R 语言 DESeq 包介绍

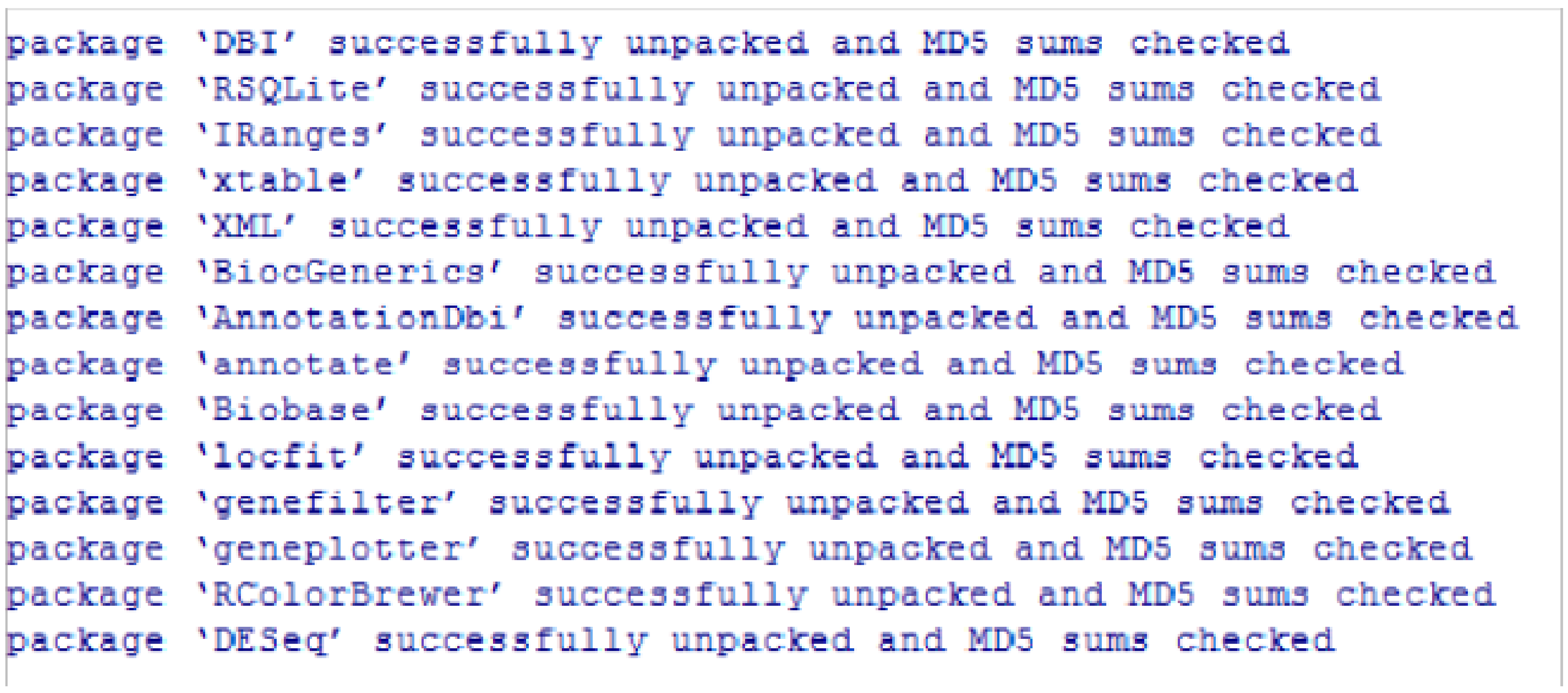
分析 RNA 序列数据的一个主要任务是探测基因的差异表达， DESeq 包提供了测试差异表达的方法，应用负二项分布和收缩的分布方程估计。

1. 包的安装

输入如下命令， DESeq和相关的包就可以自动下载和安装。

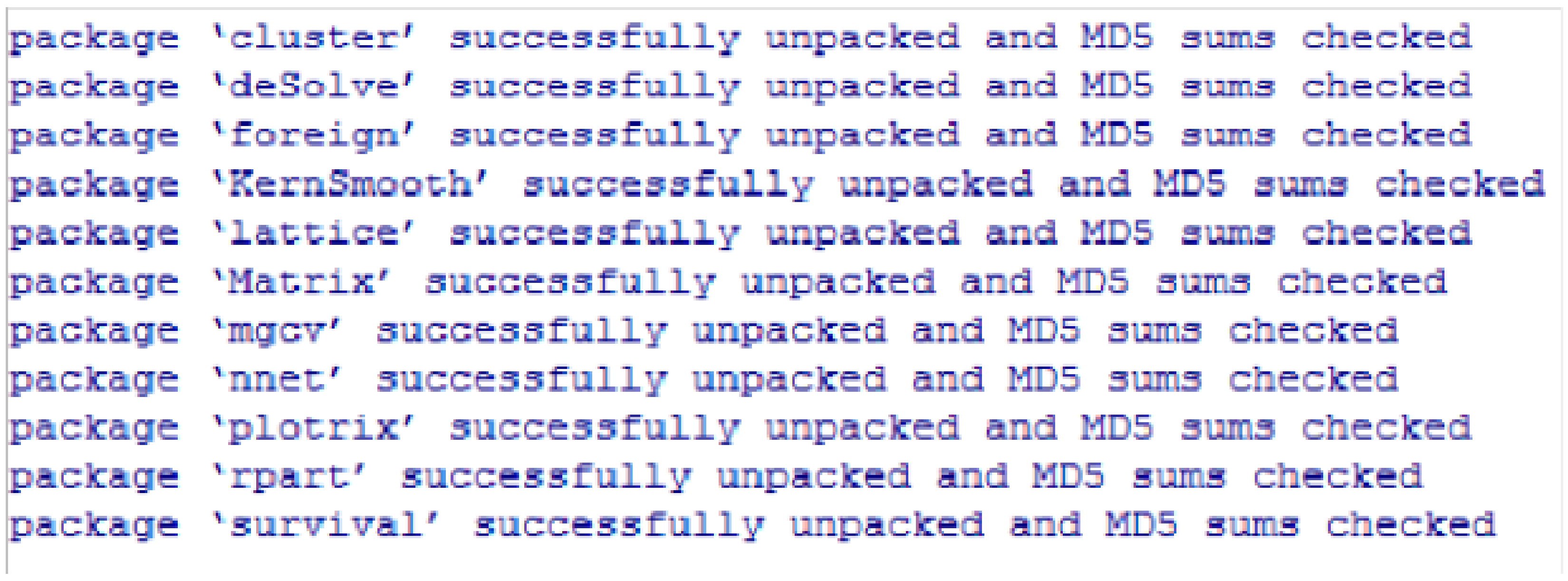
```
> source("http://bioconductor.org/biocLite.R")
> biocLite("DESeq")
```

