The Pasilla Data and Analysis with DESeq

BST 226

Statistical Methods for Bioinformatics

David M. Rocke

Pasilla Data

- Pasilla is a drosophila gene that regulates splicing and is orthologous to the mammalian genes NOVA1 and NOVA2.
- Four samples of drosophila cells were treated with an interfering RNA for pasilla, and three were controls (as listed in the paper(
- The data set in Bioconductor has three treated and four controls. So we will use the annotations in the pasilla Bioconductor data set.
- The `pasillaGenes` data is 14470 by 7 and the `pasillaExons` data set is 498 by 7.

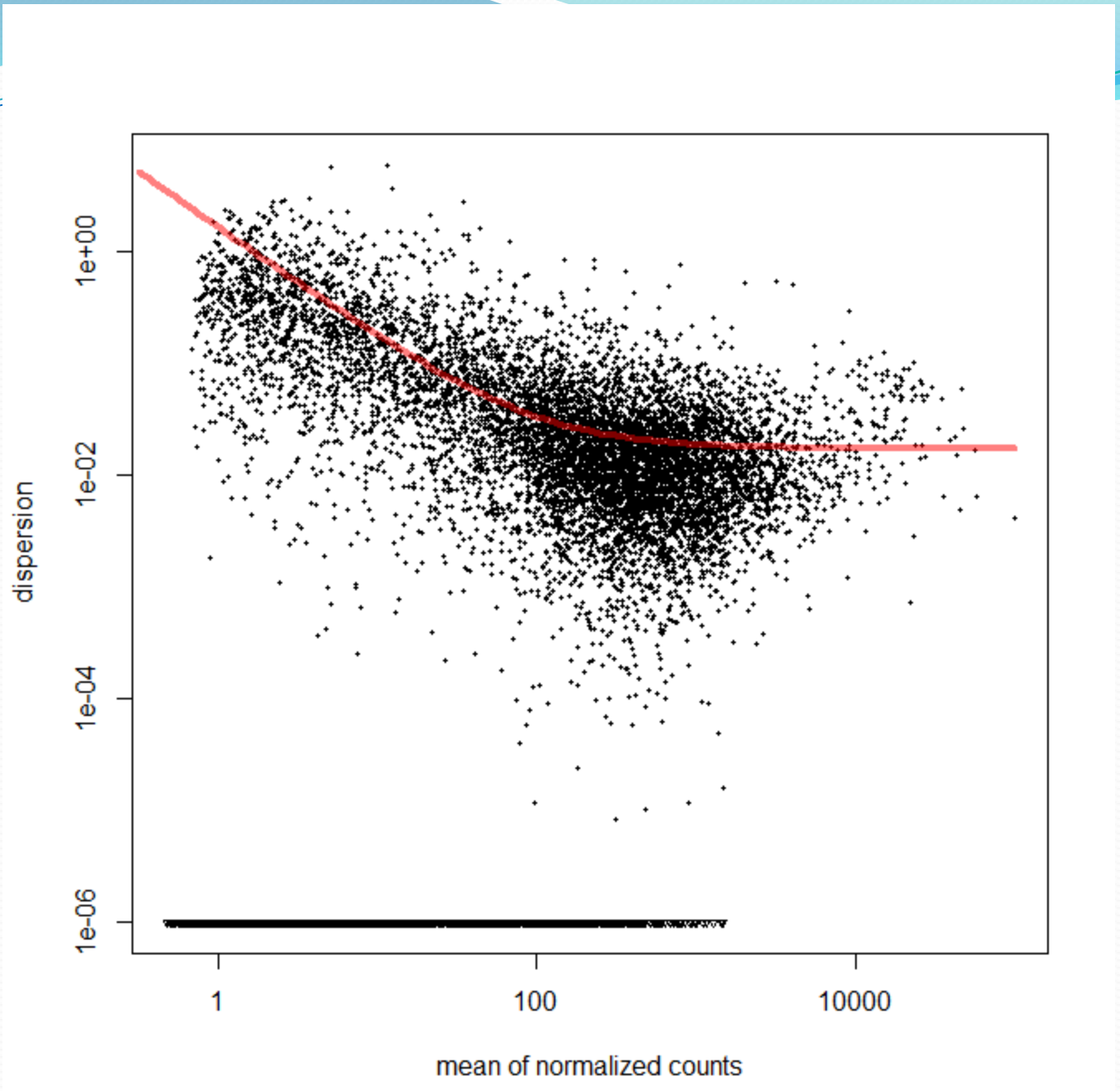
- The basic unit of analysis is a count table, consisting of one row per gene/exon/protein and one column per sample, with counts as the entries in the matrix.
- The samples need to be biological replicates, not technical replicates.
- The objects `pasillaGenes` and `pasillaExons` are `CountDataSet` objects.
- They contain the count table and other information that can be used to do the analysis.
- At this point, we will not cover coversion of raw RNA-Seq data to a `CountDataObject`
- `library(pasilla)` also loads the DESeq package.

```
> library(pasilla)
> data(pasillaGenes)
> dim(counts(pasillaGenes))
[1] 14470     7

# counts() extracts the count table from a CountDataSet object

> pData(pasillaGenes)
             sizeFactor condition        type
treated1fb           NA   treated single-read
treated2fb           NA   treated  paired-end
treated3fb           NA   treated  paired-end
untreated1fb         NA untreated single-read
untreated2fb         NA untreated single-read
untreated3fb         NA untreated  paired-end
untreated4fb         NA untreated  paired-end

# pData() extracts the information about the samples.
# conditions just gets the variables

> conditions(pasillaGenes)
  treated1fb   treated2fb   treated3fb untreated1fb untreated2fb untreated3fb untreated4fb
     treated      treated      treated    untreated    untreated    untreated    untreated
Levels: treated untreated
```

