Tutorial
Statistical analysis of RNA-Seq data

Ignacio González

Plateforme Bioinformatique – INRA Toulouse
Plateforme Biostatistique – IMT Université Toulouse III

Toulouse, 20/21 novembre 2014

Contents

1 Introduction

1.1 Motivation

RNA sequencing (RNA-seq) technologies have recently come to prominence as technique for characterizing RNA transcripts and comparative analyses of their abundances. It is used in many areas of biology including functional genomics, developmental biology and cancer biology. In RNA-seq, mapped reads are typically aggregated to counts at some level of interest, such as transcripts, exons, or genes. The count for a given gene or region quantifies the expression of that gene or region. We are very often interested in which genes or regions have different levels of expression under different experimental conditions – this is known as differential expression.

The nature of the count data produced in RNA-seq experiments introduces challenges for the analysis of differential expression. The computational and statistical methods for analyzing RNA-seq data are recent, and successful methods for the analysis of microarray data are not directly applicable to RNA-seq data. It seems that dedicated tools, designed especially for count data, will be required to extract the best possible information from RNA-seq data.

Several emerging tools within *Bioconductor* (a repository for open-source software for bioinformatics) have been developed for the differential analysis of count data, including *DESeq*, *DESeq2*, *edgeR*, *baySeq*, *limma* (`voom` function), ... but best practices are not yet clearly established.

1.2 Scope of this tutorial

This tutorial introduces use of R and Bioconductor tools for analysis of RNA-seq count data. The goals are to: (1) develop familiarity with R / Bioconductor software for statistical analysis; (2) expose key statistical issues in the analysis of sequence data; and (3) provide inspiration and a framework for their own projects.

Our tutorial will consist of a hands-on demonstration, walking through an end-to-end analysis of a typical RNA-seq data, going from uploading raw count data to exploring significant variation in transcriptional levels between different groups of samples.

The tutorial focuses on using the *DESeq*, *DESeq2* and *edgeR* packages. It gives an overview of the theory behind the tools featured in the packages and illustrates their features through examples using public and simulated datasets. We will go through the following steps:

1. design and quality assessment of RNA-seq experiments
2. count data normalization
3. statistical models for count data
4. hypothesis and multiple testing
5. statistical analysis of differences
6. independent filtering
7. downstream interpretive analyses

This tutorial is accompanied with a group of practical exercises, with a series of short remarks, exploring the diversity of tasks for which R / Bioconductor are appropriate for the statistical analysis of RNA-seq data.

2 Experimental design

Proper experimental design is the key to any gene expression study. This is needed to ensure that questions of interest can be answered and that this can be done accurately, given experimental constraints, such as cost of reagents and availability of mRNA.

In the simplest case, the aim is to compare expression levels between two conditions, e.g., controlled versus treated or wild-type versus mutant.

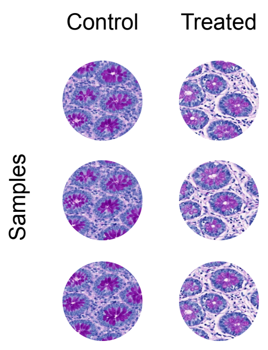


Figure 1: Example of a simple design: control versus treated groups.

More complicated experimental designs can include additional experimental factors, potentially with multiple levels (e.g., multiple mutants, doses of a drug or time points) or may need to account for additional covariates (e.g. experimental batch or sex) or the pairing of samples (e.g., paired tumour and normal tissues from individuals).

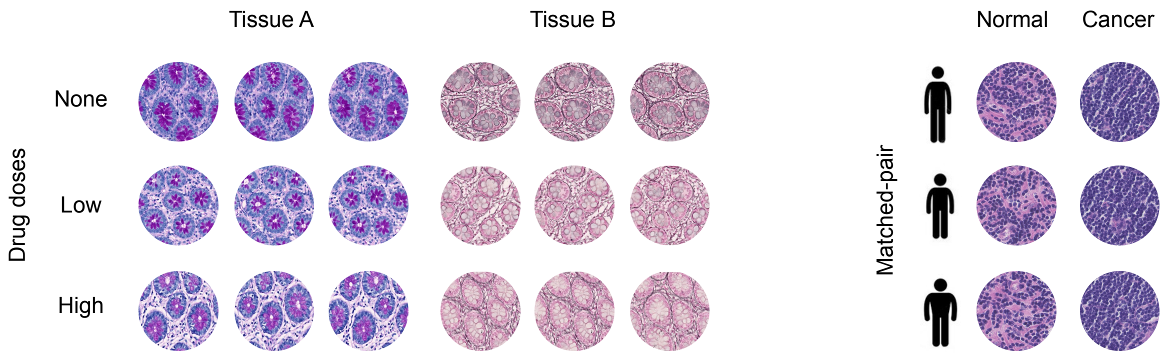
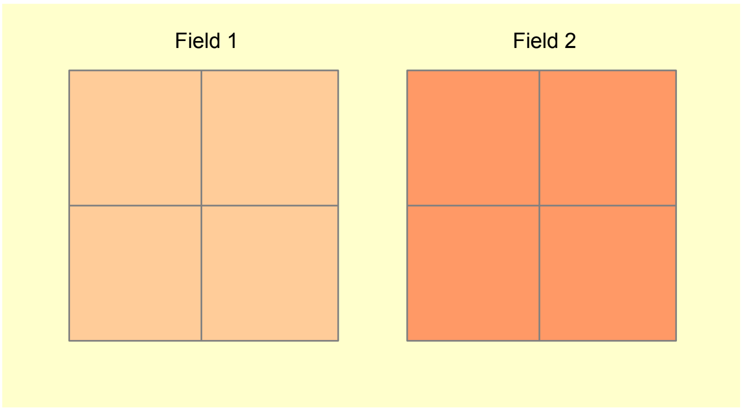


Figure 2: Examples of complex designs. Left: experimental design with two factors, the *tissue* at two levels (A and B) and the *drug* at three levels (*None*, *Low* and *High*). Right: paired experimental design from *Normal* and *Cancer* tissues.

2.1 Motivation

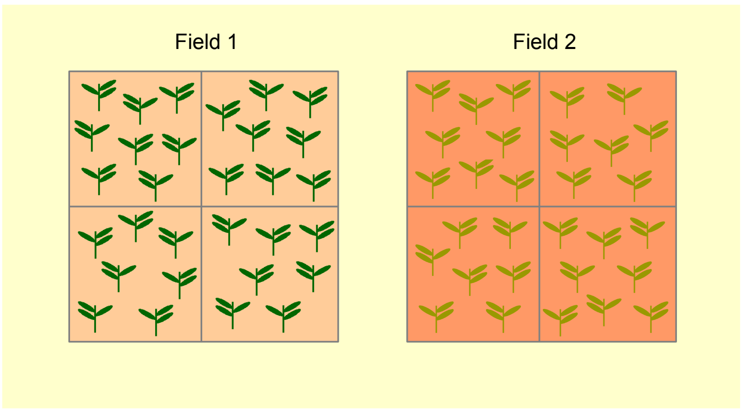
Suppose that you are interested in finding differences between two plant samples, a control and treatment group.

Say there are two fields and four plots per field.



A bad experimental design

A naive design is to put all samples from the same group in a field, for example, all control samples in the "Field 1" and all treated samples in the "Field 2".



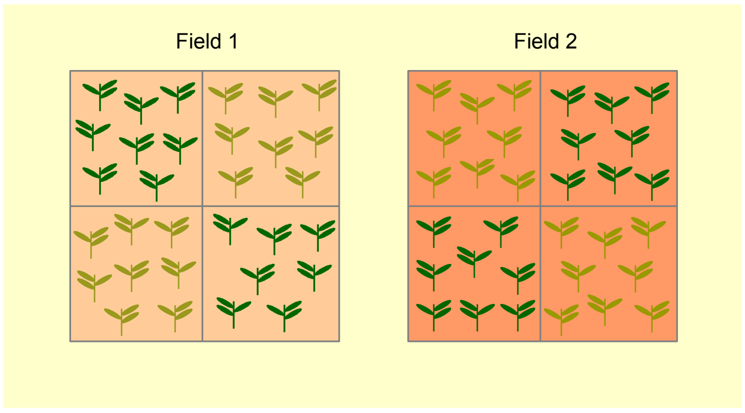
The reason why it is bad design is simple: the observed effect is due to what?

- to the real effect of the treatment on the plants?, or
- to the environmental conditions of each field on the plants?

In statistics, field and group effects are said to be confounded. Whatever the statistical methods, confounded effects can not be separated **after** data have been collected.

A good experimental design

A better approach for the above example is **not to** put all samples from an experimental group in a same field, but make sure each field contains samples from both the control and treatment groups.



Now field and group effects can be estimated separately. In statistics, the field effect and the experimental group effect are no longer confounded.

2.2 The ABC's of experimental design: replication, ramdomization, blocking

The experimenter is interested in the effect of some process or intervention (the "treatment") on experimental units (the "plants"). Many different plants (replications) are assigned randomly to one of two treatments (randomization) then grouped and placed in each field (blocking).

Replication – Measurements are usually subject to variation and uncertainty. Then, measurements are repeated and full experiments are replicated to help identify the sources of variation, to better estimate the true effects of treatments, to further strengthen the experiment's reliability and validity.

Randomization – It is the process of assigning individuals at random to groups or to different groups in an experiment. This reduces bias by equalizing so-called factors (independent variables) that have not been accounted for in the experimental design.

Blocking – Experimental units are grouped into homogeneous clusters in an attempt to improve the comparison of treatments by randomly allocating the treatments within each cluster or 'block'. Blocking reduces known but irrelevant sources of variation between units and thus allows greater precision in the estimation of the source of variation under study.

- These principles are well known but their implementation, in the NGS context, often requires significant planning and statistical expertise.
- In the absence of a proper design, it is impossible to separate "biological variation" from "technical variation".

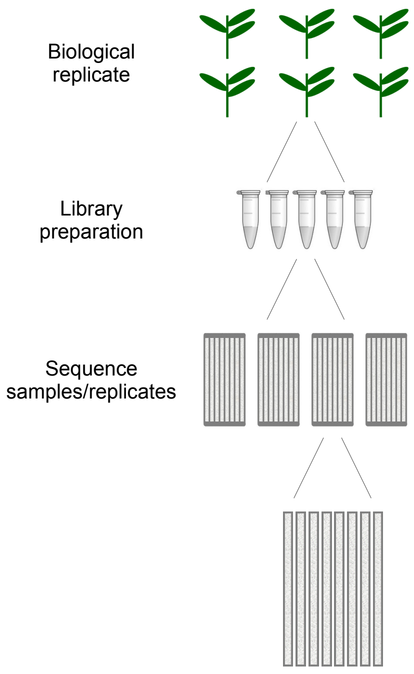
2.3 Sources of variation in RNA-seq experiments

In a RNA-seq experiment, there are two sources of variability that may affect the results:

Biological variation – is intrinsic to all organisms; it may be influenced by genetic or environmental factors, as well as by whether the samples are pooled or individual.

Technical variation – is the variability in the measurements from a sample subject that persists even under identical experimental conditions. That is, the uncertainty with which the abundance of each gene in each sample is estimated by the sequencing technology.

The design of an RNA-seq experiment can be considered as having three layers.



The many sources of variation in an RNA-seq experiment can be partitioned along these three layers.

top layer – at the top layer of the experiment are the experimental units, which involve biological variation.

middle layer – technical variation is introduced during the generation of libraries of cDNA fragments, which involves various ligations of adaptors and PCR amplifications.

bottom layer – beyond the library preparation effect, there are also other technology-specific effects. For example, variation from one flow cell to another resulting in flow cell effect. In addition, there exits variation between the individual lanes within a flow cell due to systematic variation in sequencing cycling and/or base-calling.

Among these sources of variation

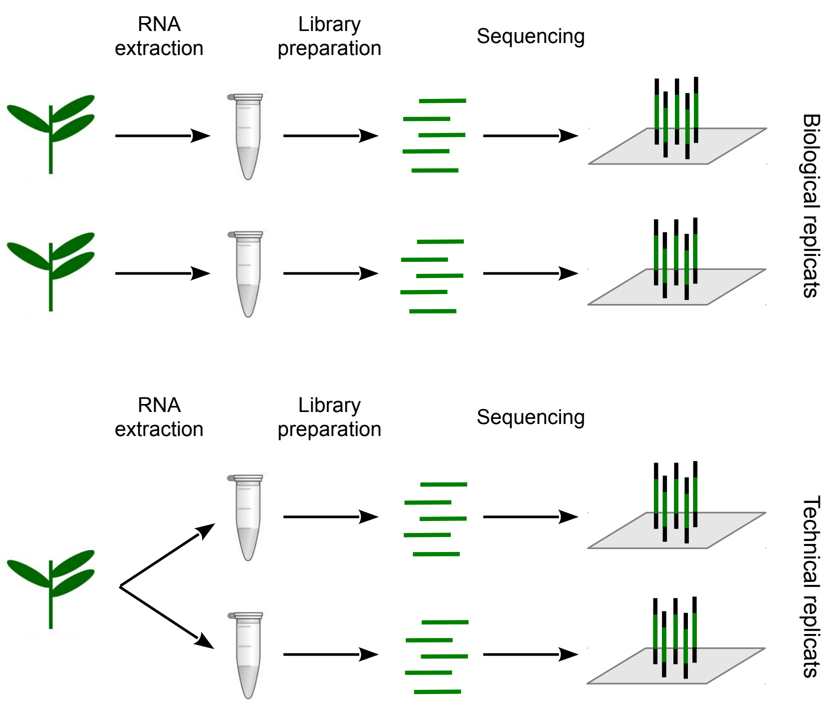
$$\text{lane effect} < \text{flow cell effect} < \text{library preparation effect} \ll \text{biological effect}$$

2.4 Biological and technical replicates

To mitigate the effect of biological and technical variability, it is generally accepted that at least three biological and two technical replicates per biological replicate be performed for each experiment.

Biological replicates – consist of different biological samples that are processed through the sequencing process separately, for example, two cultures of S. cerevisiae are grown separately and sequenced separately. The biological replicates allow for the estimation of within-treatment group (biological) variability, provide information that is necessary for making inferences between treatment groups, and give rise to conclusions that can be generalized.

Technical replicates – in RNA-Seq, technical replicates are replicates where the biological material is the same in each replicate but the technical steps used to measure gene expression are performed separately. For example, a single culture of S. cerevisiae is grown and two samples are drawn from it and sequenced separately. In true technical replicates, all steps are replicated, including library preparation.



2.5 Design choice

In a experiment several designs, that seem equally suited can be devised, and we need some principles for choosing one from several possibilities. Such principles are discussed below, where we focus on the question of identifying differentially expressed genes between two conditions or treatments.

The most important design issues will be determined by the resources, the technical constraints, and the scientific hypotheses under investigation. These will dictate the number of treatments, the number of biological replicates per treatment, the number of unique bar codes that can be included in a single lane, and the number of lanes available for sequencing.

2.5.1 Multiplex versus typical design

In **multiplex design** DNA fragments are labeled or barcoded with sample specific sequences that allow multiple samples to be included in the same sequencing reaction while maintaining with high fidelity sample identities downstream.

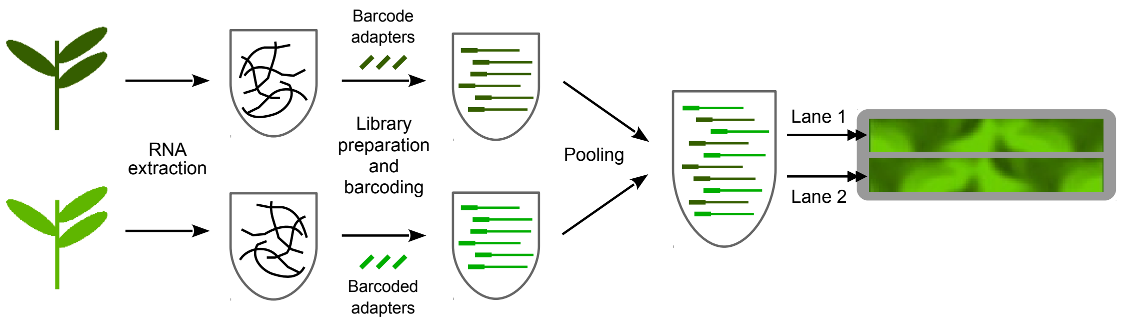


Figure 3: Multiplex RNA-seq design

- All the samples of RNA are pooled into the same batch and then sequenced in one lane of a flow cell.
- Any library preparation effects are the same for all the samples, and all effects due to lane will be the same for all samples.

- This can be achieved by barcoding the RNA immediately after fragmentation.

Multiplexing can be used as a control quality feature, apart of increasing the number of samples per sequencing run, it offers the flexibility to construct balanced blocked designs for the purpose of testing differential expression.

In **typical design**, independent samples of RNA are loaded into different lanes of the flow cell such that sequencing reactions occur independently between samples.

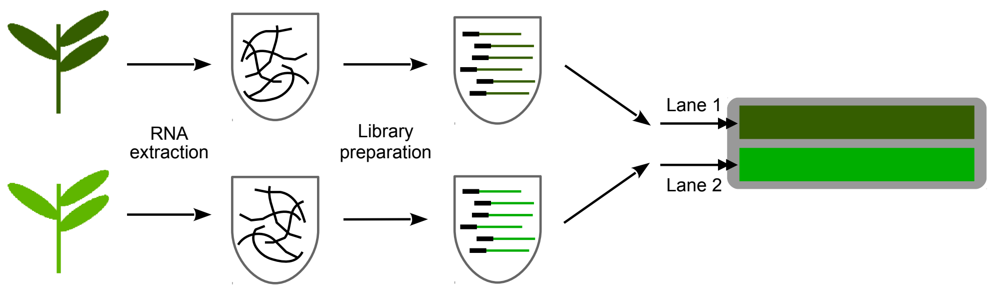


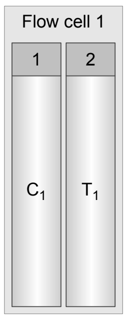
Figure 4: Typical RNA-seq design

2.5.2 Biologically unreplicated design

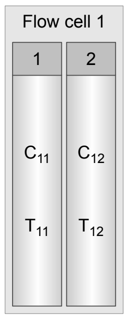
Material at disposal:

control samples treated samples flow cell

- **A biologically unreplicated unblocked design** with RNA isolated from subjects within each group (C_1; T_1) and loaded into individual lanes (e.g., the RNA from the control subject is sequenced in lane 1).



- **A biologically unreplicated balanced block design** with C_1 split (bar coded) into two technical replicates (C_{11}, C_{12}) and T_1 split into two technical replicates (T_{11}, T_{12}) and input to lanes 1 and 2.



Limitations of unreplicated data:

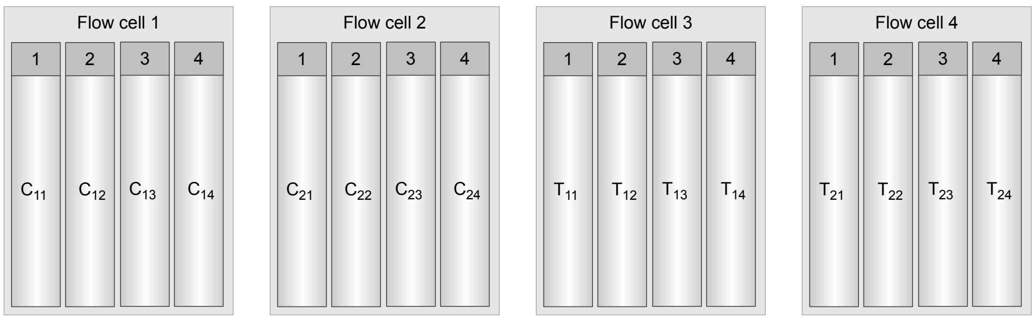
- Complete lack of knowledge about biological variation.
- Without an estimate of variability (i.e. within treatment groups), there is no basis for inference (between treatment groups).
- The results only apply to the specific subjects included in the study.

2.5.3 Biologically replicated design

Material to disposition:

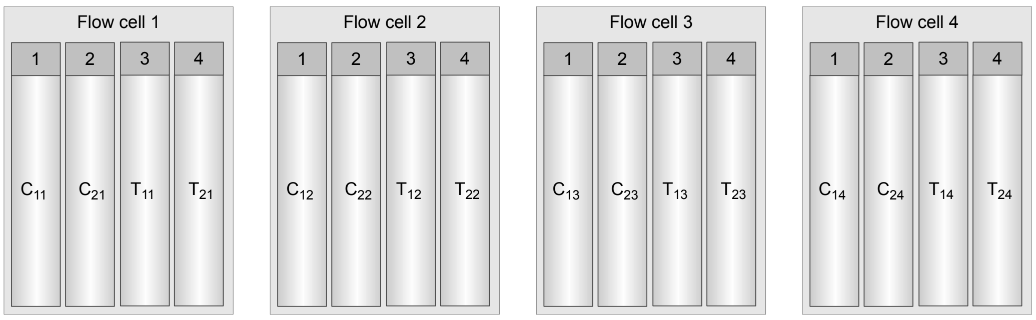
control samples treated samples flow cell

- **A biologically replicated unblocked design** with two subjects from control group C (C_1, C_2) and two subjects from treatment group T (T_1, T_2).

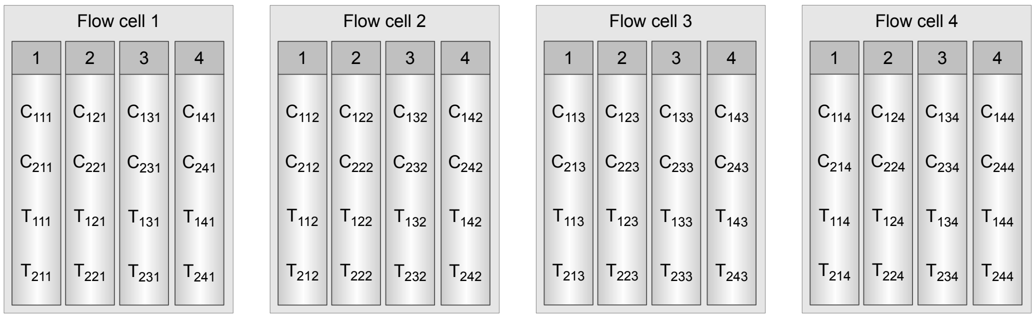


  - Any differences in expression between control and treatment are confounded with possible lane effects that may persist across flow cells.
  - Without careful planning an unblocked design faces a fundamental problem for generalizing the results: **confounding** effects might occur.
  - If the treatment effects can not be separated from possible confounding factors, then for any given gene, there is no way of knowing whether the observed difference in abundance between treatment groups is due to biology or to technology.

- **A biologically replicated balanced incomplete block design** and blocking without multiplexing for two treatment groups (C, T) with two subject per treatment group (C_1, C_2; T_1, T_2) and four technical replicates of each (e.g., C_{11}, C_{12}; C_{13}, C_{14}) and input to four flow cell.



- **A biologically replicated balanced block design** with each subject (e.g., C_1) split (bar coded) into four technical replicates (e.g., $C_{11}, \ldots, C_{14}$) and input to four lanes in each flow cell.



Two main sources of variation that may contribute to confounding effects:

- Batch effects: errors that occur after random fragmentation of the RNA until it is put on the flow cell (PCR, reverse transcription).
- Lane effects: errors that occur from the flow cell until obtaining the data from the sequencing machine (bad sequencing cycles, base-calling).

Bar coding in the balanced block design results in four technical replicates of each sample, while balancing batch and lane effects and blocking on lane. The balanced block design also allows partitioning of batch and lane effects from the within-group biological variability.

2.6 Conclusion

- Replication, randomization and blocking are essential components of any well planned and properly analyzed design.
- The best way to ensure reproducibility and accuracy of results is to include independent biological replicates (technical replicates are no substitute).
- The results from unreplicated data cannot be generalized beyond the sample tested (here, differential expression).
- To acknowledge anticipated nuisance factors (e.g., lane, batch, and flow-cell effects) in the design.
- Realizing of course that it is not possible to determine whether batch and/or lane effects are present a priori, we recommend the use of block designs to protect against observed differences that are attributable to these potential sources of variation.
- NGS platforms allow us to work with the concepts of randomization and blocking (multiplexing). These designs are as good as, if not better than, their unblocked counterparts when batch and/or lane effects are present.

3 Input data and preparations

The starting point for an RNA-Seq experiment is a set of N RNA samples, typically associated with a variety of treatment conditions. Each sample is sequenced, short reads are mapped to the appropriate genome, and the number of reads mapped to each genomic feature of interest is recorded. The set of genewise counts for sample j makes up the expression profile or library for that sample. The expected size of each count is the product of the library size and the relative abundance of that gene in that sample.

As input for data analysis, we expect count data in the form of a rectangular table of integer values. The table cell in the g-th row and the j-th column of the table tells how many reads have been mapped to gene g in sample j.

As a running example, we will use the gene level read counts from the *pasilla* data package. This data set is from an experiment on *Drosophila melanogaster* cell cultures and investigated the effect of RNAi knock-down of the splicing factor *pasilla* (Brooks et al. 2011). The detailed transcript of how we produced the pasilla count table from second generation sequencing (Illumina) FASTQ files is provided in the vignette of the data package *pasilla*. The table is supplied by the *pasilla* package as a text file of tab-separated values. The function `system.file` tells us where this file is stored on your computer.

```
datafile = system.file("extdata/pasilla_gene_counts.tsv", package = "pasilla")
```

To read this file with *R*, we use the function `read.table`.

```
rawCountTable = read.table(datafile, header = TRUE, row.names = 1)
```

Here, `header = TRUE` indicates that the first line contains column names and `row.names = 1` means that the first column should be used as row names. This leaves us with a `data.frame` containing integer count values.

Looking at the `rawCountTable` to see how it is formatted.

```
head(rawCountTable)

            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
```

Creating description factors for samples and rename the `rawCountTable` columns.

```
condition = c("control", "control", "control", "control", "treated", "treated", "treated")
libType = c("single-end", "single-end", "paired-end", "paired-end", "single-end",
            "paired-end", "paired-end")

colnames(rawCountTable)[1:4] = paste0("control", 1:4)
```

Now, we have data input as follows.

```
head(rawCountTable)

            control1 control2 control3 control4 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
```

4 Data exploration and quality assessment

Data quality assessment (QA) and exploration are essential steps of any data analysis. These steps should typically be performed very early in the analysis of a new data set, preceding or in parallel to the normalization step and differential expression testing.

