Statistical Testing, Including Multiple Testing and Filtering / Weighting

Bernd Klaus[1]

European Molecular Biology Laboratory (EMBL),
Heidelberg, Germany
[1]bernd.klaus@embl.de

April 4, 2016

Contents

1 Required packages and other preparations

```
set.seed(999)

library(gplots)
library(RColorBrewer)
library(ggplot2)
library(plyr)
library(dplyr)
library(magrittr)
library(tidyr)
library(mutoss)
library(qvalue)
library(st)
library(ALL)
library(hgu95av2.db)
library(genefilter)
library(IHW)
library("DESeq")
library("ggplot2")
library("methods")
library("airway")
library("DESeq2")
data("airway")
```

2 Introduction to statistical hypothesis testing

In science it is common to ask if two things are different Are men taller than women? Is the risk of cancer different in smokers and non-smokers? Is the probability of getting type II different for different genetic backgrounds? Is this gene differentially expressed in cancer? When we make two measurements and compare, we almost always see some difference. But will wee see it again if we measure again? If someone else measures? Statistical testing can help us answer this question.

Here we deal with questions related to the statistical testing biological hypothesis. Does the mean gene expression over ALL patients differ from that over AML patients? That is, does the mean gene expression level differ between experimental conditions? Is the mean gene expression different from zero? How can it be tested whether the frequencies of nucleotide sequences of two genes are different? What is the probability of a certain micro RNA to have more than a certain number of purines?

Many population parameters are used to define families of theoretical distributions. In any research (empirical) setting the specific values of such parameters are unknown so that these must be estimated. Once estimates are available it becomes possible to statistically test biologically important hypotheses. This lab gives several basic examples of statistical testing and some of its background.

As a conceptual example for a typical testing situation, let μ_0 be a number representing the hypothesized population mean by a researcher on the basis of experience and knowledge from the field. With respect to the population mean the null hypothesis can be formulated as $H_0 : \mu = \mu_0$ and the alternative hypothesis as $H_1 : \mu \neq \mu_0$. These are two statements of which the latter is the opposite of the first: Either H_0 or H_1 is true. The alternative hypothesis is true if $H_1 : \mu < \mu_0$ or $H_1 : \mu > \mu_0$ holds true. This type of alternative hypothesis is called "two-sided". In case $H_1 : \mu > \mu_0$ or $H_1 : \mu < \mu_0$, it is called "one-sided".

Such a null hypothesis will be statistically tested against the alternative using a suitable distribution of a statistic (e.g. standardized mean). After conducting the experiment, the value of the statistic can be computed from the data. By comparing the value of the statistic with its distribution, the researcher draws a conclusion with respect to the null hypothesis: H_0 is rejected or it is not. The probability to reject H_0, given the truth of H_0, is called the significance level which is generally denoted by α. We shall follow the habit in statistics to use $\alpha = 0.05$, but it will be completely clear how to adapt the procedure in case other significance levels are desired.

This workflow can be summarized as follows:

1. Set up hypothesis H_0 (that you want to reject)
2. Find a test statistic T that should be sensitive to (interesting) deviations from H_0
3. Figure out the null distribution of T, the distribution of T under the assumption that H_0 holds
4. Compute the actual value of T for the data at hand
5. Compute p–value = the probability of observing that value, or more extreme, assuming the null distribution.
6. Test Decision: Rejection of H_0 - yes / no ?

3 The two group comparison as a fundamental example for testing

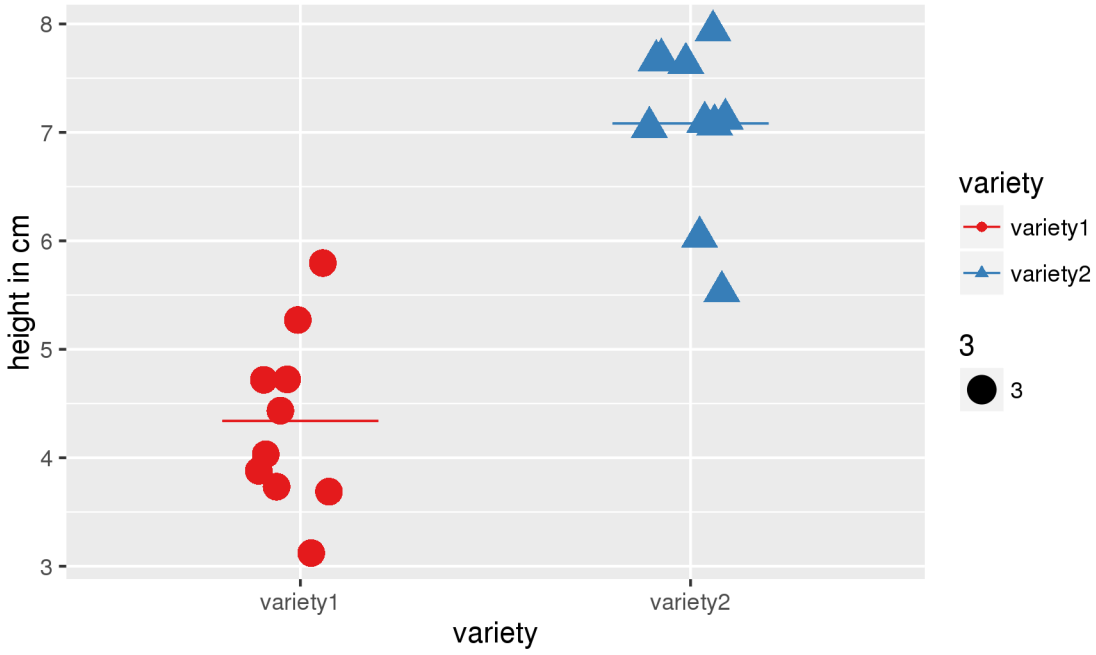
Imagine a researcher who would like to compare the height of two plant varieties. If she only takes one measurement of the plant height and observes a difference of say 2cm, it is impossible to say whether this difference is due to natural variation. On the other hand, if multiple plants of each variety are measured, and it turns out that the height differences are always somewhere around 2cm, the observed difference is less likely due to chance. This is illustrated in the figure below: the difference is strong relative to the variability between the measurements.

The data might look like this:

```
fBar <- function(x){
    tmp <- mean(x, na.rm = T)
    data.frame(ymax = tmp, y = tmp, ymin = tmp)
}

variety1 <- rnorm(10, mean = 5)
variety2 <- rnorm(10, mean = 7, sd = 1/sqrt(2))

plantData <- data.frame(height  = c(variety1,variety2) ,
                        variety = sort(rep(c("variety1", "variety2"),10)) )

head(plantData)
```

```
    height  variety
1    4.72 variety1
2    3.69 variety1
3    5.80 variety1
4    5.27 variety1
5    4.72 variety1
6    4.43 variety1
```

```
(ggplot(aes(x = variety, y = height, color = variety, shape = variety, size = 3),
       data = plantData)

    + geom_jitter(width = 0.3, height = 0)


    + scale_color_brewer(palette = "Set1")

    + ylab("height in cm")

    + geom_errorbar(stat = "summary", width = 0.4,
                    fun.data = fBar, size = 0.4))
```



Side Note: Technical vs. biological replicates

When referring to replicates it is important to distinguish between biological and technical replicates. Technical replicates refer to experimental samples isolated from one biological sample, e.g. extracting RNA from the cells of a mouse and then preparing 3 sequencing libraries from this while a biological replication means extracting RNA from three different mice for the comparisons of interest. It is not sufficient "to pipette an experiment again" since this is not biological, but merely a “technical" replication. In general, technical replicates tend to show less variability than biological replicates, thus leading to false positive results.

3.1 How to test for differences via permutations

The observed mean difference between the the two varieties is 2.743 cm. How easy would it be for a difference of 2.743 cm minutes to occur just by chance?

To answer this, we suppose there really is no difference between the two groups, that variety1 and variety2 are just labels. So what would happen if we assign labels randomly? How often would a difference like 2.743 cm occur?

We'll pool all twenty observations, randomly pick 10 of them to label basic and label the rest extended, and compute the difference in means between the two groups. We'll repeat that many times, say ten thousand, to get the permutation distribution shown below. The observed statistic 2.743 cm is also shown; the fraction of the distribution to the right of that value is the probability that random labeling would give a difference that large.

In a permutation test, we obtain the null distribution from the data, rather than analytically as e.g. in a t–test.

3.2 Run the permutation test

```
## helper function to compute permutation p-value

mDiff <- function(data, group){
  data$group <- NULL
  data$group <- group
  tmp <-  data %>%
    group_by(group)  %>%
    summarize(m = mean(data, na.rm=TRUE))
  as.numeric(tmp[2,"m"] - tmp[1,"m"])
}


## function to compute the permutation test
permTestTwoGroups <- function(group1, group2,
                              twoSided = TRUE, permutations = 1e4){

  stopifnot(is.numeric(group1),  is.numeric(group2),
            length(group1) > 0,  length(group2) > 0,
            is.vector(group1), is.vector(group2),
           is.logical(twoSided))

  inputData <- data.frame(data  = c(group1, group2),
                      group = rep(c("group1", "group2"),
                                  c(length(group1), length(group2))))

  # compute the observed difference between the groups
  obsDiff  <-  mDiff(inputData, inputData$group)

  # compute sampling distribution and p--value
  samplingDist <- c(replicate(as.integer(permutations),
                              mDiff(inputData, sample(inputData$group))),
                              obsDiff)

  # compute two-sided p-value
  pvalP <- 2*min(1 - ecdf(samplingDist)(abs(obsDiff)),
                 ecdf(samplingDist)(abs(obsDiff)))

  # create plot of the sampling distribution
```

```
  samplingDistPlot <- (qplot(samplingDist, fill = I("orange4"),
                             main = "Sampling distribution mean difference",
                              binwidth = 0.1)
  +  geom_vline(xintercept = obsDiff, size = 2))

  return(list(
    samplingDist = samplingDist,
    samplingDistPlot = samplingDistPlot,
    obsDiff = obsDiff,
    pval = pvalP
  ))

}
```

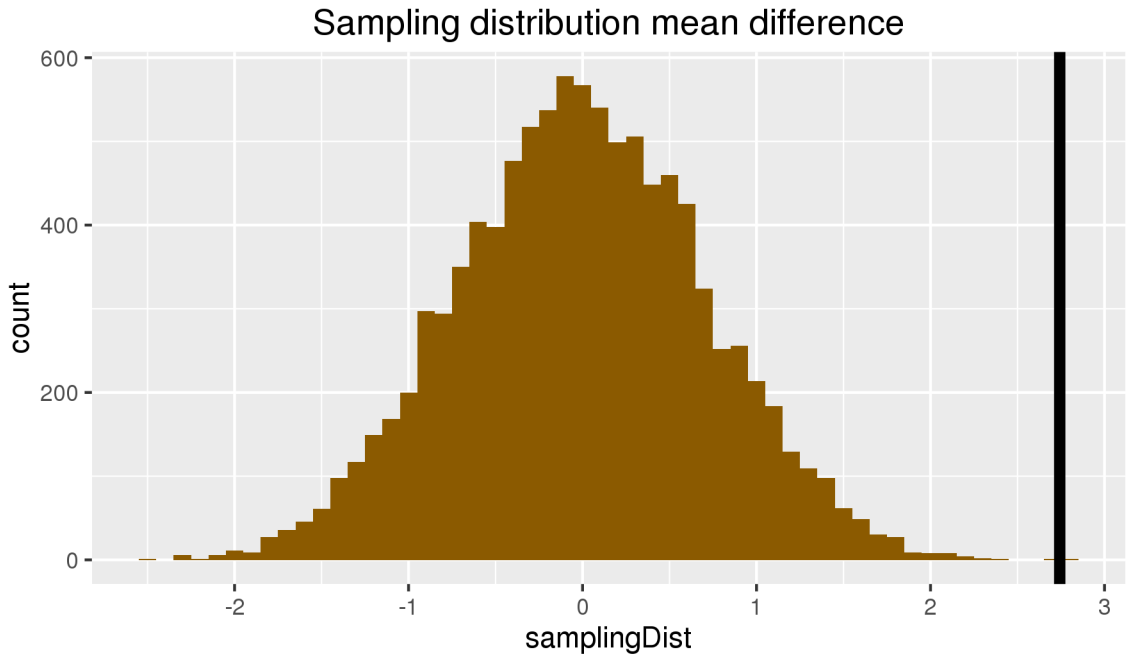
We now compute a difference for each label permutation and plot it. We compute a **two-sided p-value** by looking how many of the computed differences are less than the observed one of **mean(variety2) - mean(variety1)** (lower p–value) and how many are greater than '**mean(variety2) - mean(variety1)**. The two–sided p–value corresponds to a test of the null hypothesis that the mean difference between the two varieties is different from zero.

The lower p–value corresponds to a test of the null hypothesis that the mean difference is less than zero while the upper p–value corresponds to a test of the null hypothesis that the mean difference is greater than zero. The lower and upper p–values correspond to so–called **one–sided** tests. In order to compute the two–sided p–value, we compute both one–sided ones and then take twice the smaller one.

```
## draw 1e4 times a permutation of the  sample labels  and compute the
## two sided p--value


testResult <- permTestTwoGroups(group1 = variety1,
                  group2 = variety2)

testResult$pval
```

```
   [1] 2e-04
```

```
testResult$samplingDistPlot
```



In this case, the probability, the p–value, is 2×10^{-4}; it would be rare for a difference this large to occur by chance. The distribution that is shown in the figure is called a **sampling distribution**. It describes how the our **test statistic** would be distributed if the **null hypothesis** was true, i.e. if there was no difference between the the two varieties. The lower the variability of the data and the higher the sample size, the "thinner" the sampling distribution will be.

