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

Lecture 10 – Linear Models Part IV

Amrit Singh

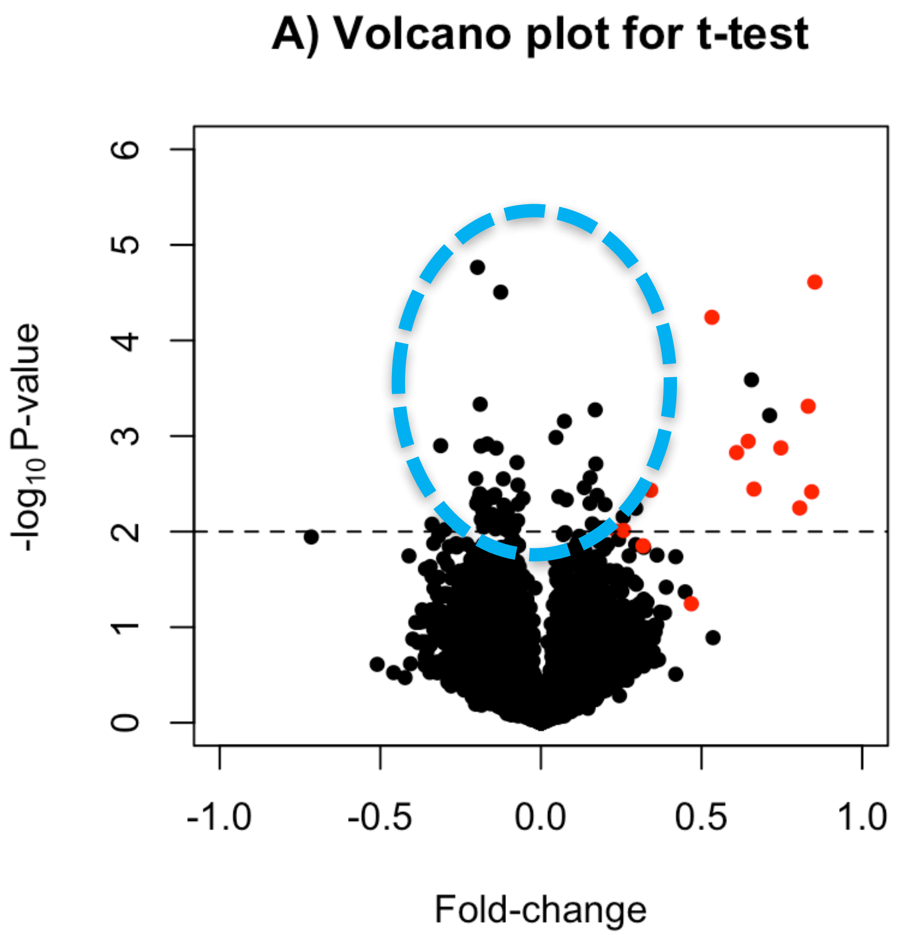
Feb 06 2016

Based on slides by Sara Mostafavi & Dr. Jennifer Bryan

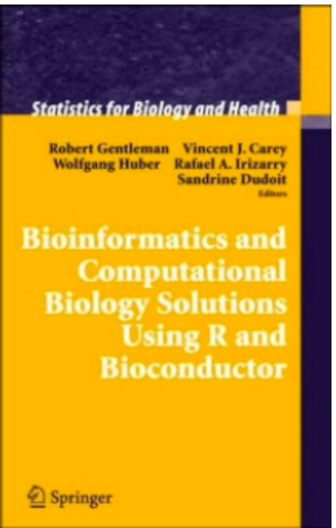
Motivation – data("spikein95")

spikein95

- 2 groups of 3 samples each
- groups differ w/r to concentration of 16 probesets



1470 differentially expressed genes!! – **majority have very small variances**


Statistics for Biology and Health
Robert Gentleman Vincent J. Carey
Wolfgang Huber Rafael A. Irizarry
Sandrine Dudoit
Editors
Bioinformatics and Computational Biology Solutions Using R and Bioconductor
Springer

outline

- Review
  - Linear regression framework
- Large scale differential expression analysis:
  - Assessing ALL genes (in a univariate way)
    - i.e., same model, except run it >20K times
  - Empirical Bayes → moderated test statistic
  - running Limma in R

$$Y = X\alpha + \varepsilon$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \begin{bmatrix} \alpha_0 \cdot 1 + \alpha_1 \cdot x_1 \\ \alpha_0 \cdot 1 + \alpha_1 \cdot x_2 \\ \vdots \\ \alpha_0 \cdot 1 + \alpha_1 \cdot x_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} = \begin{bmatrix} \alpha_0 + \alpha_1 x_1 + \varepsilon_1 \\ \alpha_0 + \alpha_1 x_2 + \varepsilon_2 \\ \vdots \\ \alpha_0 + \alpha_1 x_n + \varepsilon_n \end{bmatrix}$$

$$y_i = \alpha_0 + \alpha_1 x_i + \varepsilon_i$$

Here we are just fitting a line but using matrix notation to handle all *n* observations at once, more elegantly.
Big pay-offs ensue .....

Industrial scale model fitting is good because things like this are not recomputed 30K times unnecessarily*

$Y = X\alpha + \varepsilon$ regression model

$\hat{\alpha} = (X^T X)^{-1} X^T Y$ the MLE and OLS estimator of α

$\hat{\sigma}^2 = \frac{1}{n-p} \hat{\varepsilon}^T \hat{\varepsilon}$ the estimated error variance

$\hat{V}(\hat{\alpha}) = \hat{\sigma}^2 (X^T X)^{-1}$ the estimated covariance matrix of $\hat{\alpha}$

How test $H_0 : \alpha_j = 0$?

With a t-statistic. Under H_0, we have (at least approximately) that:

$$\frac{\hat{\alpha}_j}{\widehat{se}(\hat{\alpha}_j)} \sim t_{n-p}$$

so a p-value is obtained by computing a tail probability for the observed value of $\hat{\alpha}_j$ from a t_{n-p} distribution.

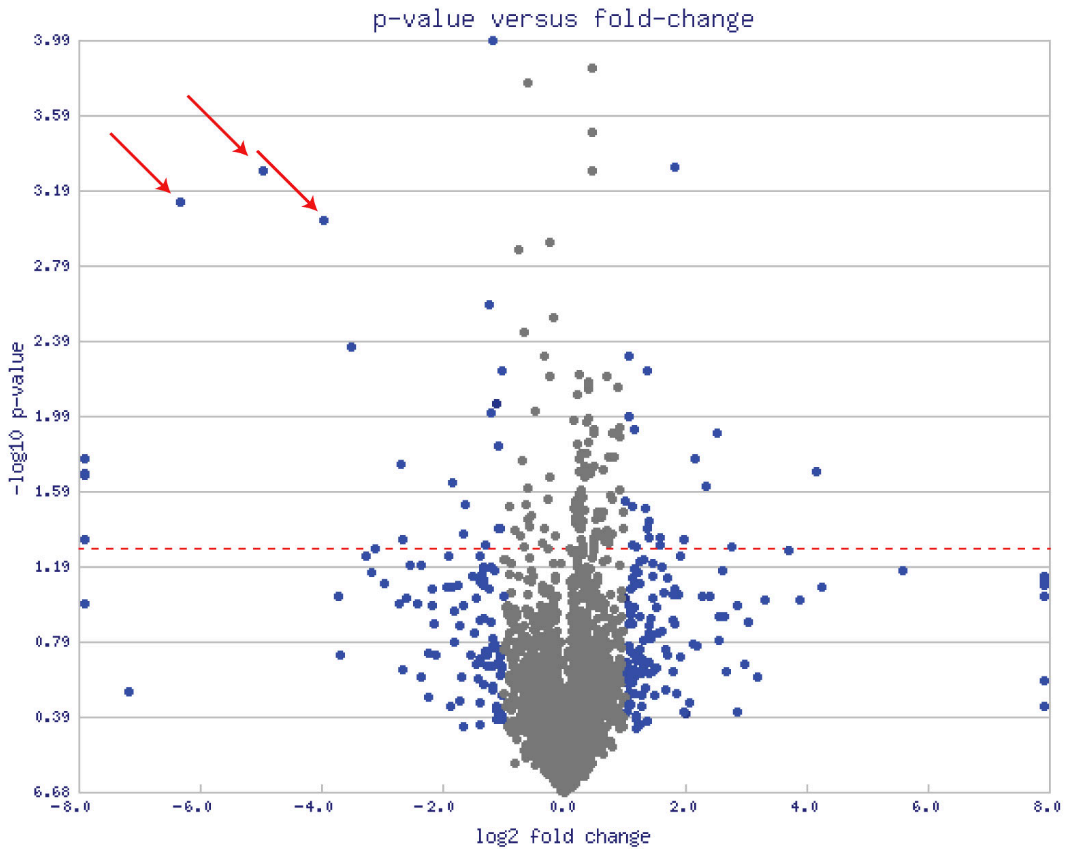
* under the hood, lm() is doing something more clever and numerically stable than this