Our purpose is the detection of differentially expressed genes, and, in particular, we are looking for samples whose experimental treatment suffered from an anormality that renders the data points obtained from these particular samples detrimental to our purpose.

The QA processes run on the raw data might reveal technical issues, but other biases might still be present in your data and the best (only?) way to control for those is visual. For example, one could compare replicates (in which case, biological replicates are best), by: looking differences in count distribution between samples, plotting a scatterplot of replicates, sample clustering, . . .

4.1 Data transformation

For data exploration and visualisation, it is useful to work with transformed versions of the count data. As the count values distribution is highly skewed, the $\log_2$ transformation helps to approximately normalize the distributions.

```
ggplot(rawCountTable, aes(x = control1)) + geom_histogram(fill = "#525252", binwidth = 2000)
```

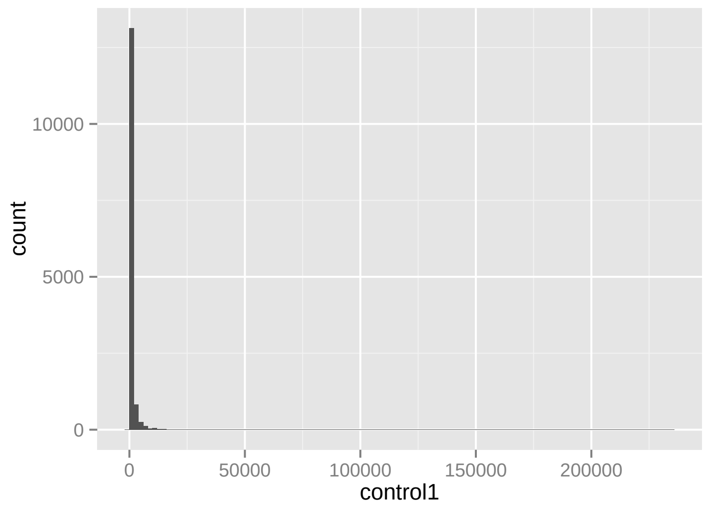


Figure 5: Histogram of `control1` sample from raw counts data.

Log base 2 is typically used as it facilitates the conversion back to the original scale: a difference of 1 on the log base 2 scale corresponds to a fold change of 2 on the original count scale. Since count values for a gene can be zero in some conditions (and non-zero in others), we advocates the use of pseudocounts, i.e. transformations of the form

$$y = \log_2(K + 1) \quad \text{or more generally,} \quad y = \log_2(K + k_0);$$

where K represents the count values and k_0 is a positive constant.

```
pseudoCount = log2(rawCountTable + 1)

ggplot(pseudoCount, aes(x = control1)) + ylab(expression(log[2](count + 1))) +
  geom_histogram(colour = "white", fill = "#525252", binwidth = 0.6)
```

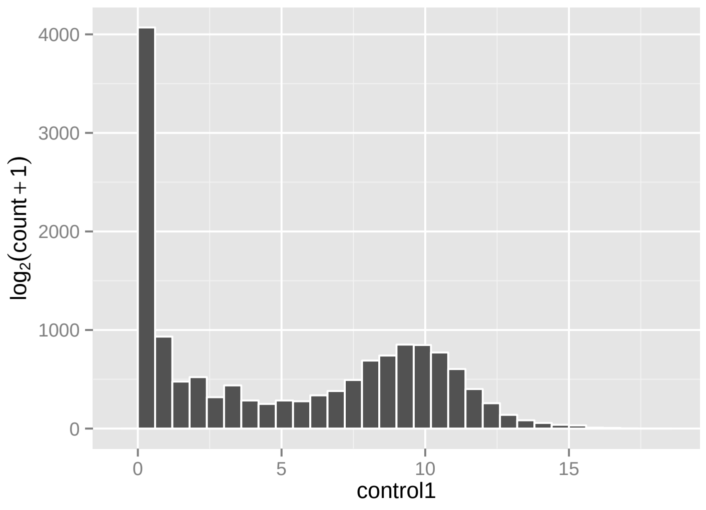


Figure 6: Histogram of `control1` sample from pseudocounts data.

Even more, 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 observable quantity (i.e., here, the expression strength of a gene) does not depend on the mean. In RNA-Seq data, however, variance grows with the mean, with larger variances for larger counts. For example, if one performs PCA 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 strategy to avoid this is to take the logarithm of the "normalized" count values plus a small constant; 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.

In order to make counts approximately homoskedastic, the packages *DESeq*, *DESeq2* and *edgeR* offers functions to transform the data:

```
varianceStabilizingTransformation(...){DESeq, DESeq2}
rlog(...){DESeq2}
cpm(..., prior.count = 2, log = TRUE){edgeR}
```

Note that these transformations are provided for applications other than differential testing.

For differential testing to operate on raw counts and to use discrete distributions.

4.2 Between-sample distribution

melt(...){*reshape*}

ggplot(...){*ggplot2*}

Visualizing the between-sample distribution of counts is useful to contraste the distribution of gene-level expression values on different samples. It can for example be used to display the effects of between-samples before and after filtering and/or normalization.

4.2.1 Boxplots

The boxplot method provides an easy way to visualize the distribution of pseudocounts in each sample. In the boxplot display, a box is formed with sides at the 25-th and 75-th percentiles of the distribution. A line is also drawn within the box at the level of the median. Whiskers are also drawn extending beyond each end of the box with points beyond the whiskers typically indicating outliers.

```
df = melt(pseudoCount, variable_name = "Samples")
df = data.frame(df, Condition = substr(df$Samples, 1, 7))

ggplot(df, aes(x = Samples, y = value, fill = Condition)) + geom_boxplot() + xlab("") +
  ylab(expression(log[2](count + 1))) + scale_fill_manual(values = c("#619CFF", "#F564E3"))
```

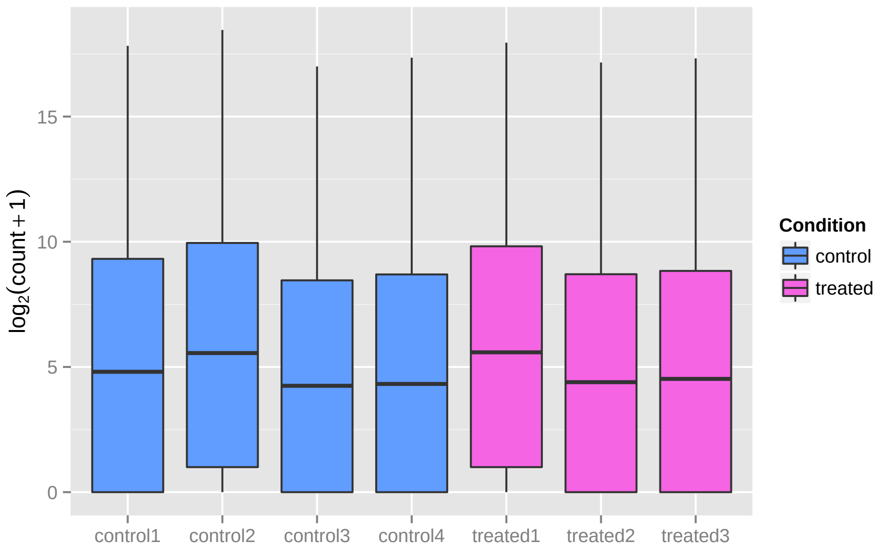


Figure 7: Parallel boxplots from pseudocounts data.

4.2.2 Histograms and density plots

Pseudocounts distributions can also be summarized by means of a histogram or density plot. Histograms provide more detail by enabling, for example, the detection of a secondary mode in the distribution.

```
ggplot(df, aes(x = value, colour = Samples, fill = Samples)) + ylim(c(0, 0.25)) +
  geom_density(alpha = 0.2, size = 1.25) + facet_wrap(~ Condition) +
  theme(legend.position = "top") + xlab(expression(log[2](count + 1)))
```

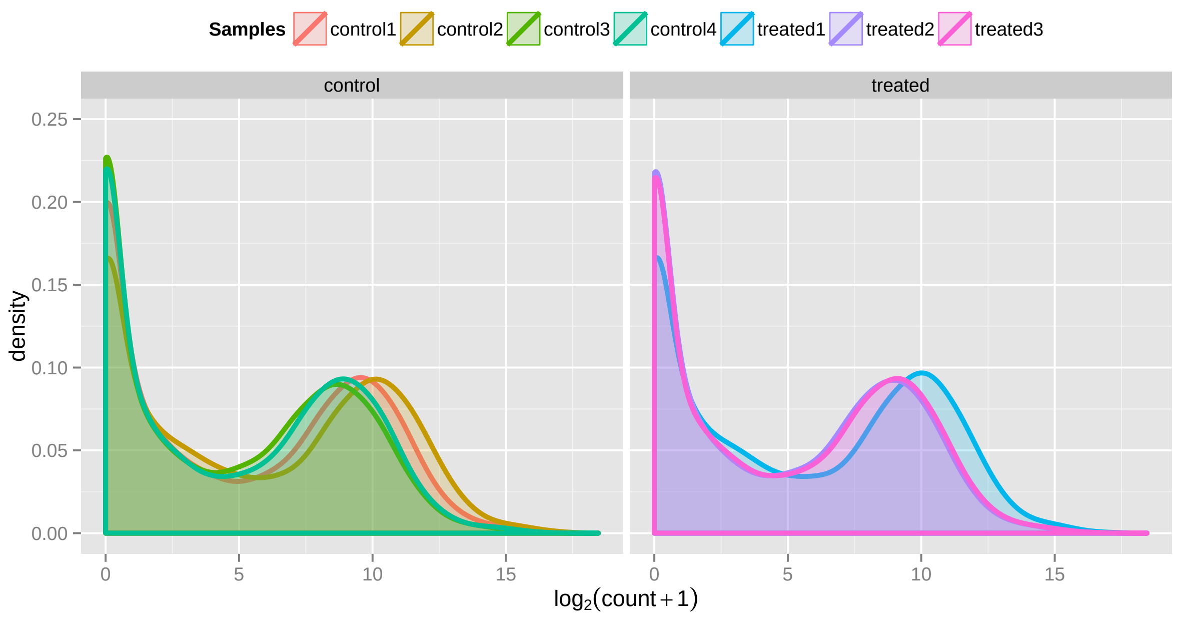


Figure 8: Densities plot displays smoothed empirical densities for the individual samples in each condition.

Figure 8 shows the densities of speudocounts for the samples displayed in Figure 7. As there is evidence of bi-modality, these displays provide additional information what is not captured by boxplots.

4.3 MA-plot between samples

An MA-plot is a plot of log-fold change (M-values, i.e. the log of the ratio of level counts for each gene between two samples) against the log-average (A-values, i.e. the average level counts for each gene across the two samples).

The MA-plot is a useful to visualize reproducibility between samples of an experiment. From a MA-plot one can see if normalization is needed.

In MA plot, genes with similar expression levels in two samples will appear around the horizontal line $y = 0$. A lowess fit (in red) is plotted underlying a possible trend in the bias related to the mean expression. Below we show the code to produce a simple MA-plot (e.g. between control 1 and control 2 samples).

```
x = pseudoCount[, 1]
y = pseudoCount[, 2]

## M-values
M = x - y

## A-values
A = (x + y)/2

df = data.frame(A, M)

ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
  geom_hline(color = "blue3") + stat_smooth(se = FALSE, method = "loess", color = "red3")
```

MA-plot between treated samples.

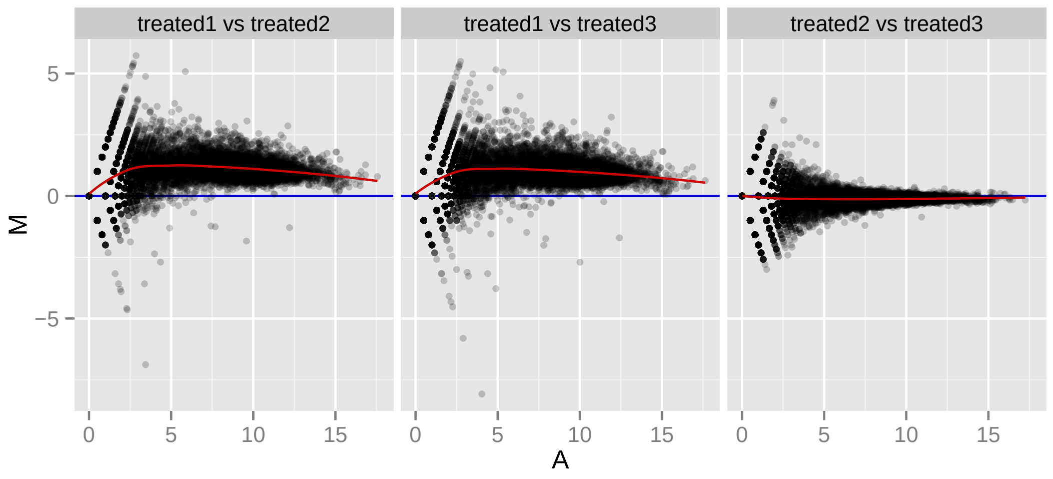


Figure 9: MA plot between treated samples

MA-plot between control samples.

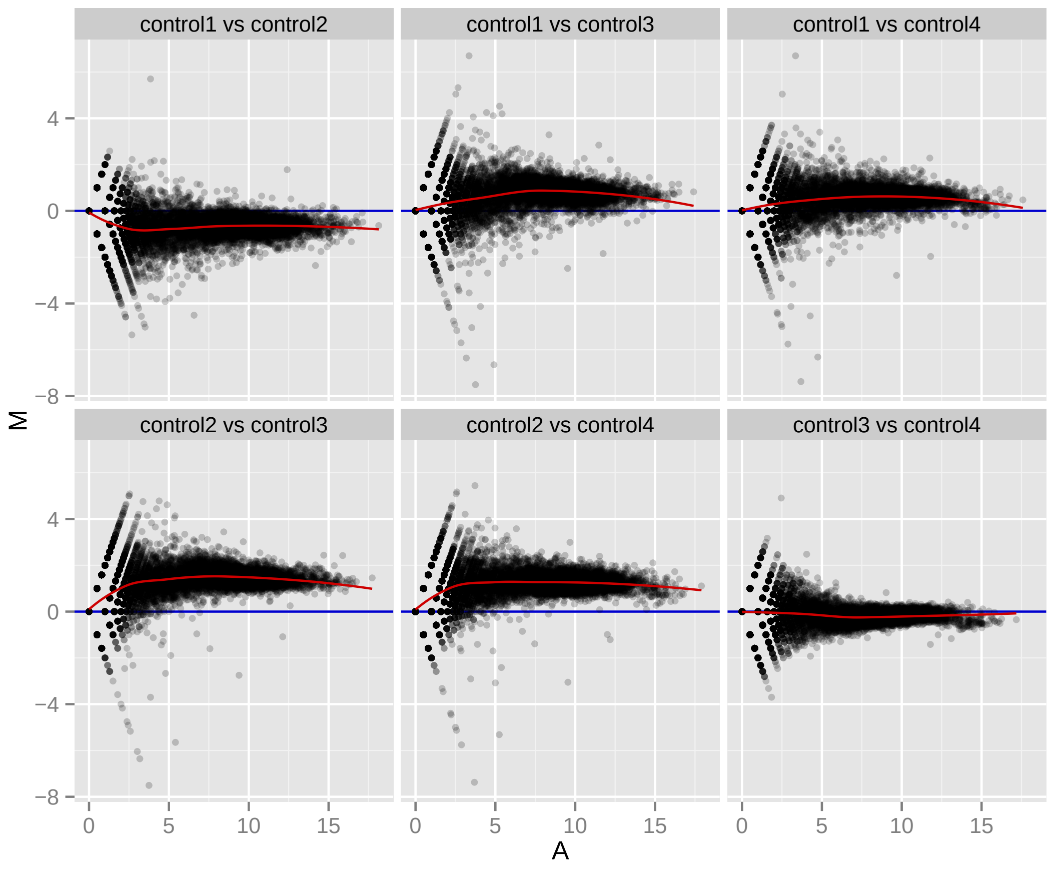


Figure 10: MA plot between control samples

4.4 Clustering of the sample-to-sample distances

`cim(...)`{*mixOmics*}

`heatmap(...)`{*stats*}

To explore the similarities and dissimilarities between samples, it is often instructive to look a clustering image map (CIM) or heatmap of sample-to-sample distance matrix.

A heatmap is a two-dimensional, rectangular, colored grid. It displays data that themselves come in the form of a rectangular matrix:

– the color of each rectangle is determined by the value of the corresponding entry in the matrix,

– the rows and columns of the matrix are rearranged independently according to some hierarchical clustering method, so that similar rows and columns are placed next to each other, respectively.

Below we show how to produce such a CIM from the pseudocounts data.

```
mat.dist = pseudoCount
colnames(mat.dist) = paste(colnames(mat.dist), libType, sep = " : ")
mat.dist = as.matrix(dist(t(mat.dist)))
mat.dist = mat.dist/max(mat.dist)

hmcol = colorRampPalette(brewer.pal(9, "GnBu"))(16)
cim(mat.dist, col = rev(hmcol), symkey = FALSE, margins = c(9, 9))
```

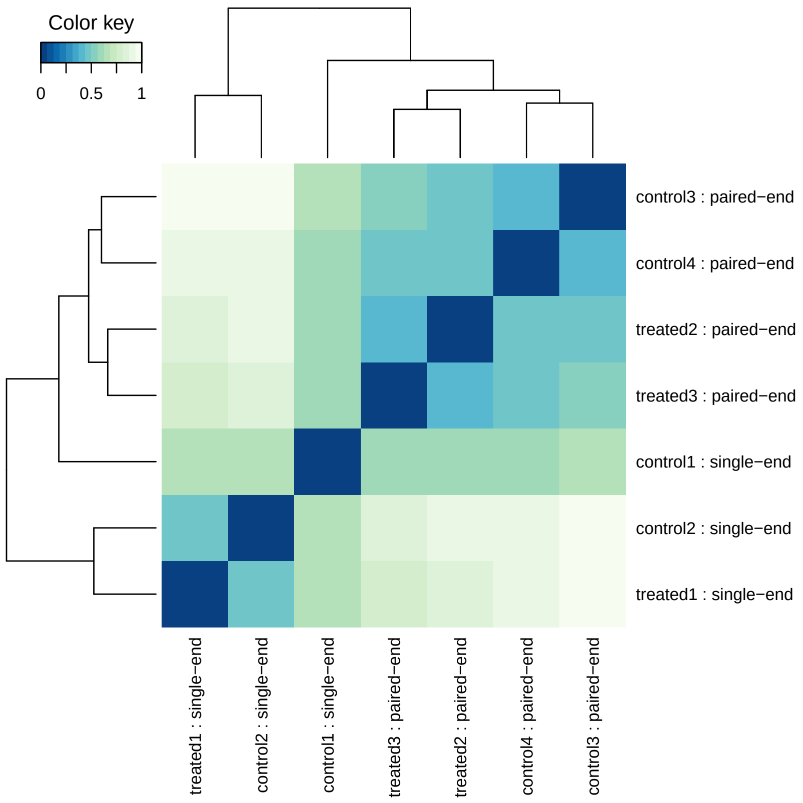


Figure 11: Heatmap showing the Euclidean distances between the samples as calculated from the pseudocount data.

4.5 Principal component plot of the samples

plotPCA(...){*DESeq, DESeq2*}

This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects. The plot is produced from the output of Principal Componenet Analysis (PCA) on the transposed of the given counts matrix.

PCA is used to reduce multidimensional datasets to lower dimensions for analysis; it is a technique that can determine the key features of high-dimensional datasets. In the context of RNA-Seq analysis, PCA essentially clusters samples by groups of the most significantly dysregulated genes. Clustering first by the most significant group, then by progressively less significant groups.

Given the experimental design of the dataset that we are attempting to analyse here (i.e., samples belong to only two distinct groups), a control group and a treatment group, there should be a clear separation of both groups of samples by the first component.

PCA is performed on the top genes selected by highest row variance. Suppose `ntop` indicates how many of the most variable genes should be used for calculating the PCA, then

```
rv = rowVars(pseudoCount)
select = order(rv, decreasing = TRUE)[1:ntop]
pca = prcomp(t(pseudoCount[select, ]))
```

The `plotPCA` is a simple helper function that perform the PCA and makes the plot

```
annot = AnnotatedDataFrame(data = data.frame(condition, libType,
                                              row.names = colnames(pseudoCount)))
expSet = new("ExpressionSet", exprs = as.matrix(pseudoCount), phenoData = annot)

plotPCA(expSet, intgroup = c("condition", "libType"))
```

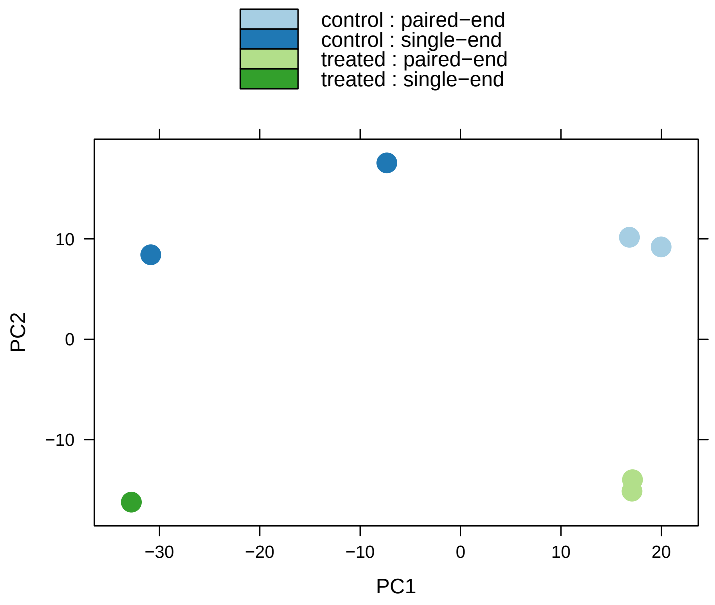


Figure 12: PCA plot. The 7 samples shown in the 2D plane spanned by their first two principal components. For this data set, no batch effects besides the known effects of *condition* and *libType* are discernible.

In this plot, samples extracted by single-end and paired-end methods tend to cluster together, suggesting a batch effect.

4.6 Multi-dimensional scaling plot

plotMDS(...){*edgeR*}

The purpose of multidimensional scaling (MDS) is to provide a visual representation of the pattern of proximities (i.e., similarities or distances) among a set of objects. MDS takes a set of similarities and returns a set of points such that the distances between the points are approximately equal to the similarities.

The similarities between each pair of samples (columns) is the root-mean-square deviation (Euclidean distance) for the top genes. Distances on the plot can be interpreted as leading $\log_2$-fold-change, meaning the typical (root-mean-square) $\log_2$-fold-change between the samples for the genes that distinguish those samples.

The function `plotMDS` has an argument `gene.selection` which, set to `"common"`, chooses as top genes those with the largest root-mean-square deviations between samples. Note `D[i, j]` the deviation between the samples `i` and `j`, and `top` how many of the most distinguished genes should be used in calculating the MDS, then

```
x = pseudoCount
s = rowMeans((x - rowMeans(x))^2)
o = order(s, decreasing = TRUE)
x = x[o, ]
x = x[1:top, ]
D[i, j] = sqrt(colMeans((x[, i] - x[, j])^2))
```

If `gene.selection` is `"pairwise"`, then a different set of top genes is selected for each pair of samples.

```
x = pseudoCount
D[i, j] = sqrt(mean(sort((x[, i] - x[, j])^2, decreasing = TRUE)[1:top]))
```

The `plotMDS` is a simple helper function that performs the MDS and makes the plot

```
fac = factor(paste(condition, libType, sep = " : "))
colours = brewer.pal(nlevels(fac), "Paired")

plotMDS(pseudoCount, col = colours[as.numeric(fac)], labels = fac)
```

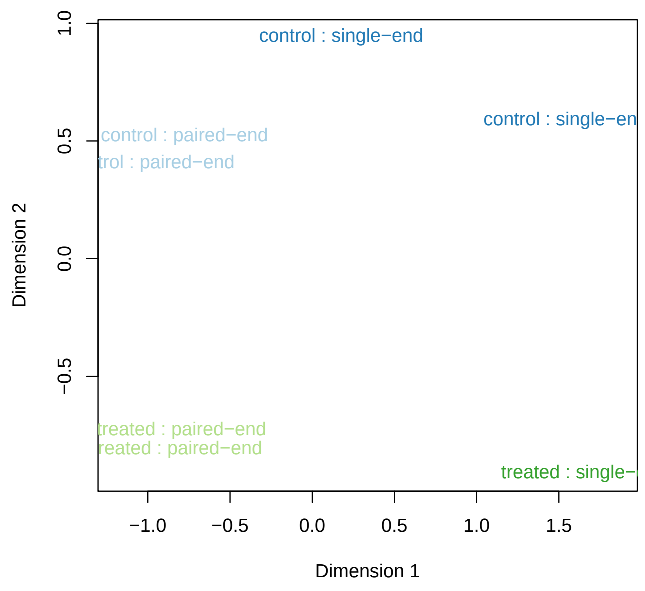


Figure 13: MDS plot. Samples from each *libType* tend to cluster together, suggesting a bacth effet.

5 Raw data filtering

For downstream analysis, it is usually worthwhile to remove genes that appear to be very lowly expressed in any of the experimental conditions. This is called filtering.

There are a number of ways that filtering can be done. One way is to filter genes with a total read count smaller than a given threshold (Sultan et al. 2008) and filter genes with at least one zero count in each experimental condition (Bottomly et al. 2011); however, selecting an arbitrary threshold value to filter genes in this way does not account for the overall sequencing depth.

To account differences in library size, filter has also been proposed based on counts per million (CPM) (Robinson, McCarthy, and Smyth 2010), calculated as the raw counts divided by the library sizes and multiplied by one million. Genes with a CPM value less than a given cutoff (e.g. 1 or 100) in more samples (ignoring condition labels) than the size of the smallest group are subsequently filtered from the analysis. In "Improving test results" (Section 12), we present other procedures for filter read counts arising from replicated data.

