STAT 540
Class meeting 07
Monday, January 26, 2015

Dr. Gabriela Cohen Freue
Department of Statistics

Based on previous preparation by Dr. Jennifer (Jenny) Bryan

Comparing *more than two* groups

Linear Models with R by Julian Faraway, Chapman & Hall/CRC Texts in Statistical Science, 2004.

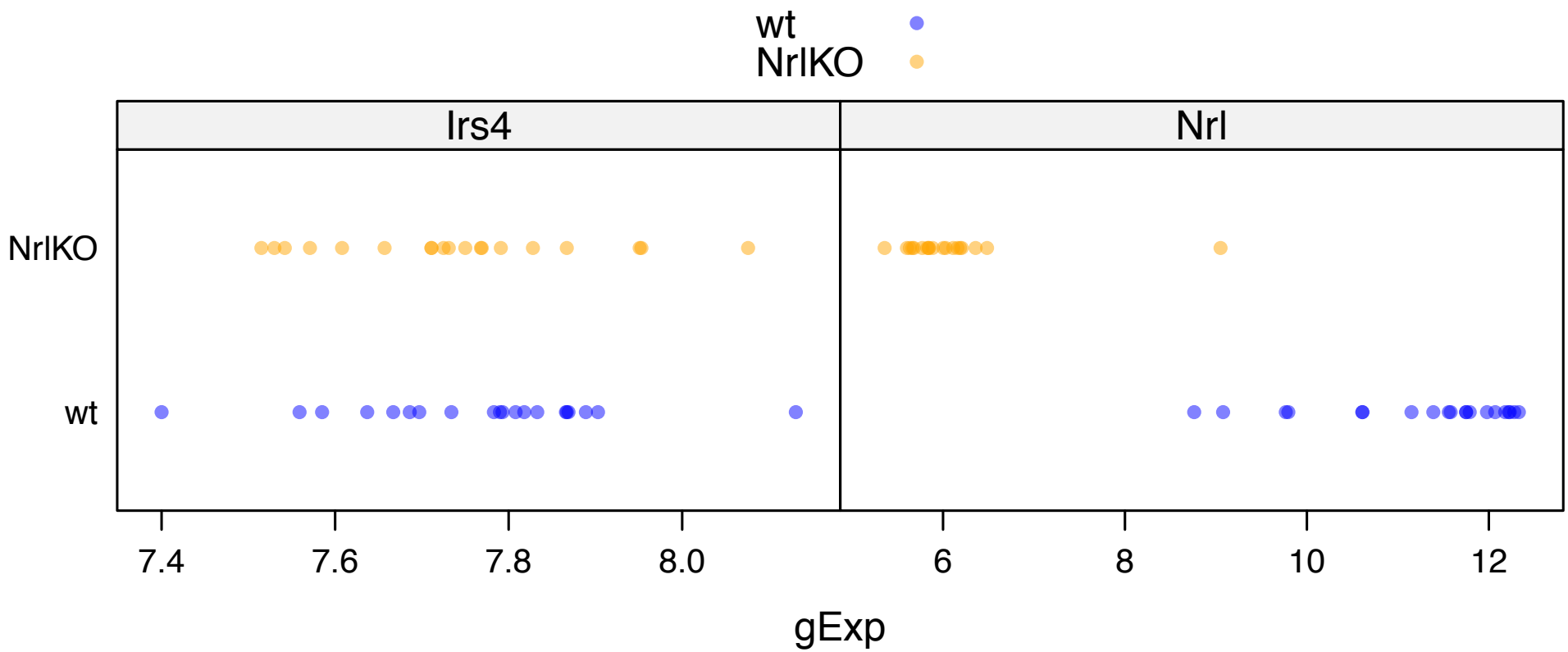
One can find a related “eBook” or “PDF book” -- entitled “Practical Regression and Anova using R” -- in various places on the web. It seems to be an earlier, but very mature draft of the official book.

Applied Linear Statistical Models by Neter, Kutner, Nachtsheim, Wasserman. 4th ed, Irwin, 1996. (There is a more recent 5th edition.)

Venables WN, Ripley BD (2002) Modern applied statistics with S. Springer.

<u>An Introduction to R</u> (an "official" R document)

Do we think the orange's and blue's are generated by different underlying distributions?

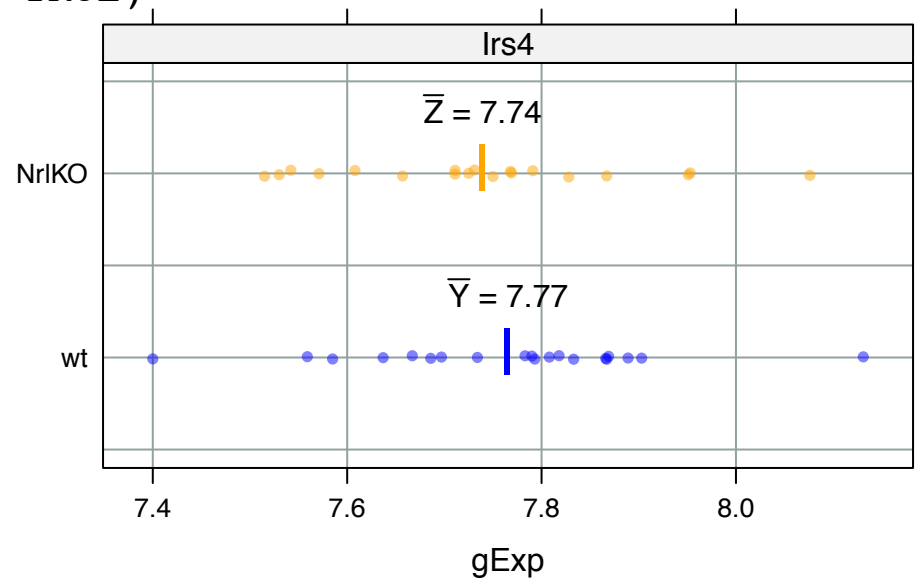


Irs4 (insulin receptor substrate 4) was selected at random as a boring non differentially expressed gene; NrlKO ~= wt

Nrl (neural retina leucine zipper gene) is the gene that was knocked out in half the mice; obviously should be differentially expressed; NrlKO << wt

```
> t.test(gExp ~ gType, miniDat,
+        subset = gene == "Irs4", var.equal = TRUE)
```

two sample t test




```
> summary(aov(gExp ~ gType, miniDat,
+             subset = gene == "Irs4"))
```

(one-way) analysis of variance
"ANOVA"

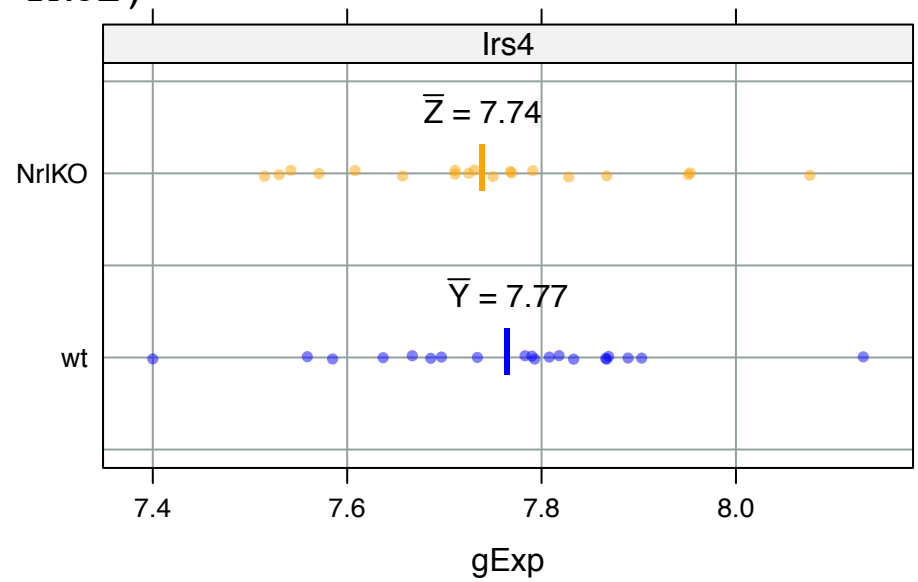
```
> summary(lm(gExp ~ gType, miniDat,
+            subset = gene == "Irs4"))
```

linear model
linear regression

```
> t.test(gExp ~ gType, miniDat,
+        subset = gene == "Irs4", var.equal = TRUE)

    Two Sample t-test

data:  gExp by gType
t = 0.5286, df = 37, p-value = 0.6002
<snip, snip>
sample estimates:
   mean in group wt mean in group NrlKO
           7.765750           7.739684
```




```
> summary(aov(gExp ~ gType, miniDat,
+             subset = gene == "Irs4"))
            Df Sum Sq Mean Sq F value Pr(>F)
gType        1 0.0066 0.00662   0.279    0.6
Residuals   37 0.8764 0.02369
```

```
7.739684 - 7.765750 = -0.026066
```

```
-0.5286494 ^ 2 = 0.2794702
```

```
> summary(lm(gExp ~ gType, miniDat,
+            subset = gene == "Irs4"))
<snip, snip>
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.76575    0.03441 225.650   <2e-16 ***
gTypeNrlKO  -0.02607    0.04931  -0.529      0.6
<snip, snip>
F-statistic: 0.2795 on 1 and 37 DF,  p-value: 0.6002
```

These are not coincidences!