相关的包会自动下载安装，安装的包如下：



```
package 'DBI' successfully unpacked and MD5 sums checked
package 'RSQLite' successfully unpacked and MD5 sums checked
package 'IRanges' successfully unpacked and MD5 sums checked
package 'xtable' successfully unpacked and MD5 sums checked
package 'XML' successfully unpacked and MD5 sums checked
package 'BiocGenerics' successfully unpacked and MD5 sums checked
package 'AnnotationDbi' successfully unpacked and MD5 sums checked
package 'annotate' successfully unpacked and MD5 sums checked
package 'Biobase' successfully unpacked and MD5 sums checked
package 'locfit' successfully unpacked and MD5 sums checked
package 'genefilter' successfully unpacked and MD5 sums checked
package 'geneplotter' successfully unpacked and MD5 sums checked
package 'RColorBrewer' successfully unpacked and MD5 sums checked
package 'DESeq' successfully unpacked and MD5 sums checked
```

中间会有个选择需要更新相关的包，选择更新全部，更新的包有：



```
package 'cluster' successfully unpacked and MD5 sums checked
package 'deSolve' successfully unpacked and MD5 sums checked
package 'foreign' successfully unpacked and MD5 sums checked
package 'KernSmooth' successfully unpacked and MD5 sums checked
package 'lattice' successfully unpacked and MD5 sums checked
package 'Matrix' successfully unpacked and MD5 sums checked
package 'mgcv' successfully unpacked and MD5 sums checked
package 'nnet' successfully unpacked and MD5 sums checked
package 'plotrix' successfully unpacked and MD5 sums checked
package 'rpart' successfully unpacked and MD5 sums checked
package 'survival' successfully unpacked and MD5 sums checked
```

另外还要安装一个数据包，供下面介绍包中的方法时使用，包名为 pasilla.

2. 输入数据和准备

2.1 计数表

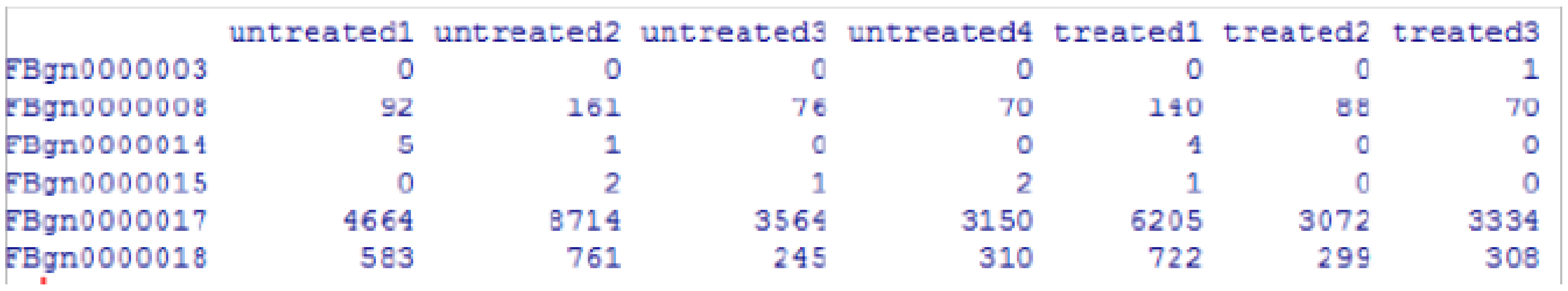
数据表的第 i 行第 j 列元素表示第 j 个样本的第 i 个基因有多少个 reads。本文使用的数据来

自于 pasilla 数据包，函数 system.file 告诉我们数据文件保存的路径。

```
> datafile = system.file( "extdata/pasilla_gene_counts.tsv", package="pasilla" )
> datafile
[1] "D:/Program Files/R/R-2.15.3/library/pasilla/extdata/pasilla_gene_counts.tsv"
```

在 R 中读取这个文件，使用 read.table 函数。

```
> pasillaCountTable = read.table( datafile, header=TRUE, row.names=1 )
> head( pasillaCountTable )
```



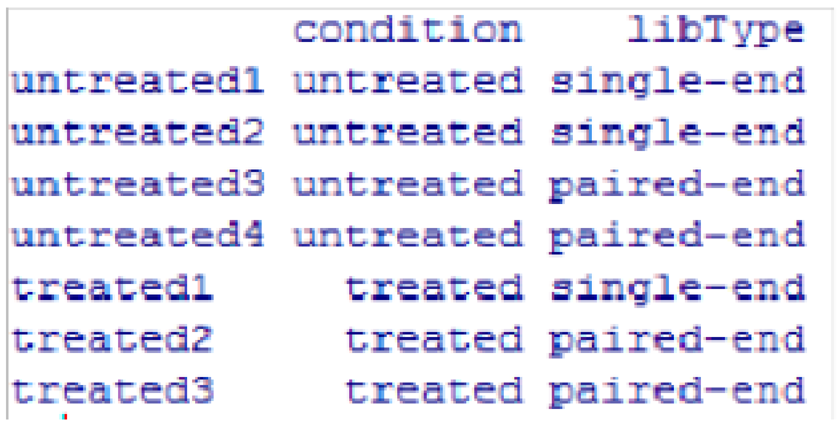
```
            untreated1 untreated2 untreated3 untreated4 treated1 treated2 treated3
FBgn0000003          0          0          0          0        0        0        1
FBgn0000008         92        161         76         70      140       88       70
FBgn0000014          5          1          0          0        4        0        0
FBgn0000015          0          2          1          2        1        0        0
FBgn0000017       4664       8714       3564       3150     6205     3072     3334
FBgn0000018        583        761        245        310      722      299      308
```

2.2 元数据

没有元数据的数据是没有用的，元数据可以分为三组，分别是样本 (行)，特征 (列)和整个实验的信息。

首先需要样本的描述信息， data.frame 的列表示各种信息，行表示 7 个样本。

```
> pasillaDesign = data.frame(
+ row.names = colnames( pasillaCountTable ),
+ condition = c( "untreated", "untreated", "untreated",
+ "untreated", "treated", "treated", "treated" ),
+ libType = c( "single-end", "single-end", "paired-end",
+ "paired-end", "single-end", "paired-end", "paired-end" ) )
> pasillaDesign
```



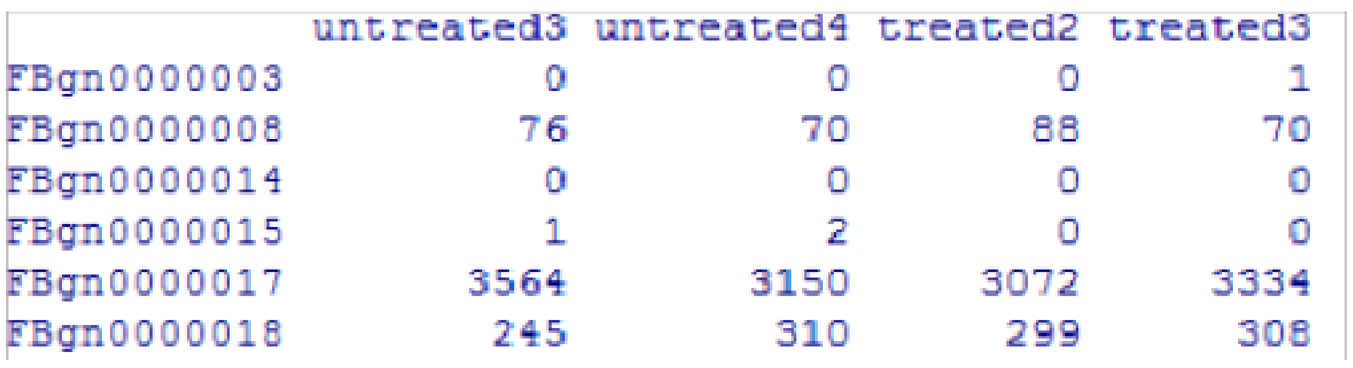
```
            condition    libType
untreated1  untreated single-end
untreated2  untreated single-end
untreated3  untreated paired-end
untreated4  untreated paired-end
treated1      treated single-end
treated2      treated paired-end
treated3      treated paired-end
```