We filter out unexpressed genes, keeping genes that are expressed in at least one sample.

```
keep = rowSums(count) > 0
filtCount = pseudoCount[keep, ]
dim(count)

[1] 14599     7

dim(filtCount)

[1] 12359     7
```

This reduces the dataset to 12359 genes.

```
df = melt(filtCount, variable_name = "Samples")
df = data.frame(df, Condition = substr(df$Samples, 1, 7))

ggplot(df, aes(x = value, colour = Samples, fill = Samples)) +
  geom_density(alpha = 0.2, size = 1.25) + facet_wrap(~ Condition) +
  theme(legend.position = "top") + xlab("pseudocounts")
```

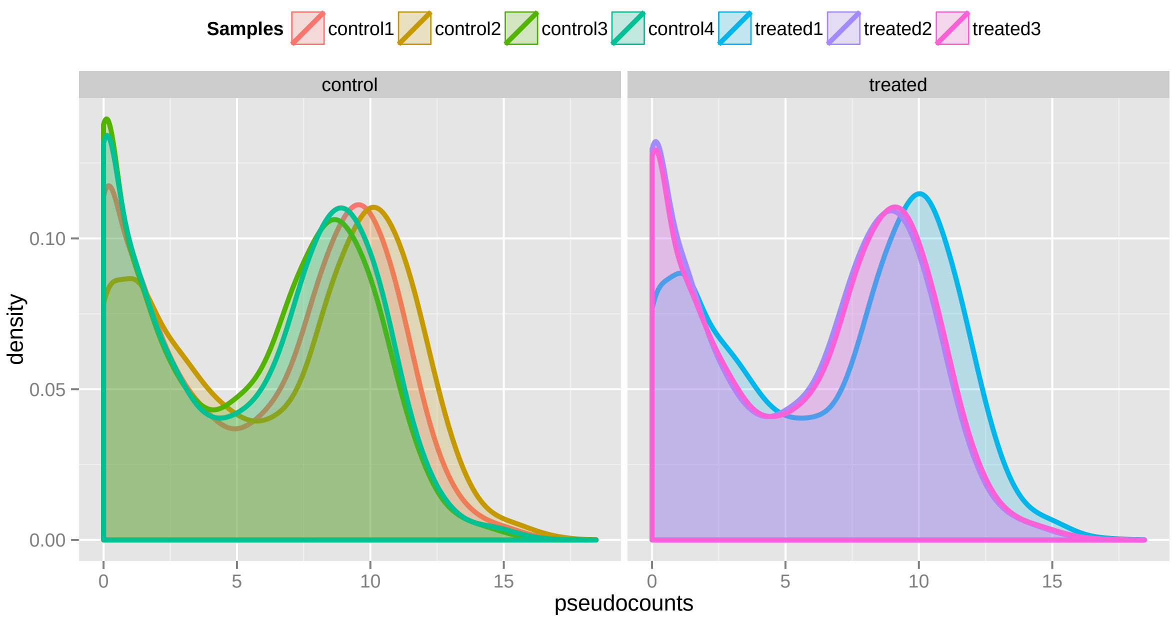


Figure 14: Densities plot for the individual samples in each condition after filtering.

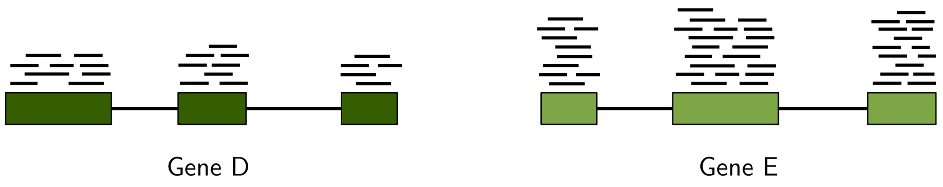
6 Interpreting read counts

For almost all experiments, biological replicates of each condition must be used. Resulting count files therefore will look something like this, where the number of reads aligning to each annotated gene are aggregated:

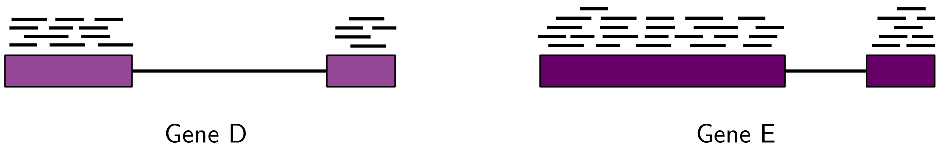
	Sample 1	Sample 2	Sample 3
Gene A	5	3	8
Gene B	17	23	42
Gene C	10	13	27
Gene D	752	615	1203
Gene E	1507	1225	2455

- In the above example we can see that Gene E has about twice as many reads aligned to it as Gene D. This could mean:

  1) Gene E is expressed with twice as many transcripts as Gene D



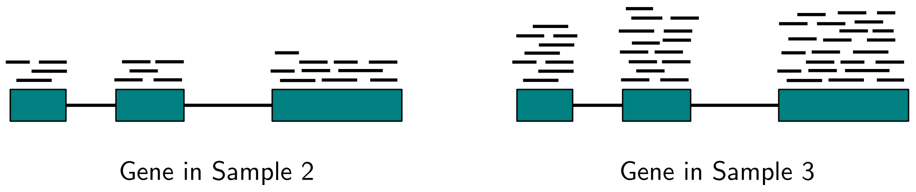
  2) Both genes are expressed with the same number of transcripts but Gene E is twice as long as Gene D and produces twice as many fragments. At same expression level, a long transcript will have more reads than a shorter transcript (Oshlack and Wakefield 2009; Bullard et al. 2010).



  As a consequence: number of reads $\neq$ expression level.

- Also we can see that genes in the sample 3 has about twice as many reads aligned to it as sample 2.

  3) Fluctuations in sequencing depth



The difference in counts between genes are likely to have been caused by some combination of all three reasons. When comparing the same gene across two conditions, reasons 1) and 2) would be expected to have the same effects on the same gene across the two conditions.

In the rest of this tutorial, we will concentrate on measuring a single gene across two conditions, rather than comparing different genes in the same condition.

7 Normalization

Normalization is a process designed to identify and remove systematic technical differences between samples that occur in the data to ensure that technical bias has minimal impact on the results. The most common symptom of the need for normalization is differences in the total number of aligned reads.

The overall strategy with normalization is to choose an appropriate baseline, and express sample counts relative to that baseline. There are several approaches to choice of appropriate baseline. Below we will describe the approaches implemented in *DESeq*, *DESeq2* and *edgeR* packages.

At lot of different normalization methods exist, but these will not be considered in this tutorial:

- some are part of models for differential expression, others are "stand-alone"
- they do not rely on similar hypotheses
- but all of them claim to remove technical bias associated with RNA-seq data

Which one is the best? This question is still an open issue ...

- How to choose a normalization adapted to our experiment and which criteria can help make this choice?
- What is the impact of the normalization step on lists of differential expression genes?

Global normalization methods

All the normalization methods considered here are global procedures, in the sense that only a single factor C_j is used to scale the counts for each sample j.

	control				treated		
Gene 1	5	1	0	0	**4**	**0**	**0**
Gene 2	0	2	1	2	**1**	**0**	**0**
Gene 3	92	161	76	70	**140**	**88**	**70**
:	:				:		
:	:				:		
:	:				:		
Gene G	15	25	9	5	**20**	**14**	**17**

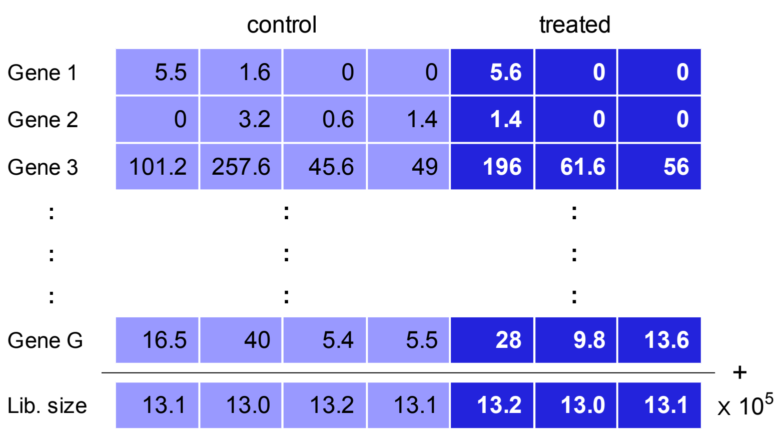
Correction multiplicative factor:

C_j	1.1	1.6	0.6	0.7	**1.4**	**0.7**	**0.8**

Column multiplication by factor C_j:

Gene 3	92	161	76	70	**140**	**88**	**70**	
C_j	1.1	1.6	0.6	0.7	**1.4**	**0.7**	**0.8**	x
Gene 3	101.2	257.6	45.6	49	**196**	**61.6**	**56**	

The purpose of the size factors C_j is to render comparable the counts of different samples, even if these samples have been sequenced with different depths.



	control				treated		
Gene 1	5.5	1.6	0	0	5.6	0	0
Gene 2	0	3.2	0.6	1.4	1.4	0	0
Gene 3	101.2	257.6	45.6	49	196	61.6	56
:		:				:	
:		:				:	
:		:				:	
Gene G	16.5	40	5.4	5.5	28	9.8	13.6
Lib. size	13.1	13.0	13.2	13.1	13.2	13.0	13.1

Normalized library sizes are roughly equal.

Framework

- K_{gj} : observed count for gene g in sample (library) j
- G : number of genes
- $D_j = \sum_{g=1}^{G} K_{gj}$: total number of reads for sample j
- N : number of samples in the experiment
- C_j : normalization factor associated with sample j

7.1 Total read count normalization (Robinson, McCarthy, and Smyth 2010)

```
cpm(..., normalized.lib.sizes = TRUE){edgeR}
```

Motivation – scaling to library size as a form of normalization makes intuitive sense, given it is expected that sequencing a sample to half the depth will give, on average, half the number of reads mapping to each gene.

Assumption – read counts are proportional to expression level and sequencing depth.

Method – divide transcript read count by total number of reads and rescale the factors to counts per million

$$C_j = \frac{10^6}{D_j}$$

Disadvantage – if a few number of genes are unique to, or highly expressed in, one experimental condition, the sequencing "real estate" available for the remaining genes in that sample is decreased. If not adjusted for, this sampling artifact can force the differential expression analysis to be skewed towards one experimental condition.

Example – suppose that genes in the Sample 2 has about twice as many reads aligned to it as Sample 1. Then, the total read count normalization would adjust Sample 1 by a facteur of 2 (see Figure 15). Now suppose that Sample 2 contains a small set of highly expressed genes. Under this scenario, Figure 16 shows the deficiency of this approach, the small fraction of highly expressed genes will skew the counts of lowly expressed genes.

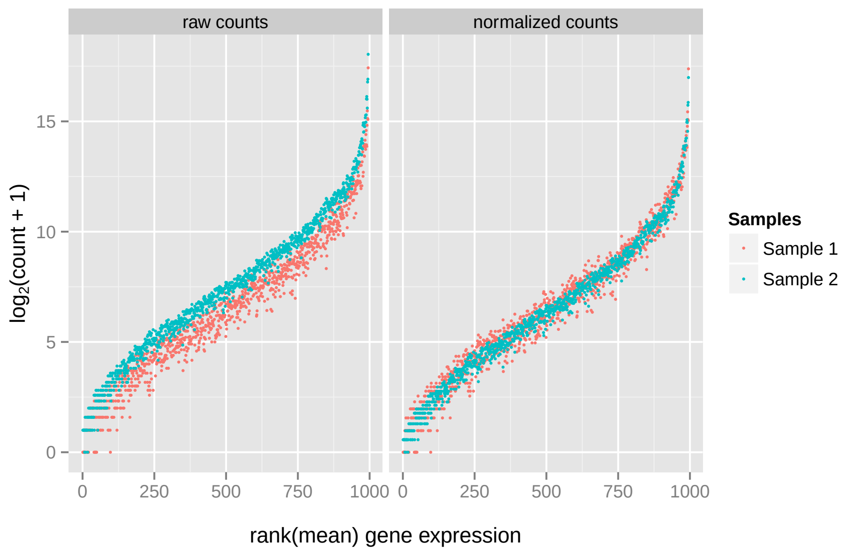


Figure 15: Relationship between counts before and after total read count normalization.

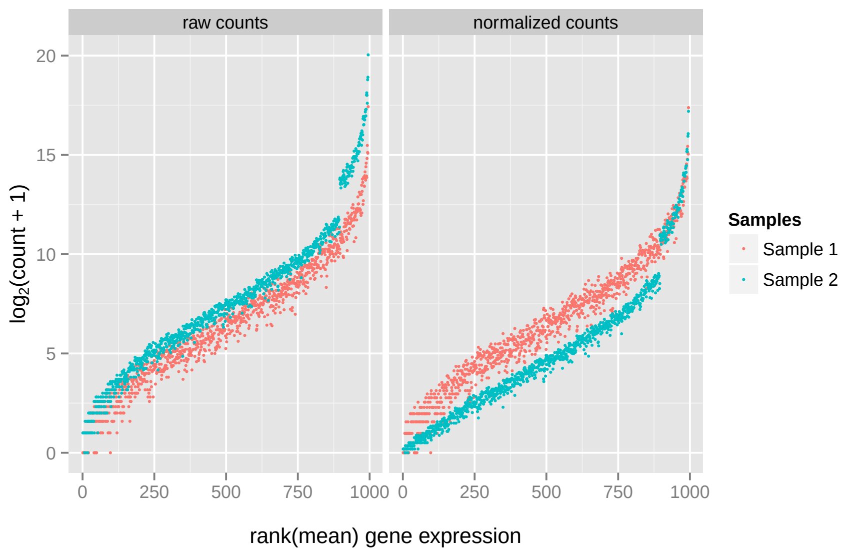


Figure 16: Relationship between counts before and after normalization as in Figure 15 but with 100 highly expressed genes.

7.2 Upper quantile normalization (Bullard et al. 2010)

```
calcNormFactors(..., method = "upperquartile", p = 0.75){edgeR}
```

Motivation – in analogy with standard techniques for normalizing microarray data, the proposal is to match the between-sample distributions of gene counts in terms of parameters such as quantiles. For instance, one could simply scale counts within samples by their median. Due to the preponderance of zero and low-count genes, the median is uninformative for the different levels of sequencing effort. Instead, it is advised to use the per-sample upper-quartile (75-th percentile).

Method –

- compute the upper quantile (p-th percentile) of sample j, $Q_j^{(p)}$, after of library-size scaling the reads count
- re-scale the upper quantile by the total reads count, $D_j Q_j^{(p)}$
- correction multiplicative factor

$$C_j = \frac{1}{D_j Q_j^{(p)}}$$

- factors should multiple to one

$$C_j = \frac{\exp\left\{\frac{1}{N}\sum_{l=1}^{N}\log\left(D_l Q_l^{(p)}\right)\right\}}{D_j Q_j^{(p)}}$$

- $p = 0.75$ upper quartile normalization

Example – consider the hypothetical example in Section 7.1. Figure 17 shows that upper quantile normalization is robust against highly expressed genes set in samples.

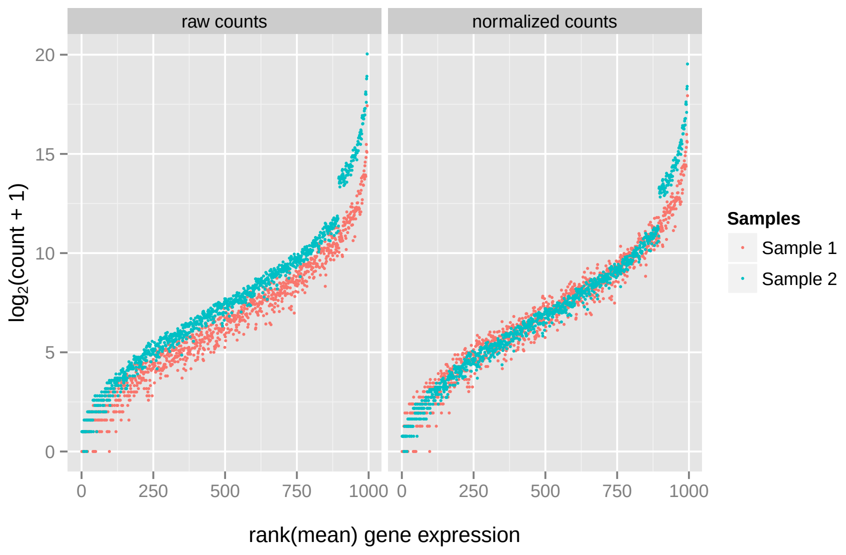


Figure 17: Relationship between counts before and after upper quantile normalization for samples as in Figure 16.

7.3 Relative Log Expression (RLE) (Anders and Huber 2010)

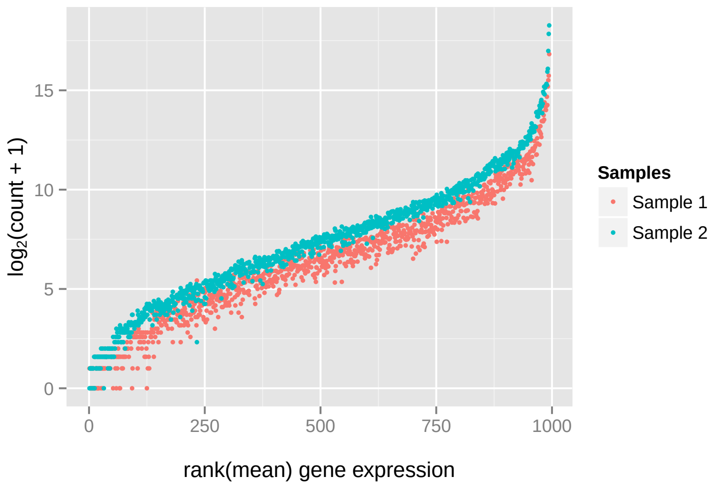
```
calcNormFactors(..., method = "RLE"){edgeR}
estimateSizeFactors(...){DESeq, DESeq2}
```

Motivation – the ratios $K_{gj}/K_{gj'}$ of expected counts for the same gene g in different samples j and j' should be equal to the size ratio $C_j/C_{j'}$ if gene g is not differentially expressed or samples j and j' are replicates. The total number of reads, D_j, may seem to be a reasonable choice for C_j. However, a few highly and differentially expressed genes may have strong influence on the total read counts, causing the ratio of total read counts not to be a good estimate for the ratio of expected counts.

Assumption – read counts are proportional to expression level and sequencing depth.

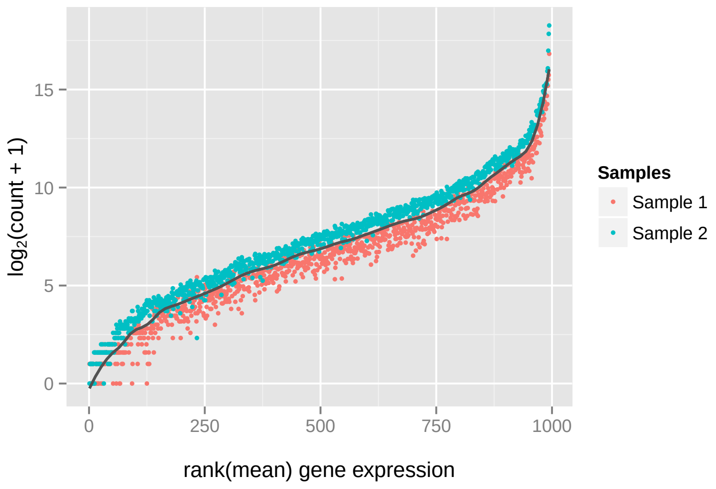
Method – each size factor estimate C_j is computed as the median of the ratios of the j-th sample's counts to those of a pseudo-reference sample – the geometric mean across samples:

- counts K_{gj} representation



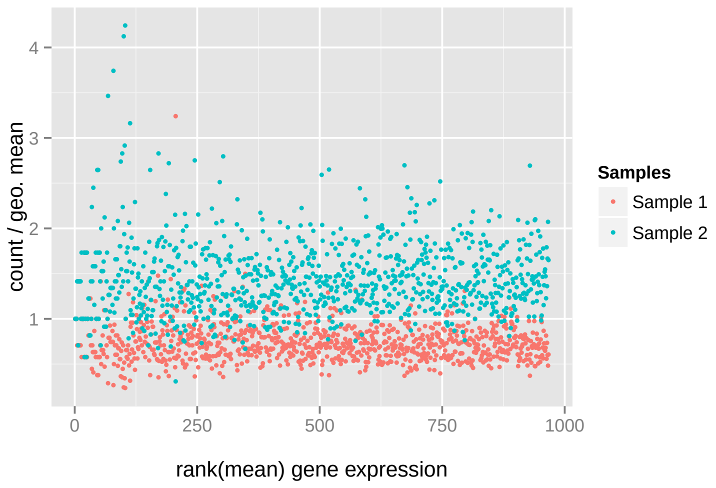
- calculation of geometric mean across samples, the pseudo-reference sample

$$\left(\prod_{l=1}^{N} K_{gl}\right)^{1/N}$$



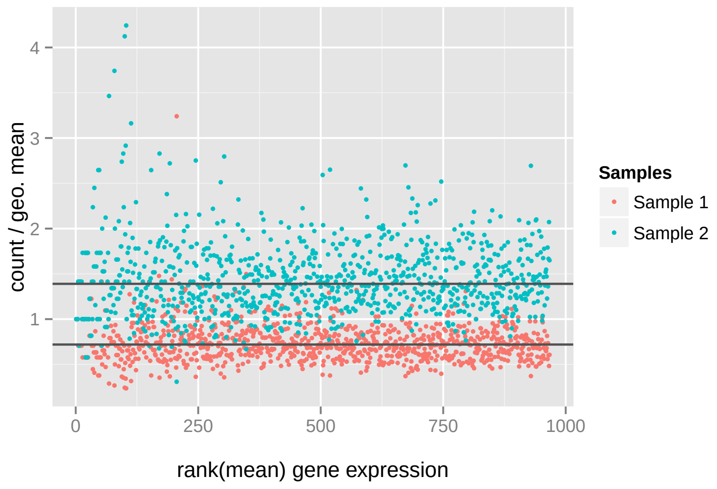
- centering samples with rapport to the pseudo-reference sample

$$\frac{K_{gj}}{\left(\prod_{l=1}^{N} K_{gl}\right)^{1/N}}$$



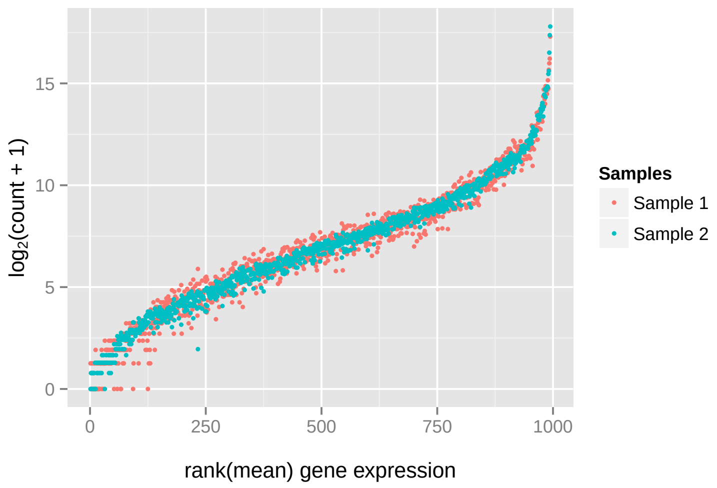
- correction factor, median across genes for each sample

$$C_j = \underset{g}{\text{median}} \left\{ \frac{K_{gj}}{\left(\prod_{l=1}^{N} K_{gl} \right)^{1/N}} \right\}$$



- factors should multiple to one $C_j = \frac{\exp\left\{ \frac{1}{N} \sum_{l=1}^{N} \log(C_l) \right\}}{C_j}$

- normalized counts



7.4 Trimmed Mean of M-values (TMM) (Robinson and Oshlack 2010)

```
calcNormFactors(..., method = "TMM"){edgeR}
```

Motivation – total read count is strongly dependent on a few highly expressed transcripts

Assumption – the majority of genes are not differentially expressed

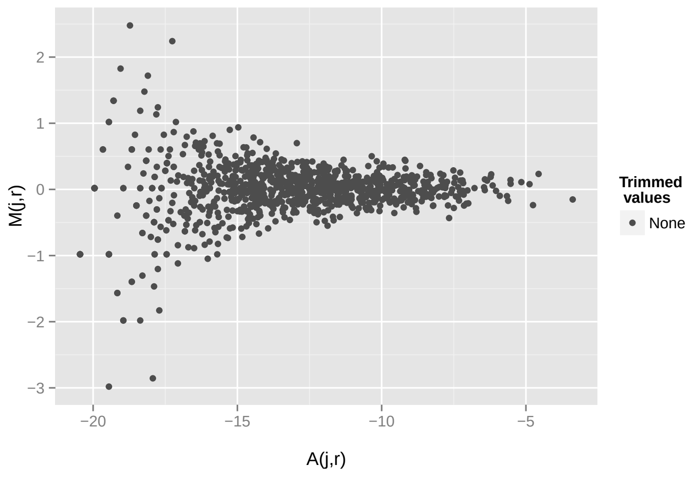
Method – a trimmed mean is the average after removing the upper and lower x% of the data. The TMM procedure is doubly trimmed, by log-fold-changes $M_g(j,r)$ (sample j relative to sample r for gene g)

$$M_g(j,r) = \log_2(K_{gj}/D_j) - \log_2(K_{gr}/D_r)$$

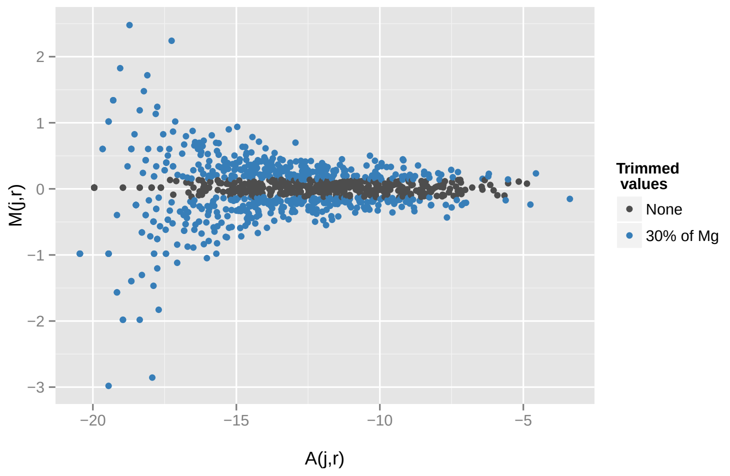
and by absolute intensity $A_g(k,r)$

$$A_g(j,r) = \frac{1}{2} \left(\log_2(K_{gj}/D_j) + \log_2(K_{gr}/D_r) \right)$$

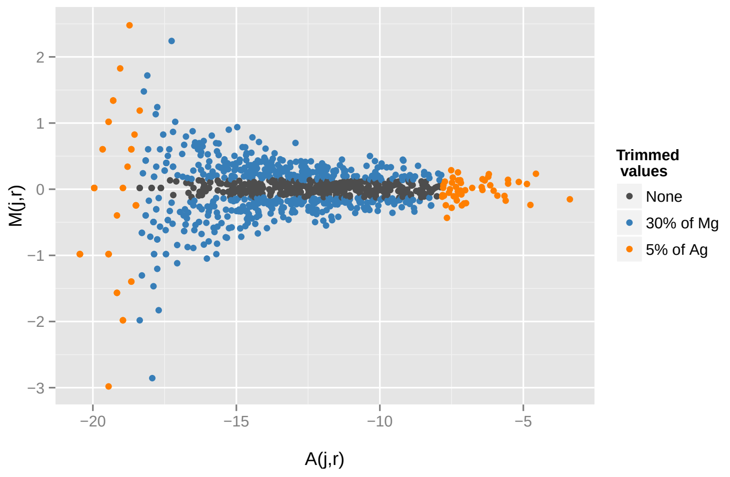
- $M(j,r)$ vs $A(j,r)$ plot



- trim the 30% more extreme $M_g(j,r)$ values



- trim the 5% more extreme $A_g(j,r)$ values



By default, the method trim the M_g values by 30% and the A_g values by 5%, but these settings can be tailored to a given experiment.

