MIS-DESeq2

Sunit Jain

January 6, 2015

Contents

Dependencies

If you're unsure that you have all the pacakges required to run this workflow. Open the `Rmd` file in your favorite text editor (I used RStudio) and change the next line from `eval=FALSE` to `eval=TRUE`. Now, when you run this workflow, the dependencies should be installed first.

Generate a read count matrix using `htseq-count`

Sample command:

```
htseq-count -f bam -r name -t CDS -o scaffold.htseq.sam -i ID -q scaffold_sortedByName.bam
all_combined.gff
```

This command was run for each sample individually.

Merging duplicate genes

I performed a self blast and looked at results that had a percent identity greater than 98%, query coverage greater than 96% and a minimum alignment length of 500 bases. Once I had this subset, I screened out the hits to exons since we won't be considering them for this experiment anyway. I was left with the following two gene pairs:

- scaffold_344578___MIS_1109813.1 scaffold_133898___MIS_10093600.14
- scaffold_219988___MIS_10179608.12 scaffold_555373___MIS_1172265.1

that had high enough similarity based on the thresholds mentioned above that their count data needed to be merged. The perl script `mergeCounts.pl` was run on each htseq-count output individually in order to accomplish this. Here is a sample command used for one of the htseq-count outputs:

```
perl mergeCounts.pl -l realDuplicateGenes.list -tsv Day_1.htseqCount.tsv -o Day_1.htseqCount.merged.tsv
```

where, realDuplicateGenes.list contains the two gene pairs mentioned above.

Import Counts into DESeq2

Once we were satisfied with the genes and their counts. We imported the count data into DESeq2.

Reads per Sample

```
##   Day_1   Day_2   Day_3 Night_4 Night_5 Night_6
## 2739735  492104  691689 1105737 1587917  969992
```

Filtering the data

Get rid of genes which did not occur frequently enough. Here we say, lets get rid of genes with counts >=2 in at least 2 samples.

```
##   Day_1   Day_2   Day_3 Night_4 Night_5 Night_6
## 2705284  480844  679334 1091297 1547976  941046
```

How many reads were removed when Min Raw Count = 2?

```
##   Day_1   Day_2   Day_3 Night_4 Night_5 Night_6
##   34451   11260   12355   14440   39941   28946
```

This reduces the dataset from 1464832 tags to about 12089. For the filtered tags, there is very little power to detect differential expression, so little information is lost by filtering.

Exploring the Dataset

The `rlog` transformation

Many common statistical methods for exploratory analysis of multidimensional data, especially methods for clustering and ordination (e.g., principal-component analysis and the like), work best for (at least approximately) homoskedastic data; this means that the variance of an observed quantity (here, the expression strength of a gene) does not depend on the mean. In RNA-Seq data, however, variance grows with the mean. For example, if one performs PCA (principal components analysis) directly on a matrix of normalized read counts, the result typically depends only on the few most strongly expressed genes because they show the largest absolute differences between samples. A simple and often used strategy to avoid this is to take the logarithm of the normalized count values plus a small pseudocount; however, now the genes with low counts tend to dominate the results because, due to the strong Poisson noise inherent to small count values, they show the strongest relative differences between samples.

As a solution, DESeq2 offers the regularized-logarithm transformation, or rlog for short. For genes with high counts, the rlog transformation differs not much from an ordinary log2 transformation. For genes with lower counts, however, the values are shrunken towards the genes' averages across all samples. Using an empirical Bayesian prior on inter-sample differences in the form of a ridge penalty, this is done such that the rlog-transformed data are approximately homoskedastic.

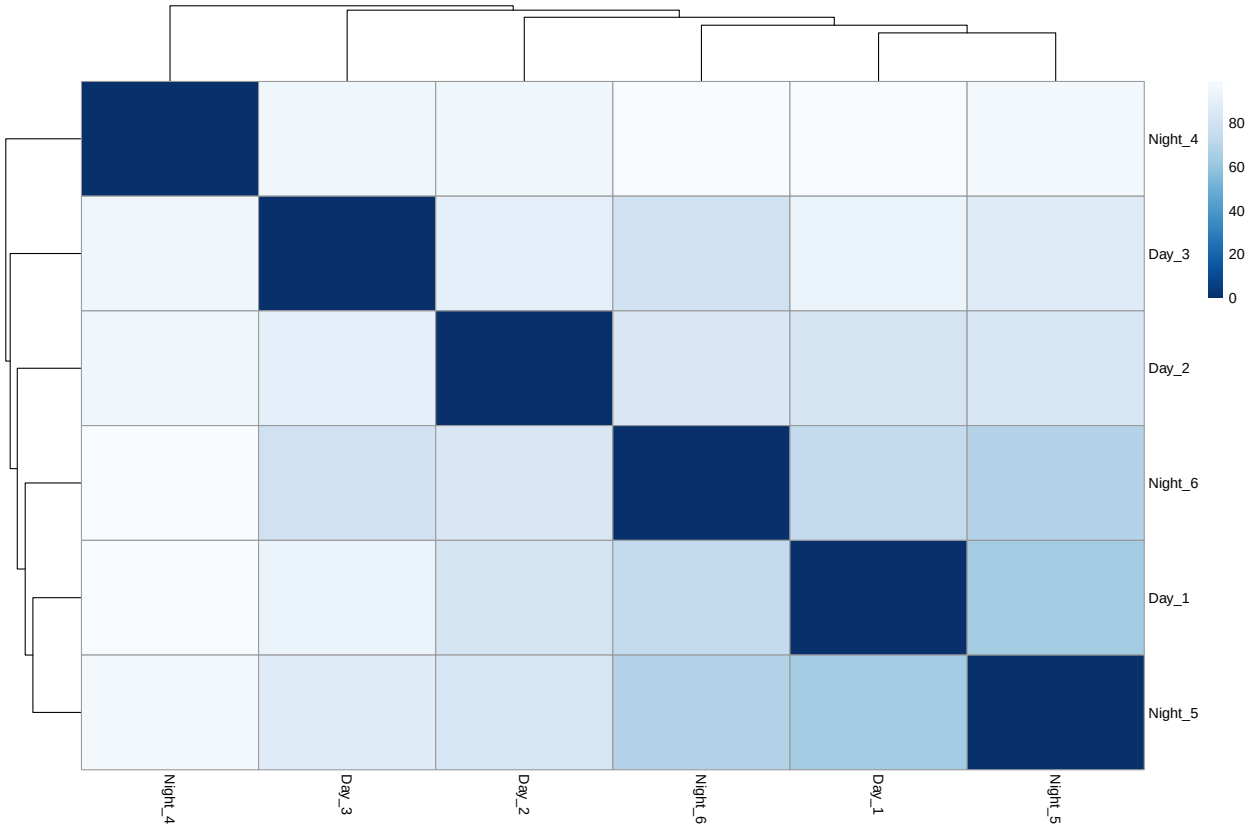
Note: the rlog transformation is provided for applications other than differential testing. For differential testing we recommend the DESeq function applied to raw counts, as described later in this workflow, which also takes into account the dependence of the variance of counts on the mean value during the dispersion estimation step.

Sample distances

A useful first step in an RNA-Seq analysis is often to assess overall similarity between samples: Which samples are similar to each other, which are different? Does this fit to the expectation from the experiment's design? We use the R function `dist` to calculate the Euclidean distance between samples. To avoid that the distance measure is dominated by a few highly variable genes, and have a roughly equal contribution from all genes, we use it on the rlog-transformed data:

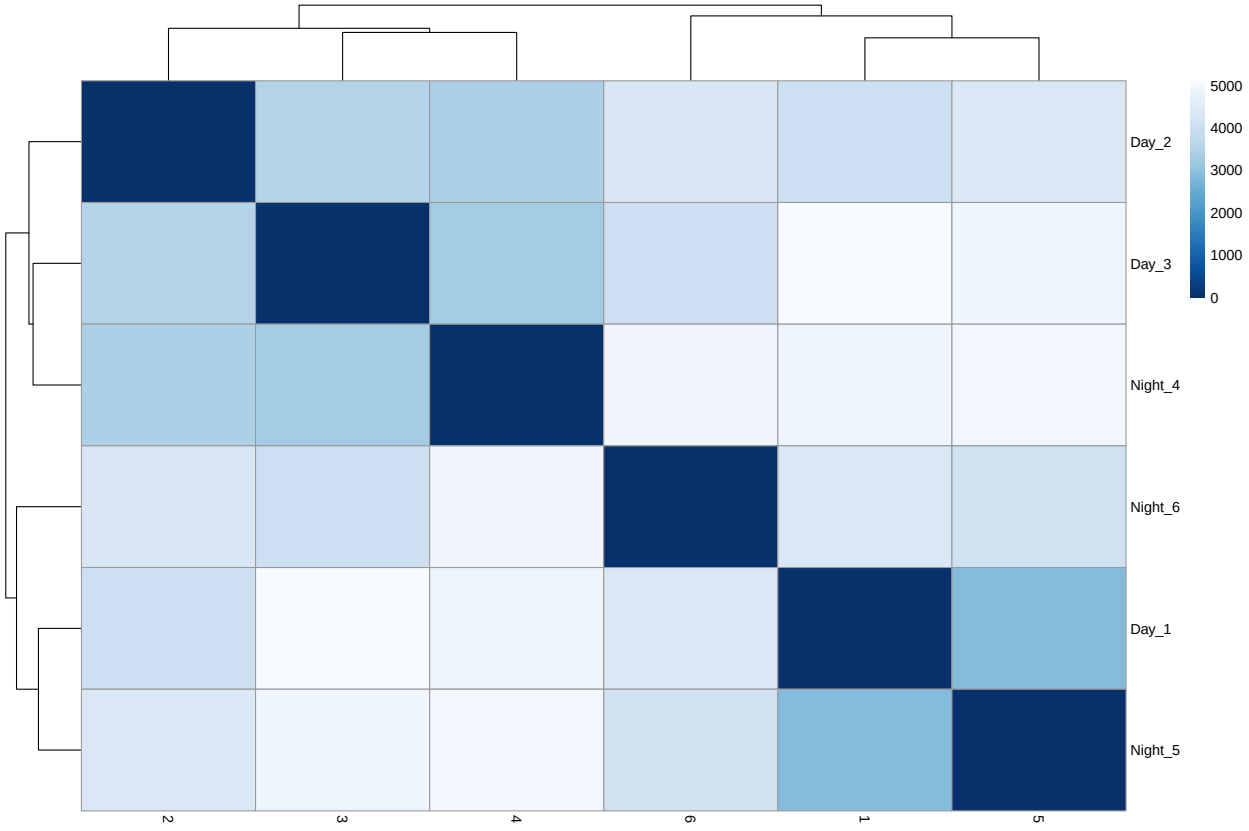
```
##            Day_1    Day_2    Day_3  Night_4  Night_5
## Day_2   81.50580
## Day_3   93.10733 89.78260
## Night_4 98.80173 95.10344 95.03750
## Night_5 63.41866 83.03560 87.19398 96.62408
## Night_6 73.46082 83.81028 79.55655 98.84027 68.63069
```

We visualize the distances in a heatmap:


Night_4
Day_3
Day_2
Night_6
Day_1
Night_5
80
60
40
20
0

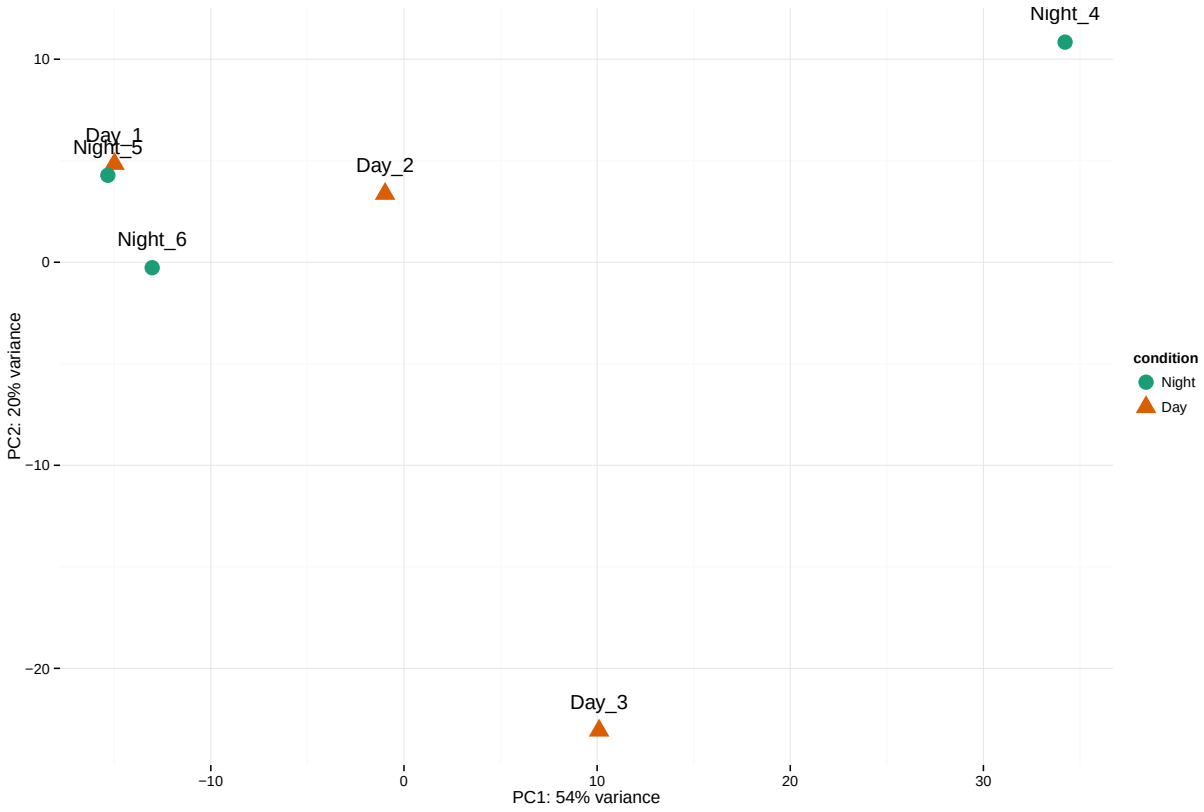
Poisson Distance

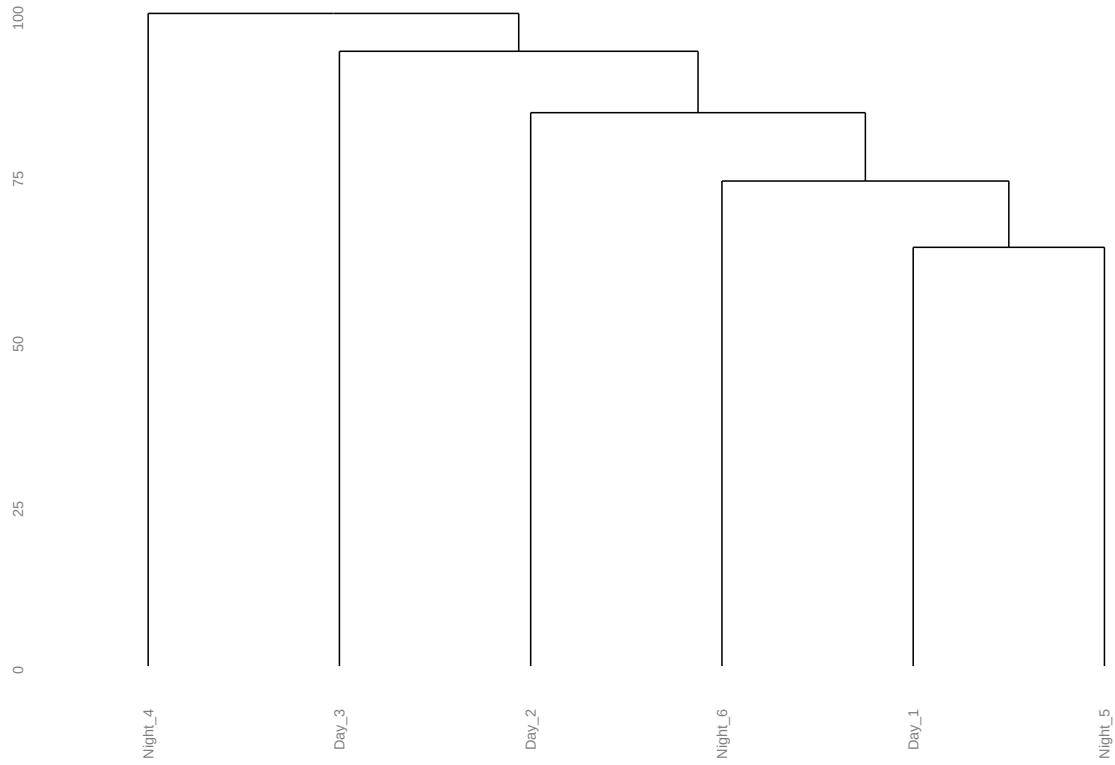
Another option for calculating sample distances is to use the Poisson Distance, implemented in the CRAN package PoiClaClu. Similar to the transformations offered in DESeq2, this measure of dissimilarity also takes the variance structure of counts into consideration when calculating the distances between samples. The PoissonDistance function takes the original count matrix (not normalized) with samples as rows instead of columns, so we need to transpose the counts in dds.



PCA plot

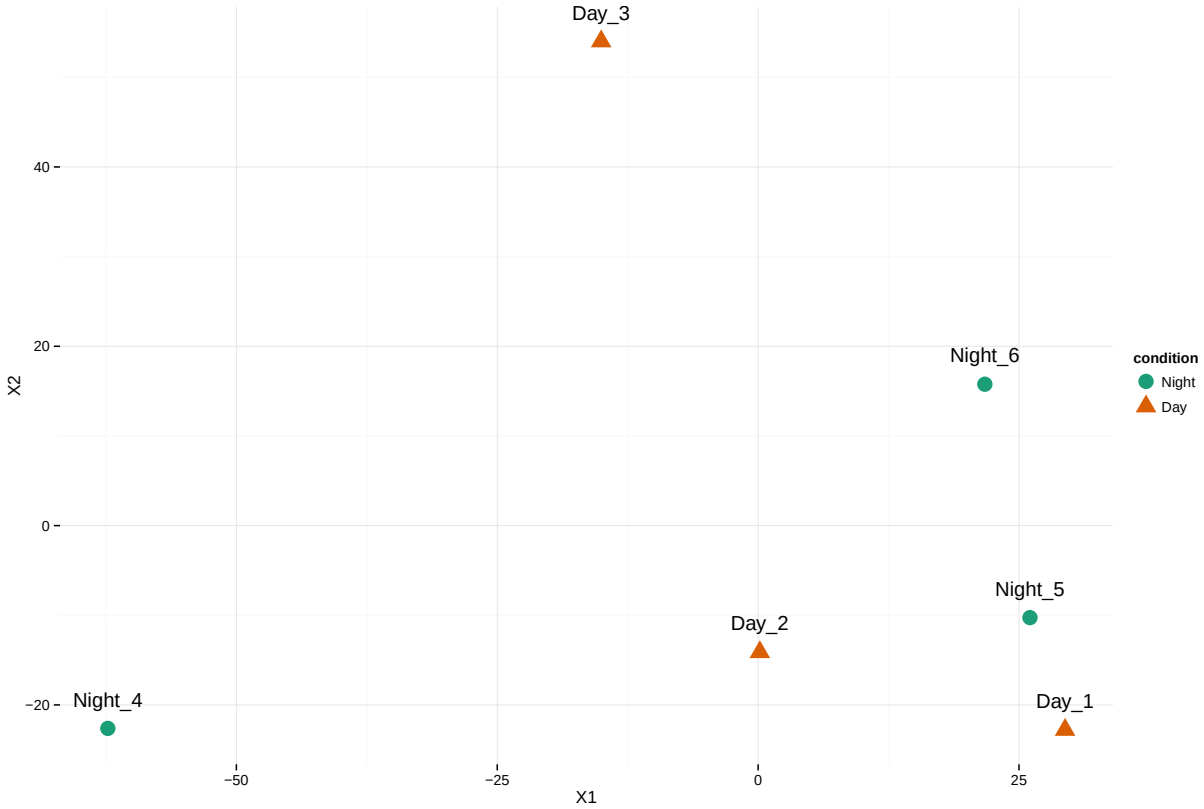
Another way to visualize sample-to-sample distances is a principal-components analysis (PCA). In this ordination method, the data points (i.e., here, the samples) are projected onto the 2D plane such that they spread out in the two directions which explain most of the differences in the data. The x-axis is the direction (or principal component) which separates the data points the most. The amount of the total variance which is contained in the direction is printed in the axis label.



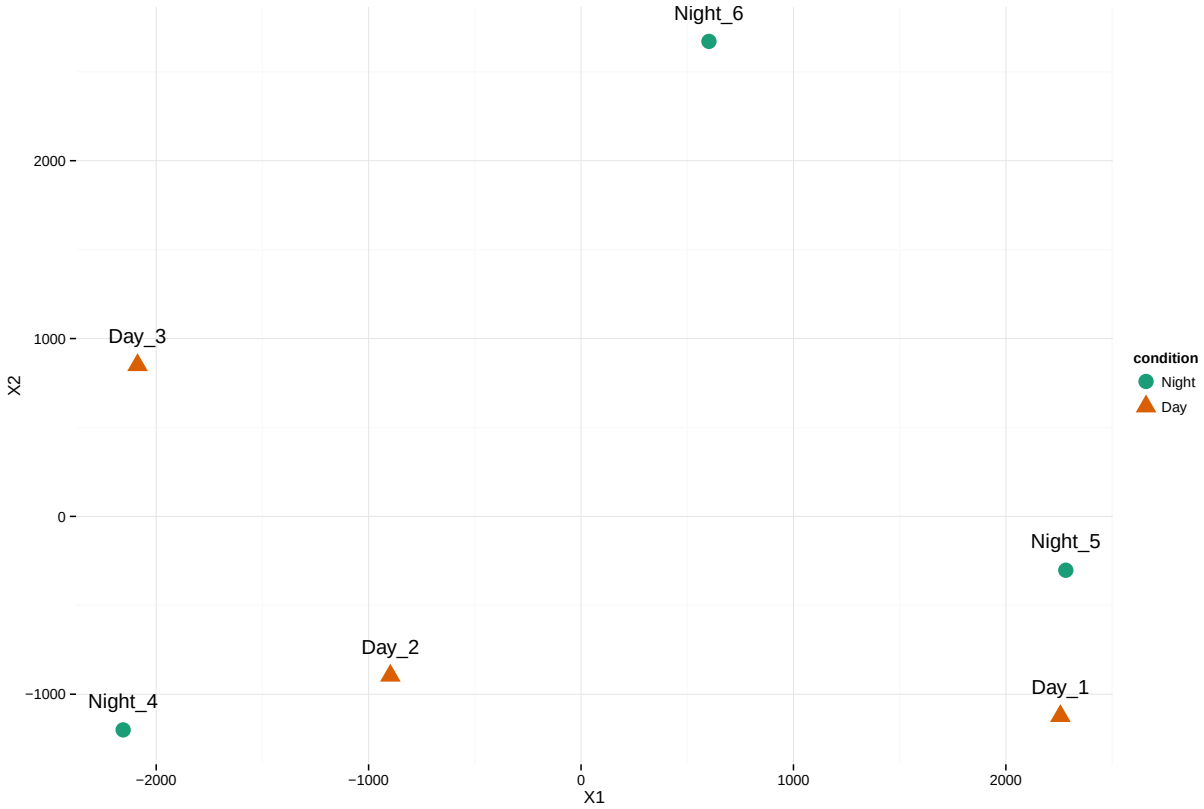

100
75
50
25
0
Night_4
Day_3
Day_2
Night_6
Day_1
Night_5

MDS plot

Another plot, very similar to the PCA plot, can be made using the multidimensional scaling (MDS) function in base R. This is useful when we don't have the original data, but only a matrix of distances. Here we have the MDS plot for the distances calculated from the rlog transformed counts:



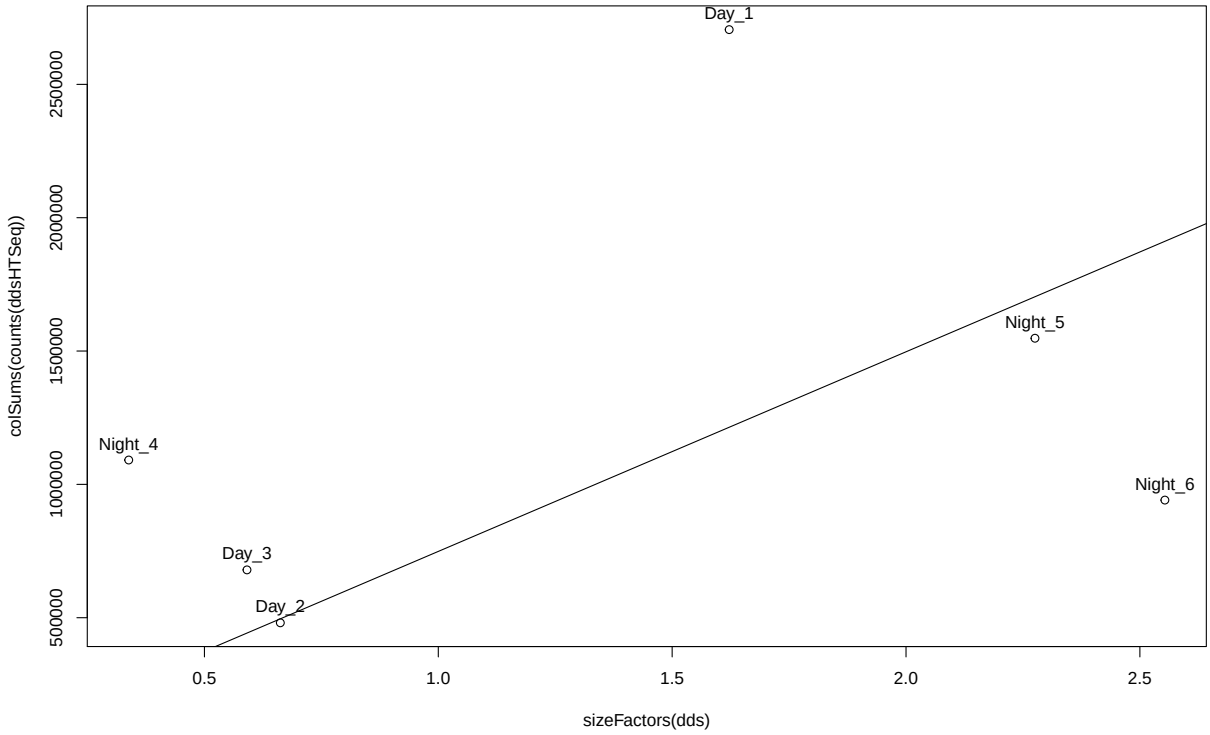
And here from the PoissonDistance:



Counts

In order to normalise the raw counts we will start by determining the relative library sizes, or size factors for each library. For example, if the counts of the expressed genes in one sample are, on average, twice as high as in another, the size factor for the first sample should be twice as large as the one for the other sample. These size factors can be obtained with the function `estimateSizeFactors`:

```
##     Day_1     Day_2     Day_3   Night_4   Night_5   Night_6
## 1.6219205 0.6622708 0.5909770 0.3381402 2.2756803 2.5534369
```




```
## null device
##           1
```

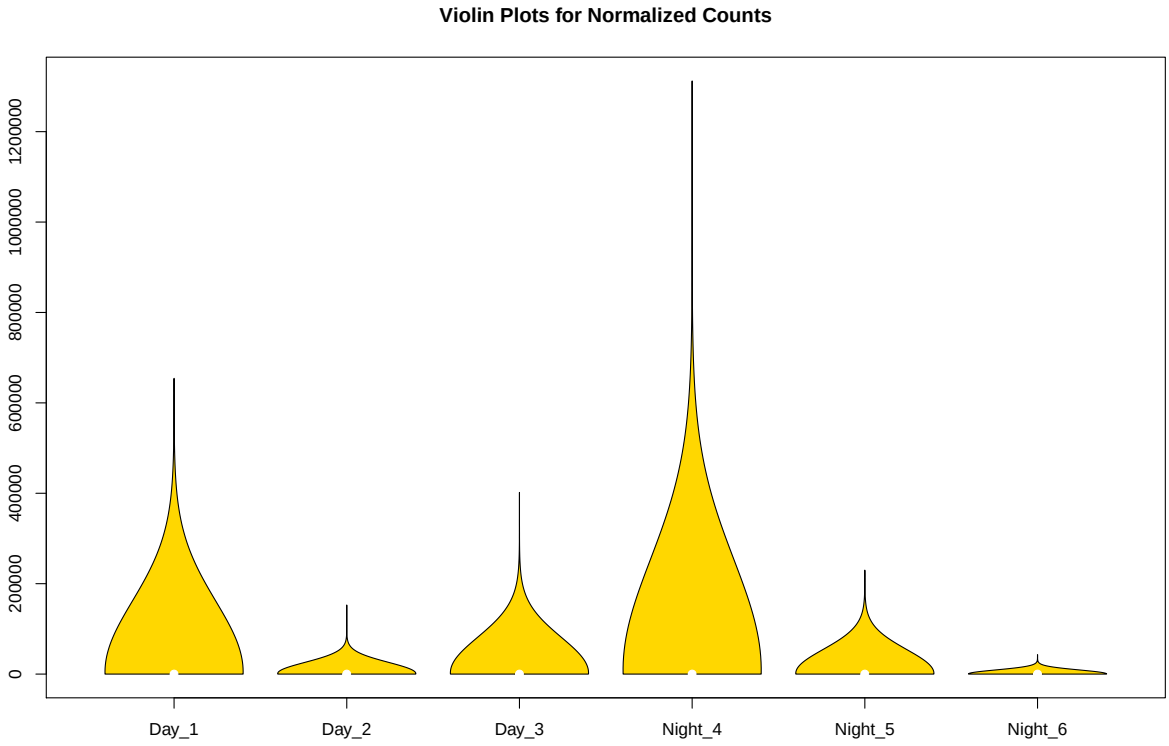
Raw vs Normalized Counts

Once we have this information, the normalised data is obtained by dividing each column of the count table by the corresponding size factor. We can perform this calculation by calling the function counts with a the `normalized` argument set as `TRUE`. Since we won't be normalizing this data, we'll set it as `FALSE`

Normalized

```
##                                 Day_1    Day_2    Day_3 Night_4   Night_5
## scaffold_0__MIS_10000001.165 3.699318 1.509956 8.460567 8.87206 8.3491517
## scaffold_0__MIS_10000001.177 1.233106 0.000000 0.000000 0.00000 3.5154323
```

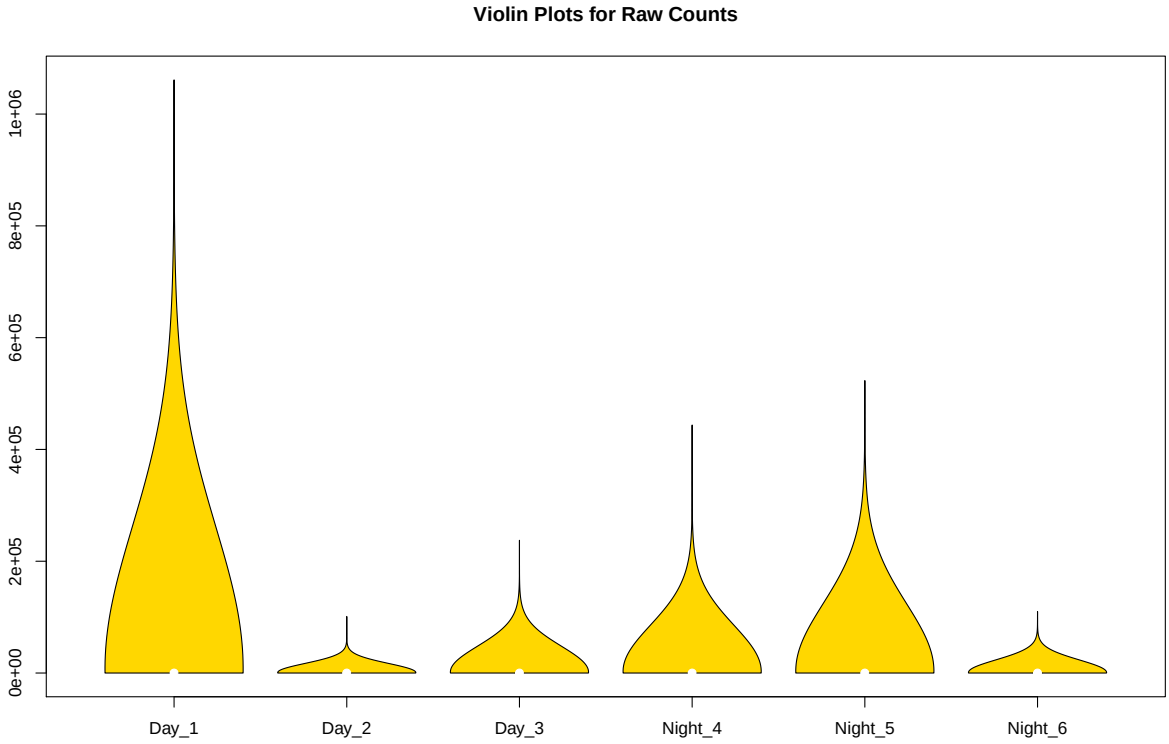
```
## scaffold_0__MIS_10000001.23  0.000000 0.000000 0.000000 0.00000 0.8788581
## scaffold_0__MIS_10000001.267 1.849659 0.000000 0.000000 0.00000 0.8788581
## scaffold_0__MIS_10000001.329 1.233106 0.000000 0.000000 0.00000 0.8788581
## scaffold_0__MIS_10000001.439 1.849659 0.000000 0.000000 0.00000 0.4394290
##                               Night_6
## scaffold_0__MIS_10000001.165 1.958145
## scaffold_0__MIS_10000001.177 0.000000
## scaffold_0__MIS_10000001.23  4.307919
## scaffold_0__MIS_10000001.267 4.307919
## scaffold_0__MIS_10000001.329 3.133032
## scaffold_0__MIS_10000001.439 1.566516
```




```
##      Day_1              Day_2              Day_3
##  Min.   :     0.0   Min.   :     0.00   Min.   :     0.0
##  1st Qu.:     0.6   1st Qu.:     0.00   1st Qu.:     0.0
##  Median :     1.8   Median :     0.00   Median :     0.0
##  Mean   :   138.0   Mean   :    60.06   Mean   :    95.1
##  3rd Qu.:     3.7   3rd Qu.:     4.53   3rd Qu.:     3.4
##  Max.   :654306.4   Max.   :152886.09   Max.   :401871.8
##     Night_4            Night_5            Night_6
##  Min.   :     0.0   Min.   :     0.00   Min.   :    0.00
##  1st Qu.:     0.0   1st Qu.:     0.44   1st Qu.:    0.39
##  Median :     0.0   Median :     1.32   Median :    1.17
##  Mean   :   267.0   Mean   :    56.27   Mean   :   30.49
##  3rd Qu.:     5.9   3rd Qu.:     3.52   3rd Qu.:    3.13
##  Max.   :1312257.5   Max.   :229977.38   Max.   :43065.88
```

