Statistical Methods for High Dimensional Biology

Statistical Inference for RNA-seq - Part II

Keegan Korthauer

24 February 2021

with slide contributions from Paul Pavlidis

Learning objectives (lectures 11 and 12)

- Understand *why* and *when* between and within sample normalization are needed
- Apply common between and within sample normalization approaches to RNA-seq counts
- Understand why the *count nature* of RNA-seq data requires modification to the Differential Expression approaches applied to microarray data (e.g. limma)
- Apply models such as limma-trend, limma-voom, DESeq2 and edgeR for inference of Differential Expression

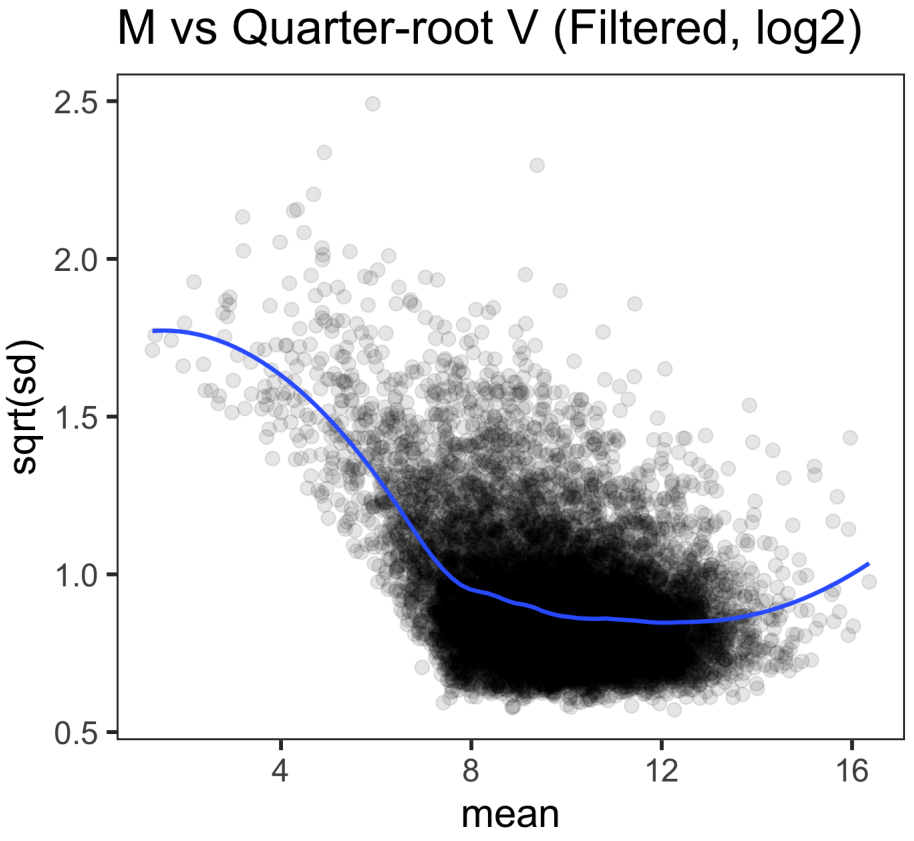
Reminder: Additional resources

- Companion document for this lecture (Rmd, md and html) that goes through much of what we discuss in greater detail can be found here
    - that link points to the markdown version - download and open the html version in your browser to make use of the table of contents feature
- Source Rmd for these slides (and all slides generated with Rmd) can be found here
- For all of the specific methods we discuss, refer to the Bioconductor pages (vignettes, reference manuals) for the most current and thorough details on implementation

How do we handle these M-V relationships in our analysis?

Options we discussed last time:

- Use a non-parametric test
- Make adjustments and model as usual
- Use a model specific for count data



One option: Voom

Mean-variance modelling at the observational level

- Falls under the category *"Make adjustments and model as usual"*
- Specifically, adjustment to regular `lm` to take into account the M-V relationship
- **Key idea**: modeling the mean-variance relationship is more important than getting the probability distribution exactly right (i.e. don't bother with other distributions like Poisson, Binomial, etc)
- Input:
  - **raw counts** (required to estimate M-V relationship), but modeling is done on log-transformed CPM values (log2(CPM + 0.5), to be precise)
  - design matrix
- Implemented in `limma` package: `voom` function

Voom steps

1. Fit linear model to $log2(CPM_{ij} + 0.5)$ values (samples i, genes j)
2. Extract the fitted quarter-root error variances $\sqrt{\hat{sd}(\epsilon_{ij})}$
3. Fit a smoothed line $\hat{f}$ to the trend between mean log count and $\sqrt{\hat{sd}(\epsilon_{ij})}$ using lowess (locally weighted regression)
4. Use the fitted lowess curve to estimate *observational weights*: $w_i = \frac{1}{\hat{f}\,(\hat{c}_{ij})^4}$ where $\hat{c}_i$ are the log2 fitted *counts* (estimated from model in step 1)
5. Fit linear model to $log2(CPM_{ij} + 0.5)$ values **using observational weights** w_{ij}
6. Compute moderated t-statistics as before (using `eBayes`)