- calculate a weighted mean of M_g

$$\mathsf{TMM}(j,r) = \frac{\sum_{g \in G^*} w_g(j,r) M_g(j,r)}{\sum_{g \in G^*} w_g(j,r)}$$

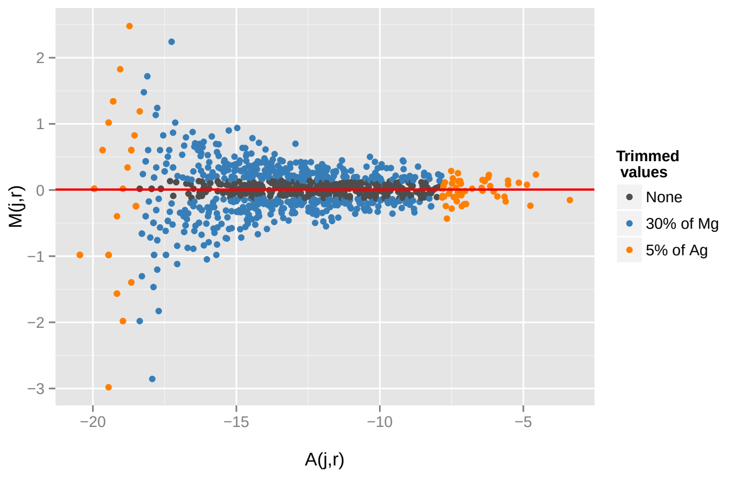
where G^* represents the set of genes with valid M_g and A_g values and not trimmed, using the percentages above.

As MA-plot above indicates, the variances of the M_g values at higher total count are lower. Within a library, any individual gene is binomial distributed with a given library size and proportion. Using the delta method (see Box 1), one can calculate an approximate variance for the M_g, and the inverse of these is used to weight the average.

$$w_g(j,r) = \left(\frac{D_j - K_{gj}}{D_j K_{gj}} + \frac{D_r - K_{gr}}{D_r K_{gr}} \right)^{-1}$$

- returning to the origin units we obtain the correction factor

$$C_j = 2^{\mathsf{TMM}(j,r)}$$



- factors should multiple to one $C_j = \dfrac{\exp\left\{ \frac{1}{N} \sum_{l=1}^{N} \log(C_l) \right\}}{C_j}$

Box 1: The delta method for variance aproximation.

The delta method

The "Delta Method" find approximations based on Taylor series expansions to the variance of functions of random variables.

Taylor Series Expansion: The Taylor series expansion of a function $f(\cdot)$ about a value a is given as:

$$f(x) = f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2!} + \cdots$$

where we can often drop the higher order terms to give the approximation:

$$f(x) \approx f(a) + f'(a)(x-a)$$

Letting $a = \mu_x$, the mean of a random variable X, a Taylor series expansion of $y = f(x)$ about μ_x gives the approximation:

$$y = f(x) \approx f(\mu_x) + f'(\mu_x)(x-\mu_x)$$

Taking the variance of both sides yields:

$$\mathsf{Var}(Y) = \mathsf{Var}(f(X)) \approx [f'(\mu_x)]^2 \mathsf{Var}(X)$$

So, if Y is any function of the random variable X, we need only calculate the variance of X and the first derivative of the function to approximate the variance of Y.

Example: Suppose $Y = \log_2(X/c)$, c a constant. Then $f(x) = \log_2(x/c)$ and $f'(x) = 1/(x\ln 2)$, so that:

$$\mathsf{Var}(Y) \approx \left[\frac{1}{\mu_x \ln 2}\right]^2 \mathsf{Var}(X)$$

Now suppose $X \sim \mathsf{Bin}(N,p)$. Then $\mu_x = Np$ and $\mathsf{Var}(X) = Np(1-p)$, so that:

$$\mathsf{Var}(\log_2(X/N)) \approx \left[\frac{1}{Np\ln 2}\right]^2 Np(1-p) = \frac{1}{\ln^2 2} \cdot \frac{1-p}{Np}$$

If $p = K/N$ for $0 < K < N$, then

$$\mathsf{Var}(\log_2(X/N)) \approx \frac{1}{\ln^2 2} \cdot \frac{N-K}{NK}$$

8 Hypothesis testing

Our focus:

- Based on a count table, we want to detect differentially expressed genes between different conditions.

  How can we detect genes for which the counts of reads change between conditions more systematically than as expected by chance?

- We would like to use statistical testing to decide whether, for a given gene, an observed difference in read counts is significant, that is, whether it is greater than what would be expected just due to natural random variation.

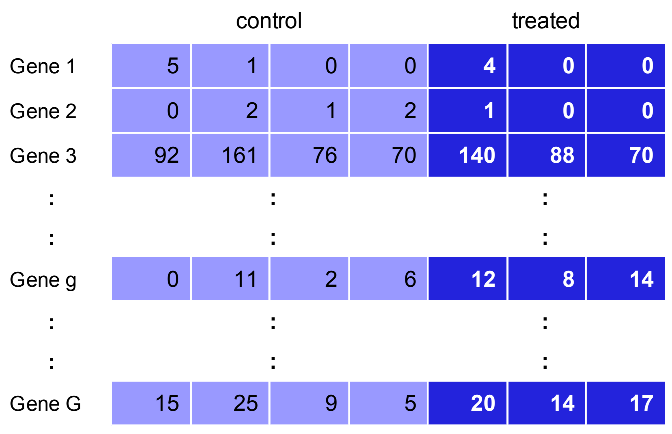
8.1 Testing

The idea of hypothesis testing is that:

- you formalize a hypothesis (H_1, the alternative hypotesis) into a statement such as:

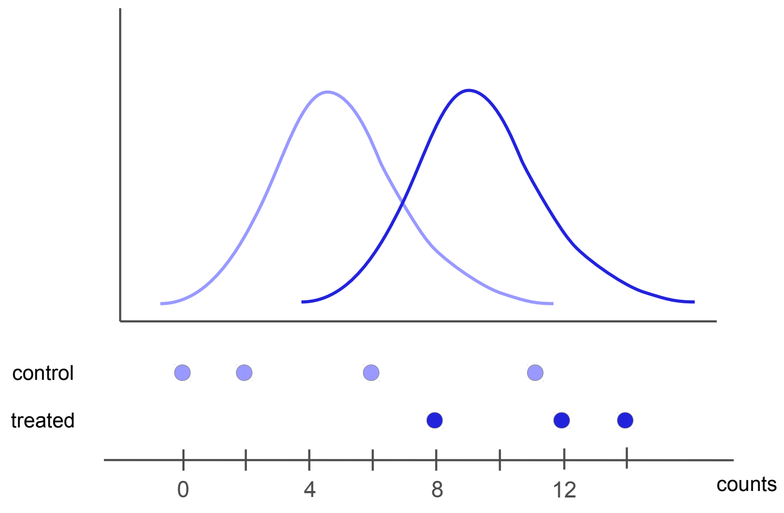
  the gene g is differentially expressed between the conditions

- collect appropriate data

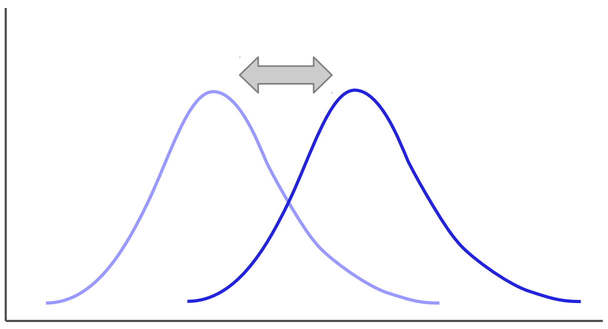


	control				treated		
Gene 1	5	1	0	0	4	0	0
Gene 2	0	2	1	2	1	0	0
Gene 3	92	161	76	70	140	88	70
:		:				:	
:		:				:	
Gene g	0	11	2	6	12	8	14
:		:				:	
:		:				:	
Gene G	15	25	9	5	20	14	17

- fit the model for each gene



- use statistics to quantify the difference



- then determine if the hypothesis is true or not.

How to quantify the difference?

The statistical tests do not give a simple answer of whether the hypothesis is true or not. What a statistical test determines is how likely that null hypothesis is to be true.

The null hypothesis is the hypothesis that nothing is going on (it is often labelled as H_0). So, the associated null hypothesis (H_0) is:

the gene g is not differentially expressed between the conditions.

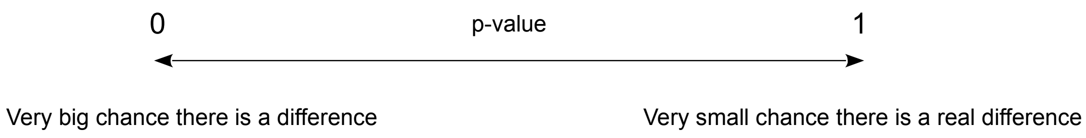
After a test statistic is computed, it is often converted to a "p-value". Then the difference is quantified in terms of the p-value. If the p-value is small then the null hypothesis is deemed to be untrue and it is rejected in favour of the alternative.

8.2 P-values

The p-value is the probability of seeing a result as extreme or more extreme than the observed data, when the null hypothesis is true.

It is not the probability that the null hypothesis is true.

Clearly the smaller the p-value the more confident we can be in the conclusions drawn from it. A p-value of 0.0001 indicates that if the null hypothesis is true the chance of seeing data as extreme or more extreme than that being tested is one in 10 000.


0
p-value
1
Very big chance there is a difference
Very small chance there is a real difference

It is a usual convention in biology to use a critical p-value of 0.05 (often called alpha, α). This means that the probability of observing data as extreme as this if the null hypothesis is true is 0.05 (5% or 1 in 20); in other words, it indicates that the null hypothesis is unlikely to be true.

There is nothing magical about p-value < 0.05, it is just a convention.

It is worth pointing out that if a p-value is less than 0.05 it does not prove that the null hypothesis is false, it just indicates that it is unlikely to be true. Indeed statistics can never prove anything, it can only suggest that a hypothesis is very likely to be true or untrue. It is also worth noting that a p-value above 0.05 certainly doesn't prove that the null hypothesis is true: it just indicates that there is not enough evidence to reject it.

Reporting p-values as a measure of evidence allows some flexibility in the interpretation of a statistical test by providing more information than a simple dichotomy of "significant" or "not significant" at a predefined level.

8.3 Acceptable errors

Errors are the inevitable consequence of results based on statistical test. Unfortunately, we only have a sample of all the individuals in a population and the statistical test only gives an indication of how likely it is that the null hypothesis is true based on the sample available. There are two ways of making the wrong inference from the test. These two types of error are usually called type I and type II errors.

In a type I error (or false-positive) the null hypothesis is really true (the gene is not differentially expressed) but the statistical test has led you to believe that it is false (there is a difference in expression). This type of error is potentially very dangerous, if a rejected hypothesis allows publication of a scientific finding, a type I error brings a "false discovery", and the risk of publication of a potentially misleading scientific result.

In a type II error (or false-negative) the null hypothesis is really false (the gene is differentially expressed) but the test has not picked up this difference. This type of error is less dangerous than the type I but should still be avoided if possible.

		Actual situation "truth"	
		H_0 true	**H_0 false**
Decision	**Do not reject H_0**	**Correct decision**	**Incorrect decision type II error**
	Reject H_0	**Incorrect decision type I error**	**Correct decision**

Table 1: $\alpha =$ P(type I error), $\beta =$ P(type II error) and power $= 1 - \beta$.

A statistical test is usually constructed to control the type I error probability, and we achieve a certain power (which is equal to one minus the type II error probability) that depends on the study design, sample size, and precision of the measurements.

9 Multiple testing

In hypothesis tests, researchers bounded the probability of making a type I error by α, an "acceptable" risk of type I errors, conventionally set at 0.05. Problems arise, however, when researchers do not perform a single hypothesis test but many of them, one for each gene. Because each test again has a probability of producing a type I error, by performing a large number of hypothesis tests a substantial number of false positives may accumulate. This problem is called the problem of multiple testing.

The key goal of multiple testing methods is to control, or at least to quantify, the flood of type I errors that arise when many hypothesis tests are performed simultaneously.

9.1 Why is multiple testing a problem?

Say you have a set of hypotheses that you wish to test simultaneously. The first idea that might come to mind is to test each hypothesis separately, using some level of significance. At first blush, this doesn't seem like a bad idea. However, consider a case where you have 20 hypotheses to test, and a significance level of 0.05. What's the probability of observing at least one significant result just due to chance?

$$\begin{aligned}
\text{P(at least one significant result)} &= 1 - \text{P(no significant results)} \\
&= 1 - (1 - 0.05)^{20} \\
&\approx 0.64
\end{aligned}$$

So, with 20 tests being considered, we have a 64% chance of observing at least one significant result, even if all of the tests are actually not significant. In genomics and other biology-related fields, it's not unusual for the number of simultaneous tests to be quite a bit larger than 20, and the probability of getting a significant result simply due to chance keeps going up.

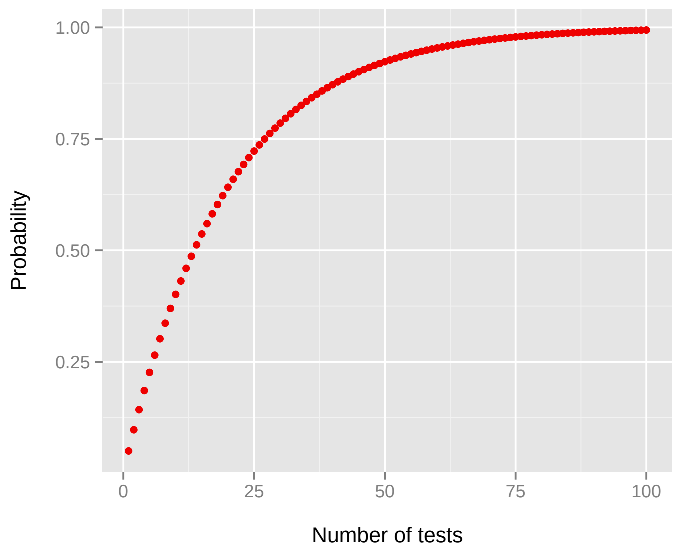
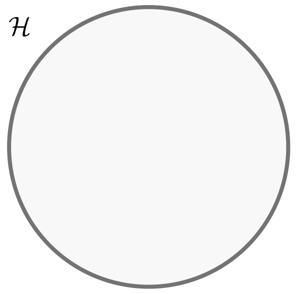


Figure 18: Probability of at least one significant result just due to chance for $\alpha = 0.05$.

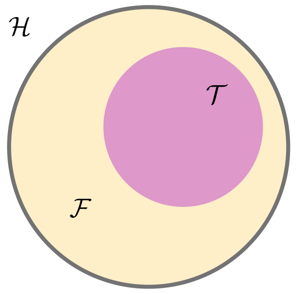
9.2 Controling multiple testing

The multiple testing problems have a simple structure:

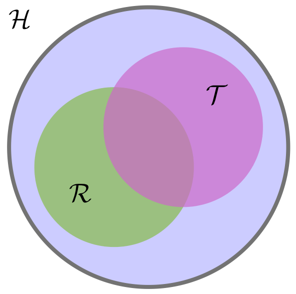
- We have a collection $\mathcal{H} = (H_1, \ldots, H_m)$ of null hypotheses, one for each gene, which we would like to reject.
- An unknown number m_0 of hypotheses in $\mathcal{H}$ is true, whereas the other $m_1 = m - m_0$ is false.



- We call the collection of true hypotheses $\mathcal{T} \subseteq \mathcal{H}$ and the remaining collection of false hypotheses $\mathcal{F} = \mathcal{H} - \mathcal{T}$.

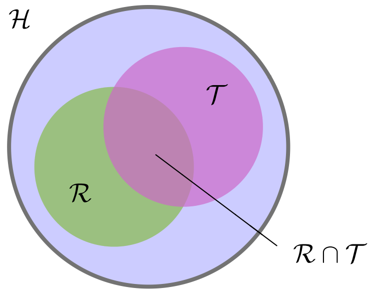


- **The goal of a multiple testing** procedure is to choose a collection $\mathcal{R} \subseteq \mathcal{H}$ of hypotheses to reject. Ideally, the set of rejected hypotheses $\mathcal{R}$ should coincide with the set $\mathcal{F}$ of false hypotheses as much as possible.

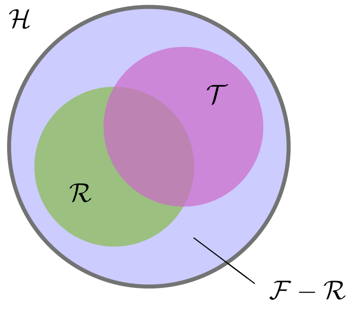


Two types of error can be made:

false positives, or type I errors, are rejections of hypotheses that are not false, that is, hypotheses in $\mathcal{R} \cap \mathcal{T}$;



false negatives, or type II errors, are failed rejections of false hypotheses, that is, hypotheses in $\mathcal{F} - \mathcal{R}$.



Type I and type II errors are in direct competition with each other, and a trade-off between the two must be made. If we reject more hypotheses, we typically have more type I errors but fewer type II errors. Multiple testing methods try to reject as many hypotheses as possible while keeping some measure of type I errors in check.

What does correcting for multiple testing mean? When people say "adjusting p-values for the number of hypothesis tests performed" what they mean is controlling a type I error measure.

This measure is usually either the number V of type I errors or the false discovery proportion (FDP) Q, defined as

$$Q = \begin{cases} V/R & \text{if } R > 0 \\ 0 & \text{otherwise} \end{cases}$$

which is the proportion of false rejections among the rejections, defined as 0 if no rejections are made. The table 2 summarize the numbers of errors occurring in a multiple hypothesis testing procedure.

	Rejected	**No rejected**	**Total**
True null hypotheses	V	$m_0 - V$	m_0
False null hypotheses	U	$m_1 - U$	m_1
Total	R	$m - R$	m

Table 2: Contingency table for multiple hypotesis testing: $m = \#$ of hypotheses, $m_0 = \#$ of true hypotheses, $R = \#$ of rejected hypotheses, $V = \text{card}(\mathcal{R} \cap \mathcal{T}) = \#$ type I errors (false positives).

In this tutorial, we focus on two types of multiple testing methods:

- those that control the **familywise error (FWER)**, give by

$$\text{FWER} = \text{P}(V > 0) = \text{P}(Q > 0)$$

- those that control the **false discovery rate (FDR)**, give by

$$\text{FDR} = \text{E}(Q)$$

The FWER focuses on the probability of accumulating one or more false-positive errors over a number of statistical tests. In a list of differentially expressed genes that satisfy an FWER criterion, we can have high confidence that there will be no errors in the entire list.

The FDR looks at the expected proportion of false positives among all of the genes initially identified as being differentially expressed – that is, among all the rejected null hypotheses.

9.3 Adjusted p-values

When testing a single hypothesis, we often do report whether not only a hypothesis but also the corresponding p-value was rejected. By definition, the p-value is the smallest chosen α level of the test at which the hypothesis would have been rejected. The direct analogue of this in the context of multiple testing is the adjusted p-value, defined as the smallest α level at which the multiple testing procedure would reject the hypothesis.

Suppose we conduct a hypotesis test for each gene $g = 1, \ldots, m$, producing unadjusted p-values $p_1, \ldots p_m$ (p-values that have not been corrected for multiple testing yet) for each of the hypotheses $H_1, \ldots, H_m$.

If we have corresponding adjusted p-values $\tilde{p}_g$, selecting all hypothesis with $\tilde{p}_g < \alpha$ controls the FWER and FDR at level α, that is, $\mathrm{P}(V > 0) \leq \alpha$ and $\mathrm{E}(Q) \leq \alpha$ respectively. Thus, an obvious choice of hypotheses to reject is the collection $\mathcal{R} = \{H_g : \tilde{p}_g \leq \alpha\}$, rejecting all hypotheses with an adjusted p-value below a threshold α.

In this situation, the multiple testing problem reduces to calculate adjusted p-values.

Calculating adjusted p-values

Algorithms to calculate adjusted p-values for the most popular control multiple testing methods: Bonferroni, Holm 1979, Hochberg 1988, Benjamini and Hochberg 1995, and Benjamini and Yekutieli 2001.

Calculating adjusted p-values

Start with (unadjusted) p-values for m hypotheses

1. Order the p-values $p_{(1)} \leq \cdots \leq p_{(m)}$
2. Multiply each $p_{(i)}$ by its adjustment factor a_i, $i = 1, \ldots, m$, given by
    a) *Bonferroni*: $a_i = m$
    b) *Holm* or *Hochberg*: $a_i = m - i + 1$
    c) *Benjamini & Hochberg*: $a_i = m/i$
    d) *Benjamini & Yekutieli*: $a_i = lm/i$, with $l = \sum_{k=1}^{m} 1/k$
3. Let $p'_{(i)} = a_i p_{(i)}$
4. If the multiplication in step 3 violates the original ordering, repair this:
    a) *Step-down (Holm)*: Increase the smallest p-value in all violating pairs:
    $$\tilde{p}_{(i)} = \max_{j=1,\ldots,i} p'_{(i)}$$
    b) *Step-up (all others)*: Decrease the highest p-value in all violating pairs:
    $$\tilde{p}_{(i)} = \min_{j=i,\ldots,m} p'_{(i)}$$
5. Set $\tilde{p}_{(i)} = \min(\tilde{p}_{(i)}, 1)$ for all i

Some minimal examples

- Calculating adjusted p-values from the Bonferroni method.
    - Suppose $m = 5$
    - $\alpha = 0.05$

Gene	$p_{(i)}$	$a_{(i)}$	$p'_{(i)}$	$\tilde{p}_{(i)}$
1	0.001	5	0.005	0.005
2	0.006	5	0.030	0.030
3	0.012	5	0.060	0.060
4	0.372	5	1.860	1.000
5	0.811	5	4.055	1.000

Gene 1 and 2 are declared differentially expressed.

- Now suppose $m = 50$
- Consider the same genes
- $\alpha = 0.05$

Gene	$p_{(i)}$	$a_{(i)}$	$p'_{(i)}$	$\tilde{p}_{(i)}$
1	0.001	50	0.050	0.050
2	0.006	50	0.300	0.300
3	0.012	50	0.600	0.600
4	0.372	50	18.600	1.000
5	0.811	50	40.550	1.000

Only gene 1 is declared differentially expressed.

- Calculating adjusted p-values from the Benjamini-Hochberg method.
  - Suppose $m = 100$
  - Consider the five genes with lowest p-values
  - $\alpha = 0.05$

Gene	$p_{(i)}$	$a_{(i)}$	$p'_{(i)}$	$\tilde{p}_{(i)}$
1	0.00010	100	0.01000	0.00550
2	0.00011	50	0.00550	0.00550
3	0.00520	33	0.17333	0.17333
4	0.02400	25	0.60000	0.60000
5	0.06600	20	1.32000	1.00000

Gene 1 and 2 are declared differentially expressed.

9.4 Diagnostic plots for multiple testing

For diagnostic of multiple testing results it is instructive to look at the histogram of p-values. A histogram of p-values is a bar graph of the number or proportion of p-values that fall within certain non-overlapping sub-intervals of [0, 1]. This simple graphic assessment can indicate when problems are present in the analysis. The most desirable shape for the p-value histogram is one in which the p-values are most dense near zero and become less dense as the p-values increase (Figure 19(a)). This shape does not indicate any problem of the methods operating on p-values and suggests that several genes are differentially expressed, though they may not be statistically significant after adjusting for multiple testing. An U-shaped histogram is also desirable (Figure 19(b)). The enrichment of low p-values stems from the differentially expressed genes, while those not differentially expressed are spread uniformly over the range from zero to one (except for the p-values from genes with very low counts, which take discrete values and so give rise to high counts for some bins at the right).