Raw

```
##                                Day_1 Day_2 Day_3 Night_4 Night_5 Night_6
## scaffold_0__MIS_10000001.165     6     1     5       3      19       5
## scaffold_0__MIS_10000001.177     2     0     0       0       8       0
## scaffold_0__MIS_10000001.23      0     0     0       0       2      11
## scaffold_0__MIS_10000001.267     3     0     0       0       2      11
## scaffold_0__MIS_10000001.329     2     0     0       0       2       8
## scaffold_0__MIS_10000001.439     3     0     0       0       1       4
```

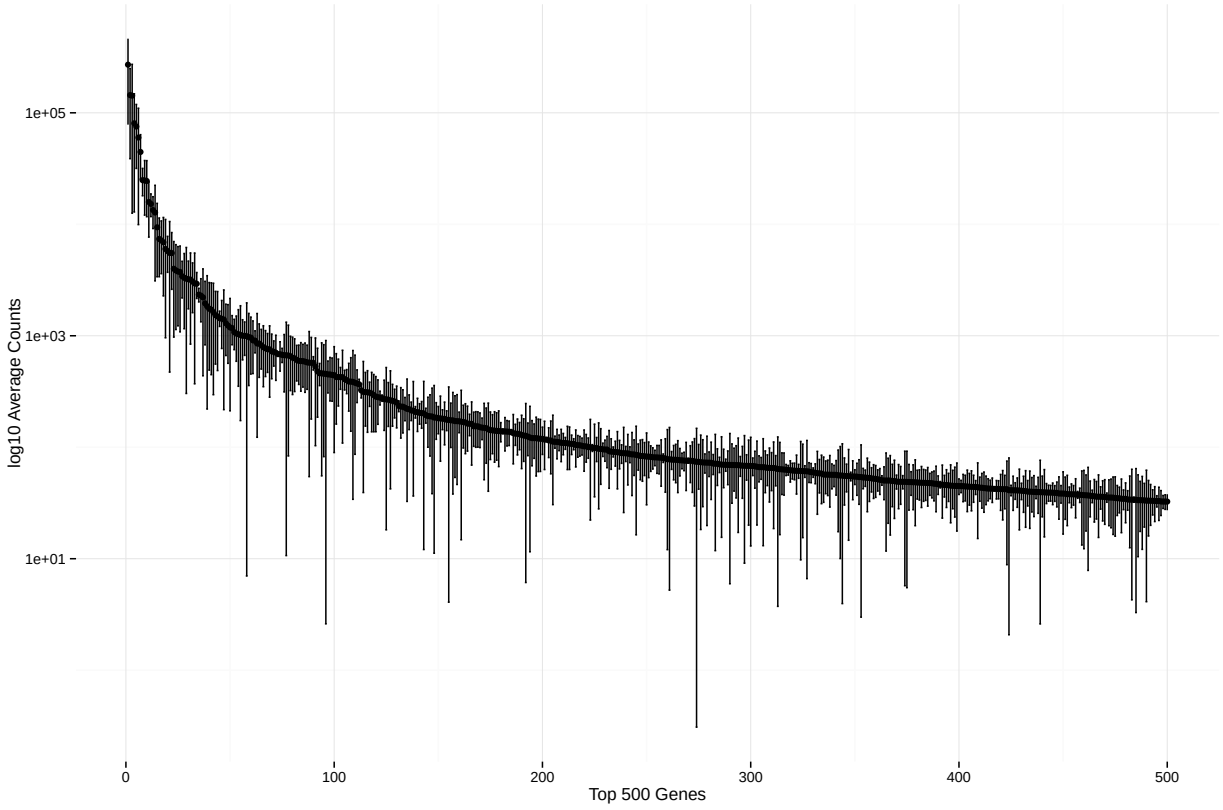



```
##      Day_1               Day_2               Day_3
##  Min.   :      0.0   Min.   :     0.00   Min.   :     0.00
##  1st Qu.:      1.0   1st Qu.:     0.00   1st Qu.:     0.00
##  Median :      3.0   Median :     0.00   Median :     0.00
##  Mean   :    223.8   Mean   :    39.78   Mean   :    56.19
##  3rd Qu.:      6.0   3rd Qu.:     3.00   3rd Qu.:     2.00
##  Max.   :1061233.0   Max.   :101252.00   Max.   :237497.00
##     Night_4             Night_5          Night_6
##  Min.   :     0.0   Min.   :     0   Min.   :     0.00
##  1st Qu.:     0.0   1st Qu.:     1   1st Qu.:     1.00
##  Median :     0.0   Median :     3   Median :     3.00
##  Mean   :    90.3   Mean   :   128   Mean   :    77.84
##  3rd Qu.:     2.0   3rd Qu.:     8   3rd Qu.:     8.00
##  Max.   :443727.0   Max.   :523355   Max.   :109966.00
```

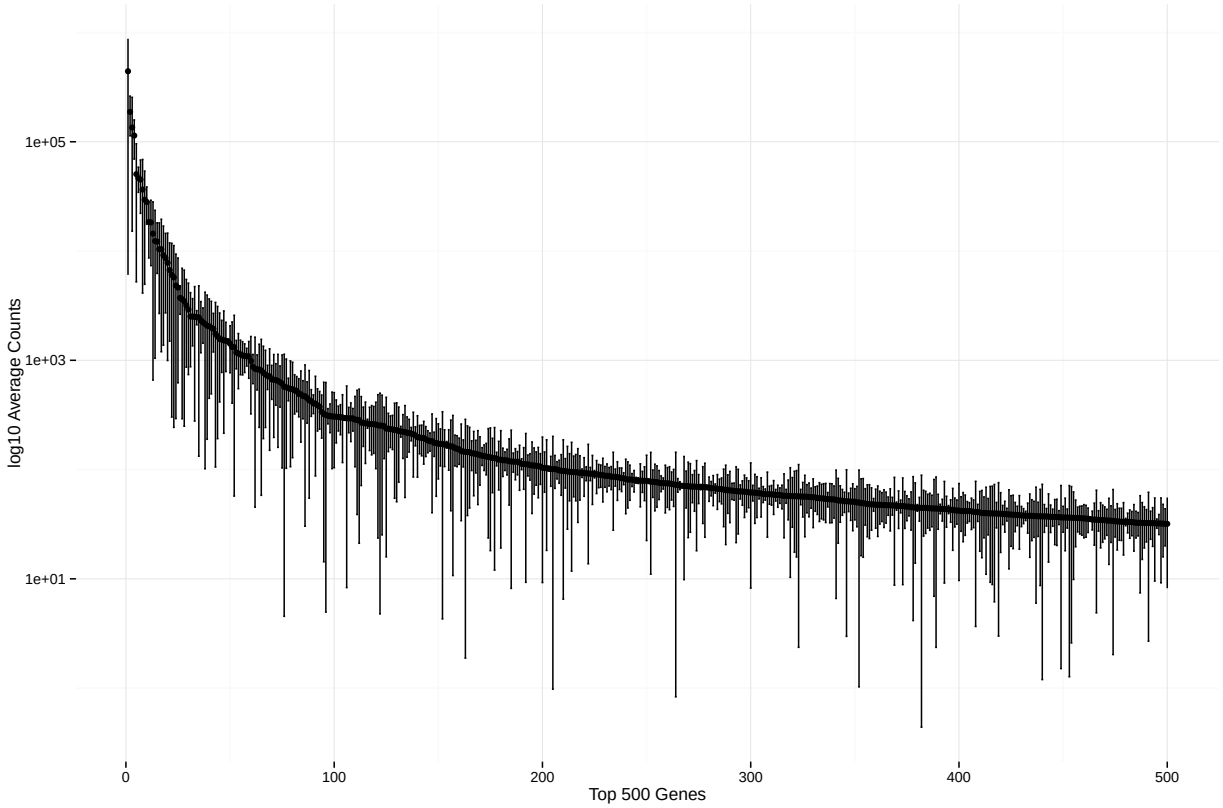
Rank Abundance

Plotting Rank Abundance for top 500 genes.

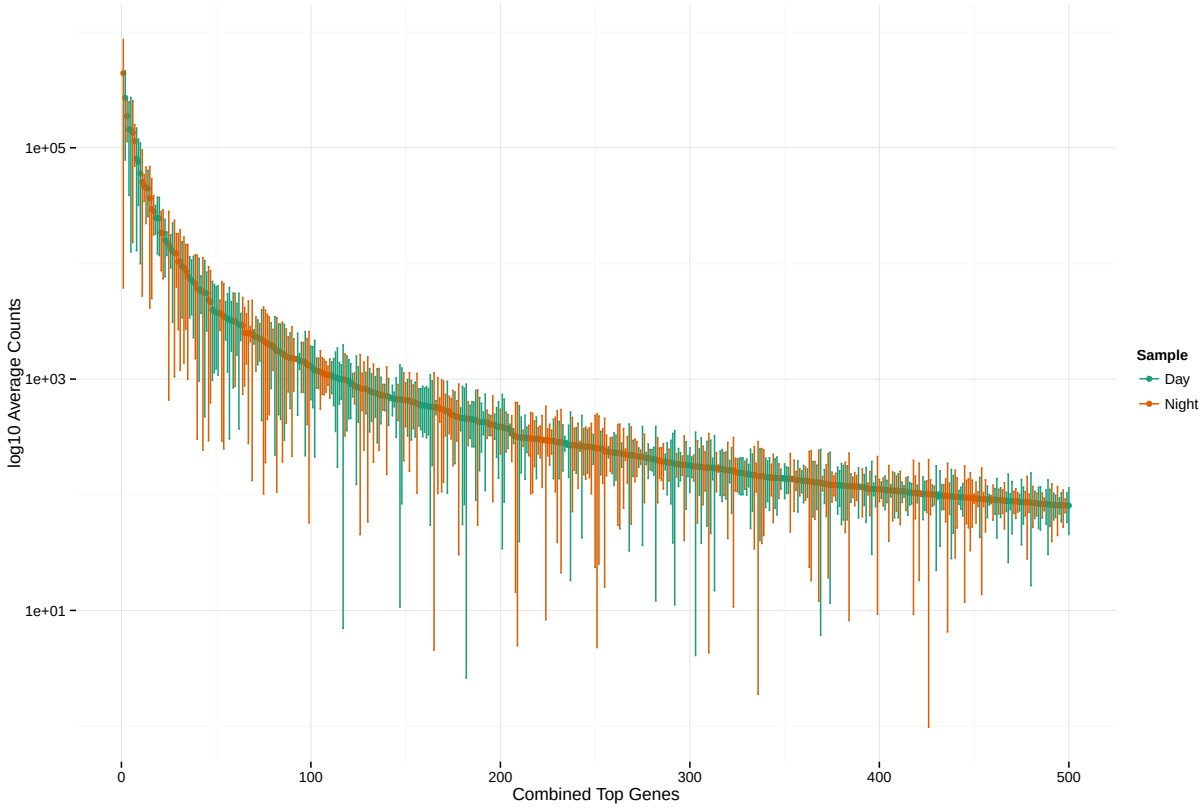
Day Counts



Night Counts



Day vs Night


1e+05
1e+03
1e+01
log10 Average Counts
0
100
200
300
400
500
Combined Top Genes
Sample
Day
Night

Differential Expression

Differential expression was calculater using the DESeq2 wrapper function over 4 processors.

