Statistical Inference for RNA-seq

Yongjin Park
(Most slides from Keegan Korthauer)

13 February, 2024

Why do we want to study/improve statistical methods for RNA-seq?

- Blood, sweat, and tear of biostatistics
- Almost all high-throughput data in biology is based on sequencing
- High-dimeionsional, $p \gg n$
- Key statistical concepts: quantification, normalization, multiple hypothesis testing

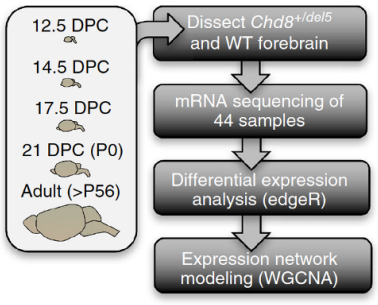
Additional resources

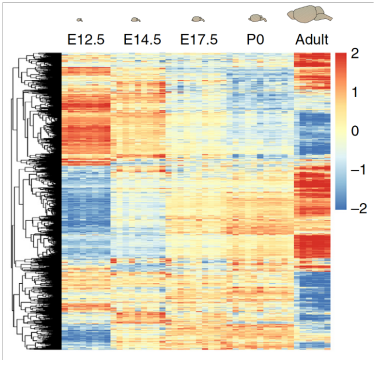
- Companion notes for this lecture with greater detail can be found here
- For all of the specific methods we discuss, refer to the Bioconductor pages (vignettes, reference manuals) for the most current and thorough details on implementation
- When you want to know specific functions in `R` (e.g., `lm`):

```
help(lm)
```

A CHD8 RNA-seq experiment (a working example)

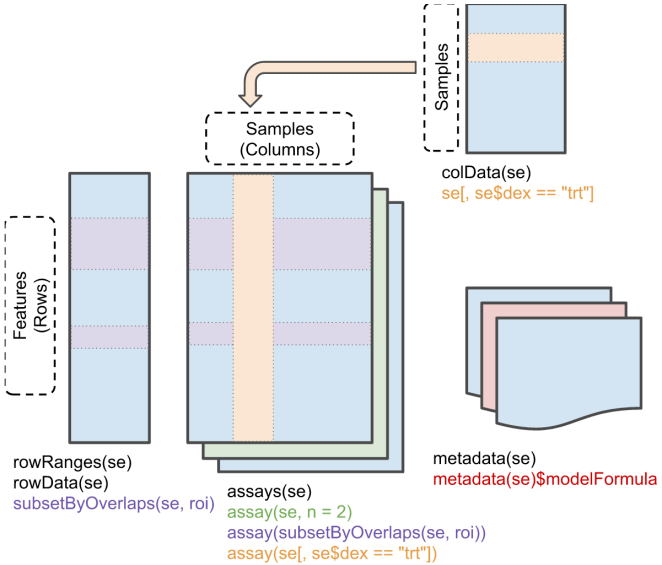
- Gompers et al. (Nature Neuroscience 2017) analyzed 26 Chd8 mutant and 18 WT mice
  - Tested for differential expression across ~12K genes accounting for sex, developmental stage and sequencing batch





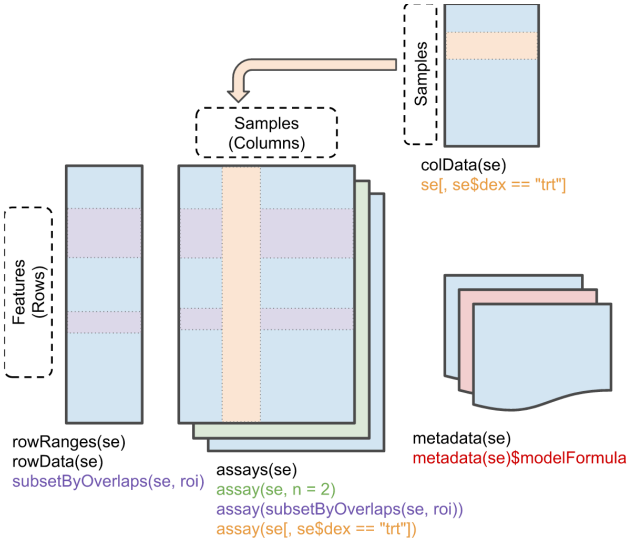
Figures from Gompers et al. (2017) paper

SummarizedExperiment as a handy container for RNA seq data



SummarizedExperiment: A special object format that is designed to contain data & metadata

When SummarizedExperiment can be useful…?



SummarizedExperiment: A special object format that is designed to contain data & metadata

- Subset rows
- Subset cols
- Storage
- Multiple assays

RNA-seq count data along with the metadata (Gompers *et al.*)

Count data: What are the rows and columns?

```
counts[1:3, 1:5]
```

```
##                Sample_ANAN001A Sample_ANAN001B Sample_ANAN001C Sample_ANAN001D
## 0610005C13Rik               20              24              20               8
## 0610007P14Rik             1714            1796            1970            1996
## 0610009B22Rik              578             866             790             858
##                Sample_ANAN001E
## 0610005C13Rik                6
## 0610007P14Rik             1864
## 0610009B22Rik              786
```

Metadata: What are the rows and columns?

```
meta[1:3, 1:5]
```

```
##                   DPC Sex Group SeqRun MappedReads
## Sample_ANAN001A 12.5   F    Mu      A    42452222
## Sample_ANAN001B 12.5   M    WT      A    54503162
## Sample_ANAN001C 12.5   M    WT      A    44978512
```

Package count and meta data into SummarizedExperiment

- What are the assays (main data)?
- What type of metadata do we have? Rows? Columns?

```
## place data into SummarizedExperiment object
sumexp <-
    SummarizedExperiment(
    assays = SimpleList(counts = counts),
    colData = DataFrame(meta)
    )
```

```
sumexp
```

```
## class: SummarizedExperiment
## dim: 20962 44
## metadata(0):
## assays(1): counts
## rownames(20962): 0610005C13Rik 0610007P14Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(44): Sample_ANAN001A Sample_ANAN001B ... Chd8.adult.S29
##   Chd8.adult.S31
## colData names(7): DPC Sex     FeatureCounts Sample
```

Package count and meta data into SummarizedExperiment

Count data

```
assays(sumexp)$counts[1:3, 1:5]
```

```
##                Sample_ANAN001A Sample_ANAN001B Sample_ANAN001C Sample_ANAN001D
## 0610005C13Rik               20              24              20               8
## 0610007P14Rik             1714            1796            1970            1996
## 0610009B22Rik              578             866             790             858
##                Sample_ANAN001E
## 0610005C13Rik                6
## 0610007P14Rik             1864
## 0610009B22Rik              786
```

Metadata

```
colData(sumexp)[1:3, 1:5]
```

```
## DataFrame with 3 rows and 5 columns
##                      DPC      Sex    Group   SeqRun MappedReads
##                 <factor> <factor> <factor> <factor>   <integer>
## Sample_ANAN001A     12.5        F       Mu        A    42452222
```

An example: Why did we create Summarizedexperiment?

```
dim(sumexp)
```

```
## [1] 20962    44
```

```
male.only <- sumexp[,sumexp$Sex == "M"]
dim(male.only)
```

```
## [1] 20962    18
```

```
dim(colData(male.only))
```

```
## [1] 18  7
```

```
dim(assays(male.only)$count)
```

```
## [1] 20962    18
```

We can treat this object seamlessly as another data matrix

```
rownames(sumexp) %>% sample(size=10)
```

```
##  [1] "Tceanc"         "Wdr34"          "Clec2h"         "Esco1"
##  [5] "Zkscan3"        "Arhgap21"       "1700081H04Rik" "Ska2"
##  [9] "Arg2"           "Rfpl4b"
```

```
colnames(sumexp) %>% head
```

```
## [1] "Sample_ANAN001A" "Sample_ANAN001B" "Sample_ANAN001C" "Sample_ANAN001D"
## [5] "Sample_ANAN001E" "Sample_ANAN001F"
```

Now we have **count** data[1]

- In the Nrl microarray experiment we worked with **continuous** microarray values
- Now we will work with the raw RNA-seq **counts** (discrete)
- These counts represent the number of reads mapping to each feature (gene or transcript) - here we have gene counts
- Seminar 6 explored how to obtain read counts from alignment (BAM or SAM) files

[1] Note that only RPKM values were provided in GEO; raw counts obtained directly from authors

What do we do now? What have we learned so far?

What do we do now? What have we learned so far?

- Lecture 3: Exploratory data analysis

What do we do now? What have we learned so far?

- Lecture 3: Exploratory data analysis
- Lecture 8: Regression analysis via ‘limma‘

What do we do now? What have we learned so far?

- Lecture 3: Exploratory data analysis
- Lecture 8: Regression analysis via ‘limma‘
- Lecture 9: Multiple hypothesis testing

We will apply all these statistical methods to RNA-seq analysis.

Today's lecture

- Exploratory Data Analysis of RNA-seq data
- Data normalization
- Statistical Inference of gene-by-gene analysis
- Multiple hypothesis testing to summarize the results

Feel free to interrupt me anytime

Today's lecture

Exploratory Data Analysis of RNA-seq data

Data normalization

Statistical Inference of gene-by-gene analysis

What do we know about the study beforehand?

