Lecture 9 – Multiple Testing

STAT/BIOF/GSAT 540: Statistical Methods for High Dimensional Biology

Keegan Korthauer

2020/02/03

Slides by: Keegan Korthauer with contributions from Gabriela Cohen Freue, Jenny Bryan, and Sara Mostafavi

Today

- practical usage of `limma`
- comparison to `lm`
- multiple testing & adjusting for multiple comparisons

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:

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

- the exact estimation 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

`limma` quickstart

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

$$E(\boldsymbol{\varepsilon}_g) = 0,\ Var(\boldsymbol{\varepsilon}_g) = \sigma_g^2,\ \varepsilon_{ig} \perp \varepsilon_{jg},$$

$$(\uparrow \text{within each gene observations are iid / constant variance})$$

- `lmFit()` function in `limma` carries out multiple linear regression on each gene
- Usage: `lmFit(myDat, desMat)`
  - `myDat` is a data frame or matrix with one row per observation (G genes by N samples)
  - `desMat` is a design matrix (output of `model.matrix(y ~ x)`; N samples by p parameters)

Let's run `limma` for the interactive model with age

$$y_i = \theta + \tau_{KO} x_i^{KO} + \tau_{Age} x_i^{Age} + \tau_{KO:Age} x_i^{KO} x_i^{Age}$$

Formulating input for `lmFit`: gene expression data

```
str(myDat)
```

```
## 'data.frame':    29949 obs. of  39 variables:
##  $ Sample_20: num  7.24 9.48 10.01 8.36 8.59 ...
##  $ Sample_21: num  7.41 10.02 10.04 8.37 8.62 ...
##  $ Sample_22: num  7.17 9.85 9.91 8.4 8.52 ...
##  $ Sample_23: num  7.07 10.13 9.91 8.49 8.64 ...
##  $ Sample_16: num  7.38 7.64 8.42 8.36 8.51 ...
##  $ Sample_17: num  7.34 10.03 10.24 8.37 8.89 ...
##  $ Sample_6 : num  7.24 9.71 10.17 8.84 8.54 ...
##  $ Sample_24: num  7.11 9.75 9.39 8.37 8.36 ...
##  $ Sample_25: num  7.19 9.16 10.11 8.2 8.5 ...
##  $ Sample_26: num  7.18 9.49 9.41 8.73 8.39 ...
##  $ Sample_27: num  7.21 8.64 9.43 8.33 8.43 ...
##  $ Sample_14: num  7.09 9.56 9.88 8.57 8.59 ...
##  $ Sample_3 : num  7.16 9.55 9.84 8.33 8.5 ...
##  $ Sample_5 : num  7.08 9.32 9.24 8.3 8.48 ...
##  $ Sample_8 : num  7.11 8.24 9.13 8.13 8.33 ...
##  $ Sample_28: num  7.34 8.27 9.47 8.38 8.4 ...
##  $ Sample_29: num  7.66 10.03 9.88 8.56 8.69 ...
##  $ Sample_30: num  7.26 9.27 10.54 8.15 8.55 ...
##  $ Sample_31: num  7.31 9.26 10.1 8.37 8.49 ...
##  $ Sample_1 : num  7.15 9.87 9.68 8.28 8.5 ...
##  $ Sample_10: num  7.28 10.29 9.91 8.42 8.68 ...
##  $ Sample_4 : num  7.18 10.16 9.72 8.32 8.5 ...
##  $ Sample_7 : num  7.15 8.95 9.3 8.17 8.41 ...
##  $ Sample_32: num  7.54 9.53 9.92 8.78 8.57 ...
##  $ Sample_33: num  7.01 8.97 9.22 8.42 8.53 ...
```

Formulating input for `lmFit`: covariate data

```
head(myDes)
```

```
##      sidChar sidNum devStage gType Age
## 12 Sample_20     20      E16    wt  -4
## 13 Sample_21     21      E16    wt  -4
## 14 Sample_22     22      E16    wt  -4
## 15 Sample_23     23      E16    wt  -4
## 9  Sample_16     16      E16 NrlKO  -4
## 10 Sample_17     17      E16 NrlKO  -4
```

```
desMat <- model.matrix(~ gType*Age,
                       data=myDes)
```

```
desMat
```

```
##    (Intercept) gTypeNrlKO Age gTypeNrlKO:Age
## 12           1          0  -4              0
## 13           1          0  -4              0
## 14           1          0  -4              0
## 15           1          0  -4              0
## 9            1          1  -4             -4
## 10           1          1  -4             -4
## 11           1          1  -4             -4
## 28           1          0   2              0
## 29           1          0   2              0
## 30           1          0   2              0
## 31           1          0   2              0
## 24           1          1   2              2
## 25           1          1   2              2
## 26           1          1   2              2
## 27           1          1   2              2
## 36           1          0   6              0
## 37           1          0   6              0
## 38           1          0   6              0
## 39           1          0   6              0
## 32           1          1   6              6
## 33           1          1   6              6
## 34           1          1   6              6
## 35           1          1   6              6
```

Computation is fast

```r
library(limma)
system.time(gFit <- lmFit(myDat, desMat))
```

```
##    user  system elapsed
##   0.143   0.014   0.158
```

- Using `lmFit` to fit an interactive model on 30K 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`

```r
summary(gFit)
```

```
##                   Length Class  Mode
## coefficients      119796 -none- numeric
## rank                   1 -none- numeric
## assign                 4 -none- numeric
## qr                     5 qr     list
## df.residual        29949 -none- numeric
## sigma              29949 -none- numeric
## cov.coefficients      16 -none- numeric
## stdev.unscaled    119796 -none- numeric
## pivot                  4 -none- numeric
## Amean              29949 -none- numeric
## method                 1 -none- character
## design               156 -none- numeric
```

```r
29949*4
```

```
## [1] 119796
```

Output of `lmFit`

```
summary(gFit)
```

```
##                  Length Class  Mode
## coefficients     119796 -none- numeric
## rank                  1 -none- numeric
## assign                4 -none- numeric
## qr                    5 qr     list
## df.residual       29949 -none- numeric
## sigma             29949 -none- numeric
## cov.coefficients     16 -none- numeric
## stdev.unscaled   119796 -none- numeric
## pivot                 4 -none- numeric
## Amean             29949 -none- numeric
## method                1 -none- character
## design              156 -none- numeric
```

```
29949*4
```

```
## [1] 119796
```

- 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()`!

eBayes()

```
summary(gFit)
```

```
##                  Length Class  Mode
## coefficients     119796 -none- numeric
## rank                  1 -none- numeric
## assign                4 -none- numeric
## qr                    5 qr     list
## df.residual       29949 -none- numeric
## sigma             29949 -none- numeric
## cov.coefficients     16 -none- numeric
## stdev.unscaled   119796 -none- numeric
## pivot                 4 -none- numeric
## Amean             29949 -none- numeric
## method                1 -none- character
## design              156 -none- numeric
```

