R for Microarray and RNAseq Processing

Pam Wu
Coding Club 2015

Installing R

- https://www.r-project.org/
- Click CRAN (on left), pick a mirror (USA), pick an operating system (if Windows, click 'base')
- https://www.rstudio.com/
- RStudio is a user-friendly programming interface, so it's optional but recommended
- For this class, I'll be using RStudio

What's the deal with R?

- R is a language for statistics with many built in functions and a wide array of libraries
- Plotting and data processing with very well-known data types (microarrays, RNAseq) is a lot easier in R than in any other language.
- But it's limited in scope, its for loops are really slow, it's not so easy when your data formats are new or unusual, and it handles memory badly
- It's best to think of it as a toolkit of pre-fab scripts for commonly used pipelines

Try some stuff out:

- 10^2 + 25
- a <- 4
- a <- a + 4
- mystr <- "hello world"
- mystr[1]
- mylist <- c(3, 4, 5)
- data <- data.frame(x= c(11,12,14), y=c(19,20,21), z=c(10,9,7))
- data[3, 2]
- data[, 2]

- **Practice:**
- Make a variable called num and set it to 7
- Get the letter r from mystr
- Get the number 19 from data
- Make a vector with the numbers from 7 to 10
- Get the first column of data and set it to a new variable

Installing packages

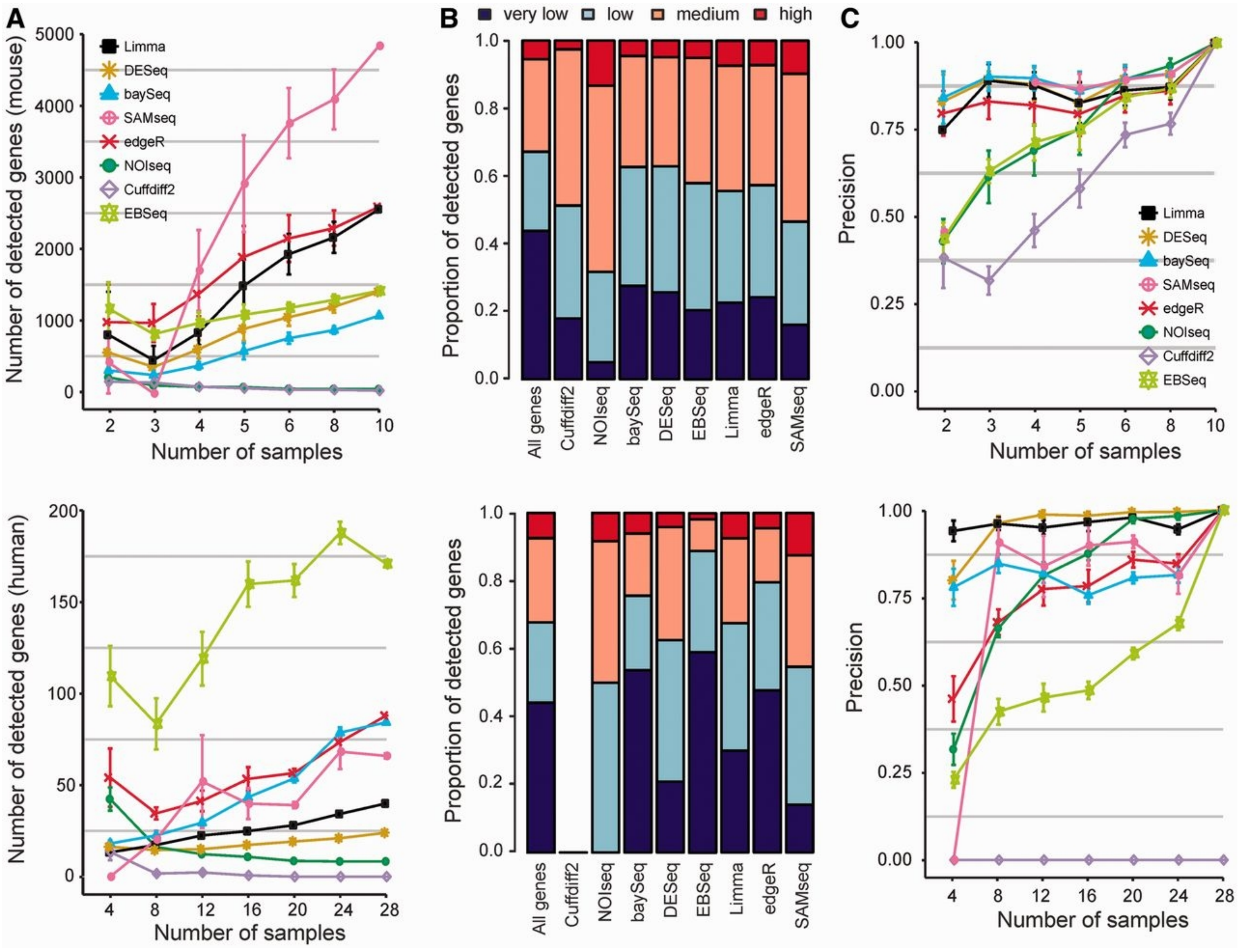
- For this next part, I suggest you do it in your Terminal
- Open a Terminal window, and write sudo R
  - install.packages("RCurl")
- We mainly use a suite of packages called Bioconductor, so in your console, type:
  - source("http://bioconductor.org/biocLite.R")
  - biocLite()
- Now install Bioconductor packages
  - biocLite(c("biomaRt", "rtracklayer", "GenomicRanges", "GenomicAlignments", "Rsamtools", "GenomicFeatures", "limma", "edgeR", "statmod"))
  - biocLite("RNAseqData.HNRNPC.bam.chr14")