这边简单地使用 R 代码进行设定，通常情况是从扩展表中读取这些数据。 为了分析这些样本，我们需要解释 single-end 和 paired-end 的方法，这边首先简单的分析 paired-end 样本。

```
> pairedSamples = pasillaDesign$libType == "paired-end"
> countTable = pasillaCountTable[ , pairedSamples ]
> condition = pasillaDesign$condition[ pairedSamples ]
```

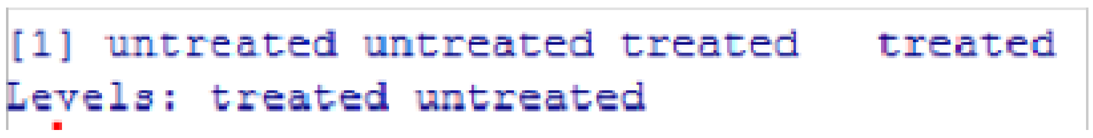
现在，我们有下面的数据输入

```
> head(countTable)
```



```
            untreated3 untreated4 treated2 treated3
FBgn0000003          0          0        0        1
FBgn0000008         76         70       88       70
FBgn0000014          0          0        0        0
FBgn0000015          1          2        0        0
FBgn0000017       3564       3150     3072     3334
FBgn0000018        245        310      299      308
```

> condition



```
[1] untreated untreated treated   treated
Levels: treated untreated
```

对于自己的数据，可以简单的创建因子。

> #not run

> condition = factor(c("untreated", "untreated", "treated", "treated"))

我们现在举例　CountDataSet，DESeq包的核心数据结构

> library("DESeq")

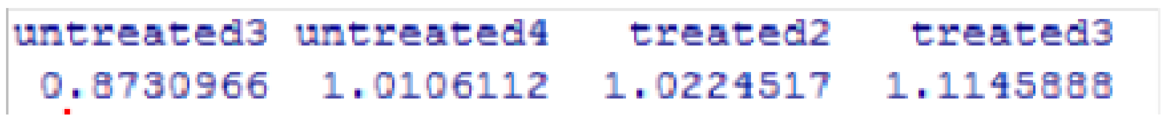
> cds = newCountDataSet(countTable, condition)

2.3 规范化

函数 estimateSizeFactors 估计统计数据的大小因子

> cds = estimateSizeFactors(cds)

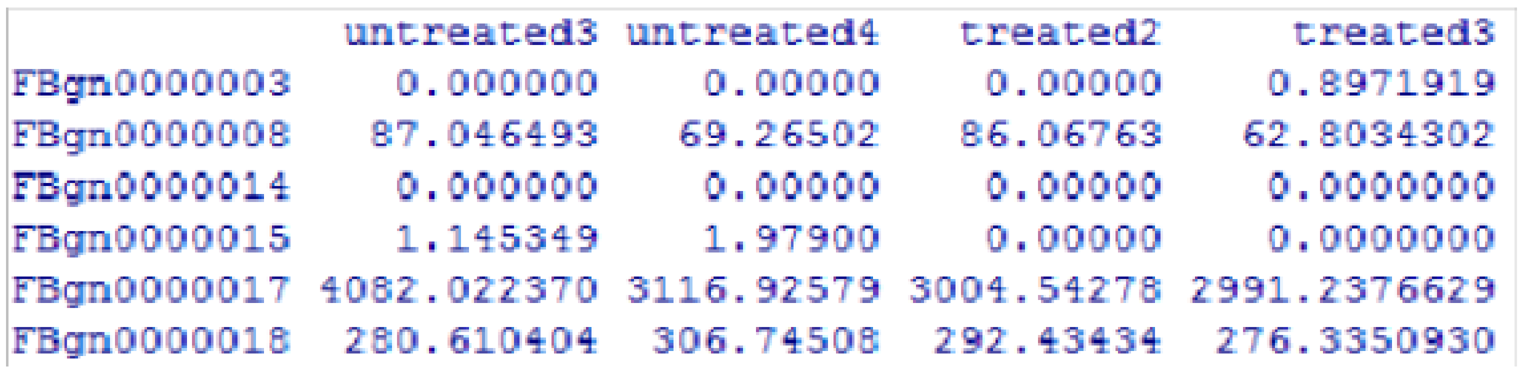
> sizeFactors(cds)



```
untreated3 untreated4   treated2   treated3
 0.8730966  1.0106112  1.0224517  1.1145888
```

如果统计数据的每列除以这列的大小因子，　　这样统计值就变成同一规模，　使它们具有可比性。函数 counts 可以做这个计算。

> head(counts(cds, normalized=TRUE))



```
              untreated3 untreated4   treated2     treated3
FBgn0000003    0.000000    0.00000    0.00000    0.8971919
FBgn0000008   87.046493   69.26502   86.06763   62.8034302
FBgn0000014    0.000000    0.00000    0.00000    0.0000000
FBgn0000015    1.145349    1.97900    0.00000    0.0000000
FBgn0000017 4082.022370 3116.92579 3004.54278 2991.2376629
FBgn0000018  280.610404  306.74508  292.43434  276.3350930
```

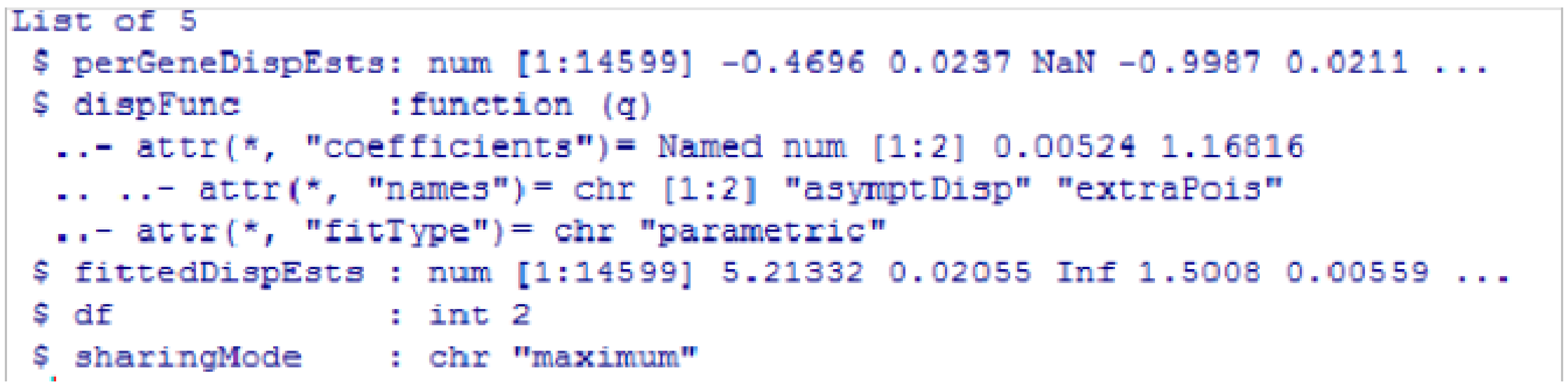
3. 方差估计

DESeq推断依靠估计典型的数据间的方差和平均关系，　　或者等效的离差和均值。　离差可以理解为生物变异系数的平方。估计离差可以使用以下命令。

> cds = estimateDispersions(cds)