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

```
##                  Length Class  Mode
## coefficients     119796 -none- numeric
## rank                  1 -none- numeric
## assign                4 -none- numeric
## qr                    5 qr     list
## df.residual       29949 -none- numeric
## sigma             29949 -none- numeric
## cov.coefficients     16 -none- numeric
## stdev.unscaled   119796 -none- numeric
## pivot                 4 -none- numeric
## Amean             29949 -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           29949 -none- numeric
## t                119796 -none- numeric
## df.total          29949 -none- numeric
## p.value          119796 -none- numeric
## lods             119796 -none- numeric
## F                 29949 -none- numeric
## F.p.value         29949 -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`	30K numbers	✓
d	residual degrees of freedom $(n-p)$	`gFit$df.residual`	$39-4=35^*$	✓
s_0^2	mean of inverse χ^2 prior for s_g^2	`ebFit$s2.prior`	0.072	
d_0	degrees of freedom for the prior	`ebFit$df.prior`	2.9	
$\tilde{s}_g^2$	posterior mean of s_g^2 (i.e. moderated residual variance)	`ebFit$s2.post`	30K numbers	

* limma can handle more complicated models where this is not the same for each gene, so this is actually a vector of 30K 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 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 (defalyt is 1)
- `lfc` lets you specify a minimum observed effect size (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)

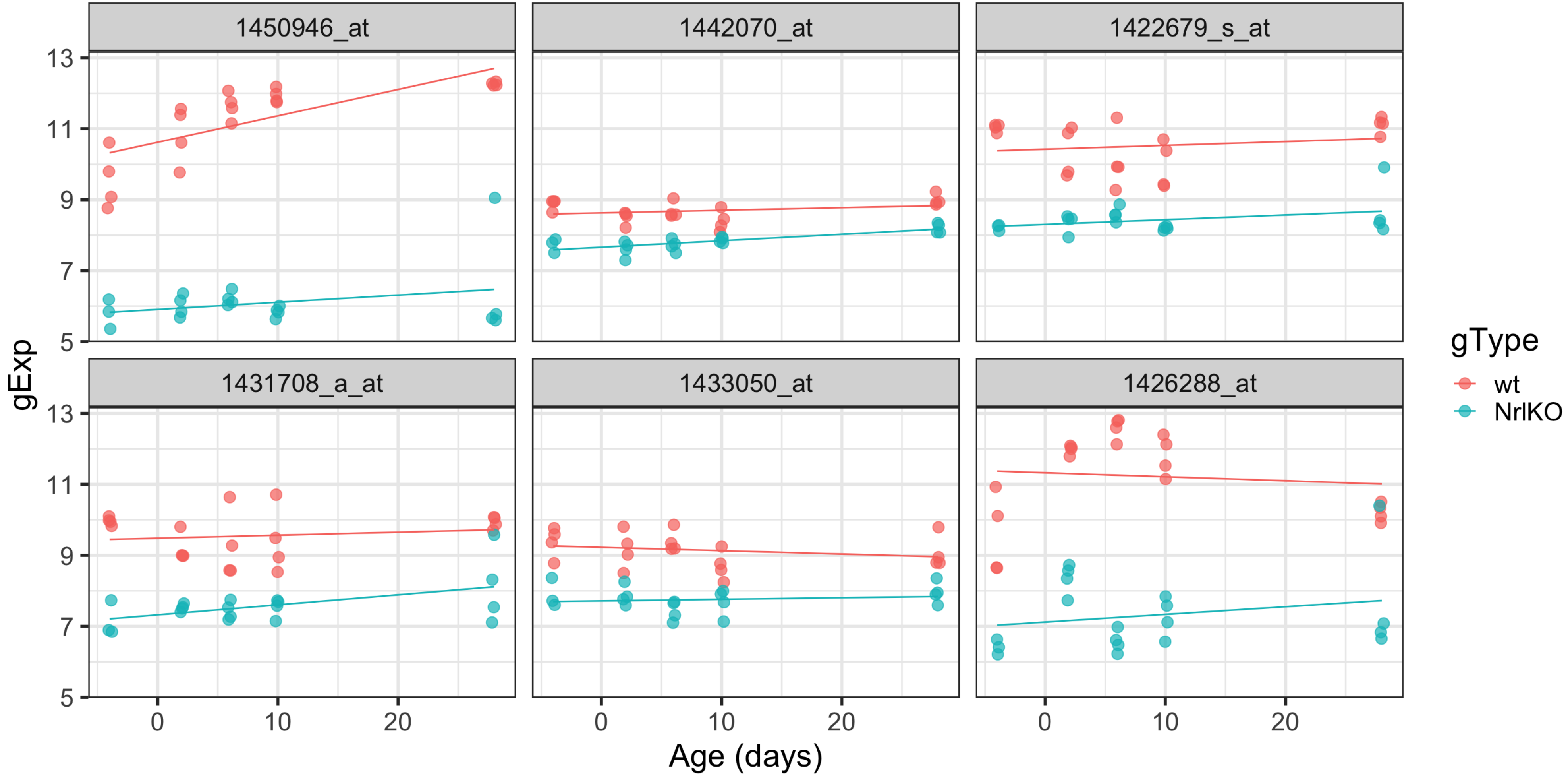
`topTable` in action: `gTypeNrlKO`

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

```
##                   logFC  AveExpr          t      P.Value    adj.P.Val        B
## 1450946_at   -4.7126638 8.733000 -15.613705 4.272742e-18 1.279644e-13 28.99515
## 1442070_at   -0.9670180 8.268231  -9.641739 9.567404e-12 1.432671e-07 16.29667
## 1422679_s_at -2.1179608 9.495128  -9.012182 5.793741e-11 4.813847e-07 14.65119
## 1431708_a_at -2.1615366 8.591436  -8.973296 6.486068e-11 4.813847e-07 14.54763
## 1433050_at   -1.5084389 8.468974  -8.899625 8.036741e-11 4.813847e-07 14.35083
## 1426288_at   -4.2101933 9.324051  -8.795830 1.088271e-10 5.432105e-07 14.07222
## 1418108_at    0.8661764 8.260641   8.415346 3.343261e-10 1.430390e-06 13.03789
## 1437528_x_at -2.4669479 8.721462  -8.110360 8.320671e-10 2.840353e-06 12.19451
## 1450770_at    1.3104718 7.357026   8.101880 8.535569e-10 2.840353e-06 12.17088
## 1457802_at   -0.8360751 7.778179  -8.004898 1.143227e-09 3.401136e-06 11.90004
```

- `topTable(ebFit, coef = 2)` is equivalent here, but much less informative!!
- this finds genes where the knockouts differ from the wild types *when age is zero*

Plotting the top 6 genes for `gTypeNrlKO`



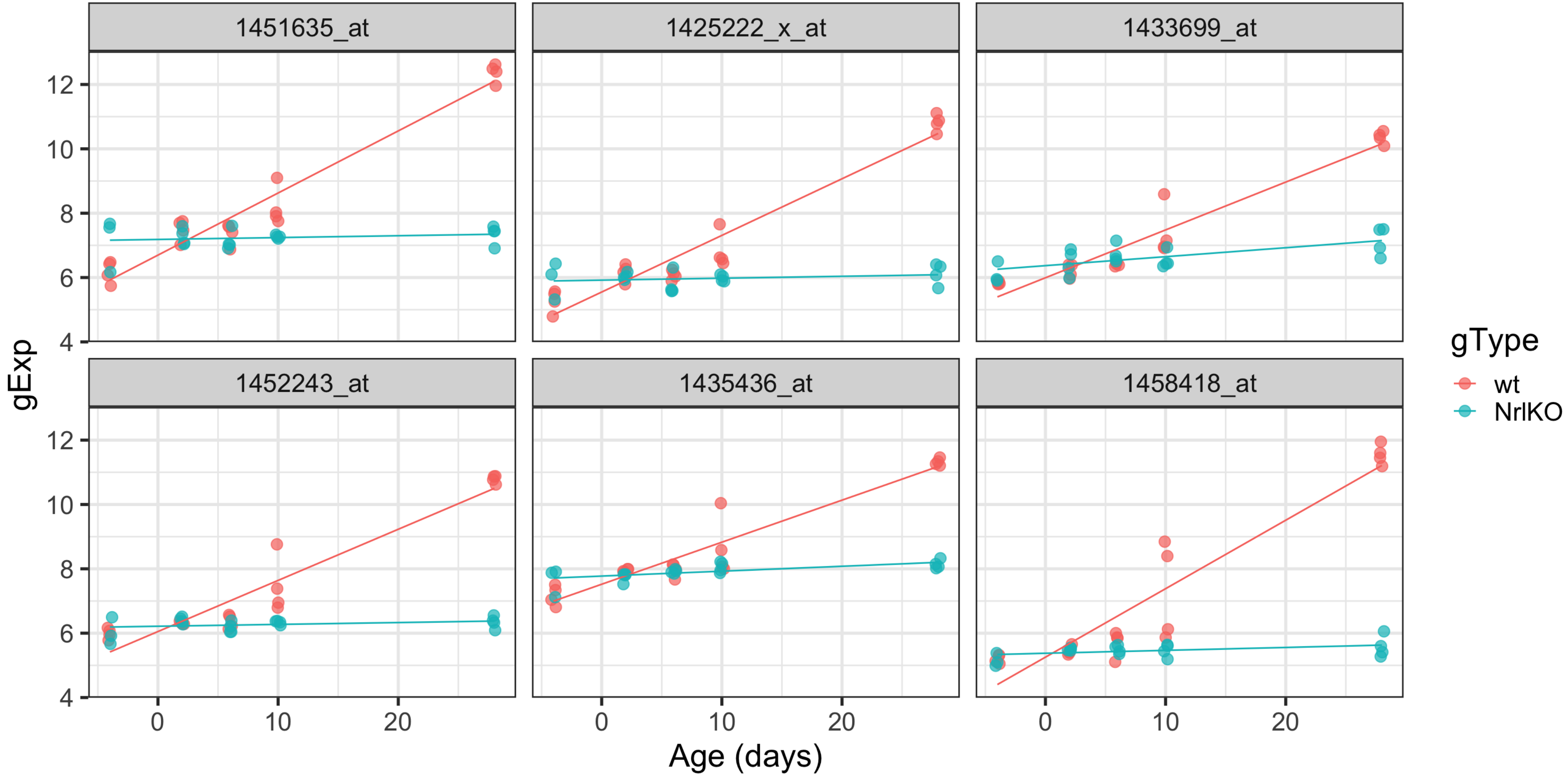
topTable in action: Age

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