Recurring theme in analysis of “high dimensional” biological data

Genes with very small p-values BUT subtle effects (small effect sizes)

- Replication rate typically lower for genes with subtle effects
- Ad hoc filters: require small pvalues and large effect sizes

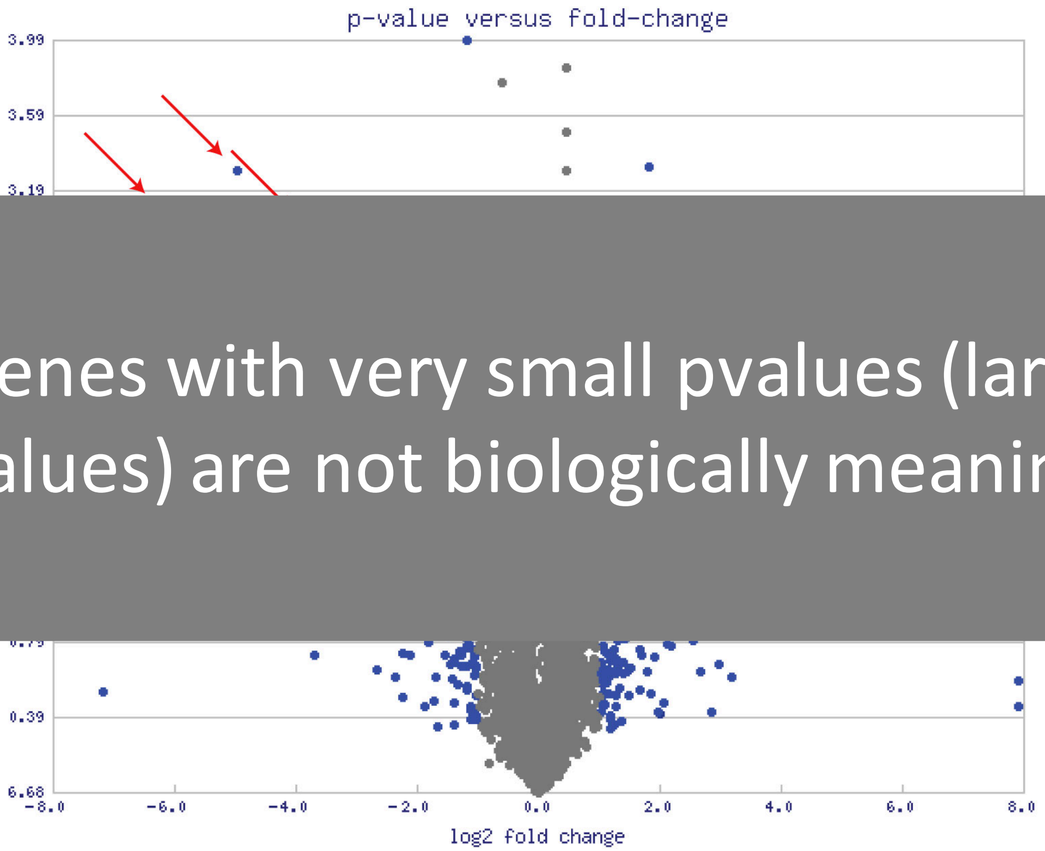
Observed (i.e., empirical) issues with the "standard" (i.e., t-test) approach for assessing differential expression



-log10 P-value

Log2 fold change (i.e., effect size)

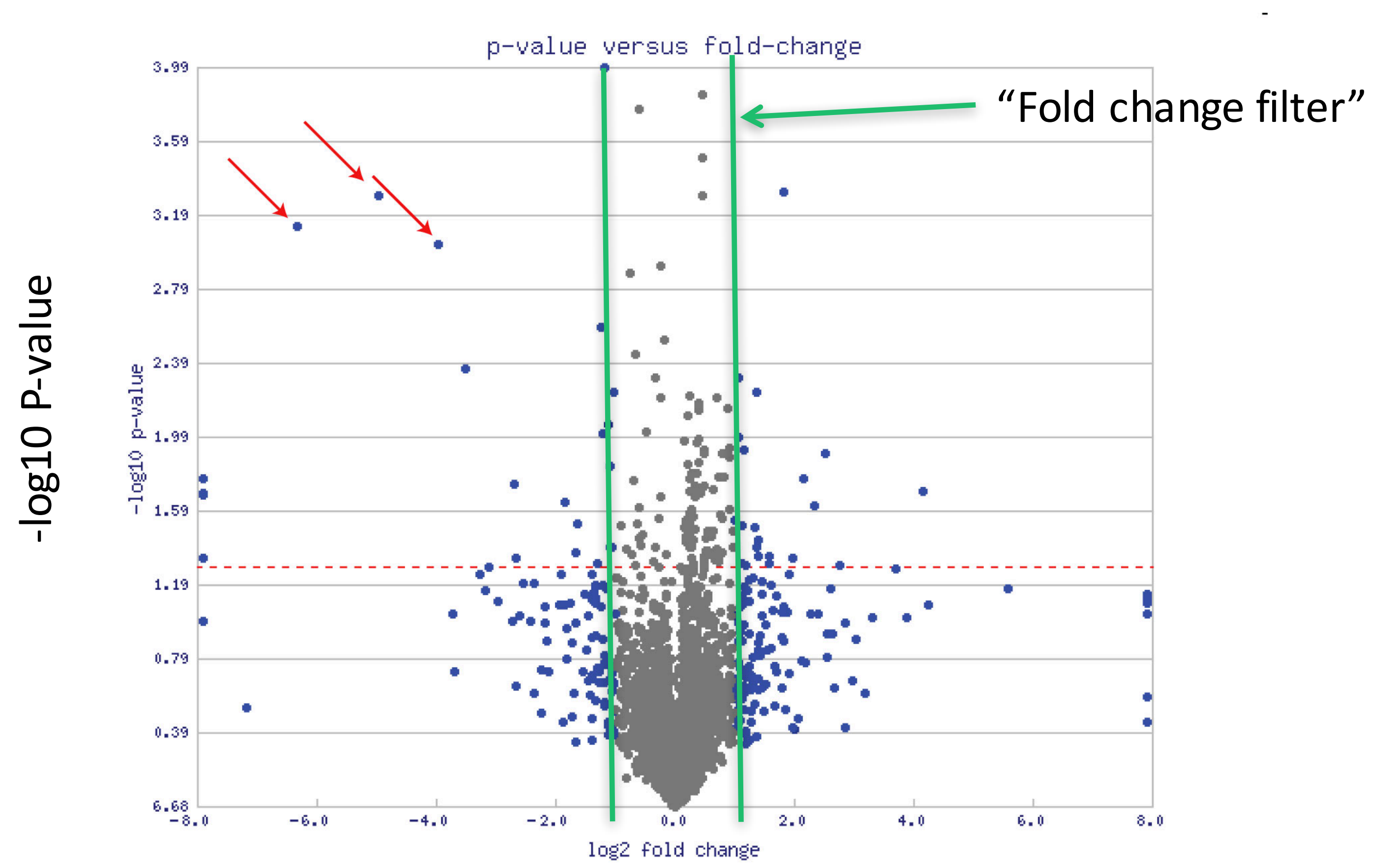
Observed (i.e., empirical) issues with the "standard" (i.e., t-test) approach for assessing differential expression



Some genes with very small pvalues (large –log10 pvalues) are not biologically meaningful.

Log2 fold change (i.e., effect size)

Observed (i.e., empirical) issues with the "standard" (i.e., t-test) approach for assessing differential expression



Log2 fold change (i.e., effect size)

How do we end up with small pvalues but subtle effects?

$$t_{gj} = \frac{\hat{\alpha}_{gj}}{SE(\hat{\alpha}_{gj})} = \frac{\hat{\alpha}_{gj}}{\underbrace{s_g}\sqrt{v_j}} \sim t_d \quad \text{under } \mathrm{H}_0$$

Small variance leads
to large t stat,
leading to small p

d=residual degree of
freedom

Let's review how we derive the test statistics from our linear model

$$Y_g = X_g \alpha_g + \varepsilon_g$$

the "g" in the subscript reminds us that we'll be fitting a model like this for each gene *g*

most of the time the design matrices X_g are, in fact, the same for all *g*; I'm going to just use *X*

also, let's record the residual degrees of freedom:

$$d_g = d = n - \text{dimension of } \alpha$$

$$Y_g = X\alpha_g + \varepsilon_g$$ the data model

$$\mathrm{var}(\varepsilon_g) = \sigma_g^2$$

$$s_g^2 = \frac{1}{n-p}\hat{\varepsilon}^T\hat{\varepsilon}$$ estimated error variance (p is the dimension of α)

$$\mathrm{var}(\hat{\alpha}_g) = (X^T X)^{-1} s_g^2 = V s_g^2$$

"unscaled covariance"

$$Y_g = X\alpha_g + \varepsilon_g$$

$$\begin{bmatrix} y_{g1} \\ y_{g2} \\ \vdots \\ \\ y_{gn_g} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{g1} \\ \mu_{g2} \\ \mu_{g3} \end{bmatrix} + \begin{bmatrix} \varepsilon_{g1} \\ \varepsilon_{g2} \\ \vdots \\ \\ \varepsilon_{gn_g} \end{bmatrix}$$