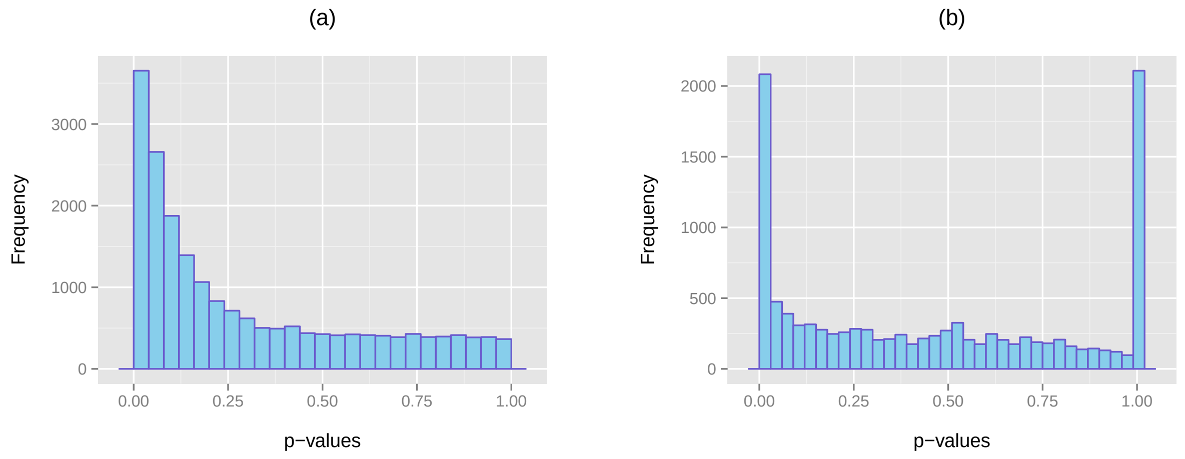
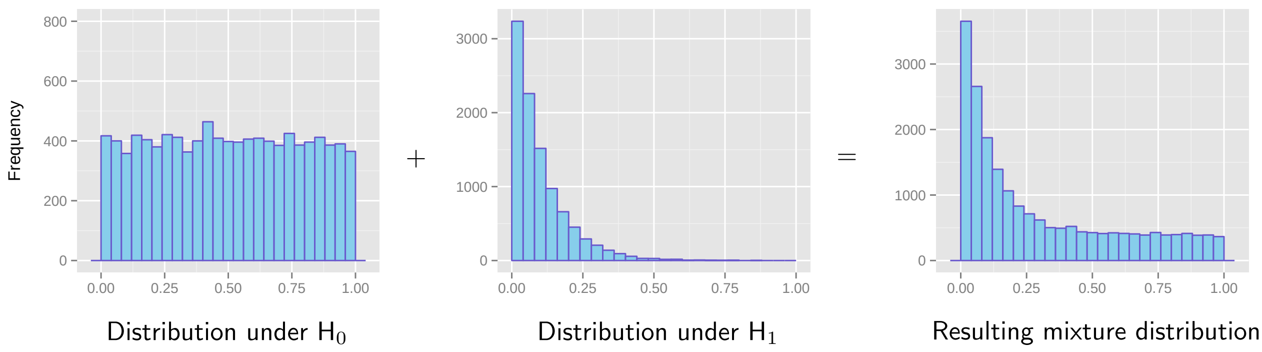


Figure 19: Desirable shape histograms of the p-values returned by the test for differential expression.

Observed p-values can be considered as mix of samples from:

- a uniform distribution (from true nulls); and
- from distributions concentrated at 0 (from true alternatives).



If the histogram of the p-values does not match a profile as shown above (Figure 19), the multiple testing approach is not reliable. A flat p-value histogram (Figure 20(a)) does not indicate any problem of the methods that operate on p-values but suggests the disappointing result that very few, if any, genes are differentially expressed. Depletion of small p-values can indicate the presence of confounding hidden variables – "batch effect" (Figure 20(b)). A p-value histogram with one or more humps (Figure 20(c)) can indicate that an inappropriate statistical test was used to compute the p-values, or a strong and extensive correlation structure is present in the data set. A p-value histogram with a choppy appearance (Figure 20(d)) indicates that the p-values are discretely distributed.

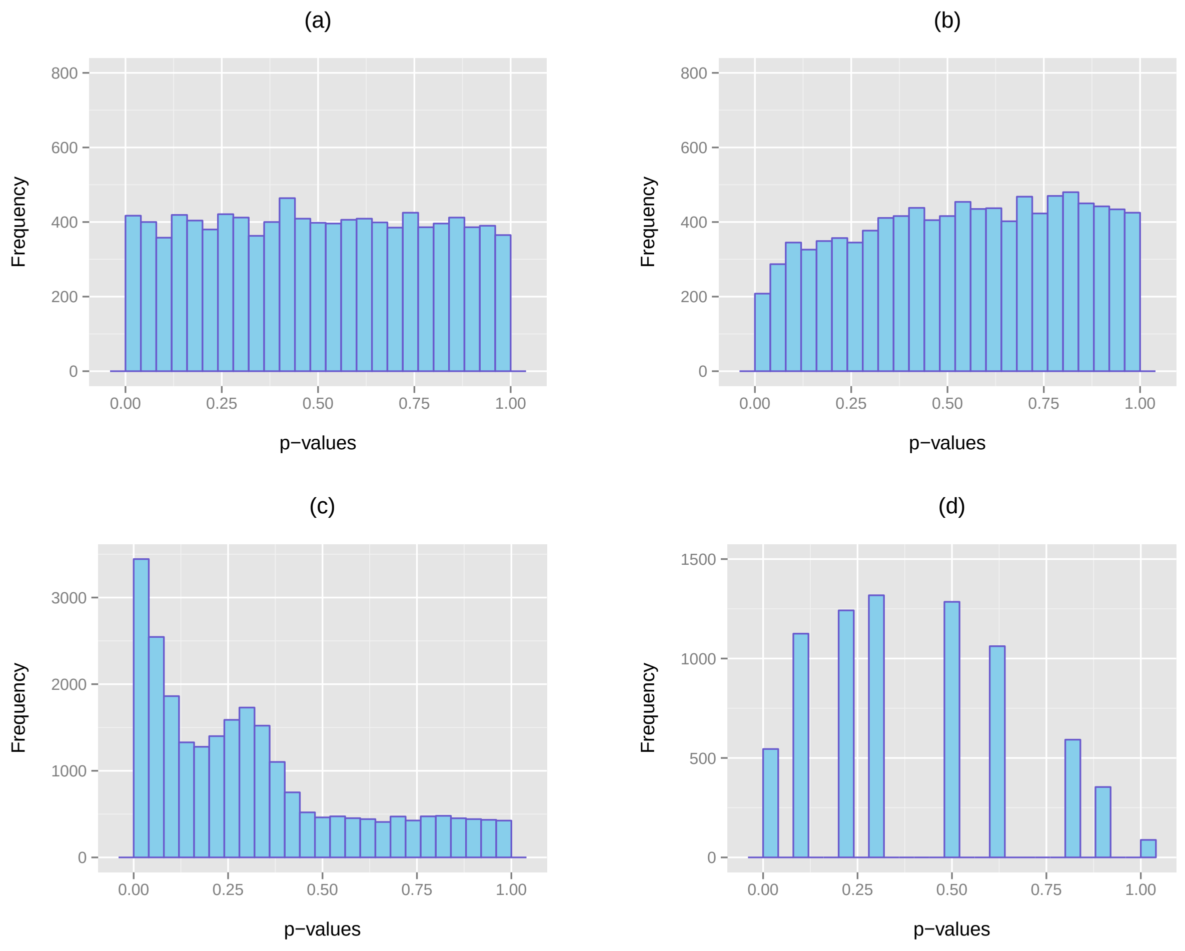
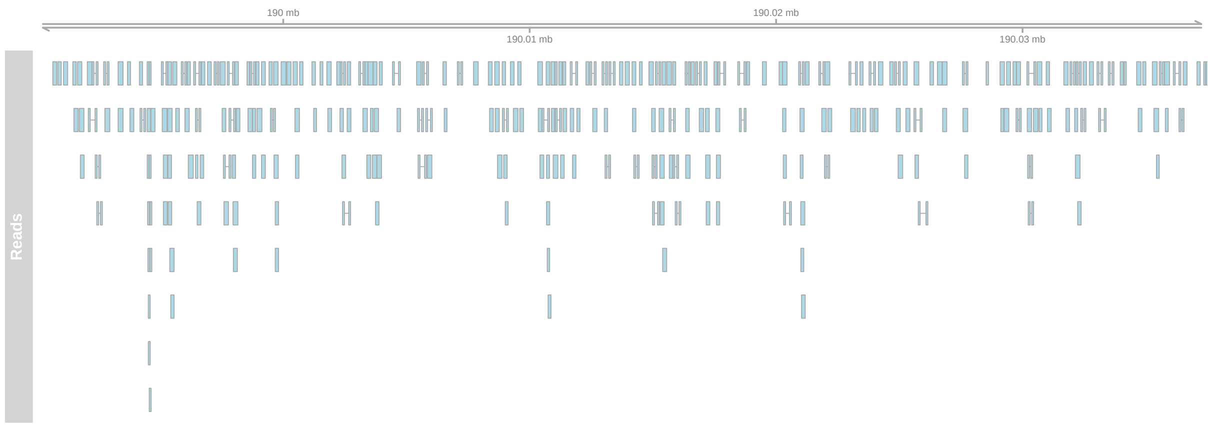


Figure 20: Examples of p-value histograms from multiple testing approach not reliable.

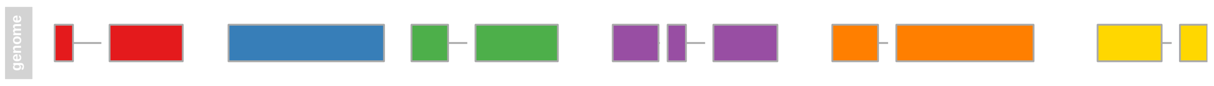
10 Differential expression analysis

10.1 Modeling count data: the Binomial and Poisson distribution

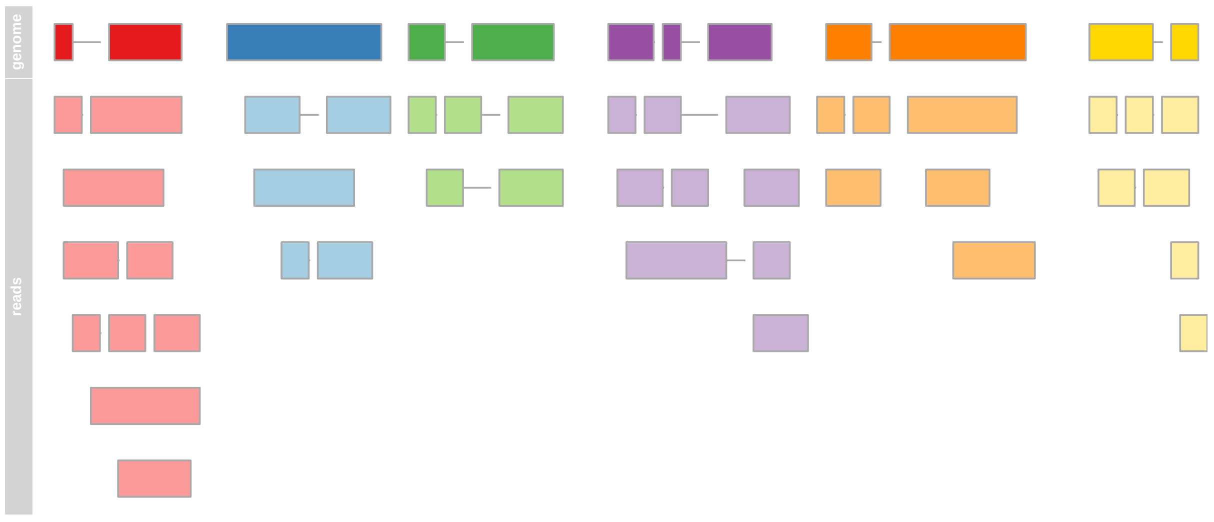
In high-throughput sequencing experiments, the raw data are millions of reads which are typically aligned to locations (regions) in the genome.



Let's suppose there are as many colors as regions g in the genome

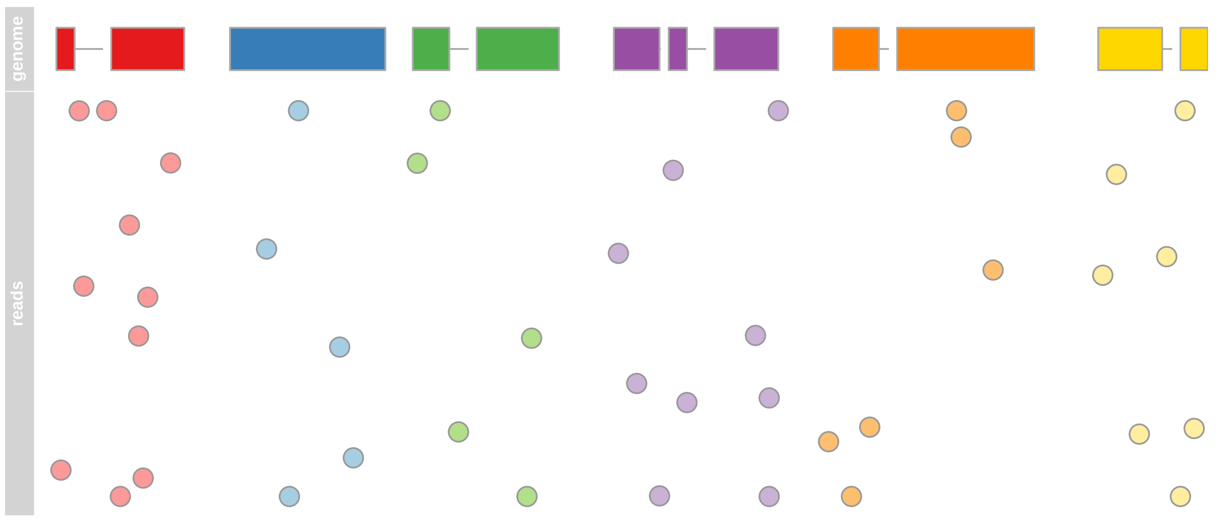


and reads which are aligned to region g take the same (light) color.



A sequenced read can be thought of as a draw of a colored ball from a bag, where there are as many colors as regions g and the bag represents a large pool of DNA fragments.

This bag contains n colored balls, the total number of sequenced reads. For each region g, independent trials consist in to draw a ball out of the bag, without looking.



Consider K_g, the number of sequenced reads which can be assigned to a particular region g.

The probability p_g (simply p to simplify notation) for drawing a ball of one color from the bag is given by the proportion of DNA fragments arising from the genomic region g.

We now derive the probability function of K_g. First we note that K_g can take the values $0, 1, 2, \ldots, n$ where n is the total number of sequenced reads. Hence, there are $n+1$ possible discrete values for K_g. We are interested in the probability that K_g takes a particular value k. If there are k aligned reads to region g (successes), then there must be $n-k$ not aligned reads to region g (failures). One way for this to happen is for the first k trials to be successes, and the remaining $n-k$ to be failures. The probability of this is given by

$$\underbrace{p \times p \times \cdots \times p}_{k \text{ of these}} \times \underbrace{(1-p) \times (1-p) \times \cdots \times (1-p)}_{n-k \text{ of these}} = p^k (1-p)^{n-k}$$

However, this is only one of the many ways that we can obtain k successes and $n-k$ failures. They could occur in exactly the opposite order: all $n-k$ failures first, then k successes. This outcome, which has the same number of successes as the first outcome but with a different arrangement, also has probability $p^k(1-p)^{n-k}$. There are many other ways to arrange k successes and $n-k$ failures among the n possible positions. Once we position the k successes, the positions of the $n-k$ failures are unavoidable.

The number of ways of arranging k successes in n trials is equal to the number of ways of choosing k objects from n objects, which is equal to

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Recall that $x! = x \times (x-1) \times (x-2) \times \ldots \times 2 \times 1$, $0!$ is defined to be 1, and that $\binom{n}{n} = \binom{n}{0} = 1$.

We can now determine the total probability of obtaining k successes. Each outcome with k successes and $n-k$ failures has individual probability $p^k(1-p)^{n-k}$, and there are $\binom{n}{k}$ possible arrangements. Hence, the total probability of k successes in n independent trials is

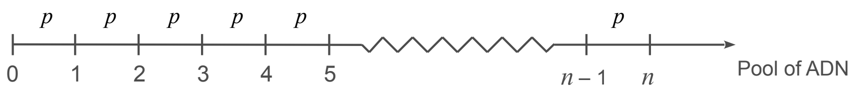
$$\mathsf{P}(K_g = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad \text{for } k = 0, 1, 2, \ldots, n$$

This is the **Binomial distribution** with parameters n and p. We write $K_g \sim \mathsf{Bin}(n,p)$.

Poisson approximation

As the number of sequenced reads becomes large and the probability p shrinks, the Binomial distribution can be approximated by the Poisson distribution:

- split the pool of DNA fragments into a large number of n intervals, each with a small probability p of success and such that the total number of successes $K_g \sim \mathsf{Bin}(n,p)$



- the number of successes, occurring in the interval $[0,\ n]$, occur independently and at a constant average rate $\lambda = np$. Then

$$\begin{aligned} \mathsf{P}(K_g = k) &= \binom{n}{k} p^k (1-p)^{n-k} \\ &= \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1 - \frac{\lambda}{n}\right)^{n-k} \\ &\approx \frac{\lambda^k}{k!} \mathrm{e}^{-\lambda} \end{aligned}$$

This is the **Poisson distribution** with parameters λ. We write $K_g \sim \mathsf{Pois}(\lambda)$.

Example

Suppose that there is a constant probability $p = 0.0001$ that an read is aligned at the region g. Then the number of sequencing reads which can be assigned to region g, K_g, in a pool of DNA of 13000 fragments is Bin(13000, 0.0001).

	k						
$\mathsf{P}(K_g = k)$	0	1	5	10	15	20	25
$\binom{n}{k} p^k (1-p)^{n-k}$	0.000074	0.000708	0.048227	0.123566	0.026498	0.001098	0.000013
$\frac{\lambda^k}{k!} \mathrm{e}^{-\lambda}$	0.000075	0.000711	0.048266	0.123502	0.026519	0.001103	0.000013

As shown in Figure 21, these two distributions are very similar.

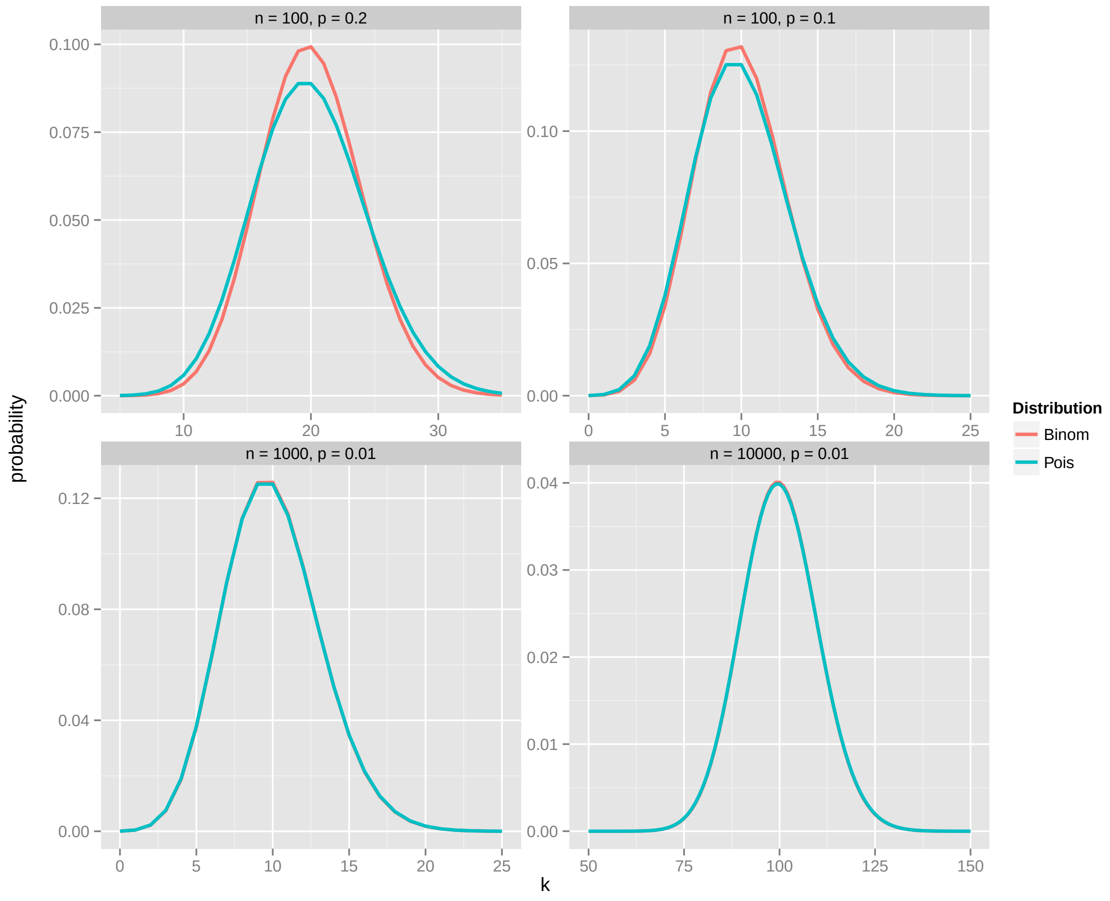


Figure 21: With a large number of trials and small probabilities, the Binomial is well approximated by a Poisson. The discrete probabilities are jointed by lines for easing the visualization.

10.2 Shot noise

Consider this situation:

- Several flow cell lanes are filled with aliquots of the same prepared library.
- The concentration of a certain transcript species is exactly the same in each lane.
- For each lane, count how often you see a read from the transcript. Will the count all be the same?

  Of course not. Even for equal concentration, the counts will vary. This theoretically unavoidable noise is called shot noise.

  Shot noise = the variance in counts that persists even if everything is exactly equal.

- Then, we make the assumption that the shot noise follows a Poisson distribution.
- For Poisson-distributed data, the variance is equal to the mean. More specifically, if $K_g \sim \text{Pois}(\lambda)$ then,

$$\text{E}(K_g) = \text{Var}(K_g) = \lambda$$

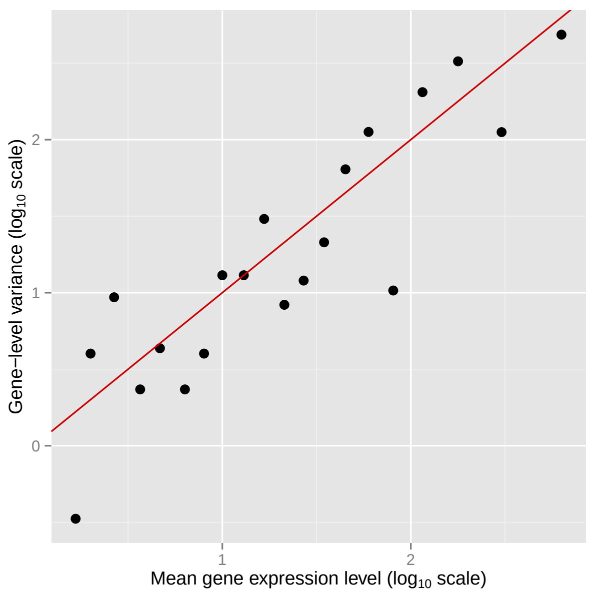


Figure 22: Example of simulated data showing the relation between mean and variance for technical replicates. The red line is the variance implied by the Poisson distribution.

10.3 Biological noise: overdispersion

While the idealized experiment described above, with repeated draws from a very large pool of DNA fragments, may be appropriate for the case of technical replicates, these assumptions are not appropriate for "biological replicates". Biological replication of an experiment implies that a new pool of DNA fragments is generated, which will not have an identical probability p_g from region g of the genome for biological replicates.

Now consider that libraries contain samples from biological replicates. So while the Poisson approximation of the binomial distribution still holds for an individual sample – as n is still large and p_g small – the variance for the counts for gene g between samples will be often much larger than the mean (see Figure 23), making the Poisson assumption restrictive.

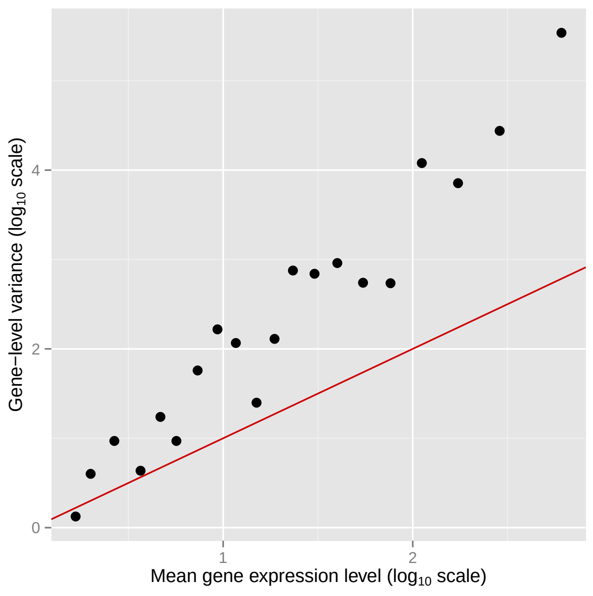


Figure 23: Example of simulated data showing the relation between mean and variance for biological replicates. The red line is the variance implied by the Poisson distribution.

When the variance in a Poisson moldel is greater than the mean, the counts are said to be "overdispersed" with respect to a Poisson distribution.

To model overdispersed counts, the Poisson distribution model can be modified as

$$\mathcal{L}(K_g|\epsilon) \sim \text{Pois}(\theta), \quad \text{with} \quad \theta = \lambda\epsilon$$

where ϵ is a nonnegative multiplicative random-effect term to model individual heterogeneity (Winkelmann 2008).

By placing a gamma distribution (see Box 2) prior on ϵ with $\alpha = \beta = r$, $\epsilon \sim \text{Gamma}(r, r)$, we have a negative binomial distribution for K_g, $K_g \sim \text{NB}(\alpha, p)$, parameterized by probability parameter $p = \lambda/(\lambda + r)$ and dispersion parameter $\alpha = r$ (see Box 3). The mean and variance is given by:

$$\text{E}(K_g) = \lambda \quad \text{and} \quad \text{Var}(K_g) = \lambda + \phi\lambda^2$$

where $\phi = 1/r$ is the inverse dispersion parameter. Thus $\text{Var}(K_g) \geq \text{E}(K_g)$ and we obtain a model for overdispersed counts. Notice that, as ϕ decreases to 0, the variance of K_g approaches the usual Poisson variance λ (i.e. np_g).