```
##                    logFC  AveExpr        t      P.Value    adj.P.Val        B
## 1451635_at   0.19298927 7.791487 20.75441 2.683063e-22 8.035505e-18 40.76356
## 1425222_x_at 0.17598212 6.514359 19.68138 1.698679e-21 2.543687e-17 38.93935
## 1433699_at   0.14869533 6.939821 17.75194 5.837968e-20 4.643826e-16 35.42839
## 1452243_at   0.15891962 6.841103 17.72029 6.202312e-20 4.643826e-16 35.36815
## 1435436_at   0.13064850 8.274615 17.53000 8.942097e-20 5.356138e-16 35.00400
## 1458418_at   0.21279155 6.270282 16.45559 7.492503e-19 3.739883e-15 32.88504
## 1424977_at   0.07815514 6.505462 16.09759 1.557777e-18 6.664837e-15 32.15426
## 1431174_at   0.16155773 7.417872 16.01078 1.863752e-18 6.977188e-15 31.97514
## 1421818_at   0.11247565 7.982205 15.58235 4.565062e-18 1.519101e-14 31.07984
## 1419069_at   0.09324302 8.222051 15.40349 6.671365e-18 1.998007e-14 30.70045
```

- `topTable(ebFit, coef = 3)` is equivalent here, but much less informative!!
- this finds genes where Age significantly affects gene expression *for wt*

Plotting the top 6 genes for `Age`



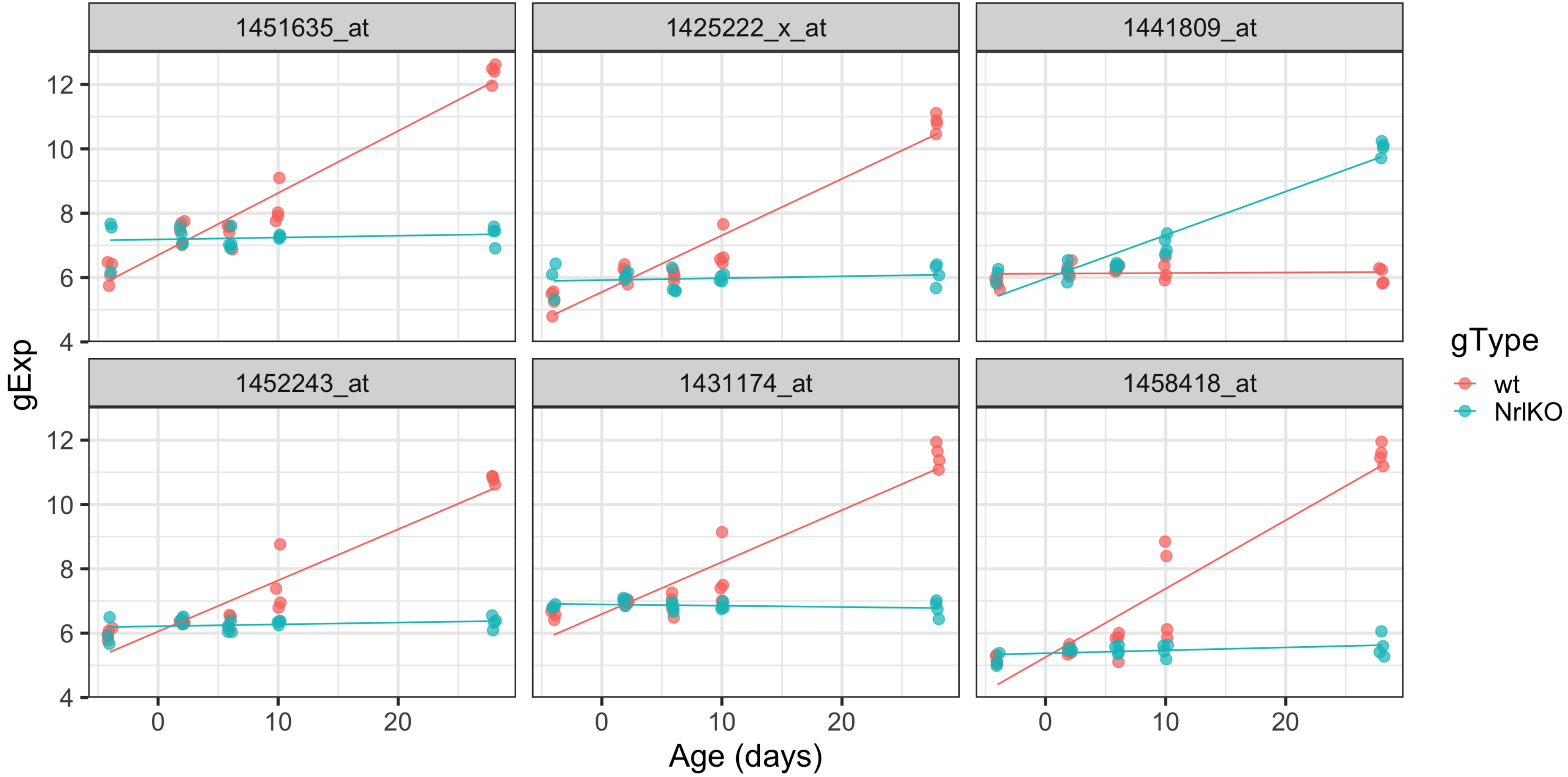
`topTable` in action: `gTypeNrlKO:Age`

