

Statistical Methods for High Dimensional Biology

STAT/BIOF/GSAT 540

Lecture 10 – Linear Models Part IV

Sara Mostafavi

FEB 03 2016

****Based on slides by Dr. Jennifer Bryan****

outline

- Review
 - Linear regression framework; terminology
 - Model selection & overfitting
- Large scale differential expression analysis:
 - Assessing ALL genes (in a univariate way)
 - i.e., same model, except run it 30K times
 - Empirical Bayes
 - Practical issues: running Limma

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

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

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

Here we are just fitting a line but using matrix notation to handle all n observations at once, more elegantly.

Big pay-offs ensue

Terminology Review

Response, “observed
variables”, outcome,
labels, target

$$Y = f(X | \theta) = X\alpha + \varepsilon,$$

$$\alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \dots \\ \alpha_p \end{bmatrix}$$

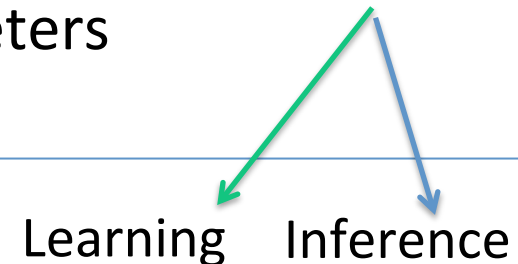
Covariates, “features”,
design matrix, factors,
explanatory variables,
“data points”

Coefficients, parameters,
weights

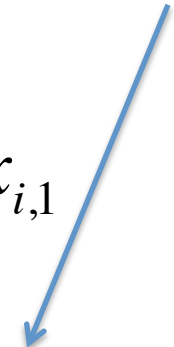
$$Y = f(X | \theta) = X\alpha + \varepsilon, \quad \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \dots \\ \alpha_p \end{bmatrix}$$

All models take as input a matrix X and *apply* a function to X :
the **GOAL** is to apply a function to X , so you get an output that is
“similar” to the vector y

Function is “fitted” to make $f(X)$ look like y , by “selecting”
appropriate parameters

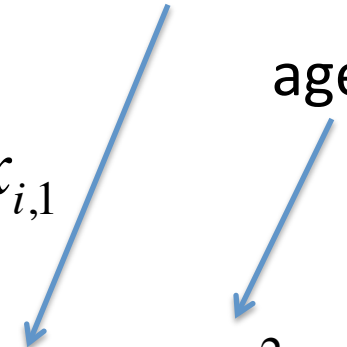


Linear model

$$y_i = \alpha_0 + \alpha_1 x_{i,1}$$


age

Polynomial regression (degree 2)

$$y_i = \alpha_0 + \alpha_1 x_{i,1} + \alpha_2 x_{i,1}^2$$


age²

Polynomial regression (degree 3)

$$y_i = \alpha_0 + \alpha_1 x_{i,1} + \alpha_2 x_{i,1}^2 + \alpha_3 x_{i,1}^3$$

We can have many “columns” in the design matrix – each column will be represented by a parameter

$$y = X\alpha + \varepsilon$$

$$X = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,p} \\ 1 & x_{2,1} & \dots & x_{2,p} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n,1} & \dots & x_{n,p} \end{bmatrix}$$

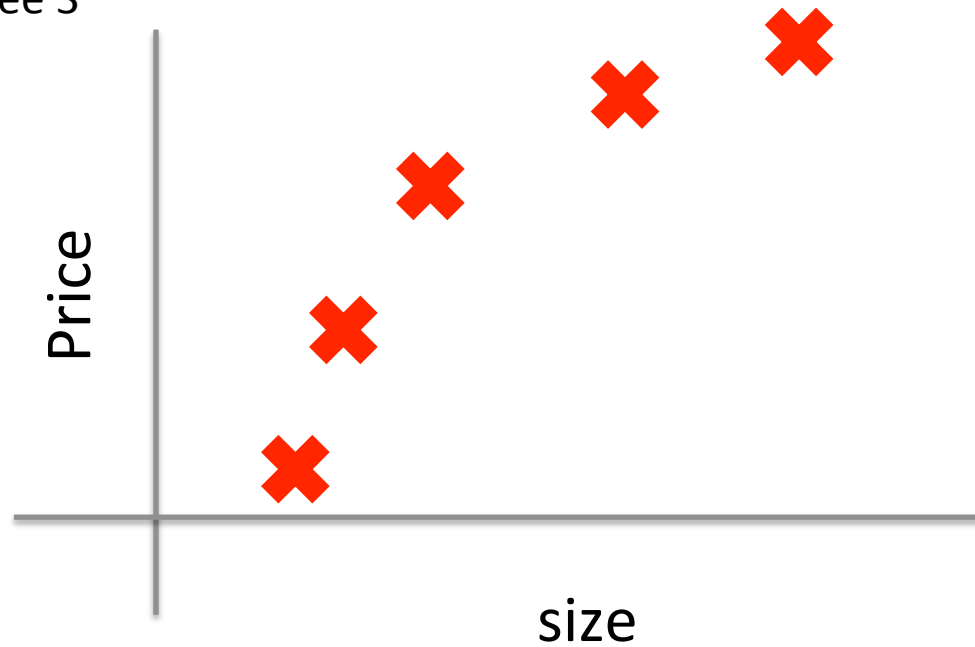
$$y_i = \alpha_0 + \alpha_1 x_{i,1} + \alpha_2 x_{i,2} + \dots + \alpha_p x_{i,p}$$

What model should we fit here?

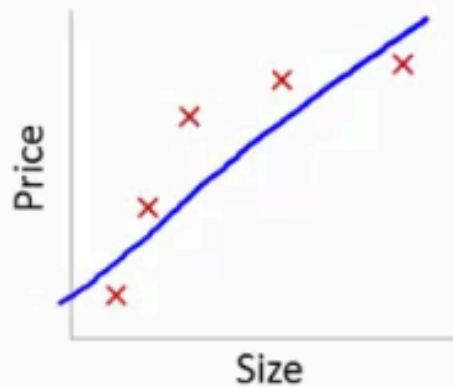
linear

Polynomial degree 2

Polynomial degree 3

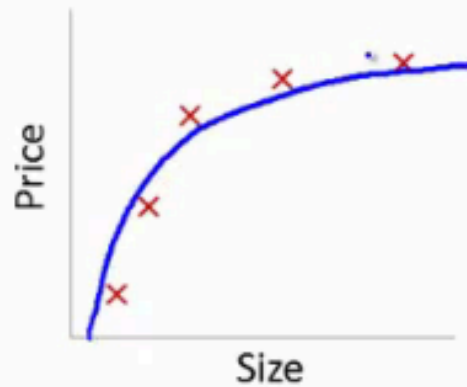


Fitting the same data, with different model: illustrating “overfitting”



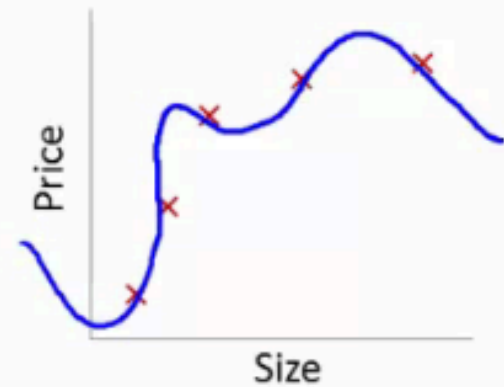
$$y_i = \alpha_0 + \alpha_1 x_{i,1}$$

High bias
(underfit)



$$y_i = \alpha_0 + \alpha_1 x_{i,1} + \alpha_2 x_{i,1}^2$$

“Just right”



$$y_i = \alpha_0 + \alpha_1 x_{i,1} + \alpha_2 x_{i,1}^2 + \alpha_3 x_{i,1}^3$$

High variance
(overfit)

We use our judgment for “small problems” or use “model selection” procedure on a large scale

Model selection

- Depending on the question/problem: F test, LRT, AIC, BIC ...