The p–value gives the probability to observe a difference of 2.743 or greater assuming that the null hypothesis is true. If this probability is very low, we can be confident that the null hypothesis is not true and thus the alternative is, i.e. that there is actually a difference between the height of the two varieties.

The name "permutation test" stems from the fact that we picked n_1 observations without replacement to label as the first sample, and labelled the others as the second sample. This is equivalent to randomly permuting all labels, hence the name. If we use all possible permutations for the test, the test is also called an exact test. However, this is computationally unfeasible for large sample sizes.

3.3 Summary: two–sample permutation test recipe

(*a*) Pool the values of the two groups
(*b*) repeat a large number of times (> 10 000)
- Draw a resample of size n_1 without replacement
- Use the remaining n_2 observations for the other sample
- Calculate the difference in means, or another statistic that com- pares samples
- Plot a histogram of the random statistic values; show the observed statistic.
- Calculate the p–value as the fraction of times the random statistics exceed or equal the observed statistic

4 The two sample t–test

Instead of a permutation test, we can use a t–test to test the difference between the two varieties. In contrast to the permutation test, the sampling distribution of the mean is obtained analytically, via the assumption of a normal distribution for both input groups. We will discuss the normal distribution next.

4.1 The Normal Distribution

The normal distribution is of key importance because it is assumed for many data generating processes. Among other things, we will look at (reprocessed) gene expression values than can be seen as realizations of a random variable X having a normal distribution.

Equivalently, one says that the data values are members of a normally distributed population with mean μ (mu) and variance σ^2 (sigma squared). It is good custom to use Greek letters for population properties and $N(\mu, \sigma^2)$ for the normal distribution. The value of the distribution function is given by $P(X \leq x)$, the probability of the population to have values smaller than or equal to x. Various properties of the normal distribution are illustrated by the examples below.

4.2 Example: Explore the Normal Distribution

To view members of the normal distribution load the *TeachingDemos*" ' package and enter the the command `vis.normal()` to launch an interactive display of densities of the normal distribution, i.e. bell-shaped curves. The curves are symmetric around μ and attain a unique maximum at $x = \mu$. If x moves further away from the mean μ, then the curves moves to zero so that extreme values occur with small probability. Move the mean and the standard deviation from the left to the right to explore their effect on the shape of the normal distribution. In particular, when the mean μ increases, then the distribution moves to the right. If σ is small/large, then the distribution is steep/flat.

4.3 The ALL data

The ALL data consist of microarrays from 128 different individuals with acute lymphoblastic leukemia (ALL). There are 95 samples with B-cell ALL and 33 with T-cell ALL and because these are different tissues and quite different diseases we consider them separately and focus on the B-cell ALL tumors.

An interesting subset, with two groups having approximately the same number of samples in each group, is the comparison of the B-cell tumors found to carry the BCR/ABL mutation to those B-cell tumors with no observed cytogenetic abnormalities. These samples are labeled BCR/ABL and NEG in the mol.biol covariate. The BCR/ABL mutation, also known as the Philadelphia chromosome, was the first cytogenetic aberration that could be associated with the development of cancer, leading the way to the current understanding of the disease. In tumors harboring the BCR/ABL translocation a short piece of chromosome 22 is exchanged with a segment of chromosome 9. As a consequence, a constitutively active fusion protein is transcribed which acts as a potent mitogene, leading to uncontrolled cell division. Not all leukemia tumors carry the Philadelphia chromosome; there are other mutations that can be responsible for neoplastic alterations of blood cells, for instance a translocation between chromosomes 4 and 11 (ALL1/AF4).

From the data, we look at the expression of the gene BCL2. The following code chunk shows the preprocessing of the data. We only select the B-Cell tumors and focus on the ones with/without a translocation.

```
data("ALL")
bALL <- ALL[, substr(ALL$BT,1,1) == "B"]
fusALL <- bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
fusALL$mol.biol <- factor(fusALL$mol.biol)
fusALL
```

```
   ExpressionSet (storageMode: lockedEnvironment)
   assayData: 12625 features, 79 samples
     element names: exprs
   protocolData: none
   phenoData
     sampleNames: 01005 01010 ... 84004 (79 total)
     varLabels: cod diagnosis ... date last seen (21 total)
     varMetadata: labelDescription
   featureData: none
   experimentData: use 'experimentData(object)'
```

```
     pubMedIds: 14684422 16243790
   Annotation: hgu95av2
```

```
sample_n(pData(fusALL), 10)
```

```
          cod  diagnosis sex age BT remission          CR    date.cr t(4;11)
   26001 26001  9/27/1997   M  21 B2        CR          CR 12/11/1997      NA
   28021 28021  3/18/1998   F  54 B3        CR DEATH IN CR  5/22/1998   FALSE
   62003 62003  12/4/1998   M  53 B4        CR          CR  1/28/1999   FALSE
   43004 43004   2/4/1997   F  37 B3        CR          CR   4/1/1997      NA
   11005 11005   6/1/1998   M  27 B2        CR DEATH IN CR   8/3/1998   FALSE
   24022 24022 12/21/1999   F  32 B4       REF         REF       <NA>   FALSE
   08012  8012 10/22/1998   M  55 B3        CR          CR   1/9/1999   FALSE
   12019 12019   9/4/1997   M  53 B2        CR          CR 11/11/1997   FALSE
   14016 14016  5/27/1999   M  53 B2      <NA>        <NA>       <NA>   FALSE
   06002  6002  3/19/1997   M  15 B2        CR          CR   6/9/1997   FALSE
         t(9;22) cyto.normal           citog mol.biol fusion protein mdr   kinet
   26001      NA          NA            <NA>      NEG           <NA> POS dyploid
   28021    TRUE       FALSE   t(9;22)+other  BCR/ABL      p190/p210 NEG hyperd.
   62003    TRUE       FALSE   t(9;22)+other  BCR/ABL           p210 NEG hyperd.
   43004      NA          NA            <NA>      NEG           <NA> NEG dyploid
   11005   FALSE       FALSE del(7q) + altro  BCR/ABL           p190 NEG dyploid
   24022    TRUE       FALSE         t(9;22)  BCR/ABL           p190 POS dyploid
   08012   FALSE       FALSE     simple alt.      NEG           <NA> NEG dyploid
   12019   FALSE        TRUE          normal      NEG           <NA> POS dyploid
   14016    TRUE       FALSE         t(9;22)  BCR/ABL           p210 NEG    <NA>
   06002   FALSE        TRUE          normal      NEG           <NA> NEG dyploid
          ccr relapse transplant               f.u date last seen
   26001  TRUE   FALSE      FALSE               CCR      7/31/2002
   28021 FALSE   FALSE      FALSE DEATH IN CR (ICR)           <NA>
   62003 FALSE    TRUE      FALSE               REL       8/8/2000
   43004  TRUE   FALSE      FALSE               CCR      3/20/2001
   11005 FALSE   FALSE      FALSE       DEATH IN CR           <NA>
   24022    NA      NA         NA              <NA>           <NA>
   08012 FALSE    TRUE      FALSE               REL       4/9/1999
   12019  TRUE   FALSE      FALSE               CCR       6/6/2002
   14016    NA      NA         NA              <NA>           <NA>
   06002 FALSE    TRUE      FALSE               REL      3/18/1998
```

```
groupsALL <- fusALL$mol.biol
expALL <- exprs(fusALL)

anno_fusALL <- plyr::ddply(AnnotationDbi::select(hgu95av2.db,
                                keys=rownames(expALL),
                                columns = c("SYMBOL", "GENENAME", "ENSEMBL"),
                                keytype="PROBEID"), "PROBEID", function(X){X[1,]})
```

```
  'select()' returned 1:many mapping between keys and columns
```

Now the the sample group is coded in the vector `groupsALL`, the expression data is in `expALL` and the annotation of the probes is in `anno_fusALL`

```
head(groupsALL)
```

```
   [1] BCR/ABL NEG     BCR/ABL NEG     NEG     NEG
   Levels: BCR/ABL NEG
```

```
head(expALL[, 1:5])
```

```
            01005 01010 03002 04007 04008
   1000_at    7.60  7.48  7.57  7.91  7.07
   1001_at    5.05  4.93  4.80  4.84  5.15
   1002_f_at  3.90  4.21  3.89  3.42  3.95
   1003_s_at  5.90  6.17  5.86  5.69  6.21
   1004_at    5.93  5.91  5.89  5.62  5.92
   1005_at    8.57 10.43  9.62  9.98 10.06
```

```
head(anno_fusALL)
```

```
       PROBEID  SYMBOL
   1   1000_at   MAPK3
   2   1001_at    TIE1
   3 1002_f_at CYP2C19
   4 1003_s_at   CXCR5
   5   1004_at   CXCR5
   6   1005_at   DUSP1
                                                    GENENAME
   1                             mitogen-activated protein kinase 3
   2 tyrosine kinase with immunoglobulin-like and EGF-like domains 1
   3          cytochrome P450, family 2, subfamily C, polypeptide 19
   4                             chemokine (C-X-C motif) receptor 5
   5                             chemokine (C-X-C motif) receptor 5
   6                                 dual specificity phosphatase 1
            ENSEMBL
   1 ENSG00000102882
   2 ENSG00000066056
   3 ENSG00000165841
   4 ENSG00000160683
   5 ENSG00000160683
   6 ENSG00000120129
```

4.4 Example: A Normal Model for gene expression of BCL2

Here we look at the gene expression values for the gene BCL2, which is in row 1152 of the data set.

```
anno_fusALL[1152,]
```

```
         PROBEID SYMBOL                                    GENENAME
   1152 2039_s_at    FYN FYN proto-oncogene, Src family tyrosine kinase
              ENSEMBL
   1152 ENSG00000010810
```

This gene encodes an integral outer mitochondrial membrane protein that blocks the apoptotic death of some cells such as lymphocytes. Constitutive expression of BCL2 is thought to be the cause of follicular lymphoma. We now develop a normal distribution model for the translocation group of the ALL data.

Suppose that the expression values of the ALL group of gene BCL2 can be represented by X which is distributed as $N(8.6, 0.5)$. From the graph of its density function, it can be observed that it is symmetric and bell-shaped around $\mu = 8.6$.

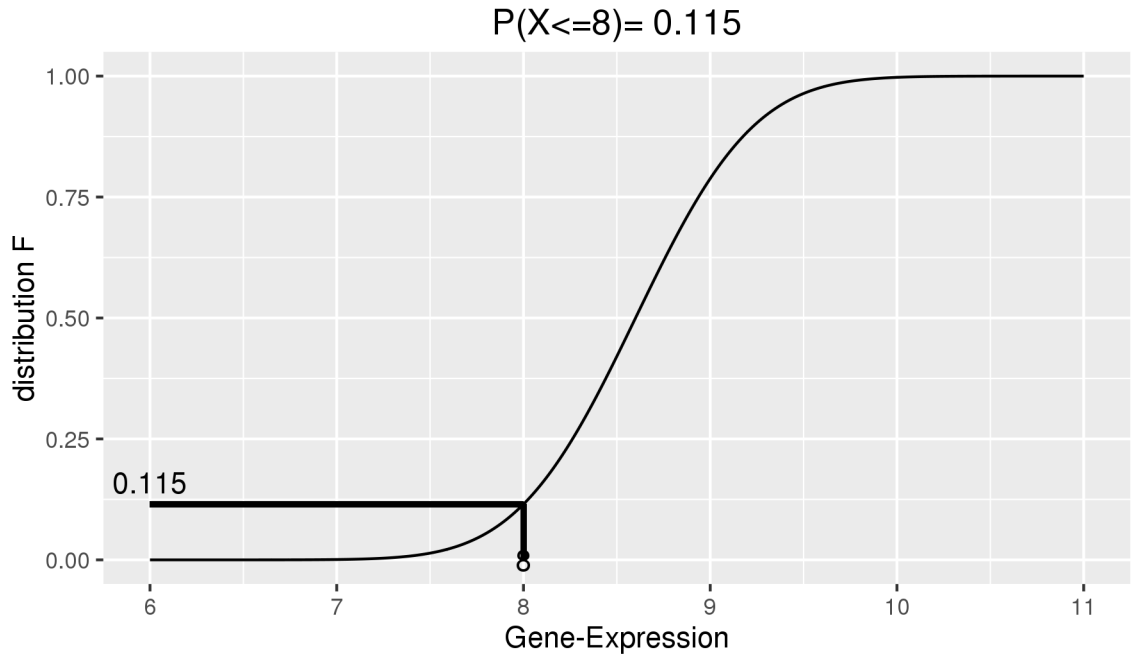
A density function may very well be seen as a histogram with arbitrarily small bars (intervals). The probability that the expression values are less than 8 is $P(X < 8) =$ `pnorm(8, 8.6, 0.5)` $= 0.115$.