The two sample t test is a special case of "analysis of variance" or "ANOVA", where the only difference is two groups vs. potentially more than two groups.

"Analysis of variance" or "ANOVA" is a special case of a linear model or linear regression, where the only real difference is categorical covariates only vs. potentially including quantitative covariates. There are also different in conventions around reporting results.

Given that you may want to model complex data, I recommend:

- get comfortable with linear models and view "group comparisons" as a special case

My model statements are going to have a different look. Instead of:

$$Y \sim F$$

I'm going to focus more on decomposing a rv into its mean and variability around it's mean:

$$Y = \mu + \varepsilon, \text{ where } \varepsilon \sim F, E(\varepsilon) = 0$$

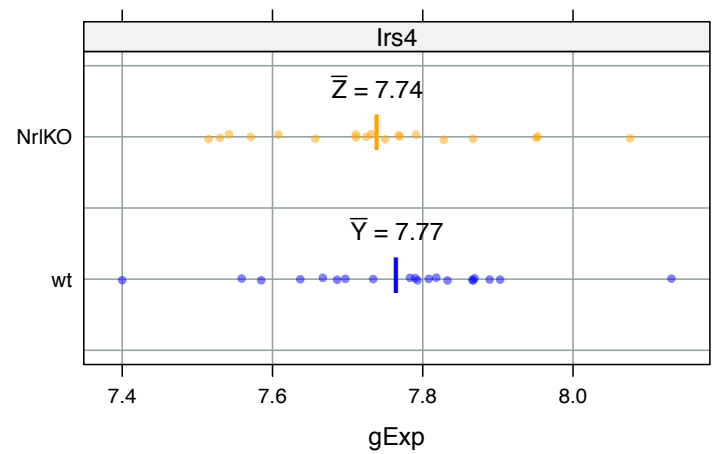
(And then we start to decompose the mean into bits, such as a reference level plus other bits that are related to covariates .....)

Notational change:

In our running example, I used Y and Z to denote the random variables corresponding to some quantity we might observe for subjects in two groups.

One group, e.g. wild type ... Y.

Other group, e.g. Nrl knockouts ... Z.



Change of plan:

We're going to follow statistical convention for regression and use Y for a variable we observe and regard as a response (like before) and X will be associated with the variables we regard as predictors or explanatory variables, e.g. the distinction between wild type and knockouts.

Imagine we are studying the response *Y* (e.g., gene expression level) for two or more groups (e.g., treatments, populations), identified by *j*:

$$Y_j = \mu_j + \varepsilon_j, \text{ where } \varepsilon_j \sim F, E(\varepsilon_j) = 0$$

Note how we allow for different expected values of Y for each treatment -- those are the μ_j's

For example, the gene expression of the healthy group:

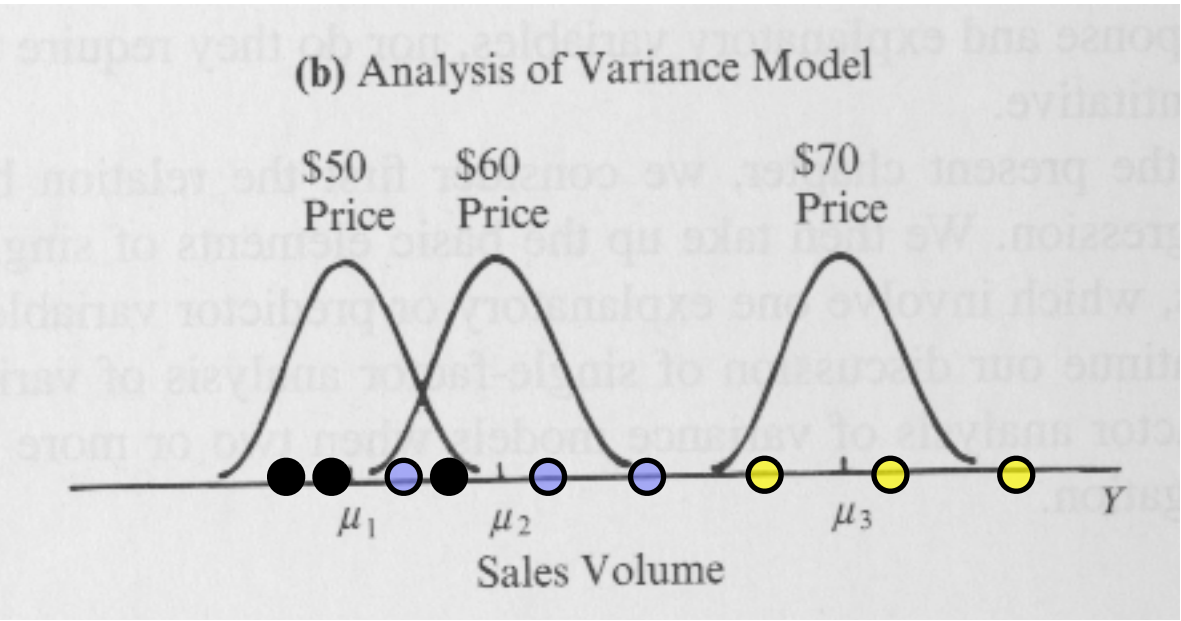
$$Y_1 = \mu_1 + \varepsilon_1, \quad \varepsilon_1 \sim F \quad E(\varepsilon_1) = 0$$

$$Y_1 \sim F, \quad E(Y_1) = \mu_1$$

*We assume that the noise -- denoted ε_j -- has a common distribution across the groups. This can certainly be relaxed but it's a nice assumption to make if you can get away with it.

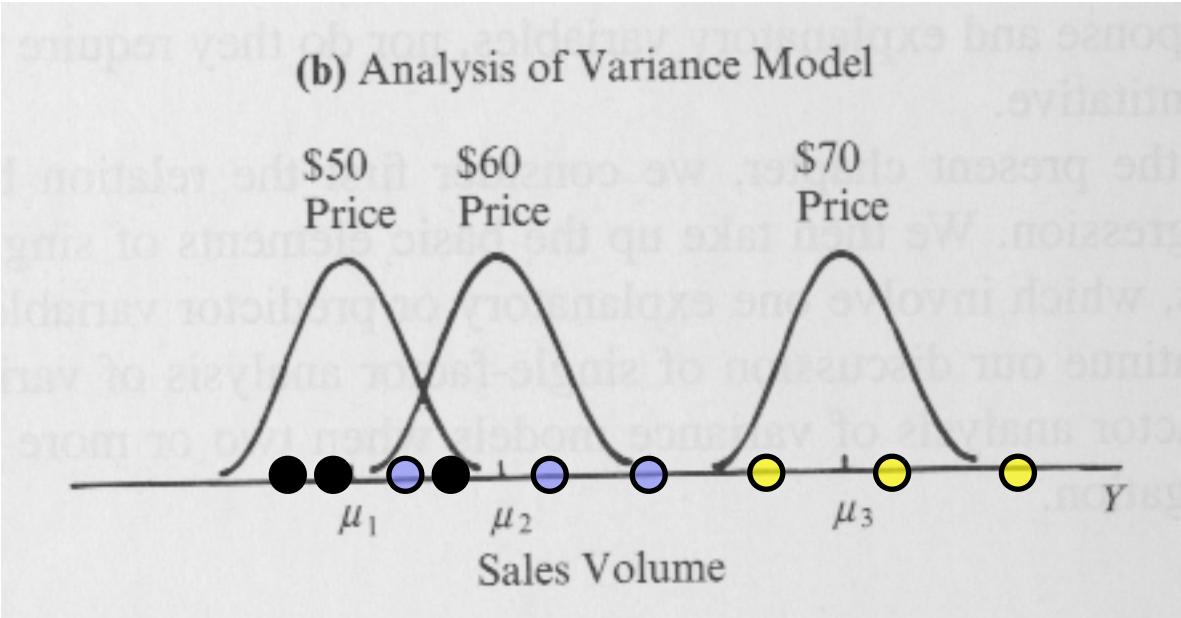
Our observed data will be observations of the Y_j, where we assume independence across observations. Individual observations or experimental units are denoted by *i*:

$$Y_{ij} = \mu_j + \varepsilon_{ij}, \text{ where } \varepsilon_{ij} \sim F, E(\varepsilon_{ij}) = 0$$