Using the same framework of linear regression, you can ask questions like:

- After accounting for “___”, does “___” have an effect on gene expression:
 - E.g., we typically want to account for effect of sex and/or gender in observational studies
 - E.g., we typically want to account for effect of treatment in disease studies where the goal is to find “causal/upstream” genes that tell us something about the disease mechanism.
- Look up F test and Likelihood Ratio Test (LRT)

`lm(yMat ~ x)`

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

$$\begin{bmatrix} y_{11} & \dots & y_{1G} \\ y_{21} & & y_{2G} \\ \vdots & & \\ y_{n1} & & y_{nG} \end{bmatrix} = X \begin{bmatrix} \alpha_1 & \dots & \alpha_G \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} & & \varepsilon_{1G} \\ \varepsilon_{21} & & \varepsilon_{2G} \\ \vdots & \dots & \\ \varepsilon_{n1} & & \varepsilon_{nG} \end{bmatrix}$$

built-in function `lm()` can do “multivariate regression” = many dependent vars (“responses”)
aka “multivariate multiple regression”

From `lm()` documentation:

If response is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

`lm` returns an object of class “lm” or for multiple responses of class `c("mlm", "lm")`.

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

Large scale inference

Overview and keywords

Incorporating “indirect evidence” into hypothesis tests, “borrowing strength” across genes **today**

- moderated t - and F-statistics (limma), empirical Bayes, SAM

Problems and solutions when conducting thousands of hypothesis tests **next Wed**

- multiplicity, multiple testing, Bonferroni, false discovery rate (FDR), q -values

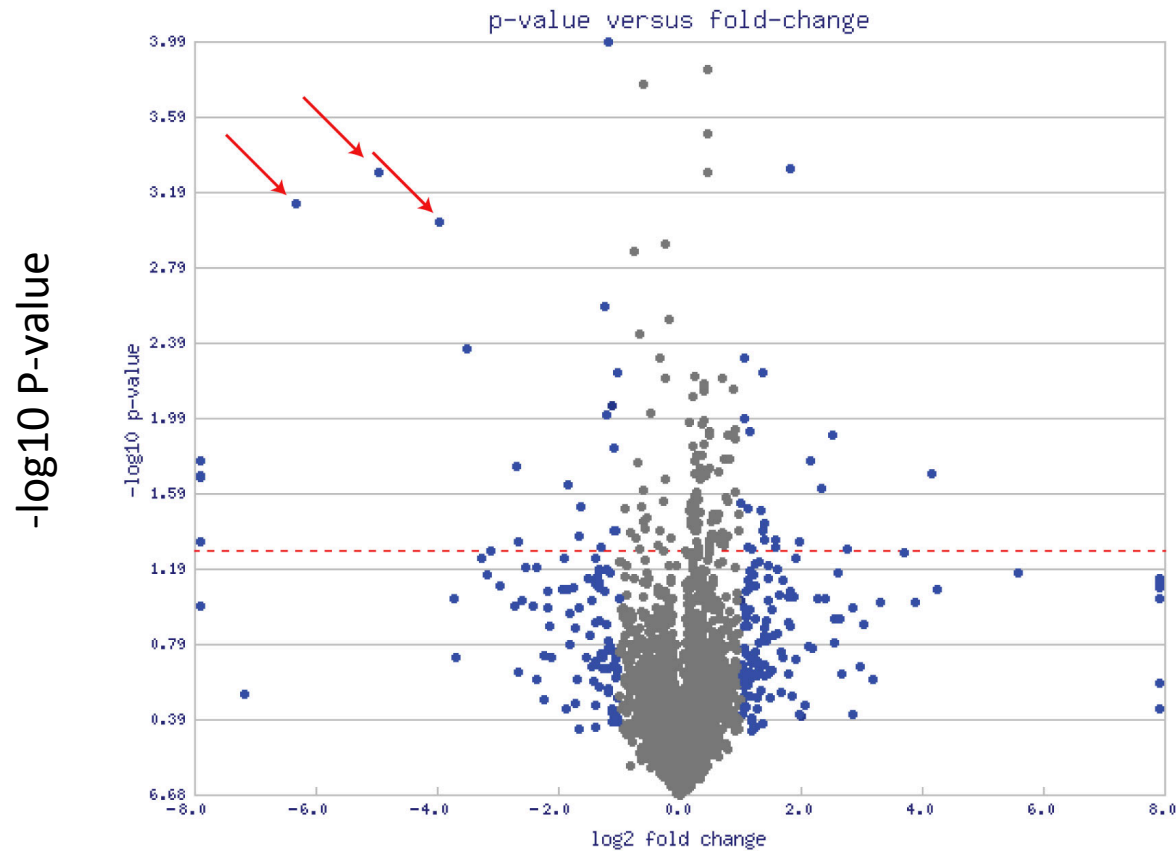
What's special about Limma, and what is eBayes?

Recurring theme in analysis of “high dimensional” biological data

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

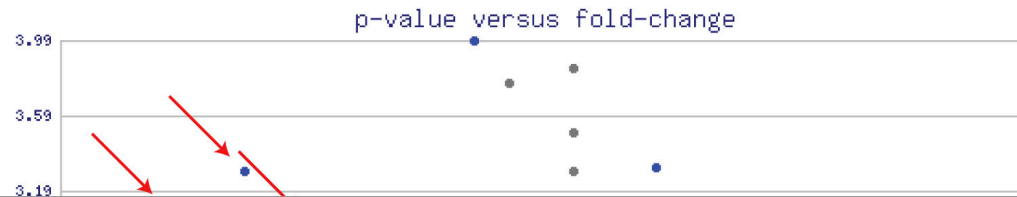
- Replication rate typically lower for genes with subtle effects
- Ad hoc filters: require small pvalues and large effect sizes

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

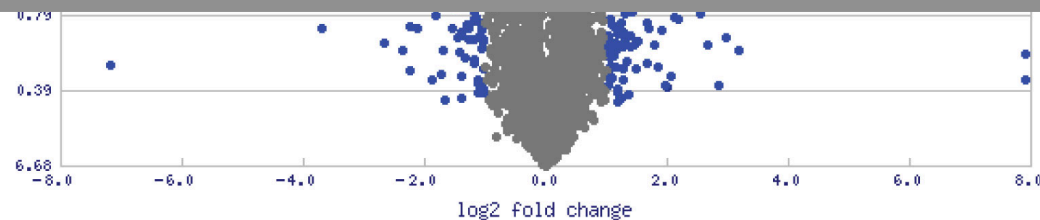


Log₂ fold change (i.e., effect size)