Removing Batch Effects

Using package `sva`. Here is how the package has been described:

> The sva package contains functions for removing batch effects and other unwanted variation in high-throughput experiment. Specifically, the sva package contains functions for the identifying and building surrogate variables for high-dimensional data sets. Surrogate variables are covariates constructed directly from high-dimensional data (like gene expression/RNA sequencing/methylation/brain imaging data) that can be used in subsequent analyses to adjust for unknown, unmodeled, or latent sources of noise. The sva package can be used to remove artifacts in three ways: (1) identifying and estimating surrogate variables for unknown sources of variation in high-throughput experiments (Leek and Storey 2007 PLoS Genetics,2008 PNAS), (2) directly removing known batch effects using ComBat (Johnson et al. 2007 Biostatistics) and (3) removing batch effects with known control probes (Leek 2014 biorXiv). Removing batch effects and using surrogate variables in differential expression analysis have been shown to reduce dependence, stabilize error rate estimates, and improve reproducibility, see (Leek and Storey 2007 PLoS Genetics, 2008 PNAS or Leek et al. 2011 Nat. Reviews Genetics).

```
## Number of significant surrogate variables is:  2
## Iteration (out of 5 ):1  2  3  4  5
```

Results before removing batch effects

As `res` is a DataFrame object, it carries metadata with information on the meaning of the columns:

```
## DataFrame with 6 rows and 2 columns
##                         type                                        description
##                  <character>                                        <character>
## baseMean       intermediate      mean of normalized counts for all samples
## log2FoldChange      results log2 fold change (MAP): condition Day vs Night
## lfcSE               results         standard error: condition Day vs Night
## stat                results         Wald statistic: condition Day vs Night
## pvalue              results      Wald test p-value: condition Day vs Night
## padj                results                           BH adjusted p-values

##
## out of 12089 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 3, 0.025%
## LFC < 0 (down)   : 0, 0%
## outliers [1]     : 78, 0.65%
## low counts [2]   : 10871, 90%
## (mean count < 10.9)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Results after removing batch effects

```
## DataFrame with 6 rows and 2 columns
##                         type                                     description
##                  <character>                                     <character>
## baseMean       intermediate      mean of normalized counts for all samples
## log2FoldChange      results log2 fold change (MAP): condition Day vs Night
## lfcSE               results         standard error: condition Day vs Night
## stat                results         Wald statistic: condition Day vs Night
## pvalue              results      Wald test p-value: condition Day vs Night
## padj                results                           BH adjusted p-values
```

```
##
## out of 12089 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)     : 61, 0.5%
## LFC < 0 (down)   : 8, 0.066%
## outliers [1]     : 0, 0%
## low counts [2]   : 11484, 95%
## (mean count < 25.1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Multiple testing

Novices in high-throughput biology often assume that thresholding these p values at a low value, say 0.05, as is often done in other settings, would be appropriate – but it is not. We briefly explain why: There are 193 genes with a p value below 0.05 among the 12089 genes, for which the test succeeded in reporting a p value.

Now, assume for a moment that the null hypothesis is true for all genes, i.e., no gene is affected by the treatment with dexamethasone. Then, by the definition of p value, we expect up to 5% of the genes to have a p value below 0.05. This amounts to 604 genes. If we just considered the list of genes with a p value below 0.05 as differentially expressed, this list should therefore be expected to contain up to 604.45/193=313.1865285% false positives.

DESeq2 uses the Benjamini-Hochberg (BH) adjustment as described in the base R p.adjust function; in brief, this method calculates for each gene an adjusted p value which answers the following question: if one called significant all genes with a p value less than or equal to this gene's p value threshold, what would be the fraction of false positives (the false discovery rate, FDR) among them (in the sense of the calculation outlined above)? These values, called the BH-adjusted p values, are given in the column padj of the res object. Hence, if we consider a fraction of 10% false positives acceptable, we can consider all genes with an adjusted p value below 10% = 0.1 as significant. How many such genes are there?

```
## [1] 69
```

We subset the results table to these genes and then sort it by the log2 fold change estimate to get the significant genes with the strongest down-regulation.

```
## log2 fold change (MAP): condition Day vs Night
## Wald test p-value: condition Day vs Night
## DataFrame with 6 rows and 6 columns
##                                  baseMean log2FoldChange     lfcSE
##                                 <numeric>      <numeric> <numeric>
```

```
## scaffold_247603__MIS_1019830.5    48.50516      -4.385639 1.1798426
## scaffold_762984__MIS_10200131.2 7636.65755      -3.871386 1.0265900
## scaffold_39942__MIS_10004005.1   276.22010      -3.429412 0.9464202
## scaffold_83360__MIS_10039751.14  183.09975      -2.921465 0.9417198
## scaffold_762984__MIS_10200131.6  180.97034      -2.810339 0.9217688
## scaffold_109736__MIS_10069576.1   70.80729      -2.556002 1.0003326
##                                      stat       pvalue        padj
##                                 <numeric>    <numeric>   <numeric>
## scaffold_247603__MIS_1019830.5  -3.717139 0.0002014918 0.004376742
## scaffold_762984__MIS_10200131.2 -3.771112 0.0001625219 0.004007650
## scaffold_39942__MIS_10004005.1  -3.623562 0.0002905734 0.005670869
## scaffold_83360__MIS_10039751.14 -3.102265 0.0019204584 0.026406303
## scaffold_762984__MIS_10200131.6 -3.048855 0.0022971536 0.030212564
## scaffold_109736__MIS_10069576.1 -2.555153 0.0106141265 0.093065892
```

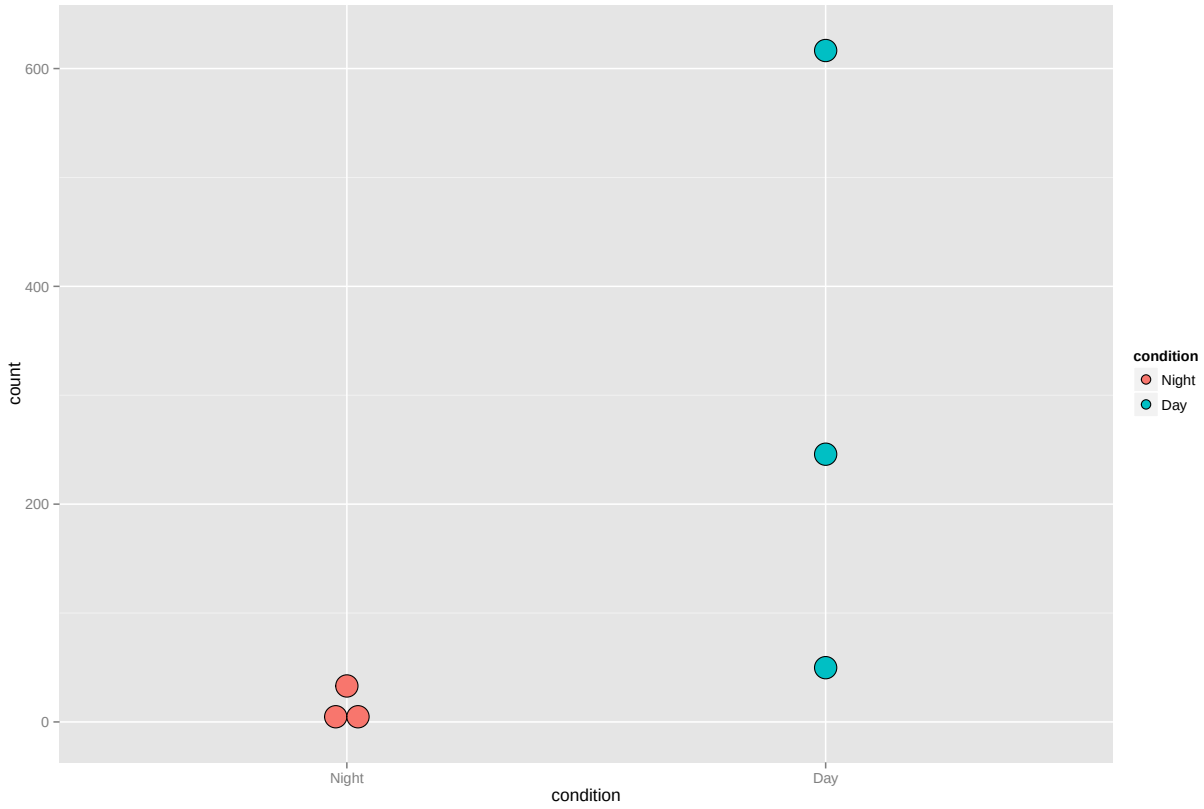
. . . and with the strongest upregulation.

```
## log2 fold change (MAP): condition Day vs Night
## Wald test p-value: condition Day vs Night
## DataFrame with 6 rows and 6 columns
##                                  baseMean log2FoldChange     lfcSE
##                                 <numeric>      <numeric> <numeric>
## scaffold_200891__MIS_10160517.7 202.78263       5.323325 1.0677017
## scaffold_12010__MIS_10012011.12 158.62925       5.252253 0.8786546
## scaffold_12010__MIS_10012011.1   45.50202       4.771448 1.0245247
## scaffold_181056__MIS_10140692.2 750.19279       4.654385 0.9769245
## scaffold_42417__MIS_10006030.1   34.16932       4.463649 1.1039401
## scaffold_7608__MIS_10007609.1   100.82771       4.264579 0.7868240
##                                      stat       pvalue         padj
##                                 <numeric>    <numeric>    <numeric>
## scaffold_200891__MIS_10160517.7  4.985779 6.171253e-07 1.146885e-04
## scaffold_12010__MIS_10012011.12  5.977609 2.264369e-09 1.369943e-06
## scaffold_12010__MIS_10012011.1   4.657231 3.204913e-06 3.231621e-04
## scaffold_181056__MIS_10140692.2  4.764324 1.894874e-06 2.292798e-04
## scaffold_42417__MIS_10006030.1   4.043380 5.268619e-05 2.125010e-03
## scaffold_7608__MIS_10007609.1    5.419991 5.960199e-08 1.802960e-05
```