- Experimental design variables:
  - **Group** (Genotype: Chd8 mutant vs WT)
  - **Sex** (M vs F, 2 level factor)[2]
  - **DPC** (days post conception, 5 level factor)
  - **Batch** (sequencing run)

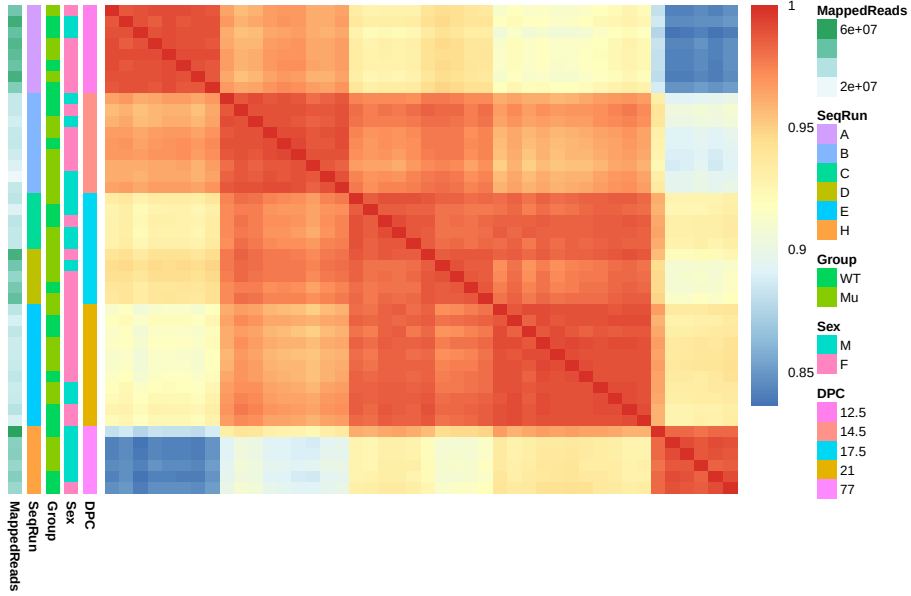
[2]Note that sex was mislabeled for some samples in the GEO entry (we are using a corrected version obtained from the authors)

Sample-sample correlation matrix to learn about variation

```
R <- cor(log1p(cpm(assays(sumexp)$counts)),
         method="spearman")
```

- Why do we need to take `log1p` $\log(x+1)$?

Sample-sample correlations reveal batch effects



What did we find in EDA?

- **Batch** and **DPC** are major sources of variation
- **Batch** and **DPC** are confounded
- One sample is a potential minor outlier

Goal: Differential expression analysis on Chd8 data

- Main variable of interest: **Group** (Genotype: Chd8 mutant vs WT)
- We'd like to fit a model for each gene so we can test for Group effect, and adjust for:
  - **Sex** (M vs F, 2 level factor)
  - **DPC** (days post conception, 5 level factor)

Goal: Differential expression analysis on Chd8 data

Using what we learned in previous lectures, we can formulate this model as:

a gene expression for a sample i

$$Y_i = \theta$$

- Baseline expression

Goal: Differential expression analysis on Chd8 data

Using what we learned in previous lectures, we can formulate this model as:

a gene expression for a sample i

$$\begin{aligned} Y_i &= \theta \\ &+ \tau_{\text{Mut}} X_{i,\text{Mut}} \end{aligned}$$

- Baseline expression
- $X_{i,\text{Mut}}$: 1 if i is mutant; otherwise, 0.

Goal: Differential expression analysis on Chd8 data

Using what we learned in previous lectures, we can formulate this model as:

a gene expression for a sample i

$$
\begin{aligned}
Y_i &= \theta \\
&+ \tau_{\text{Mut}} X_{i,\text{Mut}} \\
&+ \tau_F X_{i,\text{F}}
\end{aligned}
$$

- Baseline expression
- $X_{i,\text{Mut}}$: 1 if i is mutant; otherwise, 0.
- $X_{i,\text{F}}$: 1 if i is female; otherwise, 0.

Goal: Differential expression analysis on Chd8 data

Using what we learned in previous lectures, we can formulate this model as:

a gene expression for a sample i

$$
\begin{aligned}
Y_i &= \theta \\
&+ \tau_{\text{Mut}} X_{i,\text{Mut}} \\
&+ \tau_F X_{i,\text{F}} \\
&+ \tau_{D14.5} X_{i,D14.5} \\
&+ \tau_{D17.5} X_{i,D17.5} + \\
&+ \tau_{D21} X_{i,D21} + \\
&+ \tau_{D77} X_{i,D77}
\end{aligned}
$$

- Baseline expression
- $X_{i,\text{Mut}}$: 1 if i is mutant; otherwise, 0.
- $X_{i,\text{F}}$: 1 if i is female; otherwise, 0.
- $X_{i,D}$: 1 if and only if i's DPC $= D$
- where $D \in \{\text{D14.5, D17.5, D21, D77}\}$

Goal: Differential expression analysis on Chd8 data

Using what we learned in previous lectures, we can formulate this model as:

a gene expression for a sample i

$$
\begin{aligned}
Y_i &= \theta \\
&+ \tau_{\text{Mut}} X_{i,\text{Mut}} \\
&+ \tau_F X_{i,\text{F}} \\
&+ \tau_{D14.5} X_{i,D14.5} \\
&+ \tau_{D17.5} X_{i,D17.5} + \\
&+ \tau_{D21} X_{i,D21} + \\
&+ \tau_{D77} X_{i,D77} \\
&+ \epsilon_i
\end{aligned}
$$

- Baseline expression
- $X_{i,\text{Mut}}$: 1 if i is mutant; otherwise, 0.
- $X_{i,\text{F}}$: 1 if i is female; otherwise, 0.
- $X_{i,D}$: 1 if and only if i's DPC $= D$
- where $D \in \{\text{D14.5, D17.5, D21, D77}\}$
- ϵ: error, not explained by the previous terms

Differential expression analysis on Chd8 data

Differential expression analysis on Chd8 data

- $p = 7$ parameters to estimate in our model: $\theta, \tau_{Mut}, \tau_F, \tau_{D14.5}, \tau_{D17.5}, \tau_{D21},$ and τ_{D77}

Differential expression analysis on Chd8 data

- $p = 7$ parameters to estimate in our model: $\theta, \tau_{Mut}, \tau_F, \tau_{D14.5}, \tau_{D17.5}, \tau_{D21},$ and τ_{D77}
- $n = 44$ samples total, so our model has $n - p = 44 - 7 = 37$ degrees of freedom

Differential expression analysis on Chd8 data

- $p = 7$ parameters to estimate in our model:
  $\theta, \tau_{Mut}, \tau_F, \tau_{D14.5}, \tau_{D17.5}, \tau_{D21},$ and τ_{D77}
- $n = 44$ samples total, so our model has $n - p = 44 - 7 = 37$ degrees of freedom
- What is the null hypothesis for the test of differential expression between Chd8 Mut and WT using our model?

Differential expression analysis on Chd8 data

- $p = 7$ parameters to estimate in our model: $\theta, \tau_{Mut}, \tau_F, \tau_{D14.5}, \tau_{D17.5}, \tau_{D21},$ and τ_{D77}
- $n = 44$ samples total, so our model has $n - p = 44 - 7 = 37$ degrees of freedom
- What is the null hypothesis for the test of differential expression between Chd8 Mut and WT using our model?

Recall that since this is an additive model, the parameters represent **main effects** (not conditional)

Differential expression analysis on Chd8 data

- $p = 7$ parameters to estimate in our model: $\theta, \tau_{Mut}, \tau_F, \tau_{D14.5}, \tau_{D17.5}, \tau_{D21},$ and τ_{D77}
- $n = 44$ samples total, so our model has $n - p = 44 - 7 = 37$ degrees of freedom
- What is the null hypothesis for the test of differential expression between Chd8 Mut and WT using our model?

Recall that since this is an additive model, the parameters represent **main effects** (not conditional)

$H_0 : \tau_{\text{Mut}} = 0 \quad , H_1 : \tau_{\text{Mut}} \neq 0$

Design matrix in R

```
modm <- model.matrix(~ Sex + Group + DPC, data = colData(sumexp))
```

```
##                  (Intercept) SexF GroupMu DPC14.5 DPC17.5 DPC21 DPC77
## Sample_ANAN001A            1    1       1       0       0     0     0
## Sample_ANAN001B            1    0       0       0       0     0     0
## Sample_ANAN001C            1    0       0       0       0     0     0
## Sample_ANAN001D            1    1       1       0       0     0     0
## Sample_ANAN001E            1    1       1       0       0     0     0
## Sample_ANAN001F            1    1       0       0       0     0     0
```

Are we ready to fit the model?

Might start with the `limma` approach on the raw counts, but...

Are we ready to fit the model?

Might start with the `limma` approach on the raw counts, but...

Not so fast - we have to consider additional sources of variation!