The figure next to it illustrates the value 0.115 of the **cumulative distribution function (cdf)** at $x = 8$. It corresponds

to the area of the green colored surface below the graph of the density function in the figure.

```
f <-function(x){dnorm(x, 8.6, 0.5)}
F <-function(x){pnorm(x, 8.6, 0.5)}

x <- seq(6,11,0.01)
dataGG <- data.frame(x = x, y = f(x))
dataGG <- mutate(dataGG,  area = ifelse(x < 8, "in", "out"  ))

p<-qplot(data = dataGG, x = x, y = y, geom="line")
p<-p + geom_area(aes(ymax = y, fill = area)) +  guides(fill=FALSE)
p<-p + xlab("Gene-Expression") + ylab("density f") + annotate("text", x = 8, y = 0, label = "8")
p + labs(title = "P(X<=8)= 0.115") + scale_fill_brewer(palette = "Dark2")
```


```
dataGG = data.frame(x = x, y = F(x))
dataGG <- mutate(dataGG,  area = ifelse(x < 8, "in", "out"  ))

p<-qplot(data = dataGG, x = x, y = y, geom="line")
p<- p + annotate("text", x = 8, y = 0, label = "8")
p<- p   + annotate("text",y = .16, x = 6, label = "0.115")

p <- p + geom_segment(y=0, yend = F(8), x=8, xend = 8, size = I(1))
p <- p + geom_segment(y=F(8), yend = F(8), x=6, xend = 8, size = I(1))
p + labs(title = "P(X<=8)= 0.115") + xlab("Gene-Expression") + ylab("distribution F")
```



The probability that the expression values are greater than 9 is $P(X \geq 9) =$

```
1 - pnorm(9, 8.6, 0.5)
```

```
   [1] 0.212
```

The probability that X is between 8 and 9 equals $P(8 \leq X \leq 9) =$

```
pnorm(9, 8.6, 0.5) - pnorm(8, 8.6, 0.5)
```

```
   [1] 0.673
```

The graph of the distribution function shows that it is strictly increasing. For example, the exact value for the quantile $x_{0.025}$ can be computed by

```
qnorm(0.025,8.6,0.5)
```

```
   [1] 7.62
```

That is, the quantile $x_{0.025} = 0.92$. Hence, it holds that the probability of observing values less than 0.92 equals 0.025, that is $P(X \leq 0.92) = 0.025$, as can be verified by ' pnorm(0.92, 8.6, 0.5)'.

When X is distributed as $N(8.6, 0.5)$, then the population mean is 8.6 and the population standard deviation 0.5. To verify this we draw a random sample of size 1000 from this population by

```
x <- rnorm(1000,8.6,0.5)
```

The estimates

```
mean(x)
```

```
   [1] 8.59
```

```
#and
sd(x)
```

```
   [1] 0.485
```

are close to their population values $\mu = 8.6$ and $\sigma = 0.5$.

Exercise: Normal Model for a gene

Suppose that the distribution of the expression values for a gene is distributed according to $N(1.6, 0.42)$.

1. Compute the probability that the expression values are less than 1.2. 2. What is the probability that the expression values are between 1.2 and 2.0? 3. What is the probability that the expression values are between 0.8 and 2.4? 4. Compute the exact values for the quantiles $x_{0.025}$ and $x_{0.975}$.

5. Use `rnorm` to draw a sample of size 1000 from the population and compare the sample mean and standard deviation to that of the population.

4.5 Conducting a t-test

Suppose that gene expression data from two groups of patients (experimental conditions) are available and that the hypothesis is about the difference between the population means μ_1 and μ_2 . In particular, $H_0 : \mu_1 = \mu_2$ is to be tested against $H_1 : \mu_1 \neq \mu_2$. These hypotheses can also be formulated as $H_0 : \mu_1 - \mu_2 = 0$ and $H_1 : \mu_1 - \mu_2 \neq 0$. Suppose that gene expression data from the first group are given by $\{x_1, \ldots, x_n\}$ and that of the second by $\{y_1, \ldots, y_m\}$. Let $\bar{x}$ be the mean of the first and $\bar{y}$ that of the second, s_1 the variance of the first and s_2 that of the second. Then the t-statistic can be formulated as

$$t = \frac{\bar{x} - \bar{y} - (\mu_1 - \mu_2)}{s_1/\sqrt{n} + s_2/\sqrt{m}}$$

The decision procedure with respect to the null-hypothesis is completely analogous to the permutation test. However, the sampling distribution is **found analytically based on assumptions on the data and not by permutations**.

Note that the t–value is large if the difference between x and y is large, the standard deviations s_1 and s_2 are small, and the sample sizes are large. This means for example that higher sample sizes allow you to detect more subtle mean differences. The t–test with the assumptions of unequal variances in the groups is also known as the Welch two-sample t–test and is routinely performed.

If $s_1 = s_2$ the variance estimator of the t–test and the calculation of the degrees of freedom of the t–distribution changes slightly. This test is available by specifying `var.equal = TRUE` when calling the function `t.test`.

Example: Comparing BCL2 between BCR/ABL and NEG

The gene BCL2 plays an important role with respect to discriminating BCR/ABL from NEG patients. The null hypothesis of equal means can be tested by the function `t.test` and the appropriate factor and specification to separate the groups. (`var.equal=FALSE` by default).

```
t.test(expALL[1152,] ~ groupsALL)
```

```
    Welch Two Sample t-test

   data:  expALL[1152, ] by groupsALL
   t = 5, df = 70, p-value = 1e-05
   alternative hypothesis: true difference in means is not equal to 0
   95 percent confidence interval:
    0.392 0.956
   sample estimates:
   mean in group BCR/ABL     mean in group NEG
                    8.61                  7.93
```

```
### alternative call
t.test(expALL[1152, groupsALL == "BCR/ABL"],
       expALL[1152, groupsALL == "NEG"] )


    Welch Two Sample t-test

   data:  expALL[1152, groupsALL == "BCR/ABL"] and expALL[1152, groupsALL == "NEG"]
   t = 5, df = 70, p-value = 1e-05
   alternative hypothesis: true difference in means is not equal to 0
   95 percent confidence interval:
    0.392 0.956
   sample estimates:
   mean of x mean of y
        8.61      7.93
```

The t-value is quite large, indicating that the two means x and y differ largely from zero relative to the corresponding standard error . Since the p–value is extremely small, the conclusion is to reject the null–hypothesis of equal means. The data provide strong evidence that the population means do differ.

5 Wilcoxon rank test

In case the data are normally distributed with equal variance, the t–test is an optimal test for testing $H_0 : \mu_1 = \mu_2$ against $H_1 : \mu_1 \neq \mu_2$. If, however, the data are not normally distributed due to skewness or otherwise heavy tails, then this optimality does not hold anymore and there is no guarantee that the significance level of the test equals the intended level α. Usually, one will loose power if the normality assumption is validated, i.e. the α will be inflated.

For this reason rank type of tests are developed for which on beforehand no specific distributional assumptions need to be made. In the example below we shall concentrate on the two-sample Wilcoxon test.

To broaden our view we switch from hypotheses about means to those about distributions. An alternative hypothesis may then be formulated as that the distribution of a first group lays to the left of a second.

To set the scene let the gene expression values of the first group (x_1 to x_m) have distribution F and those of the second group (y_1 to y_n) distribution G. The null hypothesis is that both distributions are equal ($H_0 : F = G$) and the alternative that they are not.

For example that the x's are smaller (or larger) than the y's. By the two-sample Wilcoxon test the data $x_1,\ldots,x_m$, $y_1,\ldots,y_n$ are ranked and the rank numbers of the x's are summed to form the statistic W after a certain correction.

The idea is that if the ranks of x's are smaller than those of the y's, then the sum is small. The distribution of the sum of ranks is known so that a p–value can be computed on the basis of which the null hypothesis is rejected if it is smaller than the significance level α.

Example: BCL2 gene

The null hypothesis that the expression values for gene BCL2 are equally distributed for the ALL patients and the AML patients can be tested by the built–in–function `wilcox.test`, as follows.

```
wilcox.test(expALL[1152,] ~ groupsALL)


    Wilcoxon rank sum test

   data:  expALL[1152, ] by groupsALL
```

```
   W = 1000, p-value = 3e-05
   alternative hypothesis: true location shift is not equal to 0
```

Since the p–value is much smaller than $\alpha = 0.05$, the conclusion is to reject the null–hypothesis of equal distributions.

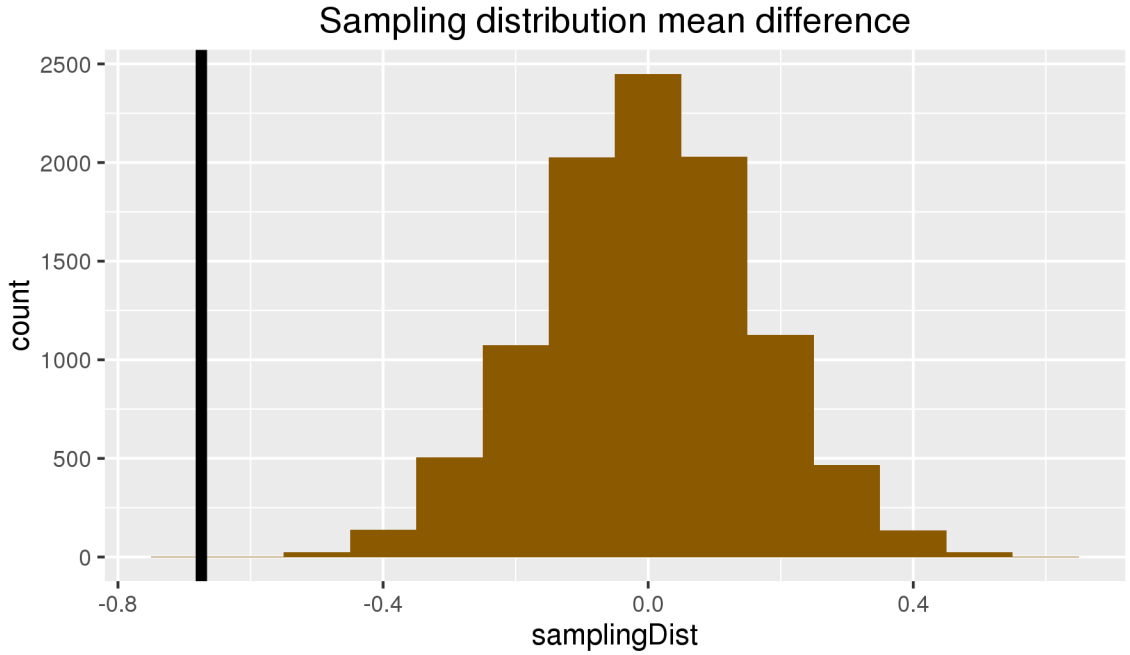
A permutation test for BCL2

We can of course also test the BCL2 gene for differential expression using a permutation test:

```
pTest <- permTestTwoGroups(expALL[1152, groupsALL == "BCR/ABL"],
       expALL[1152, groupsALL == "NEG"], permutations = 1e4 )

pTest$pval
```

```
   [1] 0
```

```
pTest$samplingDistPlot
```



The result is similar to the other tests performed.

5.1 Where permutation test do not apply

Permutation tests are helpful and their widespread use has been made possible by the computer power we have at our disposal today. However, they are not a panacea.

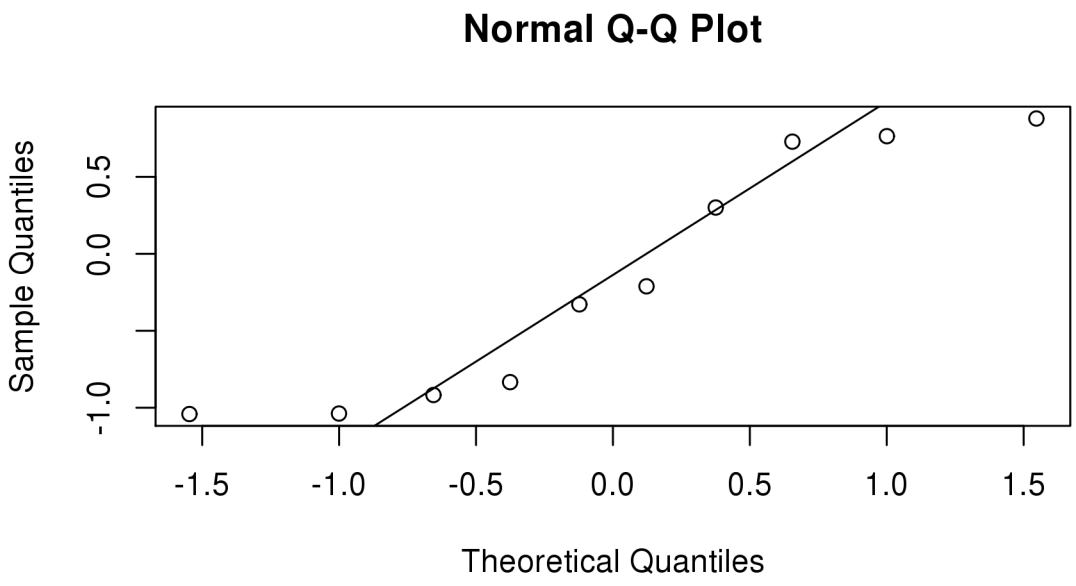
We have seen that in the two groups case, permutation testing is straightforward. The same is true for the testing of the dependence between two variables for example, while other situations are not easily covered by a permutation test, such testing a single sample mean. Also, we cannot perform a permutation test of the mean difference if the variance in the two groups differ, since then we cannot pool the data.