What would the estimated covariance matrix $\hat{V}(\hat{\alpha}_g) = s_g^2 (X^T X)^{-1} = V s_g^2$ actually look like in a concrete example?

$$\hat{V}(\hat{\alpha}_g) = \begin{bmatrix} \hat{V}(\hat{\mu}_1) & \widehat{\text{cov}}(\hat{\mu}_1, \hat{\mu}_2) & \widehat{\text{cov}}(\hat{\mu}_1, \hat{\mu}_3) \\ \widehat{\text{cov}}(\hat{\mu}_1, \hat{\mu}_2) & \hat{V}(\hat{\mu}_2) & \widehat{\text{cov}}(\hat{\mu}_2, \hat{\mu}_3) \\ \widehat{\text{cov}}(\hat{\mu}_2, \hat{\mu}_3) & \widehat{\text{cov}}(\hat{\mu}_2, \hat{\mu}_3) & \hat{V}(\hat{\mu}_3) \end{bmatrix}$$

So far, nothing new: the “regular” t statistics for gene *g* and parameters *j:*

$$t_{gj} = \frac{\widehat{\beta_{gj}}}{s_g \sqrt{v_j}} \sim t_d \text{ under } H_0$$

How do we end up with small p-values but subtle effects?

$$t_{gj} = \frac{\hat{\alpha}_{gj}}{SE(\hat{\alpha}_{gj})} = \frac{\hat{\alpha}_{gj}}{s_g\sqrt{v_j}} \sim t_d \quad \text{under } \mathrm{H}_0$$

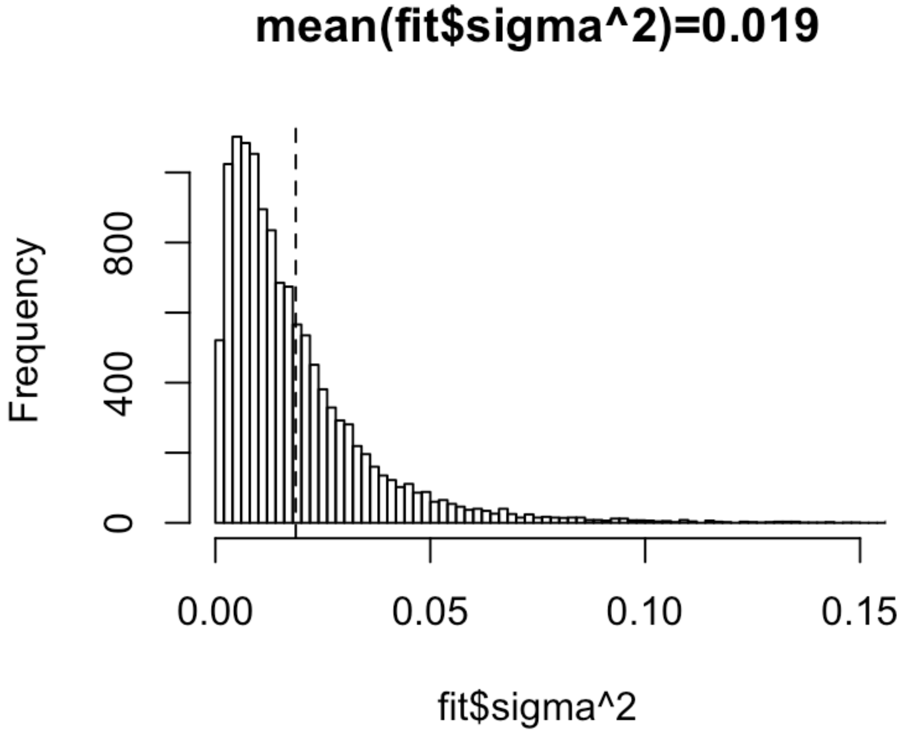
How would you modify the formula for *t* if you wanted to avoid having small p’s for genes with subtle effects?

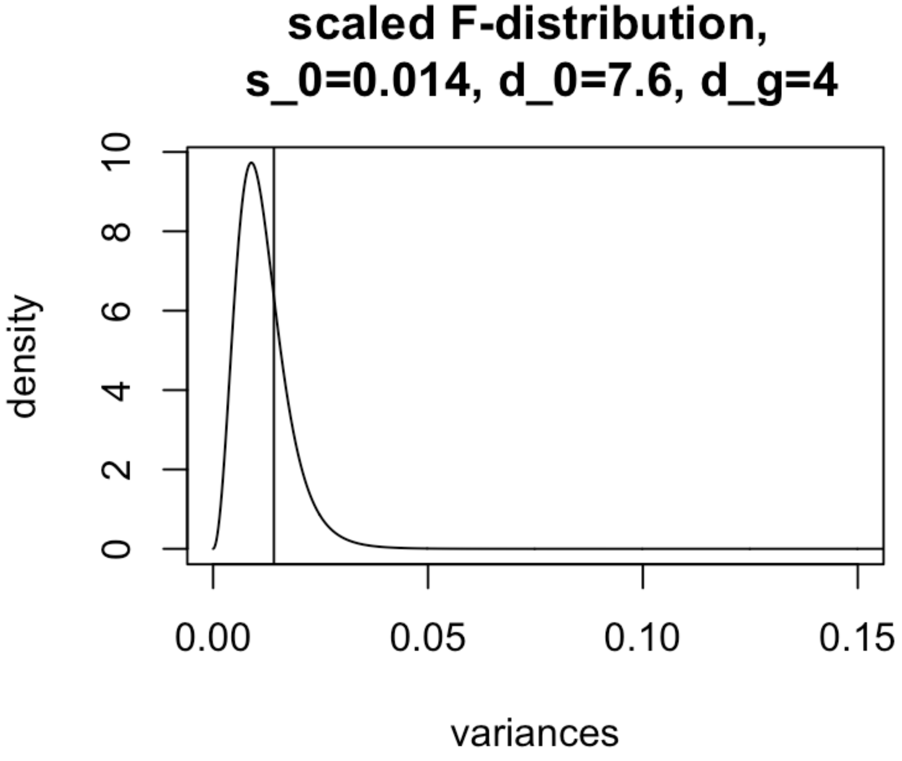
We could even come up with "weights *w*" so to take a weighted average between estimated var (s_g) and *prior* var (s_0).

$$t_{gj} = \frac{\hat{\alpha}_{gj}}{(w_0 s_0 + w_1 s_g)\sqrt{v_j}}$$

Moderated t statistics → Derived using the Empirical Bayes framework.

Modelling the distribution of all genewise variances





Fit scaled F-distribution $s^2 \sim s_0^2 F_{d_g, d_0}$

You are now entering a
Bayesian statistics
All persons proceeding past
this point must wear an approved
safety helmet, safety footwear
and high visibility jacket

Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes Theorem for a continuous random variable

Posterior distribution of parameters given data

Likelihood function of data given a set of parameters

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

Prior distribution of parameters

$$\textit{where}$$

Marginal distribution of data over all parameter values

$$P(x) = \int P(x|\theta)P(\theta)d\theta$$

Bayesian Hierarchical model

Prior distribution of gene variances

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

Hyperparameters (d_0, s_0^2)

Sampling distribution of sample variance for gene g

$$s_g^2 \mid \sigma_g^2 \sim \frac{\sigma_g^2}{d_g} \chi_{d_g}^2$$

σ_g^2 is a random variable

$$x_g^1, x_g^2, \ldots, x_g^n \overset{ind}{\sim} N(\mu_g, \sigma_g^2)$$

expression is normally distributed
- hierarchy placed on variances

Stat Appl Genet Mol Biol. 2004;3:Article3.

Estimate hyperparameters (d_0, s_0^2)

$$p(s^2 \mid d_0, s_0^2) = \int p(s^2 \mid \sigma^{-2}) p(\sigma^{-2}) d(\sigma^{-2})$$

Marginal distribution

$$\boxed{s^2 \sim s_0^2 F_{d_g, d_0}}$$

Sample variances follow a scaled F-distribution

Estimate s_0 and d_0 `?limma::fitFDist(x, df1)`

$$z_g = \log s_g^2$$

- z_g follows a Fisher's z-distribution
- hyperparameters are estimated by matching the theoretical mean and variance of the z-distribution to the observed sample mean and variance of z_g

Stat Appl Genet Mol Biol. 2004;3:Article3.