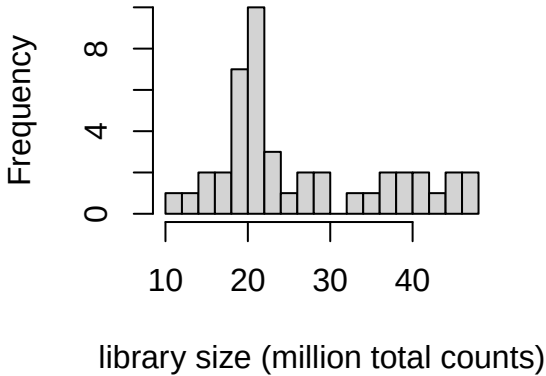
Today's lecture

Exploratory Data Analysis of RNA-seq data

Data normalization

Statistical Inference of gene-by-gene analysis

Library size (sequencing depth)



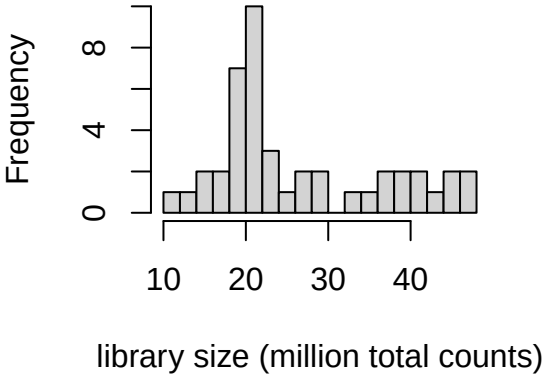
Library size

Total number of read counts per sample

Ideally this would be the same for all samples, but it isn't

Number of reads per sample depends on factors like how many samples were multiplexed and how evenly, cluster density, RNA quality, etc.

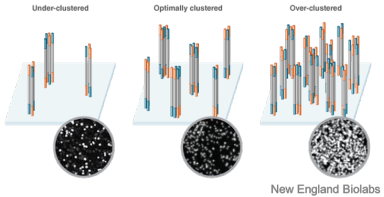
Library size (sequencing depth)



Library size

Total number of read counts per sample

Ideally this would be the same for all samples, but it isn't



Number of reads per sample depends on factors like how many samples were multiplexed and how evenly, cluster density, RNA quality, etc.

Why does library size matter?

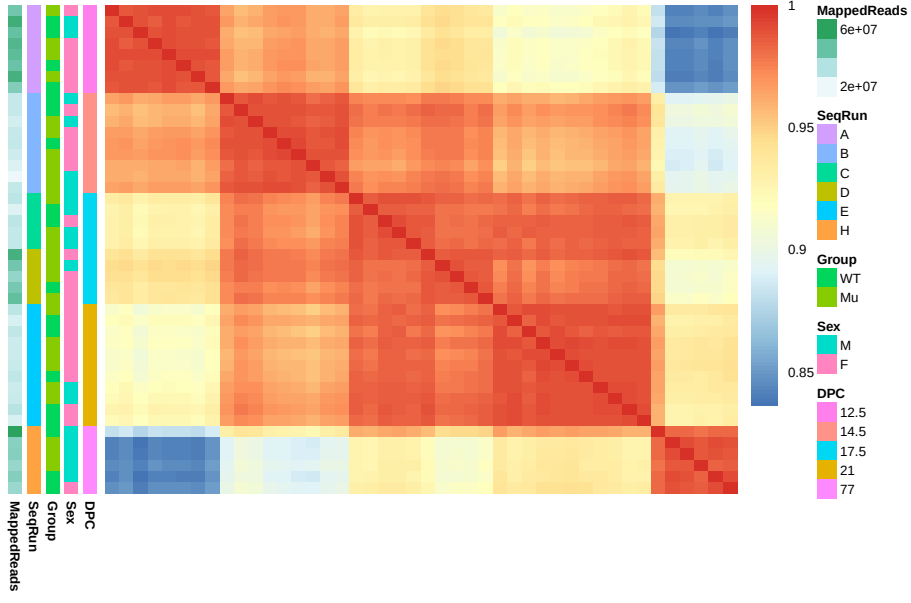
Read depth variation is a potential source of **confounding**!

- We typically want to compare gene counts **between** samples

If we sequence one group of samples 2X as much, gene counts in that sample look ~2X as large even if there's no DE!

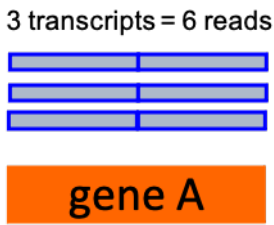
You may come across (older) literature where data was down-sampled to make library sizes the same (**not recommended**)

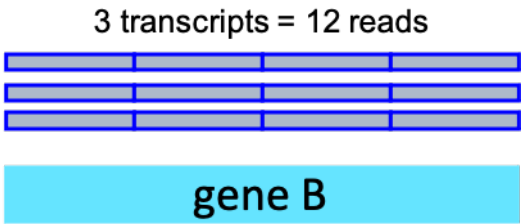
Recall: What does “MappedReads” mean here?



Within-sample comparisons

- Other factors of variation come into play if we also want to compare counts between genes within sample (less common)
- At the same expression level, longer genes/transcripts have more read counts





How can we make fair between- and within-sample comparisons?

- **Normalized expression units**: expression values adjusted for factors like library size, gene length
  - e.g.RPKM/FPKM, TPM, CPM
  - useful for visualization / clustering
- **Normalization factors**: scalar values representing relative library size of each sample
  - e.g. TMM, DESeq size factors
  - useful to include in models of raw counts to adjust for library size

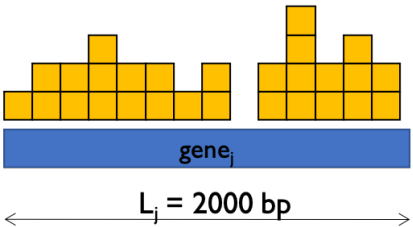
How can we make fair between- and within-sample comparisons?

- **Normalized expression units**: expression values adjusted for factors like library size, gene length
  - e.g.RPKM/FPKM, TPM, CPM
  - useful for visualization / clustering
- **Normalization factors**: scalar values representing relative library size of each sample
  - e.g. TMM, DESeq size factors
  - useful to include in models of raw counts to adjust for library size
- For analysis (e.g. DE) it is ideal to start with **raw counts**
  - raw counts required for many methods
  - can always compute normalized values from raw counts (but not vice versa)

Normalized expression units

RPKM/FPKM

reads/**f**ragments **p**er **k**b of exon per **m**illion mapped reads



If we have:

- $R_{ij} = 28$ reads in gene j, sample i
- $\Sigma_j R_{ij} = 11$ million reads in sample i

Then, **RPKM**:

$$\text{RPKM}_{ij} = \frac{R_{ij}}{\frac{L_j}{10^3}\frac{\Sigma_j R_{ij}}{10^6}} = \frac{28}{\frac{2000}{10^3}\frac{1.1\times10^7}{10^6}} = 1.27$$

Normalized expression units, continued

- **TPM**: Transcripts per million

$$TPM_{ij} = \frac{R_{ij}}{L_j} \frac{10^6}{\Sigma_j R_{ij}/L_j} = \frac{FPKM_{ij}}{\Sigma_j FPKM_{ij}/10^6}$$

Normalized expression units, continued

- **TPM**: Transcripts per million

$$TPM_{ij} = \frac{R_{ij}}{L_j} \frac{10^6}{\Sigma_j R_{ij}/L_j} = \frac{FPKM_{ij}}{\Sigma_j FPKM_{ij}/10^6}$$

- **CPM**: Counts per million

$$CPM_{ij} = \frac{R_{ij}}{\Sigma_j R_{ij}/10^6}$$

Normalized expression units, continued

- **TPM**: Transcripts per million

$$TPM_{ij} = \frac{R_{ij}}{L_j} \frac{10^6}{\Sigma_j R_{ij}/L_j} = \frac{FPKM_{ij}}{\Sigma_j FPKM_{ij}/10^6}$$

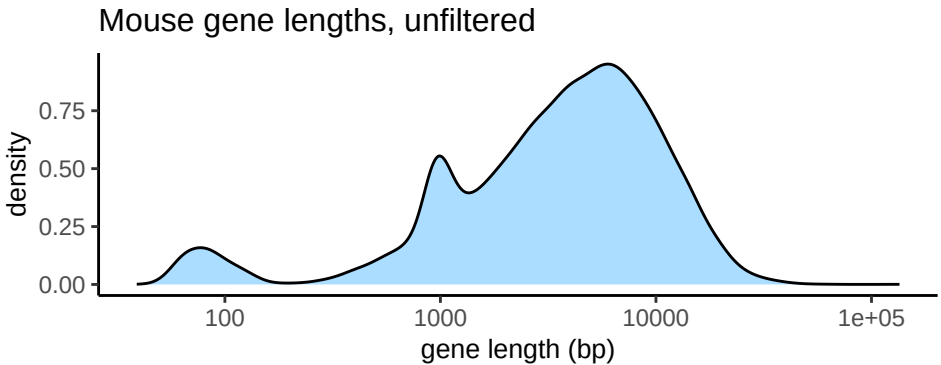
- **CPM**: Counts per million

$$CPM_{ij} = \frac{R_{ij}}{\Sigma_j R_{ij}/10^6}$$

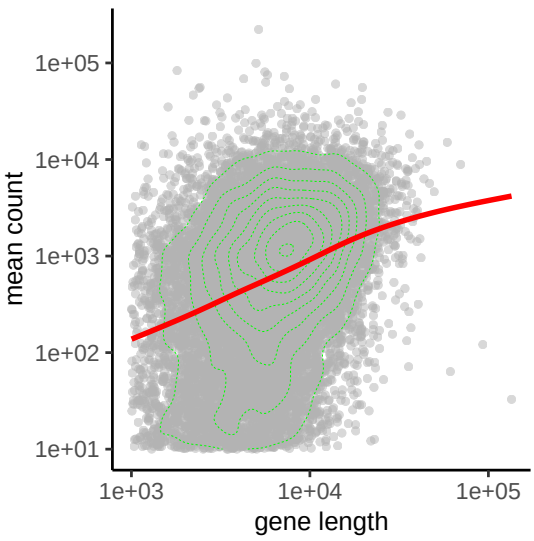
- See this useful blog post on relationship between these units
- Which of these measures are between-sample normalization measures? Within-sample? Both?