This figure and next several from Applied Linear Statistical Models by Neter, Kutner, Nachtsheim, Wasserman. 4th ed, Irwin, 1996.

$$Y_{ij} = \mu_j + \varepsilon_{ij}, \text{ where } \varepsilon_{ij} \sim F, E(\varepsilon_{ij}) = 0$$

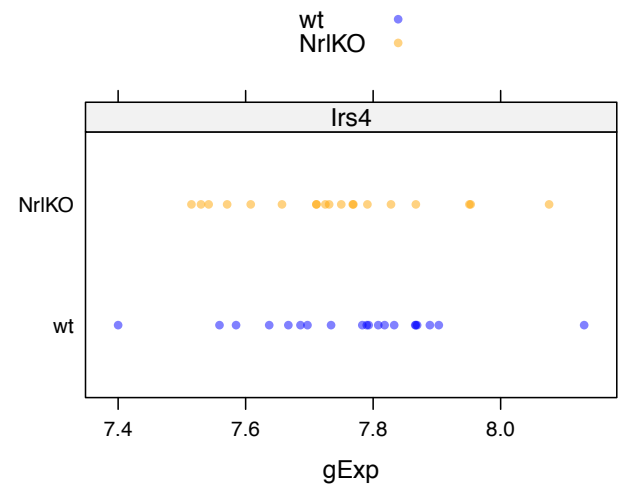


We will, of course, want to know if all the μ_j are the same or not and, if not, which ones are different.

That will be judged based on whether observed differences in sample averages are large based on the apparent background variability.

$$Y_{ij} = \mu_j + \varepsilon_{ij}, \text{ where } \varepsilon_{ij} \sim F, E(\varepsilon_{ij}) = 0$$

$$\begin{bmatrix} Y_{11} \\ \vdots \\ Y_{n_1 1} \\ Y_{12} \\ \vdots \\ Y_{n_2 2} \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \vdots \\ \varepsilon_{n_1 1} \\ \varepsilon_{12} \\ \vdots \\ \varepsilon_{n_2 2} \end{bmatrix}$$



I constructed this vector "by hand" -- whenever the Y_{ij} is from group 1, I put in μ_1, and when Y_{ij} is from group 2, I put in μ_2.

For mathematical and computational reasons, a matrix formulation is advantageous.

$$Y_{ij} = \mu_j + \varepsilon_{ij}, \text{ where } \varepsilon_{ij} \sim F, E(\varepsilon_{ij}) = 0$$

$$\begin{bmatrix} Y_{11} \\ \vdots \\ Y_{n_1 1} \\ Y_{12} \\ \vdots \\ Y_{n_2 2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \vdots \\ \varepsilon_{n_1 1} \\ \varepsilon_{12} \\ \vdots \\ \varepsilon_{n_2 2} \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \vdots \\ \varepsilon_{n_1 1} \\ \varepsilon_{12} \\ \vdots \\ \varepsilon_{n_2 2} \end{bmatrix}$$

Example of a “design matrix”; often denoted by X in the statistical world.

the column vector of the responses
one element per experimental unit

a column vector
of the errors

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

a (design) matrix that represents covariate
info, one row per experimental unit

a column vector of the parameters in the
linear model

Generic linear model, using conventional matrix formulation

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

$$X \approx \begin{bmatrix} \text{group 1} \\ \text{group 1} \\ \text{group 1} \\ \text{group 2} \\ \text{group 2} \\ \text{group 2} \\ \text{group 3} \\ \text{group 3} \\ \text{group 3} \end{bmatrix} \quad \text{or} \quad X \approx \begin{bmatrix} \text{untreated} \\ \text{untreated} \\ \text{untreated} \\ \text{treatment A} \\ \text{treatment A} \\ \text{treatment A} \\ \text{treatment B} \\ \text{treatment B} \\ \text{treatment B} \end{bmatrix}$$

How are we going to make the "design matrix" now? Obviously it needs to be numbers!

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

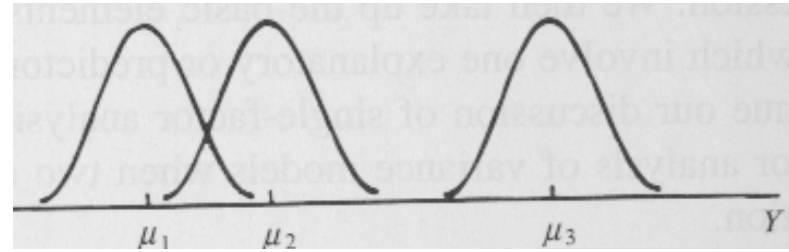
The exact form of the design matrix X and the parameter alpha are not uniquely defined. The user has some control. The two objects are tightly related to each other. This will become much more clear in examples.

$$\begin{bmatrix} Y_{11} \\ \vdots \\ Y_{n_1 1} \\ Y_{12} \\ \vdots \\ Y_{n_2 2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \vdots \\ \varepsilon_{n_1 1} \\ \varepsilon_{12} \\ \vdots \\ \varepsilon_{n_2 2} \end{bmatrix}$$

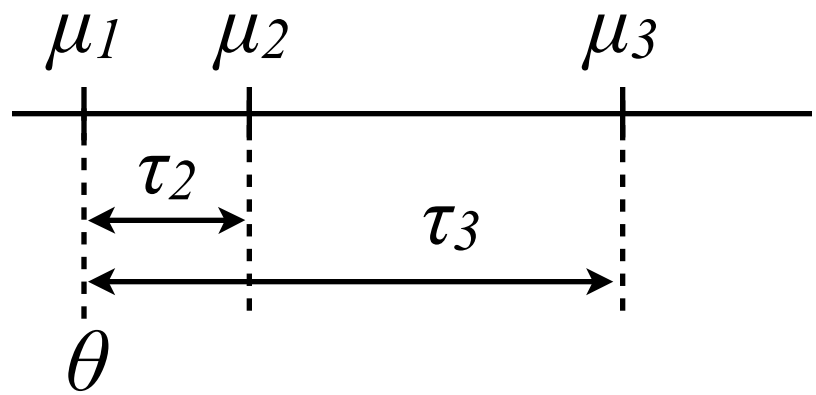
$$Y = X\alpha + \varepsilon$$

Here's an example of a design matrix X and parameter vector alpha that work together. But there are others!