5.2 Caveat: Wilcoxon test vs. t–test

The t–test requires normally distributed data in order to be valid. In practice, this leads to many people prefer Wilcoxon tests over the t–test. However, the problem with the Wilcxocon test is that it implicitly assumes **equal variances** in the two groups, since it only tests for shifts in location. So if the variances are not equal, it can give misleading results.

It actually often leads to overly low p–values. Below is a little simulation study showing this effect. Two groups of 10 normally distributed values are simulated, one with a standard deviation of 1 and another with a standard deviation of 15. There is no difference between the groups (both have mean 0), although the standard deviations are very different, so we expect a proportion of 5% significant p–values at an α–level of 5%.

```
x <- rnorm(10)
qqnorm(x)
qqline(x)
```

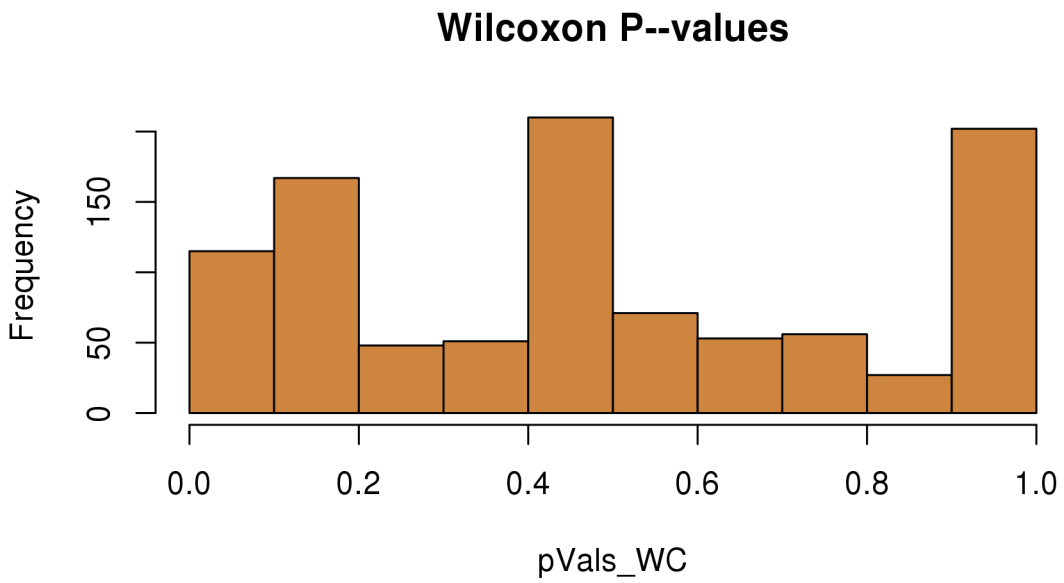



```
y <- rnorm(10)
wilcox.test(x,y)
```

```
    Wilcoxon rank sum test

   data:  x and y
   W = 30, p-value = 0.2
   alternative hypothesis: true location shift is not equal to 0
```

```
wc <- function(){
  x <- rnorm(10)
  y <- rnorm(10, sd = 15)
  tt <- wilcox.test(x,y)
  tt$p.value
}

pVals_WC <- replicate(1000, expr = wc())
```

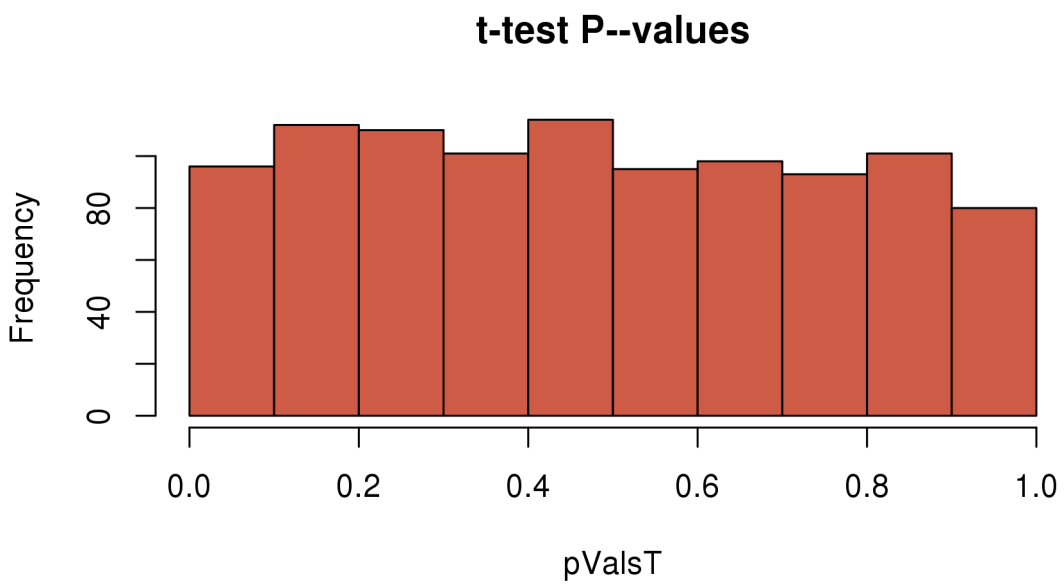
```
hist(pVals_WC, col = "tan3", main = "Wilcoxon P--values")
```




```
prop.table(table(pVals_WC < 0.05))
```

```
   FALSE  TRUE
    0.91  0.09
```

```
set.seed(999)

ttest <- function(){
  x <- rnorm(10)
  y <- rnorm(10, sd = 15)
  tt <- t.test(x,y)
  tt$p.value
}

pValsT <- replicate(1000, expr = ttest())

hist(pValsT, col = "coral3", main = "t-test P--values")
```




```
prop.table(table(pValsT < 0.05))


   FALSE  TRUE
   0.956 0.044
```

the t–test pvalues are correct (uniformly distributed) and the alpha level is kept, while the Wilcoxon test is too optimistic and has an actual level of near 10% at a nominal level of 5%. So there are too many false positives for the Wilcoxon test.

6 Chi–squared Test and the fisher test for contingency tables

The test above treat continuous data, We now turn to tests for categorical data. Typically, categorical data is represented in the form of contingency tables, where one categorization is represented by the rows and the other by the columns. A χ^2 test then tests for a for independence of rows and columns in an $r \times c$ contingency table. It will tell us, whether the row classifications are independent of the column classifications in a table like this:

$$\begin{array}{cccc|c} n_{11} & n_{12} & \cdots & n_{1c} & n_{1\cdot} \\ n_{21} & n_{22} & \cdots & n_{2c} & n_{2\cdot} \\ \vdots & \vdots & & \vdots & \vdots \\ n_{r1} & n_{r2} & \cdots & n_{rc} & n_{r\cdot} \\ \hline n_{\cdot 1} & n_{\cdot 2} & \cdots & n_{\cdot c} & n_{\cdot\cdot} \end{array}$$

The actual number observations in each cell of the table can be compared to the expected number of observations under the assumption of independent row and column classifications and is given by

$$E_{ij} = \frac{n_{i.} \times n_{.j}}{n_{..}}$$

and a χ^2 statistic can be computed as above:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

it has $(r-1)(c-1)$ degrees of freedom.

6.1 Fishers tea tasting experiment and genetics

One of the most famous examples of hypothesis testing was performed by RA Fisher on a lady that claimed could tell if milk was added before or after the tea was poured. Fisher gave the lady four pairs of cups of tea: one with milk poured first, the other after. The order was randomized. Say the lady picked 3 out 4 correctly, do we believe she has a special ability? Tests for discrete data help to answer this question by quantifying what happens by chance.

The basic question we ask is: if the lady is just guessing, what are the chances that she gets 3 or more correct? If we assume the lady is just guessing randomly, we can think of this particular examples as picking 4 balls out of an urn with 4 green (correct answer) and 4 red (incorrect answer) balls.

Under the null hypothesis that the lady is just guessing each ball has the same chance of being picked. We can then use combinatorics to figure out the probability. The probability of picking 3 is $\binom{4}{3}\binom{4}{1}/\binom{8}{4} = 16/70$. The probability of picking all correct is $\binom{4}{4}\binom{4}{0}/\binom{8}{4} = 1/70$. Thus the chance of observing a 3 or something more extreme, under the null hypothesis, is 0.24. This is the p–value. This is called Fisher's exact test and it uses the hyper geometric distribution. It is not appropriate for most the tests applied in genetics but the idea is similar.

For example, imagine we have 250 individuals, some of them have a given disease others don't. We observe that a 20% of the individuals that are homozygous for the minor allele have the disease compared to 10% of the rest. Would we see this again if we picked another 250 individuals?

Here is an example dataset

```
disease=c(rep("no",180),rep("yes",20),rep("no",40),rep("yes",10))
genotype=c(rep("AA",200),rep("aa",50))
tab=table(genotype,disease)
tab
```

```
          disease
   genotype  no yes
         aa  40  10
         AA 180  20
```

The null-hypothesis is that the 200 and 50 individuals in each group were assigned disease with the same probability. If this is the case then the probability of disease is

```
p <- mean(disease == "yes")
p
```

```
   [1] 0.12
```

The expected table is therefore

```
rbind(c(1-p,p)*sum(genotype=="aa"),c(1-p,p)*sum(genotype=="AA"))
```

```
        [,1] [,2]
   [1,]   44    6
   [2,]  176   24
```

We can compute an χ^2 statistic of seeing a deviation for the expected table as big as this one. The p-value for this table is

```
chisq.test(tab)$p.value
```

```
   [1] 0.0886
```

Note that there is not a one to one relationship between the odds ratio ($\frac{n_{11}}{n_{12}} / \frac{n_{21}}{n_{22}}$) and the p-value. If we increase the numbers but keep the difference in proportions the same, the p–value is reduced substantially:

```
tab=tab*10
chisq.test(tab)$p.value
```

```
   [1] 1.22e-09
```

6.2 Simple gene set enrichment analysis

Suppose that the number of onco–type of genes in Chromosome 1 is $n_{11} = 100$ out of a total of $n_{12} = 200$ genes and the number of onco–genes in the rest of the genome is $n_{21} = 300$ out of a total of $n_{22} = 6000$ genes as summarized in the table.

	onco–genes	non–onco-genes	row–sums
Chromosome 1	100	200	300
Rest of Genome	3000	6000	9000
column–sums	3100	6200	9300

The χ^2 test will now tell us, whether there is a significantly higher or lower proportion of onco–genes in chromosome 1 than in the rest of the genome. Chromosome 1 serves as our gene set here. In biology, over–representation is often called "enrichment" and an under–representation is called "depletion" and hence the χ^2 test for this table can be viewed as test of an onco–gene enrichment/depletion in the gene set chromosome 1:

```
dat1 <- matrix(c(100,200,3000,6000),2,byrow=TRUE)
## Chi2 test
chisq.test(dat1)
```

```

    Pearson's Chi-squared test

   data:  dat1
   X-squared = 0, df = 1, p-value = 1
```

An alternative to the χ^2 test for 2×2 tables is the Fisher–test. It tests whether the odds ratio $\frac{n_{11}}{n_{12}} / \frac{n_{21}}{n_{22}}$ is significantly different from 1, which would again indicate a "depletion" (if OR < 1) or enrichment (if OR > 1) of oncogenes in chromosome 1 in this case. As we can see the odds ratio is 1, i.e. there is neither enrichment nor depletion.

```
dat1 <- matrix(c(100,200,3000,6000),2,byrow=TRUE)

## Fisher test
fisher.test(dat1)
```

```

    Fisher's Exact Test for Count Data

   data:  dat1
   p-value = 1
   alternative hypothesis: true odds ratio is not equal to 1
   95 percent confidence interval:
```

```
    0.775 1.283
   sample estimates:
   odds ratio
            1
```

As yet another alternative, we can compare the proportion of onco–genes in chromosome 1 to the the proportion of onco–genes in the rest of the genome. As we can see, the test of proportions also returns a p–value of 1.

```
dat1 <- matrix(c(100,200,3000,6000),2,byrow=TRUE)
## Comparison of proportions of oncogenes in the two subsets
dat1.prop <- matrix(c(dat1[1,1], dat1[2,1],dat1[1,1] + dat1[1,2],
               dat1[2,1] + dat1[2,2]), 2,2,byrow=TRUE)
prop.test(dat1.prop[1,] ,dat1.prop[2,])


    2-sample test for equality of proportions without continuity
    correction

   data:  dat1.prop[1, ] out of dat1.prop[2, ]
   X-squared = 1e-28, df = 1, p-value = 1
   alternative hypothesis: two.sided
   95 percent confidence interval:
    -0.0542  0.0542
   sample estimates:
   prop 1 prop 2
    0.333  0.333
```

Let's look at some additional data: the table below shows an example of an under–representation of onco–genes in Chromosome 1

	onco–genes	non–onco-genes	row–sums
Chromosome 1	50	250	300
Rest of Genome	3000	6000	9000
column–sums	3050	6250	9300

χ^2 test

```
dat2 <- matrix(c(50,250,3000,6000),2,byrow=TRUE)
## Chi2 test
chisq.test(dat2)


    Pearson's Chi-squared test with Yates' continuity correction

   data:  dat2
   X-squared = 40, df = 1, p-value = 2e-09

## Fisher test
fisher.test(dat2)


    Fisher's Exact Test for Count Data

   data:  dat2
   p-value = 3e-10
   alternative hypothesis: true odds ratio is not equal to 1
   95 percent confidence interval:
```

```
    0.288 0.546
   sample estimates:
   odds ratio
          0.4
```

```
## Comparison of proportions of oncogenes in the two subsets
dat2.prop <- matrix(c(dat2[1,1], dat2[2,1],dat2[1,1] + dat2[1,2],
              dat2[2,1] + dat2[2,2]), 2,2,byrow=TRUE)
prop.test(dat2.prop[1,] ,dat2.prop[2,])
```

```
    2-sample test for equality of proportions with continuity correction

   data:  dat2.prop[1, ] out of dat2.prop[2, ]
   X-squared = 40, df = 1, p-value = 2e-09
   alternative hypothesis: two.sided
   95 percent confidence interval:
    -0.212 -0.122
   sample estimates:
   prop 1 prop 2
    0.167  0.333
```

Both the χ^2 test as well as the Fisher test are significant. The odds ratio is $\frac{50}{250} / \frac{3000}{6000} = 0.4$ showing a depletion (OR < 1) of oncogenes in chromosome 1. The test of proportions also gives a significant p–value.