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

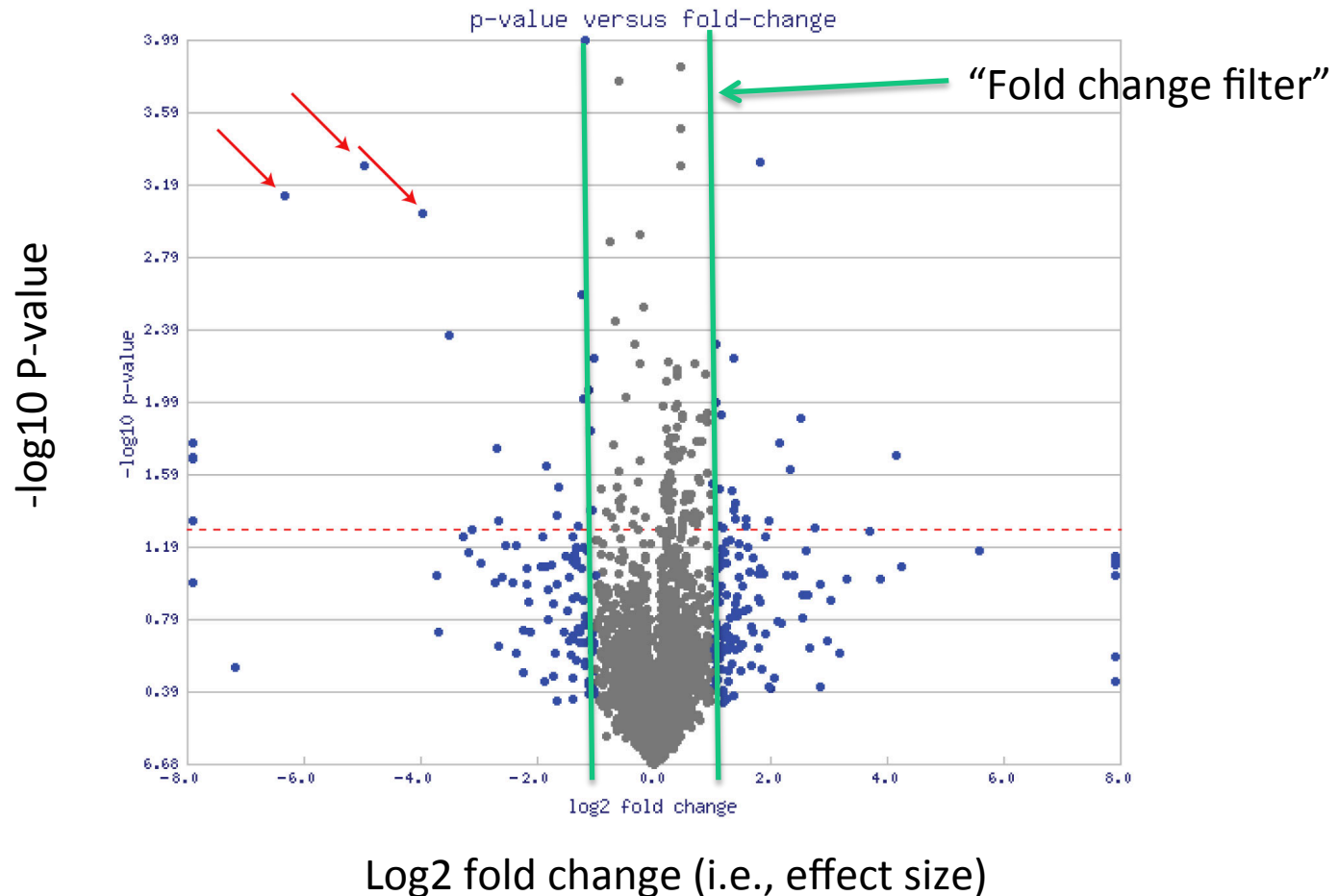


Some genes with very small pvalues (large $-\log_{10}$ pvalues) are not biologically meaningful.



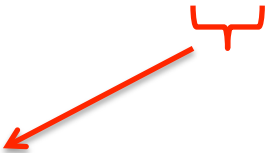
Log2 fold change (i.e., effect size)

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



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

$$t_{gj} = \frac{\hat{\alpha}_{gj}}{SE(\hat{\alpha}_{gj})} = \frac{\hat{\alpha}_{gj}}{s_g \sqrt{v_j}} \sim t_d \text{ under } H_0$$



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



d=residual degree of
freedom

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

$$t_{gj} = \frac{\hat{\alpha}_{gj}}{SE(\hat{\alpha}_{gj})} = \frac{\hat{\alpha}_{gj}}{s_g \sqrt{v_j}} \sim t_d \quad \text{under } H_0$$

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

- What if we add a little constant to the denominator?

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

- We could even come up with “weights w ” so to take a weighted average between estimated var and *prior* var.

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

Moderated t statistics → Derived using the Empirical Bayes framework.

First, we will talk about moderated t statistics (and the Empirical Bayes approach), then I'll give you some pointers on using limma with empirical bayes approach for assessing differential gene expression on a large scale.

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:

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

$$Y_g = X\alpha_g + \varepsilon_g \quad \text{the data model}$$

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

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

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

“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 $\hat{V}(\hat{\alpha}_g) = s_g^2(X^T X)^{-1} = Vs_g^2$ actually look like in a concrete example?

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

$$\beta_g = C^T \alpha_g$$

the contrasts of
interest (if you care
about the alphas only,
C = identity matrix)

$$\text{var}(\hat{\beta}_g) = C^T V C s_g^2$$

the usual estimated
covariance matrix for
the contrast
estimators

v_j = the j -th diagonal element of $C^T V C$

will come in handy
soon

“unscaled covariance”

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

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



limma imposes a hierarchical model, which describes how the gene-wise β_{gj} 's and σ_g^2 's vary across the genes

this is done by assuming a *prior distribution* for those quantities

Going from maximum likelihood estimation to Bayesian inference (not quite there; empirical Bayes is a hybrid)

MLE vs Maximum *a posteriori* (inspired by Bayesian statistics)

Recall *maximum likelihood estimation (frequentist)*:

Likelihood
function

Estimate parameters by
maximizing the *likelihood*
function

$$\begin{aligned} \operatorname{argmax}_{\theta} \quad LL(Y, X | \theta) &= \log P(Y | X, \theta) \\ &= \operatorname{argmax}_{\alpha, \sigma} \log N(X\alpha, \sigma^2) \quad \theta = \{\alpha, \sigma\} \end{aligned}$$

Maximum *a posteriori*:

Use Bayes rule and maximize the posterior on parameters

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

Prior distribution on the
parameters

Likelihood
function

Posterior on parameters

Assume a prior distribution for σ_g^2

gene-specific variances:

an inverse Chi-square prior with mean s_0^2 and d_0 degrees of freedom

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

Also a prior probability distribution for having a non-zero α

$$p(\alpha_{gj} \neq 0) = p_j$$

Interpretation: p_j is the proportion of truly differentially expressed (DE) genes w.r.t. the j -th contrast

No free lunch.

The new unknowns (or *hyperparameters*)

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

$$P(\beta_{gj} \neq 0) = p_j$$

$$\beta_{gj} \mid \sigma_g^2, \beta_{gj} \neq 0 \sim N(0, v_{0j} \sigma_g^2)$$

For now, trust we have ways of estimating them.