When to use R (limma) vs. Tophat/Cuffdiff

- There are a ton of available RNAseq processing software packages available, and many are not in R
- The packages in R are generally for sensitive quantitation of differential expression on a gene level – so it's for very basic questions about which genes are going up or down
- R packages like limma are also designed to handle small sample sizes and can do time-series experiments
- For isoform-specific information (if you're looking at lncRNA) and/or detection of splice junctions or fusions, use Tophat with Cufflinks/Cuffdiff, but the differential expression detection isn't as good

When to use R (limma) vs. Tophat/Cuffdiff

Seyednasrollah et al. Brief. In Bioinfo. (2013)



Getting started

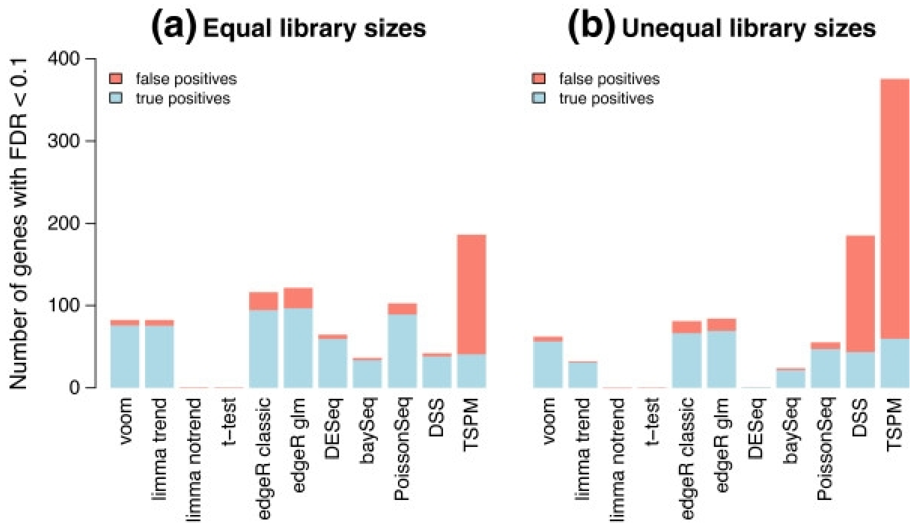
- First, make a folder called Rclass on Desktop and set your working directory to it in Rstudio
  - Session > Set Working Directory > Choose Directory
- Typically we start with FASTQ or BAM files – we're starting with BAM files, which are reads already aligned to a genome
- The data we need today is found in the RNAseqData.HNRNPC.bam.chr14 package
- Consists of RNA-seq experiments on two replicate samples from each HNRNPC knockdown (KD1 and KD2) As well as from control HeLa cells, but the data was culled to just chr14 for demo purposes

Demo time!

- (Switch to RStudio)

Why limma?