The odds ratio is very often used as a measure of association in 2×2 tables since the absolute value of the natural logarithm of the odds ratio (often called lod–score) given by

$$\text{lod} = \left| \ln \left(\frac{n_{11}}{n_{12}} / \frac{n_{21}}{n_{22}} \right) \right|$$

is only dependent on the cell contents of table, i.e. shuffling rows or columns does not change it.

Exercise: Rocky mountain spotted fever

In 747 cases of "Rocky Mountain spotted fever" from the western United States, 210 patients died. Out of 661 cases from the eastern United States, 122 died. Is the difference statistically significant? Use a prop–test as well as a Fisher–test.

7 Multiple testing

When performing a large number of tests, the Type I error is inflated: Let's assume we perform m tests with Type I error rate α (reject H_0 although H_0 is true) of 5%. Then the probability of **no false rejection** if the tests are independent is:

$$\underbrace{0.95 \cdot 0.95 \cdot \ldots \cdot 0.95}_{\text{m–times}} \ggg 0.95 \tag{1}$$

Thus, the larger the number of tests performed, the higher the probability of a false rejection (= Type I error, false positive)

However, this problems is often put aside and hypothesis testing/significance analysis is commonly used in a too simple way. Correcting for multiple testing helps to avoid false positives or discoveries. There are two key components of a multiple testing procedure:

- Error measure
- Correction procedure / estimation algorithm

7.1 Types of errors and error rates

Suppose you are testing a hypothesis that a parameter β equals zero versus the alternative that it does not equal zero. Let us assume that there are m_0 number of tests that correspond to a true null hypothesis out of m total tests and that we reject R null hypotheses in total. These are the possible outcomes:

	$\beta = 0$	$\beta \neq 0$	Hypotheses
Claim $\beta = 0$	True Positive	False Negative	$m - R$
Claim $\beta \neq 0$	False Positive	True Negative	R
Claims	m_0	$m - m_0$	m

- Type I error or false positive — Say that the parameter does not equal zero when it does
- Type II error or false negative — Say that the parameter equals zero when it doesn't

Just like ordinary significance testing tries to control the false positive rate, there are other types of rates commonly used in multiple testing procedures:

- **False positive rate** - The rate at which false results ($\beta = 0$) are called significant: $E\left[\frac{FP}{m_0}\right]$
- **Family wise error rate (FWER)** - The probability of at least one false positive $\Pr(FP \geq 1)$
- **False discovery rate (FDR)** - The rate at which claims of significance are false $E\left[\frac{FP}{FP+TP}\right]$

If p–values are correctly calculated calling all $p < \alpha$ significant will control the false positive rate at level α on average.

7.2 Control of error rates

Suppose that you perform 10,000 tests and $\beta = 0$ for all of them. and you call all $P < 0.05$ significant. Then expected number of false positives is: $10,000 \times 0.05 = 500$ false positives. How do we avoid so many false positives?

7.2.1 Controlling family-wise error rate (FWER)

The Bonferroni correction is the oldest multiple testing correction.

Basic algorithm

- Suppose you do m tests
- You want to control FWER at level α so $Pr(FP \geq 1) < \alpha$
- Calculate p–values normally
- Set $\alpha_{fwer} = \alpha/m$
- Call all p-values less than α_{fwer} significant

The bonferroni correction is easy to calculate but very conservative.

7.2.2 Controlling false discovery rate (FDR)

This is the most popular correction when performing lots of tests as in in genomics. It is often termed the Benjamini Hochberg procedure and controls the FDR.

Basic algorithm

- Suppose you do m tests

- You want to control FDR at level α so $E\left[\frac{FP}{TP+FP}\right] < \alpha$
- Calculate p–values normally
- Order the p–values from smallest to largest $p_{(1)}, ..., p_{(m)}$
- Call any $p_{(i)} \leq \alpha \times \frac{i}{m}$ significant

The FDR control procedure is still pretty easy to calculate and less conservative (possibly much less) than controlling the FWER. On the contrary, it allows for more false positives and may behave strangely under dependence.

7.2.3 Adjusted p–values

The approach indicated above is to adjust the threshold α, a different approach is to calculate "adjusted p–values". They are not p–values anymore but they can be used directly without adjusting α.

Example

- Suppose p-values are $p_1, \ldots, p_m$
- You could adjust them by taking $p_i^{fwer} = \max\{m \times p_i, 1\}$ for each p–value.
- Then if you call all $p_i^{fwer} < \alpha$ significant you will control the FWER.

7.3 Diagnostic plots for multiple testing procedures

The code below simulates $m = 200$ p–values from the mixture model

$$0.75 \cdot N(0,1) + 0.25 \cdot N(2,1)$$

,i.e. $m_0 = 150$ here and the null distribution is the standard normal distribution. **It is an important fact hat for a continuous null distributions the corresponding p–values are uniform**.

```
sd.true =1
eta0.true = 0.75

get.random.zscore = function(m=200)
{

  m0 = m*eta0.true
  m1 = m-m0
  z = c(  rnorm(m0, mean=0, sd=sd.true),
       rnorm(m1, mean=2, sd=1))
  #z = sign(rnorm(length(z)))*z

  return(z)
}

set.seed(555)
z <- get.random.zscore(200)
pv = 1- pnorm(z, sd = sd.true)
```

7.3.1 Schweder and Spjøtvoll plot

If a test statistic does not correspond to a true null hypothesis, the corresponding p–value will be very small. For large p–values which likely will correspond to true null hypotheses it then holds that

$$E[M(p)] = m_0(1-p)\,.$$

Large (probably non–null) p–values will thus be close to a straight line with slope m_0. Accordingly, small p–values (probably null) will deviate from that line.

Schweder and Spjøtvoll (Biometrika, 1982) suggested to use these facts for a diagnostic plot of the observed p-values which permits estimation of the fraction of true null hypotheses. For a series of hypothesis tests $H_1, \ldots, H_m$ with p-values p_i, they suggested plotting

$$(1-p_i, M(p_i)) \text{ for } i \in 1, \ldots, m,$$

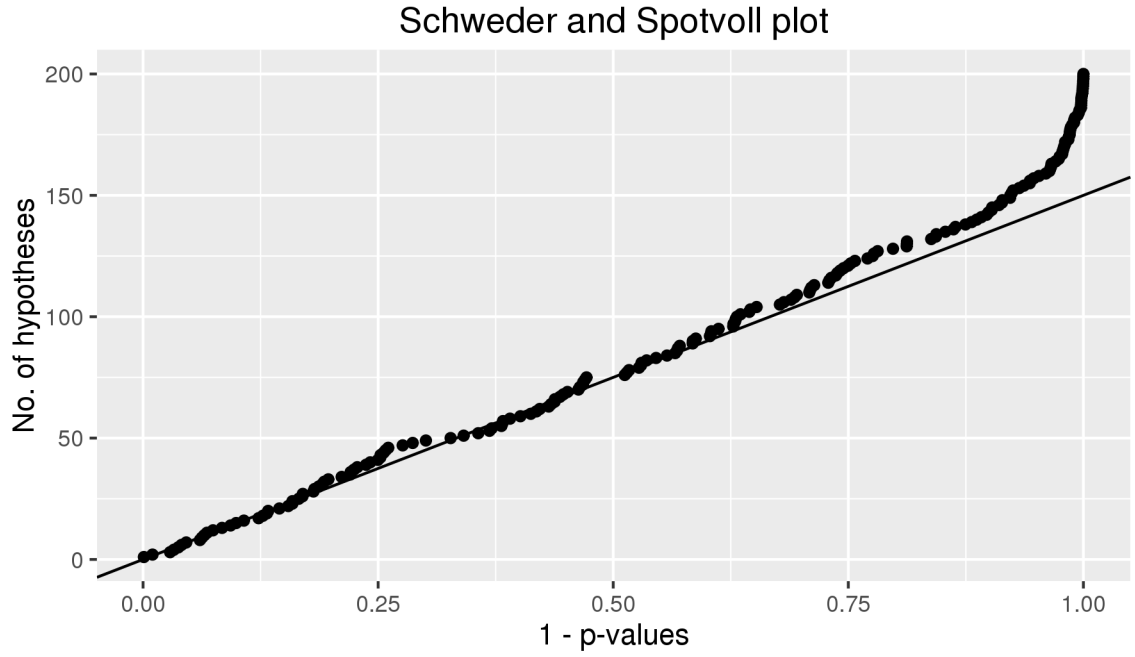
where $M(p)$ is the number of p–values greater than p. An application of this diagnostic plot to our simulated p–values can be seen in the figure.

When the first m_0 null hypotheses are true and the other $m - m_0$ are false, the cumulative distribution function of $(1-p_1, \ldots, 1-p_{m_0})$ is expected to be close to the line $F_0(t) = t$. The cumulative distribution function of $(1-p_{m_0+1}, \ldots, 1-p_m)$, on the other hand, is expected to be close to a function $F_1(t)$ which stays below F_0 but shows a steep increase towards 1 as t approaches 1. In practice, we do not know which of the null hypotheses are true, so we can only observe a mixture whose cumulative distribution function is expected to be close to

$$F(t) = \frac{m_0}{m}F_0(t) + \frac{m-m_0}{m}F_1(t).$$

In our simulated data $F_0 = 1 - N(0,1)$ and $F_1 = 1 - N(2,1)$. By looking at the figure, the points start to deviate at 150 from the straight line, as expected from the simulation model.

```
(ggplot2::qplot(sort(1-pv), 1:200, xlab = "1 - p-values", ylab = "No. of hypotheses",
       main = "Schweder and Spotvoll plot")
 + geom_abline(intercept = 0, slope = 200*eta0.true, aes(color = "coral3")))
```

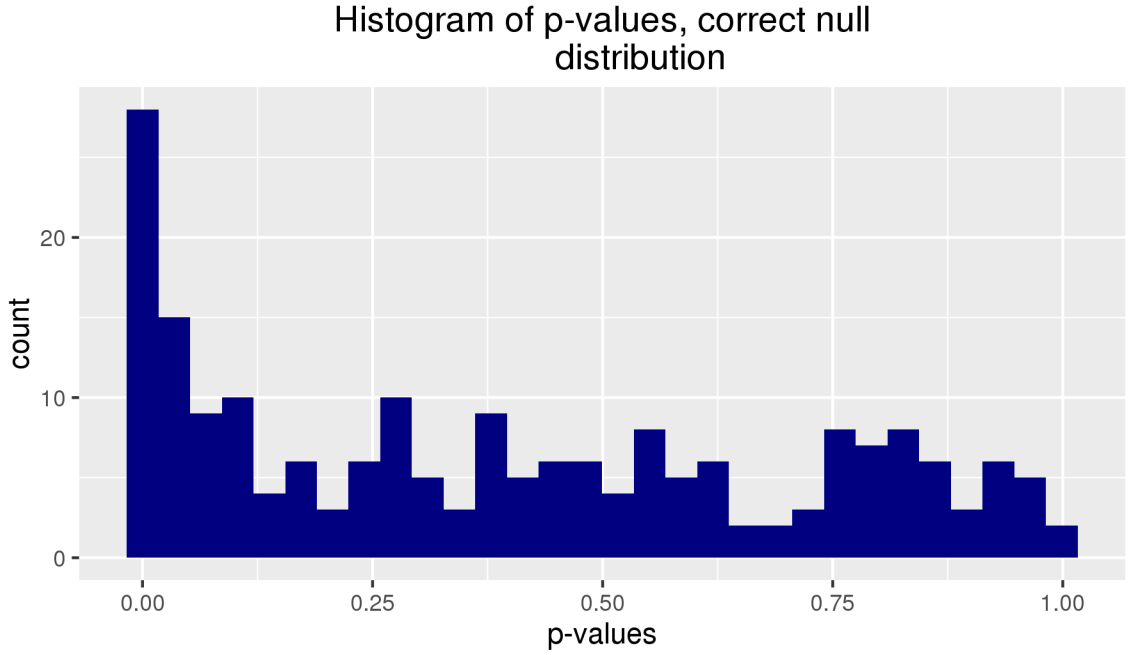


7.3.2 Histogram of p–values

As already mentioned, the p–values follow a uniform distribution on the unit interval [0,1] if they are computed using a continuous null distribution. Significant p–values thus become visible as an enrichment of p–values near zero in the histogram. A histogram of p–values should always be plotted in order to check whether they have been computed correctly.

```
ggplot2::qplot(x = pv, xlab = "p-values", main = "Histogram of p-values, correct null
        distribution",
       fill = I("navyblue"))
```