ANOVA-style
"cell means" parametrization



$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

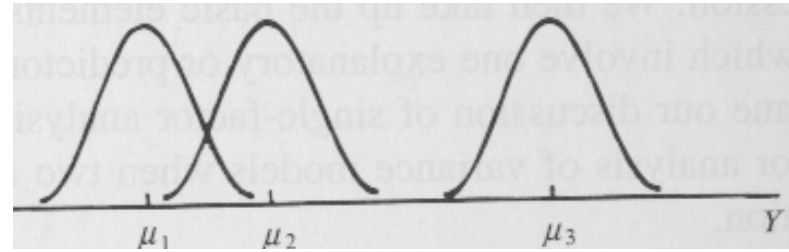


ANOVA-style
"reference + treatment effects"
parametrization

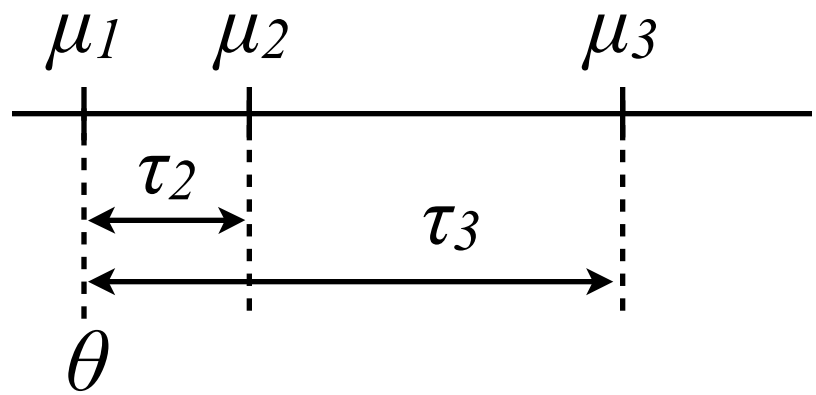
$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, \text{ where } \tau_1 = 0 \text{ by convention}$$

The same model is being used, under the hood, but it is represented -- "parametrized" -- differently. Different parametrizations are useful for different things.

ANOVA-style
"cell means" parametrization



$$Y_{ij} = \mu_j + \varepsilon_{ij}$$



ANOVA-style
"reference + treatment effects"
parametrization

$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, \text{ where } \tau_1 = 0 \text{ by convention}$$

Here's how we would represent the state of "all groups have same mean", in either parametrization:

$$\mu_1 = \mu_2 = \mu_3 \quad \Leftrightarrow \quad \tau_2 = \tau_3 = 0$$

ANOVA-style, "cell means"

$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

ANOVA-style, "ref + tx effects"

$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, (\tau_1 = 0)$$

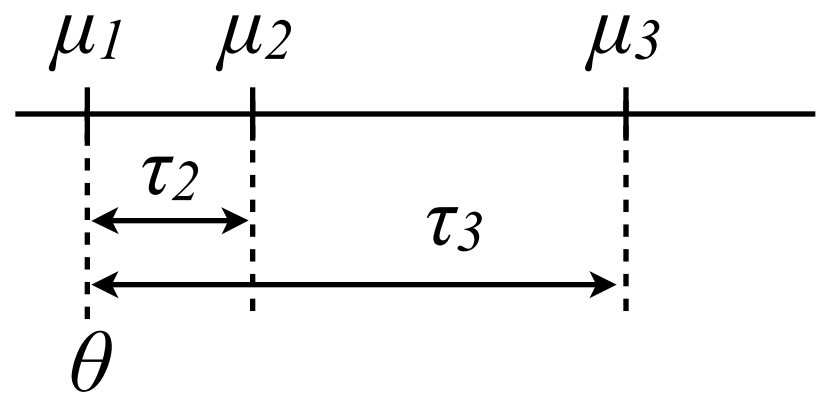
$$Y = X\alpha + \varepsilon$$

$$\begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{n_3 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \\ \vdots \\ \varepsilon_{n_3 3} \end{bmatrix}$$

$$\begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{n_3 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \\ \vdots \\ \varepsilon_{n_3 3} \end{bmatrix}$$

The design matrix specifies how the observed data relates to the regression parameters.

Note we can obtain one set of parameters from the others!



ANOVA-style, "cell means"

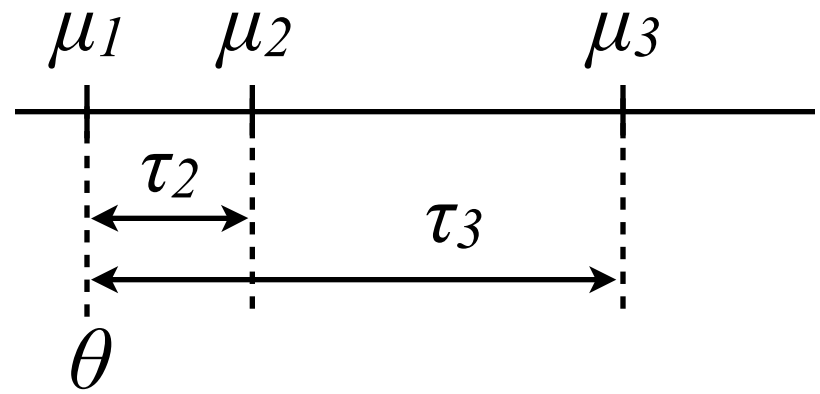
$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

$$\begin{aligned} \mu_1 &= \theta \\ \mu_2 &= \theta + \tau_2 \\ \mu_3 &= \theta + \tau_3 \end{aligned} \qquad \begin{aligned} \theta &= \mu_1 \\ \tau_2 &= \mu_2 - \mu_1 \\ \tau_3 &= \mu_3 - \mu_1 \end{aligned}$$

$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, (\tau_1 = 0)$$

ANOVA-style, "ref + tx effects"

We can do this neatly with matrix multiplication!
The matrices C below are sometimes called "contrast matrices".



ANOVA-style, "cell means"

$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

$$C^T \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix} = \mu$$

$$C^T \mu = \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, (\tau_1 = 0)$$

ANOVA-style, "ref + tx effects"

How works in practice using lm() in R

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

$$\downarrow$$

```
lm(y ~ x, data = jDat)
```

formula
y numeric
x factor

optional data.frame in which x and y are to be found (I recommend this style)

R formulas are expressed in 'Wilkinson-Rogers' notation. See Venables and Ripley 3.7 and 6.2 for an introduction. And/or read Ch. 11 of "An Introduction to R".

$$Y = X\alpha + \varepsilon$$ `lm(y ~ x, data = jDat)`

In most contexts, you can -- and should! -- just let R create the design matrix X for you.

How factors are "dummied out" is controlled by how you specify the model and the current "contrasts" setting in effect.

The path of least resistance will be "reference + treatment effects" (called "contr.treatment"; see ?options and ?contrasts and ?contr.treatment to learn more.)

If you really want to -- or must -- do it yourself, see `model.matrix()`. Also nice just for viewing and getting acquainted with the contrasts associated with a factor.

Vocabulary: **contrasts**

The word **contrasts** is used in stats for some distinct but closely related things. You've already seen that just now:

1. the "contrasts for a factor", i.e. specific choice of "dummying" out a factor in regression
2. a "contrast matrix" to map one set of parameters to another, to form linear combinations of parameters

the “contrasts for a factor”, i.e. specific choice of “dummying” out a factor in regression

This occurs on the “front end” of modelling, i.e. when specifying the model parametrization or, equivalently, when <u>specifying the contrasts for factor covariates</u> or, equivalently, when creating the design matrix.

ANOVA-style, "cell means"

$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

ANOVA-style, "ref + tx effects"

$$Y_{ij} = \theta + \tau_j + \varepsilon_{ij}, (\tau_1 = 0)$$

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

$$\begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ \\ y_{n_3 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \\ \vdots \\ \\ \varepsilon_{n_3 3} \end{bmatrix}$$

$$\begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ \\ y_{n_3 3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \tau_2 \\ \tau_3 \end{bmatrix} + \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{21} \\ \vdots \\ \\ \varepsilon_{n_3 3} \end{bmatrix}$$

```
lm(y ~ 0 + x, data = jDat)
lm(y ~ -1 + x, data = jDat)
```

```
lm(y ~ x, data = jDat)
```

Controlling parametrization (or the factor contrasts) via the model formula.

a “contrast matrix” to map one set of parameters to another, to form linear combinations of parameters

This occurs on the “back end” of modelling. Example, if a parameter you are interested in is not one of those being directly estimated, but it can be formed as a linear combination regression parameters, i.e. via a “contrast matrix”.