```r
topTable(ebFit, coef = "gTypeNrlKO:Age")
```

```
##                   logFC  AveExpr         t      P.Value    adj.P.Val        B
## 1451635_at   -0.1872290 7.791487 -13.98122 1.534659e-16 4.596151e-12 27.60530
## 1425222_x_at -0.1700389 6.514359 -13.20474 9.339083e-16 1.398481e-11 25.82726
## 1441809_at    0.1336928 6.649615  12.49900 5.125762e-15 5.117048e-11 24.14656
## 1452243_at   -0.1531601 6.841103 -11.85861 2.532480e-14 1.896131e-10 22.56637
## 1431174_at   -0.1655647 7.417872 -11.39324 8.350958e-14 5.002057e-10 21.38431
## 1458418_at   -0.2037518 6.270282 -10.94096 2.734525e-13 1.364938e-09 20.20787
## 1435436_at   -0.1154239 8.274615 -10.75394 4.500300e-13 1.925421e-09 19.71342
## 1416306_at    0.1700922 6.559769  10.40829 1.143742e-12 4.281740e-09 18.78717
## 1448602_at    0.0797366 7.599205  10.23151 1.854062e-12 5.831382e-09 18.30724
## 1451617_at   -0.2233672 7.419538 -10.20200 2.010541e-12 5.831382e-09 18.22673
```

- `topTable(ebFit, coef = 4)` is equivalent here, but much less informative!!
- this finds genes where the effect of Age is significantly different in each genotype

Plotting the top 6 genes for `gTypeNrlKO:Age`



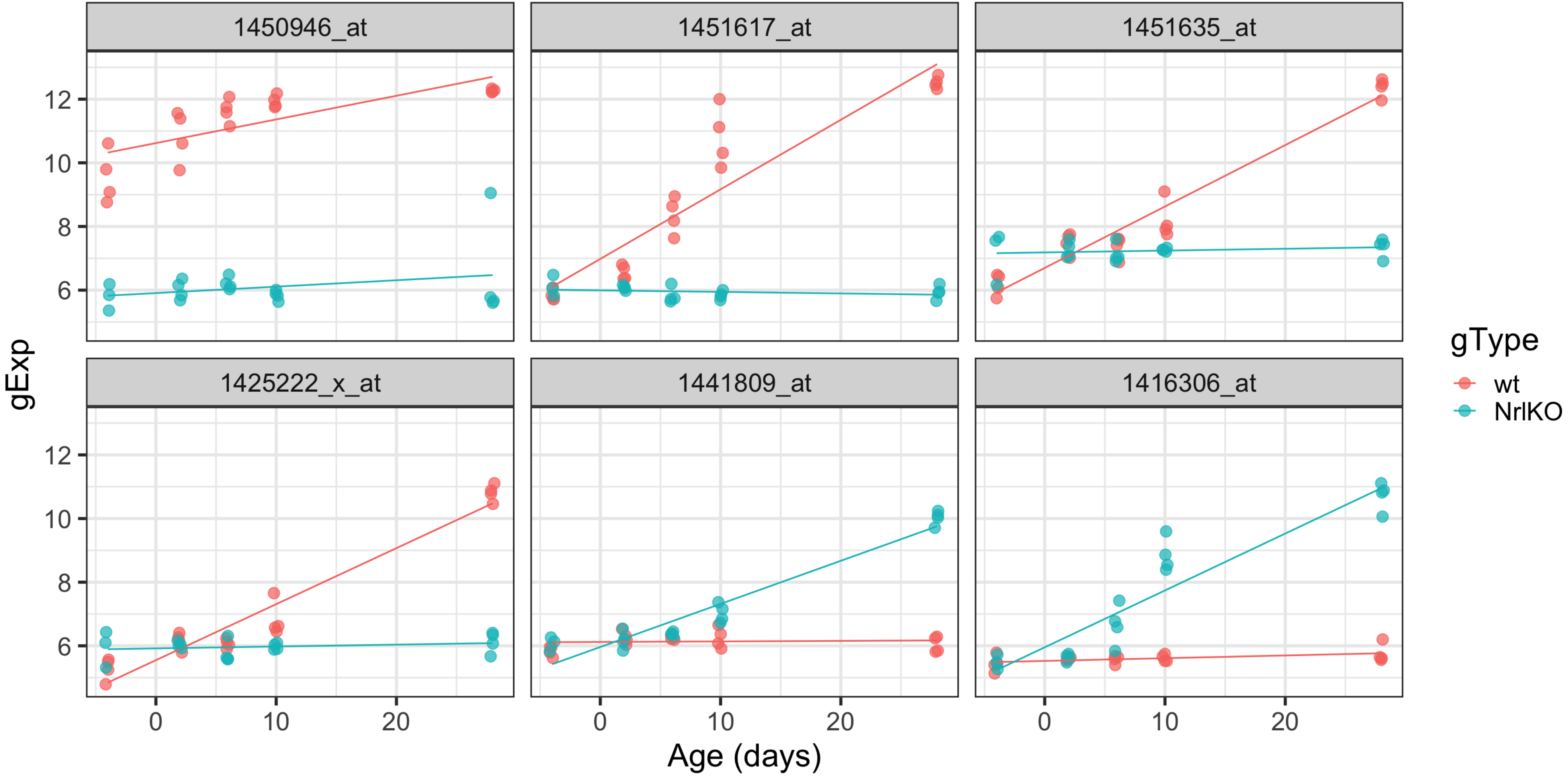
`topTable` in action: any effect of genotype