Defining prior distributions over the parameter, we can do *a posteriori inference*:

The posterior mean for gene-specific variance:

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

How to think about it:

a weighted mean of the prior (indirect evidence) and the observed (direct evidence) gene-specific variances

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

More how to think about it:

“shrinking” the observed gene-specific variance towards the “typical” variance implied by the prior

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

The weights and/or amount of shrinkage are controlled by d_0 relative to d .

If $d_0 = 0$, you ignore your prior.

If $d_0 = \infty$, you ignore your data (in terms of estimating variance!) ... implies making DE calls based on “fold-change” alone

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

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

side-by-side comparison of key quantities and results

“plain vanilla”

limma

estimated gene-wise residual variance

$$s_g^2 = \frac{1}{n-p} \hat{\boldsymbol{\varepsilon}}_g^T \hat{\boldsymbol{\varepsilon}}_g = \frac{1}{n-p} \sum_{i=1}^n \varepsilon_{gi}^2 \quad \tilde{s}_g^2 = \frac{d_0 s_0^2 + d s_g^2}{d_0 + d}$$

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 $\beta_{gj} = 0$

$$t_{gj} \sim t_d$$

$$\tilde{t}_{gj} \sim t_{d_0+d}$$

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 $d = n - p \rightarrow$ makes it closer to a standard normal \rightarrow makes tail probabilities smaller \rightarrow makes p-values smaller

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

history lesson

The moderated t has the advantage over the ordinary t statistic that large statistics are less likely to arise merely from under-estimated sample variances. This is because the posterior variance \hat{s}_g^2 offsets the small sample variances heavily in a relative sense while larger sample variances are moderated to a lesser relative degree. In this respect the moderated t statistic is similar in spirit to t -statistics with offset standard deviation. Provided that $d_0 < \infty$ and $d_g > 0$, the moderated t -statistic can be written

$$\tilde{t}_{gj} = \left(\frac{d_0 + d_g}{d_g} \right)^{1/2} \frac{\hat{\beta}_{gj}}{\sqrt{s_{*,g}^2 v_{gj}}}$$

where $s_{*,g}^2 = s_g^2 + (d_0/d_g)s_0^2$. This shows that the moderated t -statistic is proportional to a t -statistic with sample variances offset by a constant if the d_g are equal. Test statistics with offset standard deviations of the form

$$t_{*,gj} = \frac{\hat{\beta}_{gj}}{(s_g + a)\sqrt{v_{gj}}},$$

where a is a positive constant, have been used by Tusher et al (2001), Efron et al (2001) and Broberg (2003). Note that t_* offsets the standard deviation while \tilde{t} offsets the variance so the two statistics are not functions of one another. Unlike the moderated t , the offset statistic $t_{*,gj}$ is not connected in any formal way with the posterior odds of differential expression and does not have an associated distributional theory.

The moderated t -statistic \tilde{t}_{gj} may be used for inference about β_{gj} in a similar way to that in which the ordinary t -statistic would be used, except that the degrees of freedom are $d_g + d_0$ instead of d_g . The fact that the hyperparameters d_0 and s_0^2 are estimated

Moderated t -statistic particularly attractive when small sample size makes “direct evidence” estimation of gene-specific variances frightening.

Related to the test statistic in SAM / Tusher et al and CyberT of Baldi but has more theory behind it.

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*.

Practical stuff:

how do you run differential expression
analysis with limma in R?

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>

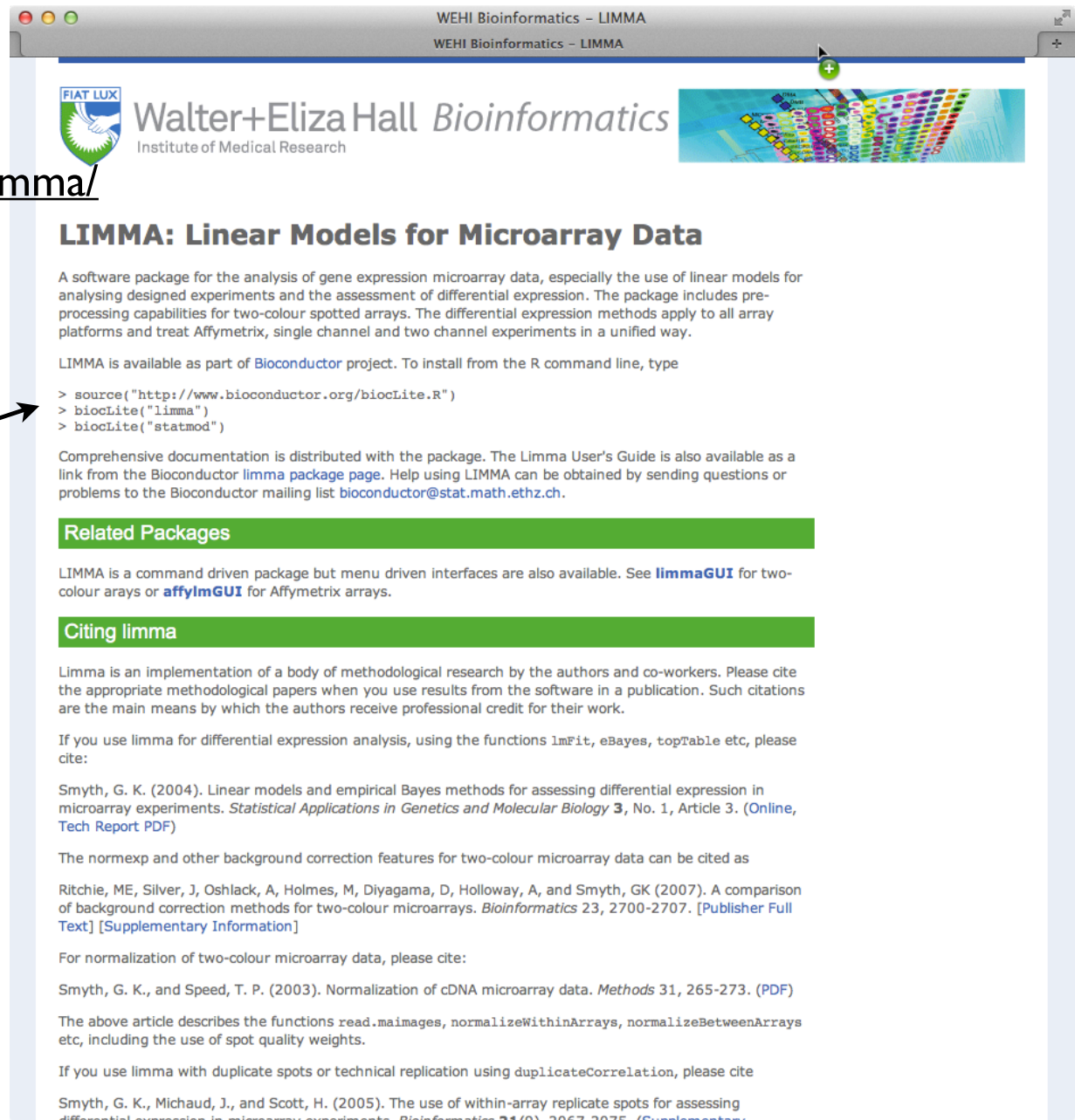
<http://bioinf.wehi.edu.au/limma/>

<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.