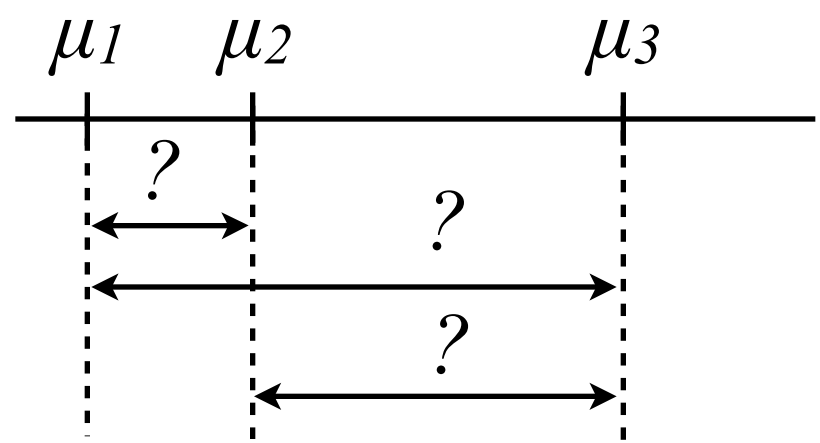
Typical use: to form a difference of group means.

ANOVA-style, "cell means"

$$Y_{ij} = \mu_j + \varepsilon_{ij}$$

Let's imagine you want to fit the model with a cell means parametrization.

But you also want to look at the differences between the cell means.



You could do that by multiplying the vector of parameter (or their estimates) by a "contrast matrix".

$$\begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = C^T \mu = \begin{bmatrix} \mu_2 - \mu_1 \\ \mu_3 - \mu_1 \\ \mu_3 - \mu_2 \end{bmatrix}$$

Why am I burdening you with this? Doesn't R and the lm() function, in particular, default to something reasonable?

Yes it does. But ...

1. Once you get beyond two group comparisons, you need to know a bit about how factors are utilized in linear models and what the resulting parameter estimates mean. One day you may even want to exert control on this.

Why am I burdening you with this? cont'd

2. A popular R package for performing linear modelling for thousands of, e.g., genes at once, while borrowing strength across the genes, is called limma (see later lectures). And, unlike lm(), limma does NOT make the design matrix for you. limma does not use the same formula interface as lm().

This is sad.

Why would you still want to use limma? Because it implements moderation of the t-statistics for regression parameters, using an empirical Bayes approach.

Why was limma written this way? For historical reasons, due to idiosyncrasies of two-channel microarrays.

UCSF Center for Bioinformatics & Molecular Biostatistics

Lecture 2:
Designed / factorial microarray experiments

Jean Yee Hwa Yang

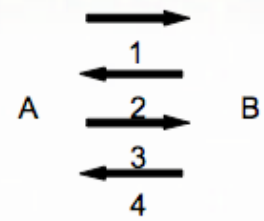
October 23, 2004
Genentech Hall Auditorium, Mission Bay, UCSF

Design matrices arising in two-channel microarray studies.

Rejoice that we do not need to do this much anymore!

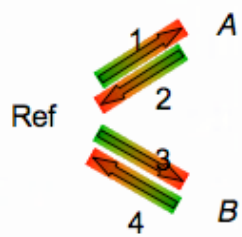
Examples of design matrix

$$Y = X\beta + \varepsilon$$



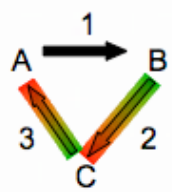
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \beta = \begin{pmatrix} \beta \\ -\beta \\ \beta \\ -\beta \end{pmatrix}$$

$$\beta \equiv B - A$$



$$\begin{pmatrix} A-\text{Ref} \\ \text{Ref}-A \\ \text{B-Ref} \\ \text{Ref-B} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 1 \\ -1 & -1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ -\beta_1 \\ \beta_1 + \beta_2 \\ -\beta_1 - \beta 2 \end{pmatrix}$$

$$\beta_1 \equiv A - \text{Ref}$$
$$\beta_2 \equiv B - A$$



$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

$$\beta_1 \equiv B - A$$
$$\beta_2 \equiv C - A$$

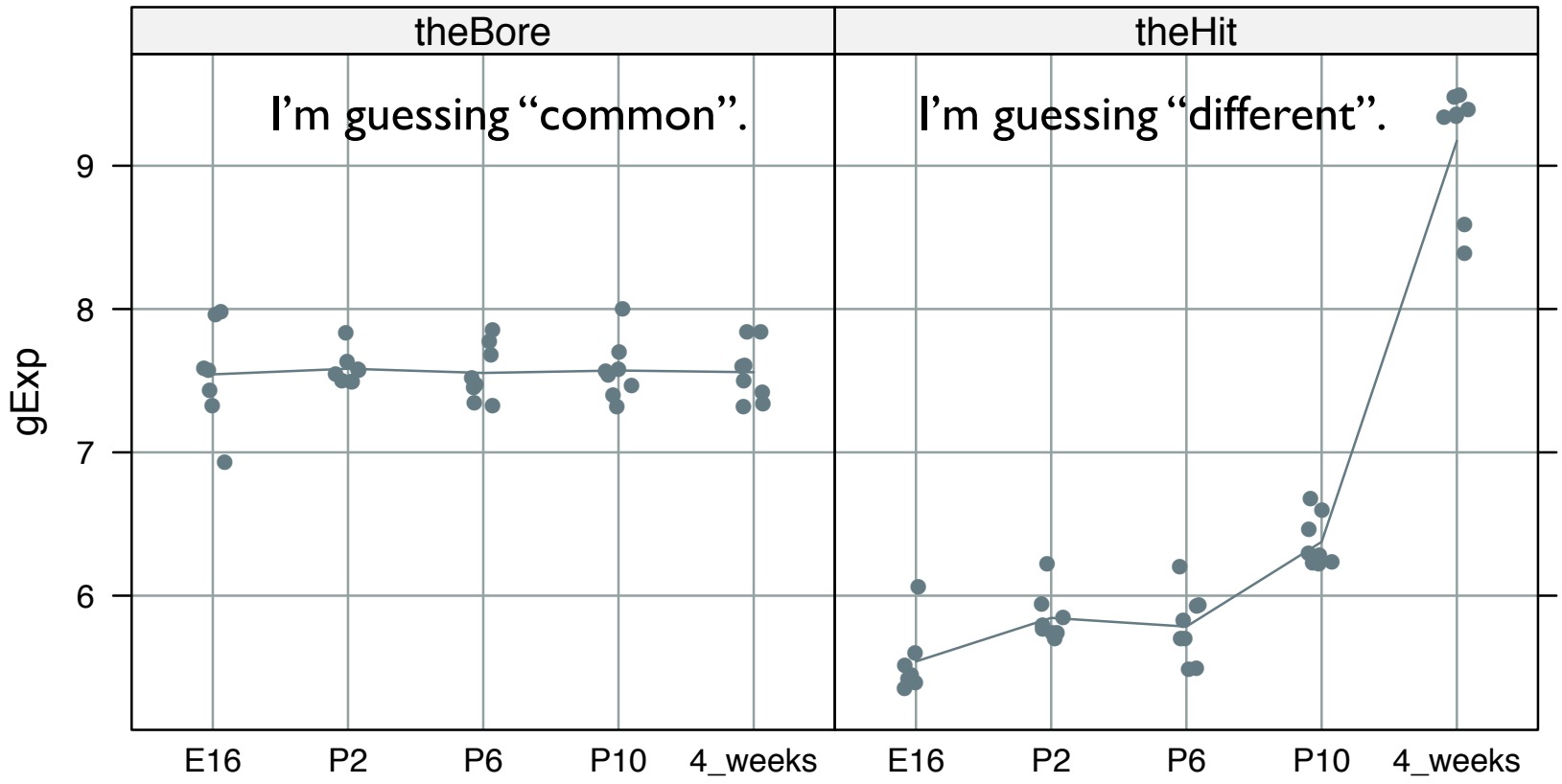
X = Design Matrix

The two-channel analytical puzzle spawned a cottage industry in microarray experimental design, where people figured out the best way to pair samples on arrays (ever heard of a “loop design”?).