How much does “gene length” vary?

- “total effective length of transcript used in assigning reads to genes”
- If all genes are same lengths, FPKM won’t do anything interesting
- In mouse, “gene length” varies mostly between ~2.5Kb - 4.3Kb; your organism may vary



How does gene length relate to counts?

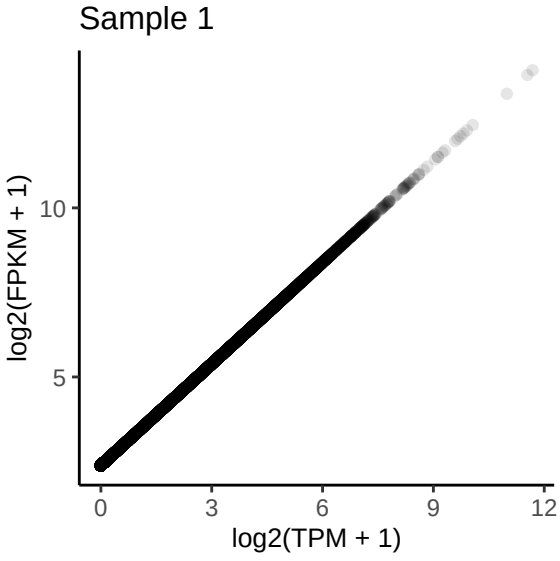


- If all genes were expressed at same level (same # molecules/cell), expect a 1:1 relation
- Of course they are not, so the effect of length is less obvious
- Rank correlation between length and mean expression in our example data is 0.588

FPKM vs. TPM: Which one should we take?

Remark

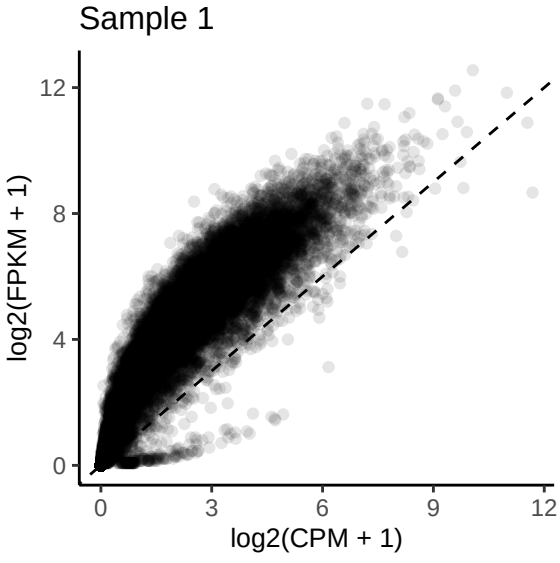
- These metrics both enable comparison of expression levels of different genes within sample.
- Any doubt about "gene length" will be propagated to both measures.



FPKM vs. CPM: Which one should we take?

Remark

- If we're comparing samples to each other, there's no important difference between FPKM/TPM and CPM
- Assuming that "effective gene length" is held constant across samples

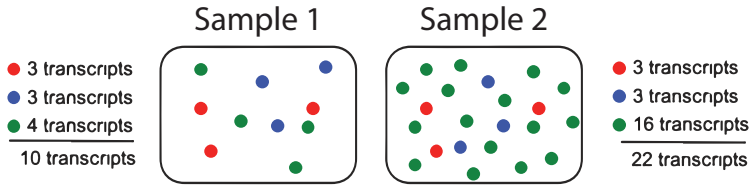


Normalization for comparison between samples?

- Computing FPKM, CPM or TPM largely corrects for differences in library size
- However, there is a complication: "Sequence space"
  - Finite number of reads $\Longrightarrow$ high expression of one gene $\uparrow$ will lead to low expression for other genes $\downarrow$
  - This is a fundamental difference from microarrays, where each spot is essentially independent
- This isn't a major problem unless there are large differences in composition between samples, but should be inspected
  - Normalization factors are generally robust to this

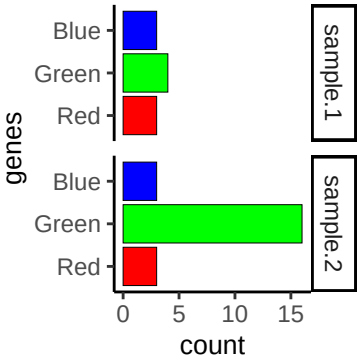
Should we always normalize within each sample? What could go wrong with that?

Consider the following example:



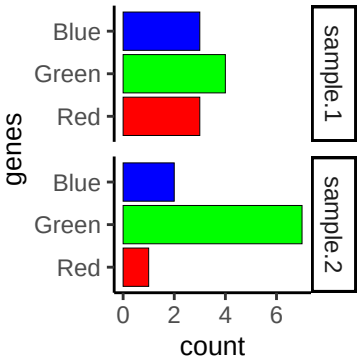

```
genes <- c("Red","Blue","Green")
n.x <- 10
x.prob <- c(3, 3, 4) %>% (function(p)p/sum(p))
n.y <- 22
y.prob <- c(3, 3, 16) %>% (function(p)p/sum(p))
```

What if we sequence “deep” enough to capture all of them?




```
##       sample.1 sample.2
## Red          3        3
## Green        4       16
## Blue         3        3
```

What if we sequence *less* than the actual counts?

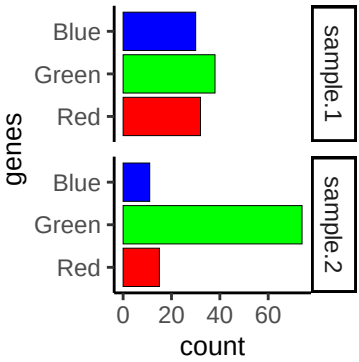



```
##       sample.1 sample.2
## Red          3        1
## Green        4        7
## Blue         3        2
```

Question

- Is it okay to normalize to have both samples have the same sequence depth?
- Are we comparing the gene "red" with the gene "green" or vice versa?

What if we sequence *more* than the actual counts?

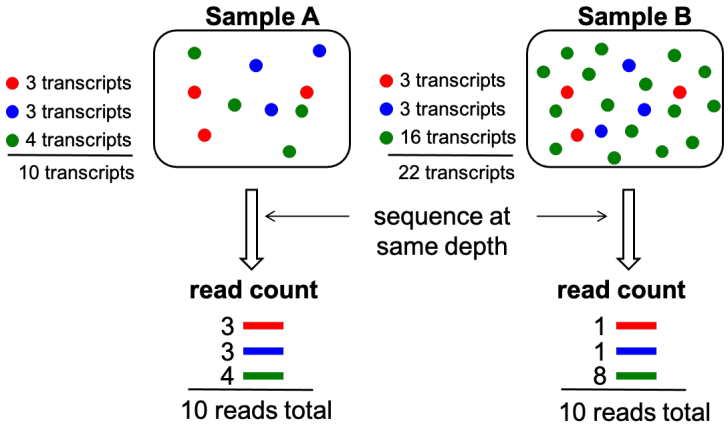



```
##       sample.1 sample.2
## Red         32        15
## Green       38        74
## Blue        30        11
```

Question

- Is it okay to normalize to have both samples have the same sequence depth?
- Are we comparing different genes across different samples?