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

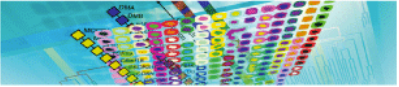
how to install limma

A screenshot of a web browser displaying the LIMMA website. The browser's address bar shows the URL 'http://bioinf.wehi.edu.au/limma/'. The website header includes the 'Walter+Eliza Hall Bioinformatics' logo and the text 'Institute of Medical Research'. The main content area is titled 'LIMMA: Linear Models for Microarray Data' and describes the software package. It includes installation instructions for R, a section for related packages, and a section for citing the software. The text is organized into sections with green headers. An arrow points from the text 'how to install limma' to the installation instructions section.

WEHI Bioinformatics – LIMMA

WEHI Bioinformatics – LIMMA

 **Walter+Eliza Hall Bioinformatics**
Institute of Medical Research



LIMMA: Linear Models for Microarray Data

A software package for the analysis of gene expression microarray data, especially the use of linear models for analysing designed experiments and the assessment of differential expression. The package includes pre-processing capabilities for two-colour spotted arrays. The differential expression methods apply to all array platforms and treat Affymetrix, single channel and two channel experiments in a unified way.

LIMMA is available as part of [Bioconductor](#) project. To install from the R command line, type

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("limma")
> biocLite("statmod")
```

Comprehensive documentation is distributed with the package. The Limma User's Guide is also available as a link from the Bioconductor [limma package page](#). Help using LIMMA can be obtained by sending questions or problems to the Bioconductor mailing list bioconductor@stat.math.ethz.ch.

Related Packages

LIMMA is a command driven package but menu driven interfaces are also available. See [limmaGUI](#) for two-colour arrays or [affyImGUI](#) for Affymetrix arrays.

Citing limma

Limma is an implementation of a body of methodological research by the authors and co-workers. Please cite the appropriate methodological papers when you use results from the software in a publication. Such citations are the main means by which the authors receive professional credit for their work.

If you use limma for differential expression analysis, using the functions `lmFit`, `eBayes`, `topTable` etc, please cite:

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology* 3, No. 1, Article 3. ([Online](#), [Tech Report PDF](#))

The normexp and other background correction features for two-colour microarray data can be cited as

Ritchie, ME, Silver, J, Oshlack, A, Holmes, M, Diyagama, D, Holloway, A, and Smyth, GK (2007). A comparison of background correction methods for two-colour microarrays. *Bioinformatics* 23, 2700-2707. [[Publisher Full Text](#)] [[Supplementary Information](#)]

For normalization of two-colour microarray data, please cite:

Smyth, G. K., and Speed, T. P. (2003). Normalization of cDNA microarray data. *Methods* 31, 265-273. ([PDF](#))

The above article describes the functions `read.maimages`, `normalizeWithinArrays`, `normalizeBetweenArrays` etc, including the use of spot quality weights.

If you use limma with duplicate spots or technical replication using `duplicateCorrelation`, please cite

Smyth, G. K., Michaud, J., and Scott, H. (2005). The use of within-array replicate spots for assessing differential expression in microarray experiments. *Bioinformatics* 21(9), 2067-2075. ([Supplementary](#)

..

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 8 January 2015

<http://www.bioconductor.org/packages/release/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

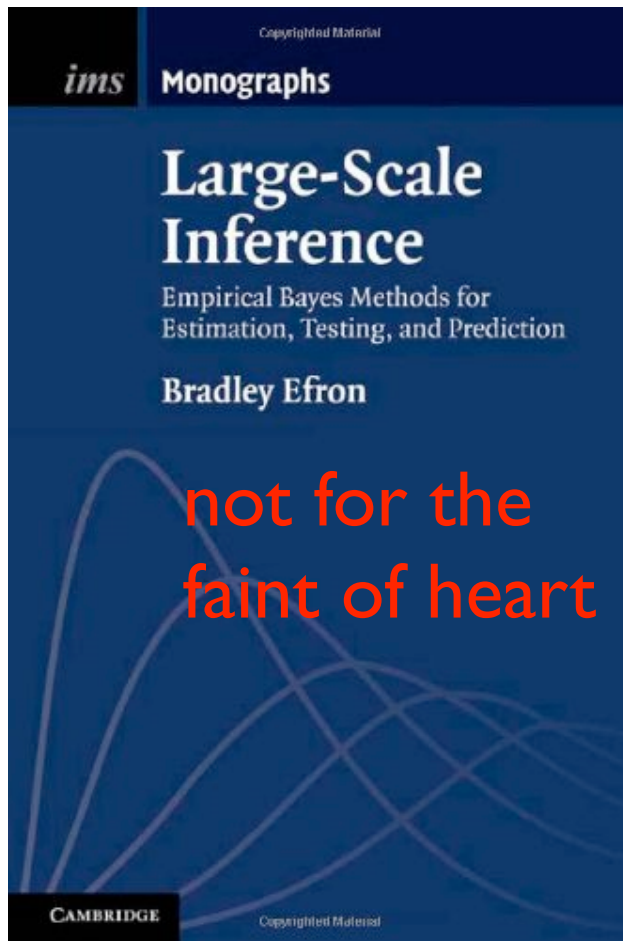
13.2 Fitted Model Objects p. 61

13.3 Multiple Testing Across Contrasts p. 62

Ch. 16 and Ch. 17: Case Studies

Large-Scale Inference
Empirical Bayes Methods for Estimation, Testing, and Prediction
Bradley Efron
Cambridge University Press 2010

Book related to course (Stats 329) at Stanford, so check [class website](#) for PDFs that look very much like ... chapters of this book



The Future of Statistics

Bradley Efron
Stanford University

quite accessible and
mercifully short

“Strange, as one gets older you’re expected to know more about the future.”

An essay found here:

<http://www-stat.stanford.edu/~ckirby/brad/other/2009Future.pdf>

and described as a piece written
for the Encyclopedia of Statistics

The approach requires two matrices to be specified. The first is the *design matrix*, which provides a representation of the different RNA targets that have been hybridized to the arrays. The second is the *contrast matrix*, which allows the coefficients defined by the design matrix to be combined into contrasts of interest. Each contrast corresponds to a comparison of interest between the RNA targets. For very simple experiments, the contrast matrix may not need to be specified explicitly.

$$Y_g = X_g \alpha_g + \varepsilon_g \quad \beta_g = C^T \alpha_g$$

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

$$C^T \begin{bmatrix} \mu_{g1} \\ \mu_{g2} \\ \mu_{g3} \end{bmatrix} = \beta_g$$

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

$$\begin{bmatrix} y_{g1} \\ y_{g2} \\ \vdots \\ y_{gn_g} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{g1} \\ \mu_{g2} \\ \mu_{g3} \\ \mu_{g4} \end{bmatrix} + \begin{bmatrix} \varepsilon_{g1} \\ \varepsilon_{g2} \\ \vdots \\ \varepsilon_{gn_g} \end{bmatrix}$$

4 groups
 “cell means” or “groups means” parametrization
 4 parameters in linear model

Less silly example that
 shows why one might
 want to form contrasts.

$$\beta_g = C^T \alpha_g$$

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_{g1} \\ \mu_{g2} \\ \mu_{g3} \\ \mu_{g4} \end{bmatrix} = \begin{bmatrix} \mu_{g2} - \mu_{g1} \\ \mu_{g3} - \mu_{g1} \\ \mu_{g4} - \mu_{g1} \\ \mu_{g3} - \mu_{g2} \\ \mu_{g4} - \mu_{g2} \\ \mu_{g4} - \mu_{g3} \end{bmatrix}$$

$$C^T \begin{bmatrix} \mu_{g1} \\ \mu_{g2} \\ \mu_{g3} \\ \mu_{g4} \end{bmatrix} = \beta_g$$

Imagine you are interested in all six pairwise comparisons between groups.
 No valid parametrization of the linear model will deliver that (too many parameters; linear dependence of those parameters).
 Therefore, forming contrasts is unavoidable.

limma was developed in an era when two-channel microarrays were the dominant platform (though one-channel arrays were already popular as well)






to accommodate both and, frankly, to deal with two-channel arrays at all, limma left it to the user to specify her design matrix ...

then assumes she will usually need to use a contrast matrix to use the fitted model results to produce estimates and statistics for the parameters she actually cares about

it IS still relevant today

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

"The greater the ignorance the greater the dogmatism."

— William Osler

"In truth, there are only two kinds of people; those who accept dogma and know it, and those who accept dogma and don't know it."

— G.K. Chesterton

"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



"Everything you've learned in school as "obvious" becomes less and less obvious as you begin to study the universe. For example, there are no solids in the universe. There's not even a suggestion of a solid. There are no absolute continuums. There are no surfaces. There are no straight lines."

— Richard Buckminster Fuller