The distributional result assumes that the typical prior gene-wise error variance s_0^2 and its associated degrees of freedom d_0 are known, which of course they are not. In practice, they will be estimated from the data itself (which is what the term empirical Bayes refers to).

These are examples of hyperparameters. In a full blown Bayesian approach, the user would specify *a priori*.

Moderate genewise variances

$$\sigma_g^2 | s_g^2 \sim \frac{d_0 s_0^2 + d_g s_g^2}{\chi^2_{d_0+d_g}}$$

Posterior distribution

$$\boxed{\tilde{s}_g^2 = s^2_{g[\text{moderated}]} = \frac{d_g s_g^2 + d_0 s_0^2}{d_g + d_0}} = \lambda s_g^2 + (1-\lambda) s_0^2$$

$$\text{with } \lambda = \frac{d_g}{d_g + d_0} \in (0,1)$$

```
?limma::squeezeVar(var, df1)
```

EuPA Open Proteomics, 2015 (7):11-19
Ann. Appl. Stat., Volume 10, Number 2 (2016), 946-963

The posterior mean for gene-specific variance:

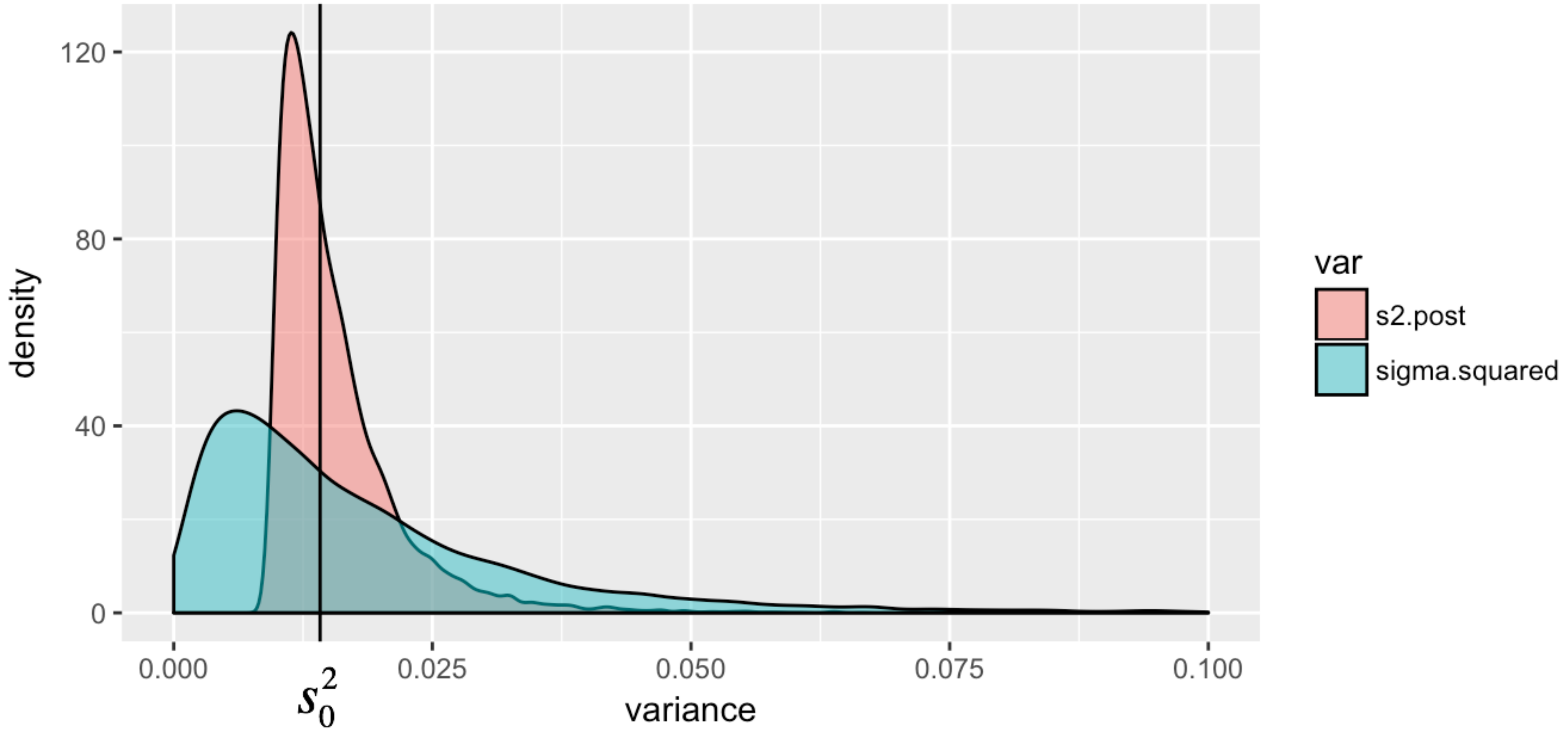
$$\tilde{s}_g^2 = \frac{d_0 s_0^2 + d_g s_g^2}{d_0 + d_g}$$

How to think about it:
a weighted mean of the prior (indirect evidence) and the observed (direct evidence) gene-specific variances

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

More how to think about it:
"shrinking" the observed gene-specific variance towards the "typical" variance implied by the prior

Shrink variances – spikein95 data



`s2.prior`	estimated prior value for `sigma^2`
`s2.post`	numeric vector giving the posterior values for `sigma^2`

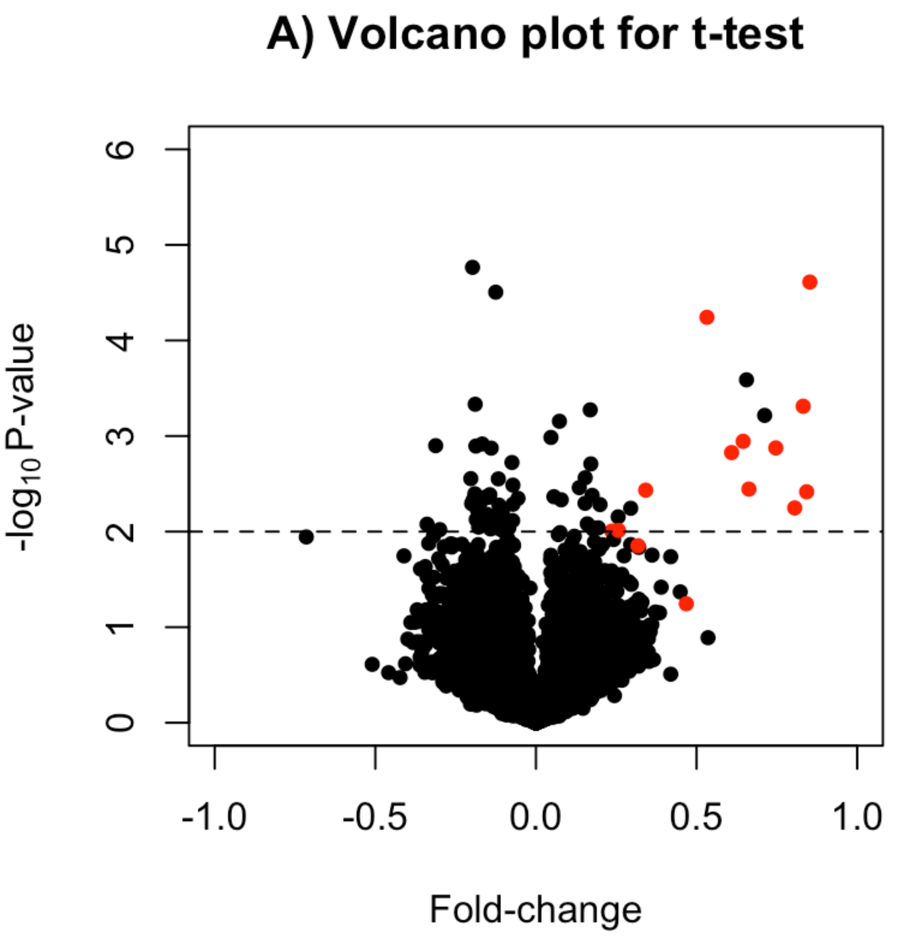
use this posterior mean to get a *moderated* t-statistic