- Conventional methods of differential protein expression quantification (ANOVA, T-test) offered too little sensitivity after FDR adjustments
- Limma was developed for microarray analysis and later adapted to handle counts data for RNAseq, which is analogous to spectral counts in proteomics
- Much of the lost power from pairwise testing procedures comes from over-simplistic variance estimation and the fact that the conventional tests consider each gene comparison separately



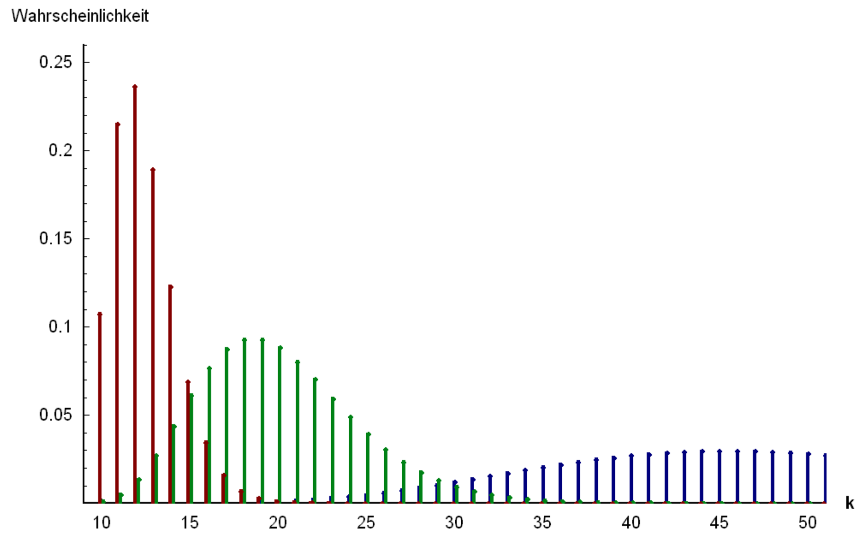
Law et al. Genome Biology, 2014

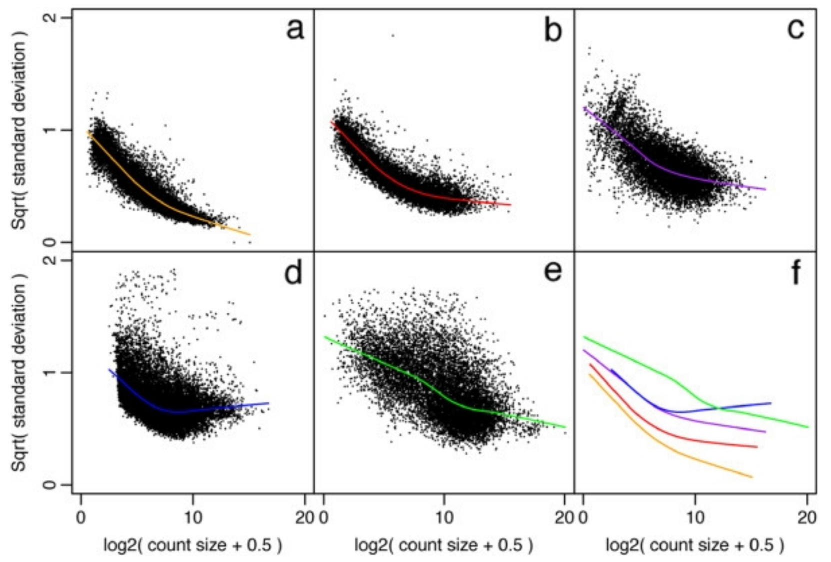
“To identify in an unbiased fashion both known and novel proteins that interact with dystrophin in skeletal and cardiac muscle, we adapted a multi-factor sample analysis commonly used in genomics (edgeR) to the analysis of protein spectral counts.”

Johnson et al. PLoS ONE, 2012

Handling counts data in DE

- One way to handle counts data is to use a discrete probability distribution, such as the negative binomial, but only if you are confident about the choice of error model
- Another way is to transform the discrete data into continuous data and then adjust for the fact that high counts have higher variance (and small log-counts have higher variance) and then use normal-based methods