函数 estimateDispersions 做了三步，首先估计每条基因的离差，然后，通过估计匹配一条曲线，最后，　每个基因分配一个离差，　从每条基因估计值和匹配值选一个。为了让用户知道中间过程，　fitInfo 对象被储存下来。

> str(fitInfo(cds))



```
List of 5
 $ perGeneDispEsts: num [1:14599] -0.4696 0.0237 NaN -0.9987 0.0211 ...
 $ dispFunc       :function (q)
  ..- attr(*, "coefficients")= Named num [1:2] 0.00524 1.16816
  .. ..- attr(*, "names")= chr [1:2] "asymptDisp" "extraPois"
  ..- attr(*, "fitType")= chr "parametric"
 $ fittedDispEsts : num [1:14599] 5.21332 0.02055 Inf 1.5008 0.00559 ...
 $ df             : int 2
 $ sharingMode    : chr "maximum"
```

函数 plotDispEsts 可以画出每条基因的估计值和平均正常统计值的关系。

> plotDispEsts(cds)

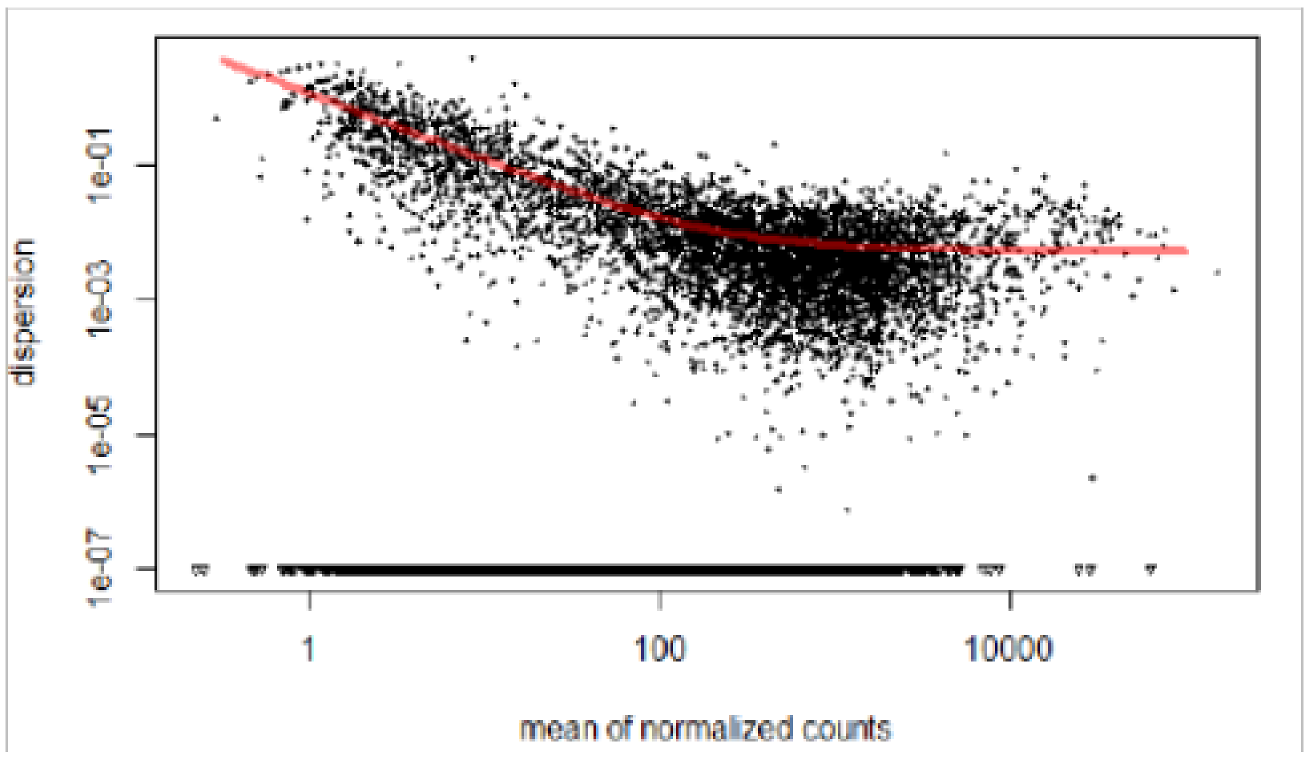
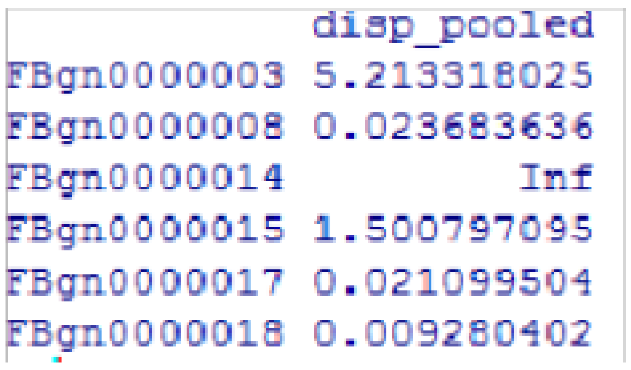


图 1 经验的 (黑点)和匹配的 (红线)离差值与平均正常统计值的关系图

在任何情况下，可以被子序列测试使用的离差值被存储在 cds 的特征数据集里。

> head(fData(cds))



```
            disp_pooled
FBgn0000003 5.213318025
FBgn0000008 0.023683636
FBgn0000014         Inf
FBgn0000015 1.500797095
FBgn0000017 0.021099504
FBgn0000018 0.009280402
```

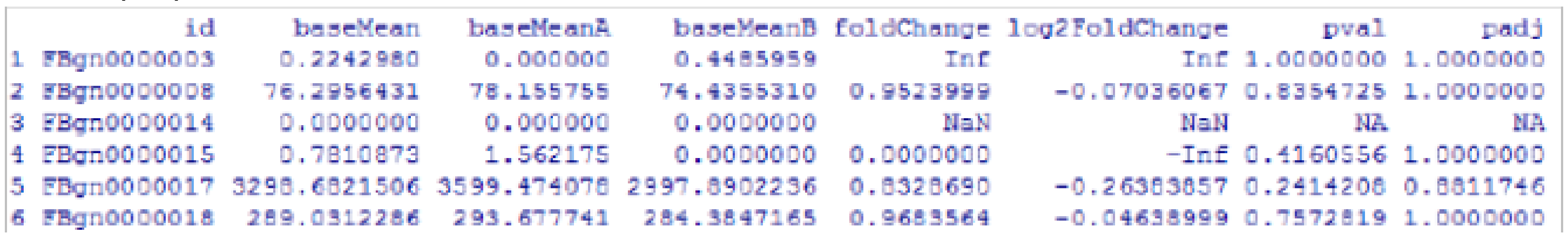
4. 推断：称之为差异表达

4.1 两个实验条件下的标准比较

为了看在条件 ”untreated"和 ”treated" 下是否有差异表达，我们简单的使用 nbinomTest 函数，

> res = nbinomTest(cds, "untreated", "treated")