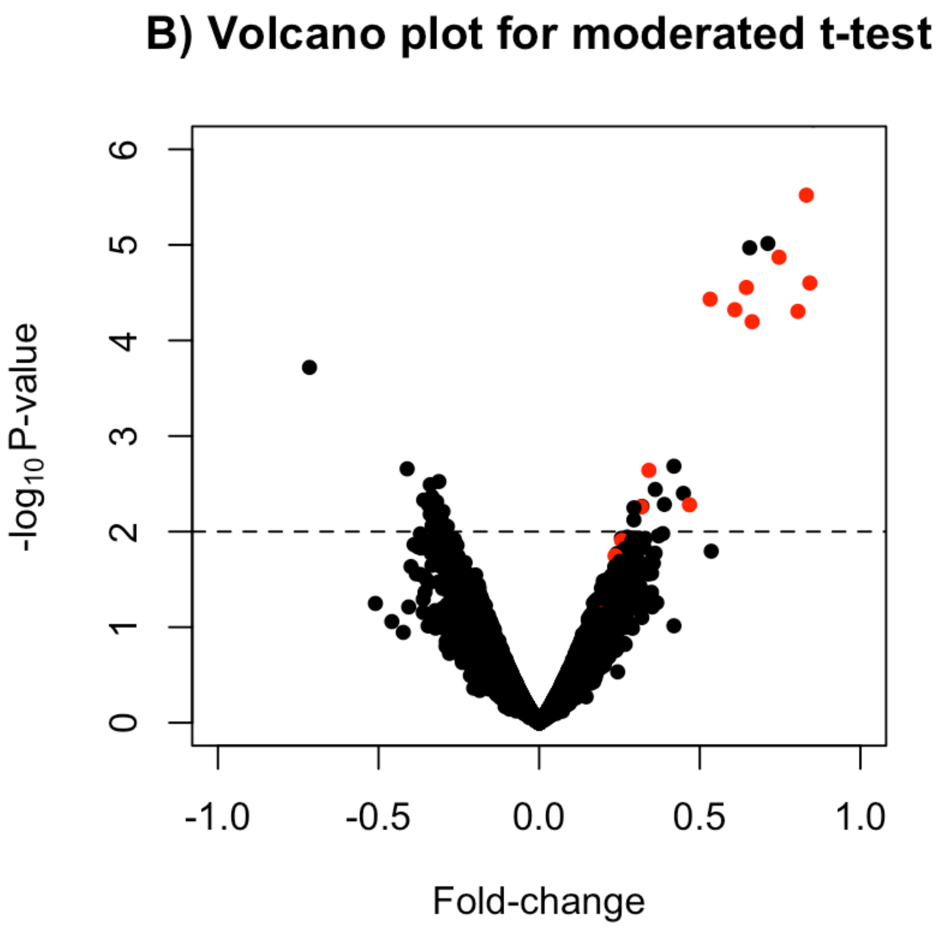
$$\tilde{t}_{gj} = \frac{\hat{\beta}_{gj}}{\tilde{s}_g \sqrt{v_j}}$$

under limma assumptions, we have the null distribution for this moderated t-statistic

$$\tilde{t}_{gj} \sim t_{d_0 + d_g} \text{ under } H_0$$

Revisit - spikein95





1470 differentially expressed genes!! – **majority have very small variances**

267 differentially expressed genes

side-by-side comparison of key quantities and results

	"plain vanilla"	limma
estimated gene-wise residual variance	$s_g^2 = \frac{1}{n-p} \hat{\varepsilon}_g^T \hat{\varepsilon}_g = \frac{1}{n-p} \sum_{i=1}^{n} \varepsilon_{gi}^2$	$\tilde{s}_g^2 = \frac{d_0 s_0^2 + d_g s_g^2}{d_0 + d_g}$
t-statistic for $H_0 : \beta_{gj} = 0$	$t_{gj} = \frac{\hat{\beta}_{gj}}{s_g \sqrt{v_j}}$	$\tilde{t}_{gj} = \frac{\hat{\beta}_{gj}}{\tilde{s}_g \sqrt{v_j}}$
distribution of the t-statistic when $\beta_{gj} = 0$	$t_{gj} \sim t_{d_g}$	$\tilde{t}_{gj} \sim t_{d_0 + d_g}$

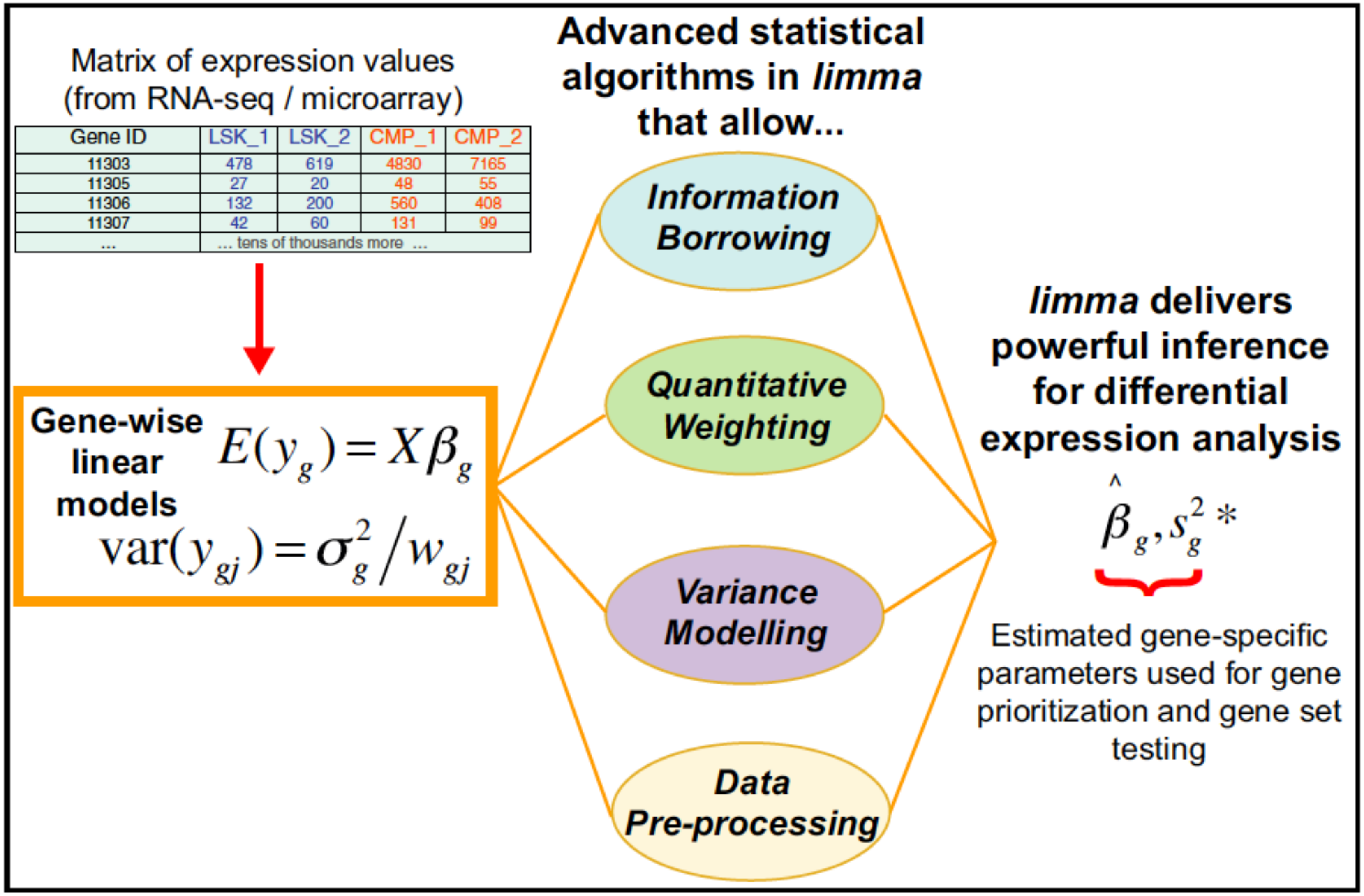
moderated variances will be "shrunk" towards the typical gene-wise variance, relative to raw sample residual variances

should result in fewer small variances and large t-stats

degrees of freedom for null distribution goes up, relative to default $d = n - p$ → makes it closer to a standard normal → makes tail probabilities smaller → makes p-values smaller

overall, when all is well, limma will deliver statistical results that are more stable, more powerful

Linear Models for Microarrays and RNA-Seq Data (limma)


Matrix of expression values (from RNA-seq / microarray)
Advanced statistical algorithms in *limma* that allow...
Information Borrowing
Quantitative Weighting
Variance Modelling
Data Pre-processing
Gene-wise linear models
limma delivers powerful inference for differential expression analysis
Estimated gene-specific parameters used for gene prioritization and gene set testing

Gene ID	LSK_1	LSK_2	CMP_1	CMP_2
11303	478	619	4830	7165
11305	27	20	48	55
11306	132	200	560	408
11307	42	60	131	99
...	... tens of thousands more ...			