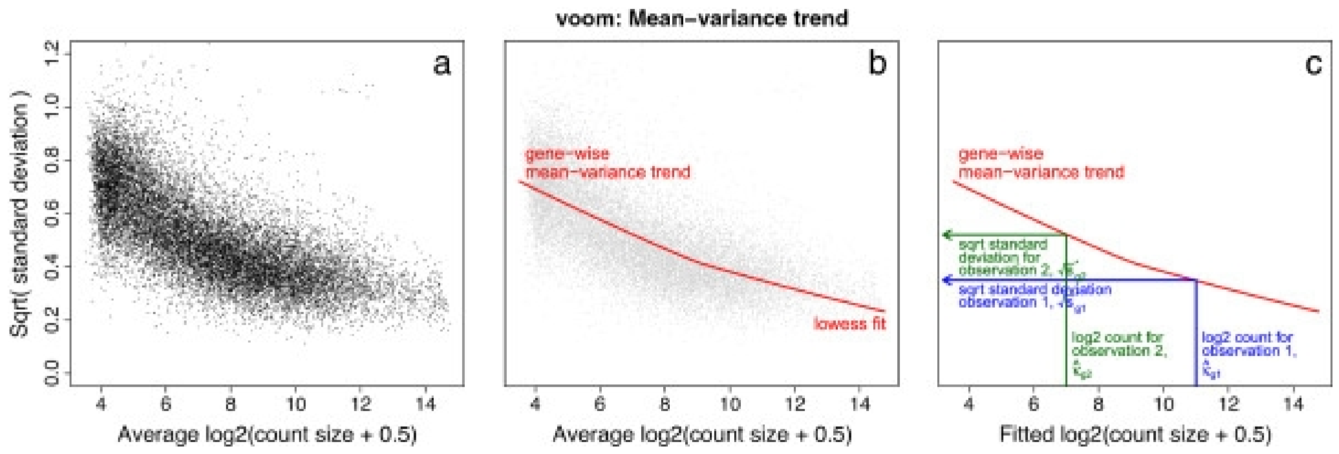




Law et al. Genome Biology, 2014

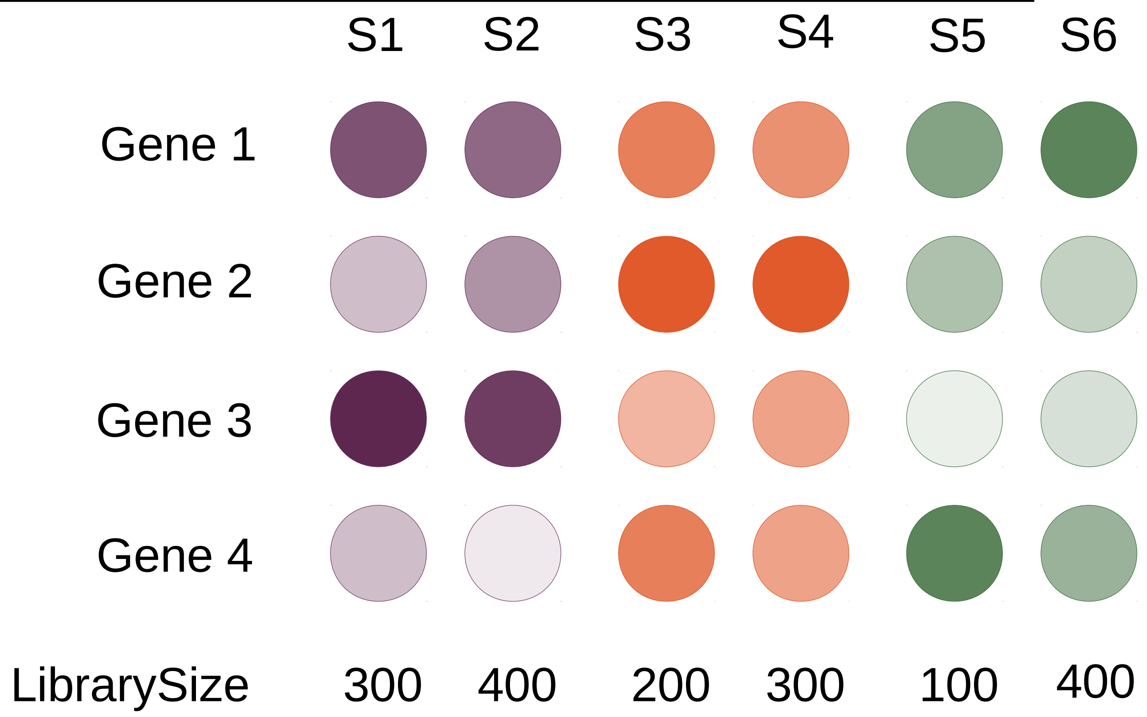
Handling counts data in DE

- Count data is normalized for library size and then log transformed to obtain continuous values
- Each sample is then weighted by its sample size
- Voom (variance modeling at the observational level) fits a curved spline to the mean-variance relationship
- Each data point gets a precision weight that is inversely related to how high the variance is at that approximate log-count level