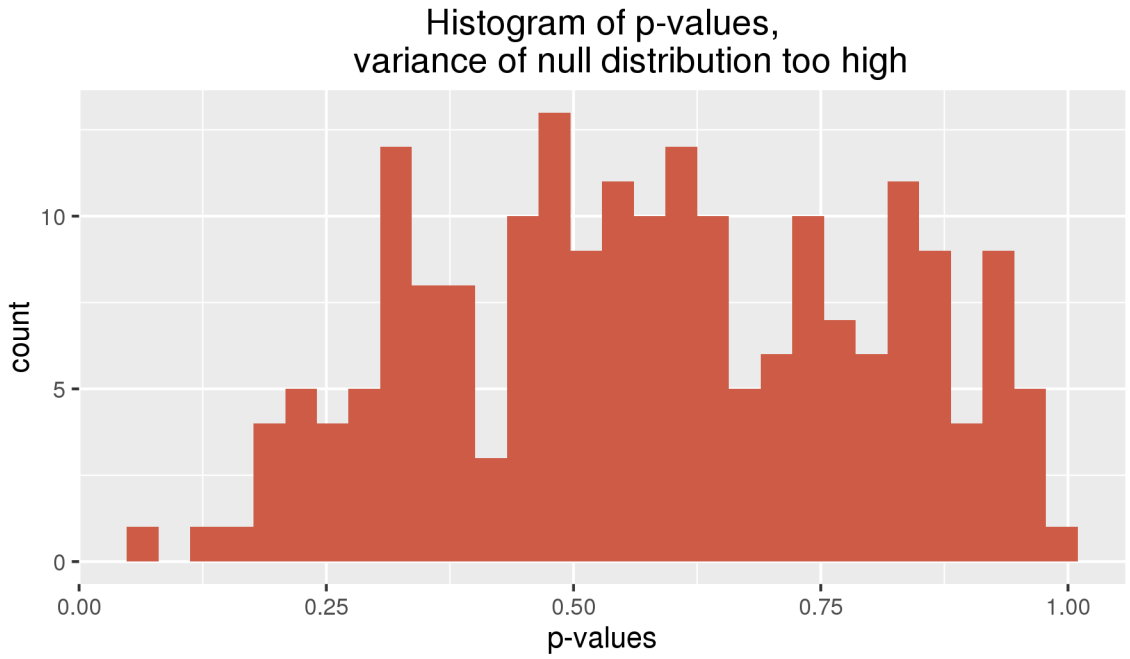
```
  `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



We see that our p–values are uniformly distributed under the null hypotheses. Computing the p–values assuming a $N(0, 2)$ null distribution changes the picture.

```
ggplot2::qplot(x = pnorm(z, sd = 2) , xlab = "p-values", main = "Histogram of p-values,
      variance of null distribution too high",
       fill = I("coral3"))
```

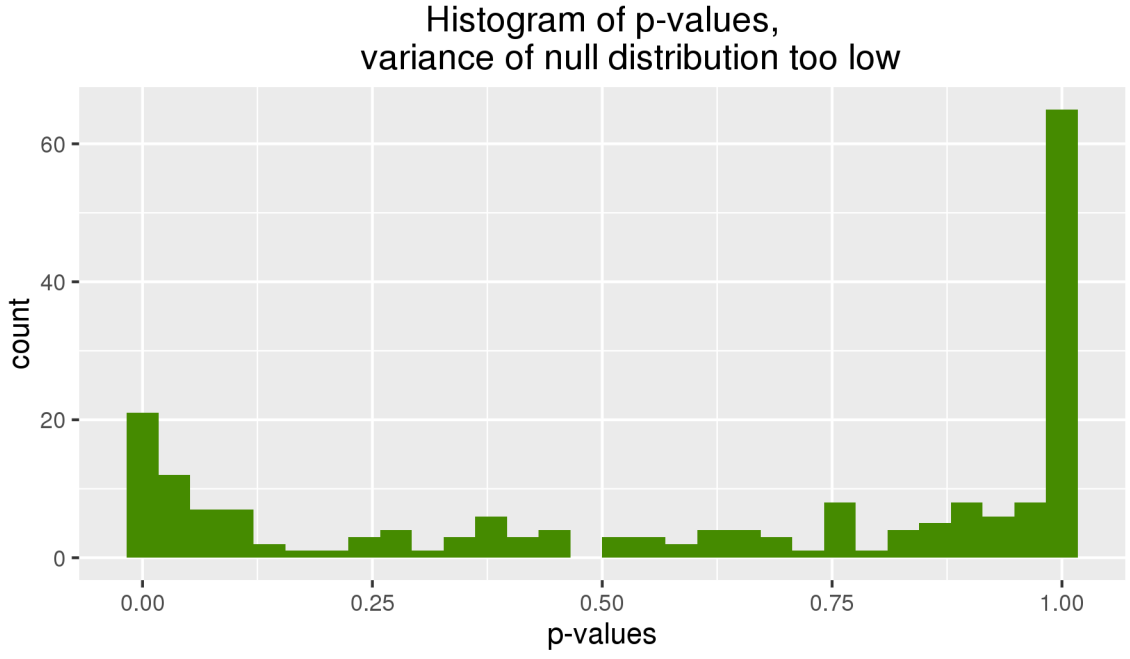
```
  `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



If the assumed variance of the null distribution is too high, we often see hill–shaped p–value histogram. If the variance is too low, we get a U–shaped histogram, with peaks at both ends.

```
ggplot2::qplot(x = pnorm(z, sd = 0.5) , xlab = "p-values", main = "Histogram of p-values,
      variance of null distribution too low",
       fill = I("chartreuse4"))
```

```
`stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```



7.4 Computing multiple testing adjustments

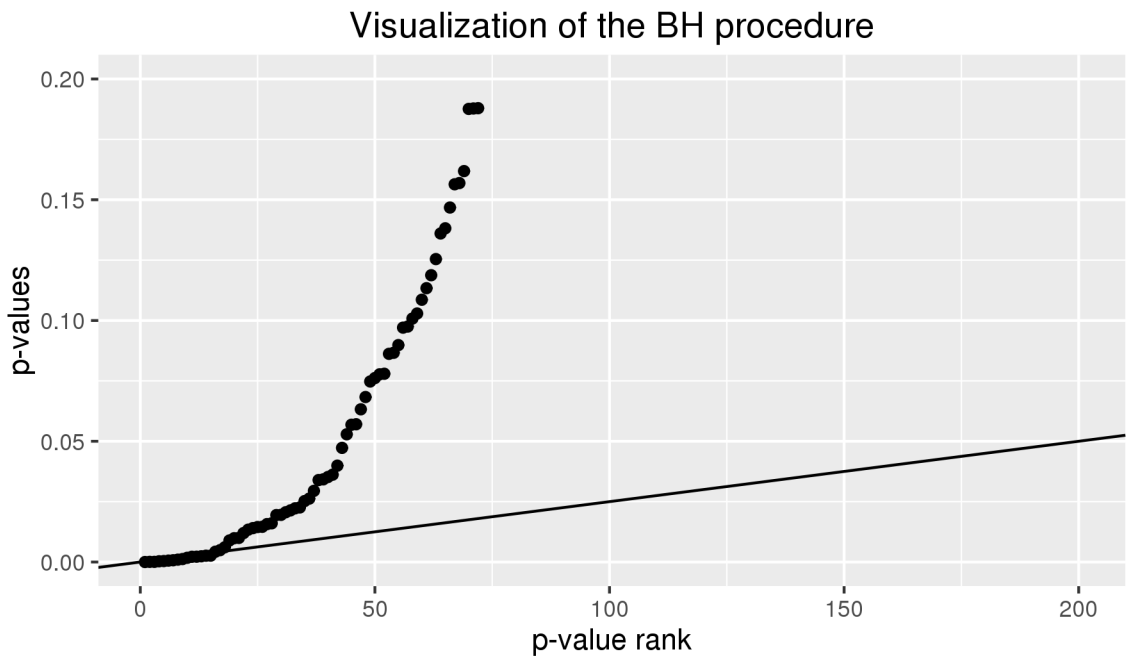
The most commonly used multiple testing adjustments can be computed using the function `padjust`. To compute the Benjamini Hochberg adjusted p–values simply specify `method = "BH"`. For FEWR control we choose `method = "bonferroni"`.

```
alpha = 0.05
pv.BH <- p.adjust(pv, method = "BH")
table(pv.BH < 0.05)
```

```
FALSE  TRUE
  185    15
```

```
(ggplot2::qplot(rank(pv), pv, xlab = "p-value rank", ylab="p-values"
       ,main = "Visualization of the BH procedure")
+ geom_abline(intercept = 0, slope = alpha/200, aes(color = "coral3"))
+ ylim(c(0, 0.2) ))
```

```
  Warning:  Removed 128 rows containing missing values (geom_point).
```




```
pv.FWER <- p.adjust(pv, method = "bonferroni")
table(pv.FWER < 0.05)
```

```
FALSE  TRUE
  197     3
```

The figure illustrates of the Benjamini-Hochberg multiple testing adjustment. The black line shows the p-values (y-axis) versus their rank (x-axis), starting with the smallest p-value from the left, then the second smallest, and so on. The red line is a straight line with slope α/m, where m is the number of tests, and α is a target false discovery rate. FDR is controlled at the value α if the genes are selected that lie to the left of the rightmost intersection between the red and black lines: here, this results in 15 significant p–values. Thus, the procedure is relatively conservative since we actually have simulated 25 non–null p–values. The Bonferroni correction is clearly not suitable here as we only get 3 significant p–values.

7.5 Modifying the BH procedure to gain power and the q–value

The *mutoss* package provides many more multiple testing adjustments. In addition, it also has the function `ABH_ pi0_ est` that estimates the proportion π_0 of the null model (in our case 75%) for us based on the Schweder and Spjøtvoll plot. Here, it estimates π_0 as 0.83 which is quite far away from the true value. However, with only 200 test statistics, it also difficult to estimate π_0 reliably.

```
 ABH_pi0_est(pv)

   $pi0
   [1] 0.83
```

```
pv.OracleBH <- oracleBH(pValue=pv, alpha=alpha, pi0=0.75)



    Benjamini-Hochberg's (1995) oracle linear-step-up Procedure

   Number of hyp.:  200
   Number of rej.:  17
      rejected  pValues adjPValues
   1       153 2.44e-06   0.000367
   2       170 6.71e-05   0.003512
   3       154 7.02e-05   0.003512
   4       105 3.11e-04   0.011677
   5       185 4.00e-04   0.012013
   6       168 5.84e-04   0.014611
   7       200 7.28e-04   0.015605
   8       182 9.89e-04   0.018535
   9       199 1.23e-03   0.020530
   10      189 1.75e-03   0.026286
   11      194 2.17e-03   0.026464
   12      160 2.21e-03   0.026464
   13      156 2.38e-03   0.026464
   14      166 2.62e-03   0.026464
   15      190 2.65e-03   0.026464
   16      177 4.27e-03   0.040020
   17      164 4.92e-03   0.043379
```

The BH procedure implicitly assumes $\pi_0 = 1$, i.e. that there are no non–null p-values in its original form. Thus, we can gain power, by plugging in an estimate of π_0. Indeed, we gain 2 more rejections.

The q–value is an FDR estimation procedure roughly defined as a BH procedure combined with a pi_0 estimate — pretty much like the oracle BH procedure above and is very popular in genomics. It tries to estimate π_0 from the p–value histogram. Note that it actually tries to estimate the FDR for a test statistic rather than just providing a control of the FDR as the BH procedure.