$$E(y_g) = X\beta_g$$

$$\mathrm{var}(y_{gj}) = \sigma_g^2 / w_{gj}$$

$$\underbrace{\hat{\beta}_g, s_g^2}\ *$$

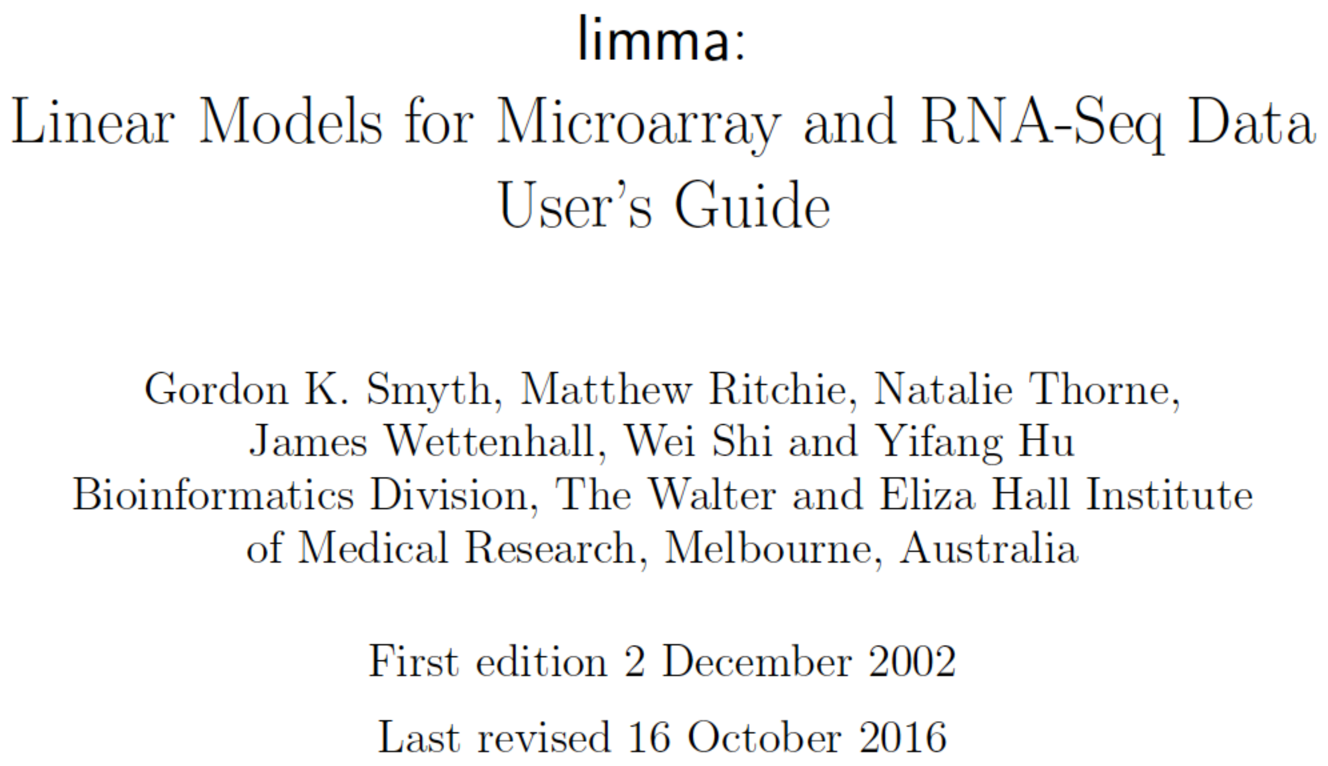
Smyth, Gordon K. (2004) "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments," Statistical Applications in Genetics and Molecular Biology: Vol. 3 : Iss. 1, Article 3.
DOI: 10.2202/1544-6115.1027
Available at: http://www.bepress.com/sagmb/vol3/iss1/art3
link no longer works now that SAGMB has been assimilated into ~~the Borg~~ deGruyter

But you should probably regard this as more definitive:
(Smyth describes as a "Reprint PDF with corrections") and it's dated 30 June 2009)
http://www.statsci.org/smyth/pubs/ebayes.pdf

http://bioinf.wehi.edu.au/limma/

http://www.statsci.org/smyth/index.html

Bioinformatics and Computational Biology Solutions Using R and Bioconductor -- eBook | Robert Gentleman, Vincent J. Carey, Wolfgang Huber, Rafael A. Irizarry, and Sandrine Dudoit, Springer 2005. Chapters 23 (limma: Linear Models for Microarray, by Smyth) and 14 (Analysis of Differential Gene Expression Studies, by Scholtens and von Heydebreck) are especially relevant.



limma:
Linear Models for Microarray and RNA-Seq Data User's Guide

Gordon K. Smyth, Matthew Ritchie, Natalie Thorne,
James Wettenhall, Wei Shi and Yifang Hu
Bioinformatics Division, The Walter and Eliza Hall Institute
of Medical Research, Melbourne, Australia

First edition 2 December 2002

Last revised 16 October 2016

http://www.bioconductor.org/packages/devel/bioc/vignettes/limma/inst/doc/usersguide.pdf specific sections I **strongly encourage** you to read

Ch. 8 Linear Models Overview
8.1 Introduction p. 35
8.2 Single-Channel Designs p. 36

Ch. 9 Single-Channel Experimental Designs
9.1 Introduction p. 40
9.2 Two Groups p.40
9.3 Several Groups p.42
9.4 Additive Models and Blocking p. 43
9.5 Interaction Models: 2 x 2 Factorial Designs p. 43*

Ch. 13 Statistics for Differential Expression
13.1 Summary Top-Tables p. 60
13.2 Fitted Model Objects p. 61
13.3 Multiple Testing Across Contrasts p. 62

Ch. 16 and Ch. 17: Case Studies

Chapter 7

Filtering

Note that filtering methods involving variances should not be used. The limma algorithm analyses the spread of the genewise variances. Any filtering method based on genewise variances will change the distribution of variances, will interfere with the limma algorithm and hence will give poor results.

Functions that make your life easier:

functions	
model.matrix	Takes in your “factors” and makes a design matrix
lmFit	Fits the linear model to all genes (each gene separately) – replace gene with “feature” depending on your data.
makeContrasts	Create the contrast matrix C that you desire
eBayes	Use output of linear regression to compute moderated t statistics
topTable	Query your results; sort your pvalues; sort genes; Adjust for multiple comparisons etc

23.8 Several groups

The above approaches for two groups extend easily to any number of groups. Suppose that three RNA targets are to be compared using Affymetrix arrays. Suppose that the three targets are called "RNA1," "RNA2," and "RNA3" and that the column `targets$Target` indicates which one was hybridized to each array. An appropriate design matrix can be created using

```
> f <- factor(targets$Target, levels = c("RNA1",
+     "RNA2", "RNA3"))
```

x is categorical, a factor specifies 3 groups

make design matrix, request "cell means" parametrization

```
> design <- model.matrix(~0 + f)
> colnames(design) <- c("RNA1", "RNA2", "RNA3")
```

To make all pair-wise comparisons between the three groups, one could proceed

fit linear model using least squares

```
> fit <- lmFit(eset, design)
> contrast.matrix <- makeContrasts(RNA2 - RNA1,
+     RNA3 - RNA2, RNA3 - RNA1, levels = design)
> fit2 <- contrasts.fit(fit, contrast.matrix)
> fit2 <- eBayes(fit2)
```

specify contrasts of interest; here, all three pairwise comparisons

moderate the test stats

A list of top genes for RNA2 versus RNA1 can be obtained from

```
> topTable(fit2, coef = 1, adjust = "fdr")
```

produce FDR-adjusted p-values, a la Benjamini-Hochberg, for gene-wise test of H_0: contrast #1 = 0, sort in order of statistical significance, and report the top hits

```
> system.time(jFit <- lmFit(prDat, jDesMat))
   user  system elapsed
  0.345   0.113   0.459
```

using limma's lmFit() function to perform two-way ANOVA for ~30K probesets

this takes a trivial amount of time

the hard parts for the analyst are choosing the model, choosing how to parametrize it and digesting the results