Voom illustration

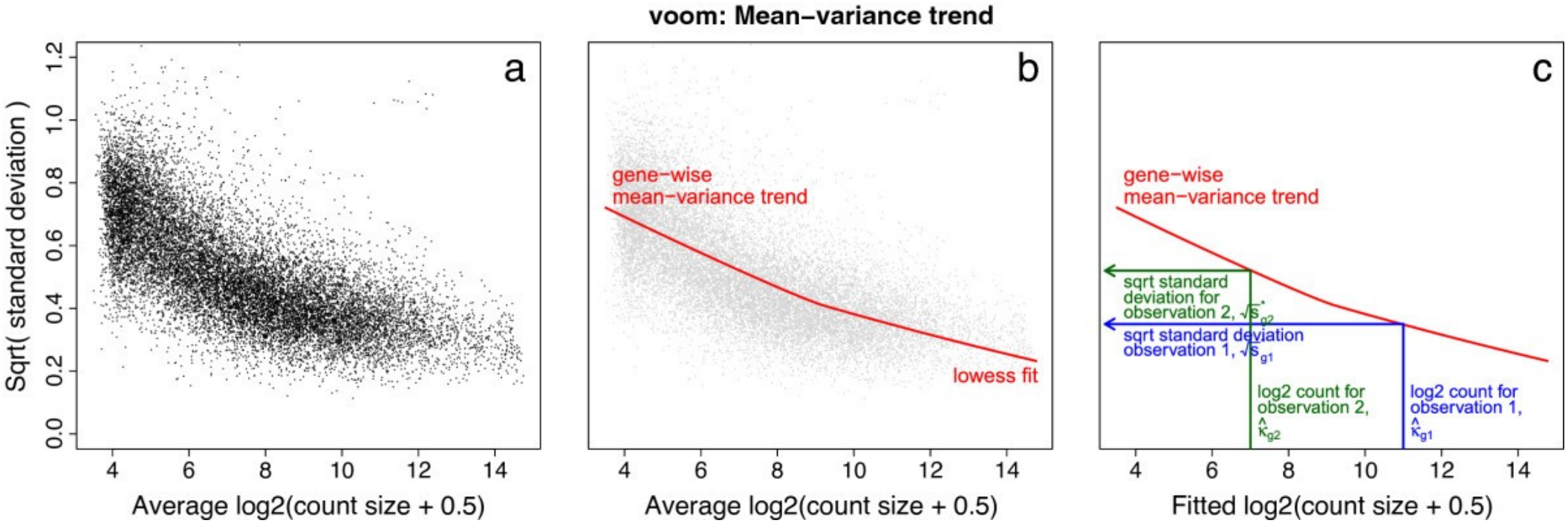


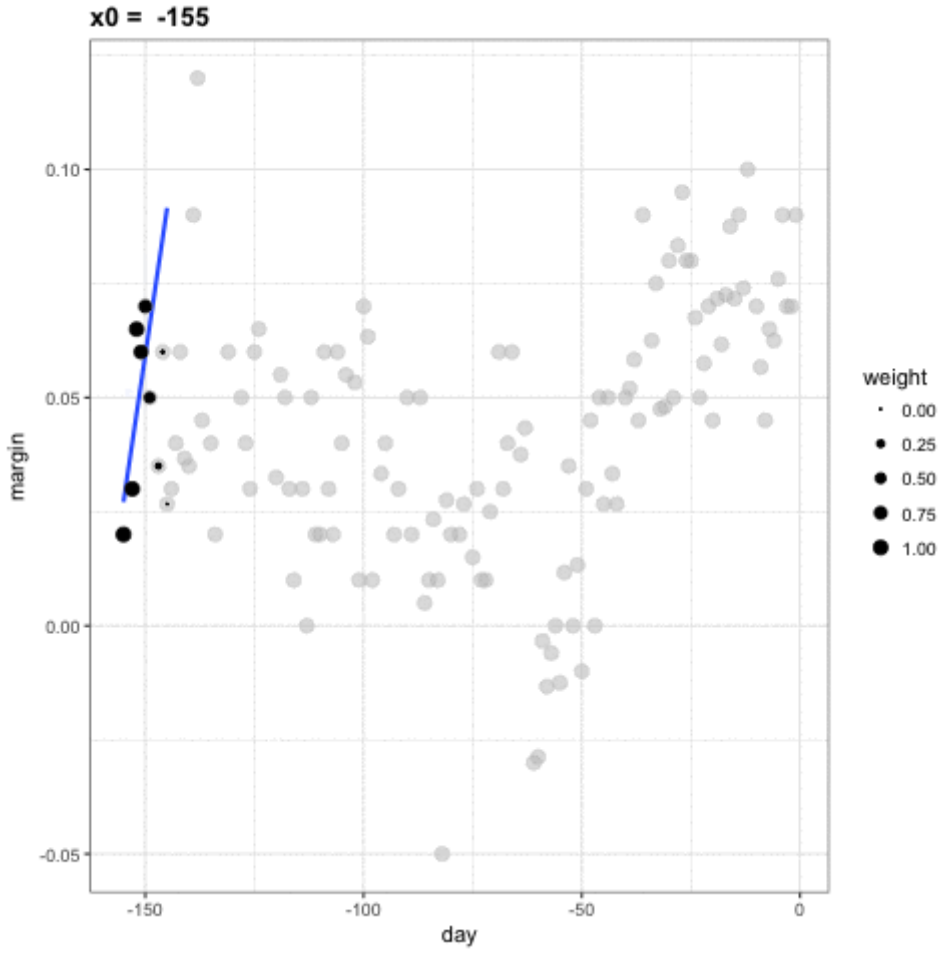
Figure 2, Law et. al, 2014

$$w_i = \frac{1}{\hat{f}\,(\hat{c}_{ij})^4} = \frac{1}{(\sqrt{s_{ij}})^4} = \frac{1}{s_{ij}^2}$$

lowess

- locally weighted regression fits a smooth curve to approximate the relationship between independent and dependent variables
- Each smoothed value is given by a weighted linear least squares regression over the **span** (a neighborhood of the independent variable)
- Smoothing span is adjustable
- Generalization to locally weighted polynomial regression and inclusion of multiple independent variables: `loess`

Image source: "Introduction to Data Science" by Irizarry



Why quarter-root variance?

- The squared **coefficient of variation** $(CV = \frac{\sigma}{\mu})$ for RNA-seq counts is roughly $CV^2 = \frac{1}{\lambda} + \phi$
  - λ is expected size of count
  - ϕ is a measure of biological variation (*overdispersion*)
- First term arises from technical variability associated with sequencing, and gradually decreases with increasing expected count size
- Second term remains roughly constant

CV of RNA-seq counts should be a decreasing function of count size for small to moderate counts, and asymptote to a value that depends on biological variability (Law et. al, 2014)

- Standard deviation of log2(CPM) is approximately equal to CV of the counts (by Taylor's theorem)

$$sd(log_2(CPM)) \approx \sqrt{\frac{1}{\lambda} + \phi}$$

- Square root of standard deviation used as distribution is more symmetric

Weighted least squares (WLS) regression

- OLS: $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$
- WLS: $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{W}\boldsymbol{y}$, where W is a diagonal matrix of weights
- **Intuition**: in minimizing the sum of squared errors, we put less weight on data points that are less precise:

$$\hat{\beta}_j = argmin_{\beta_j}\Big(\sum_{i=1}^{n} w_{ij}(X_i^T\beta_j - y_{ij})^2\Big)$$

- Optimal weights for this purpose: inverse variance
- Remedies heteroskedasticity
- Note: parameter estimates $\hat{\boldsymbol{\beta}}$ assume weights (variances) are known

limma-voom

- limma-voom is the application of limma to log2(CPM + 0.5) values, with inverse variance observational weights estimated from the M-V trend
- This alleviates the problem of heteroskedasticity and (hopefully) improves estimates of residual standard error
- Gene-specific variance estimates are 'shrunken' to borrow information across all genes:

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

- Note that s_g^2 estimates are affected by voom weights
  - recall that s_g^2 is the sum of squared residuals $\frac{1}{n-p}\hat{\boldsymbol{\epsilon}}_g{}^T \hat{\boldsymbol{\epsilon}}_g$
  - under WLS $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{g}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{y} - \boldsymbol{X}(\boldsymbol{X^T W X})^{-1}\boldsymbol{X^T W y}$

limma-voom, continued

- Moderated t statistics are then calculated using the shrunken gene-specific variance estimates: $\tilde{t}_{gj} = \frac{\hat{\beta}_{gj}}{\tilde{s}_g \sqrt{v_{jj}}}$
  - recall that under OLS, v_{jj} is the j^{th} diagonal element of $(X^T X)^{-1}$
  - under WLS, v_{jj} is the j^{th} diagonal element of $(X^T W X)^{-1}$
- Degrees of freedom for moderated t statistic: $n - p + d_0$
- If prior degrees of freedom (d_0, estimated from data) is large compared to $n - p$, moderated statistics have a bigger effect compared to using regular t statistics
  - i.e. in general, shrinkage matters more for small sample sizes