```r
topTable(ebFit, coef = c("gTypeNrlKO", "gTypeNrlKO:Age"))[,-3]
```

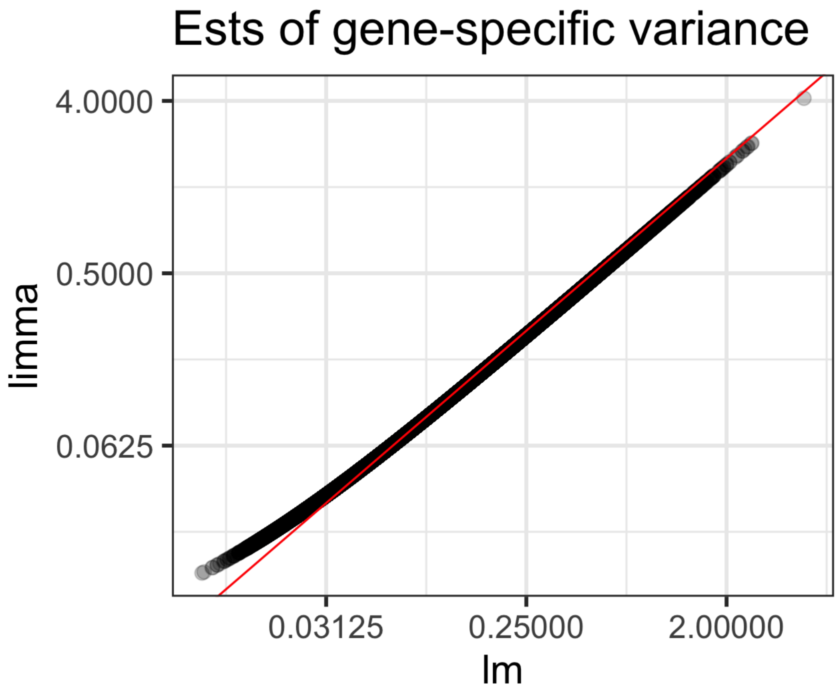
```
##              gTypeNrlKO gTypeNrlKO.Age        F      P.Value    adj.P.Val
## 1450946_at   -4.7126638    -0.05417495 247.5942 1.742207e-22 5.217737e-18
## 1451617_at   -0.9882116    -0.22336722 129.5363 1.138568e-17 1.159891e-13
## 1451635_at    0.4869960    -0.18722902 129.3777 1.161866e-17 1.159891e-13
## 1425222_x_at  0.3698245    -0.17003891 119.4619 4.311475e-17 3.228109e-13
## 1441809_at   -0.1579619     0.13369281 116.4515 6.540545e-17 3.917656e-13
## 1416306_at    0.4283527     0.17009218 113.1603 1.042657e-16 5.204423e-13
## 1449526_a_at  0.9935188     0.04233017 108.7521 1.983646e-16 8.486887e-13
## 1452243_at    0.1639021    -0.15316014 105.8493 3.066949e-16 1.148151e-12
## 1451763_at   -0.6821021    -0.14008860 102.6096 5.048816e-16 1.593666e-12
## 1430128_a_at -0.6669079    -0.12776553 102.2729 5.321267e-16 1.593666e-12
```

- `topTable(ebFit, coef = c(2,4))` is equivalent here, but much less informative!!
- this finds genes where any (additive/interaction) effect of genotype is significant

Plotting the top 6 genes for any effect of genotype

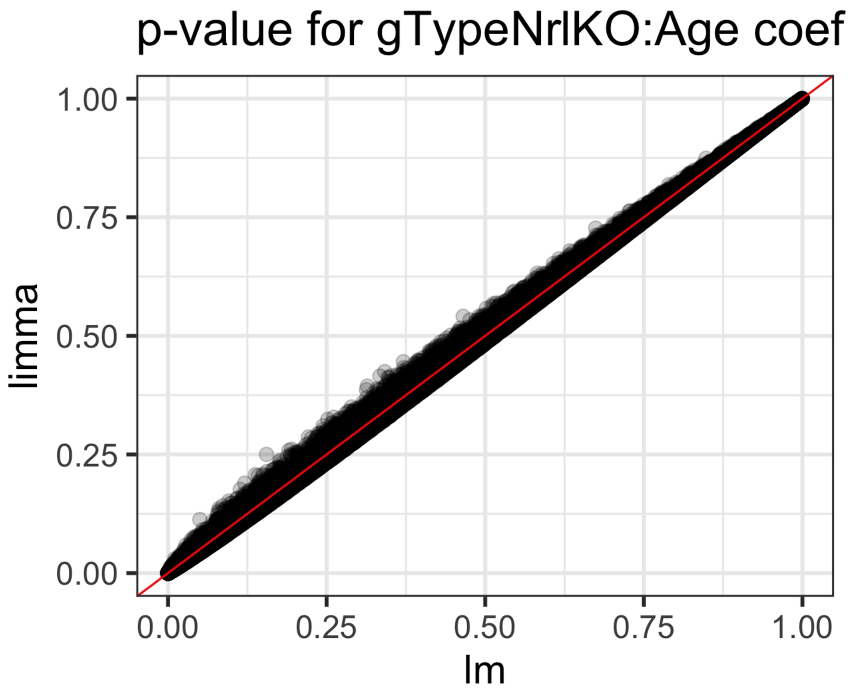


Comparison of s_g^2 and $\tilde{s}_g^2$ (shrinkage!)



- For small variances, limma *increases* the estimates
- For large variances, limma *decreases* the estimates

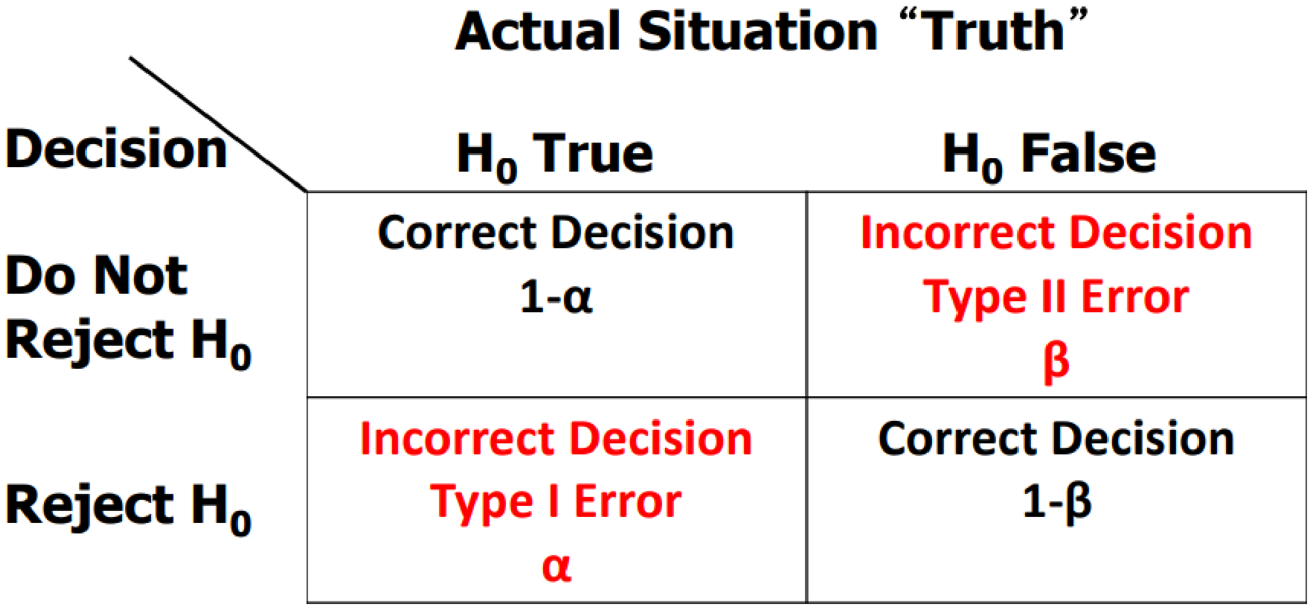
Comparison of interaction coefficient p-values