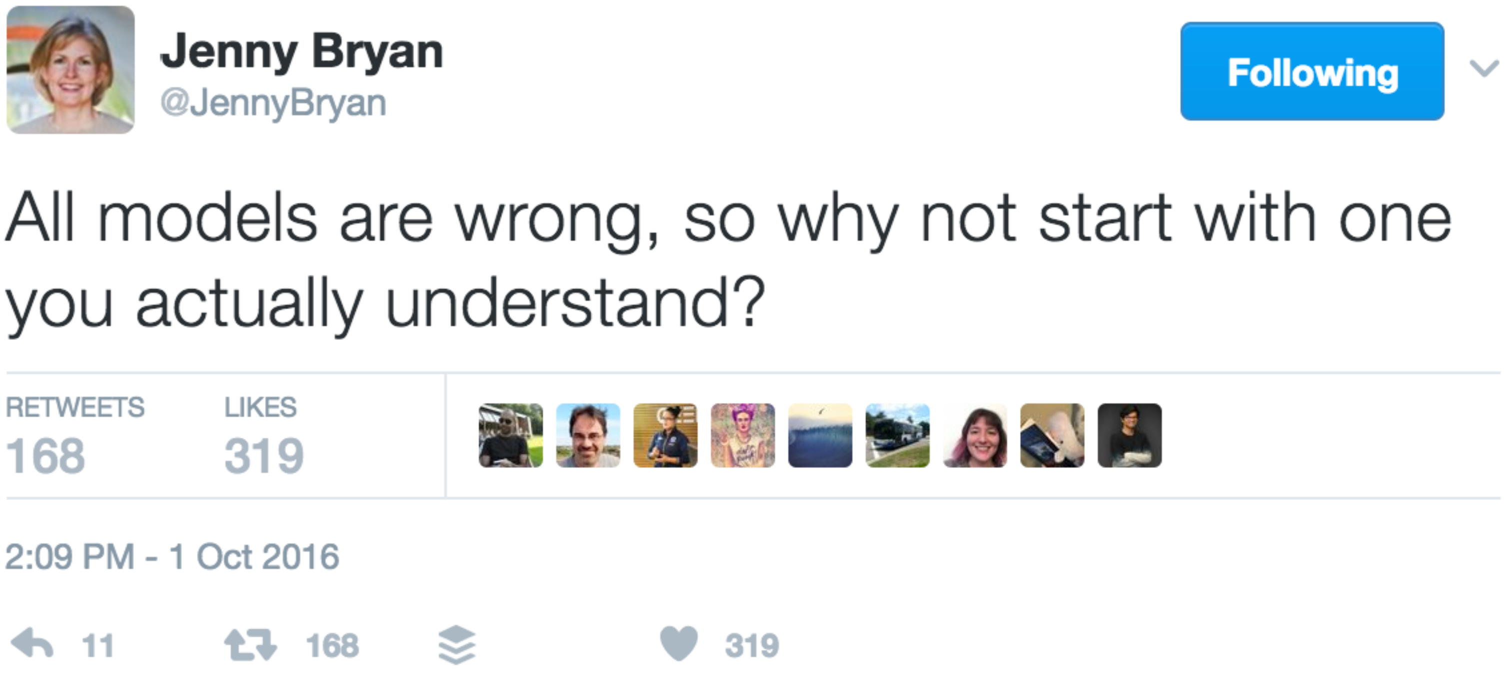
novices will be surprised what a non-issue the number of genes, number of samples is wise words (I find) relevant to science, statistical analysis, frequentism vs. Bayesianism, pragmatic approaches vs. mathematically pure ones, ...

All models are ~~wrong~~, but some are useful. (George E. P. Box)

simplification

"An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem." John Tukey

"Absolute certainty is a privilege of uneducated minds and fanatics. It is, for scientific folk, an unattainable ideal." Cassius J. Keyser



Jenny Bryan
@JennyBryan

Following

All models are wrong, so why not start with one you actually understand?

RETWEETS 168 LIKES 319

2:09 PM - 1 Oct 2016

11 168 319

```
> jDesMat <- model.matrix(~ gType * devStage, prDes)
> ## ridiculous machination to print a version to screen with small
> ## variable names
> foo <- jDesMat
> colnames(foo) <- paste0("X", formatC(seq_len(ncol(jDesMat)), width = 2, flag = "0"))
> cbind(prDes, foo)
          sample devStage gType X01 X02 X03 X04 X05 X06 X07 X08 X09 X10
Sample_20     20      E16    wt   1   0   0   0   0   0   0   0   0   0
Sample_21     21      E16    wt   1   0   0   0   0   0   0   0   0   0
Sample_22     22      E16    wt   1   0   0   0   0   0   0   0   0   0
Sample_23     23      E16    wt   1   0   0   0   0   0   0   0   0   0
Sample_16     16      E16 NrlKO   1   1   0   0   0   0   0   0   0   0
Sample_17     17      E16 NrlKO   1   1   0   0   0   0   0   0   0   0
Sample_6       6      E16 NrlKO   1   1   0   0   0   0   0   0   0   0
Sample_24     24       P2    wt   1   0   1   0   0   0   0   0   0   0
Sample_25     25       P2    wt   1   0   1   0   0   0   0   0   0   0
Sample_26     26       P2    wt   1   0   1   0   0   0   0   0   0   0
Sample_27     27       P2    wt   1   0   1   0   0   0   0   0   0   0
Sample_14     14       P2 NrlKO   1   1   1   0   0   0   1   0   0   0
Sample_3       3       P2 NrlKO   1   1   1   0   0   0   1   0   0   0
Sample_5       5       P2 NrlKO   1   1   1   0   0   0   1   0   0   0
Sample_8       8       P2 NrlKO   1   1   1   0   0   0   1   0   0   0
Sample_28     28       P6    wt   1   0   0   1   0   0   0   0   0   0
Sample_29     29       P6    wt   1   0   0   1   0   0   0   0   0   0
Sample_30     30       P6    wt   1   0   0   1   0   0   0   0   0   0
Sample_31     31       P6    wt   1   0   0   1   0   0   0   0   0   0
Sample_1       1       P6 NrlKO   1   1   0   1   0   0   0   1   0   0
Sample_10     10       P6 NrlKO   1   1   0   1   0   0   0   1   0   0
Sample_4       4       P6 NrlKO   1   1   0   1   0   0   0   1   0   0
Sample_7       7       P6 NrlKO   1   1   0   1   0   0   0   1   0   0
Sample_32     32      P10    wt   1   0   0   0   1   0   0   0   0   0
Sample_33     33      P10    wt   1   0   0   0   1   0   0   0   0   0
Sample_34     34      P10    wt   1   0   0   0   1   0   0   0   0   0
Sample_35     35      P10    wt   1   0   0   0   1   0   0   0   0   0
Sample_13     13      P10 NrlKO   1   1   0   0   1   0   0   0   1   0
Sample_15     15      P10 NrlKO   1   1   0   0   1   0   0   0   1   0
Sample_18     18      P10 NrlKO   1   1   0   0   1   0   0   0   1   0
Sample_19     19      P10 NrlKO   1   1   0   0   1   0   0   0   1   0
Sample_36     36  4_weeks    wt   1   0   0   0   0   1   0   0   0   0
Sample_37     37  4_weeks    wt   1   0   0   0   0   1   0   0   0   0
Sample_38     38  4_weeks    wt   1   0   0   0   0   1   0   0   0   0
Sample_39     39  4_weeks    wt   1   0   0   0   0   1   0   0   0   0
Sample_11     11  4_weeks NrlKO   1   1   0   0   0   1   0   0   0   1
Sample_12     12  4_weeks NrlKO   1   1   0   0   0   1   0   0   0   1
Sample_2       2  4_weeks NrlKO   1   1   0   0   0   1   0   0   0   1
Sample_9       9  4_weeks NrlKO   1   1   0   0   0   1   0   0   0   1
```

human- and computer-readable info on factor levels for each sample = `prDes`

lm.fit- and lmFit-ready encoding of factor levels for each sample = a design matrix = X or X_g

θ

τ_{NrlKO}

β_{P2}, *etc*

$(\tau\beta)_{NrlKO,P2}$, *etc*

```
> cbind(colnames(jDesMat))
      [,1]
 [1,] "(Intercept)"
 [2,] "gTypeNrlKO"
 [3,] "devStageP2"
 [4,] "devStageP6"
 [5,] "devStageP10"
 [6,] "devStage4_weeks"
 [7,] "gTypeNrlKO:devStageP2"
 [8,] "gTypeNrlKO:devStageP6"
 [9,] "gTypeNrlKO:devStageP10"
[10,] "gTypeNrlKO:devStage4_weeks"
```

```
> jDesMat <- model.matrix(~ gType * devStage, prDes)
> jFit <- lmFit(prDat, jDesMat)

> ebFit <- eBayes(jFit)
```

```
> summary(jFit)
                  Length Class      Mode
coefficients      299490 -none-     numeric
rank                   1 -none-     numeric
assign                10 -none-     numeric
qr                     5 qr         list
df.residual        29949 -none-     numeric
sigma              29949 -none-     numeric
cov.coefficients     100 -none-     numeric
stdev.unscaled    299490 -none-     numeric
pivot                 10 -none-     numeric
genes                  1 data.frame list
Amean              29949 -none-     numeric
method                 1 -none-     character
design               390 -none-     numeric
```

```
> summary(ebFit)
                  Length Class      Mode
coefficients      299490 -none-     numeric
rank                   1 -none-     numeric
assign                10 -none-     numeric
qr                     5 qr         list
df.residual        29949 -none-     numeric
sigma              29949 -none-     numeric
cov.coefficients     100 -none-     numeric
stdev.unscaled    299490 -none-     numeric
pivot                 10 -none-     numeric
genes                  1 data.frame list
Amean              29949 -none-     numeric
method                 1 -none-     character
design               390 -none-     numeric
df.prior               1 -none-     numeric
s2.prior               1 -none-     numeric
var.prior             10 -none-     numeric
proportion             1 -none-     numeric
s2.post            29949 -none-     numeric
t                 299490 -none-     numeric
df.total           29949 -none-     numeric
p.value           299490 -none-     numeric
lods              299490 -none-     numeric
F                  29949 -none-     numeric
F.p.value          29949 -none-     numeric
```

see all this new stuff? topTable() will help you use it to find interesting genes

limma workflow

limma is designed to help you out
AFTER you've applied eBayes()