Effect of sequence space

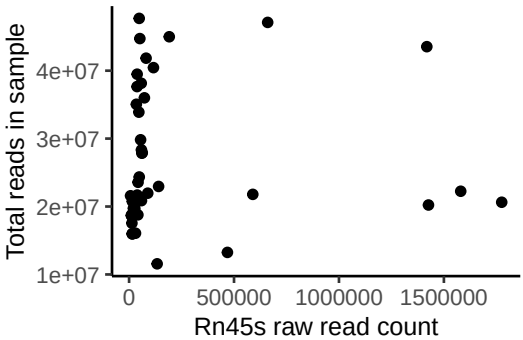


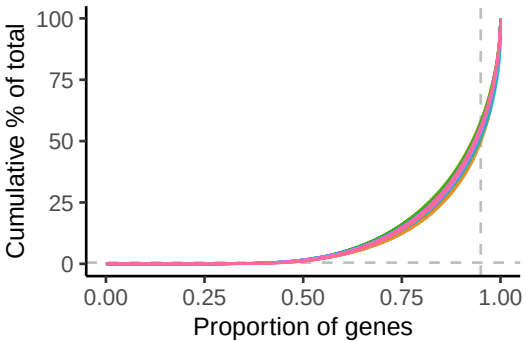
- By CPM or FPKM, red & blue appear down-regulated in sample B (only green is DE)
- Adjusting expression levels in Sample B by a factor of 3 would be needed

See Robinson and Oshlack (2010)

Sequence space in our example data

- Rn45s gene has $> 100,000$ mean reads per sample ($> 5\%$ of reads in some samples)
- ~5% of the genes take up ~50% of reads, but this is consistent across samples
- Side note: Rn45s is potentially a contaminant - a ribosomal RNA that should have been removed during sample prep, which involved poly-A selection





TMM normalization – accounting for "sequence space"

TMM (trimmed means of M-values)

- M-values: per-gene ratios of counts among samples
- Trimmed: extreme values are ignored
- Values adjusted to have product=1
- Assumes that no more than half of genes are DE

- Calculate with `edgeR::calcNormFactors`

```
# construct DGEList object
dge <-
    DGEList(
        assays(sumexp)$counts
    )
# calculate normlaization
# factor (check the default)
dge <- calcNormFactors(dge)
```

Additional preprocessing (generally helpful): filtering lowly expressed genes

- Common step which can be beneficial for a few reasons:
  - Genes with very low mean expression across samples may be uninteresting
  - Fitting models on a smaller number of genes can be faster
  - May obtain a more 'well-behaved' association between mean and variance, which might affect some methods (e.g. `Voom`)
- No universal threshold (depending on the study)

Filtering out lowly expressed genes

Here: Keep genes with ≥ 2 samples that have CPM > 10

```
assays(sumexp)$cpm <- cpm(counts, log = FALSE,
                          normalized.lib.sizes = FALSE)
keep <- which(rowSums(assays(sumexp)$cpm > 10) >= 2)
length(keep)
```

```
## [1] 12158
```

```
sumexp <- sumexp[keep,]
```

Remark

Slice out both `rowData` and `assays`.

Today's lecture

Exploratory Data Analysis of RNA-seq data

Data normalization

Statistical Inference of gene-by-gene analysis

Differential expression: Why we need new methods

- **Goal**: accurate p-values for our hypothesis tests
  - Accurate: "Uniform under the null"
  - Properties relied upon for inference from t-statistics may not hold for count data
  - RNA-seq often contains relatively small sample size
- Perhaps most important: **Heteroskedasticity** and **Overdispersion**
  - Strong mean-variance relationship expected with count data
    - violation of constant variance assumption of linear models
    - over- or under- shrinkage of genes, depending on variance levels
  - Biological variance over and above binomial sampling variance

Statistics of counts: Binomial

- Number of reads observed for gene g in a given sample is a random variable
- Say RNA for gene g is present "in the cell" at about 1 out of every 1,000,000 molecules
  - Abundance $a_g = 1/1,000,000 = 1 \times 10^{-6}$ ("probability of success")
- If we randomly pick $R_i = \Sigma_g R_{ig} = 1,000,000$ molecules ("reads" = "trials"), how many gene g RNAs will we see? $E(R_{ig}|R_i) = \ ?$

Statistics of counts: Binomial

- Number of reads observed for gene g in a given sample is a random variable
- Say RNA for gene g is present "in the cell" at about 1 out of every 1,000,000 molecules
  - Abundance $a_g = 1/1,000,000 = 1 \times 10^{-6}$ ("probability of success")
- If we randomly pick $R_i = \Sigma_g R_{ig} = 1,000,000$ molecules ("reads" = "trials"), how many gene g RNAs will we see? $E(R_{ig}|R_i) = \ ?$
- But could get 0, 2, 3, 4, ... etc just by chance: this is a **Binomial** distribution
  - probability distribution of the number of successes in n trials, each with probability of success p is $(Binomial(n,p))$
  - mean $= np$
  - our example: $R_{ig} \sim Binomial(R_i, a_g)$ where $n = R_i$ and $p = a_g$

Statistics of counts: Poisson

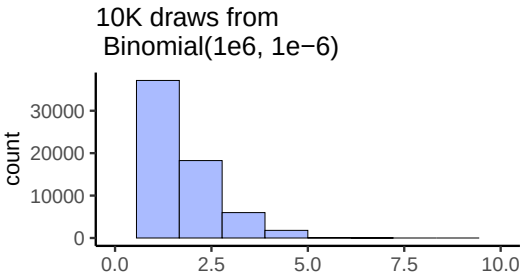
- Poisson distribution counts discrete occurrences along a continuous interval of time/space
  - parameterized by a rate parameter λ
  - key difference from Binomial: number of events can be infinitely large
- For count data, the variance is a function of the mean.
  - Binomial: mean $= np$, variance $= np(1-p)$
  - Poisson: mean $=$ variance $= \lambda$

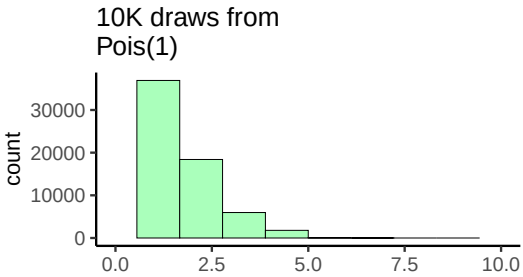
Statistics of counts: approximations (moment matching)

- **Binomial approximated by Poisson**: for large n and small np (rule of thumb: $n > 20$ & $np < 5$)
  - Approximately $R_{ig} \sim Poisson(R_i a_g)$
- **Binomial approximated by Normal**: For large np (rule of thumb: np & $n(1-p) > 5$)
  - Approximately $R_{ig} \sim Normal(R_i a_g,\, R_i a_g(1-a_g))$

Statistics of counts: approximations (moment matching)

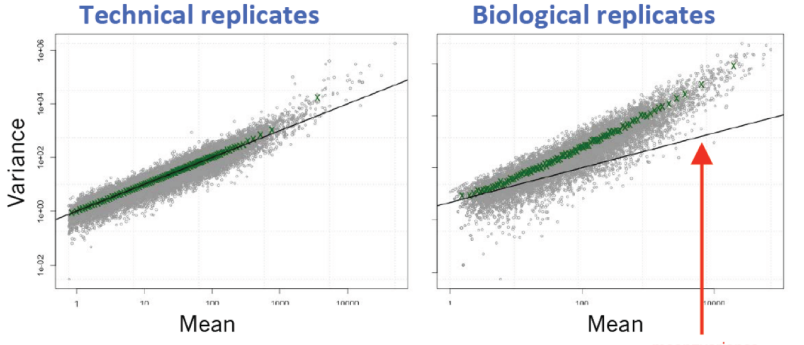
- **Binomial approximated by Poisson**: for large n and small np (rule of thumb: $n > 20$ & $np < 5$)
  - Approximately $R_{ig} \sim Poisson(R_i a_g)$
- **Binomial approximated by Normal**: For large np (rule of thumb: np & $n(1-p) > 5$)
  - Approximately $R_{ig} \sim Normal(R_i a_g,\ R_i a_g(1-a_g))$





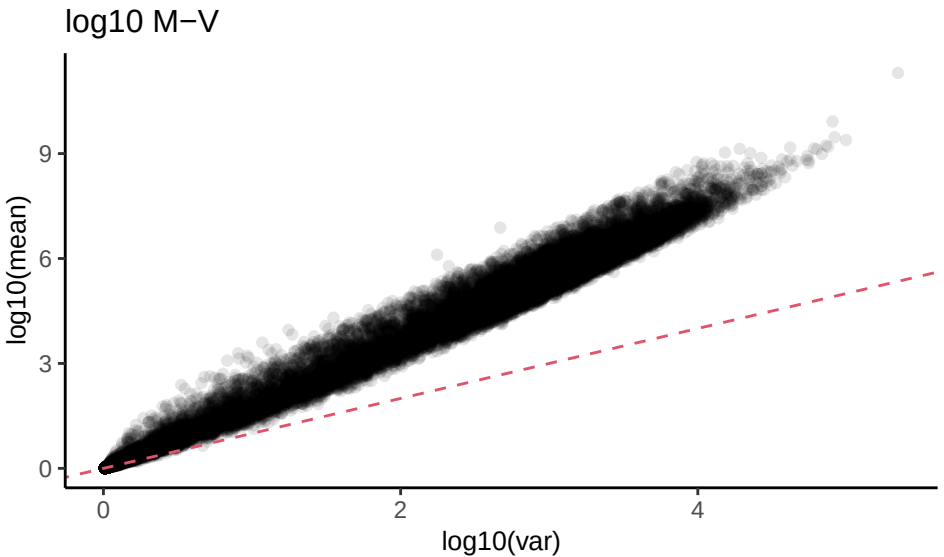
Overdispersion = non-linear relationship between the mean and variance in RNA-seq data

Poisson OK for technical replicates, but **does not capture biological variability**



Overdispersion (across genes) is commonly observed!