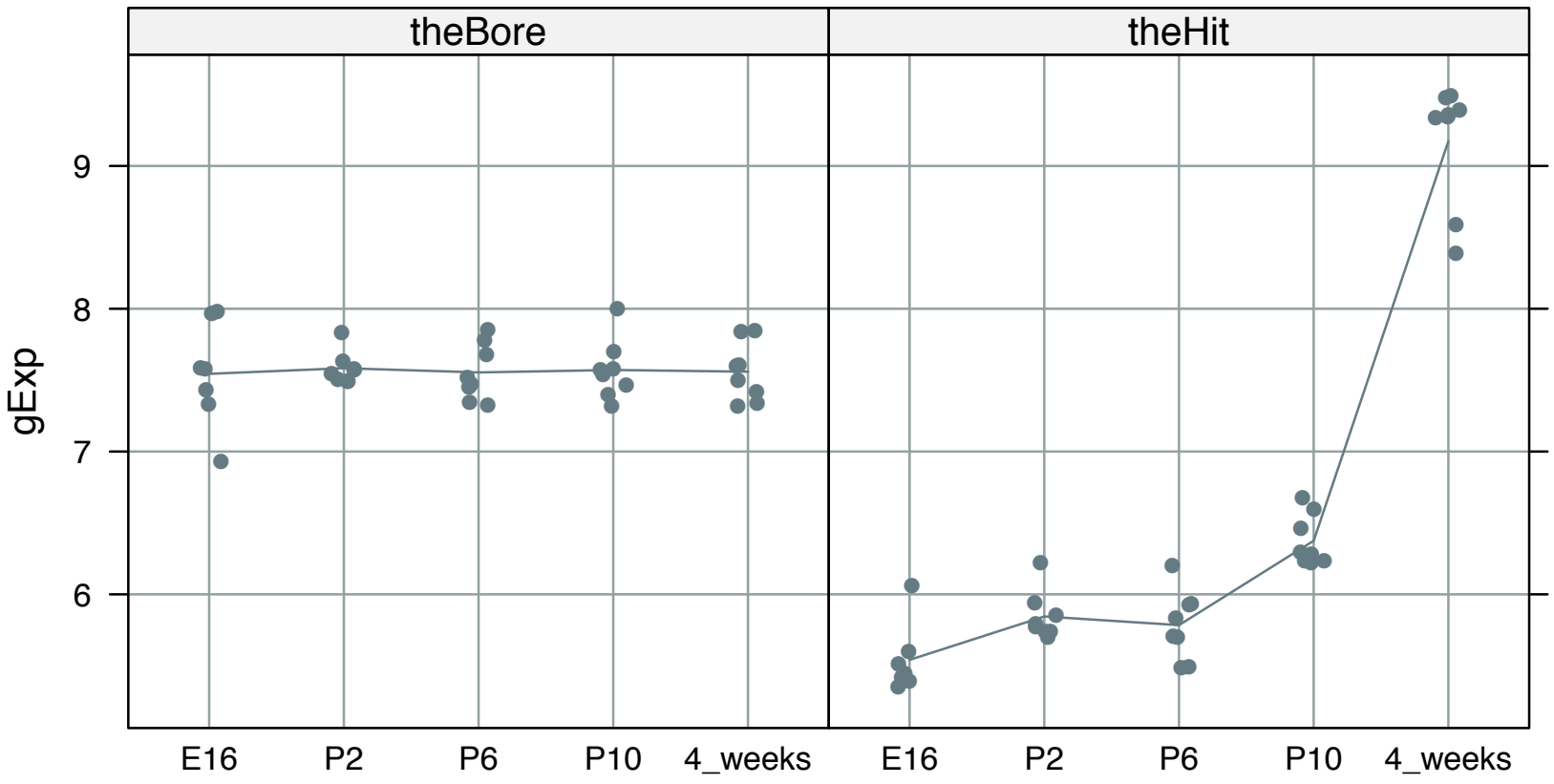
Luckily, this additional layer of book-keeping and experimental design difficulty is fading as single channel platforms achieve dominance (single channel microarrays and, more recently, mRNA-Seq etc.).

you’ve earned a nice relaxing look at some data!

Do we think the expression levels at different developmental stages are generated by different underlying distributions? Or a common one?



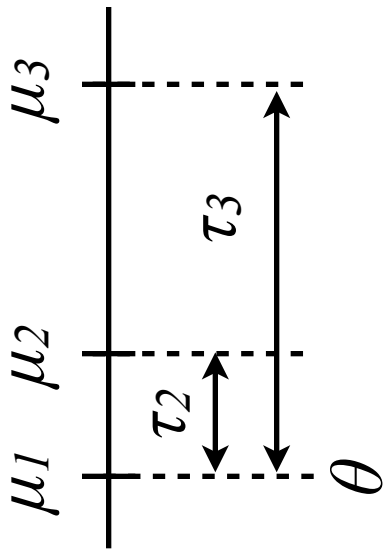
```
> with(miniDat,
+      tapply(gExp, list(devStage, gene), mean))
         theBore   theHit
E16     7.544143 5.540857
P2      7.583500 5.844875
P6      7.554000 5.784250
P10     7.571000 6.375125
4_weeks 7.559000 9.173375
```

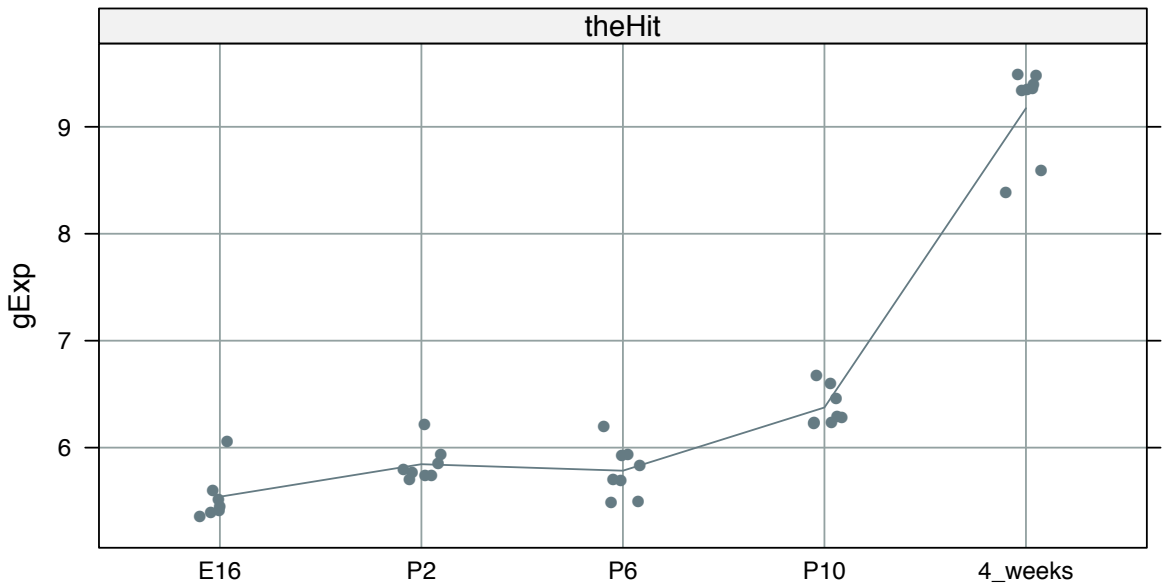


```
> data.frame(cellMeans = theHitAvgs,
+            txEffects = theHitAvgs - theHitAvgs[1])
        cellMeans txEffects
E16      5.540857 0.0000000
P2       5.844875 0.3040179
P6       5.784250 0.2433929
P10      6.375125 0.8342679
4_weeks  9.173375 3.6325179
```

the mu's = "cell means"
.... estimated by sample avg @ each devStage

(theta, the tau's) = ref + tx effects
.... estimated by (E16 avg, other avgs - E16 avg)

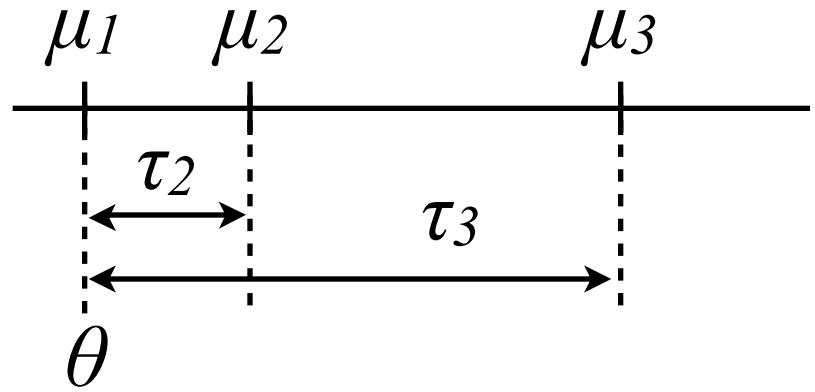




$$Y = X\alpha + \varepsilon$$

$$\alpha = (\theta, \tau_{P2}, \tau_{P6}, \tau_{P10}, \tau_{4\_weeks})$$

```
        cellMeans txEffects
E16      5.540857 0.0000000
P2       5.844875 0.3040179
P6       5.784250 0.2433929
P10      6.375125 0.8342679
4_weeks  9.173375 3.6325179
```