> head(res)



```
           id     baseMean   baseMeanA    baseMeanB foldChange log2FoldChange      pval      padj
1 FBgn0000003    0.2242980    0.000000    0.4485959        Inf            Inf 1.0000000 1.0000000
2 FBgn0000008   76.2956431   78.155755   74.4355310  0.9523999    -0.07036067 0.8354725 1.0000000
3 FBgn0000014    0.0000000    0.000000    0.0000000        NaN            NaN        NA        NA
4 FBgn0000015    0.7810873    1.562175    0.0000000  0.0000000           -Inf 0.4160556 1.0000000
5 FBgn0000017 3298.6821506 3599.474078 2997.8902236  0.8328690    -0.26383857 0.2414208 0.8811746
6 FBgn0000018  289.0312286  293.677741  284.3847165  0.9683564    -0.04638999 0.7572819 1.0000000
```

各列表示的是：

id	feature identier
baseMean	mean normalised counts, averaged over all samples from both conditions
baseMeanA	mean normalised counts from condition A
baseMeanB	mean normalised counts from condition B
foldChange	fold change from condition A to B
log2FoldChange	the logarithm (to basis 2) of the fold change
pval	pvalue for the statistical signicance of this change
padj	pvalue adjusted for multiple testing with the Benjamini-Hochberg procedure (see the R functionp.adjust), which controls false discovery rate (FDR)