```
pv.Qvals <- Qvalue(pv)



    Storey's (2001) q-value Procedure

   Number of hyp.:  200
   Estimate of the prop. of null hypotheses:  0.759
```

```
table(pv.Qvals$qValues < 0.05)
```

```
   FALSE  TRUE
     183    17
```

Although the number of p–values is low, the q–value procedure estimates pi_0 correctly and provides the same number of rejected hypotheses as oracle BH procedure.

8 Regularized t–tests for small n, large p problems

In microarray analyses, on usually uses a variant of a (regularized) t–statistic that is suitable for high-dimensional data and large-scale multiple testing such as the one implemented in the Bioconductor package *limma*.

The basic statistic used for significance analysis in *limma* is the moderated t–statistic, which is computed for each gene separately. It has the same interpretation as an ordinary t–statistic except that the standard errors have been moderated across genes, i.e., shrunk toward a common value, using a simple Bayesian model. This has the effect of borrowing information from the ensemble of genes to aid with inference about each individual gene. Moderated t-statistics lead to p- values in the same way that ordinary t-statistics do except that the degrees of freedom are increased, reflecting the greater reliability associated with the smoothed standard errors.

8.1 Some details of the *limma* method

The empirical Bayes method in *limma* assumes an inverse Chi-square prior for the variance σ^2 with mean s_0^2 and degrees of freedom f_0. These parameters are estimated from data and not set beforehand, hence this is an "empirical" Bayesian method.

The posterior values for the residual variances are given by

$$s_j^2 = \frac{f_0 s_0^2 + f\sigma^2}{f_0 + f}$$

Where f denotes the degrees of freedom for a gene. For two group comparison $f = n_1 + n_2 - 2$, where n_1 and n_2 are the number of samples in each of the groups.

8.2 Shrinkage estimation

The most important aspect of the *limma* approach is that *limma* performs a SHRINKAGE of the variances towards a target, which is given by s_0^2, the prior mean variance and the shrinkage intensity is $\frac{f_0}{f_0+f}$.

A t–test with $f_0 + f$ degrees of freedom using the shrunken variances is then computed to assess differential expression. There are many other ways to perform shrinkage, which will commonly improve the estimation of the variance by sharing information across genes. A wide selection of these statistics is implemented in the package *st* We use the `modt.stat` from the package to compute *limma*s moderated t–statistic for BCL2 in the ALL data:

```
modt.stat(t(expALL), groupsALL)[1152]
```

```
   2039_s_at
        4.72
```

```
### p-value using the normal distribution
2 - 2*pnorm(modt.stat(t(expALL), groupsALL)[1152])
```

```
   2039_s_at
    2.34e-06
```

```
### slightly higher t-value than the ordinary t-test
t.test(expALL[1152,] ~ groupsALL, var.equal=FALSE)$statistic
```

```
      t
   4.76
```

Since information about the overall variance of the genes in the data set is used to compute the variance for CCND3, the whole data set has to be provided in order to compute the moderated t–statistic.

Exercise: Group comparison for gene GYPC

The gene GYPC plays an important role in regulating the mechanical stability of red cells. I can be found in line 8197 of the expALL data set. (Try `grep("GYPC", anno_fusALL$SYMBOL))`.

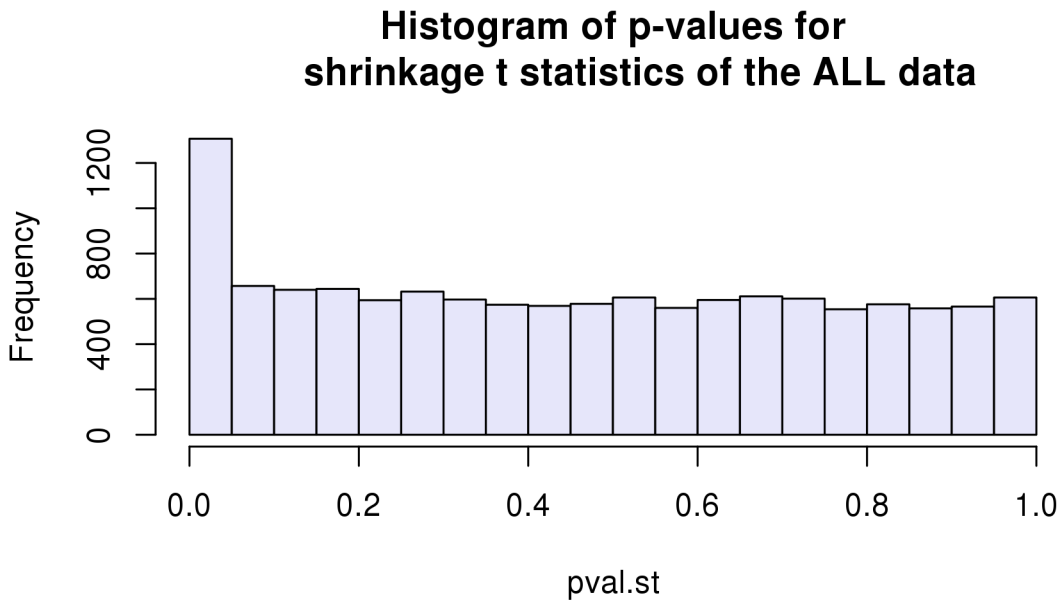
Test for the equality of the means by an appropriate t–test. Is the experimental effect very strong? Also, try testing the hypothesis using a moderated t–test and a wilcoxon test.

8.3 Multiple testing applied to the ALL data set

We illustrate some multiple testing approaches with the expALL data set. For this we use the shrinkage t statistic from the *st* package which implements an analytical rather than a Bayesian shrinkage approach. First, we compute the t–statistics and the associated tow–sided p–values using a standard normal distribution as the null model. The resulting p–value histogram looks good.

```
sts <- shrinkt.stat(t(expALL), groupsALL)

   Number of variables: 12625
   Number of observations: 79
   Number of classes: 2

   Estimating optimal shrinkage intensity lambda.freq (frequencies): 1
   Estimating variances (pooled across classes)
   Estimating optimal shrinkage intensity lambda.var (variance vector): 0.0382
```

```
### p-value using the normal distribution
pval.st <- 2 - 2*pnorm(abs(sts))
hist(pval.st, col = "lavender", main = "Histogram of p-values for
     shrinkage t statistics of the ALL data")
```



We can compute a standard BH adjustment to obtain the number of differentially expressed genes at an FDR of 5%.

```
pv.st.BH <- p.adjust(pval.st, method = "BH")
table(pv.st.BH< 0.05)


   FALSE  TRUE
   12398   227
```

Exercise: Multiple testing for the ALL data

Try other multiple testing procedures like q–values on the p–values obtained from the shrinkage t statistics. Can you gain power? Produce a p–value histogram where significant statistics are indicated by color-fill.

9 Indepdendent filtering of hypotheses

We have seen in the previous sections that multiple testing approaches, with thousands of tests, are often used in analyses of genome-scale data. For instance, we have seen that in the analyses of differential gene expression based on RNA-Seq or microarray data, a common approach is to apply a statistical test, one by one, to each of thousands of genes, with the aim of identifying those genes that have evidence for a statistical association of their expression measurements with the experimental covariate(s) of interest.

Another instance is differential binding detection from ChIP-Seq data. 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. Typically, this results in increased detection power at the same experiment-wide type I error, as measured in terms of the false discovery rate. A good choice for a filtering criterion is one that

1. is statistically independent from the test statistic under the null hypothesis,
2. is correlated with the test statistic under the alternative, and
3. does not notably change the dependence structure –if there is any– of the joint test statistics (including those corresponding to true nulls and to true alternatives).

The benefit from filtering relies on property 2. The statistical validity of filtering relies on properties 1 and 3. For many practically useful combinations of filter criteria with test statistics, property 1 is easy to prove. Property 3 is more complicated, but rarely presents a problem in practice: if, for the multiple testing procedure that is being used, the correlation structure of the tests was acceptable without filtering, the filtering should not change that.

9.1 Example data set

For illustration, let us use the `pasillaGenes` dataset from the Bioconductor package *pasilla*; this is an RNA-Seq dataset from which we extract gene-level read counts for two replicate samples the were measured for each of two biological conditions: normally growing cells and cells treated with dsRNA against the *Pasilla* mRNA, which led to RNAi interference (RNAi) mediated knockdown of the Pasilla gene product.

```
data("pasillaGenes")
```

We perform a standard analysis with *DESeq* to look for genes that are differentially expressed between the normal and Pasilla-knockdown conditions, indicated by the factor variable `condition`. In the generalized linear model (GLM) analysis, we adjust for an additional experimental covariate `type`, which is however not of interest for the differential expression. For more details, please see the vignette of the *DESeq* package.

```
cds  <- estimateSizeFactors( pasillaGenes )
cds  <- estimateDispersions( cds )
fit1 <- fitNbinomGLMs( cds, count ~ type + condition )
fit0 <- fitNbinomGLMs( cds, count ~ type  )
```

```
res <- data.frame(
filterstat <- rowMeans(counts(cds)),
pvalue    <- nbinomGLMTest( fit1, fit0 ),
row.names <-featureNames(cds) )
```

The details of the anove analysis are not important for explaing the benefits of filtering, the essential output is contained in the columns of the dataframe `res`:

- `filterstat`: the filter statistic, here the average number of counts per gene across all samples, irrespective of sample annoation,
- `pvalue`: the test p-values,

Each row of the dataframe corresponds to one gene:

```
dim(res)
```

```
[1] 14470     3
```

```
head(res)
```

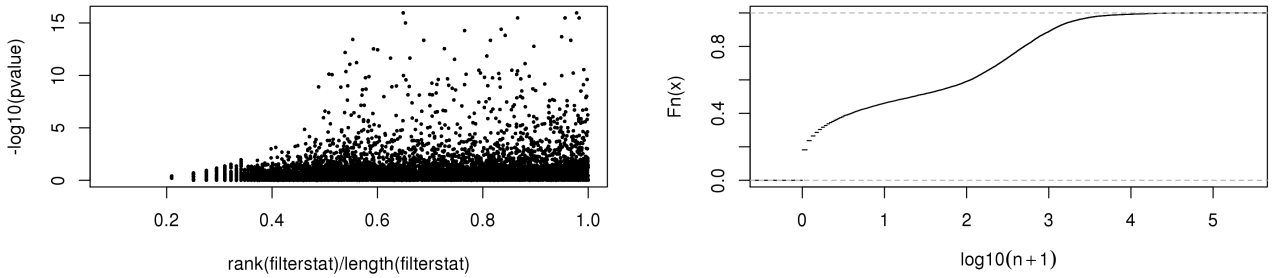
```
              filterstat....rowMeans.counts.cds..
FBgn0000003                                 0.143
FBgn0000008                                54.714
FBgn0000014                                 0.429
FBgn0000015                                 0.857
FBgn0000017                              2507.429
FBgn0000018                               239.286
              pvalue....nbinomGLMTest.fit1..fit0. row.names....featureNames.cds.
FBgn0000003                                0.4601                     FBgn0000003
FBgn0000008                                0.9747                     FBgn0000008
FBgn0000014                                0.8395                     FBgn0000014
FBgn0000015                                0.7274                     FBgn0000015
FBgn0000017                                0.0963                     FBgn0000017
FBgn0000018                                0.6794                     FBgn0000018
```

9.2 Qualitative assessment of the filter statistic

First, consider the Figure below, which shows that among the approximately 40% of genes with lowest overall counts, `filterstat`, there are essentially none that achieved an (unadjusted) p-value less than 0.003 (this corresponds to about 2.5 on the $-\log_{10}$-scale).

```
with(res,
  plot(rank(filterstat)/length(filterstat), -log10(pvalue), pch=16, cex=0.45))
```

```
trsf = function(n) log10(n+1)
plot(ecdf(trsf(res$filterstat)), xlab=body(trsf), main="")
```



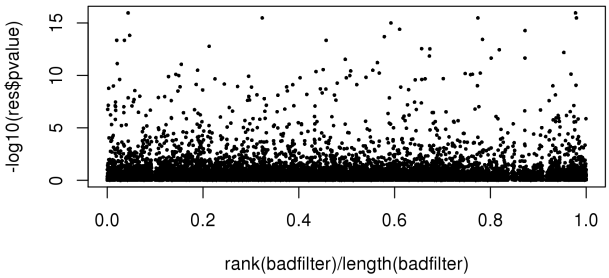
Left: scatterplot of the rank (scaled to $[0,1]$) of the filter criterion `filterstat` (x-axis) versus the negative logarithm of the test `pvalue` (y-axis). Right: the empirical cumulative distribution function (ECDF) shows the relationships between the values of `filterstat` and its quantiles.

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

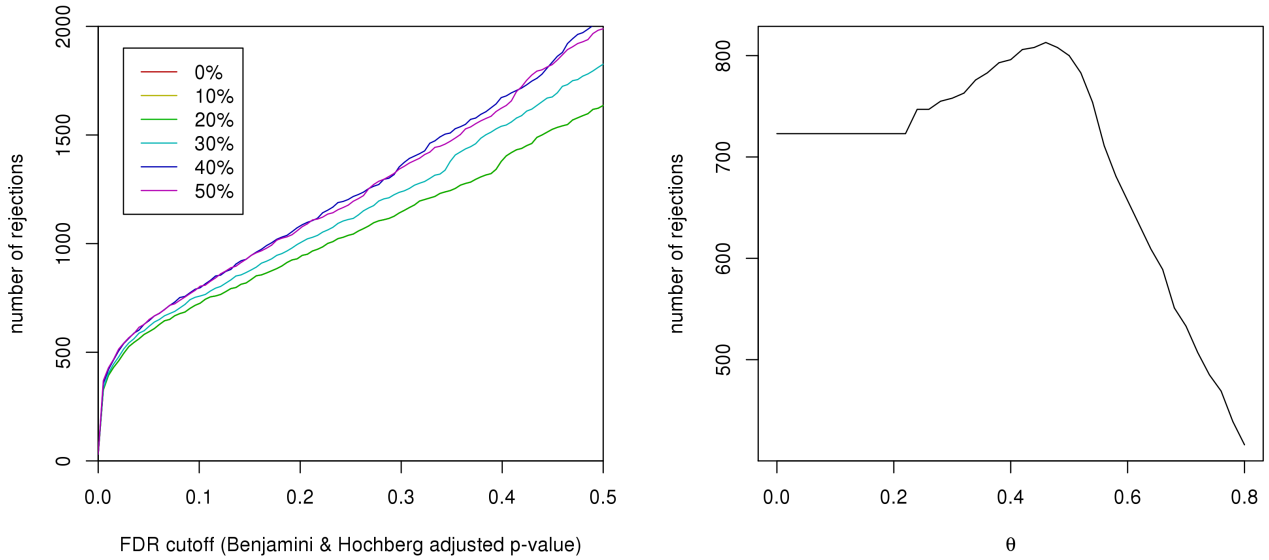
For comparison, suppose you had chosen a less useful filter statistic, say, the gene identifiers interpreted as a decimal number. The analogous scatterplot to that of Figure 1 is shown in Figure 2.