- 12261 genes where limma p-value is *larger* than lm
- 17688 genes where limma p-value is *smaller* than lm

Multiple testing

Error rates

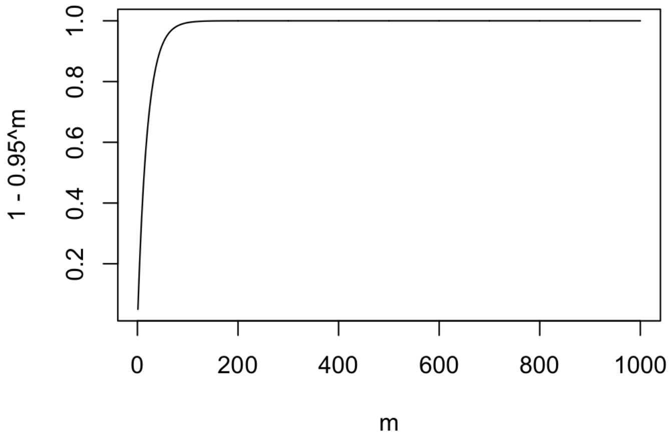


	Actual Situation “Truth”	
Decision	H_0 True	H_0 False
Do Not Reject H_0	Correct Decision 1-α	Incorrect Decision Type II Error β
Reject H_0	Incorrect Decision Type I Error α	Correct Decision 1-β

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

Type I Error rate for m tests

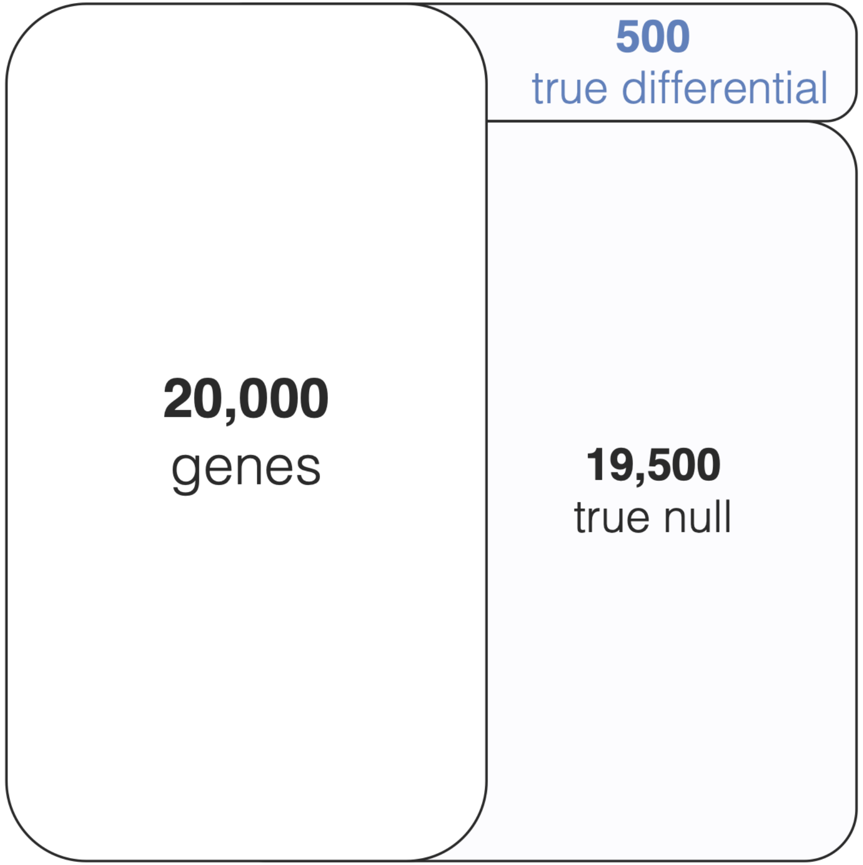
- $P(\text{incorrect decision}|H_0) = \alpha; \text{let } \alpha = 0.05$
- $P(\text{correct decision}|H_0) = 1 - \alpha = 0.95$
- $P(\text{correct decision on } m \text{ tests}|H_0) = (1-\alpha)^m = 0.95^m$
- $P(\text{at least one incorrect decision on } m \text{ tests}|H_0) = 1 - (1-\alpha)^m = 1 -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



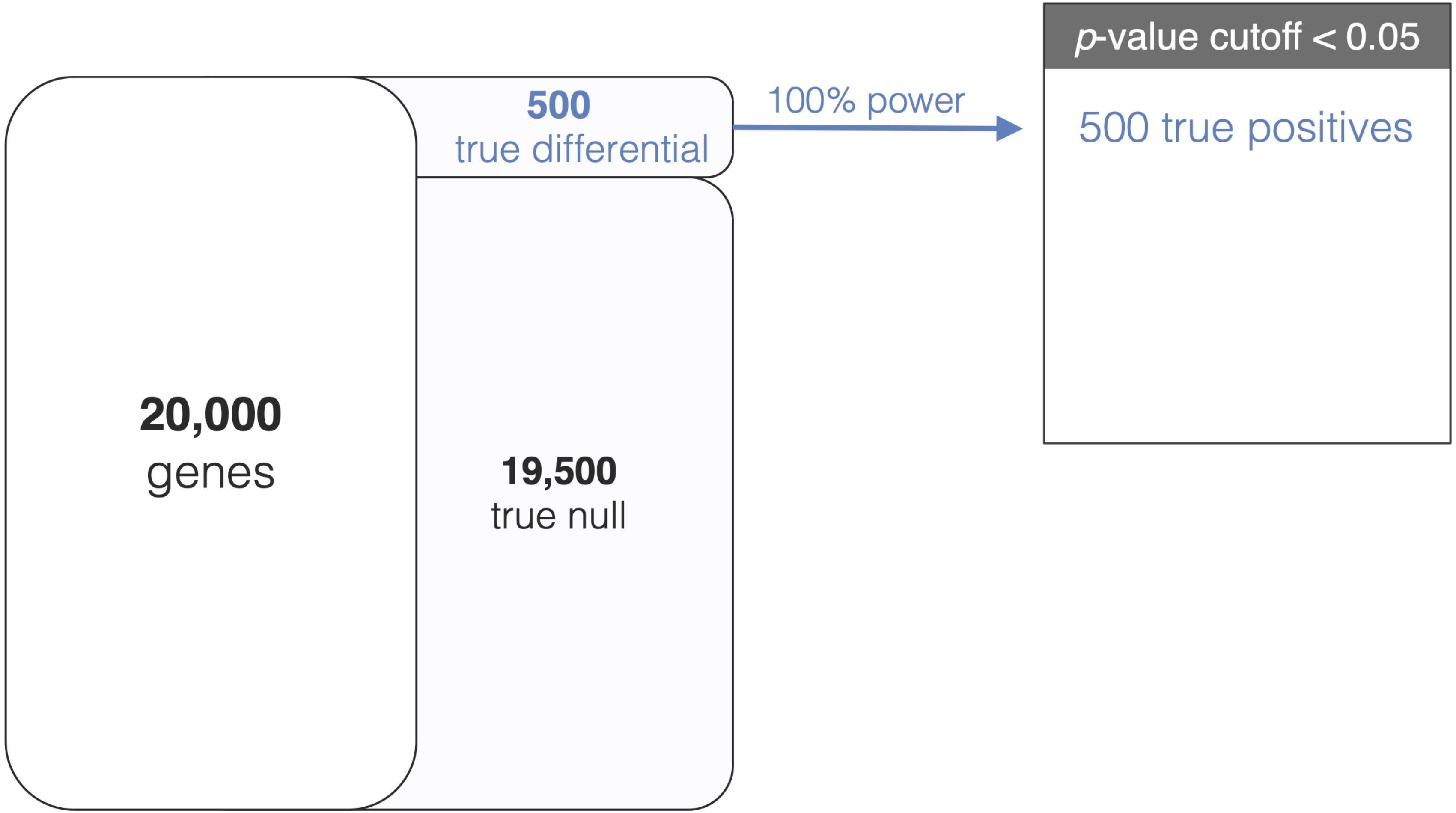
Multiple comparisons and error rates

20,000
genes

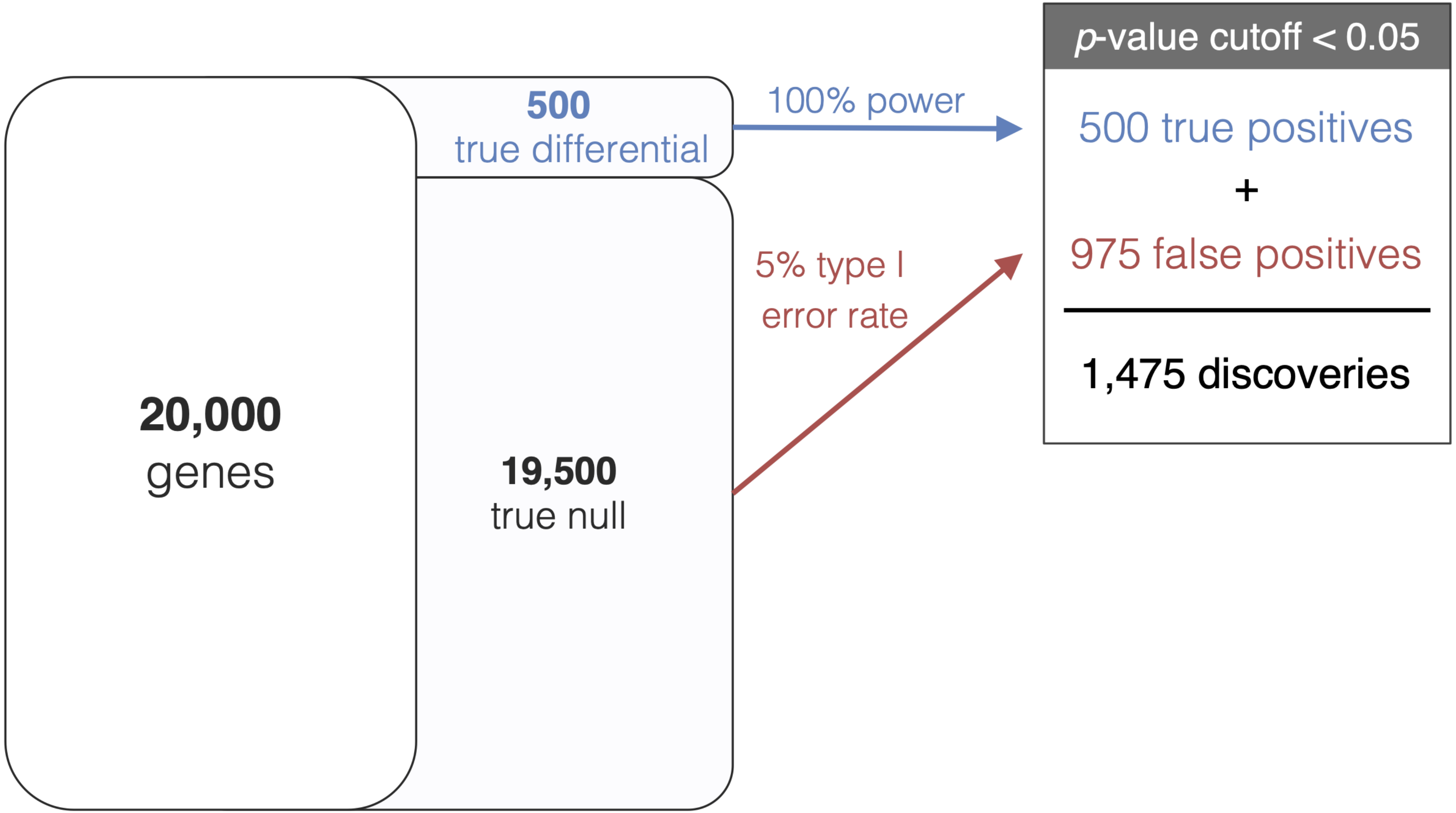
Multiple comparisons and error rates



Multiple comparisons and error rates



Multiple comparisons and error rates



p-value cutoff < 0.05

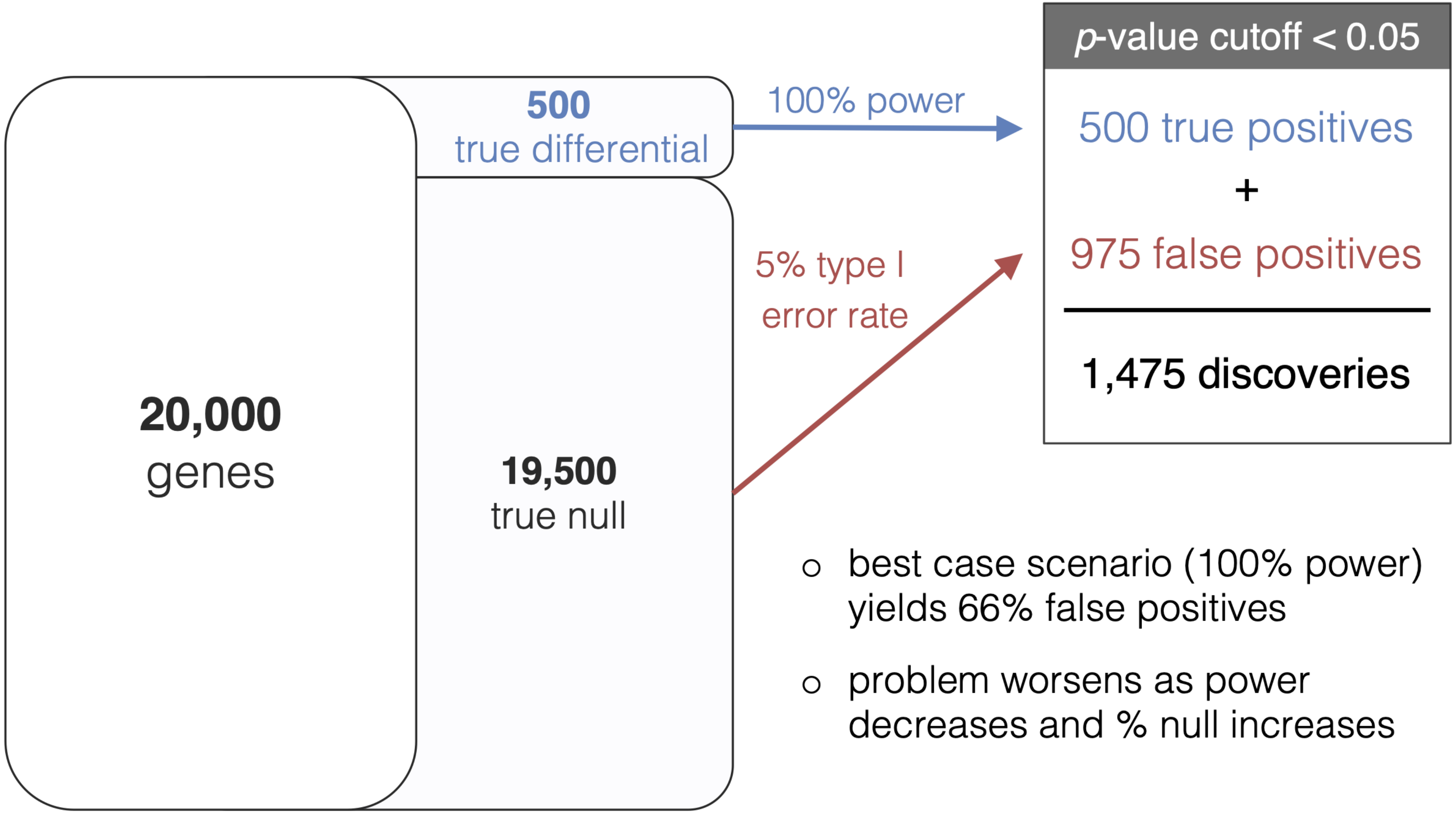
500 true positives

+

975 false positives

1,475 discoveries

Multiple comparisons and error rates



- 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:**

$$p_i^{Bon} = mp_i \text{ (more technically correct: } p_i^{Bon} = min(mp_i, 1))$$

$$\text{Then, threshold } p_i^{Bon} \text{ at } \alpha$$