我们首先画 log2 折叠变换和平均正常统计量的关系，红色的点表示在 10%FDR的基因。

```
> plotMA(res)
```

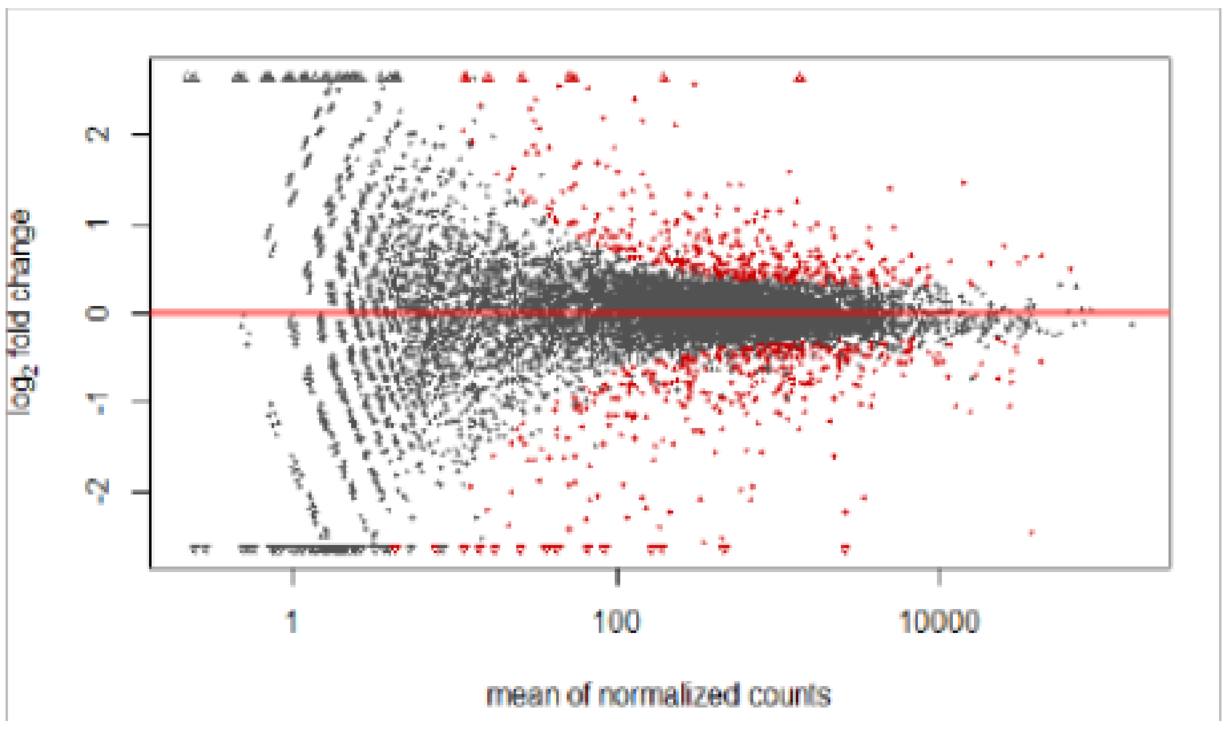


图 2 log2 折叠变换和平均正常统计量的关系图

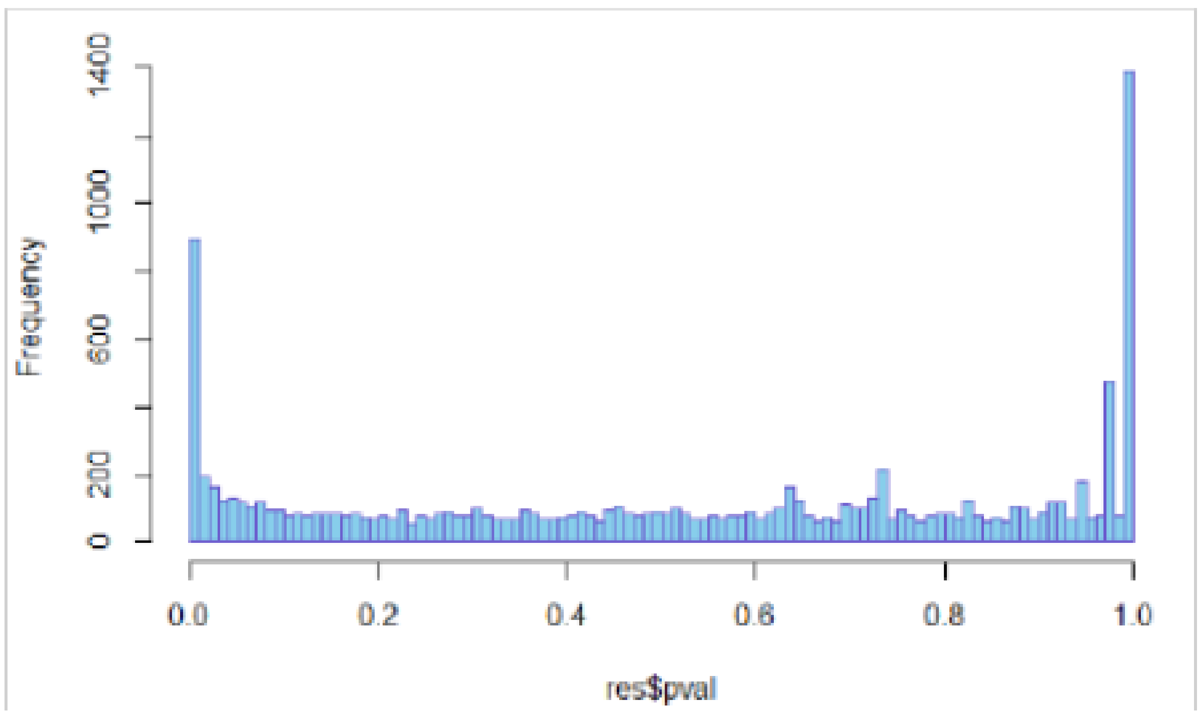


图 3 从 nbinomTest 的 p 值统计直方图

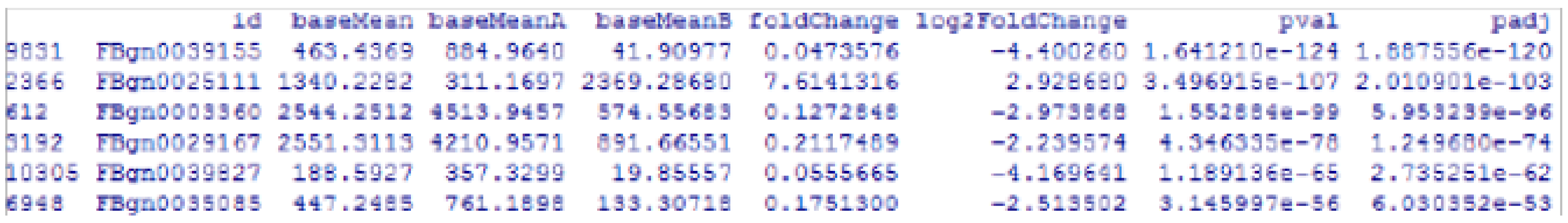
```
> hist(res$pval, breaks=100, col="skyblue", border="slateblue", main="")
```

我们可以通过 FDR 过滤有效的基因。

```
> resSig = res[ res$padj < 0.1, ]
```

列举最有效的差异表达基因

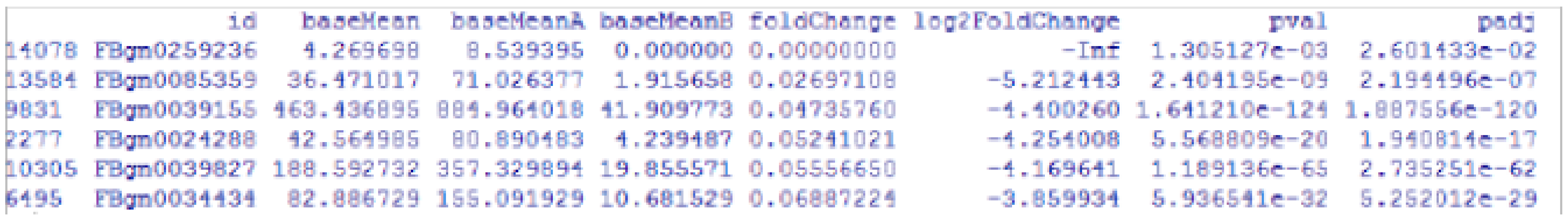
```
> head( resSig[ order(resSig$pval), ] )
```



```
                 id  baseMean baseMeanA  baseMeanB foldChange log2FoldChange          pval          padj
9831  FBgn0039155  463.4369  884.9640   41.90977  0.0473576      -4.400260 1.641210e-124 1.887556e-120
2366  FBgn0025111 1340.2282  311.1697 2369.28680  7.6141316       2.928680 3.496915e-107 2.010901e-103
612   FBgn0003360 2544.2512 4513.9457  574.55683  0.1272848      -2.973868  1.552884e-99  5.953239e-96
3192  FBgn0029167 2551.3113 4210.9571  891.66551  0.2117489      -2.239574  4.346335e-78  1.249680e-74
10305 FBgn0039827  188.5927  357.3299   19.85557  0.0555665      -4.169641  1.189136e-65  2.735251e-62
6948  FBgn0035085  447.2485  761.1898  133.30718  0.1751300      -2.513502  3.145997e-56  6.030352e-53
```

我们也想看到有效基因的最强下调。

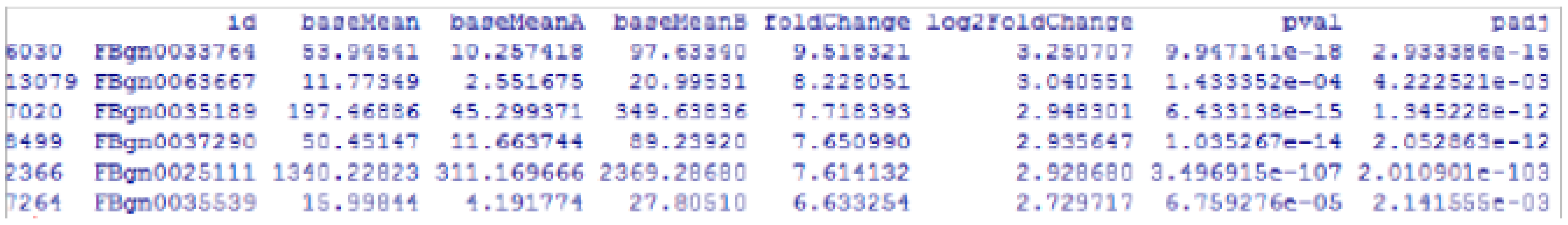
```
> head( resSig[ order( resSig$foldChange, -resSig$baseMean ), ] )
```



```
                 id   baseMean  baseMeanA baseMeanB foldChange log2FoldChange          pval          padj
14078 FBgn0259236   4.269698   8.539395  0.000000 0.00000000           -Inf  1.305127e-03  2.601433e-02
13584 FBgn0085359  36.471017  71.026377  1.915658 0.02697108      -5.212443  2.404195e-09  2.194496e-07
9831  FBgn0039155 463.436895 884.964018 41.909773 0.04735760      -4.400260 1.641210e-124 1.887556e-120
2277  FBgn0024288  42.564985  80.890483  4.239487 0.05241021      -4.254008  5.568809e-20  1.940814e-17
10305 FBgn0039827 188.592732 357.329894 19.855571 0.05556650      -4.169641  1.189136e-65  2.735251e-62
6495  FBgn0034434  82.886729 155.091929 10.681529 0.06887224      -3.859934  5.936541e-32  5.252012e-29
```

或者是最强上调

```
> head( resSig[ order( -resSig$foldChange, -resSig$baseMean ), ] )
```



```
                 id   baseMean  baseMeanA  baseMeanB foldChange log2FoldChange          pval          padj
6030  FBgn0033764   53.94541  10.257418   97.63390   9.518321       3.250707  9.947141e-18  2.933386e-15
13079 FBgn0063667   11.77349   2.551675   20.99531   8.228051       3.040551  1.433352e-04  4.222521e-03
7020  FBgn0035189  197.46886  45.299371  349.63836   7.718393       2.948301  6.433138e-15  1.345228e-12
8499  FBgn0037290   50.45147  11.663744   89.23920   7.650990       2.935647  1.035267e-14  2.052863e-12
2366  FBgn0025111 1340.22823 311.169666 2369.28680   7.614132       2.928680 3.496915e-107 2.010901e-103
7264  FBgn0035539   15.99844   4.191774   27.80510   6.633254       2.729717  6.759276e-05  2.141555e-03
```

可以保存这些输出到指定 CSV文件，命令如下

```
> write.csv( res, file="My Pasilla Analysis Result Table.csv" )
```