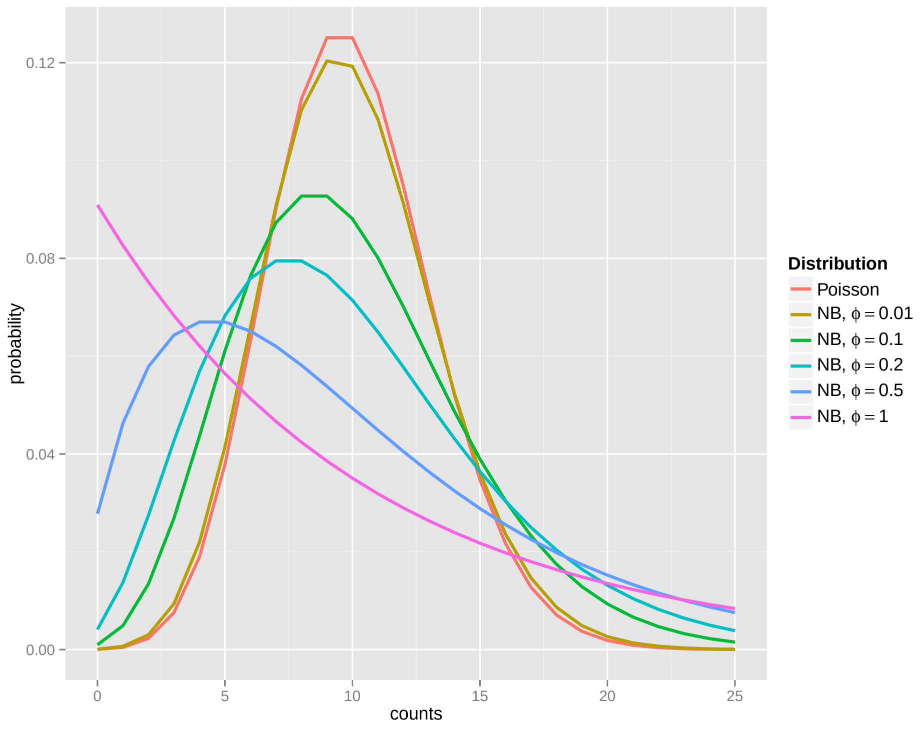


Figure 24: Poisson and negative binomial distributions, all with a mean value equal to 10. As the dispersion parameter, ϕ goes to zero, the negative binomial distribution converges to a Poisson. The discrete probabilities are jointed by lines for easing the visualization.

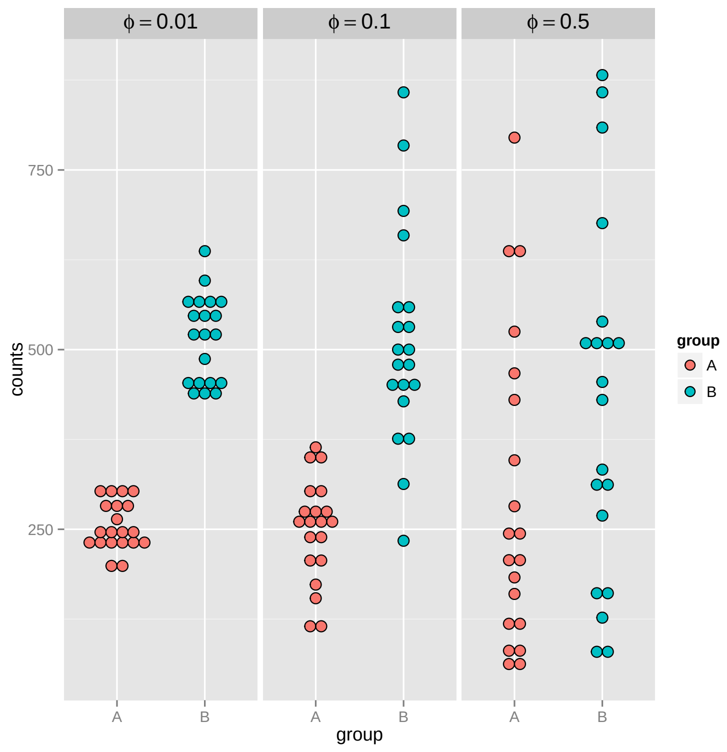


Figure 25: Example of simulated negative binomial counts with different dispersion parameters. For each plot, the true mean for group A is 250, and the true mean for group B is 500. The dispersion is changed between three values: 0.01, 0.1, 0.5.

Box 2: The Gamma distribution.

The gamma distribution

A random variable X that is gamma-distributed with shape α and rate β is denoted $X \sim \text{Gamma}(\alpha, \beta)$. The corresponding probability density function in the shape-rate parametrization is

$$g(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \quad \text{for } x \geq 0 \text{ and } \alpha, \beta > 0$$

where $\Gamma(\cdot)$ is the gamma function, and can be considered as an extension of the factorial function to continuous arguments. When the argument is a positive integer, it holds that $\Gamma(\alpha) = (\alpha - 1)!$.

The relation between the two parameters and the mean and variance of the gamma distribution are:

$$\text{E}(X) = \frac{\alpha}{\beta} \quad \text{and} \quad \text{Var}(X) = \frac{\alpha}{\beta^2}.$$

Box 3: Deduction of the Poisson gamma / negative binomial distribution.

Poisson gamma / negative binomial distribution

Count data are commonly modeled with the Poisson distribution $K \sim \text{Pois}(\lambda)$, whose mean and variance are both equal to λ. Due to heterogeneity (difference between individuals), the variance is often much larger than the mean, making the Poisson assumption not realistic. The negative binomial distribution arises by placing a gamma distribution prior on a nonnegative multiplicative random-effect term ϵ of λ. That is, we can view the negative binomial as a Poisson(θ) distribution, with $\theta = \lambda\epsilon$, where ϵ is itself a random variable distributed as a gamma distribution with shape β and scale α. Let

$$\mathcal{L}(K|\epsilon) \sim \text{Pois}(\theta), \quad \text{with} \quad \theta = \lambda\epsilon \quad \text{and} \quad \epsilon \sim \text{Gamma}(\alpha, \beta)$$

The marginal probability of K is then given by

$$\begin{aligned}
\text{P}(K = k) &= \int_0^\infty \text{Pois}(k; \lambda\epsilon) \cdot \text{Gamma}(\epsilon; \alpha, \beta) d\epsilon \\
&= \int_0^\infty \frac{(\lambda\epsilon)^k \exp(-\lambda\epsilon)}{k!} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} \epsilon^{\alpha-1} \exp(-\beta\epsilon) d\epsilon \\
&= \frac{\lambda^k \beta^\alpha}{k!\Gamma(\alpha)} \int_0^\infty \epsilon^{k+\alpha+1} \exp(-(\lambda+\beta)\epsilon) d\epsilon \\
&= \frac{\lambda^k \beta^\alpha}{k!\Gamma(\alpha)} \cdot \frac{1}{(\lambda+\beta)^{k+\alpha}} \int_0^\infty x^{k+\alpha-1} \text{e}^{-x} dx, \qquad \text{changing variables} \\
&= \frac{\Gamma(\alpha+k)}{k!\Gamma(\alpha)} \left(\frac{\lambda}{\lambda+\beta}\right)^k \left(\frac{\beta}{\lambda+\beta}\right)^\alpha, \qquad \text{using the fact that} \quad \int_0^\infty x^n \text{e}^{-x} dx = \Gamma(n+1)
\end{aligned}$$

Defining

$$p = \frac{\lambda}{\lambda+\beta}$$

the last equation can be written as

$$\text{P}(K = k) = \frac{\Gamma(\alpha+k)}{k!\Gamma(\alpha)} p^k (1-p)^\alpha$$

which is the negative binomial distribution, also known as the Gamma-Poisson distribution, parameterized by the probability parameter p and dispersion parameter α. We write $K \sim \text{NB}(\alpha, p)$, the mean and variance are given by:

$$\text{E}(K) = \alpha\frac{p}{(1-p)} = \frac{\alpha\lambda}{\beta}, \qquad \text{Var}(K) = \alpha\frac{p}{(1-p)^2} = \frac{\alpha\lambda}{\beta}\left(1 + \frac{\lambda}{\beta}\right)$$

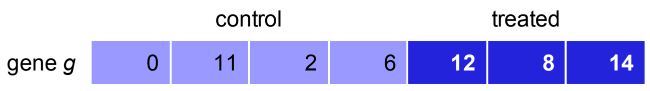
10.4 Identifying differentially expressed genes

We discuss the simple setting of a RNA Seq experiment that compares the expression levels of genes in a control condition with those in a treatment condition. Genes in the two conditions will invariably be measured with different read counts. These differences will represent either true, biological differences between the two conditions, or experimental noise. Statistical models and tests are used to distinguish between the two possibilities.

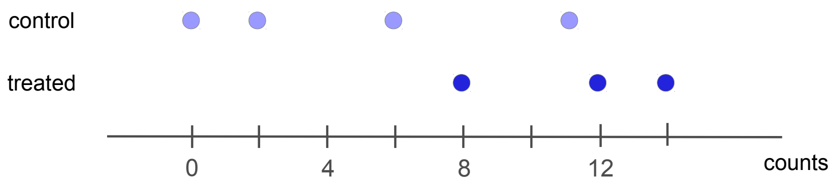
Testing for differential expression

Suppose that we have m_A replicate samples for control condition (group A) and m_B samples for treatment condition (group B). For each gene g, we would like to weigh the evidence in the data for differential expression of that gene between the two conditions. To this end, statistic tests will be defined.

We first consider a single gene g,



at first glance at the dot plots for the two groups, several questions come in mind when we are about to formulate a statistical test.



- First, do the data come from Normal distributions?
  Unfortunately we can not say much about the distributions as the samples are too small. However there does not seem to be any clear lack of symmetry.
- Second, are the two distributions similar in shape?
  Again it is hard to say much with such small samples, though the control condition data seem to have longer tails.
- Finally, is there any difference in the centers of location?
  The plots suggest a difference with treatment condition values being larger on average.

Perhaps the most natural test for differential expression between multiple libraries involved pooling the normalized libraries in each condition and using the standard two-sample difference in proportions test or the Fisher exact test. The normalized data are rounded to produce integer values that can be used with the Fisher exact test.

Fisher exact test (see Box 4) fixes the marginal totals of the 2×2 contingency table (Table 3) and tests differential expression using the null hypotheses that treatment do not affect gene expression levels (i.e., that the two groups are independent).

	Group A	Group B	Total
Gene g	n_{g_A}	n_{g_B}	n_g
Remaining genes	$N_A - n_{g_A}$	$N_B - n_{g_B}$	$N - n_g$
Total	N_A	N_B	N

Table 3: 2×2 contingency table for gene g. The cell counts n_{g_A} (n_{g_B}) represent the count for gene g in group A (B) and $N_A - n_{g_A}$ ($N_B - n_{g_B}$) represent the count for the remaining genes for group A (B). The marginal row total for gene g is denoted $n_g = n_{g_A} + n_{g_B}$ and for remaining genes $N - n_g$. N_A and N_B are the marginal total for group A and B, and $N = N_A + N_B$ is the grand total.

The exact method calculates the probability of each table and then sum the probability of the given table and every other "unusual" table. The "unusual" tables are those that have probabilities less than or equal to the given table. If the total probability of such unusual tables is "small" – i.e., if it is a rare event to observe such unlikely tables – we can reject the null hypothesis that treatment do not affect gene expression levels.

Box 4: Fisher exact test.

Fisher's exact test

The Fisher exact test looks at a 2×2 contingency table which displays how different treatments have produced different outcomes. Its null hypothesis is that treatments do not affect outcomes. If the columns represent the study group and the rows represent the outcome, then the null hypothesis could be interpreted as the probability of having a particular outcome not being influenced by the study group, and the test evaluates whether the two study groups differ in the proportions with each outcome.

	Group A	Group B	Total
outcome 1	a	b	$a+b$
outcome 2	c	d	$c+d$
Total	$a+c$	$b+d$	n

2×2 contingency table representing the cells by the letters a, b, c and d, and represent the grand total by n.

The test is based upon calculating directly the probability of obtaining the results that we have shown (or results more extreme) if the null hypothesis is actually true, using all possible 2×2 tables that could have been observed, for the same row and column totals as the observed data. What we are trying to establish is how extreme our particular table (combination of cell frequencies) is in relation to all the possible ones that could have occurred given the marginal totals.

Under the null hypothesis of no association Fisher showed that the probability of obtaining the frequencies a, b, c and d in table is given by the hypergeometric distribution:

$$\frac{\binom{a+b}{a}\binom{c+d}{c}}{\binom{n}{a+c}}$$

In order to calculate the significance of the observed data, i.e. the total probability of observing data as extreme or more extreme if the null hypothesis is true, we will use a simple example to show the calculation of the p-value. Consider the following particular 2×2 table:

$$\begin{array}{cc|c} 3 & 16 & 19 \\ 4 & 1 & 5 \\ \hline 7 & 17 & 24 \end{array} \quad \text{with probability} \quad \frac{\binom{19}{3}\binom{5}{4}}{\binom{24}{7}} = 0.014.$$

The set of similar tables includes just six tables:

$$\left|\begin{array}{cc|cc|cc|cc|cc|cc} 7 & 12 & 6 & 13 & 5 & 14 & 4 & 15 & 3 & 16 & 2 & 17 \\ 0 & 5 & 1 & 4 & 2 & 3 & 3 & 2 & 4 & 1 & 5 & 0 \end{array}\right|$$

having respective probabilities equal to 0.1456, 0.392, 0.336, 0.112, 0.014 and 0.00049.

Two-sided test is based on the probabilities of the tables, and take as "more extreme" all tables with probabilities less than or equal to that of the observed table, the p-value being the sum of such probabilities.

Therefore, the probability of obtaining our results more extreme is the sum of the probabilities $0.014 + 0.00049 = 0.0145$.

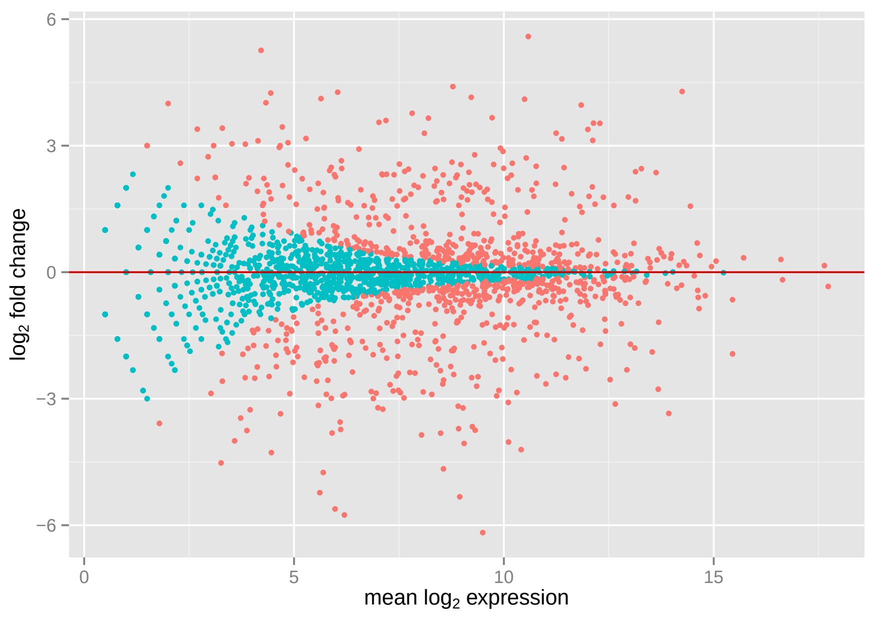


Figure 26: MA-plot between control and treatment of the gene expression counts. Pink dots represent significantly differentially expressed genes as established by Fisher's exact test at 1%; Blue dots represent genes with similar expression. The red horizontal line at zero provides a visual check for symmetry.

Figure 26 illustrates the behavior of Fisher's exact test for testing differential expression, between two condition groups, for every gene in the RNA-Seq data set. Fisher's exact test becomes

- more conservative as expression values decrease to zero, a point concurrent with the fact that genes with small expression values also demonstrate larger variability; and
- less conservative for the higher expression values, a point concurrent with the fact that genes with larger expression values demonstrate small variability.

The figure 27 illustrates this phenomenon for Poisson distributed data. For example, the difference in expression of a gene measured with 2 reads versus 4 reads is inherently less certain than the differences in expression of a gene measured with 100 reads versus 200 reads, even though both differences are nominally a 2× fold change.

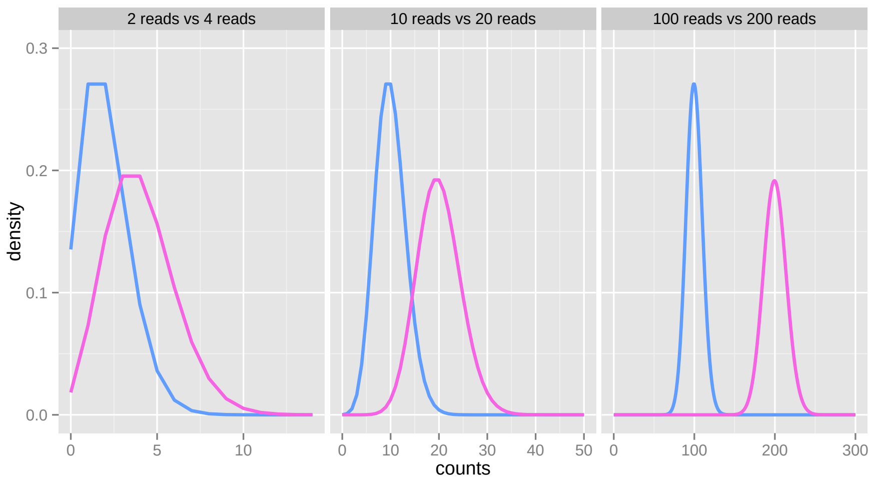


Figure 27: Poisson distribution showing the uncertainty present in a 2× fold change using count data at different counts.

This shows that Fisher's exact test only deals with the shot noise and fails to take into account the variability between libraries.

The negative binomial model

In order to properly capture shot- and biological-noise, the negative binomial model is introduced, including both types of variation in a hierarchical fashion:

- Suppose that the proportion (or the relative abundance) θ_g of gene g is selected from an underlying gamma distribution representing the between-library (biological) variation. Assume that θ_g is distributed according to

  $$\theta_g = \lambda_g \epsilon_g \quad \text{with} \quad \epsilon_g \sim \text{Gamma}(r_g, r_g),$$

  where ϵ_g is the nonnegative multiplicative random-effect term to model biological heterogeneity.
- Given λ_g, the count K_g is binomial with the chosen proportion as a parameter and

  $$\mathcal{L}(K_g|\epsilon_g) \sim \text{Pois}(\theta_g)$$

  The mean of K_g is given by $\text{E}(K_g) = \lambda_g$ and the variance is the sum of a *shot noise* term and a *biological noise* term

  $$\text{Var}(K_g) = \underbrace{\lambda_g}_{\text{shot noise}} + \underbrace{\phi_g \lambda_g^2}_{\text{biological noise}}$$

That is, K_g follow a negative binomial distribution with parameters ϕ_g and p_g, $K_g \sim \text{NB}(\phi_g^{-1}, p_g)$, parametrised in terms of its mean λ_g and variance σ_g^2, via

$$p_g = \frac{\lambda_g}{\sigma_g^2} \quad \text{and} \quad \phi_g = \frac{\sigma_g^2 - \lambda_g}{\lambda_g^2}$$

Hypothesis testing

We wish to test whether the relative abundance under experimental condition A is the same as that in condition B, leading to a null hypothesis of

$$\text{H}_0 : \lambda_{gA} = \lambda_{gB}, \quad \text{for each gene } g.$$

To this end, we define, as test statistic, the total counts in each condition,

$$n_{g_A} = \sum_{j:\rho(j)=A} K_{gj}, \qquad n_{g_B} = \sum_{j:\rho(j)=B} K_{gj}$$

and their overall sum $n_g = n_{g_A} + n_{g_B}$, where $\rho(j)$ is the experimental condition of sample j, $\rho(j) \in \{A, B\}$. Then we can construct an exact test similar to the Fisher's exact test for contingency tables but replacing the hypergeometric probabilities with NB. Conditioning on the total, n_g, we can calculate the probability of observing class totals equal to or more extreme than those observed, giving an exact method for assessing differential expression.

From the noise model described above, we show below that – under the null hypothesis – we can compute the probabilities of the events $n_{g_A} = a$ and $n_{g_B} = b$ for any pair of numbers a and b. We denote this probability by $p(a, b)$. The p-value, P_g, of a pair of observed count sums (n_{g_A}, n_{g_B}) is then the sum of all probabilities less or equal to $p(n_{g_A}, n_{g_B})$, given that the overall sum is n_g:

$$P_g = \sum_{\substack{a+b=n_g \\ p(a,b) \le p(n_{g_A}, n_{g_B})}} p(a, b)$$

the variables a and b in the above sums take the values $0, \ldots, n_g$.

Computation of $p(a, b)$. First, assume that, under the null hypothesis, counts from different samples are independent. Then,

$$p(a, b) = P(n_{g_A} = a)P(n_{g_B} = b).$$

Thus, the problem is to compute the probability of the event $n_{g_A} = a$, and, analogously, of $n_{g_B} = b$.

Now assume that libraries are normalized, so that counts are then independent and approximately identically distributed. We approximate its distribution by a NB distribution whose parameters we obtain from those of the K_{gj}: let n_{g_A} and n_{g_B} be the sum of pseudocounts for class A and class B, respectively, over the number of libraries, m_A and m_B. Under the null hypothesis,

$$n_{g\rho} \sim \text{NB}(m_\rho \phi_g^{-1}, p_g), \quad \rho \in \{A, B\}.$$

In practice, we do not know the parameters ϕ_g and p_g (i.e. λ_g and σ_g^2), and we need to estimate them from the data. Typically, the number of replicates is small, and further modelling assumptions need to be made in order to obtain useful estimates.

10.5 Estimating dispersion parameters

When a negative binomial model is fitted, within-group variability, i.e., the variability between replicates, is modeled by the dispersion parameter ϕ_g, which describes the variance of counts via $\text{Var}(K_g) = \lambda_g + \phi_g \lambda_g^2$.

The parallel nature of sequencing data allows some possibilities for borrowing information from the ensemble of genes which can assist in inference about each gene individually. The easiest way to share information between genes is to assume that all genes have the same mean-variance relationship, in other words, the dispersion is the same for all the genes. An extension to this "common dispersion" approach is to put a mean-dependent trend on a parameter in the variance function, so that all genes with the same expected count have the same variance.

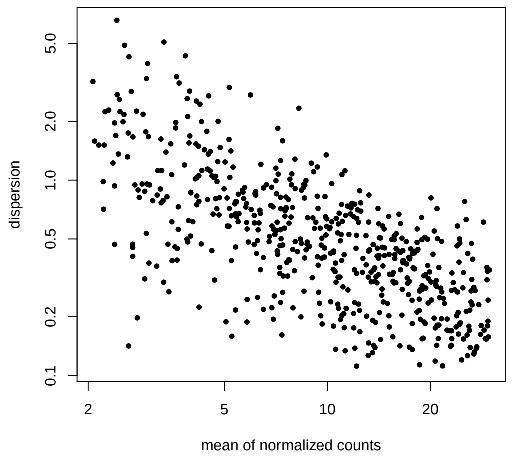
However, the truth is that the gene expression levels have non-identical and dependent distribution between genes, which makes the above assumptions too naive. A more general approach allowing gene-specific dispersion is necessary.

DESeq, DESeq2 approach

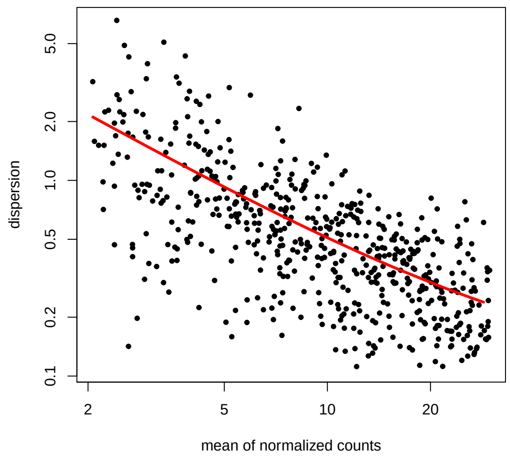
```
estimateDispersions(...){DESeq, DESeq2}
```

The function `estimateDispersions` proceeds as follows:

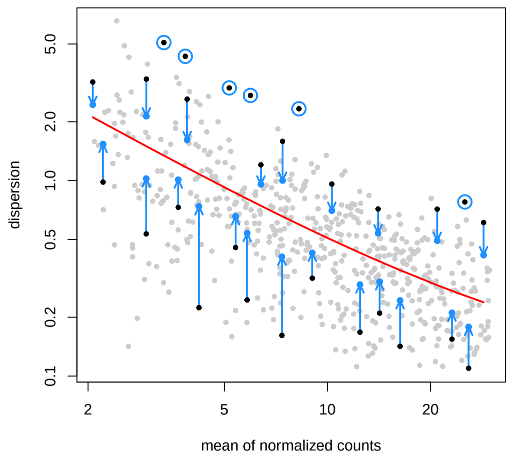
- First, it use the count data for each gene separately to get preliminary gene-wise dispersion estimates by maximum likelihood estimation (McCarthy, Chen, and Smyth 2012). Unless one has many samples, these values fluctuate strongly around their true values.



- Next, it fits a trend curve to the maximum likelihood estimates to capture the dependence of these estimates on average expression strength (red line).



- Finally, the fitted curve is used as a prior mean for a second estimation round, which results in the final estimates of dispersion (arrow heads). This can be understood as a shrinkage (along the blue arrows) of the noisy gene-wise estimates towards the consensus represented by the red line. The black points circled in blue 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. For clarity, only a subset of genes is highlighted.