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

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

$$\text{Then, threshold } p_i \text{ at } \alpha^{Bon}$$

Multiple test correction is an active area of statistical research

- 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:

$$p_1^H = mp_1$$

$$p_2^H = (m-1)p_2$$

$$p_3^H = (m-2)p_3$$

$$\vdots$$

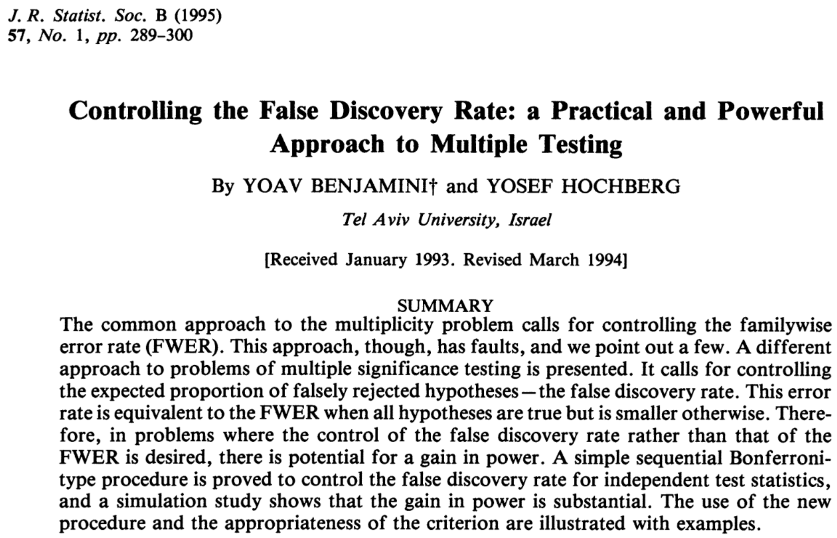
$$\Rightarrow FWER \leq \alpha$$

How practical is the FWER in high-throughput biology?

- Why do we care so much about making one error??
- One extreme 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 mistakes, 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]

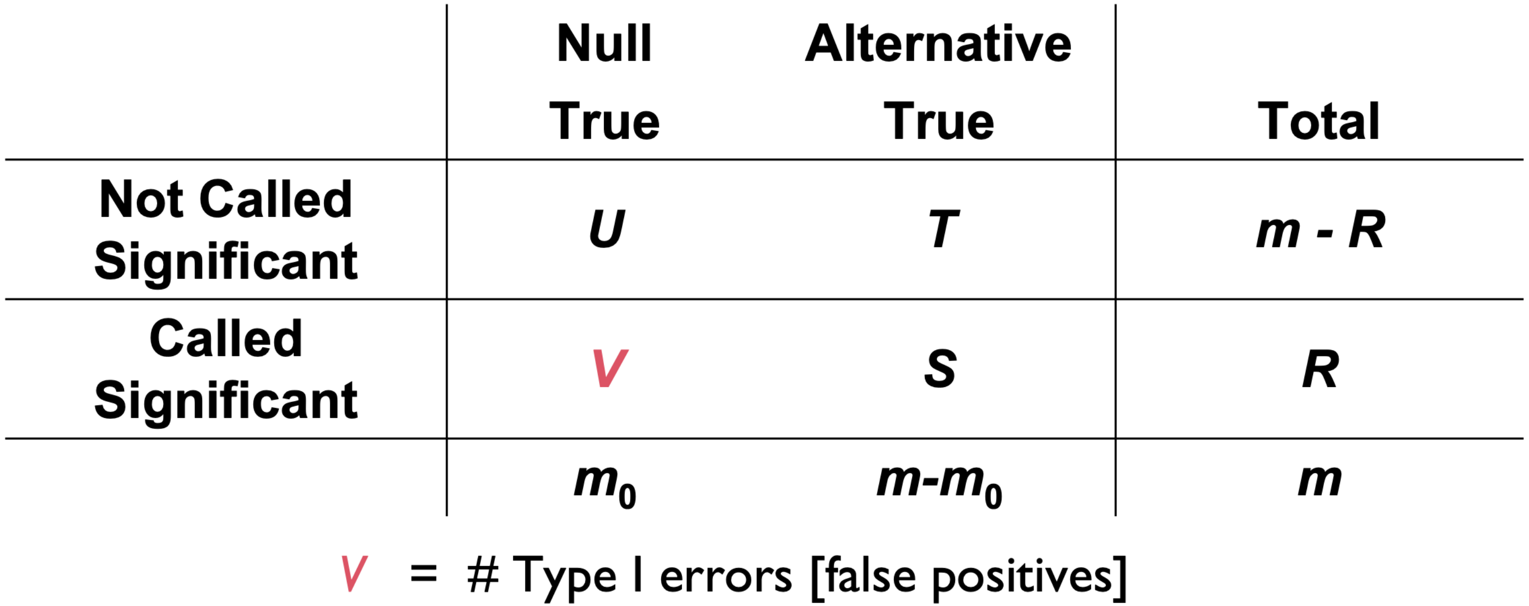
SUMMARY

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 60K citations!!

False Discovery Rate

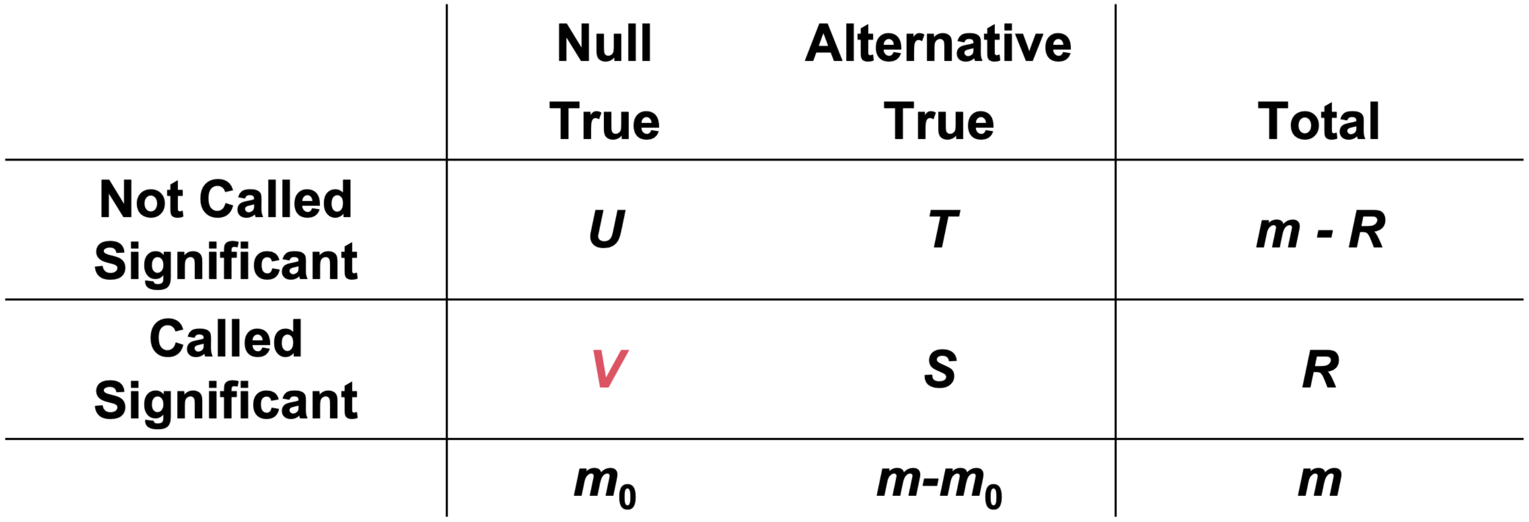


	Null True	Alternative True	Total
Not Called Significant	*U*	*T*	*m - R*
Called Significant	*V*	*S*	*R*
	m_0	$m-m_0$	*m*

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]

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

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 significant features (m_0) are truly null

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

- 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

- 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-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

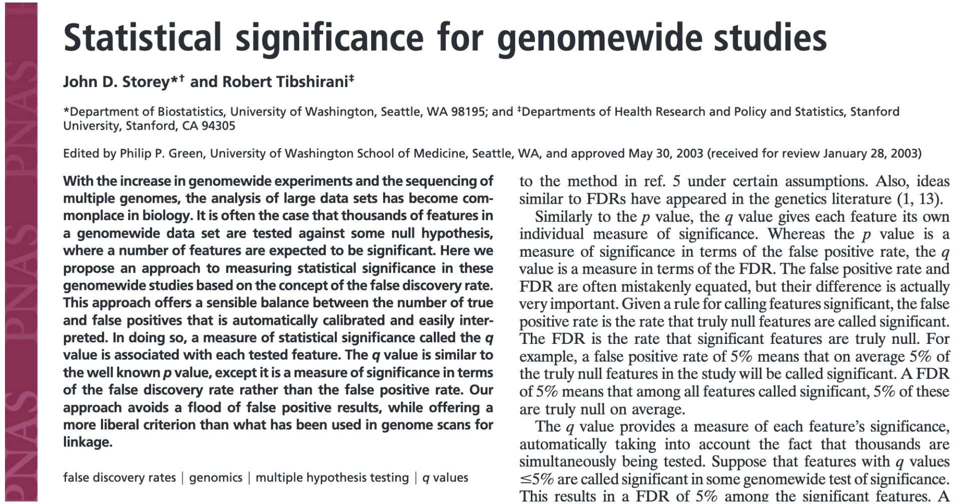
```r
topTable(ebFit, coef = "gTypeNrlKO")
```