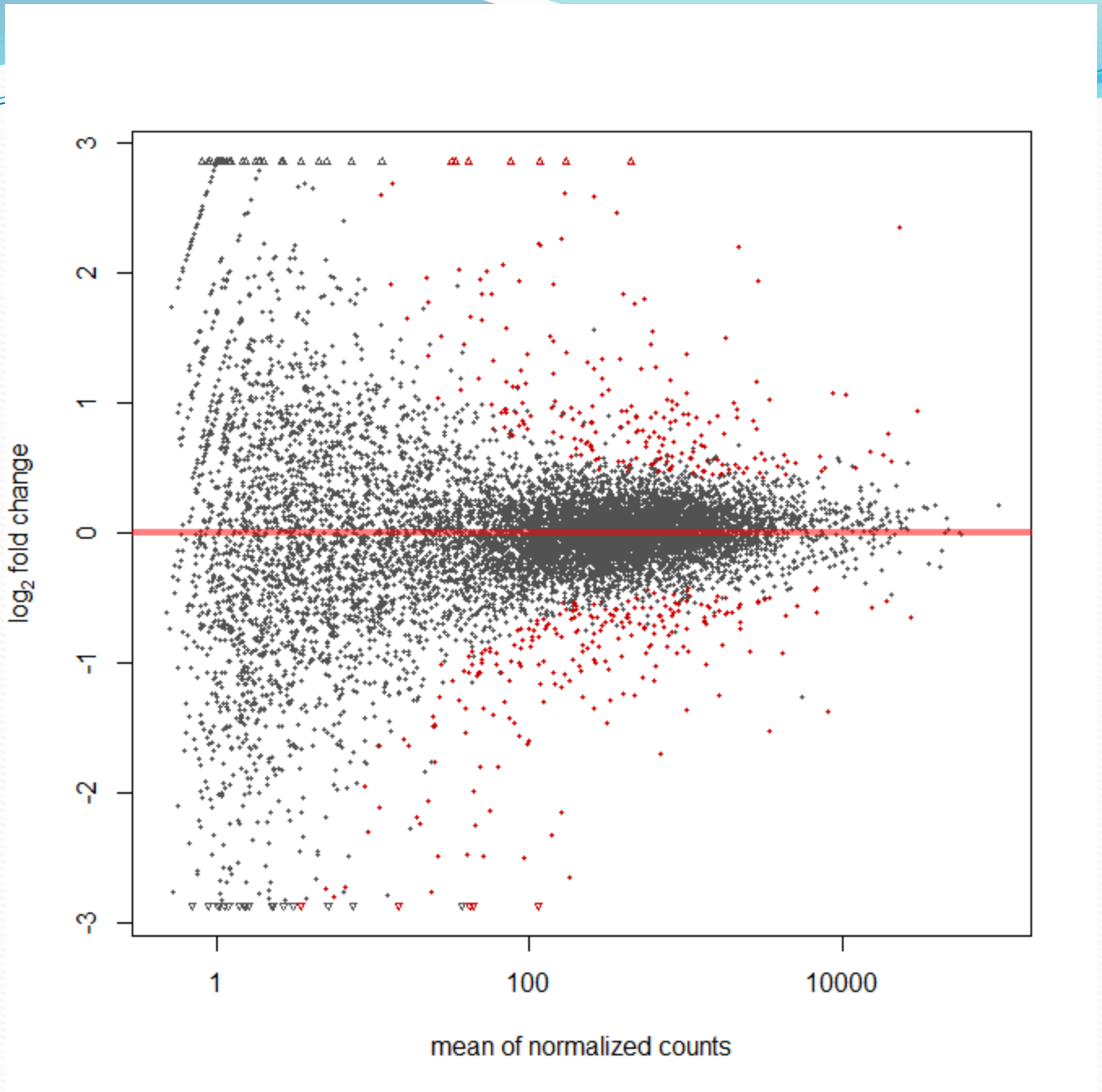
Some workflow for DESeq

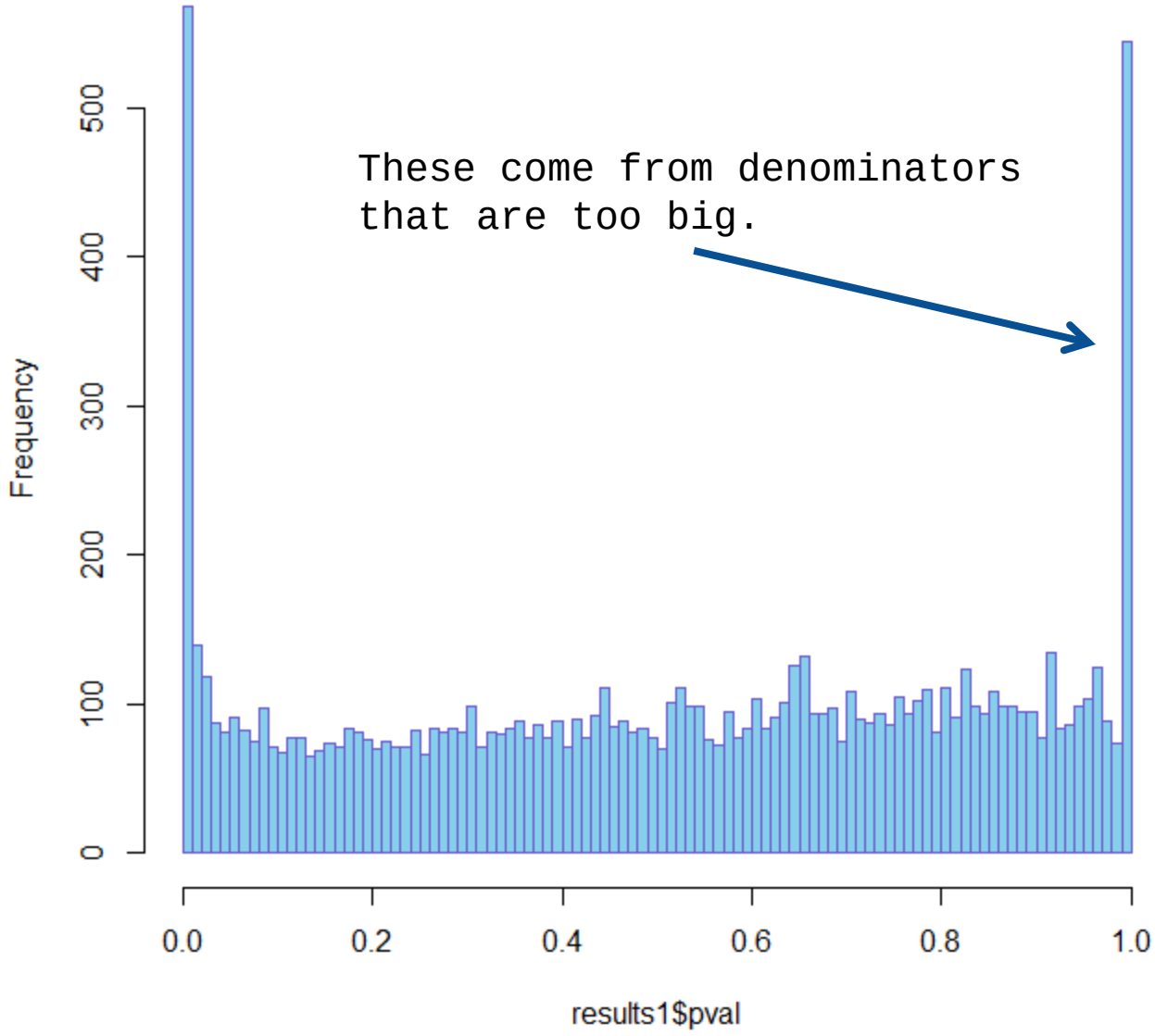
```
> head(counts(pasillaGenes))
            treated1fb treated2fb treated3fb untreated1fb untreated2fb untreated3fb untreated4fb
FBgn0000003          0          0          1            0            0            0            0
FBgn0000008         78         46         43           47           89           53           27
FBgn0000014          2          0          0            0            0            1            0
FBgn0000015          1          0          1            0            1            1            2
FBgn0000017       3187       1672       1859         2445         4615         2063         1711
FBgn0000018        369        150        176          288          383          135          174
……

> sum(apply(counts(pasillaGenes),1,sum)==0)
[1] 2634                                              # 2634/14470 genes have no counts
> zeroes <- apply(counts(pasillaGenes) == 0,1,sum)
> sum(zeroes <= 3)                                    # 9874/14470 have at least 4 non-zero counts
[1] 9874
> pG2 <- newCountDataSet(counts(pasillaGenes)[zeroes <= 3,],conditions(pasillaGenes))
> pData(pG2)
             sizeFactor condition