```

> 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)
```

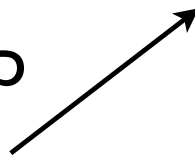
```
> summary(jFit)
```

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

limma workflow

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

- Projects, assignments, paper critiques:
 - Paper critique example will be available later today!
 - Groups have been assigned “advisors”, see who joined your repos
 - Assignment—need to know the lecture and seminar material!
- Next week:
 - Monday is BC’s “family day” (no classes)
 - Wednesday: guest lecture by Dr. Bernard Ng (correcting for multiple testing & introduction to analysis of imaging data)
- Go through the rest of the slides, **and the seminar for today**, to get familiar with “topTable” and “eBayes” for summarizing your results!

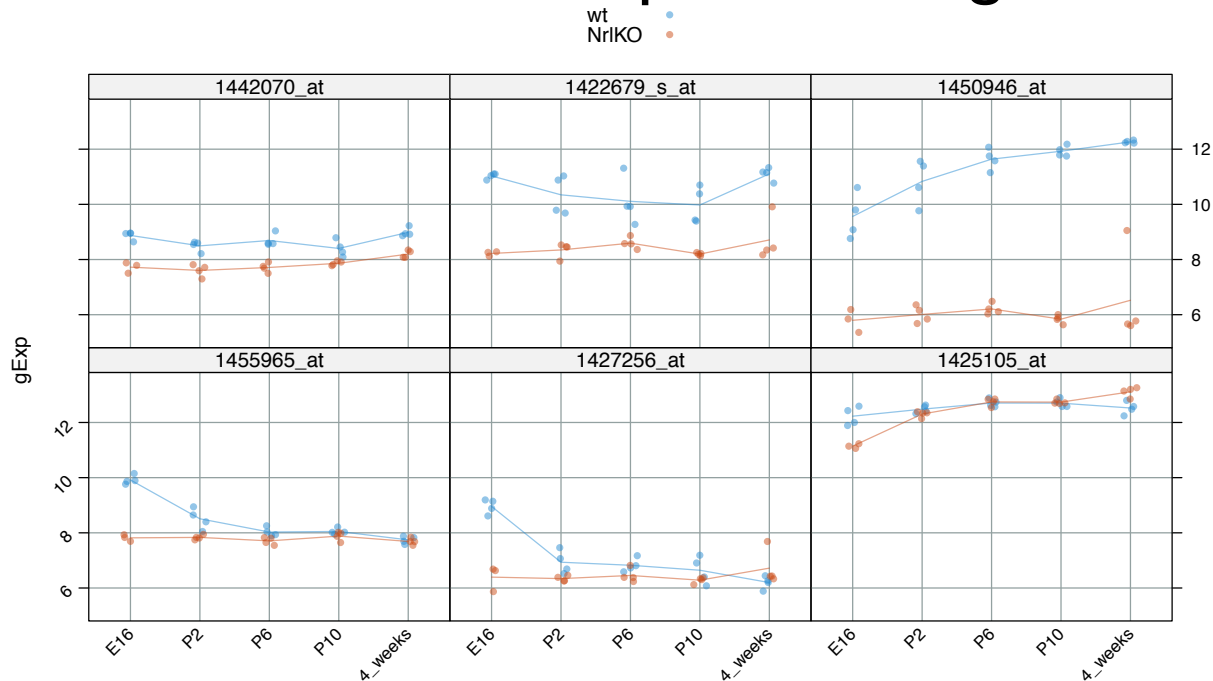
```
> ## this is equivalent: topTable(ebFit, coef = 2)
> ## but much more self-documenting; USE NAMES!
> ## you will be glad you did when you revisit/read your code
```

```
> topTable(ebFit, coef = "gTypeNr1KO")
```

	ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
27064	1455965_at	-2.099167	8.125436	-14.808462	6.702431e-16	2.007311e-11	23.349723
8288	1427256_at	-2.563917	6.784538	-9.572804	6.316787e-11	9.459072e-07	14.131751
6744	1425105_at	-1.084167	12.502308	-7.619877	1.084129e-08	8.070340e-05	9.622431
18296	1442070_at	-1.149417	8.268231	-7.546188	1.328641e-08	8.070340e-05	9.440861
4866	1422679_s_at	-2.810333	9.495128	-7.484259	1.577053e-08	8.070340e-05	9.287671
23505	1450946_at	-3.764833	8.733000	-7.421078	1.879238e-08	8.070340e-05	9.130832
4843	1422643_at	-1.813083	7.965949	-7.419730	1.886286e-08	8.070340e-05	9.127482
24384	1452114_s_at	-1.702000	6.519538	-7.135296	4.175503e-08	1.563152e-04	8.414647
6606	1424894_at	1.166500	6.961487	6.926615	7.519817e-08	2.384014e-04	7.884984
21504	1448076_at	1.140500	6.453897	6.906506	7.960246e-08	2.384014e-04	7.833658

devStage	E16	P2	P6	P10	4_weeks
gType					
wt	θ	β_{P2}	β_{P6}	β_{P10}	β_{4_weeks}
Nr1KO	τ_{Nr1KO}	$(\tau\beta)_{Nr1KO,P2}$	$(\tau\beta)_{Nr1KO,P6}$	$(\tau\beta)_{Nr1KO,P10}$	$(\tau\beta)_{Nr1KO,4_weeks}$

this finds genes where the knockouts differ from the wild types at the reference developmental stage level E16



```

> colnames(coef(ebFit))          # parameters in the linear model
[1] "(Intercept)"                 "gTypeNrlKO"                 "devStageP2"
[4] "devStageP6"                  "devStageP10"                "devStage4_weeks"
[7] "gTypeNrlKO:devStageP2"       "gTypeNrlKO:devStageP6"      "gTypeNrlKO:devStageP10"
[10] "gTypeNrlKO:devStage4_weeks"
> grep("gType", colnames(coef(ebFit))) # isolate the genotype terms
[1] 2 7 8 9 10
> hits3 <- topTable(ebFit,
+                   coef = grep("gType", colnames(coef(ebFit))),
+                   n = Inf)
> stripplotIt(hitDat <- prepareData(c(head(rownames(hits3), 3),
+                                       tail(rownames(hits3), 3))),
+             scales = list(rot = c(45, 0)))

