe multiple comparison procedure
  - Bonferroni or Holm FWER correction
- 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

> **Note**
>
> Implicit assumption for all multiple testing correction methods: p-value distribution is “well-behaved”

What does this mean?

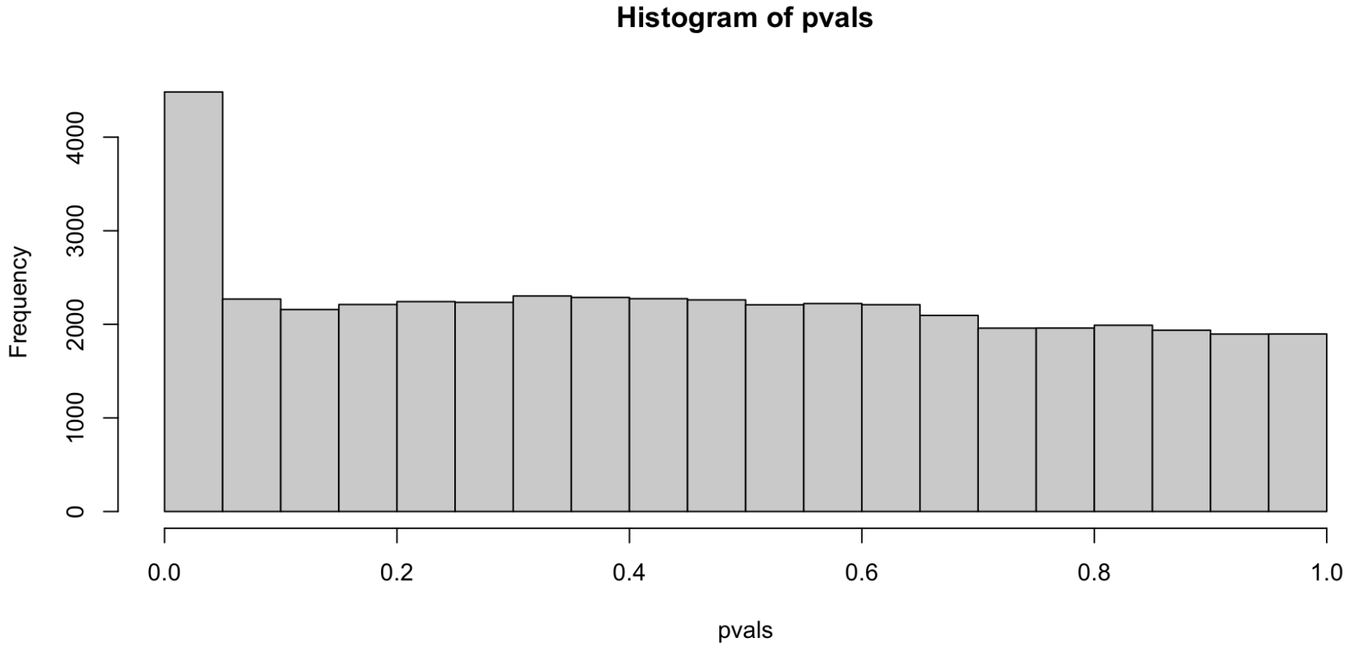
Primarily, that the distribution of p-values *under the null* is **uniform** (i.e. flat)



p-value distributions

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

▶ Code



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
  - Limitation: lack of flexibility to adjust for multiple covariates

- Another possible solution: estimate sampling distribution or p-values "empirically" using resampling techniques
  - **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
  - Limitation: often computationally intensive for genomics; not optimal for small sample sizes