Differential expression analysis on Chd8 data

- Recall: We'd like to fit am **additive** model for each gene so we can test for Group (Chd8 mutant vs WT) effect, and adjust for:
  - Sex (M vs F)
  - DPC (days post conception, 5 levels)

$$Y_i = \theta + \tau_{Mut}x_{i,Mut} + \tau_F x_{i,F} + \tau_{D14.5}x_{i,D14.5} + \tau_{D17.5}x_{i,D17.5} + \tau_{D21}x_{i,D21} + \tau_{D77}x_{i,D77} + \epsilon_i$$

$$x_{i,Mut} = \begin{cases} 1 \text{ if sample } i \text{ is Mutant} \\ 0 \text{ otherwise} \end{cases}, \quad x_{i,F} = \begin{cases} 1 \text{ if sample } i \text{ is Female} \\ 0 \text{ otherwise} \end{cases}, \quad x_{i,D\#} = \begin{cases} 1 \text{ if sample } i \text{ is DPC\#} \\ 0 \text{ otherwise} \end{cases}$$

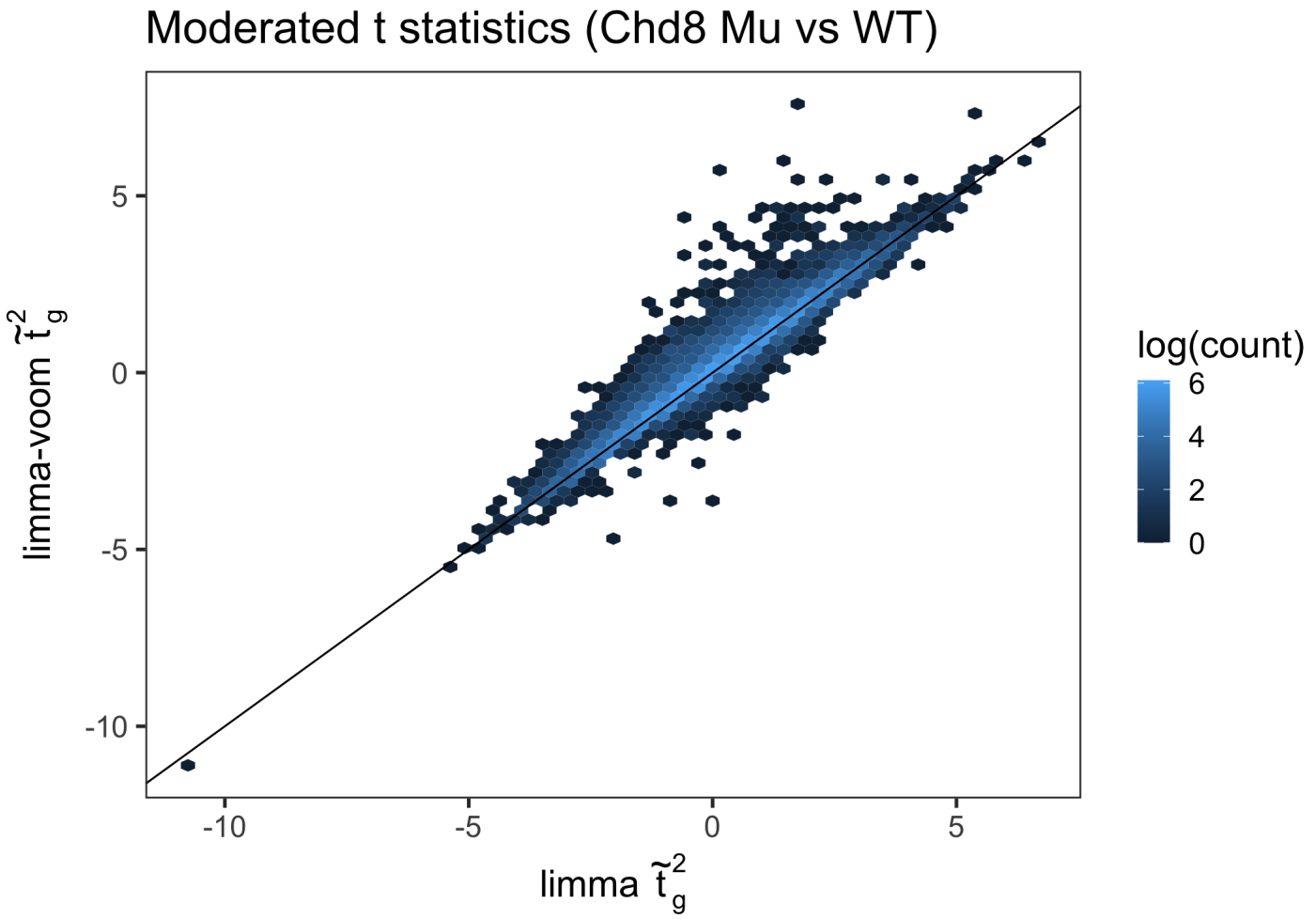
where $D\# \in \{D14.5, D17.5, D21, D77\}$

- Our model has $n - p = 44 - 7 = 37$ degrees of freedom
- We will focus on the null hypothesis of the **main effect** of Group $H_0 : \tau_{Mut} = 0$

limma-voom in action

```
vw <- voom(assays(sumexp)$counts,
           design = model.matrix(~ Sex + Group + DPC, data = colData(sumexp)),
           plot = TRUE, span = 0.5)
```

limma-voom vs limma



Another option: limma-trend

Limma-trend uses the M-V relationship at the gene level, whereas voom uses observational level trends (Law et. al, 2014)

- Gene-wise variances are shrunken toward a global M-V trend, instead of toward a constant pooled variance:

$$\tilde{s}_g^2 = \frac{d_0 s_{0g}^2 + d s_g^2}{d_0 + d}$$

- Notice the g subscript on s_{0g}^2! The prior variance is different for each gene (unlike in regular limma)
- Based on the M-V trend, s_{0g}^2 is (typically) higher for lowly expressed genes