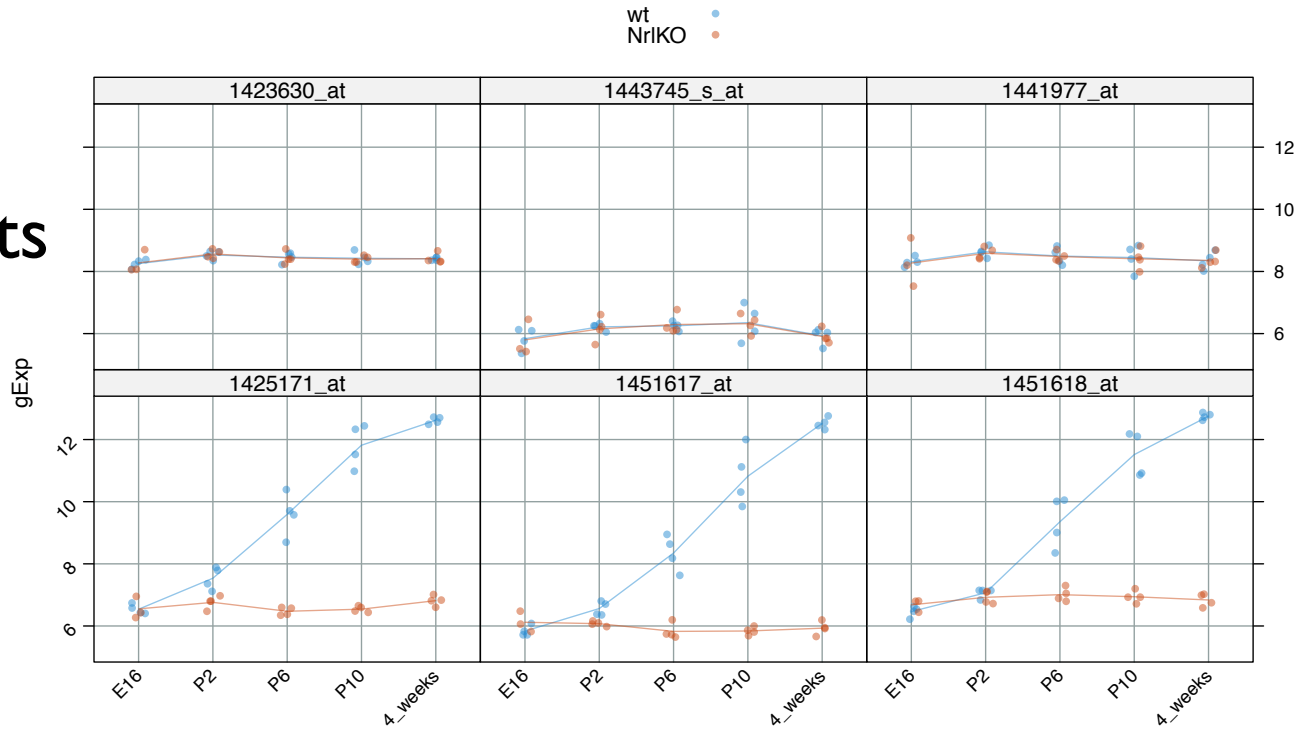
```

devStage	E16	P2	P6	P10	4_weeks
gType					
wt	θ	β_{P2}	β_{P6}	β_{P10}	β_{4_weeks}
NrlKO	τ_{NrlKO}	$(\tau\beta)_{NrlKO,P2}$	$(\tau\beta)_{NrlKO,P6}$	$(\tau\beta)_{NrlKO,P10}$	$(\tau\beta)_{NrlKO,4_weeks}$

this finds genes where genotype makes a difference,
somehow, somewhere

not hits

hits



```
> colnames(coef(ebFit))                # parameters in the linear model
[1] "(Intercept)"                       "gTypeNr1KO"                "devStageP2"
[4] "devStageP6"                        "devStageP10"               "devStage4_weeks"
[7] "gTypeNr1KO:devStageP2"             "gTypeNr1KO:devStageP6"     "gTypeNr1KO:devStageP10"
[10] "gTypeNr1KO:devStage4_weeks"
```

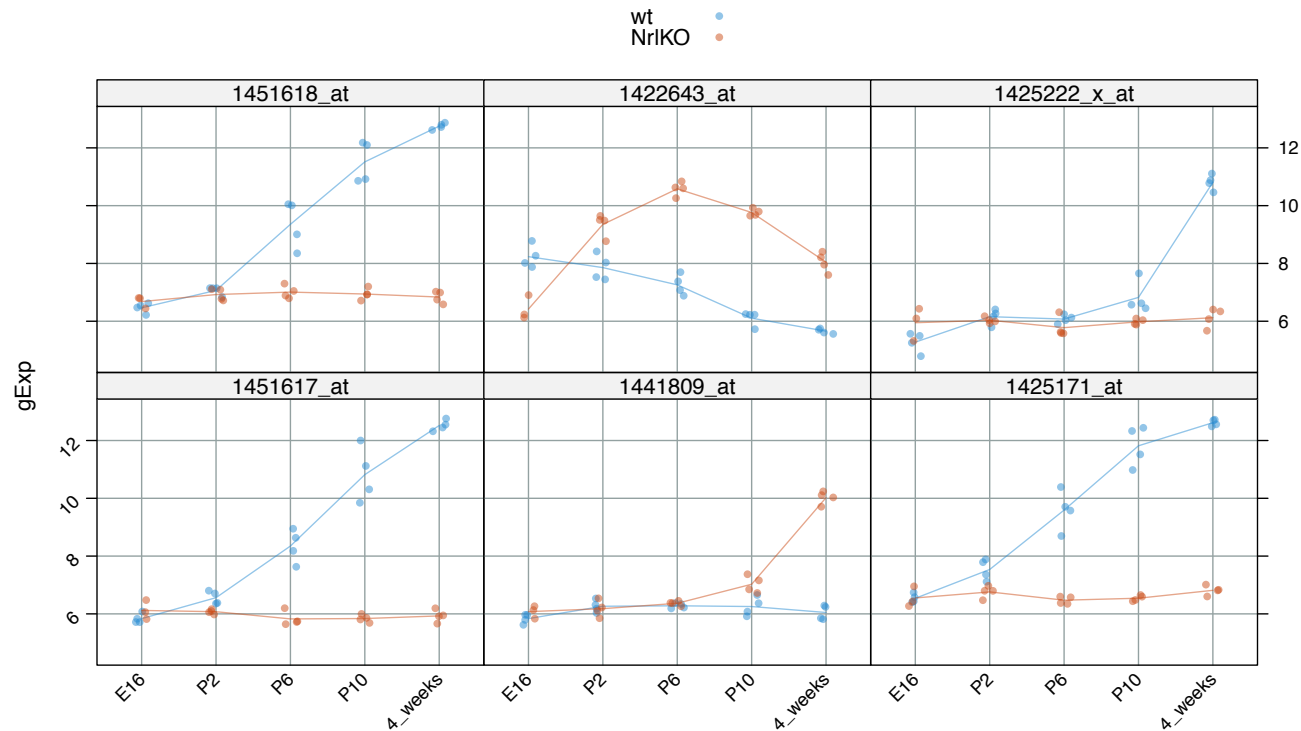
```
> grep(":", colnames(coef(ebFit)))      # isolate the interaction terms
[1] 7 8 9 10
```