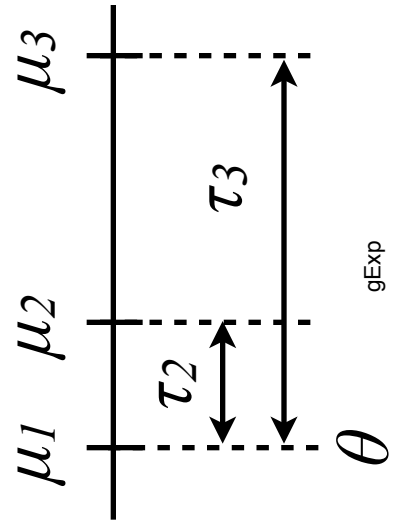


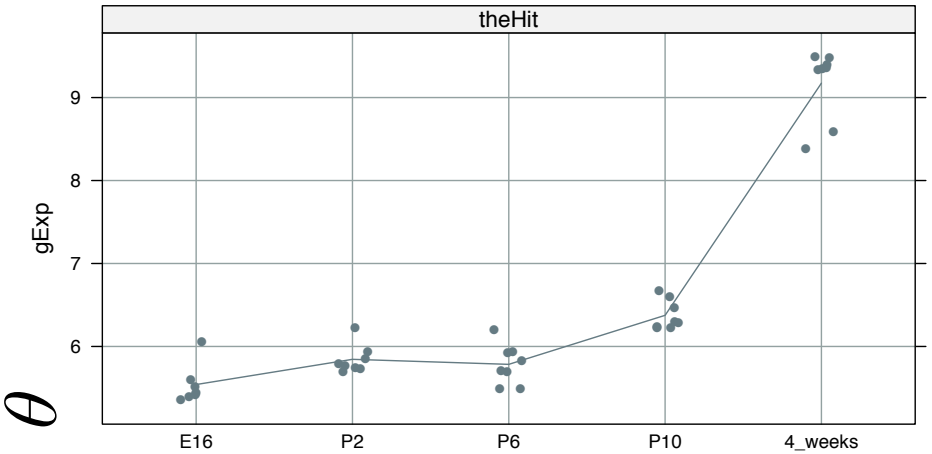
$$Y = X\alpha + \varepsilon$$

$$\alpha = (\theta, \tau_{P2}, \tau_{P6}, \tau_{P10}, \tau_{4\_weeks})$$

```
> hitFit <- lm(gExp ~ devStage, miniDat, gene == "theHit")

> summary(hitFit)$coef
                 Estimate Std. Error   t value     Pr(>|t|)
(Intercept)     5.5408571  0.1021381 54.248698 1.307554e-34
devStageP2      0.3040179  0.1398583  2.173756 3.678022e-02
devStageP6      0.2433929  0.1398583  1.740282 9.085489e-02
devStageP10     0.8342679  0.1398583  5.965093 9.559065e-07
devStage4_weeks 3.6325179  0.1398583 25.972843 5.266481e-24
```





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

$$\alpha = (\theta, \tau_{P2}, \tau_{P6}, \tau_{P10}, \tau_{4\_weeks})$$

in the context of this model we generally test null hypotheses of two types:

$H_0 : \tau_j = 0$

vs

$H_0 : \tau_j \neq 0$

for each j individually

$H_0 : \tau_j = 0$

vs

$H_0 : \tau_j \neq 0$

for all j at the same time

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

$$\alpha = (\theta, \tau_{P2}, \tau_{P6}, \tau_{P10}, \tau_{4\_weeks})$$

$H_0 : \tau_j = 0$

vs

$H_0 : \tau_j \neq 0$

for each j individually

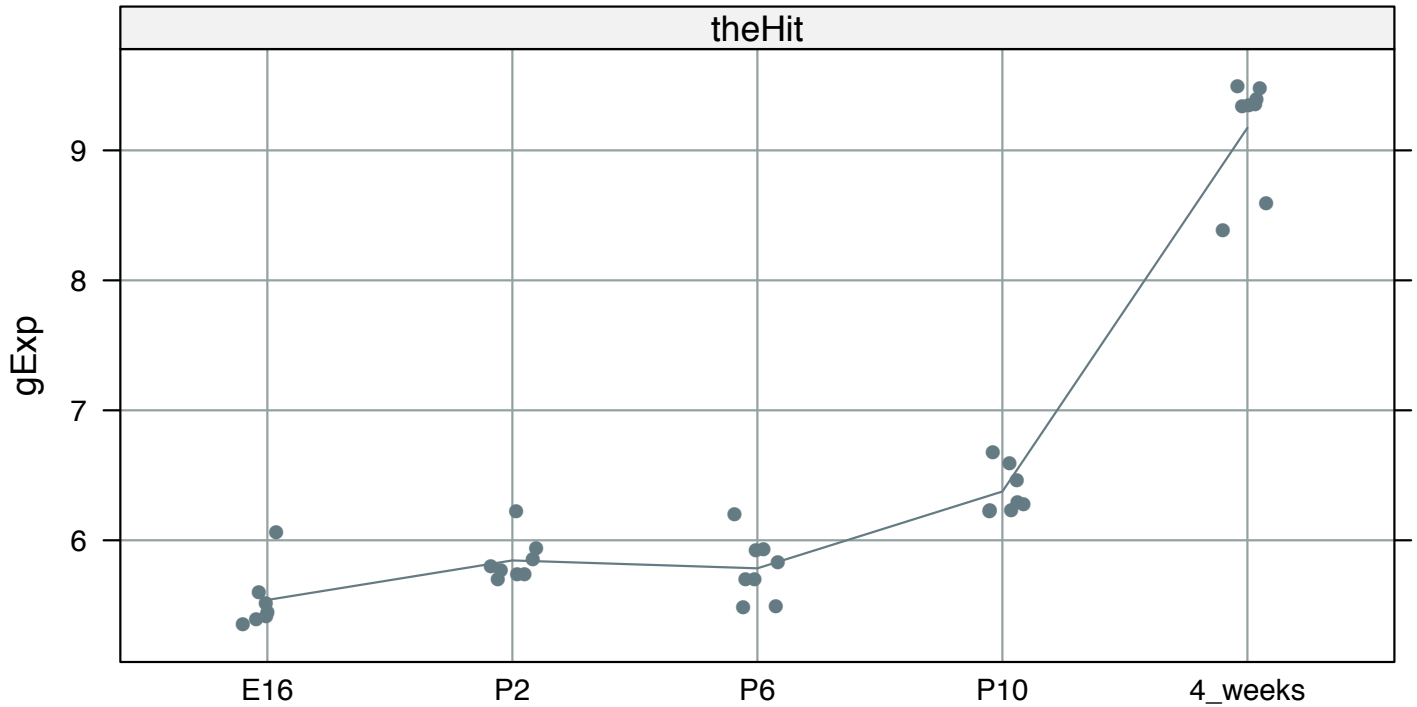
$H_0 : \tau_j = 0$

vs

$H_0 : \tau_j \neq 0$

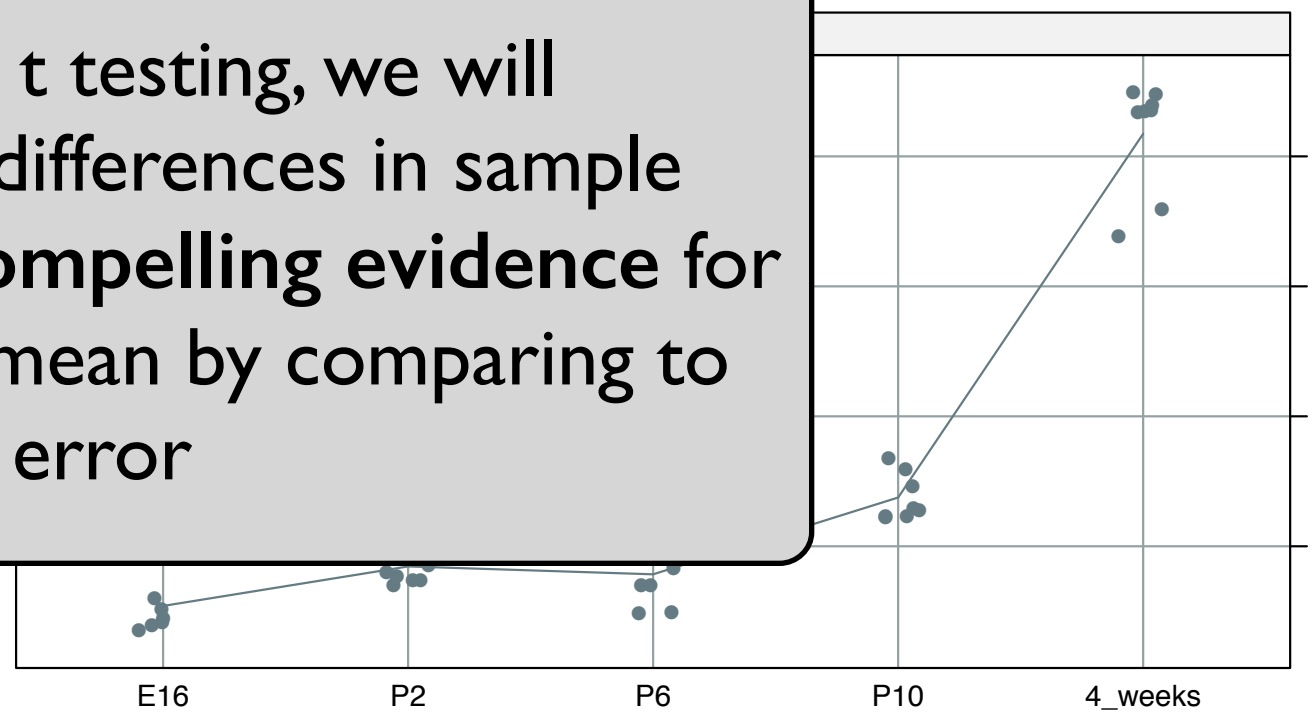
for all j at the same time

```
> summary(hitFit)
Call:
lm(formula = gExp ~ devStage, <blah, blah>)
<snip, snip>
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.5409     0.1021  54.249  < 2e-16 ***
devStageP2        0.3040     0.1399   2.174   0.0368 *
devStageP6        0.2434     0.1399   1.740   0.0909 .
devStageP10       0.8343     0.1399   5.965 9.56e-07 ***
devStage4_weeks   3.6325     0.1399  25.973  < 2e-16 ***
---
<snip, snip>
F-statistic: 243.4 on 4 and 34 DF,  p-value: < 2.2e-16
```




```
> summary(hitFit)
Call:
lm(formula = gExp ~ devStage, <blah, blah>)
<snip, snip>
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.5409     0.1021  54.249  < 2e-16 ***
devStageP2        0.3040     0.1399   2.174   0.0368 *
devStageP6        0.2434     0.1399   1.740   0.0909 .
devStageP10       0.8343     0.1399   5.965 9.56e-07 ***
devStage4_weeks   3.6325     0.1399  25.973  < 2e-16 ***
---
<snip, snip>
F-statistic: 243.4 on 4 and 34 DF,  p-value: < 2.2e-16
```

as with two sample t testing, we will decide if observed differences in sample averages present **compelling evidence** for true differences in mean by comparing to a relevant standard error




```
> summary(hitFit)
Call:
lm(formula = gExp ~ devStage, <blah, blah>)
<snip, snip>
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.5409     0.1021  54.249  < 2e-16 ***
devStageP2        0.3040     0.1399   2.174   0.0368 *
devStageP6        0.2434     0.1399   1.740   0.0909 .
devStageP10       0.8343     0.1399   5.965 9.56e-07 ***
devStage4_weeks   3.6325     0.1399  25.973  < 2e-16 ***
---
<snip, snip>
F-statistic: 243.4 on 4 and 34 DF,  p-value: < 2.2e-16
```

what if we -- how would we -- force R to parametrize the model differently, e.g. using "cell means"?

```
> hitFitCellMeans <- lm(gExp ~ 0 + devStage, miniDat, gene == "theHit")

> summary(hitFitCellMeans)

Call:
lm(formula = gExp ~ 0 + devStage, <blah, blah>)

<snip, snip>

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
devStageE16      5.54086    0.10214   54.25   <2e-16 ***
devStageP2       5.84488    0.09554   61.18   <2e-16 ***
devStageP6       5.78425    0.09554   60.54   <2e-16 ***
devStageP10      6.37512    0.09554   66.73   <2e-16 ***
devStage4_weeks  9.17337    0.09554   96.02   <2e-16 ***
---
<snip, snip>
Residual standard error: 0.2702 on 34 degrees of freedom
F-statistic:  4804 on 5 and 34 DF,  p-value: < 2.2e-16
```

parameter estimates = estimated means for each devStage = sample averages

Yay for interpretability!

```
        theHitAvgs
E16       5.540857
P2        5.844875
P6        5.784250
P10       6.375125
4_weeks   9.173375
```

what if we -- how would we -- force R to parametrize the model differently, e.g. using "cell means"?

```
> hitFitCellMeans <- lm(gExp ~ 0 + devStage, miniDat, gene == "theHit")

> summary(hitFitCellMeans)

Call:
lm(formula = gExp ~ 0 + devStage, <blah, blah>)

<snip, snip>

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
devStageE16      5.54086    0.10214   54.25   <2e-16 ***
devStageP2       5.84488    0.09554   61.18   <2e-16 ***
devStageP6       5.78425    0.09554   60.54   <2e-16 ***
devStageP10      6.37512    0.09554   66.73   <2e-16 ***
devStage4_weeks  9.17337    0.09554   96.02   <2e-16 ***
---
<snip, snip>
Residual standard error: 0.2702 on 34 degrees of freedom
F-statistic:  4804 on 5 and 34 DF,  p-value: < 2.2e-16
```

BUT what null hypotheses do these p-values correspond to????

```
        theHitAvgs
E16       5.540857
P2        5.844875
P6        5.784250
P10       6.375125
4_weeks   9.173375
```

what if we -- how would we -- force R to parametrize the model differently, e.g. using "cell means"?

```
> hitFitCellMeans <- lm(gExp ~ 0 + devStage, miniDat, gene == "theHit")

> summary(hitFitCellMeans)

Call:
lm(formula = gExp ~ 0 + devStage, <blah, blah>)

<snip, snip>

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
devStageE16      5.54086    0.10214   54.25   <2e-16 ***
devStageP2       5.84488    0.09554   61.18   <2e-16 ***
devStageP6       5.78425    0.09554   60.54   <2e-16 ***
devStageP10      6.37512    0.09554   66.73   <2e-16 ***
devStage4_weeks  9.17337    0.09554   96.02   <2e-16 ***
---
<snip, snip>
Residual standard error: 0.2702 on 34 degrees of freedom
F-statistic:  4804 on 5 and 34 DF,  p-value: < 2.2e-16
```

These p-values are for these tests:

$$H_0 : \mu_j = 0$$

Probably not what you're really interested in! Boo.

```
        theHitAvgs
E16       5.540857
P2        5.844875
P6        5.784250
P10       6.375125
4_weeks   9.173375
```

Different parametrizations are useful for different things, but in some aspects, such as residual error, they are equivalent.

```
hitFit <- lm(gExp ~ devStage, miniDat, gene == "theHit")
```

```
Residual standard error: 0.2702 on 34 degrees of freedom
Multiple R-squared: 0.9663,    Adjusted R-squared: 0.9623
F-statistic: 243.4 on 4 and 34 DF,  p-value: < 2.2e-16
```

```
hitFitCellMeans <- lm(gExp ~ 0 + devStage, miniDat, gene == "theHit")
```

```
Residual standard error: 0.2702 on 34 degrees of freedom
Multiple R-squared: 0.9986,    Adjusted R-squared: 0.9984
F-statistic:  4804 on 5 and 34 DF,  p-value: < 2.2e-16
```

?? Note: The artificiality of the "group means" model is highlighted here, in that overall significance arises from comparison to no model at all, i.e. E(Yj) = 0.