Example: Here is a simple example of estimating dispersions using the *DESeq2* approach. Given a DESeqDataSet object dds, we estimate the dispersions using the following commands.

```
## Constructs a simulated dataset of negative binomial data
set.seed(2)
dds = makeExampleDESeqDataSet()

## Estimate the size factors for the dds DESeqDataSet
dds = estimateSizeFactors(dds)

## Estimate the dispersions for negative binomial distributed data
dds = estimateDispersions(dds)

## Plot dispersion estimates
plotDispEsts(dds, ymin = 0.01)
```

The plotDispEsts is a simple helper function that plots the per-gene dispersion estimates together with the fitted mean-dispersion relationship.

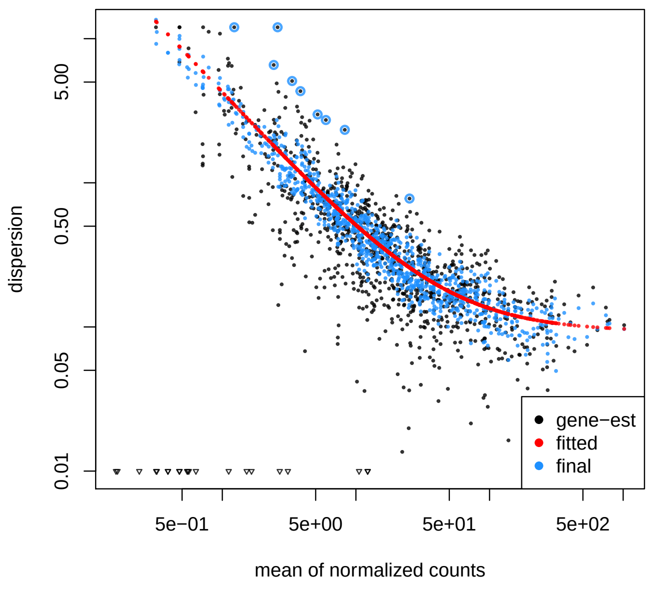


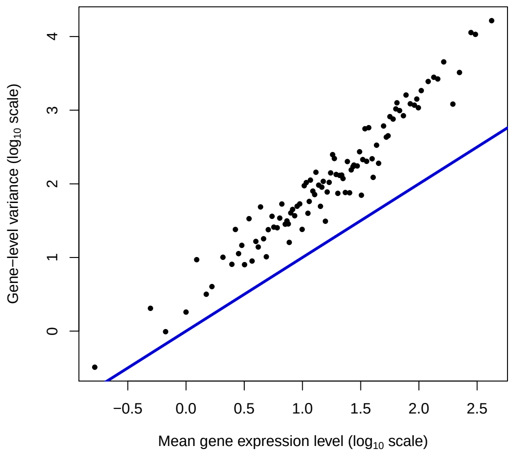
Figure 28: The dispersion estimate plot shows the gene-wise estimates (black), the fitted values (red), and the final maximum a posteriori estimates(blue). Points beyond the lower bound `ymin` are drawn as triangles at `ymin`.

edgeR approach

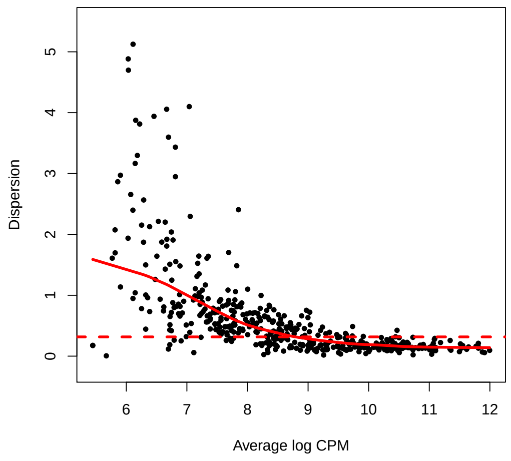
```
estimateCommonDisp(...){edgeR}
estimateTagwiseDisp(...){edgeR}
```

The *edgeR* approach implements the empirical Bayes strategy proposed by Robinson and Smyth 2007; Robinson and Smyth 2008 for estimating the tagwise negative binomial dispersions:

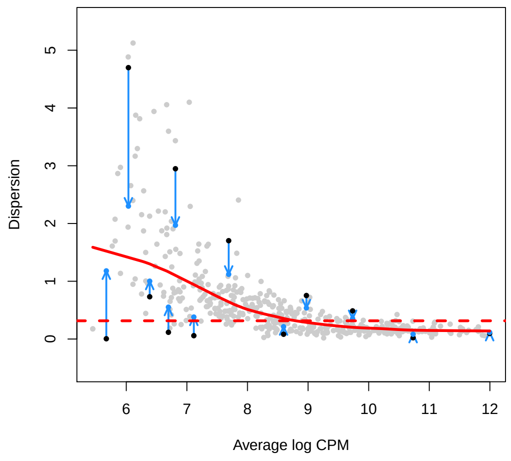
- First, it use the count data for estimating a common dispersion parameter for all the genes by conditional maximum likelihood. To obtain the estimate for the common dispersion, pseudocounts (counts adjusted so that the library sizes are equal) are calculated under the Poisson model (dispersion is zero) and these pseudocounts are used to give an estimate of the common dispersion. The blue line is the variance implied by the Poisson model.



- Next, it adjustes profile likelihood as the prior distribution for tagwise dispersions (red line). The dashed red line is the dispersion estimate using the Poisson model.



- Finally, the adjusted profile is used as a prior value for a second estimation round, which results in the final estimates of dispersion (arrow heads). This apply an empirical Bayes strategy for squeezing the tagwise dispersions towards the prior value more or less strongly depending on how reliably the individual genewise dispersion can be estimated. Genes for which the counts are very low provide relatively little statistical information for estimating their own dispersion so, in these cases, the prior value dominates and the genewise dispersions are squeezed heavily towards the overall trend. For clarity, only a subset of genes is highlighted.



Example: Here is a simple example of estimating dispersions using the *edgeR* approch. Given the simulated dataset in the above example, we estimate the dispersions using the following commands.

```
set.seed(2)
y = counts(makeExampleDESeqDataSet())

## Creates a DGEList object from counts matrix
dge = DGEList(y)

## Estimate common dispersion
dge = estimateCommonDisp(dge)

## Estimate tagwise dispersions
dge = estimateTagwiseDisp(dge)

## Plot genewise biological coefficient of variation
plotBCV(dge, cex = 0.8)
```

The `plotBCV` is a simple helper function that plots the per-gene dispersion estimates together with the fitted mean-dispersion relationship.

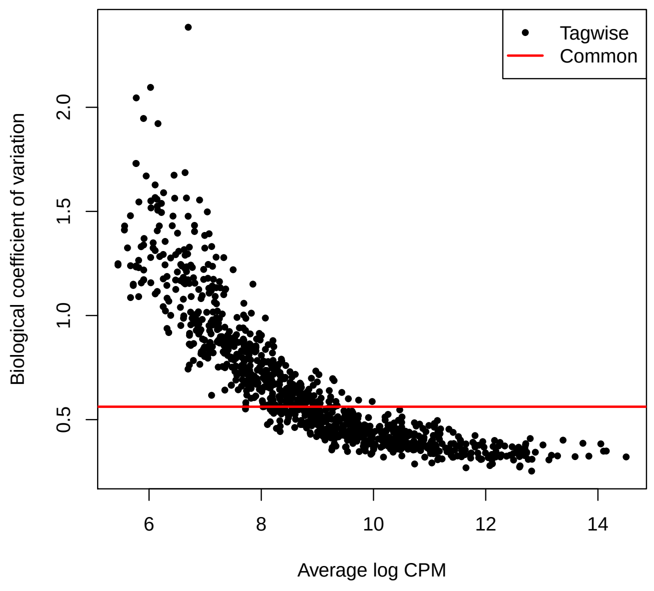


Figure 29: Plot genewise biological coefficient of variation (BCV) against gene abundance (in $\log_2$ counts per million). The BCV is the square root of the negative binomial dispersion.

10.6 Inference: calling differential expression

nbinomWaldTest(...){*DESeq*, *DESeq2*}

exactTest(...){*edgeR*}

Once negative binomial models are fitted and dispersion estimates are obtained for each gene, it is straight-forward to look for differentially expressed genes. To contrast two conditions, e.g., to see whether there is differential expression between conditions "control" and "treated", we simply call the function nbinomWaldTest in *DESeq2* (or *DESeq*) package or the function exactTest in *edgeR* package. Its performs the tests as described in the previous section and returns a data frame with the p-values and other useful information.

DESeq2 analysis example:

```
## Simulated data
set.seed(1)
dds = makeExampleDESeqDataSet(n = 1000, m = 6, betaSD = 1)

## DESeq2 analysis
dds = estimateSizeFactors(dds)
dds = estimateDispersions(dds)
dds = nbinomWaldTest(dds)

## Extract results from DESeq2 analysis
resDESeq2 = results(dds)
resDESeq2
```

```
log2 fold change (MAP): condition B vs A
Wald test p-value: condition B vs A
```

```
DataFrame with 1000 rows and 6 columns
          baseMean log2FoldChange     lfcSE      stat    pvalue      padj
         <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
gene1       12.247         1.5333    0.8054    1.9039   0.05693    0.2096
gene2       22.568         1.3896    0.6556    2.1197   0.03403    0.1554
gene3        3.961        -1.8592    0.9755   -1.9059   0.05666    0.2096
gene4      143.035        -0.5476    0.3421   -1.6010   0.10939    0.3082
gene5       16.301        -0.2533    0.6843   -0.3702   0.71125    0.8570
...            ...            ...       ...       ...       ...       ...
gene996     9.5873        1.33286    0.8254   1.61480    0.1064    0.3014
gene997     6.6044        0.08247    0.8567   0.09627    0.9233    0.9695
gene998     8.5560       -0.78911    0.7933  -0.99472    0.3199    0.5937
gene999     0.9542        0.66981    0.9694   0.69098    0.4896        NA
gene1000    3.7779        0.68939    1.0170   0.67789    0.4978    0.7372
```

The result of the columns of `resDESeq2` (a *DESeqResults* object) is as follows:

	feature identifier
`baseMean`	mean normalised counts, averaged over all samples from both conditions
`log2FoldChange`	the logarithm (to basis 2) of the fold change
`lfcSE`	standard errors of logarithm fold change
`stat`	test statistics
`pvalue`	p value for the statistical significance of this change
`padj`	p value adjusted for multiple testing with the Benjamini-Hochberg procedure (see the R function `p.adjust`), which controls false discovery rate (FDR)

edgeR analysis example:

```
## Simulated data in the DESeq2 analysis example
set.seed(1)
y = makeExampleDESeqDataSet(n = 1000, m = 6, betaSD = 1)
group = colData(y)$condition
y = counts(y)

## edgeR analysis
dge = DGEList(counts = y, group = group)
dge = estimateCommonDisp(dge)
dge = estimateTagwiseDisp(dge)
et = exactTest(dge)

## Extract results from edgeR analysis
resEdgeR = topTags(et)
resEdgeR

Comparison of groups:  B-A
         logFC logCPM    PValue       FDR
gene61  -3.105 12.593 9.070e-14 9.070e-11
gene442 -3.504 11.183 3.115e-11 1.558e-08
gene166 -2.620 12.449 8.150e-11 2.717e-08
gene967 -3.152 10.480 4.151e-10 1.038e-07
gene545 -3.008 10.595 5.346e-10 1.069e-07
gene841  2.513 13.989 1.024e-09 1.706e-07
gene295  2.754 14.523 1.734e-09 2.477e-07
gene78  -6.259  8.303 5.220e-09 6.525e-07
gene737  3.068 12.494 8.559e-09 9.510e-07
gene397 -2.486 11.300 1.249e-07 1.249e-05
```

The result of the columns of `resEdgeR` (a *TopTags* object) is as follows:

	feature identifier
`logFC`	the logarithm (to basis 2) of the fold change
`logCPM`	the average $\log_2$ counts-per-million (CPM) for each tag
`PValue`	p value for the statistical significance of this change
`FDR`	p value adjusted for multiple testing with the Benjamini-Hochberg procedure (see the R function `p.adjust`), which controls false discovery rate (FDR)

11 Interpreting the differential expression analysis results

Proper interpretation of the large data tables generated by any method for analysis of gene expression requires tools for effective mining of the resultats. Visualization is a good way for this, giving overall summary statistics for differential expression analysis results. In this Section, we will introduce three different tools that are used a lots in differential expression data interpretation: MA-plot, volcano plot, and gene clustering.

11.1 MA-plot

`plotMA(...)`{*DESeq, DESeq2*}

`plotSmear(...)`{*edgeR*}

A so-called MA-plot provides a useful overview for an experiment with a two-group comparison. This plot represents each gene with a dot. The x axis is the average expression over the mean of normalized counts (A-values), the y axis is the $\log_2$ fold change between treatments (M-values). Genes with an adjusted p-value below a threshold (e.g. 0.1) are often highlighted. *DESeq*, *DESeq2* and *edgeR* provides a simple helper function that makes a MA-plot.

In *DESeq2*, the function `plotMA` incorporates a prior on $\log_2$ fold changes, resulting in moderated estimates from genes with low counts and high dispersion, as can be seen by the narrowing of spread of leftmost points in the right plot Figure 30.

```
DESeq2::plotMA(resDESeq2, main = "DESeq2", ylim = c(-4, 4))
```

The left plot Figure 30 shows the "unshrunken" $\log_2$ fold changes as produced by the `plotMA` function in *DESeq*.

```
DESeq::plotMA(resDESeq, main = "DESeq", ylim = c(-4, 4))
```

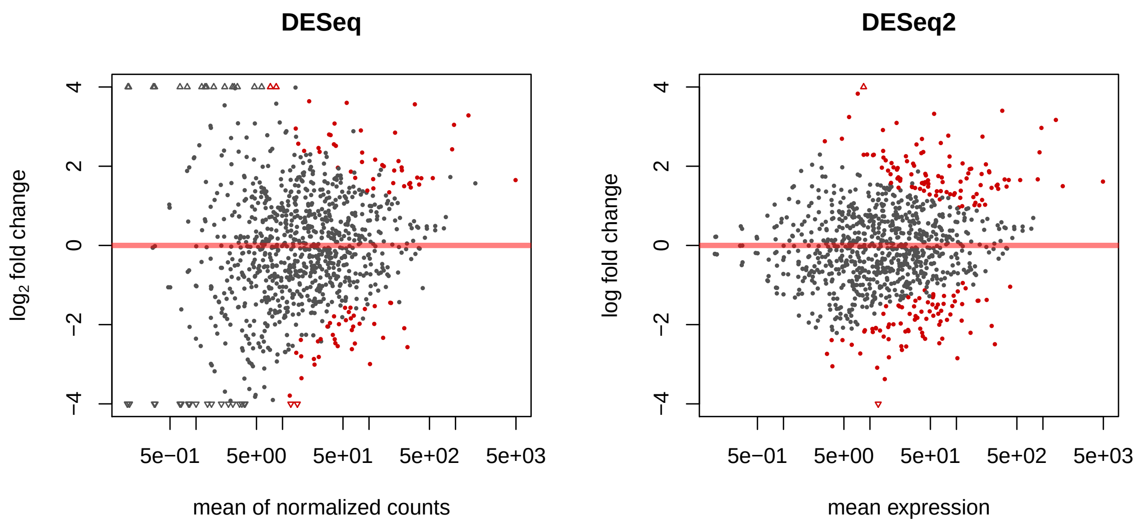


Figure 30: MA-plot generated by the `plotMA` function in *DESeq* (left) and *DESeq2* (right). Points will be colored red if the adjusted p-value is less than 0.1. Points which fall out of the window are plotted as open triangles pointing either up or down.

For a particular gene, a $\log_2$ fold change of -1 for condition treated vs control means that the treatment induces a change in observed expression level of $2^{-1} = 0.5$ compared to the control condition. Analogously, a $\log_2$ fold change of 1 means that the treatment induces a change in observed expression level of $2^1 = 2$ compared to the control condition.

In *edgeR*, the function `plotSmear` resolves the problem of plotting tags that have a total count of zero for one of the groups by adding the "smear" of points at low A value. The points to be smeared are identified as being equal to the minimum estimated concentration in one of the two groups.

```
de = decideTestsDGE(resEdgeR, p.value = 0.01)
de.genes = rownames(resEdgeR)[as.logical(de)]

plotSmear(resEdgeR, de.tags = de.genes, cex = 0.5)
abline(h = c(-2, 2), col = "blue")
```

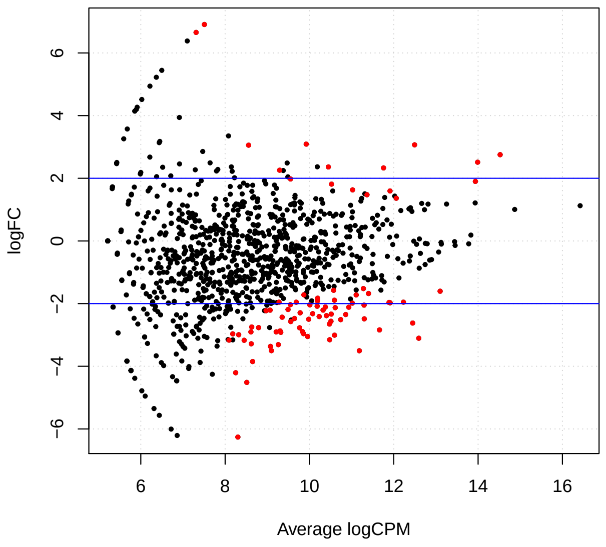


Figure 31: MA-plot generated by the `plotSmear` function in *edgeR*. Points will be colored red if the adjusted p-value is less than 0.01. A "smear" of points at low A value is presented to the left of the minimum A for counts that were low (e.g. zero in 1 library and non-zero in the other). The horizontal blue lines show 4-fold changes.

11.2 The volcano plot

The "volcano plot" is an effective and easy-to-interpret graph that summarizes both fold-change and a measure of statistical significance from a statistical test (usually a p-value). It is a scatter-plot of the negative $\log_{10}$-transformed p-values from the gene-specific test (on the y-axis) against the $\log_2$ fold change (on the x-axis).

This results in datapoints with low p-values (highly significant) appearing towards the top of the plot. The $\log_2$ of the fold-change is used so that changes in both directions (up and down) appear equidistant from the center. Plotting points in this way results in two regions of interest in the plot: those points that are found towards the top of the plot that are far to either the left- or the right-hand side. These represent values that display large magnitude fold changes (hence being left- or right- of center) as well as high statistical significance (hence being towards the top).

For illustration we will use the result in the `resEdgeR` object from *edgeR* analysis. We construct a table containing the $\log_2$ fold change and the negative $\log_{10}$-transformed p-values:

```
tab = data.frame(logFC = resEdgeR$table[, 1], negLogPval = -log10(resEdgeR$table[, 3]))
head(tab)

   logFC negLogPval
1  1.322     0.6966
2  1.074     0.8097
3 -3.214     1.9119
4 -1.092     2.2188
```

```
5 -0.785      0.5153
6 -1.774      0.3573
```

The volcano plot can then be generated using the standard plot command:

```
par(mar = c(5, 4, 4, 4))
plot(tab, pch = 16, cex = 0.6, xlab = expression(log[2]~fold~change),
     ylab = expression(-log[10]~pvalue))
```

We can identify genes (points) in the two regions of interest on the plot: points with large magnitude fold changes (being left- or right- of center) and points with high statistical significance (being towards the top).

```
## Log2 fold change and p-value cutoffs
lfc = 2
pval = 0.01

## Selecting interest genes
signGenes = (abs(tab$logFC) > lfc & tab$negLogPval > -log10(pval))

## Identifying the selected genes
points(tab[signGenes, ], pch = 16, cex = 0.8, col = "red")
abline(h = -log10(pval), col = "green3", lty = 2)
abline(v = c(-lfc, lfc), col = "blue", lty = 2)
mtext(paste("pval =", pval), side = 4, at = -log10(pval), cex = 0.8, line = 0.5, las = 1)
mtext(c(paste("-", lfc, "fold"), paste("+", lfc, "fold")), side = 3, at = c(-lfc, lfc),
      cex = 0.8, line = 0.5)
```

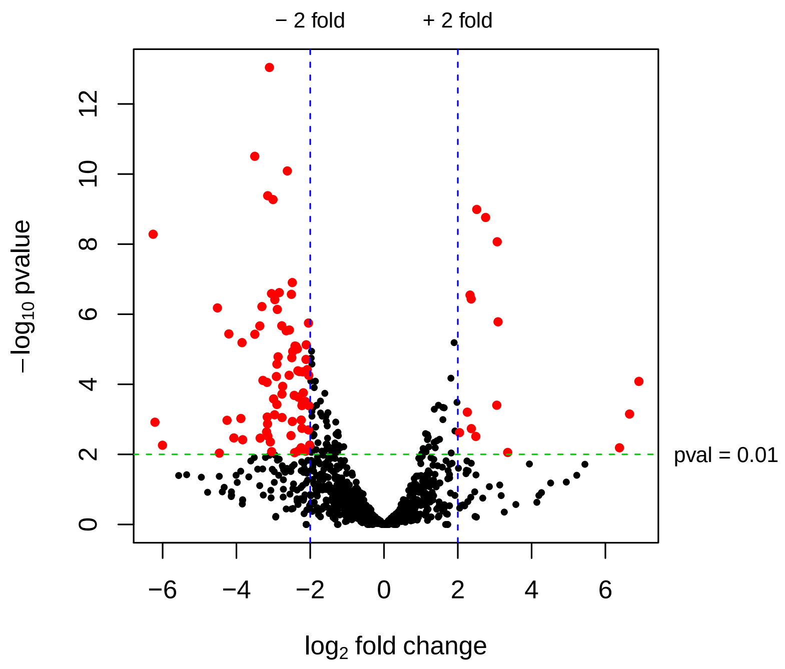


Figure 32: Volcano plot. The red points indicate genes-of-interest that display both large-magnitude fold-changes (x-axis) as well as high statistical significance ($-\log_{10}$ of p-value, y-axis). The dashed green-line shows the p-value cutoff (`pval = 0.01`) with points above the line having p-value < 0.01 and points below the line having p-value > 0.01. The vertical dashed blue lines shows 2-fold changes.

11.3 Gene clustering

To explore the degrees of similarity and more distant relationships among groups of closely related genes, it is often instructive to combine clustering methods with a graphical representation of the "primary" count table by means of the so-called clustering image map (CIM) or heatmap.

A CIM (or heatmap) is a two-dimensional, rectangular, colored grid, representing each data point (rectangle) with a color that quantitatively and qualitatively reflects the original experimental observations. The rows (and/or columns) of the matrix are rearranged (independently) according to some hierarchical clustering method, so that genes or groups of genes with similar expression patterns are adjacent. The computed dendrogram (tree) resulting from the clustering is added to a side of the image to indicate the relationships among genes (see Figure 33).

In order to test for differential expression, we operate on raw counts and use discrete distributions as described in the previous Section. However for visualization or clustering, it is advisable to work with transformed versions of the count data, since common statistical methods for clustering and ordination, work best for (at least approximately) homoskedastic data; this means that the variance of an observable quantity (i.e., here, the expression strength of a gene) does not depend on the mean. In RNA-Seq data, however, variance grows with the mean.

As a solution, *DESeq*, *DESeq2* and *edgeR* packages offers functions that give transformed data approximately homoskedastic:

`varianceStabilizingTransformation(...)`{*DESeq*, *DESeq2*}

`rlog(...)`{*DESeq2*}

`cpm(..., prior.count = 2, log = TRUE)`{*edgeR*}

Note that these transformations are provided for applications other than differential testing.

Making a CIM (heatmap)

`cim(...)`{*mixOmics*}

`heatmap(...)`{*stats*}

For illustration we use the results from *DESeq2* analysis. The first step to make a heatmap with RNA-seq data is to transform the raw counts of reads to (approximately) homoskedastic data:

```
rld = rlog(dds, blind = FALSE)
head(assays(rld)[[1]])
```

```
         A_1   A_2    A_3    B_1    B_2    B_3
gene1 3.5285 2.014 2.0094 3.8225 3.7885 3.9581
gene2 3.1669 3.872 4.0000 3.8338 5.4116 4.7196
gene3 2.1987 2.232 1.9313 1.3090 1.0763 1.4795
gene4 7.5230 7.296 7.3013 6.9305 6.8584 6.8765
gene5 4.2345 4.143 3.7839 3.2492 4.5035 3.4866
gene6 0.8037 1.064 0.2801 0.2783 0.8155 0.2754
```

Since the clustering is only relevant for genes that actually are differentially expressed, one usually carries it out only for a gene subset of most highly differential expression. Here, for demonstration, we select those genes that have adjusted p-values below 0.01.

```
de = (resDESeq2$padj < 0.01)
de[is.na(de)] = FALSE
```

There are 80 genes selected for the CIM.

```
cim(t(assay(rld)[de, ]), dendrogram = "column", xlab = "Genes", ylab = "Samples",
    col = colorRampPalette(brewer.pal(9, "Blues"))(255), symkey = FALSE, lhei = c(1, 3))
```

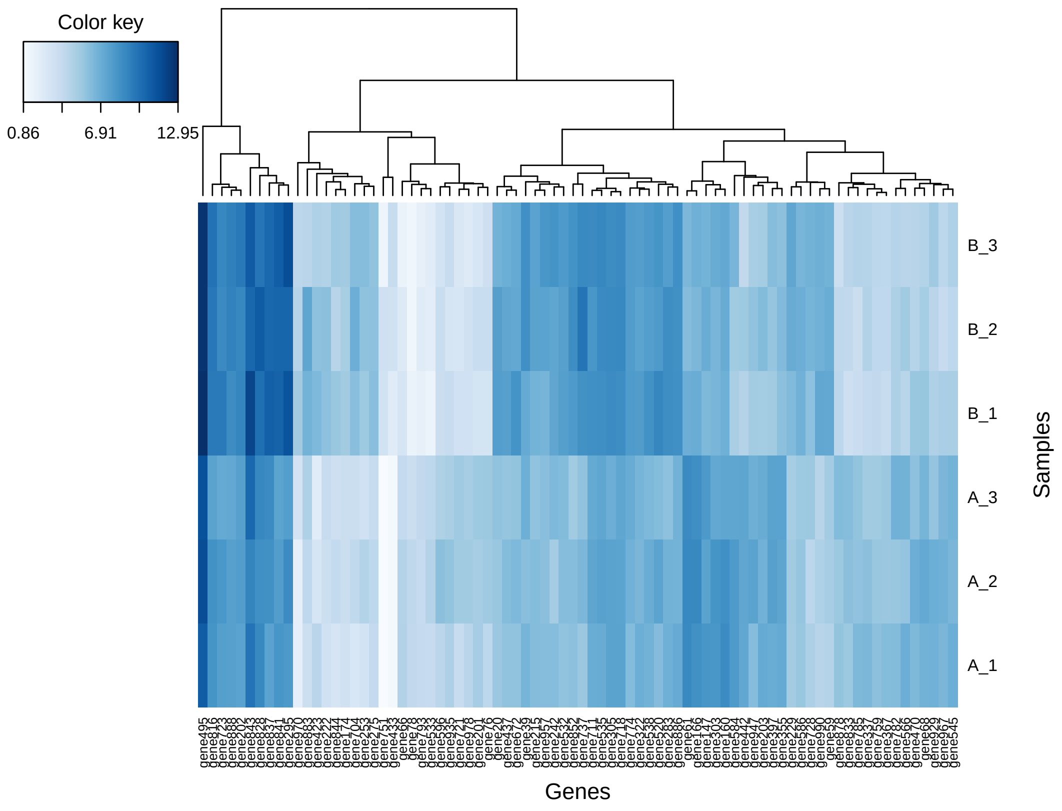


Figure 33: CIM for statistically significant genes. Rows represent individual samples and columns represent each gene. Each cell in the matrix represents the expression level of a gene feature in an individual sample. Dark-blue and white in cells reflect high and low expression levels, respectively, as indicated in the color key bar (`rlog`-transformed value). Hierarchical clustering was derived using the Euclidean distance as the similarity measure and the Ward clustering as agglomeration method.