In the CHD8 data:



Heteroskedastic sampling (across samples): Variance changes with the mean

Ordinary Least Square

Assume all errors have the same variance (within gene)

$\min \sum_i (Y_i - X_i\beta)^2$

Weighted Least Square

Some observations/points have higher/lower variance (within gene)

$\min \sum_i W_i (Y_i - X_i\beta)^2$

Discussion: Why bother heteroskedastic properties and mean-variance relationships?

In other words, why can't we just use traditional statistical methods (built on asymptotic normality assumptions[3])?

[3] $n \to \infty$, random variables will behave like Normal

Discussion: Why bother heteroskedastic properties and mean-variance relationships?

In other words, why can't we just use traditional statistical methods (built on asymptotic normality assumptions[3])?

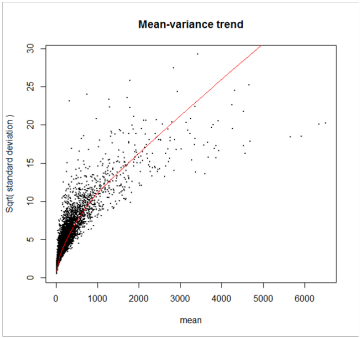
How to deal with heteroskedacity

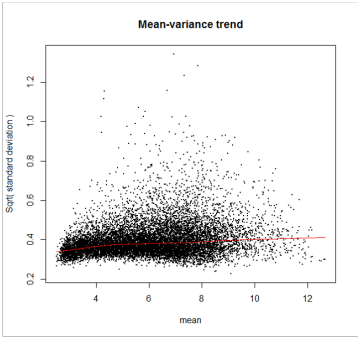
1. Transformation: Make them homoskedastic & reverse the mean-variance relationships
2. Modelling: generalized linear model of count data with overdispersion parameters
3. Non-parametric tests (e.g., top-scoring pair, Geman *et al.* 2004)

[3] $n \to \infty$, random variables will behave like Normal

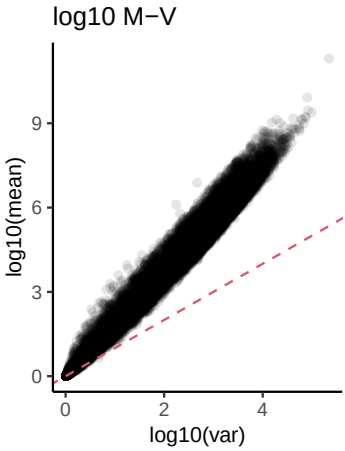
Make adjustments & model as usual: transformation

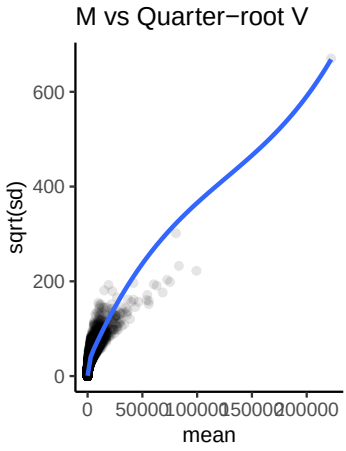
- For microarray data, taking logs is often deemed sufficient to reduce M-V trends
- We'll use plots like this which are mean vs $\sqrt{sd}$ (quarter root variance) instead of mean vs. variance (you'll see why later on)

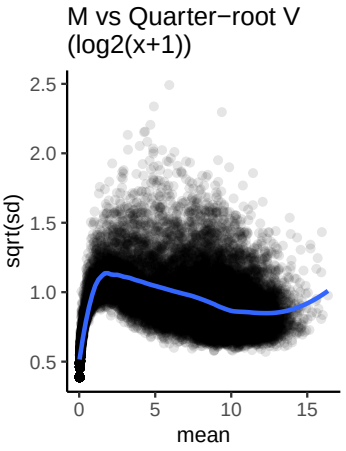




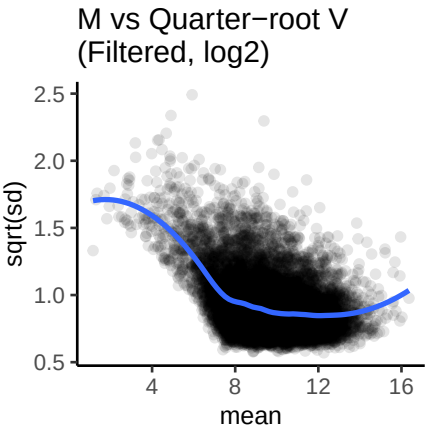
Chd8 data & effect of log transform (the problem lingers)



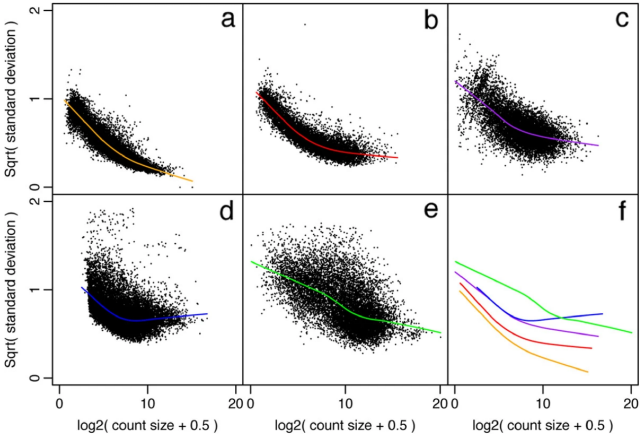




Mean variance trends in various RNA-seq datasets



Chd8 dataset (Filtered to remove lowly expressed genes, log2-transformed)



Panels (a)-(e) represent datasets with increasing expected biological variability

Source: Law et al. 2014

One option: Voom can "straighten" crooked mean-variance relationships

Read this paper "voom: precision weights unlock linear model analysis tools for RNA-seq read counts" by Law et al. (2014)

Mean-variance modelling at the observational level

Falls under the category *"Make adjustments and model as usual"*

Specifically, adjustment to regular `lm` to account for M-V relationship + `limma`

One option: Voom can "straighten" crooked mean-variance relationships

Read this paper "voom: precision weights unlock linear model analysis tools for RNA-seq read counts" by Law et al. (2014)

Mean-variance modelling at the observational level

Falls under the category *"Make adjustments and model as usual"*

Specifically, adjustment to regular `lm` to account for M-V relationship + `limma`

Key take-away of the Voom paper:

1. heteroskedastic, sample-specific higher variance $\implies$ getting more weight in OLS
2. modelling the mean-variance relationship is more important than getting the probability distribution exactly right

Voom implementation

Voom Input:

1. **raw counts** (required to estimate M-V relationship), but modeling is done on log-transformed CPM values ($(\log_2(CPM + 0.5)$ to be precise)
2. design matrix

Voom Output:

- Precision weights and moderated t-statistics
- Implemented in `limma::voom()` function

Voom steps

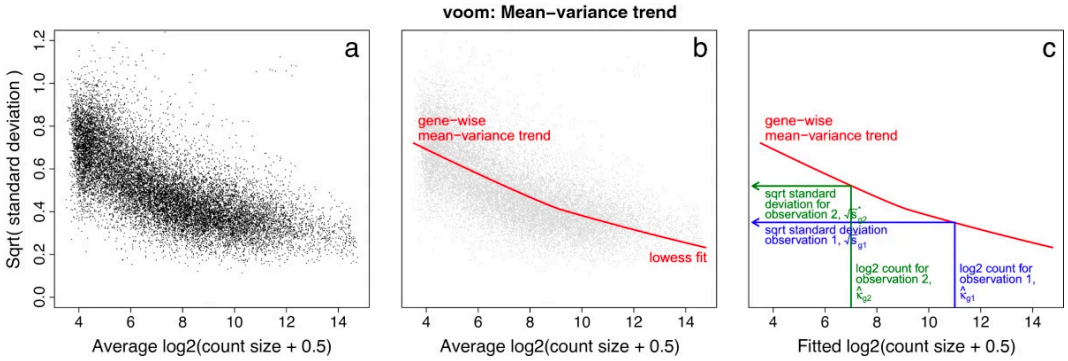
1. Fit linear model to $\log_2(CPM_{ig} + 0.5)$ values (samples i) for each gene g
2. Extract the fitted quarter-root error variance estimates $s_g^{1/2} = \sqrt{\text{sd}(\hat{\varepsilon_{ig}})}$
3. Fit a smoothed line $\hat{f}$ to the trend between mean log counts and $s_g^{1/2}$ using lowess (locally weighted regression)

Voom steps

1. Fit linear model to $\log_2(CPM_{ig} + 0.5)$ values (samples i) for each gene g
2. Extract the fitted quarter-root error variance estimates $s_g^{1/2} = \sqrt{\text{sd}(\hat{\varepsilon_{ig}})}$
3. Fit a smoothed line $\hat{f}$ to the trend between mean log counts and $s_g^{1/2}$ using lowess (locally weighted regression)
4. Use the fitted lowess curve to estimate **precision weights**: $w_{ig} = 1/\hat{f}(\hat{c}_{ig})^4$ where $\hat{c}_{ig}$ are the $\log_2$ *fitted* counts (estimated from model in step 1)
5. Fit linear model to $\log_2(\text{CPM}_{ig} + 0.5)$ values using **precision weights** w_{ig}
6. Compute moderated t-statistics as before (using `eBayes` from `limma`)