Diagnostic Plots

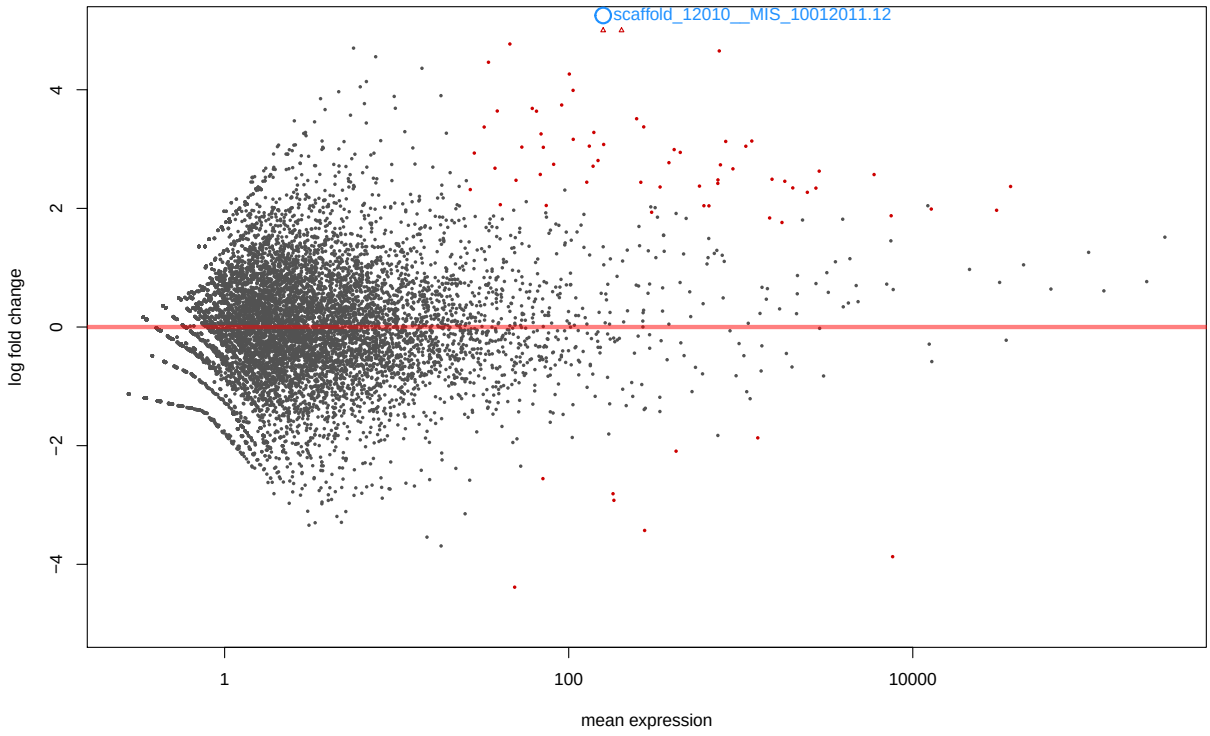
Plot Counts

A quick way to visualize the counts for a particular gene is to use the plotCounts function, which takes as arguments the DESeqDataSet, a gene name, and the group over which to plot the counts.



MA-Plots

An "MA-plot" provides a useful overview for an experiment with a two-group comparison. The log2 fold change for a particular comparison is plotted on the y-axis and the average of the counts normalized by size factor is shown on the x-axis ("M" for minus, because a log ratio is equal to log minus log, and "A" for average).

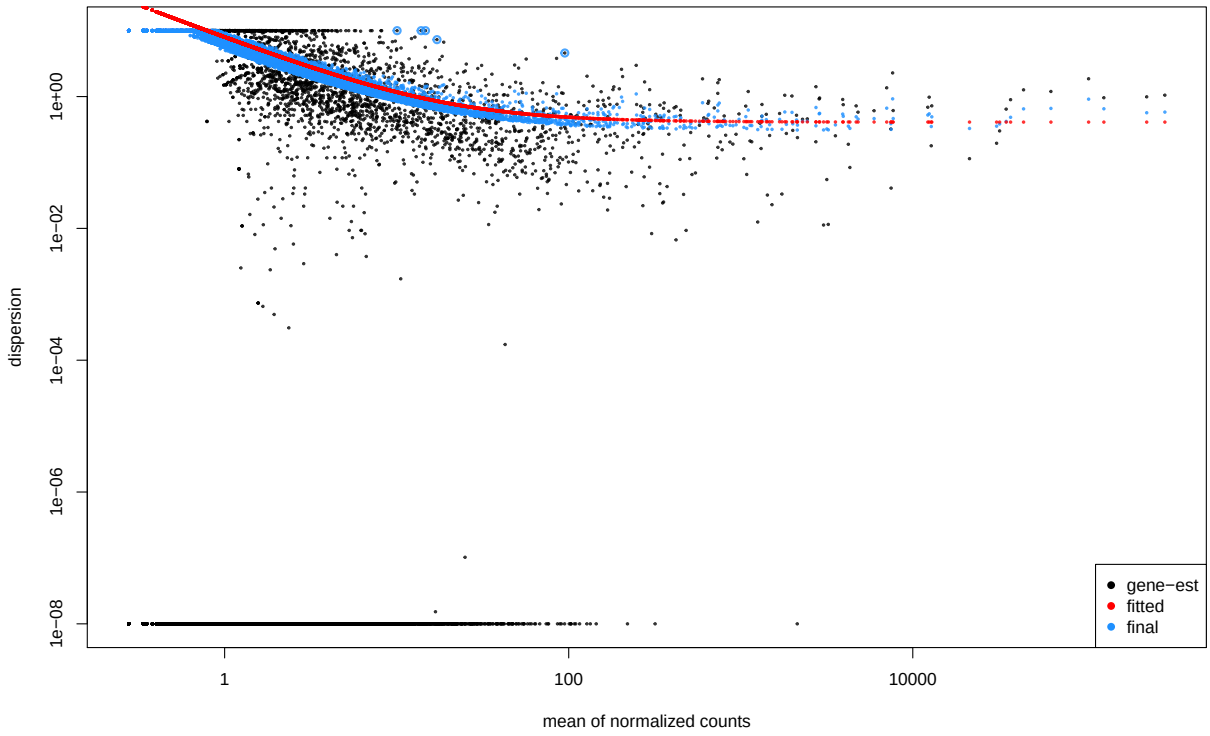


Each gene is represented with a dot. Genes with an adjusted p-value below a threshold (here 0.1, the default) are shown in red. The DESeq2 package incorporates a prior on log2 fold changes, resulting in moderated log2 fold changes from genes with low counts and highly variable counts, as can be seen by the narrowing of spread of points on the left side of the plot. This plot demonstrates that only genes with a large average normalized count contain sufficient information to yield a significant call.

Dispersion Estimataion

Whether a gene is called significant depends not only on its LFC but also on its within-group variability, which DESeq2 quantifies as the dispersion. For strongly expressed genes, the dispersion can be understood as a squared coefficient of variation: a dispersion value of 0.01 means that the gene's expression tends to differ by typically sqrt(0.01)=10% between samples of the same treatment group. For weak genes, the Poisson noise is an additional source of noise.

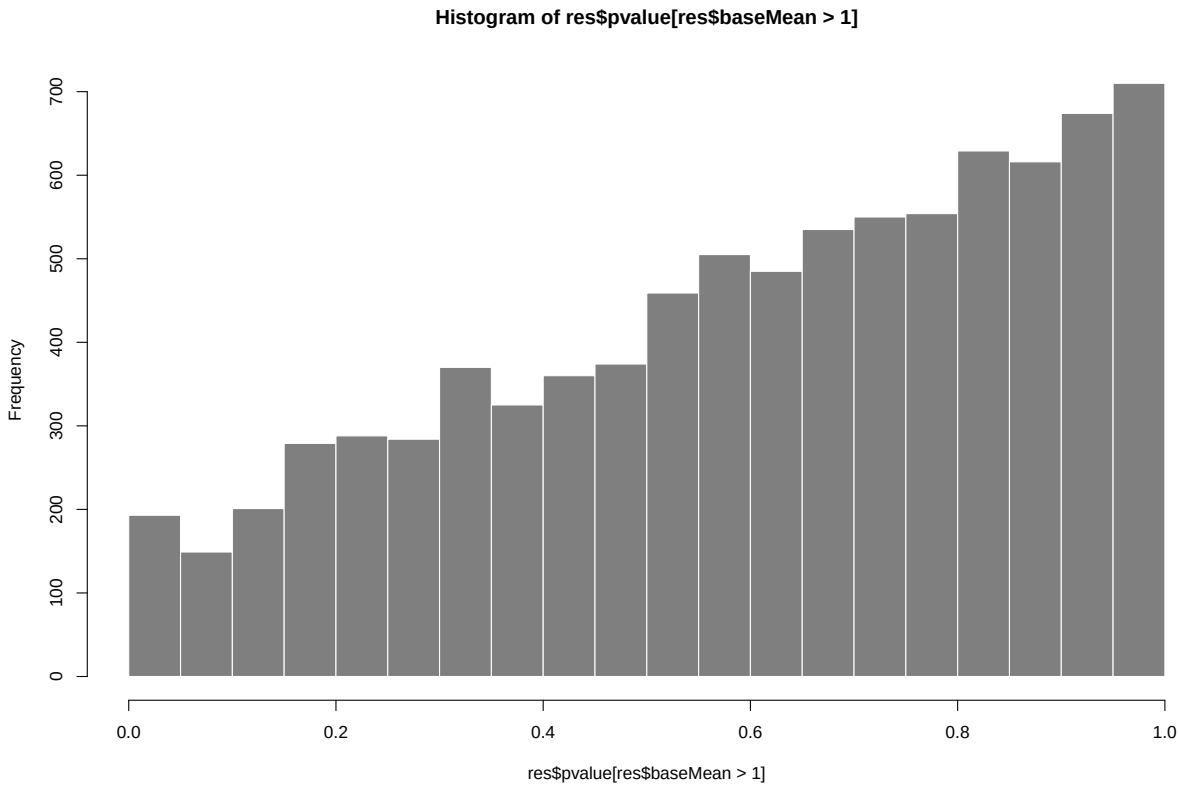
The function plotDispEsts visualizes DESeq2's dispersion estimates:



The black points are the dispersion estimates for each gene as obtained by considering the information from each gene separately. Unless one has many samples, these values fluctuate strongly around their true values. Therefore, we fit the red trend line, which shows the dispersions' dependence on the mean, and then shrink each gene's estimate towards the red line to obtain the final estimates (blue points) that are then used in the hypothesis test. The blue circles above the main "cloud" of points are genes which have high gene-wise dispersion estimates which are labelled as dispersion outliers. These estimates are therefore not shrunk toward the fitted trend line.