treated1fb           NA   treated                     # pG2 is the data set with genes eliminated
treated2fb           NA   treated                     # that have too many zero counts
treated3fb           NA   treated
untreated1fb         NA untreated
untreated2fb         NA untreated
untreated3fb         NA untreated
untreated4fb         NA untreated
```

```
> pG2 <- estimateSizeFactors(pG2)
> pData(pG2)
             sizeFactor condition
treated1fb    1.5116926   treated
treated2fb    0.7843521   treated
treated3fb    0.8958321   treated
untreated1fb  1.0499961 untreated
untreated2fb  1.6585559 untreated
untreated3fb  0.7117763 untreated
untreated4fb  0.7837458 untreated

> pG2 <- estimateDispersions(pG2)
> str(fitInfo(pG2))
List of 5
 $ perGeneDispEsts: num [1:9874] 0.048 -0.1054 0.0125 0.0171 0.0758 ...
 $ dispFunc       :function (q)
  ..- attr(*, "coefficients")= Named num [1:2] 0.0171 1.6282
  .. ..- attr(*, "names")= chr [1:2] "asymptDisp" "extraPois"
  ..- attr(*, "fitType")= chr "parametric"
 $ fittedDispEsts : num [1:9874] 0.0483 1.8154 0.0178 0.0244 0.5398 ...
 $ df             : int 5
 $ sharingMode    : chr "maximum“

# Fits a per gene estimate, then a fitted line, and keeps the larger of the two.

> plotDispEsts(pG2)
```