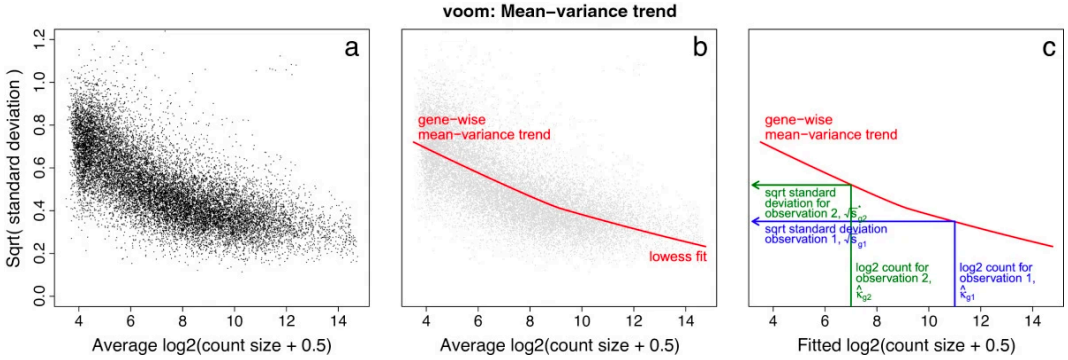
Voom illustration

Figure 2, Law et. al, 2014



Voom illustration

Figure 2, Law et. al, 2014



$$w_{ig} = \frac{1}{\hat{f}(\hat{c}_{ig})^4} = \frac{1}{(\sqrt{s_{ig}})^4} = \frac{1}{s_{ig}^2}$$

What do we do with these ‘precision weights’?

How can we actually incorporate these precision weights in the regression fit?

Weighted least squares (WLS) regression

- OLS: $\hat{\beta}_g = (X^T X)^{-1} X^T y_g$
- WLS: $\hat{\beta}_g = (X^T W_g X)^{-1} X^T W_g y_g$, where W_g is a diagonal matrix of weights for gene g
- **Intuition**: in minimizing the RSS, we put less weight on data points that are less precise:

$$\hat{\beta}_g = \underset{\beta_{g1},\ldots,\beta_{gp}}{\arg\min} \sum_{i=1}^{n} w_{ig}(x_{i1}\beta_{g1} + \ldots + x_{ip}\beta_{gp} - y_{ig})^2$$

[4] Note: parameter estimates $\hat{\beta}_g$ assume weights (variances) are known

Weighted least squares (WLS) regression

- OLS: $\hat{\beta}_g = (X^T X)^{-1} X^T y_g$
- WLS: $\hat{\beta}_g = (X^T W_g X)^{-1} X^T W_g y_g$, where W_g is a diagonal matrix of weights for gene g
- **Intuition**: in minimizing the RSS, we put less weight on data points that are less precise:

$$\hat{\beta}_g = \underset{\beta_{g1},\ldots,\beta_{gp}}{\arg\min} \sum_{i=1}^{n} w_{ig}(x_{i1}\beta_{g1} + \ldots + x_{ip}\beta_{gp} - y_{ig})^2$$

- Optimal weights to correct for heteroskedasticity: inverse variance[4]

[4] Note: parameter estimates $\hat{\beta}_g$ assume weights (variances) are known

limma-voom

- **limma-voom** is the application of `limma` to $log_2(CPM + 0.5)$ 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:

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

limma-voom

- **limma-voom** is the application of `limma` to $log_2(CPM + 0.5)$ 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 $\frac{1}{n-p} \hat{\epsilon_g}^T \hat{\epsilon_g}$
  - under WLS $\hat{\epsilon}_g = y_g - X\hat{\beta}_g = y_g - X(X^T W_g X)^{-1} X^T W_g y_g$

limma-voom, continued

- Moderated t statistics are then calculated using the shrunken gene-specific variance estimates: $\tilde{t}_g = \frac{\hat{\beta}_{ig}}{\tilde{s}_g \sqrt{v_{ii}}}$
  - recall that under OLS, v_{ii} is the i^{th} diagonal element of $(X^T X)^{-1}$
  - under WLS, v_{ii} is the i^{th} diagonal element of $(X^T W_g X)^{-1}$
- Recall:
  - Degrees of freedom for moderated t statistic: $n - p + d_0$
  - If d_0 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: Our **additive** model for each gene to test for Group (Chd8 mutant vs WT) effect, and adjust for:

$$\begin{aligned} Y_i &= \theta + \tau_{Mut} X_{i,Mut} + \tau_F X_{i,F} \\ &\quad + \tau_{D14.5} X_{i,D14.5} + \tau_{D17.5} X_{i,D17.5} + \tau_{D21} X_{i,D21} + \tau_{D77} X_{i,D77} \\ &\quad + \epsilon_i \end{aligned}$$

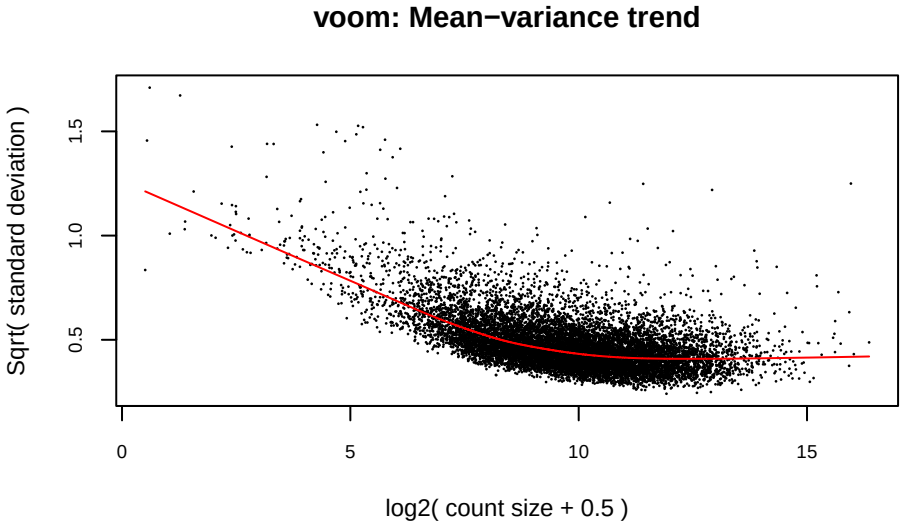
- Our model has $n - p = 44 - 7 = 37$ degrees of freedom
- We will focus on the null hypothesis of the **main effect** of Group $H_0 : \tau_{Mut} = 0$

limma-voom in action

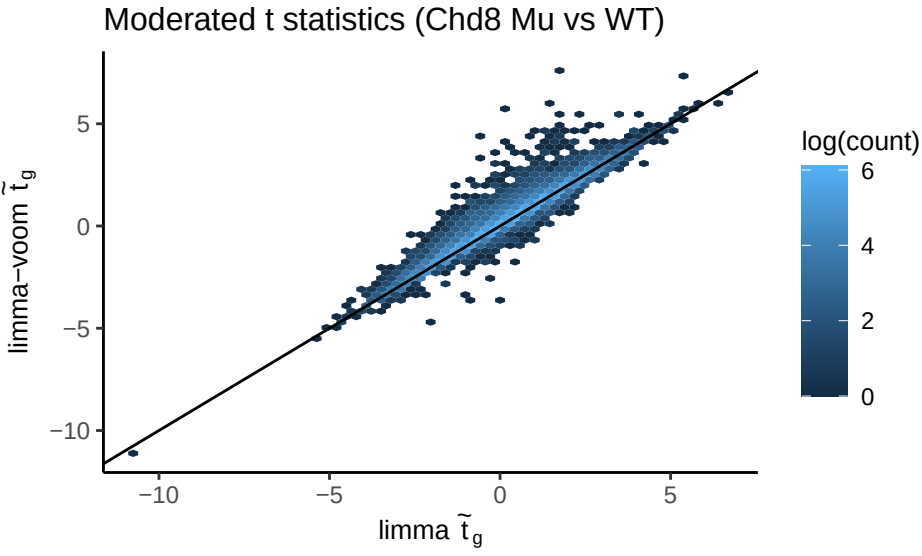
```
.mod <-
    model.matrix( ~ Sex +
                  Group +
                  DPC,
      data = colData(sumexp))

# Estimate voom weights;
# plot M-V trend
vw <-
    voom(assays(sumexp)$counts,
         design=.mod,
         plot=T, span = 1/2)
```

```
# run limma with voom weights
lvfit <- lmFit(vw, model.mat)
lvfit <- eBayes(lvfit)
```

limma-voom in action



limma-voom vs. limma:



Another option: limma-trend

- Compared to voom, limma-trend treats all observations within a gene the same
- Compared to regular limma, limma-trend shrinks gene-specific variances toward a **global M-V trend**, instead of toward a constant pooled variance:

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

Another option: limma-trend

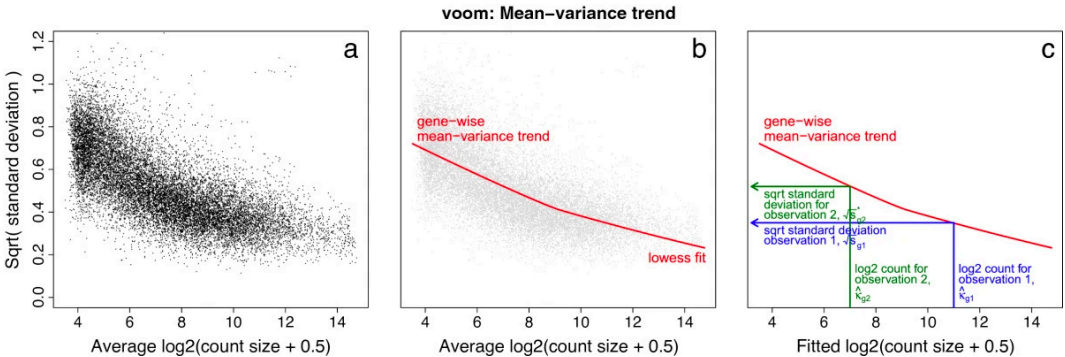
- Compared to voom, limma-trend treats all observations within a gene the same
- Compared to regular limma, limma-trend shrinks gene-specific variances toward a **global M-V trend**, instead of toward a constant pooled variance:

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

- 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 vs voom?