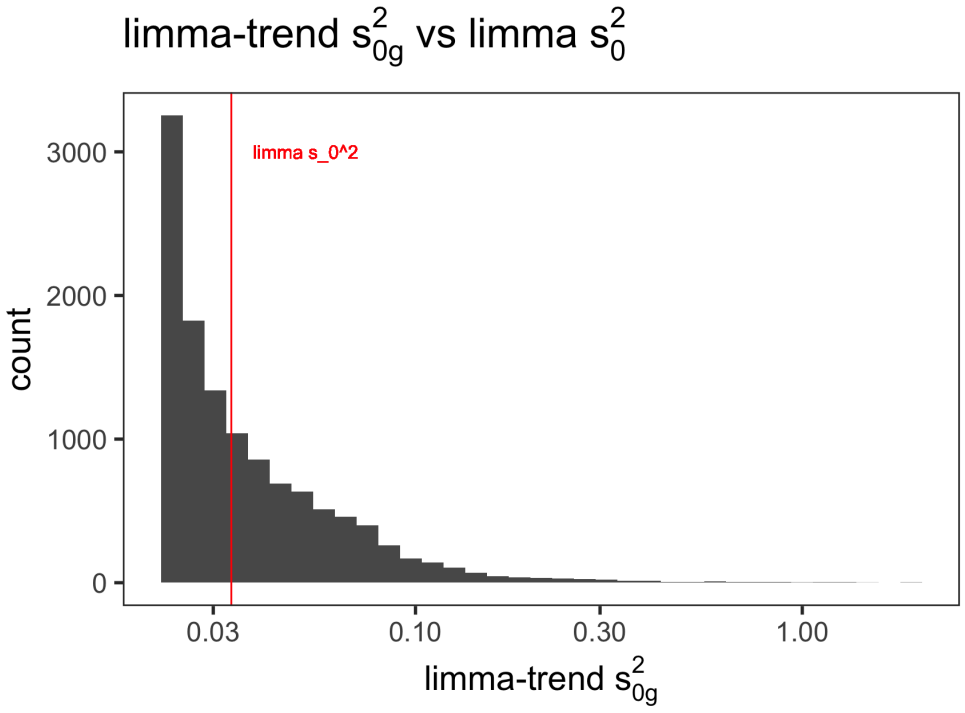
limma-trend in action

```
ltfit <- lmFit(cpm(assays(sumexp)$counts, log
               design =  model.matrix(~ Sex +
                                      data =
ltfit <- eBayes(ltfit, trend = TRUE)

# limma s^2_0
str(lfit$s2.prior)
```

```
##  num 0.0334
```

```
# limma-trend s^2_{0g}
str(ltfit$s2.prior)
```

```
##  Named num [1:12021] 0.0287 0.051 0.0274 0.023 0.
##  - attr(*, "names")= chr [1:12021] "0610007P14Rik
```

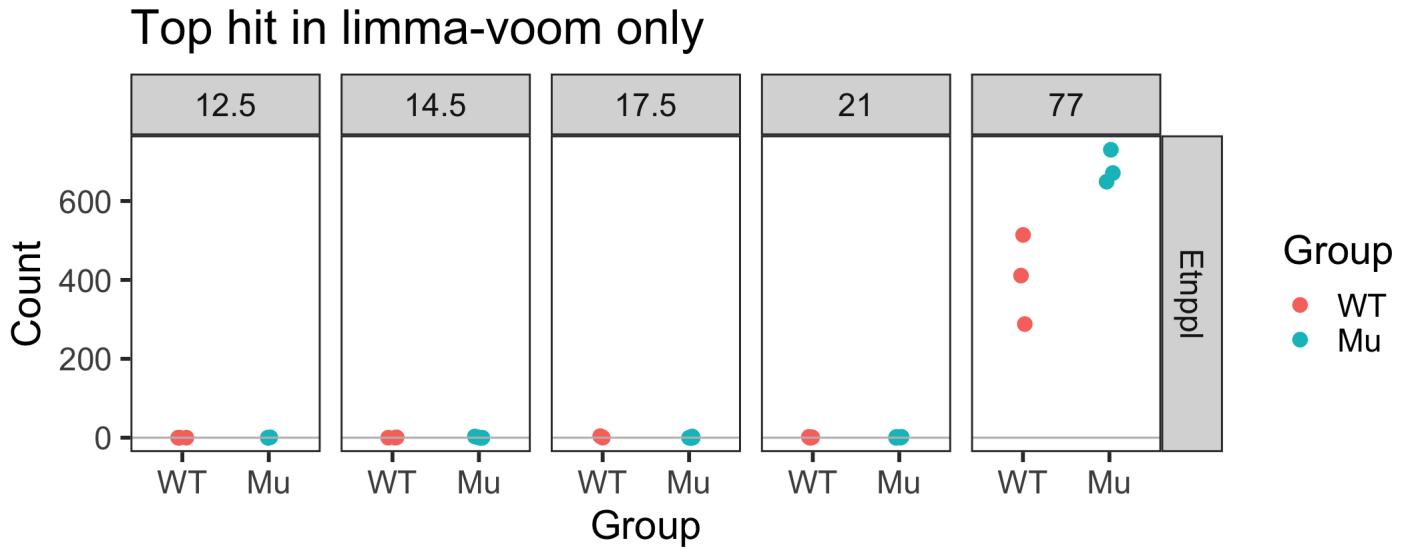


Nuances for limma-trend and limma-voom

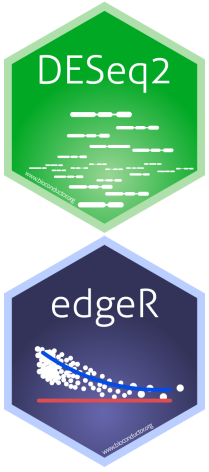
- If M-V relationship is flat, limma-voom and limma-trend have practically no effect
  - for limma-voom, weights will be all equal
  - for limma-trend, s^2_{0g} will be constant across genes
- Even if it's not flat, the impact is seen most prominently in lowly expressed genes

limma-voom 'false positives'?

- This is one of the top genes for differential expression by Group by voom (but not other methods)
- Why does this happen?
  - For voom the weighting means that very low expression values are going to have little effect on the model fit
  - Inspecting the weights for this gene they are about 30-40x higher for DPC 77 observations
- Whether this is a false positive is a matter of opinion, but lesson is: *always look at the data*



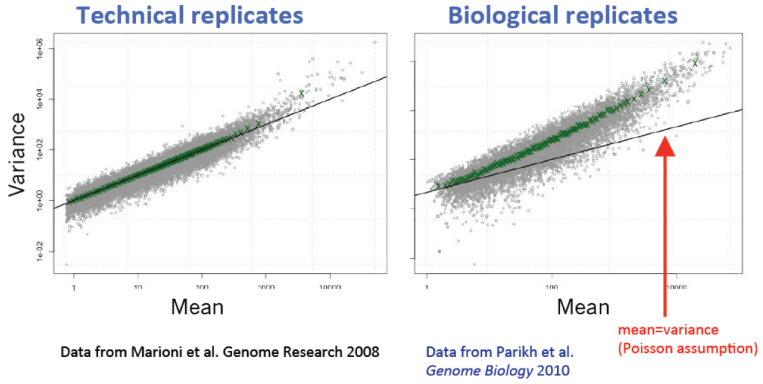
Another option: directly model counts



- Methods: `edgeR`, `DESeq2`
- Both assume counts have underlying *Negative Binomial distribution* and fit **generalized linear models**
    - **Generalized linear models** (GLM) are a generalization of OLS that allows for response variables that have error distribution models other than a normal distribution
- Still fit models gene-by-gene as we've discussed so far
- Many similarities with limma: empirical Bayes-based moderation of parameters and addressing the M-V trend

Why Negative Binomial distribution?