对于弱表达基因，需要加更强的改变，这边以两个 untreated 基因为例。

```
> ncu = counts( cds, normalized=TRUE )[ , conditions(cds)=="untreated" ]
```

Ncu 这边是两列的矩阵

```
>plotMA(data.frame(baseMean = rowMeans(ncu),
+     log2FoldChange = log2( ncu[,2] / ncu[,1] )),
+     col = "black")
```

画出如图 4 所示的图像。

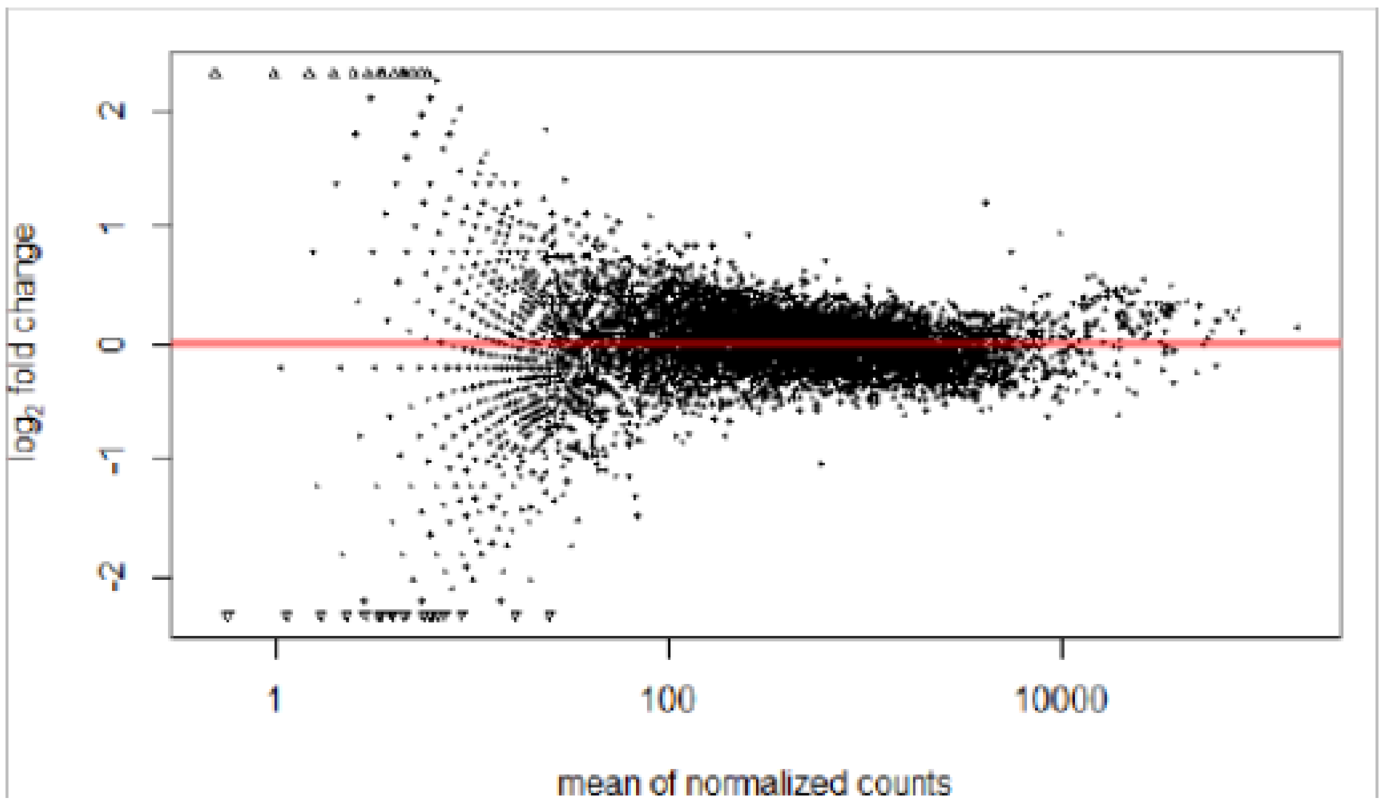


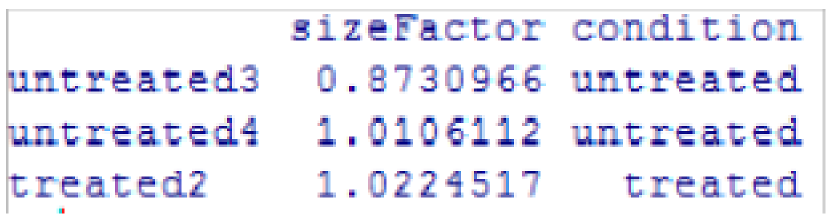
图 4 两个 untreated 复制平均表达长度的 log2 折叠变换图

4.2 部分不复制的情况

如果只有一种条件的基因复制， 而另一种条件下的不复制。 那么只有复制的条件的可以用来估计离差。为了证明，我们截取仅有三个样本的数据对象。

```
> cdsUUT = cds[ , 1:3]
```

> pData(cdsUUT)



```
           sizeFactor condition
untreated3  0.8730966 untreated
untreated4  1.0106112 untreated
treated2    1.0224517   treated
```