Data is normalized for library size and log transformed

E(S, g) =

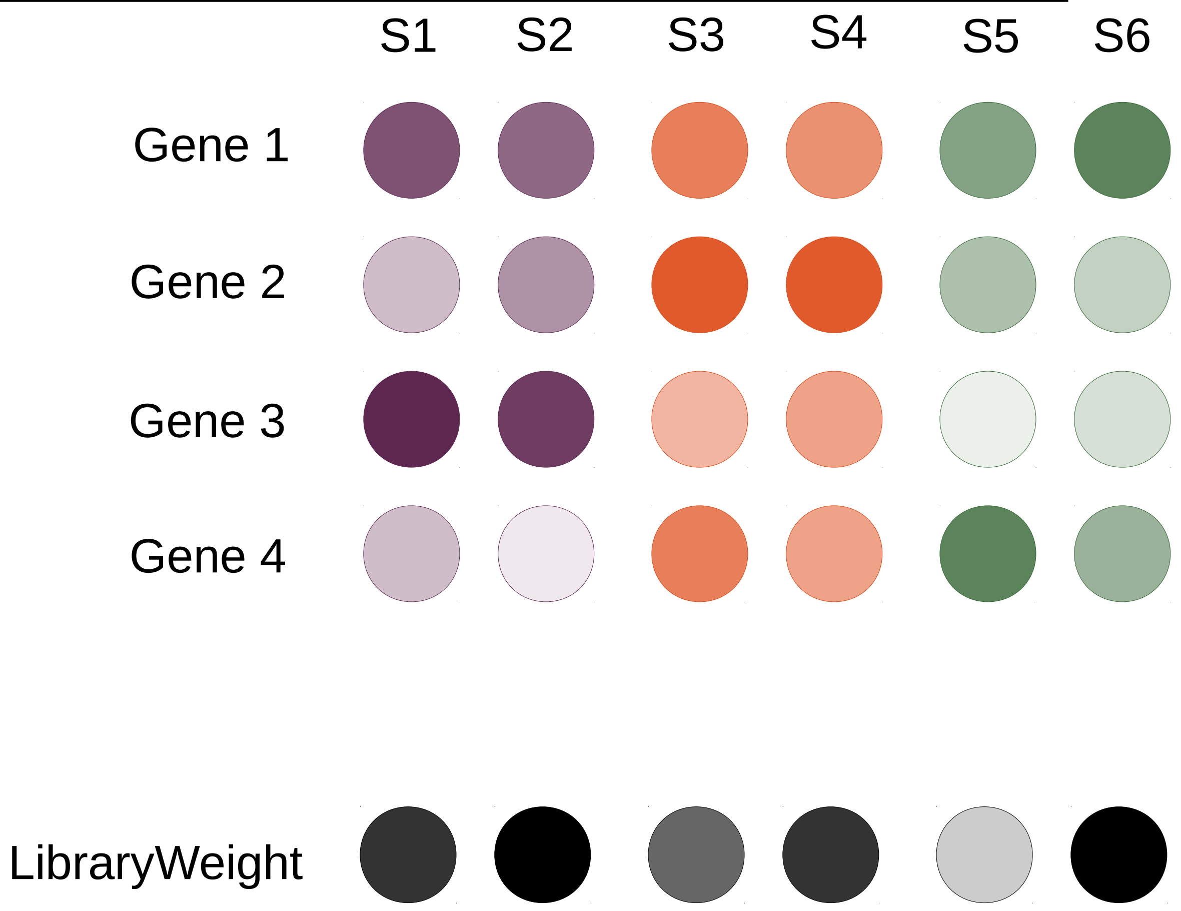


● **= log2(numCounts/librarySize)**

Library weights are calculated

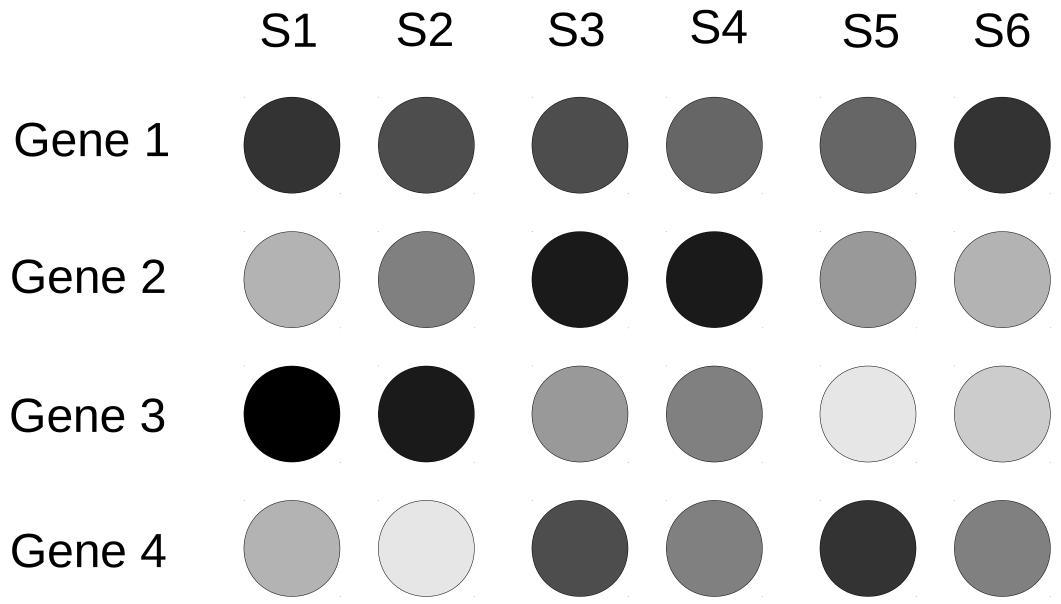
E(S, g) =

lw(S) =

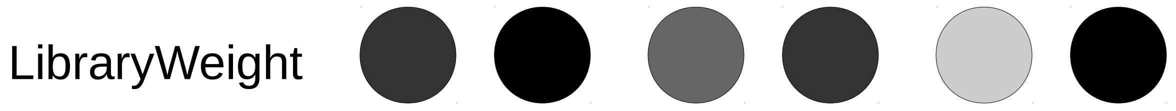


Observation weights are calculated

pw(S, g) =

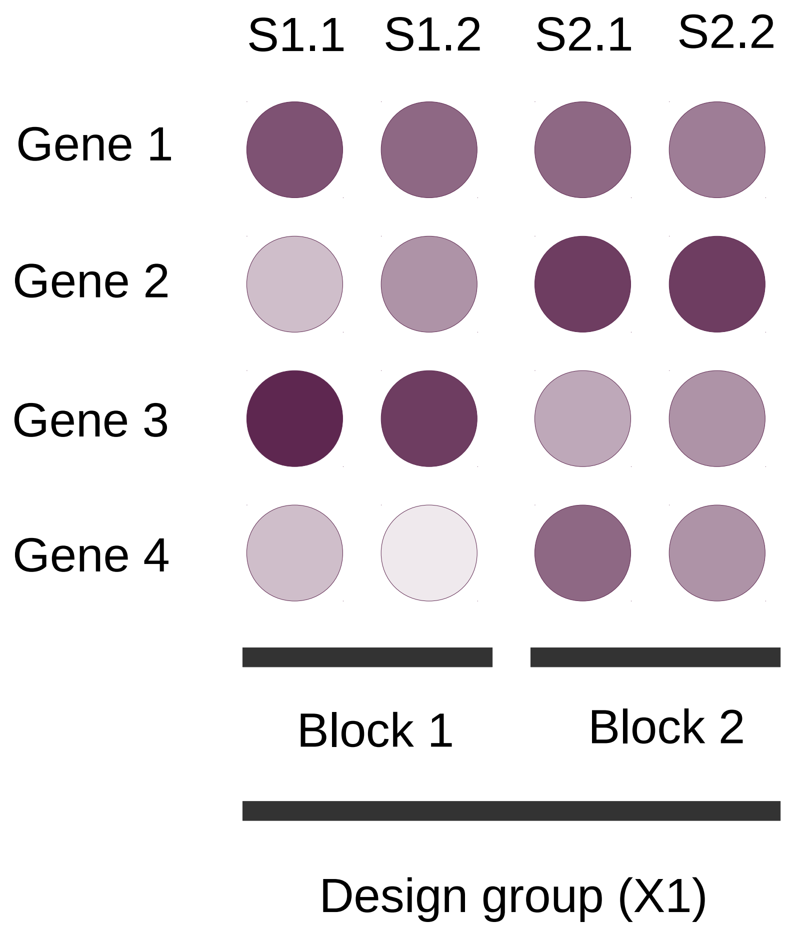


lw(S) =



Combining samples and replicates

- It's not necessary with this model to pre-merge technical replicates
- Instead, samples are blocked together and their inter-replicate correlation is calculated and fed later into the model
- With both technical and biological replicates, each set of technical replicates is represented by a block
- Each biological replicate is a design group (more complicated for microarrays)