- Negative Binomial is also known as a Poisson-Gamma mixture
  - i.e. A Poisson with a rate parameter that is Gamma-distributed (instead of fixed)
  - The Gamma distribution on means captures the biological variance (overdispersion) that can't be accommodated by Poisson alone
- "Overdispersed Poisson" (variance > mean)
- Key problem: estimating dispersion from small datasets is tricky



Negative Binomial GLM

$$\sigma_g^2 = \mu_g(1 + \mu_g\phi_g)$$

where ϕ_g is the dispersion for gene g (if $\phi_g = 0$, get Poisson)

- We can perform inference about μ_g using GLM (e.g. using likelihood ratio tests)
- To do so, we need to treat ϕ_g as known (so first need to estimate it)

Estimation of dispersion is the main issue addressed by methods like `edgeR` and `DEseq2`

Disperson estimation

- One option is to assume ϕ is a set parametric function of the mean μ (e.g. quadratic)
- More flexible approach is to use empirical Bayes techniques:
  - Dispersion is gene-specific but moderated toward the observed trend with the mean

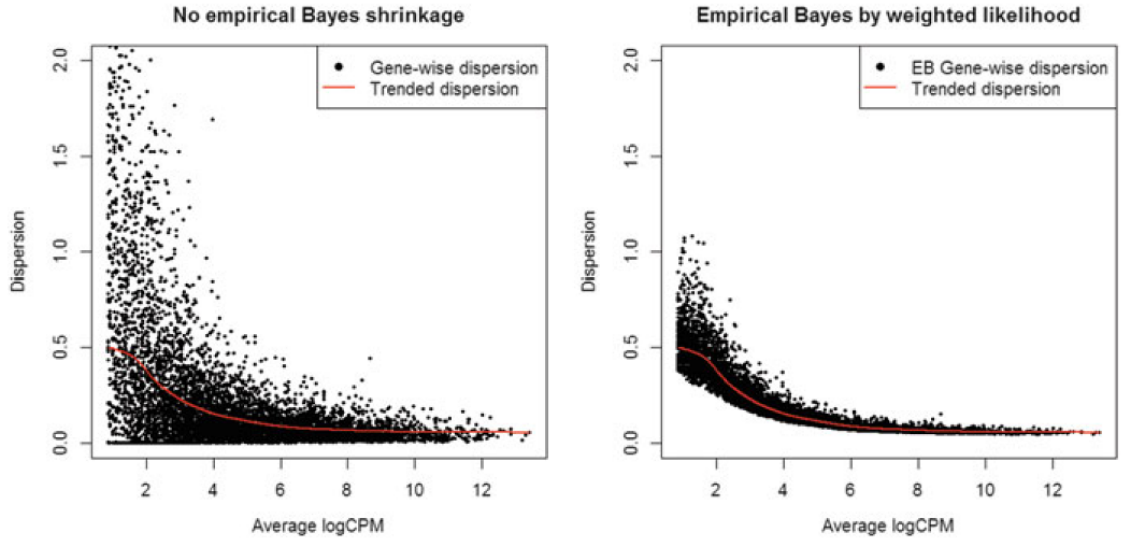


Fig. 3.2 The empirical Bayes shrinkage by weighted likelihood on simulated data. The plot on the *left* shows the dispersion estimates without empirical Bayes shrinkage. For each gene, the gene-wise dispersion estimate is obtained using the information of that gene only. The plot on the *right* shows the gene-wise dispersion estimates after empirical Bayes shrinkage. Gene-wise dispersion estimates are squeezed towards the dispersion trend which represents the use of prior information

DESeq2 vs edgeR

- These methods are very similar overall
- Major differences between the methods lie in how they filter low-count genes, estimate prior degrees of freedom, deal with outliers in dispersion estimation, and moderate dispersion of genes with high within-group variance or low counts
    - Also slight differences in types of hypothesis tests (quasi-likelihood in edgeR and Wald test in DESeq2)
- Many of these choices can be altered by changing default parameter settings in both methods (see user manuals)

DESeq2 vs edgeR

edgeR

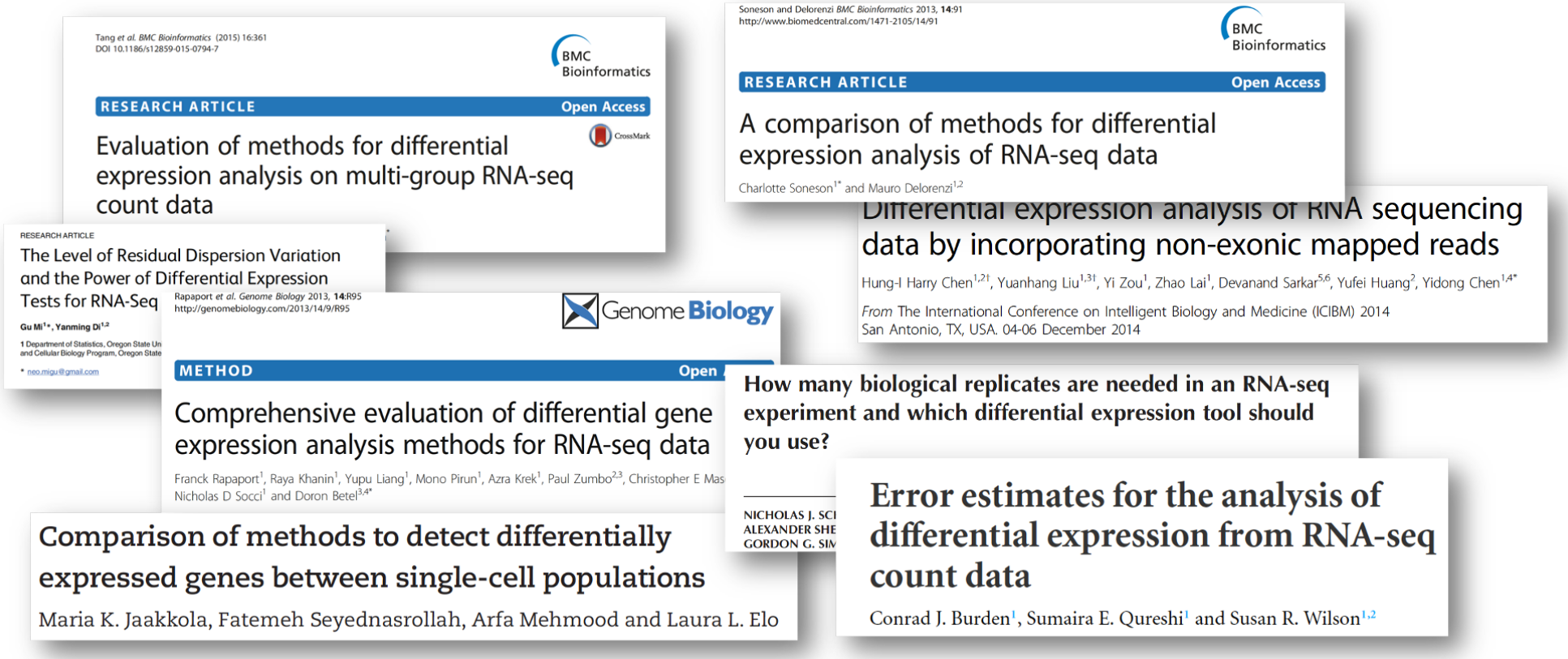
```
dge <- DGEList(assays(sumexp)$counts)
dge <- calcNormFactors(dge)
dge <- estimateDisp(dge,
                    design = model.matrix(~ Sex + Group + DPC,
                                          data = colData(sumexp)),
                    robust = TRUE)

edgeR_fit <- glmQLFit(dge,
                      design = model.matrix(~ Sex + Group + DPC,
                                            data = colData(sumexp)))
```