```
##                   logFC  AveExpr          t      P.Value    adj.P.Val        B
## 1450946_at   -4.7126638 8.733000 -15.613705 4.272742e-18 1.279644e-13 28.99515
## 1442070_at   -0.9670180 8.268231  -9.641739 9.567404e-12 1.432671e-07 16.29667
## 1422679_s_at -2.1179608 9.495128  -9.012182 5.793741e-11 4.813847e-07 14.65119
## 1431708_a_at -2.1615366 8.591436  -8.973296 6.486068e-11 4.813847e-07 14.54763
## 1433050_at   -1.5084389 8.468974  -8.899625 8.036741e-11 4.813847e-07 14.35083
## 1426288_at   -4.2101933 9.324051  -8.795830 1.088271e-10 5.432105e-07 14.07222
## 1418108_at    0.8661764 8.260641   8.415346 3.343261e-10 1.430390e-06 13.03789
## 1437528_x_at -2.4669479 8.721462  -8.110360 8.320671e-10 2.840353e-06 12.19451
## 1450770_at    1.3104718 7.357026   8.101880 8.535569e-10 2.840353e-06 12.17088
## 1457802_at   -0.8360751 7.778179  -8.004898 1.143227e-09 3.401136e-06 11.90004
```

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



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 $\leq$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 eaverage)

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 proportion of tests that are truly null
- q-value:

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

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

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

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

BH vs q-value in our example

Rank (j)	**P-value**	$\hat{p}^{BH}$	$\hat{q}\left(p_i\right)$
1	0.0008	0.008	0.008
2	0.009	0.045	0.045
3	0.127	0.423	0.423
4	0.205	0.513	0.513
5	0.396	0.792	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	1.000	0.993
10	0.993	0.993	0.993

Compounding issues of multiple comparisons

- What if you're not only testing 30K 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)

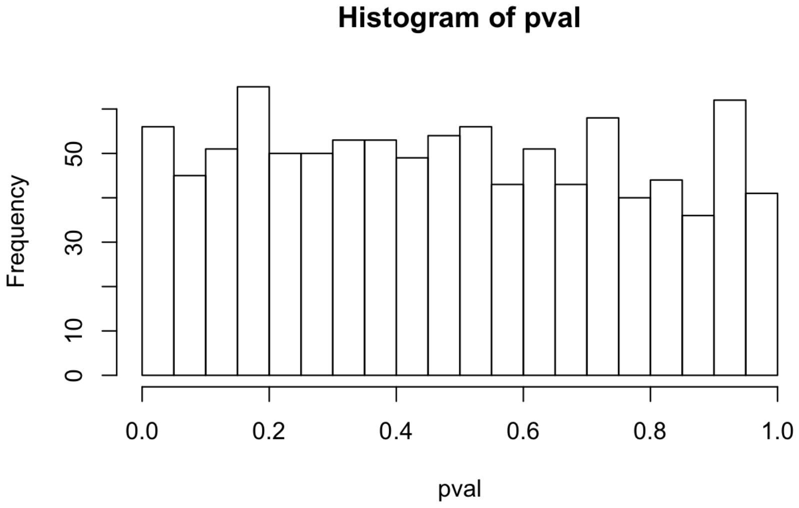
  - p-values are combined, adjusted globally, then separated back out and sorted

Assumptions about p-values

- Implicit assumption for all multiple testing correction methods:

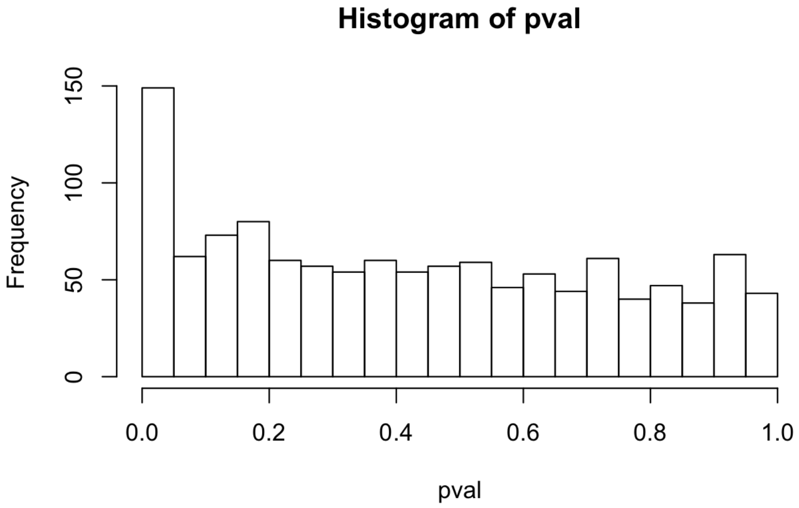
p-value distribution is "well-behaved"

- What does this mean?
  - primarily, that the distribution of p-values under the null is **uniform**



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)

- Solution: compute p-values "empirically" using resampling/permutation/bootstrap techniques

- **Bootstrap**: take repeated random samples with replacement from your data and compute statistic; repeat many times and use bootstrap statistics as your sampling distribution rather than a t, Normal, F, χ^2, etc

- **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

- Downside: often computationally intensive for genomics