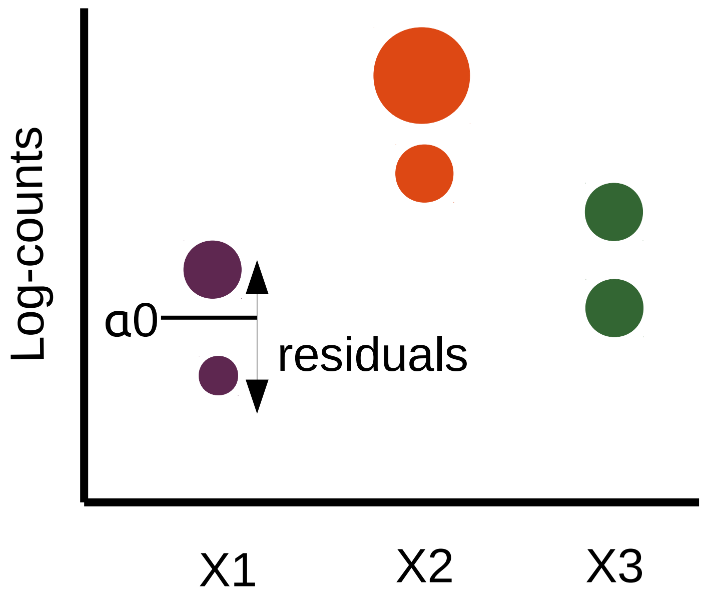


Fitting observations to a linear model

- Multiple linear regression involves fitting for each gene based on each sample's expression
- A slope and intercept, along with the residuals to the fitted linear relationship, is extracted
- The correlation between technical replicates, library size weights, and the matrix of precision weights are all taken into account in the determining the fit
- Question it asks: are the ɑ coefficients sufficiently different to predict the same gene for the comparisons of interest?

$E(g) = X * \alpha$

$var(g) = W(g) * \sigma^2$



$\alpha = [\alpha 0, \alpha 1, \alpha 2]$

Fitting observations to a linear model

- The variance of the coefficients is the product of the gene variance and the residual variance
- To make comparisons, you define a contrast matrix (C) that specifies all the comparisons you want to make (i.e. orange vs purple, green vs. purple) and you need to get the variance for the estimators of the contrasts, β
- The contrast estimators (β) are assumed to be normal, but the residuals are assumed to follow a scaled Chi square distribution
- **The point of all of this is that a t-statistic can be calculated out of the β contrast estimates and their calculated variances (and an F-statistic for any difference in contrasts, not shown)**

$$C = \begin{array}{c|cc|c} & O/P & G/P & \\ & -1 & -1 & P \\ & 1 & 0 & O \\ & 0 & 1 & G \end{array}$$

$E(g) = X * \alpha$

$var(g) = W(g) * \sigma^2$

$var(\alpha) = var(g) * s^2$

$var(\beta) = C^T * C * var(\alpha)$

$$t_{gj} = \frac{\hat{\beta}_{gj}}{s_g \sqrt{v_{gj}}}$$

Adding power with empirical Bayes

- The great advantage of RNAseq and proteomics is that thousands of genes are measured in parallel, but T-tests and ANOVA assume each gene was measured one at a time, which loses power, especially after the multiple testing correction
- Lonnstedt and Speed (2002) developed a method to re-adjust the residual variance to obtain a posterior value that can be plugged back into the t-statistic calculation
- **Thus, the residual variance gets shrunken down and the t-statistic increases**

$$\tilde{s}_g^2 = E(\sigma_g^2 \mid s_g^2) = \frac{d_0 s_0^2 + d_g s_g^2}{d_0 + d_g}.$$

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

d0 and s0 are hyperparameters that are estimated using the distributions of s^2

By using the distribution of the residual variances (across all genes) to modulate each t-statistic, power is increased through the parallel processing of all genes