DESeq2

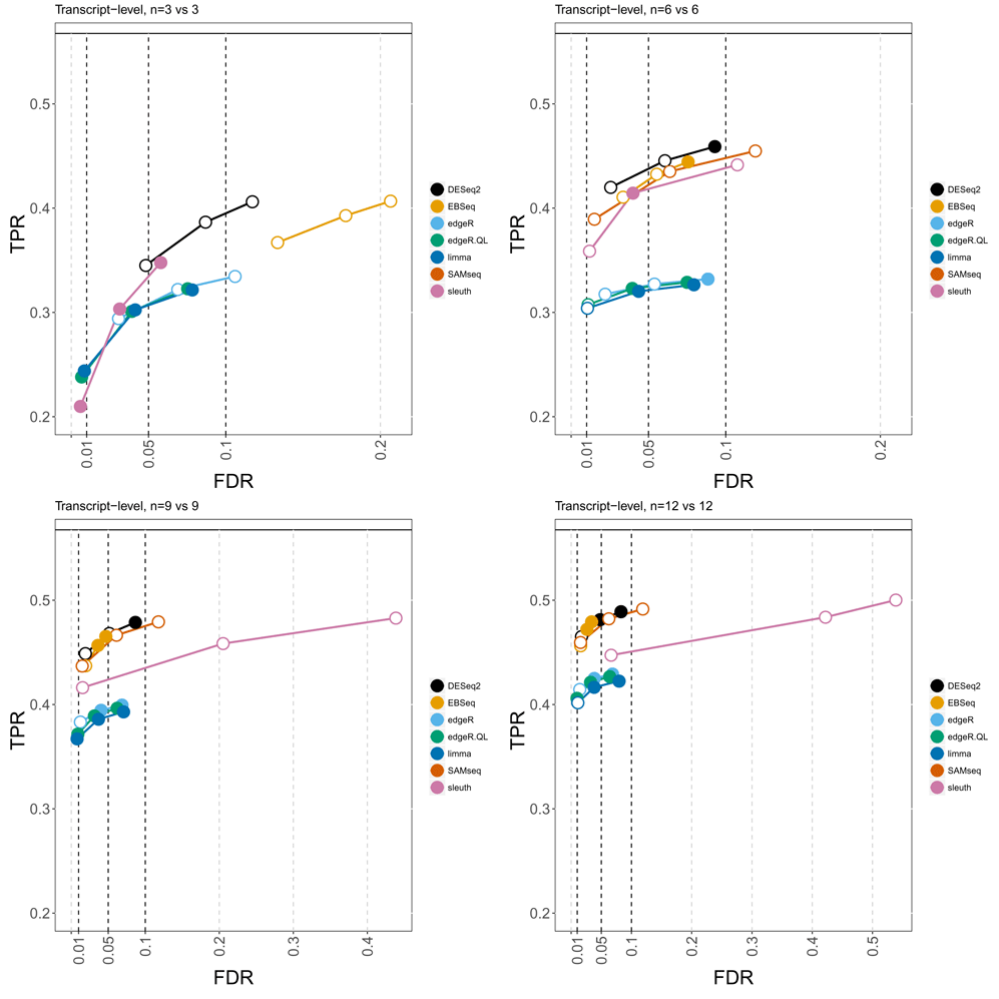
```
dds <- DESeqDataSet(sumexp,
                    design = model.matrix(~ Sex + Group + DPC,
                                          data = colData(sumexp)))
dds <- estimateSizeFactors(dds)
dds <- DESeq(dds)
```

```
## using supplied model matrix

## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

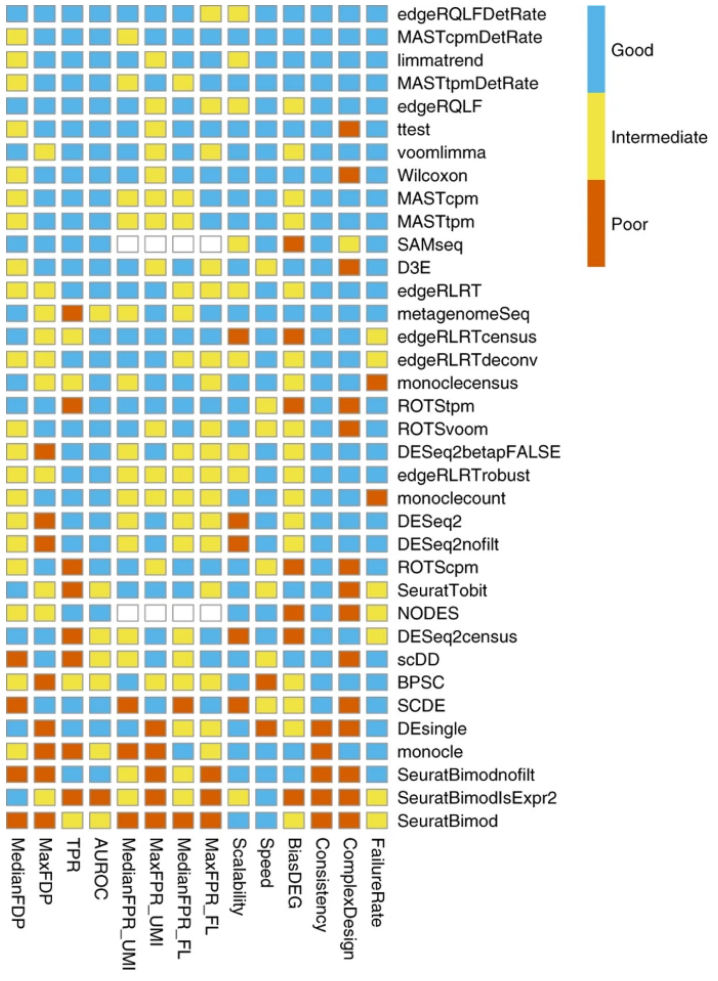
How to choose a method?



Example comparison 1 Love et al. (2018)



Example comparison 2 (for single-cell RNA-seq)



Soneson & Robinson (2018)

How to choose a method?

- No established gold standards