P-Value Histogram

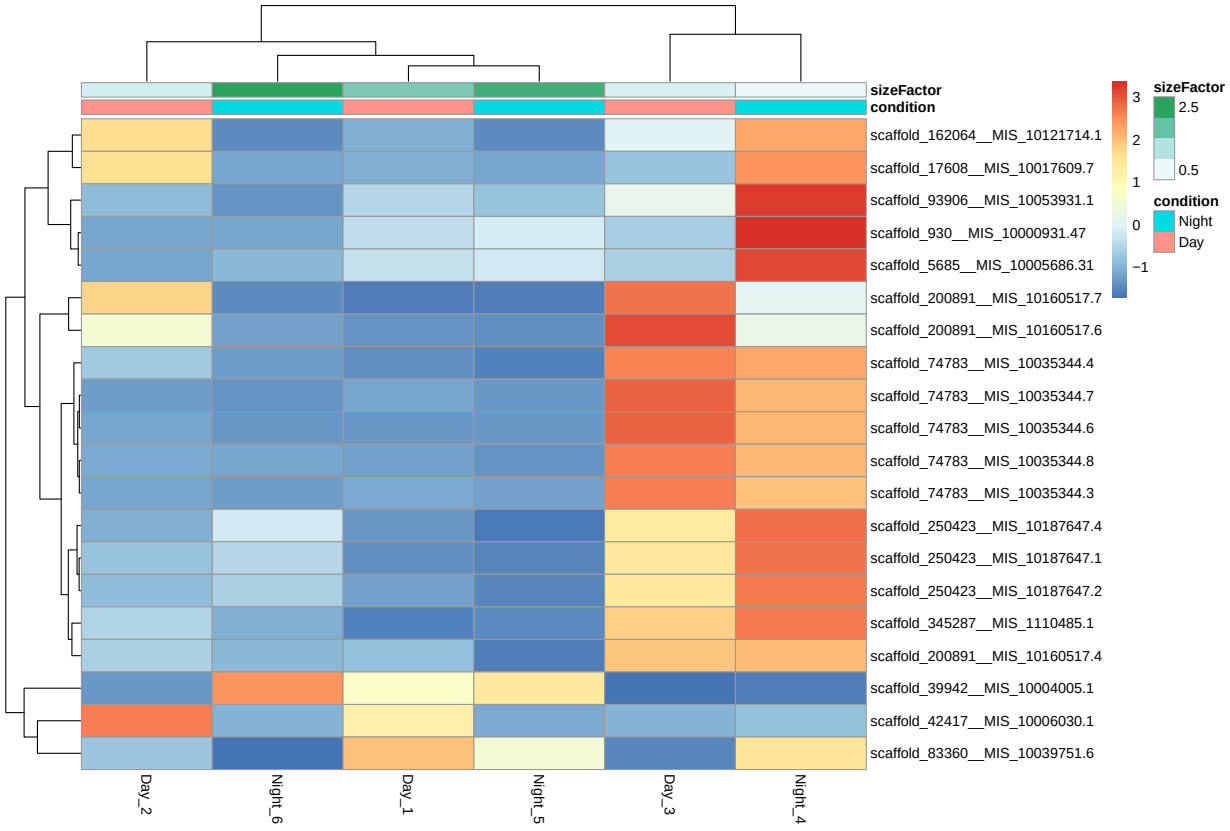
Another useful diagnostic plot is the histogram of the p values.



Gene clustering

In the sample distance heatmap made previously, the dendrogram at the side shows us a hierarchical clustering of the samples. Such a clustering can also be performed for the genes. Since the clustering is only relevant for genes that actually carry signal, one usually carries it out only for a subset of most highly variable genes. Here, for demonstration, let us select the 20 genes with the highest variance across samples. We will work with the rlog transformed counts:

The heatmap becomes more interesting if we do not look at absolute expression strength but rather at the amount by which each gene deviates in a specific sample from the gene's average across all samples. Hence, we center each genes' values across samples, and plot a heatmap. We provide the column side colors to help identify the treated samples (in blue) from the untreated samples (in grey).



We can now see blocks of genes which covary across patients. Note that a set of genes at the top of the heatmap are separating the N061011 cell line from the others. At the bottom of the heatmap, we see a set of genes for which the treated samples have higher gene expression.

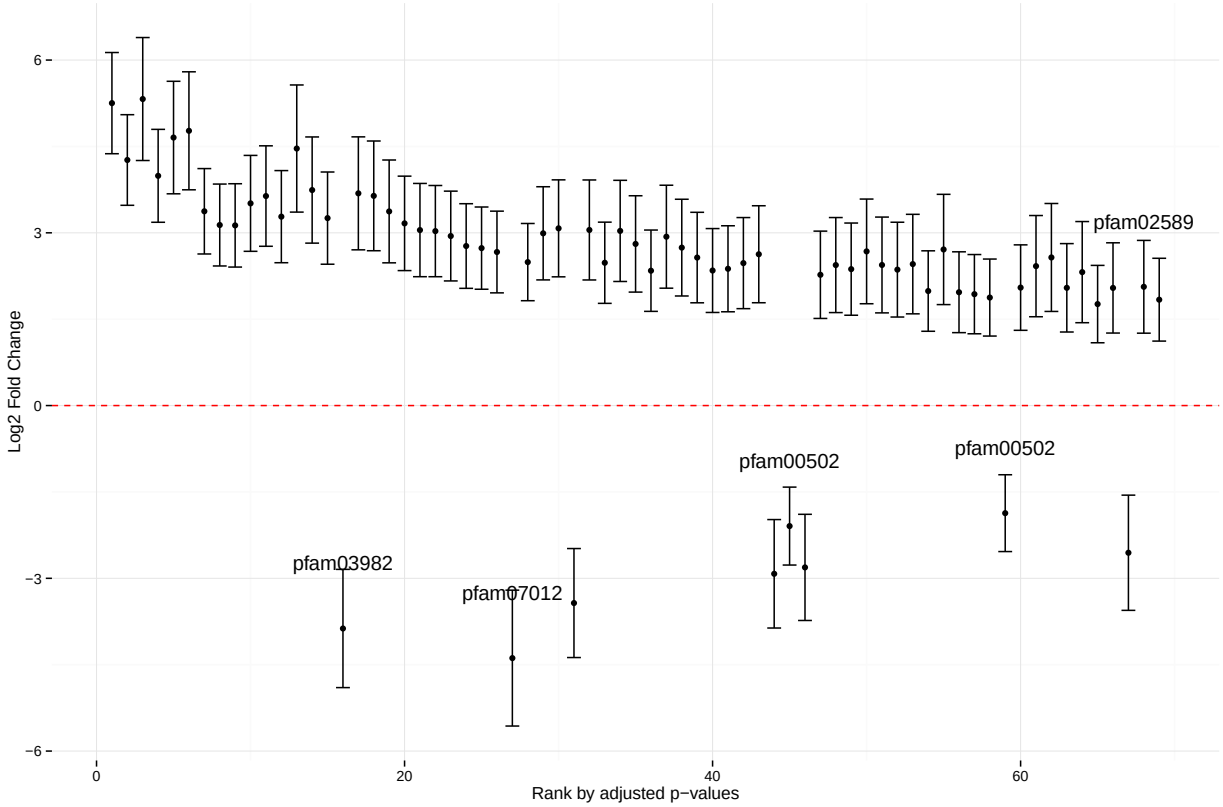
Significant Genes

Total Significant genes: 69

Name	log2FoldChange	padj	IMG_Product	IM
scaffold_12010___MIS_10012011.12	5.252254	0.0000014	NA	NA
scaffold_7608___MIS_10007609.1	4.264579	0.0000180	NA	NA
scaffold_200891___MIS_10160517.7	5.323325	0.0001147	NA	NA
scaffold_356736___MIS_10195009.1	3.989740	0.0001147	NA	NA
scaffold_181056___MIS_10140692.2	4.654385	0.0002293	NA	NA
scaffold_12010___MIS_10012011.1	4.771448	0.0003232	NA	NA
scaffold_356736___MIS_10195009.4	3.373959	0.0004632	NA	NA
scaffold_430758___MIS_10197677.1	3.135407	0.0007875	NA	NA
scaffold_19008___MIS_10019009.27	3.128740	0.0010325	NA	NA
scaffold_12010___MIS_10012011.3	3.511416	0.0015019	NA	NA
scaffold_41583___MIS_10036250.27	3.638513	0.0016652	NA	NA
scaffold_195616___MIS_10155245.4	3.280628	0.0020754	NA	NA
scaffold_12010___MIS_10012011.7	3.742720	0.0021250	NA	NA
scaffold_356736___MIS_10195009.2	3.255334	0.0021250	NA	NA
scaffold_42417___MIS_10006030.1	4.463649	0.0021250	NA	NA
scaffold_133743___MIS_10093445.6	3.163940	0.0040076	NA	NA
scaffold_140713___MIS_10100395.5	2.666652	0.0040076	NA	NA
scaffold_162064___MIS_10121714.1	3.685371	0.0040076	NA	NA
scaffold_163615___MIS_10123264.4	3.641586	0.0040076	NA	NA
scaffold_19008___MIS_10019009.26	3.371516	0.0040076	NA	NA
scaffold_19008___MIS_10019009.28	2.734130	0.0040076	NA	NA
scaffold_232359___MIS_10185986.7	3.029628	0.0040076	NA	NA
scaffold_242878___MIS_10186954.9	2.770531	0.0040076	NA	NA
scaffold_6544___MIS_10006545.32	3.047633	0.0040076	NA	NA
scaffold_748781___MIS_10200104.4	2.943991	0.0040076	NA	NA
scaffold_762984___MIS_10200131.2	-3.871386	0.0040076	Diacylglycerol acyltransferase	pfa
scaffold_19008___MIS_10019009.29	2.491403	0.0043767	NA	NA
scaffold_247603___MIS_1019830.5	-4.385639	0.0043767	Curlin associated repeat	pfa
scaffold_12010___MIS_10012011.8	2.991201	0.0045363	NA	NA
scaffold_12010___MIS_10012011.4	3.077654	0.0051856	NA	NA
scaffold_39942___MIS_10004005.1	-3.429413	0.0056709	NA	NA
scaffold_36112___MIS_10001001.11	3.049487	0.0081417	NA	NA
scaffold_425595___MIS_1167750.1	2.479972	0.0081417	NA	NA
scaffold_200195___MIS_10159822.1	3.033188	0.0098389	NA	NA
scaffold_276564___MIS_10189820.1	2.807072	0.0136859	NA	NA
scaffold_407786___MIS_10197014.2	2.341947	0.0153152	NA	NA
scaffold_12010___MIS_10012011.15	2.742948	0.0169443	NA	NA
scaffold_138956___MIS_10098641.7	2.570315	0.0169443	NA	NA
scaffold_232359___MIS_10185986.6	2.932548	0.0169443	NA	NA
scaffold_778707___MIS_1178057.1	2.345330	0.0194168	NA	NA
scaffold_230594___MIS_10185835.2	2.375447	0.0216992	NA	NA
scaffold_113303___MIS_10073104.2	2.473834	0.0253712	NA	NA
scaffold_622706___MIS_1174630.1	2.627902	0.0254262	NA	NA
scaffold_83360___MIS_10039751.14	-2.921465	0.0264063	NA	NA
scaffold_55518___MIS_10017391.9	-2.092896	0.0266415	Phycobilisome protein	pfa
scaffold_762984___MIS_10200131.6	-2.810339	0.0302126	NA	NA
scaffold_47856___MIS_10037233.11	2.271049	0.0352622	NA	NA
scaffold_34818___MIS_10034794.3	2.369180	0.0383357	NA	NA
scaffold_564989___MIS_10199597.1	2.439948	0.0383357	NA	NA