dispersion
1e+00
1e-02
1e-04
1e-06
1
100
10000
mean of normalized counts

```
> head(fData(pG2))
            disp_pooled
FBgn0000008  0.04825177
FBgn0000015  1.81543824
FBgn0000017  0.01776668
FBgn0000018  0.02443547
FBgn0000024  0.53977071
FBgn0000032  0.01968189
……

> results1 <- nbinomTest(pG2,"treated","untreated")
> head(results1)
           id     baseMean    baseMeanA   baseMeanB foldChange log2FoldChange       pval      padj
1 FBgn0000008   52.2256776   52.7483285   51.833689  0.9826603    -0.02523529 0.94438010 1.0000000
2 FBgn0000015    0.9053584    0.5925969    1.139929  1.9236170     0.94382157 1.00000000 1.0000000
3 FBgn0000017 2358.2434078 2105.0313621 2548.152442  1.2105057     0.27560986 0.06364366 0.5630982
4 FBgn0000018  221.2415562  210.6010886  229.221907  1.0884175     0.12223204 0.62637472 1.0000000
5 FBgn0000024    3.1149405    4.1231901    2.358753  0.5720700    -0.80573647 0.77278869 1.0000000
6 FBgn0000032  624.8641873  604.7716429  639.933596  1.0581409     0.08153171 0.75141467 1.0000000

> sum(results1$padj < .1)
[1] 463

> plotMA(results1)
> hist(results1$pval,breaks=100,col="skyblue",border="slateblue",main="")
```


log2 fold change
3
2
1
0
-1
-2
-3
1
100
10000
mean of normalized counts


These come from denominators
that are too big.
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
results1$pval

```
> head(counts(pG2)[results1$pval>.99,],20)
```