```
> hits <- topTable(ebFit, coef = grep(":", colnames(coef(ebFit))))
```

```
> stripplotIt(hitDat <- prepareData(head(rownames(hits), 6)),
+             scales = list(rot = c(45, 0)))
```

devStage	E16	P2	P6	P10	4_weeks
gType					
wt	θ	β_{P2}	β_{P6}	β_{P10}	β_{4_weeks}
Nr1KO	τ_{Nr1KO}	$(\tau\beta)_{Nr1KO,P2}$	$(\tau\beta)_{Nr1KO,P6}$	$(\tau\beta)_{Nr1KO,P10}$	$(\tau\beta)_{Nr1KO,4_weeks}$

this finds genes where the interaction terms, *en masse*, are significant



```
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)
```

number: maximum number of genes to list

p.value: cutoff value for adjusted p-values. Only genes with lower p-values are listed.

lfc: minimum absolute log2-fold-change required. 'topTable' and 'topTableF' include only genes with (at least one) absolute log-fold-changes greater than 'lfc'. 'topTreat' does not remove genes but ranks genes by evidence that their log-fold-change exceeds 'lfc'.

By default 'number' probes are listed. Alternatively, by specifying 'p.value' and 'number=Inf', all genes with adjusted p-values below a specified value can be listed.

number lets you control size of hit list via #

p.value lets you control size of hit list via p-value cutoff

lfc lets you specify minimum observed effect size

```
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)
```

`sort.by`: character string specifying statistic to rank genes by. Possibilities for 'topTable' and 'topTableF' are '"logFC"', '"AveExpr"', '"t"', '"P"', '"p"', '"B"' or '"none"'. '"M"' is allowed as a synonym for '"logFC"' for backward compatibility. Other permitted synonyms are '"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' minus '"B"'.

`resort.by`: character string specifying statistic to sort the selected genes by in the output data.frame. Possibilities are the same as for 'sort.by'.

`sort.by` and `resort.by` give control over ordering

```
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)
```

TopTable output for all probes in original (unsorted) order can be obtained by `'topTable(fit,sort="none",n=Inf)'`. However `'write.fit'` or `'write'` may be preferable if the intention is to write the results to a file. A related method is `'as.data.frame(fit)'` which coerces an `'MArrayLM'` object to a `data.frame`.

ways to get everything in original row/gene order, for various purposes

```
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)
```

`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"`.

`adjust.method` specifies if/how to adjust p-value for multiple testing ... learn about that in next lecture


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

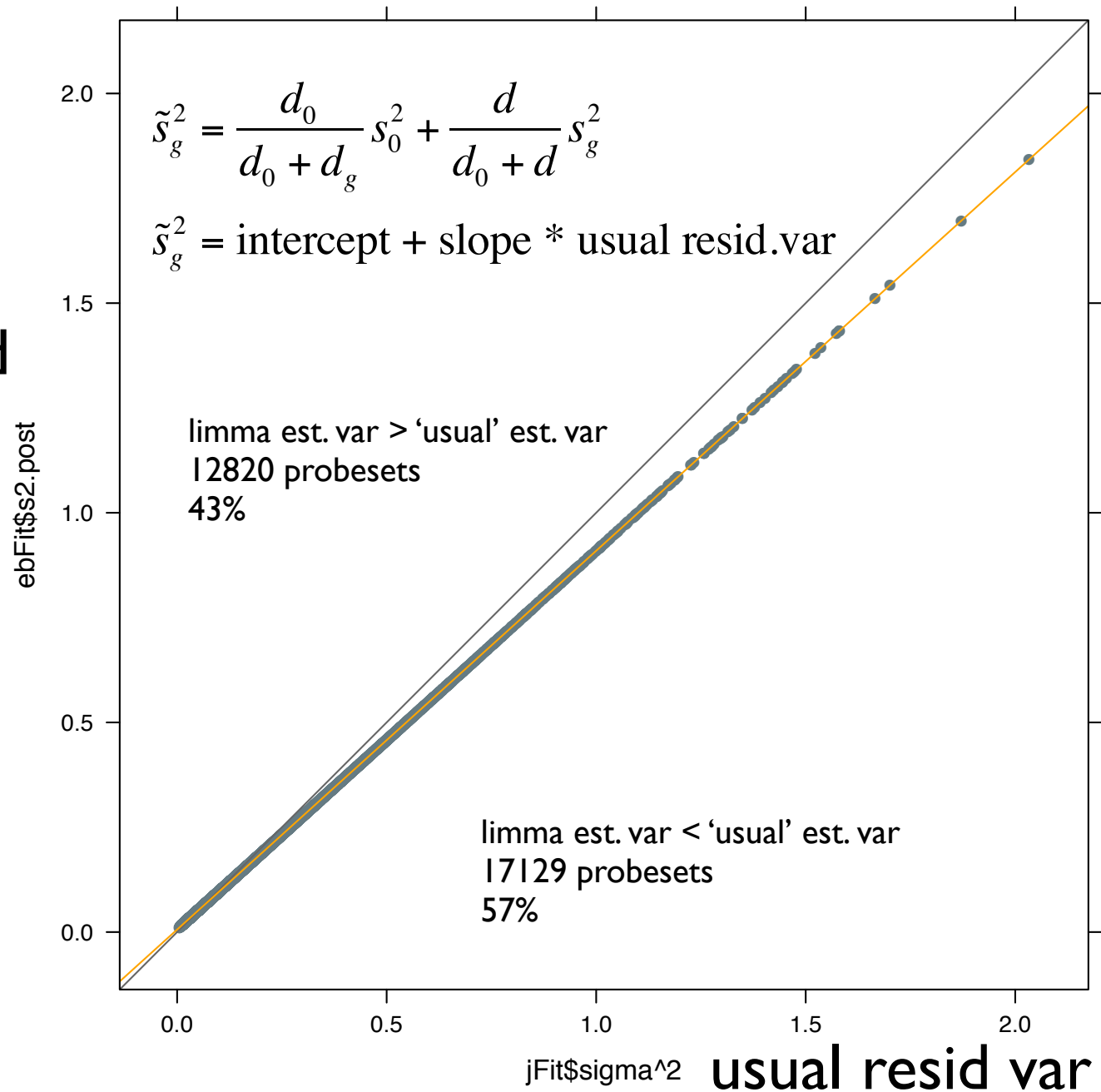
connecting some of the stuff in `jFit` and `ebFit`
with the underlying theory

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

math	plain English	R/limma	value in current example
s_g^2	the usual residual variance	<code>jFit\$sigma^2</code>	30K numbers!
d	the usual residual degrees of freedom, $n - p$	<code>jFit\$df.residual</code>	39 - 10 = 29*
s_0^2	mean of inverse Chi-square prior for the s_g^2	<code>ebFit\$s2.prior</code>	0.0616
d_0	degrees of freedom for the prior	<code>ebFit\$df.prior</code>	3.1026
\tilde{s}_g^2	the posterior mean, i.e. moderated residual variance	<code>ebFit\$s2.post</code>	30K numbers!

* limma can handle more complicated models where this is not the same for each gene, so this is actually a vector holding 30K copies of “29”

limma's
posterior,
moderated
resid var



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

$$\tilde{s}_g^2 = \text{intercept} + \text{slope} * \text{usual resid.var}$$

limma's
posterior,
moderated
resid var