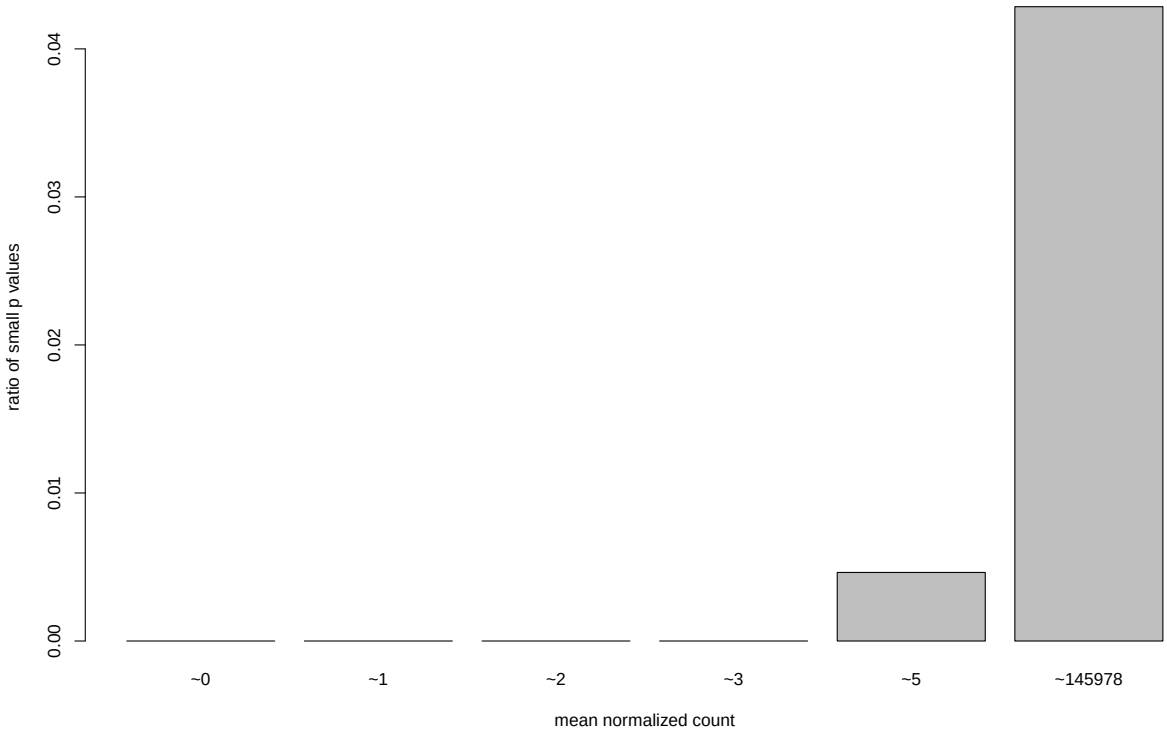
Name	log2FoldChange	padj	IMG_Product	IM
scaffold_65654___MIS_10026723.10	2.677312	0.0390149	NA	NA
scaffold_654999___MIS_10199933.4	2.441415	0.0396365	NA	NA
scaffold_34818___MIS_10034794.2	2.360733	0.0480828	NA	NA
scaffold_407786___MIS_10197014.1	1.988858	0.0499648	NA	NA
scaffold_92040___MIS_10052095.5	2.457089	0.0499648	NA	NA
scaffold_74783___MIS_10035344.4	2.710810	0.0509513	NA	NA
scaffold_115914___MIS_10075695.2	1.875820	0.0521974	NA	NA
scaffold_19008___MIS_10019009.30	1.968143	0.0521974	NA	NA
scaffold_65654___MIS_10026723.6	1.934619	0.0521974	NA	NA
scaffold_92250___MIS_10052301.5	-1.868457	0.0521974	Phycobilisome protein	pfa
scaffold_9458___MIS_10009459.6	2.048751	0.0580288	NA	NA
scaffold_131262___MIS_10090974.15	2.421476	0.0581513	NA	NA
scaffold_655996___MIS_1175590.3	2.571980	0.0592087	NA	NA
scaffold_352076___MIS_10194766.4	2.044935	0.0741571	NA	NA
scaffold_276003___MIS_10189776.3	2.317193	0.0788023	NA	NA
scaffold_276913___MIS_10189851.1	1.762433	0.0810952	NA	NA
scaffold_74783___MIS_10035344.6	2.042968	0.0836855	NA	NA
scaffold_109736___MIS_10069576.1	-2.556002	0.0930659	NA	NA
scaffold_164019___MIS_10123668.4	1.838601	0.0930659	NA	NA
scaffold_242781___MIS_1015452.4	2.062530	0.0930659	Uncharacterised ACR, YkgG family COG1556	pfa

Plot


6
3
0
−3
−6
Log2 Fold Change
0
20
40
60
Rank by adjusted p−values
pfam02589
pfam00502
pfam00502
pfam03982
pfam07012

Independent Filtering

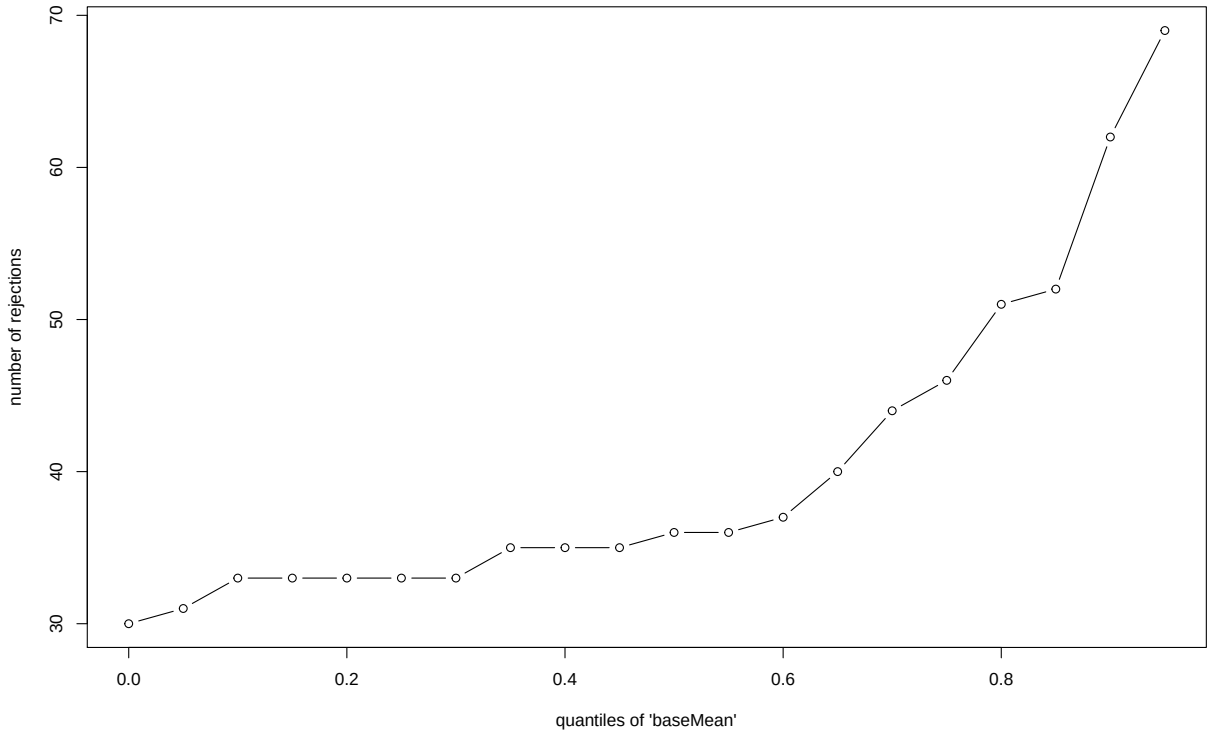
The MA plot highlights an important property of RNA-Seq data. For weakly expressed genes, we have no chance of seeing differential expression, because the low read counts suffer from so high Poisson noise that any biological effect is drowned in the uncertainties from the read counting. We can also show this by examining the ratio of small p values (say, less than, 0.01) for genes binned by mean normalized count:



At first sight, there may seem to be little benefit in filtering out these genes. After all, the test found them to be non-significant anyway. However, these genes have an influence on the multiple testing adjustment, whose performance improves if such genes are removed. By removing the weakly-expressed genes from the input to the FDR procedure, we can find more genes to be significant among those which we keep, and so improved the power of our test. This approach is known as independent filtering.

The DESeq2 software automatically performs independent filtering which maximizes the number of genes which will have adjusted p value less than a critical value (by default, alpha is set to 0.1). This automatic independent filtering is performed by, and can be controlled by, the results function. We can observe how the number of rejections changes for various cutoffs based on mean normalized count. The following optimal threshold and table of possible values is stored as an attribute of the results object.

```
##      95%
## 25.09761
```



The term independent highlights an important caveat. Such filtering is permissible only if the filter criterion is independent of the actual test statistic. Otherwise, the filtering would invalidate the test and consequently the assumptions of the BH procedure. This is why we filtered on the average over all samples: this filter is blind to the assignment of samples to the treatment and control group and hence independent. The independent filtering software used inside DESeq2 comes from the genefilter package, which contains a reference to a paper describing the statistical foundation for independent filtering.

Session Info

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.4 (Yosemite)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] knitr_1.10.5              sva_3.14.0
##  [3] genefilter_1.50.0         mgcv_1.8-7
##  [5] nlme_3.1-121              BiocParallel_1.2.9
##  [7] dplyr_0.4.2               tidyr_0.2.0
##  [9] vioplot_0.2               sm_2.2-5.4
## [11] ggdendro_0.1-15           ggplot2_1.0.1
## [13] PoiClaClu_1.0.2           pheatmap_1.0.7
## [15] RColorBrewer_1.1-2        DESeq2_1.8.1
## [17] RcppArmadillo_0.5.200.1.0 Rcpp_0.11.6
## [19] GenomicRanges_1.20.5      GenomeInfoDb_1.4.1
## [21] IRanges_2.2.5             S4Vectors_0.6.2
## [23] BiocGenerics_0.14.0
##
## loaded via a namespace (and not attached):
##  [1] locfit_1.5-9.1       lattice_0.20-33      assertthat_0.1
##  [4] digest_0.6.8         R6_2.1.0             plyr_1.8.3
##  [7] futile.options_1.0.0 acepack_1.3-3.3      RSQLite_1.0.0
## [10] evaluate_0.7         highr_0.5            lazyeval_0.1.10
## [13] annotate_1.46.1      Matrix_1.2-2         rpart_4.1-10
## [16] rmarkdown_0.7        proto_0.3-10         labeling_0.3
## [19] splines_3.2.1        geneplotter_1.46.0   stringr_1.0.0
## [22] foreign_0.8-65       munsell_0.4.2        htmltools_0.2.6
## [25] nnet_7.3-10          gridExtra_2.0.0      Hmisc_3.16-0
## [28] XML_3.98-1.3         MASS_7.3-43          grid_3.2.1
## [31] xtable_1.7-4         gtable_0.1.2         DBI_0.3.1
## [34] magrittr_1.5         formatR_1.2          scales_0.2.5
## [37] stringi_0.5-5        XVector_0.8.0        reshape2_1.4.1
## [40] latticeExtra_0.6-26  futile.logger_1.4.1  Formula_1.2-1
## [43] lambda.r_1.1.7       tools_3.2.1          Biobase_2.28.0
## [46] survival_2.38-3      yaml_2.1.13          AnnotationDbi_1.30.1
## [49] colorspace_1.2-6     cluster_2.0.3
```