	treated1fb	treated2fb	treated3fb	untreated1fb	untreated2fb	untreated3fb	untreated4fb
FBgn0000015	1	0	1	0	1	1	2
FBgn0000055+FBgn0000056	6	2	4	4	7	0	6
FBgn0000075	2	0	1	0	2	2	1
FBgn0000099	2	1	0	0	2	1	1
FBgn0000115	480	293	331	367	581	235	270
FBgn0000303+FBgn0015323	3	0	3	2	4	1	1
FBgn0000357	3	2	2	0	5	3	2
FBgn0000551	0	2	0	1	1	2	0
FBgn0000592	3	0	2	2	1	2	1
FBgn0000658	6	1	2	5	4	2	0
FBgn0001105	3539	1826	2009	2212	4076	1629	1803
FBgn0001319	2	1	1	0	3	1	1
FBgn0001324	1922	1167	1248	1451	2235	1014	1011
FBgn0002036	55	27	35	45	54	18	37
FBgn0002563	1	4	2	0	7	0	3
FBgn0002571	0	1	1	1	1	0	1
FBgn0002592	5	5	5	1	11	4	3
FBgn0002732	2	1	3	4	4	1	0
FBgn0002863	3	1	0	5	1	0	0
FBgn0002936	4	3	1	1	5	0	4

Not just small counts! These very large p-values come from using fitted variances instead of observed variances. Observed are better for large n. Their method never reduces the estimated variance from the observed level, so it is conservative. Otherwise, one could use the EB type method as with gene expression arrays.

```
> pData(pasillaGenes)
             sizeFactor condition        type
treated1fb           NA   treated single-read
treated2fb           NA   treated  paired-end
treated3fb           NA   treated  paired-end
untreated1fb         NA untreated single-read
untreated2fb         NA untreated single-read
untreated3fb         NA untreated  paired-end
untreated4fb         NA untreated  paired-end
> pD2 <- pData(pasillaGenes)[,-1]                          # omit sizeFactor
> pG3 <- newCountDataSet(counts(pG2),pD2)                  # sizeFactor is inserted
> pData(pG3)
             sizeFactor condition        type
treated1fb           NA   treated single-read
treated2fb           NA   treated  paired-end
treated3fb           NA   treated  paired-end
untreated1fb         NA untreated single-read
untreated2fb         NA untreated single-read
untreated3fb         NA untreated  paired-end
untreated4fb         NA untreated  paired-end
> pG3 <- estimateSizeFactors(pG3)
> pG3 <- estimateDispersions(pG3)                          # dispersion based on 2x2 table df=3
```

```
> fit1 <- fitNbinomGLMs(pG3,count~type+condition)
.........
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: algorithm did not converge
3: glm.fit: algorithm did not converge
4: glm.fit: algorithm did not converge
5: glm.fit: algorithm did not converge
6: glm.fit: algorithm did not converge
7: glm.fit: algorithm did not converge
8: glm.fit: algorithm did not converge

> fit0 <- fitNbinomGLMs(pG3,count~type)
.........

> str(fit1)
'data.frame':   9874 obs. of  5 variables:
 $ (Intercept)       : num  5.754 -0.103 11.037 7.608 1.795 ...
 $ typesingle-read   : num  -0.099 -1.43694 0.00675 0.30721 0.84524 ...
 $ conditionuntreated: num  -0.0117 0.6724 0.2746 0.0691 -1.0655 ...
 $ deviance          : num  2.86 1.99 2.6 1.24 3.55 ...
 $ converged         : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 - attr(*, "df.residual")= num 4

> pvalsGLM <- nbinomGLMTest(fit1,fit0)
> padjGLM <- p.adjust(pvalsGLM,"BH")
```

```
> tab1 <- table("treat only" = results1$padj < .1,"treat in 2x2" = padjGLM < .1)
> addmargins(tab1)


          treat in 2x2
treat only FALSE TRUE  Sum
     FALSE  9107  304 9411
     TRUE     12  451  463
     Sum    9119  755 9874

> head(fit1)
            (Intercept) typesingle-read conditionuntreated deviance converged
FBgn0000008   5.7538539    -0.098997616        -0.01170229 2.857311      TRUE
FBgn0000015  -0.1029054    -1.436940767         0.67235046 1.992255      TRUE
FBgn0000017  11.0373447     0.006749605         0.27460192 2.599667      TRUE
FBgn0000018   7.6083767     0.307211262         0.06912117 1.242985      TRUE
FBgn0000024   1.7952847     0.845239047        -1.06548420 3.545482      TRUE
FBgn0000032   9.2197012     0.060957307         0.06940660 2.219547      TRUE
```

Exercise

- Try the analysis with pasillaExons
- Can you analyze the chicken proteomics data with DESeq?
- If so, are the results any different?