  - Simulations somewhat unsatisfying (depend on specific settings)
  - In real data, the truth is unknown
- The most popular and widely used methods tend to give similar results
- **edgeR** and **DESeq2** are very similar in design
  - might be expected to work better for small sample sizes or low read depth
- **limma-trend** or **limma-voom** also sound choices
  - work equally well when library sizes don't vary much
  - might not do as well when sample size or depth is very low

Comparing methods on the Chd8 dataset

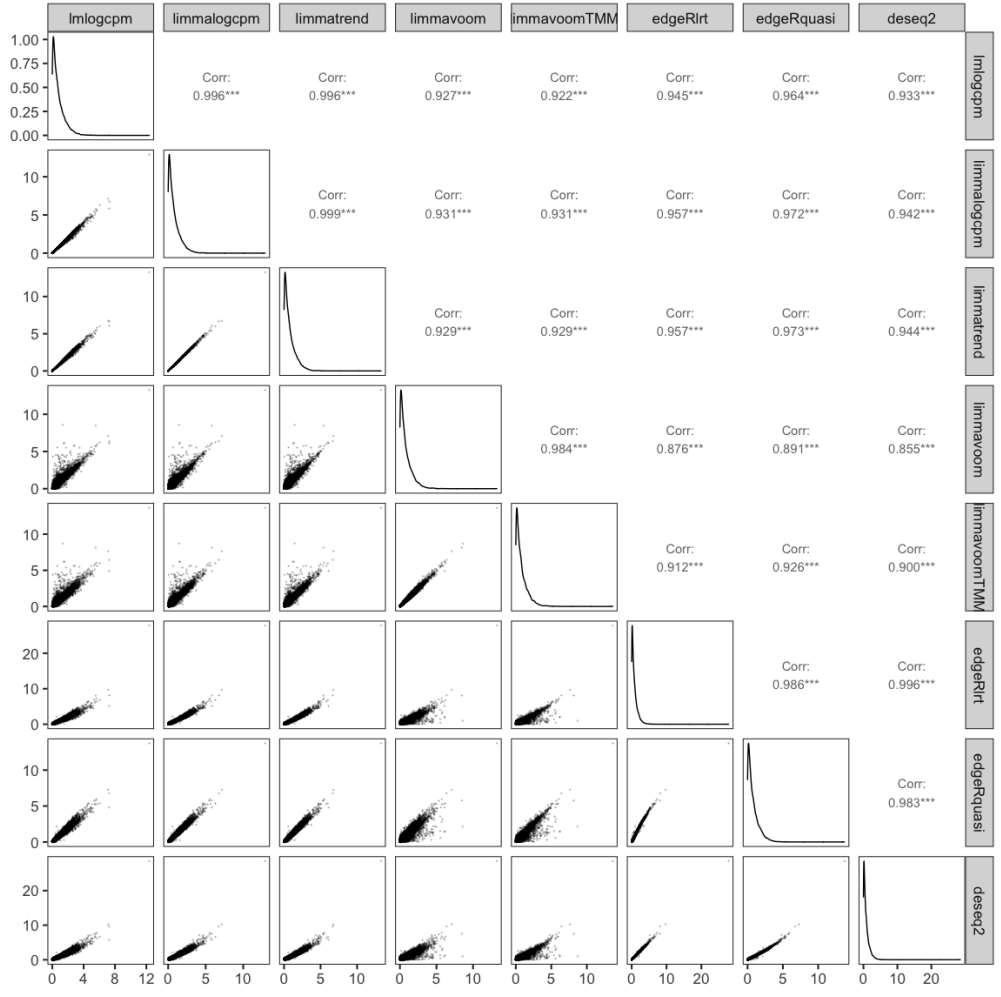
tl;dr version: there isn't a big huge difference

Possible reasons why:

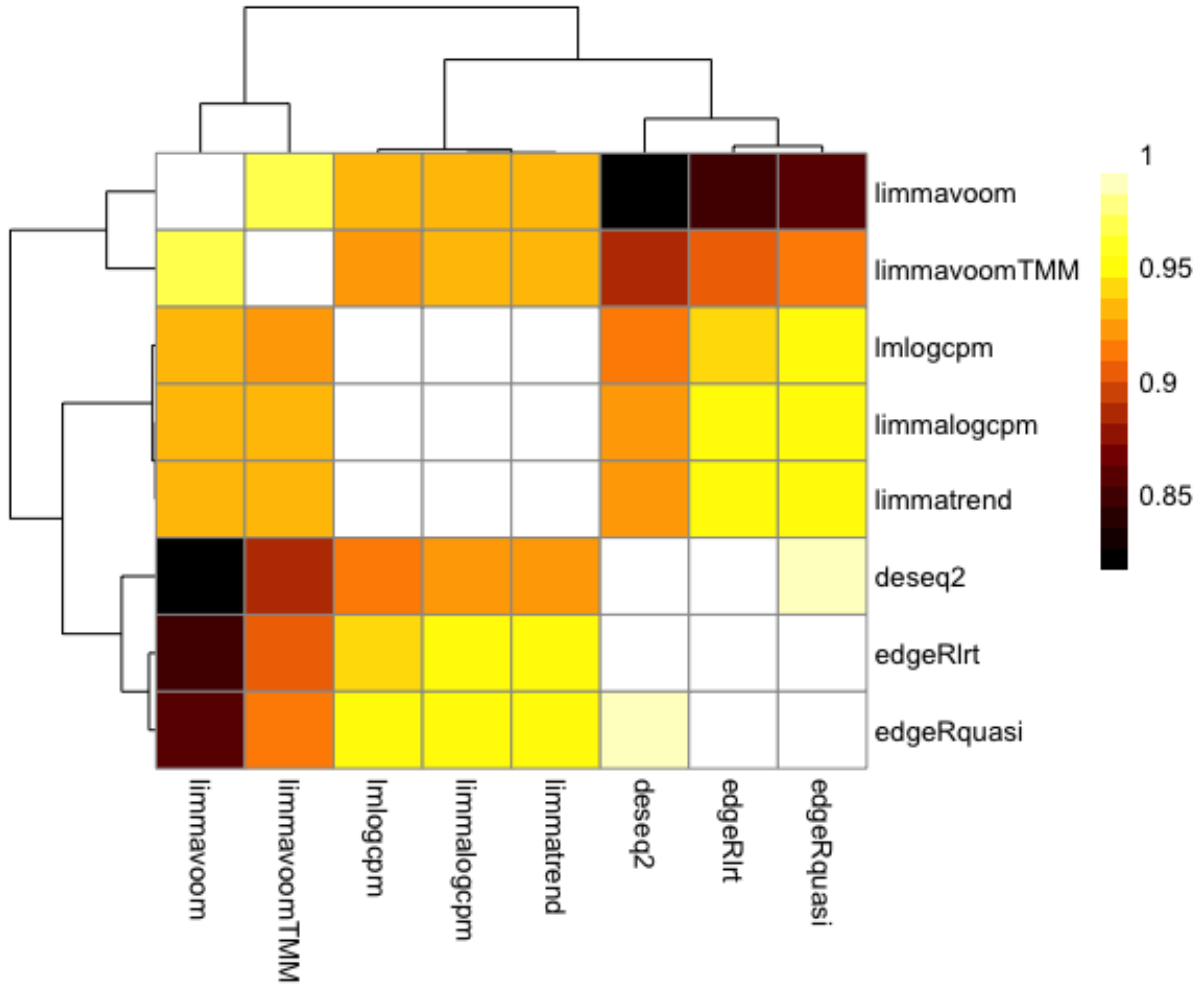
- methods have been converging in approach
- modeling count data directly with GLMs is more important for smaller samples sizes, lower read depth

Check out the comparisons in detail in the companion notes

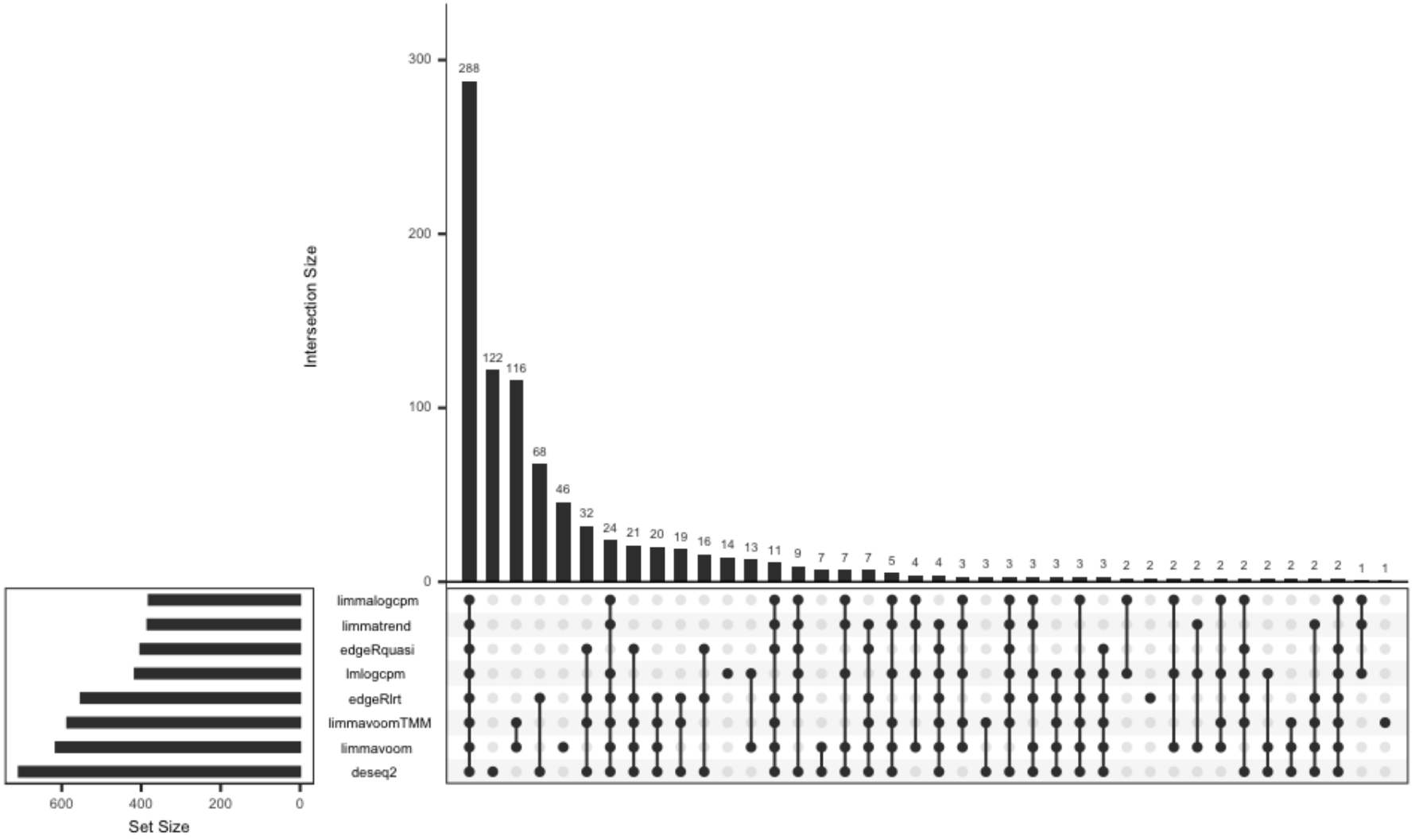
Comparisons of p-values for Chd8 mutation effect



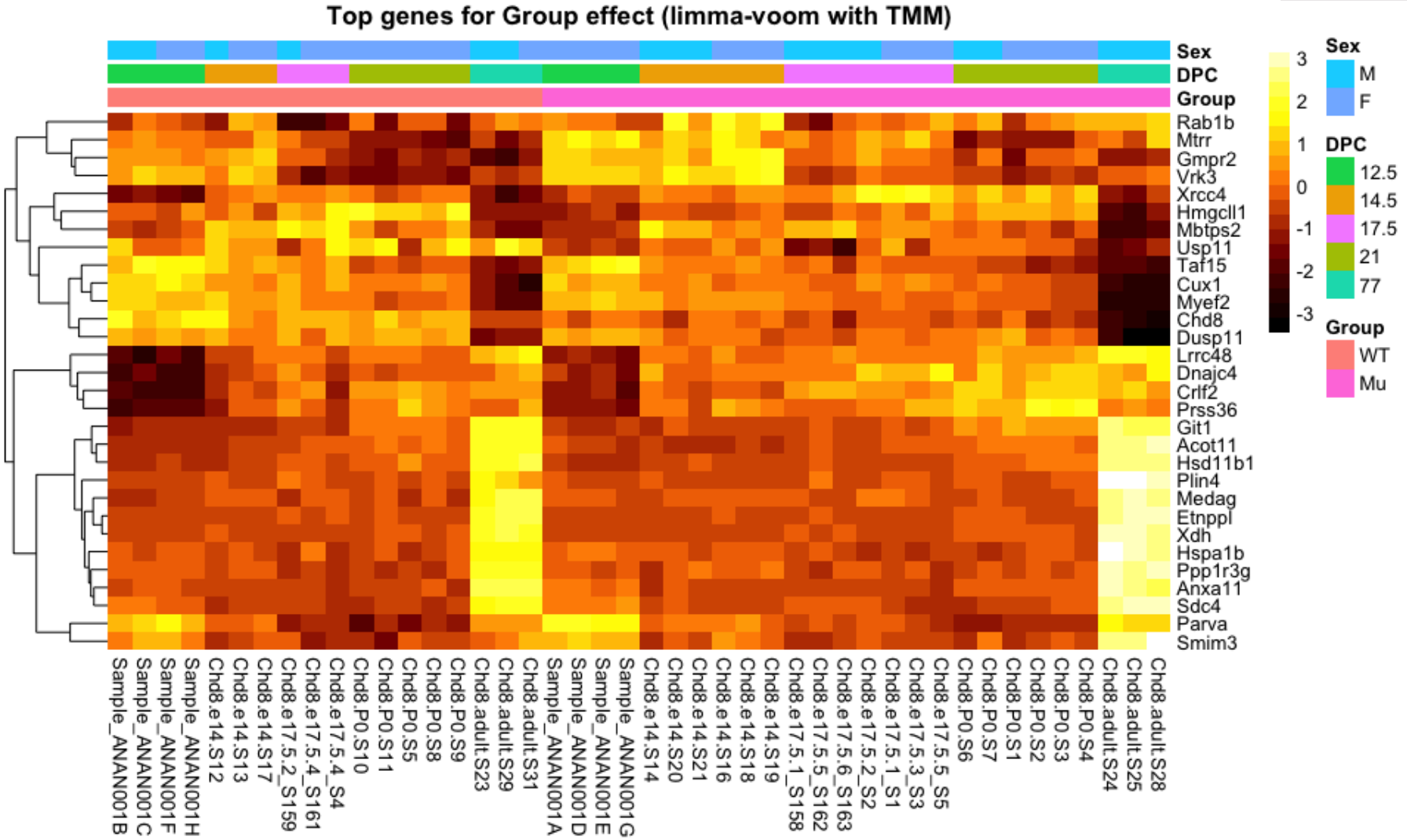
Correlation of p-value ranks for the effect of Chd8 mutation



Overlap of genes with FDR < 0.05 for the effect of Chd8 mutation



Heatmap of top 30 genes by limma-voom applied to TMM



Heatmap of top 30 genes by limma-trend, adjusted for DPC effect