U)

fit a separate linear model for
each response, e.g. gene

`lmFit(...)`

fitted models

apply an Empirical Bayes
procedure for moderating
estimates of error variance

`eBayes(...)`

extract estimated parameters
or p-values or ...
compare big models to small
etc etc

`topTable(...)`

```
> jDesMat <- model.matrix(~ gType * devStage, prDes)
> jFit <- lmFit(prDat, jDesMat)
> ebFit <- eBayes(jFit)
```

```
> topTable(ebFit)
                       ID X.Intercept.  gTypeNrlKO devStageP2 devStageP6
1438940_x_at 1438940_x_at      12.8625  0.05750000     0.0850     0.1325
1436884_x_at 1436884_x_at      12.9275  0.05916667     0.1775     0.3225
1456736_x_at 1456736_x_at      12.3225 -0.07583333     0.1625     0.3050
1455897_x_at 1455897_x_at      13.0575  0.01916667    -0.0150     0.1100
1451240_a_at 1451240_a_at      12.9975 -0.03083333     0.3100     0.2800
1454613_at     1454613_at      12.4675 -0.28750000    -0.1075    -0.0500
1450084_s_at 1450084_s_at      12.6350 -0.04500000     0.0825     0.0525
1437192_x_at 1437192_x_at      12.9425  0.07750000     0.2750     0.3000
1449676_at     1449676_at      12.7075 -0.05750000    -0.0750    -0.1525
1438657_x_at 1438657_x_at      12.7825  0.15083333     0.1400     0.1250
             devStageP10 devStage4_weeks gTypeNrlKO.devStageP2
1438940_x_at      0.3425          0.3500            0.01750000
1436884_x_at      0.0300          0.0250           -0.24166667
1456736_x_at      0.2075          0.0725            0.03583333
1455897_x_at      0.4325          0.4775            0.09083333
1451240_a_at      0.2800         -0.3700            0.23083333
1454613_at       -0.1025         -0.3825            0.15500000
1450084_s_at      0.1725          0.2600            0.13000000
1437192_x_at      0.2925         -0.0050           -0.19750000
1449676_at       -0.1725         -0.5075            0.17250000
1438657_x_at     -0.1850         -0.4500           -0.30083333
             gTypeNrlKO.devStageP6 gTypeNrlKO.devStageP10
1438940_x_at            0.05500000            -0.12000000
1436884_x_at           -0.37666667            -0.10166667
1456736_x_at           -0.02416667            -0.14416667
1455897_x_at            0.19583333            -0.06666667
1451240_a_at            0.28833333             0.18583333
1454613_at              0.15000000             0.27250000
1450084_s_at            0.05000000             0.06250000
1437192_x_at           -0.23750000            -0.21500000
1449676_at              0.28500000             0.28250000
1438657_x_at           -0.29833333             0.04166667
             gTypeNrlKO.devStage4_weeks  AveExpr        F      P.Value
1438940_x_at               -0.132500000 13.05872 63728.49 5.063198e-66
1436884_x_at               -0.009166667 12.99538 52673.21 1.077659e-64
1456736_x_at                0.060833333 12.43154 51040.87 1.786135e-64
1455897_x_at               -0.094166667 13.28590 49456.68 2.962710e-64
1451240_a_at                0.720833333 13.23128 48588.27 3.937053e-64
1454613_at                  0.430000000 12.29897 43799.55 2.081658e-63
1450084_s_at               -0.010000000 12.75333 43532.81 2.296095e-63
1437192_x_at                0.030000000 13.09359 42553.50 3.308144e-63
1449676_at                  0.515000000 12.62205 42311.46 3.625302e-63
1438657_x_at                0.086666667 12.73179 42145.13 3.861878e-63
                adj.P.Val
1438940_x_at 1.516377e-61
1436884_x_at 1.613741e-60
1456736_x_at 1.783099e-60
1455897_x_at 2.218255e-60
1451240_a_at 2.358216e-60
1454613_at   9.823679e-60
1450084_s_at 9.823679e-60
1437192_x_at 1.156594e-59
1449676_at   1.156594e-59
1438657_x_at 1.156594e-59
```

however, you can't just use topTable() on auto-pilot

you still must know and describe what you consider a "hit" to be

```
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)

topTableF(fit, number=10, genelist=fit$genes, adjust.method="BH",
          sort.by="F", p.value=1, lfc=0)
```

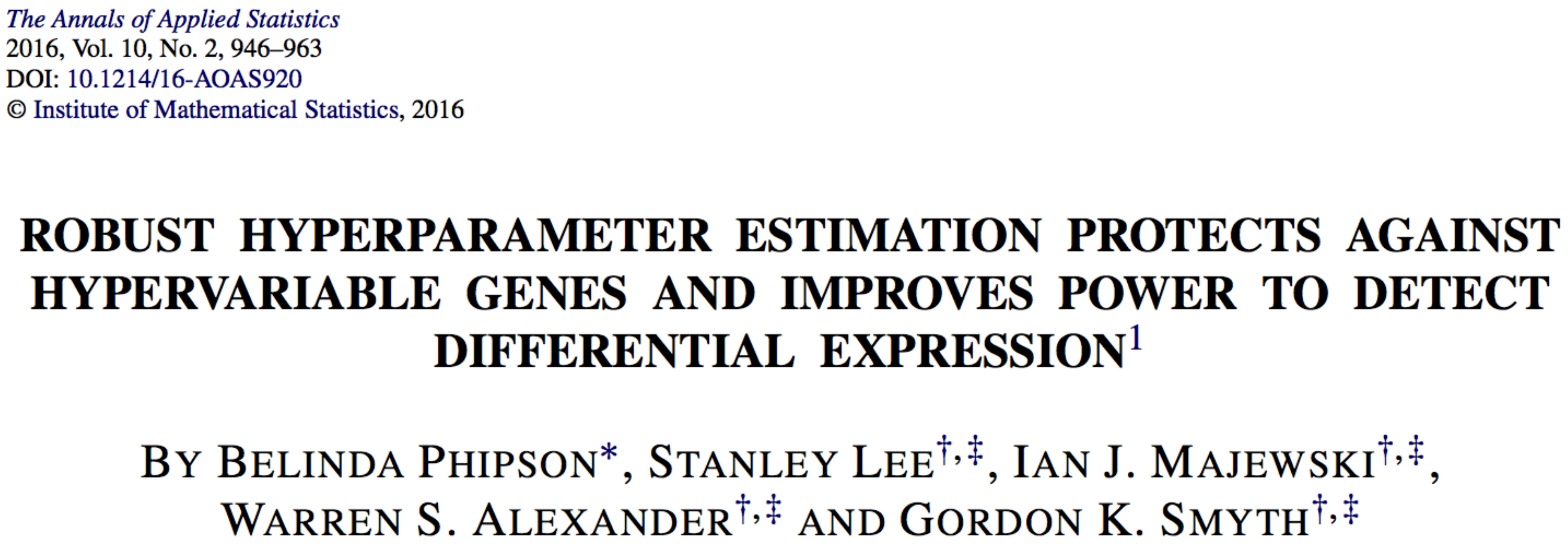
```
   coef: column number or column name specifying which coefficient or
         contrast of the linear model is of interest. For 'topTable',
         can also be a vector of column subscripts, in which case the
         gene ranking is by F-statistic for that set of contrasts.

    'topTableF' ranks genes on the basis of moderated F-statistics for
    one or more coefficients. If 'topTable' is called with 'coef' has
    length greater than 1, then the specified columns will be
    extracted from 'fit' and 'topTableF' called on the result.
    'topTable' with 'coef=NULL' is the same as 'topTableF', unless the
    fitted model 'fit' has only one column.
```

`coef` argument is where you specify what it is you want to test for equality with zero

Recent limma feature

Estimating gene-specific variance priors!



The Annals of Applied Statistics
2016, Vol. 10, No. 2, 946–963
DOI: 10.1214/16-AOAS920
© Institute of Mathematical Statistics, 2016

ROBUST HYPERPARAMETER ESTIMATION PROTECTS AGAINST HYPERVARIABLE GENES AND IMPROVES POWER TO DETECT DIFFERENTIAL EXPRESSION[1]

BY BELINDA PHIPSON*, STANLEY LEE†,‡, IAN J. MAJEWSKI†,‡, WARREN S. ALEXANDER†,‡ AND GORDON K. SMYTH†,‡

```
eBayes(fit, robust=TRUE)
```