Figure 2, Law et. al, 2014



- limma obtains one *overall* prior variance for all genes
- limma-trend obtains a prior variance for *each gene* based on it's mean using a smoothed curve of the mean vs variance
- limma-voom obtains a weight for *each observation* from such a curve

limma-trend in action

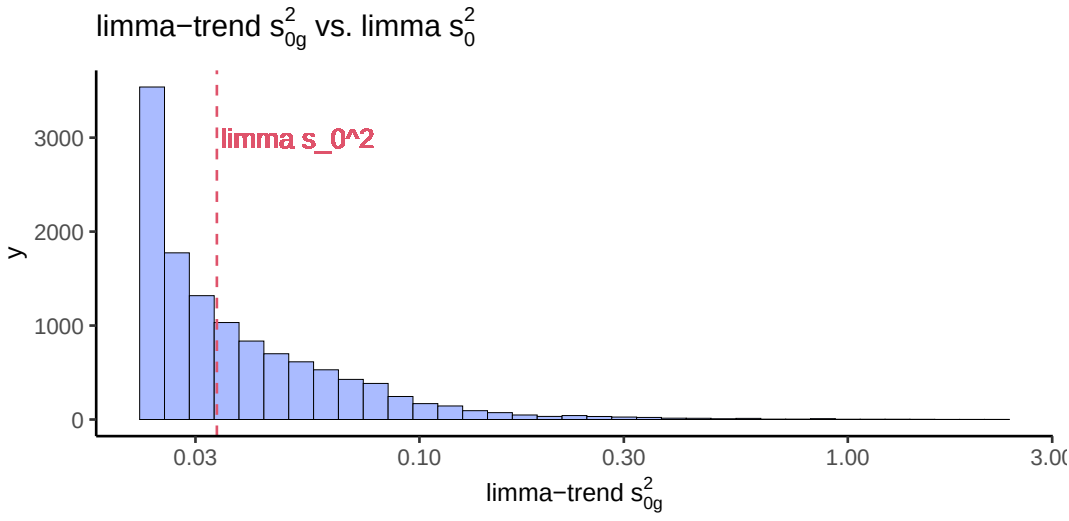
```
mm <- model.matrix(~ Sex + Group + DPC,
                   data = colData(sumexp))
ltfit <- lmFit(cpm(assays(sumexp)$counts,
                   log = TRUE),
               design =  mm)
ltfit <- eBayes(ltfit, trend = TRUE)

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

```
##  Named num [1:12158] 0.0287 0.051 0.0274 0.023 0.0268 ...
##  - attr(*, "names")= chr [1:12158] "0610007P14Rik" "0610009B22Rik"
```

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

```
##  num 0.0337
```

limma-trend in action



`limma-trend` and `limma-voom`: When they will work

- 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 M-V isn't flat, impact is most prominent in lowly expressed genes
- `limma-voom` might produce statistical artifact if the weight estimation becomes brittle
- *always look at the data*

Alternative option: use count models

`DESeq2` `edgeR`

Both assume counts have underlying *Negative Binomial distribution* and fit **generalized linear models**

DESeq2 vs. edgeR: They work similarly

edgeR

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

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

Explore results with `topTags`

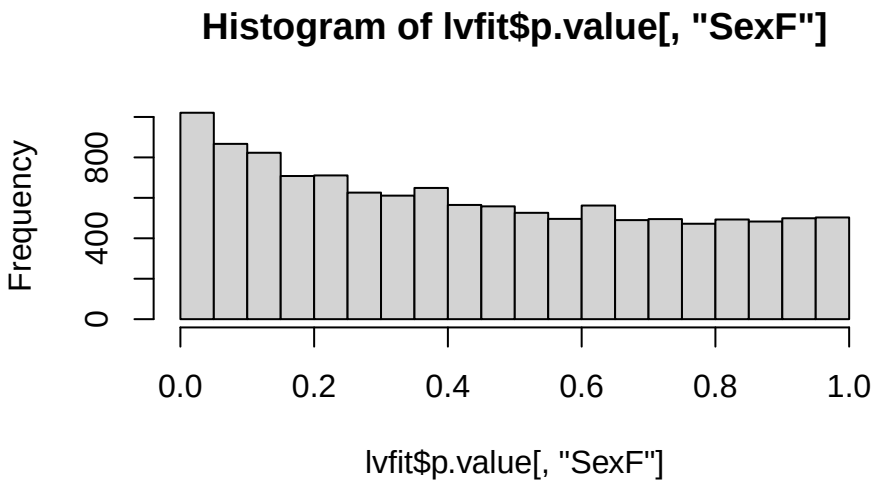
DESeq2

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

dds <- estimateSizeFactors(dds)

dds <- DESeq(dds)
```

Explore results with `results`

We will discuss GLM, including Negative Binomial, later in Lect14!

Before you go: Always inspect your p-value distribution



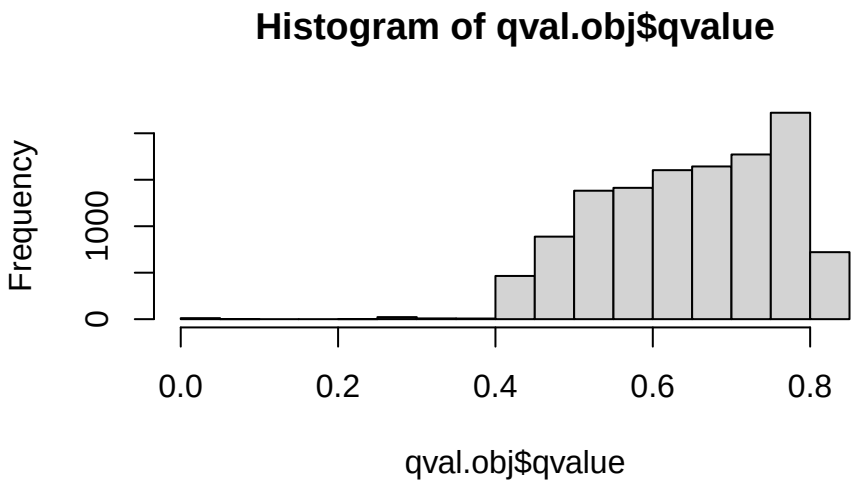
John Storey's FDR correction

```
qval.obj <- qvalue(lvfit$p.value[,"SexF"])
```

```
names(qval.obj)
```

```
## [1] "call"        "pi0"         "qvalues"     "pvalues"     "lfdr"
## [6] "pi0.lambda" "lambda"      "pi0.smooth"
```

John Storey's FDR correction



Top results

```
topTable(lvfit, coef="GroupMu", number=5, lfc=1.5)
```

```
##          logFC   AveExpr        t      P.Value    adj.P.Val        B
## Plin4 2.068068 -1.859445 7.529917 2.443131e-09 1.389053e-05 8.375762
## Gabrp 3.185171 -3.062475 4.658718 3.155014e-05 7.740824e-03 1.965542
## Krt5  2.785548 -2.228423 4.586066 3.980591e-05 8.711863e-03 1.756130
```

Top results

```
topTable(lvfit, coef="SexF", number=5, lfc=1.5)
```

```
##              logFC    AveExpr         t      P.Value    adj.P.Val        B
## Kdm5d   -10.404765 -0.8064703 -41.89461 4.324862e-36 5.258168e-32 44.56316
## Uty      -9.371695 -1.0833585 -38.19536 1.960770e-34 1.191952e-30 43.96359
## Ddx3y   -10.520081 -0.3448537 -33.27225 5.608562e-32 2.272963e-28 42.27344
## Xist      9.973158  4.6980141  22.98111 1.569246e-25 4.769722e-22 41.16651
## Eif2s3y -10.431961 -0.7349739 -19.41865 1.068643e-22 2.598513e-19 32.67112
```

Companion notes for this lecture with greater detail can be found here

https://github.com/STAT540-UBC/resources/blob/main/rnaseqdiffex-examp