11.4 Exporting results

An HTML report of the results with plots and sortable/filterable columns can be exported using the *ReportingTools* package on generate tables of differentially expressed genes as determined by the *DESeq*, *DESeq2* and *edgeR* packages. For a code example, see the *RNA-seq differential expression* vignette at the *ReportingTools* page, or the manual page for the `publish` method.

A plain-text file of the results can be exported using the base *R* functions `write.csv` or `write.delim`.

```
write.csv(as.data.frame(resDESeq2), file = "results_DESeq2.csv")
```

12 Improving test results

Because hypothesis tests are performed for gene-by-gene differential analyses, the obtained p-values must be adjusted to correct for multiple testing. However, procedures to adjust p-values to control the number of detected false positives often lead to a loss of power to detect truly differentially expressed genes due to the large number of hypothesis tests performed. To reduce the impact of such procedures, independent data filters are often used to identify and remove genes that appear to generate an uninformative signal; this in turn moderates the correction needed to adjust for multiple testing.

For illustration, we perform a standard analysis with *DESeq* package using a simulate dataset to look for genes that are differentially expressed between the "A" and "B" conditions, indicated by the factor variable `condition`.

```
conditions = rep(c("A", "B"), c(3, 3))
cds = newCountDataSet(countData, conditions)
cds = estimateSizeFactors(cds)
cds = estimateDispersions(cds)
cds = cds[rowSums(counts(cds)) > 0, ]
resNoFilt = nbinomTest(cds, "A", "B")
```

12.1 Independent filtering

Consider the MA-plot Figure 34.

```
DESeq::plotMA(resNoFilt)
```

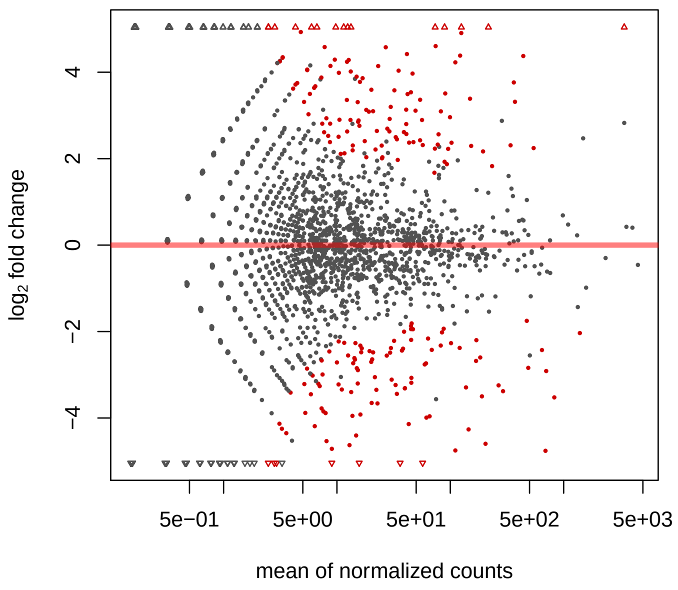


Figure 34: MA-plot from *DESeq* data analysis.

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:

```
## Create bins using the quantile function
pos = (resNoFilt$baseMean > 0)
qs = quantile(resNoFilt$baseMean[pos], 0:10/10)

## "cut" the genes into the bins
bins = cut(resNoFilt$baseMean[pos], qs)

## Rename the levels of the bins using the middle point
levels(bins) = paste0("~", round(0.5 * qs[-1] + 0.5 * qs[-length(qs)]))

## Calculate the ratio of p values less than 0.01 for each bin
ratios = tapply(resNoFilt$pval[pos], bins, function(p) mean(p < 0.01, na.rm = TRUE))

## Plot these ratios
barplot(ratios, xlab = "mean normalized count", ylab = "ratio of small p-values")
```

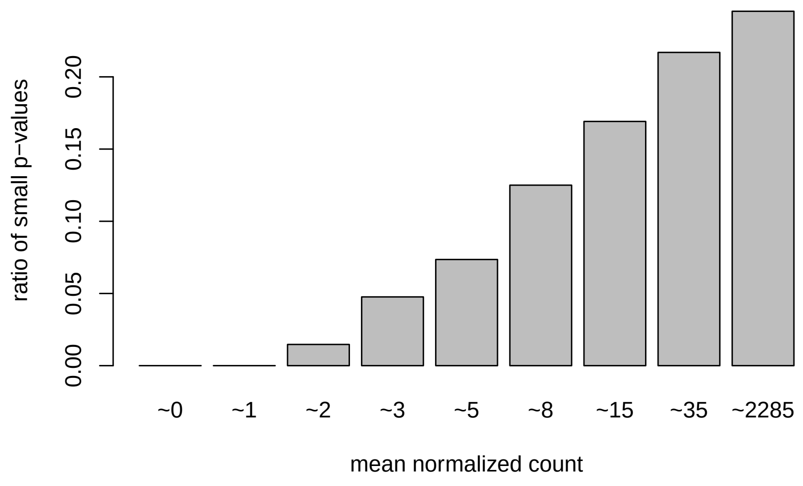


Figure 35: Ratio of small p values for groups of genes binned by mean normalized count.

The idea of independent filtering is to filter out those tests from the procedure that have no, or little chance of showing significant evidence, without even looking at their test statistic.

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.

Filtering criteria

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 FDR procedure.

A simple filtering criterion is the overall sum of normalized counts irrespective of biological condition.

```
rsFilt = rowSums(counts(cds, normalized = TRUE))
```

This filter is blind to the assignment of samples to the treatment and control group and hence independent.

Why does it work?

First, consider Figure 36, which shows that among the 40% of genes with lowest overall counts, `rsFilt`, there are essentially none that achieved an (unadjusted) p-value less than 0.009 (this corresponds to about 2 on the $-\log_{10}$-scale).

```
rank.scaled = rank(rsFilt)/length(rsFilt)

plot(rank.scaled, -log10(resNoFilt$pval), pch = 16, cex = 0.45, ylim = c(0, 20),
     xlab = "rank scaled to [0, 1]", ylab = expression(-log[10](p-value)))
```

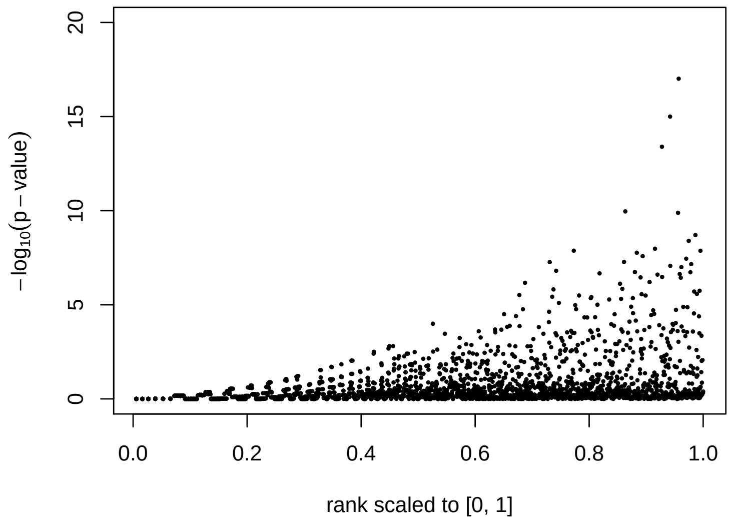


Figure 36: Scatterplot of the rank (scaled to $[0, 1]$) of the filter criterion `rsFilt` (x-axis) versus the negative logarithm of the p-values `resNoFilt$pval` (y-axis).

This means that by dropping the 40% genes with lowest `rsFilt`, we do not loose anything substantial from our subsequent results.

Now, consider the p-value histogram in Figure 37.

```
theta = 0.4
pass = (rsFilt > quantile(rsFilt, probs = theta))
pvals = resNoFilt$pval

h1 = hist(pvals[!pass], breaks = 30, plot = FALSE)
h2 = hist(pvals[pass], breaks = 30, plot = FALSE)
colori = c('do not pass' = "khaki", 'pass' = "powderblue")

barplot(height = rbind(h1$counts, h2$counts), beside = FALSE, col = colori, space = 0,
        main = "", ylab = "frequency")
text(x = c(0, length(h1$counts)), y = 0, label = c(0, 1), adj = c(0.5, 1.7), xpd = NA)
legend("top", fill = rev(colori), legend = rev(names(colori)))
```

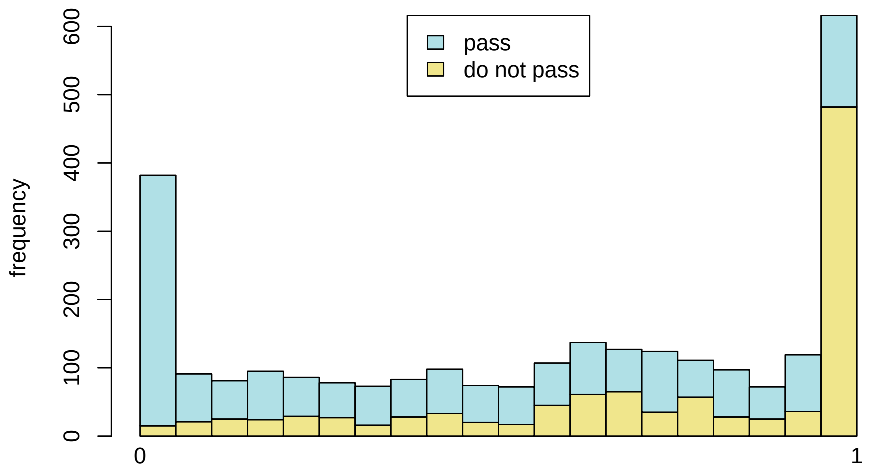


Figure 37: Histogram of p-values for all tests (`pvals`). The area shaded in blue indicates the subset of those that pass the filtering, the area in khaki those that do not pass.

It shows how the filtering ameliorates the multiple testing problem – and thus the severity of a multiple testing adjustment – by removing a background set of hypotheses whose p-values are distributed more or less uniformly in $[0, 1]$.

Above, we consider as a filter criterion `rsFilt`, the overall sum of counts (irrespective of biological condition), and remove the genes in the lowest 40% quantile (as indicated by the parameter `theta`). We perform the testing as before on the filtered data:

```
cdsFilt = cds[pass, ]
cdsFilt = newCountDataSet(counts(cdsFilt), conditions)
cdsFilt = estimateSizeFactors(cdsFilt)
cdsFilt = estimateDispersions(cdsFilt)
resFilt = nbinomTest(cdsFilt, "A", "B")
```

Let us compare the number of genes found at an FDR of 0.1 by this analysis with that from the previous one (`resNoFilt$padj`).

```
padjFiltForComparison = rep(1, length(resNoFilt$padj))
padjFiltForComparison[pass] = resFilt$padj
tab = table('no filtering' = resNoFilt$padj < 0.1,
            'with filtering' = padjFiltForComparison < 0.1)
dimnames(tab)[[1]] = dimnames(tab)[[2]] = c("no significant", "significant")
tab = addmargins(tab)

tab

                with filtering
no filtering     no significant significant  Sum
  no significant           2442          56 2498
  significant                 0         225  225
  Sum                      2442         281 2723
```

The analysis with filtering found an additional 56 genes, an increase in the detection rate by about 25%, while 225 genes were only found by the previous analysis.

A bad filter example

For comparison, suppose you had chosen a less useful filter statistic, say, the mean of normalized counts from "A" condition.

```
badfilter = rowMeans(counts(cds, normalized = TRUE)[, conditions == "A"])
```

The analogous scatterplot to that of Figure 36 is shown in Figure 38.

```
rank.scaled = rank(badfilter)/length(badfilter)

plot(rank.scaled, -log10(resNoFilt$pval), pch = 16, cex = 0.45, ylim = c(0, 20),
     xlab = "rank scaled to [0, 1]", ylab = expression(-log[10](p-value)))
```

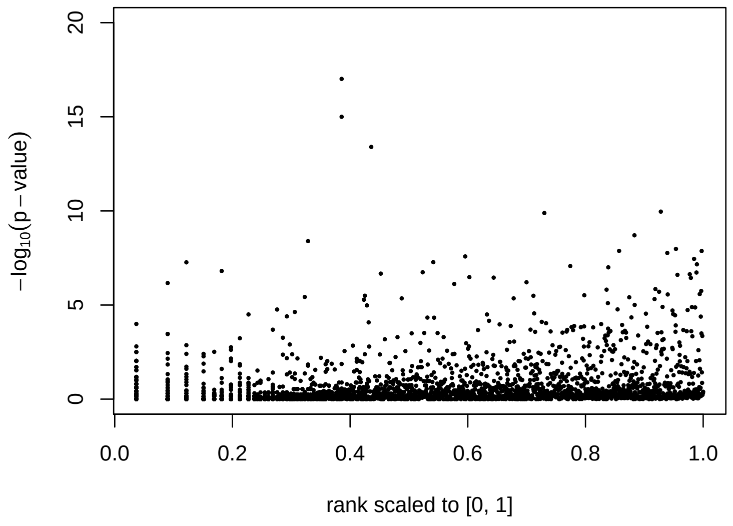


Figure 38: Scatterplot analogous to Figure 36, but with `badfilter`.

This demonstrates that non-independent filters for which dependence exists between the filter and test statistic (e.g. making use of condition labels to filter genes with average expression in at least one condition less than a given threshold), can in some cases lead to a loss of control of experiment-wide error rates.

12.2 How to choose the filter?

The `results` function of the *DESeq2* package performs independent filtering by default using the mean of normalized counts as a filter statistic. A threshold on the filter statistic is found which optimizes the number of adjusted p-values lower than a significance level `alpha`.

The independent filtering is performed using the `filtered_p` function of the *genefilter* package, and all of the arguments of `filtered_p` can be passed to the `results` function. Please refer to the vignette *Diagnostic plots for independent filtering* in the *genefilter* package for a discussion on

- best choice of filter criterion and
- the choice of filter cutoff.

Alternative filtering can be provided from *HTSFilter* package. *HTSFilter* implements a data-based filtering procedure based on the calculation of a similarity index among biological replicates for read counts arising from replicated transcriptome sequencing (RNA-seq) data.

The *HTSFilter* package is able to accommodate unnormalized or normalized replicated count data in the form of a *matrix* or *data.frame* (in which each row corresponds to a biological feature and each column to a biological sample), a *CountDataSet* (the S4 class associated with the *DESeq* package), one of the S3 classes associated with the *edgeR* package (*DGEList*, *DGEExact*, *DGEGLM*, and *DGELRT*), or *DESeqDataSet* (the S4 class associated with the *DESeq2* package).

Example: To filter high-throughput sequencing data in the form of a *CountDataSet* (the class used within the *DESeq* pipeline for differential analysis), we proceed as follows:

```
library(HTSFilter)

cdsHTSFilt = HTSFilter(cds, plot = FALSE)$filteredData
resHTSFilt = nbinomTest(cdsHTSFilt, "A", "B")

DESeq::plotMA(resHTSFilt)
```

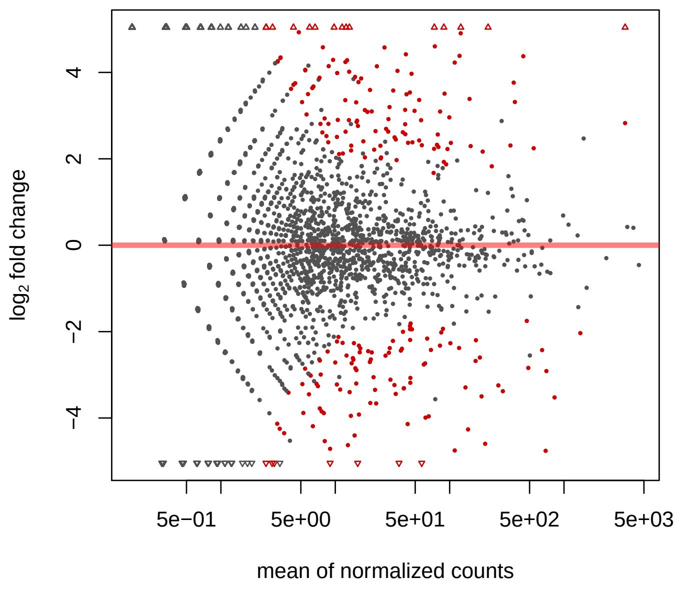


Figure 39: MA-plot from *DESeq* data analysis as in Figure 34 after *HTSFilter* filter.

13 Session information

The following is the session info that generated this tutorial:

```
sessionInfo()

R version 3.1.0 alpha (2014-03-23 r65266)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C               LC_TIME=fr_FR.UTF-8
 [4] LC_COLLATE=fr_FR.UTF-8     LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
 [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] HTSFilter_1.4.0           DESeq2_1.4.5              RcppArmadillo_0.4.450.1.0
 [4] Rcpp_0.11.2               GenomicRanges_1.16.4      GenomeInfoDb_1.0.2
 [7] IRanges_1.22.10           genefilter_1.46.1         xtable_1.7-4
[10] pasilla_0.4.0             matrixStats_0.10.0        mixOmics_5.0-3
[13] MASS_7.3-34               RColorBrewer_1.0-5        reshape_0.8.5
[16] ggplot2_1.0.0             edgeR_3.6.8               limma_3.20.9
[19] DESeq_1.16.0              lattice_0.20-29           locfit_1.5-9.1
[22] Biobase_2.24.0            BiocGenerics_0.10.0       knitr_1.6

loaded via a namespace (and not attached):
 [1] annotate_1.42.1      AnnotationDbi_1.26.0 BiocStyle_1.2.0      colorspace_1.2-4
 [5] DBI_0.3.1            digest_0.6.4         evaluate_0.5.5       formatR_1.0
 [9] geneplotter_1.42.0   grid_3.1.0           gtable_0.1.2         highr_0.3
[13] igraph_0.7.1         labeling_0.3         munsell_0.4.2        pheatmap_0.7.7
[17] plyr_1.8.1           proto_0.3-10         R.methodsS3_1.6.1    reshape2_1.4
[21] RGCCA_2.0            rgl_0.94.1131        RSQLite_0.11.4       scales_0.2.4
[25] splines_3.1.0        stats4_3.1.0         stringr_0.6.2        survival_2.37-7
[29] tools_3.1.0          XML_3.98-1.1         XVector_0.4.0
```

14 Bibliography

Experimental design

Auer, Paul L. and R. W. Doerge (June 2010). "Statistical Design and Analysis of RNA Sequencing Data". In: *Genetics* 185.2, pp. 405–416. ISSN: 1943-2631. DOI: 10.1534/genetics.110.114983. URL: http://dx.doi.org/10.1534/genetics.110.114983.

Fang, Zhide and Xiangqin Cui (May 2011). "Design and validation issues in RNA-seq experiments". In: *Briefings in Bioinformatics* 12.3, pp. 280–287. ISSN: 1477-4054. DOI: 10.1093/bib/bbr004. URL: http://dx.doi.org/10.1093/bib/bbr004.

Robles, Jose et al. (Sept. 2012). "Efficient experimental design and analysis strategies for the detection of differential expression using RNA-Sequencing". In: *BMC Genomics* 13.1, pp. 484+. ISSN: 1471-2164. DOI: 10.1186/1471-2164-13-484. URL: http://dx.doi.org/10.1186/1471-2164-13-484.

Input data and preparations

Brooks, Angela N. et al. (Feb. 2011). "Conservation of an RNA regulatory map between Drosophila and mammals". In: *Genome Research* 21.2, pp. 193–202. ISSN: 1549-5469. DOI: 10.1101/gr.108662.110. URL: http://dx.doi.org/10.1101/gr.108662.110.

Raw data filtering

Bottomly, Daniel et al. (Mar. 2011). "Evaluating Gene Expression in C57BL/6J and DBA/2J Mouse Striatum Using RNA-Seq and Microarrays". In: *PLoS ONE* 6.3, e17820+. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0017820. URL: http://dx.doi.org/10.1371/journal.pone.0017820.

Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth (Jan. 2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." In: *Bioinformatics (Oxford, England)* 26.1, pp. 139–140. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp616. URL: http://dx.doi.org/10.1093/bioinformatics/btp616.

Sultan, Marc et al. (Aug. 2008). "A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome." In: *Science (New York, N.Y.)* 321.5891, pp. 956–960. ISSN: 1095-9203. DOI: 10.1126/science.1160342. URL: http://dx.doi.org/10.1126/science.1160342.

Interpreting read counts

Bullard, James et al. (Feb. 2010). "Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments". In: *BMC Bioinformatics* 11.1, pp. 94+. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-94. URL: http://dx.doi.org/10.1186/1471-2105-11-94.

Oshlack, Alicia and Matthew J. Wakefield (Dec. 2009). "Transcript length bias in RNA-seq data confounds systems biology". In: *Biology Direct* 4.1, pp. 14–10. ISSN: 1745-6150. DOI: 10.1186/1745-6150-4-14. URL: http://dx.doi.org/10.1186/1745-6150-4-14.

Normalization

Anders, Simon and Wolfgang Huber (Oct. 2010). "Differential expression analysis for sequence count data". In: *Genome Biology* 11.10, R106+. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-10-r106. URL: http://dx.doi.org/10.1186/gb-2010-11-10-r106.

Bullard, James et al. (Feb. 2010). "Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments". In: *BMC Bioinformatics* 11.1, pp. 94+. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-94. URL: http://dx.doi.org/10.1186/1471-2105-11-94.

Robinson, Mark and Alicia Oshlack (Mar. 2010). "A scaling normalization method for differential expression analysis of RNA-seq data". In: *Genome Biology* 11.3, R25+. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-3-r25. URL: http://dx.doi.org/10.1186/gb-2010-11-3-r25.

Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth (Jan. 2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." In: *Bioinformatics (Oxford, England)* 26.1, pp. 139–140. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp616. URL: http://dx.doi.org/10.1093/bioinformatics/btp616.

Hypothesis testing

Benjamini, Yoav and Yosef Hochberg (1995). "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1, pp. 289–300. ISSN: 00359246. DOI: 10.2307/2346101. URL: http://dx.doi.org/10.2307/2346101.

Benjamini, Yoav and Daniel Yekutieli (2001). "The control of the false discovery rate in multiple testing under dependency". In: *Annals of Statistics*. Vol. 29, pp. 1165–1188. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.8492.

Goeman, Jelle J. and Aldo Solari (May 2014). "Multiple hypothesis testing in genomics". In: *Statistics in Medicine* 33.11, pp. 1946–1978. ISSN: 02776715. DOI: 10.1002/sim.6082. URL: http://dx.doi.org/10.1002/sim.6082.

Hochberg, Yosef (Dec. 1988). "A sharper Bonferroni procedure for multiple tests of significance". In: *Biometrika* 75.4, pp. 800–802. ISSN: 1464-3510. DOI: 10.1093/biomet/75.4.800. URL: http://dx.doi.org/10.1093/biomet/75.4.800.

Holm, Sture (1979). "A Simple Sequentially Rejective Multiple Test Procedure". In: *Scandinavian Journal of Statistics* 6.2, pp. 65–70. ISSN: 03036898. DOI: 10.2307/4615733. URL: http://dx.doi.org/10.2307/4615733.

Differential expression analysis

McCarthy, Davis J., Yunshun Chen, and Gordon K. Smyth (May 2012). "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation". In: *Nucleic Acids Research* 40.10, pp. 4288–4297. ISSN: 1362-4962. DOI: 10.1093/nar/gks042. URL: http://dx.doi.org/10.1093/nar/gks042.

Robinson, Mark D. and Gordon K. Smyth (Nov. 2007). "Moderated statistical tests for assessing differences in tag abundance." In: *Bioinformatics (Oxford, England)* 23.21, pp. 2881–2887. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btm453. URL: http://dx.doi.org/10.1093/bioinformatics/btm453.

— (Apr. 2008). "Small-sample estimation of negative binomial dispersion, with applications to SAGE data." In: *Biostatistics (Oxford, England)* 9.2, pp. 321–332. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxm030. URL: http://dx.doi.org/10.1093/biostatistics/kxm030.

Winkelmann, Rainer (2008). *Econometric analysis of count data*. URL: http://www.worldcat.org/isbn/9783540783893.

Improving test results

Bourgon, Richard, Robert Gentleman, and Wolfgang Huber (May 2010). "Independent filtering increases detection power for high-throughput experiments". In: *Proceedings of the National Academy of Sciences* 107.21, pp. 9546–9551. ISSN: 1091-6490. DOI: 10.1073/pnas.0914005107. URL: http://dx.doi.org/10.1073/pnas.0914005107.

Rau, Andrea et al. (Sept. 2013). "Data-based filtering for replicated high-throughput transcriptome sequencing experiments". In: *Bioinformatics* 29.17, pp. 2146–2152. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btt350. URL: http://dx.doi.org/10.1093/bioinformatics/btt350.