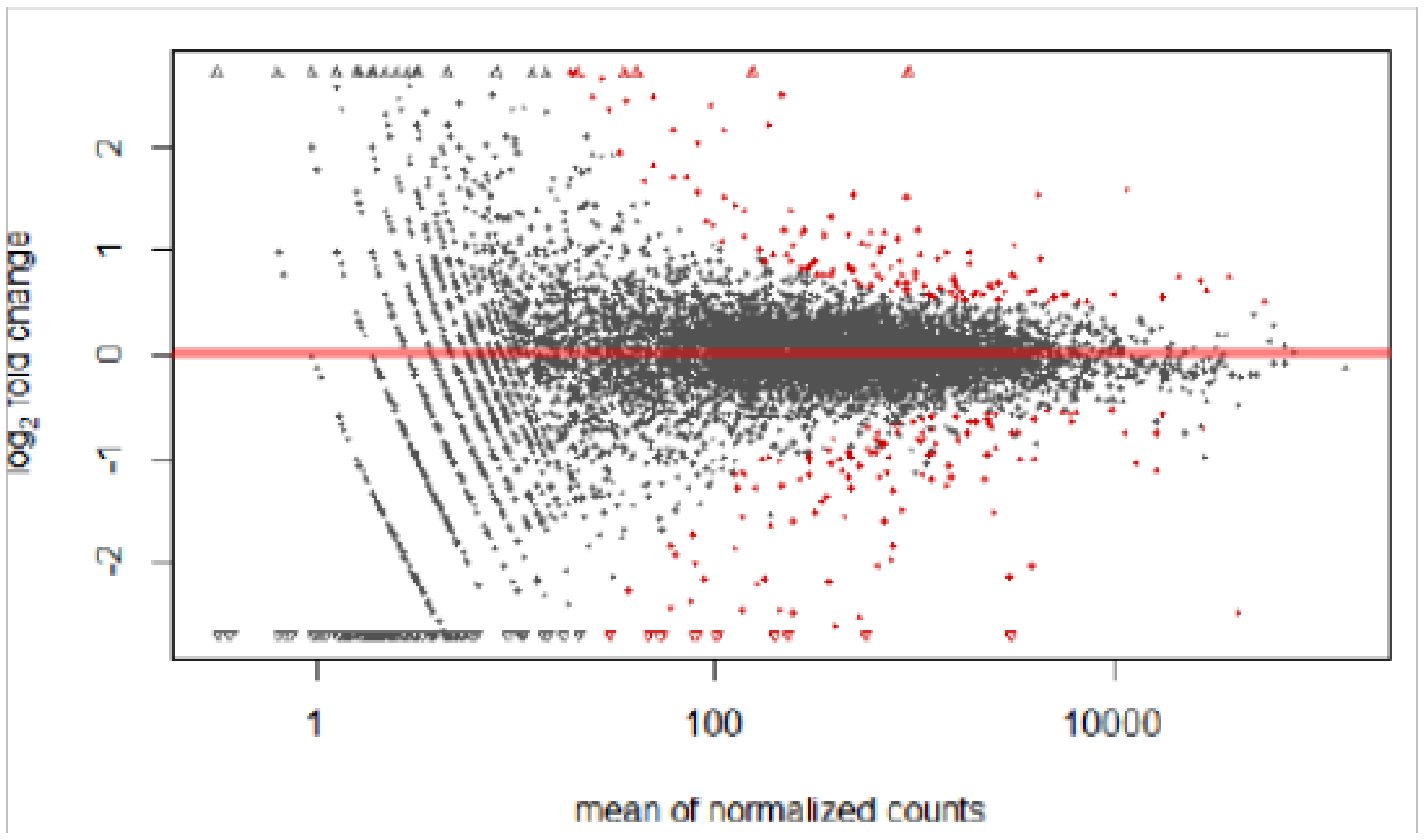


图 5 “ treated" vs. untreated" 对比的 MvA 图,用两个 treated 和一个 untreated 样本

现在我们可以像刚才那样分析了。

>cdsUUT = estimateSizeFactors(cdsUUT)

>cdsUUT = estimateDispersions(cdsUUT)

>resUUT = nbinomTest(cdsUUT, "untreated", "treated")

生成类似的图，用

>plotMA(resUUT)

4.3 没有任何复制的情况

DESeq包可以在没有复制的情况分析方法如下。

> cds2 = cds[,c("untreated3", "treated3")]

> cds2 = estimateDispersions(cds2, method="blind", sharingMode="fit-only")

> res2 = nbinomTest(cds2, "untreated", "treated")

> plotMA(res2)

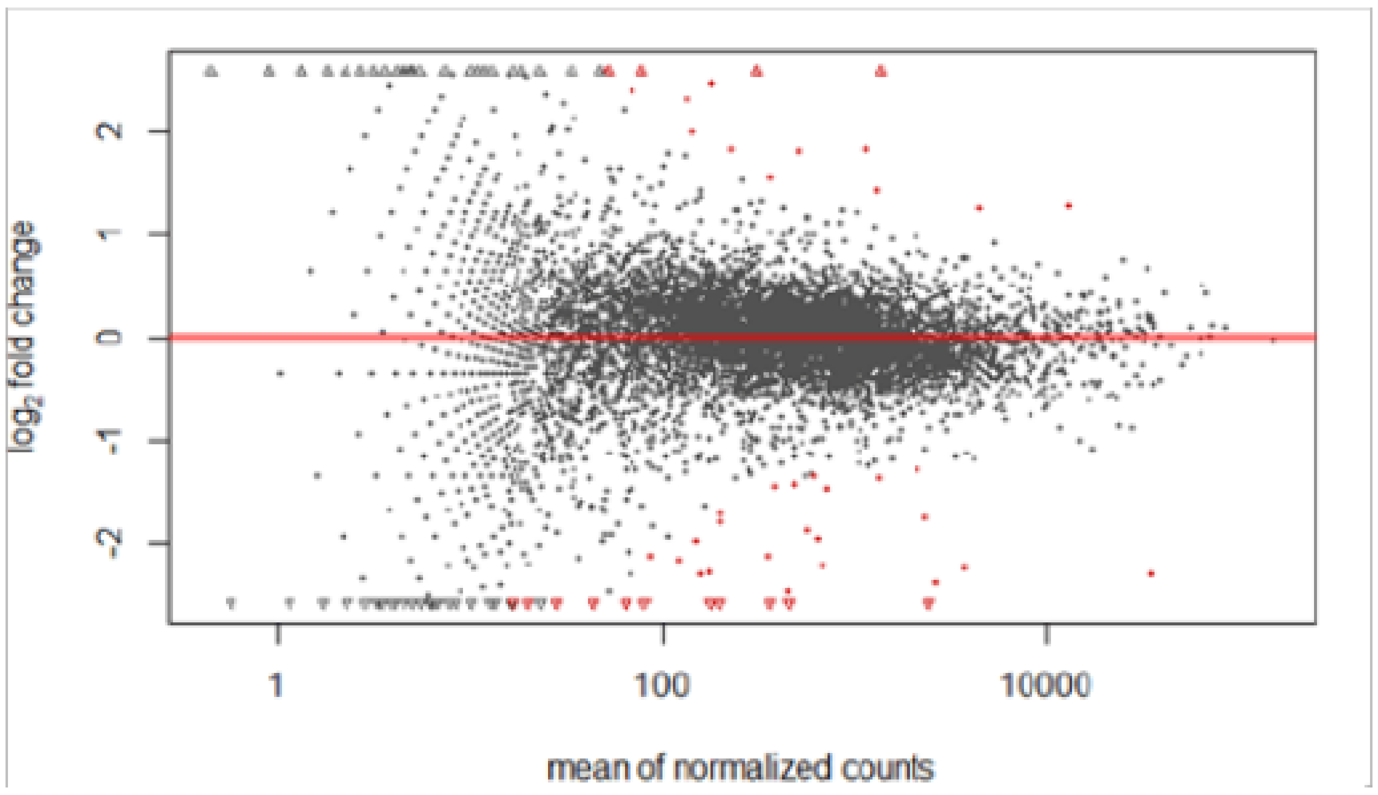
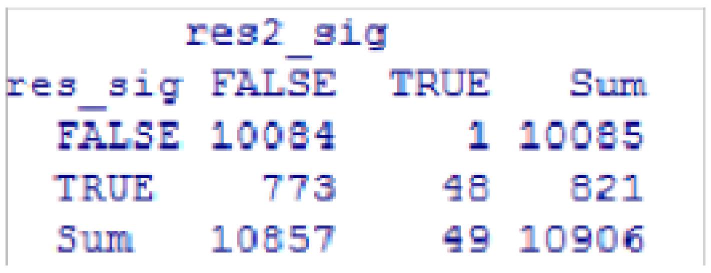


图 6 MvA 图，从没有复制的测试

> addmargins(table(res_sig = res$padj < .1, res2_sig = res2$padj < .1))



```
         res2_sig
res_sig  FALSE  TRUE   Sum
  FALSE  10084     1 10085
  TRUE     773    48   821
  Sum    10857    49 10906
```

DESeq包还有很丰富的内容，由于本人水平比较低，只能介绍这么多了。

参考文献

Simon Anders and Wolfgang Huber (2010): Differential expression analysis for sequence count data.Genome Biology 11:R106