```
badfilter = as.numeric(gsub("[+]*FBgn", "", rownames(res)))
```

```
plot(rank(badfilter)/length(badfilter), -log10(res$pvalue), pch=16, cex=0.45)
```



Scatterplot analogous to Figure 1, but with `badfilter`.



Left panel: the plot shows the number of rejections (i. e. genes detected as differentially expressed) as a function of the FDR threshold (x-axis) and the filtering cutoff θ (line colours, specified as quantiles of the distribution of the filter statistic). The plot is produced by the `rejection_plot` function. Note that the lines for $\theta = 0\%$ and 10% are overplotted by the line for $\theta = 20\%$, since for the data shown here, these quantiles correspond all to the same set of filtered genes (cf. Figure 1). Right panel: the number of rejections at FDR=10% as a function of θ.

9.3 How to choose the filter statistic and the cutoff?

The `filtered_p` function in the *genefilter* package calculates adjusted p-values over a range of possible filtering thresholds. Here, we call this function on our results from above and compute adjusted p-values using the method of Benjamini and Hochberg (BH) for a range of different filter cutoffs.

```
theta = seq(from=0, to=0.5, by=0.1)
pBH = filtered_p(filter=res$filterstat, test=res$pvalue, theta=theta, method="BH")
```

```
head(pBH)
```

```
          0%   10%   20%   30%   40%   50%
[1,] 0.895 0.895 0.895    NA    NA    NA
[2,] 0.997 0.997 0.997 0.998 0.995 0.993
[3,] 0.981 0.981 0.981    NA    NA    NA
[4,] 0.960 0.960 0.960 0.970    NA    NA
[5,] 0.593 0.593 0.593 0.517 0.452 0.412
[6,] 0.951 0.951 0.951 0.964 0.938 0.924
```

The rows of this matrix correspond to the genes (i. e., the rows of `res`) and the columns to the BH-adjusted p-values for the different possible choices of cutoff `theta`. A value of `NA` indicates that the gene was filtered out at the corresponding filter cutoff. The `rejection_plot` function takes such a matrix and shows how rejection count (R) relates to the choice of cutoff for the p-values. For these data, over a reasonable range of FDR cutoffs, increased filtering corresponds to increased rejections.

```
rejection_plot(pBH, at="sample",
               xlim=c(0, 0.5), ylim=c(0, 2000),
               xlab="FDR cutoff (Benjamini & Hochberg adjusted p-value)", main="")
```

The plot is shown in the left panel of the Figure.

9.3.1 Choice of filtering cutoff

If we select a fixed cutoff for the adjusted p-values, we can also look more closely at the relationship between the fraction of null hypotheses filtered and the total number of discoveries. The `filtered_R` function wraps `filtered_p` and just returns rejection counts. It requires you to choose a particular p-value cutoff, specified through the argument `alpha`.

```
theta = seq(from=0, to=0.8, by=0.02)
rejBH = filtered_R(alpha=0.1, filter=res$filterstat, test=res$pvalue, theta=theta, method="BH")
```

Because overfiltering (or use of a filter which is inappropriate for the application domain) discards both false and true null hypotheses, very large values of θ reduce power in this example:

```
plot(theta, rejBH, type="l",
     xlab=expression(theta), ylab="number of rejections")
```

The plot is shown in the right panel of Figure 3.

9.3.2 Choice of filtering statistic

We can use the analysis of the previous section 9.3.1 also to inform ourselves about different possible choices of filter statistic. We construct a dataframe with a number of different choices.

```
filterChoices = data.frame(
  `mean`   = res$filterstat,
  `geneID` = badfilter,
  `min`    = rowMin(counts(cds)),
  `max`    = rowMax(counts(cds)),
  `sd`     = rowSds(counts(cds))
)
rejChoices = sapply(filterChoices, function(f)
  filtered_R(alpha=0.1, filter=f, test=res$pvalue, theta=theta, method="BH"))
```

```
myColours = brewer.pal(ncol(filterChoices), "Set1")
```

```
matplot(theta, rejChoices, type="l", lty=1, col=myColours, lwd=2,
        xlab=expression(theta), ylab="number of rejections")
legend("bottomleft", legend=colnames(filterChoices), fill=myColours)
```

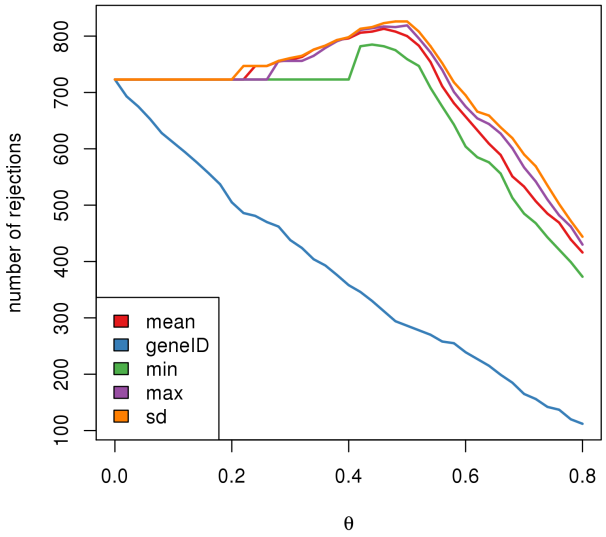
The result indicates that for the data at hand, `mean`, `max` and `sd` provide similar performance, whereas the other choices are less effective.

10 Weighting as an extension of filtering — IHW

In previous section, we introduced gene–wise filtering, which in or excludes genes based on a filter statistic. This can be seen as p–value weighting wiht 0/1 weights, wher all p-values either get a weight of 0 or 1.

Similarly, we can try to choose continous weights for the p–values in order to increase the detection power. IHW (Independent Hypothesis Weighting) tries to achieve exactly this: In addition to the p–values it allows you to specify a covariate for each test. The covariate should be informative of the power or prior probability of each individual test, but is chosen such that the p-values for those hypotheses that are truly null do not depend on the covariate, see Data-driven hypothesis weighting increases detection power in big data analytics.

The the input of IHW is the following:



The number of rejections at FDR=10% as a function of θ (analogous to the right panel in Figure 3) for a number of different choices of the filter statistic.

- a vector of p-values (of length m)
- a matching vector of covariates
- the significance level $\alpha \in (0,1)$ at which the False Discovery Rate should be controlled.

IHW then calculates weights for each p-value (non-negative numbers $w_i \geq 0$ such that $\sum_{i=1}^{m} w_i = m$). IHW also returns a vector of adjusted p-values by applying the procedure of Benjamini Hochberg to the weighted p-values $P_i^{\text{weighted}} = \frac{P_i}{w_i}$.

The weights allow different prioritization of the individual hypotheses, based on their covariate. A hypothesis with weight > 1 gets prioritized in the testing procedure, and the higher the weight the higher the prioritization. On the other hand, a hypothesis with weight equal to 0 cannot be rejected and essentially is filtered out of the procedure.

We will show now how to use the IHW package in analysing for RNA-Seq differential gene expression and then also mention some other examples where the method is applicable. We use *airway* RNA-Seq dataset and *DESeq2* .

10.1 IHW and the airway RNA–Seq data

```
dds <- DESeqDataSet(se = airway, design = ~ cell + dex)
dds <- DESeq(dds)
de_res <- as.data.frame(results(dds))
```

The output is a 'data.frame' object, which includes the following columns for each gene:

```
colnames(de_res)
```

```
   [1] "baseMean"       "log2FoldChange" "lfcSE"          "stat"
   [5] "pvalue"         "padj"
```

In particular, we have p-values and baseMean (i.e., the mean of normalized counts) for each gene. As argued in the DESeq2 paper, these two statistics are approximately independent under the null hypothesis. Thus we have all the ingredient necessary for a IHW analysis (p-values and covariates), which we will apply at a significance level 0.1.

First load IHW:

```
ihw_res <- ihw(pvalue ~ baseMean,  data=de_res, alpha = 0.1)
```

This returns an object of the class 'ihwResult'. We can get e.g. the total number of rejections.

```
rejections(ihw_res)
```

```
   [1] 4959
```

And we can also extract the adjusted p-values:

```
head(adj_pvalues(ihw_res))
```

```
   [1] 0.000986       NA 0.157601 0.839755 0.931588 1.000000
```

```
sum(adj_pvalues(ihw_res) <= 0.1, na.rm = TRUE) == rejections(ihw_res)
```

```
   [1] TRUE
```

We can compare this to the result of applying the method of Benjamini and Hochberg to the p-values only:

```
padj_bh <- p.adjust(de_res$pvalue, method = "BH")
sum(padj_bh <= 0.1, na.rm = TRUE)
```

```
   [1] 4137
```

We thus get a lot more rejections! How did we get this power? Essentially it was possible by assigning appropriate weights to each hypothesis. We can retrieve the weights as follows:

```
head(weights(ihw_res))
```

```
   [1] 2.39   NA 2.39 2.48 1.87 0.00
```

10.2 Computation of the weights

Internally, what happened was the following: We split the hypotheses into n different strata (here $n = 22$) based on increasing value of baseMean and we also randomly split them into k folds (here $k = 5$). Then, for each combination of fold and stratum, we learned the weights. The discretization into strata facilitates the estimation of the distribution function conditionally on the covariate and the optimization of the weights. The division into random folds helps us to avoid "overfitting" the data, something which can result in loss of control of the False Discovery Rate.

In particular, each hypothesis test gets assigned a weight depending on the combination of its assigned fold and stratum.

We can also see this internal representation of the weights as a (n X k) matrix:

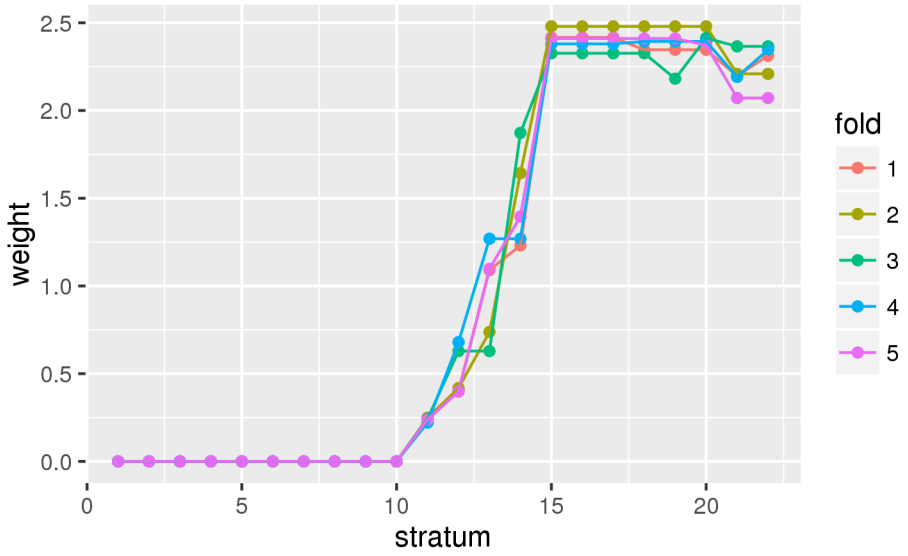
```
weights(ihw_res, levels_only=TRUE)
```

```
          [,1]  [,2]  [,3]  [,4]  [,5]
    [1,] 0.000 0.000 0.000 0.000 0.000
    [2,] 0.000 0.000 0.000 0.000 0.000
    [3,] 0.000 0.000 0.000 0.000 0.000
    [4,] 0.000 0.000 0.000 0.000 0.000
    [5,] 0.000 0.000 0.000 0.000 0.000
    [6,] 0.000 0.000 0.000 0.000 0.000
    [7,] 0.000 0.000 0.000 0.000 0.000
    [8,] 0.000 0.000 0.000 0.000 0.000
    [9,] 0.000 0.000 0.000 0.000 0.000
   [10,] 0.000 0.000 0.000 0.000 0.000
   [11,] 0.238 0.248 0.243 0.221 0.239
   [12,] 0.399 0.417 0.629 0.680 0.399
   [13,] 1.091 0.738 0.629 1.269 1.097
   [14,] 1.231 1.643 1.872 1.269 1.396
```

```
   [15,] 2.416 2.480 2.326 2.380 2.411
   [16,] 2.416 2.480 2.326 2.380 2.411
   [17,] 2.416 2.480 2.326 2.380 2.411
   [18,] 2.346 2.480 2.326 2.393 2.411
   [19,] 2.346 2.480 2.181 2.393 2.411
   [20,] 2.346 2.480 2.413 2.393 2.372
   [21,] 2.198 2.209 2.366 2.192 2.071
   [22,] 2.312 2.209 2.366 2.347 2.071
```

Finally, IHW contains a convenience function to visualize the estimated weights:

```
plot(ihw_res)
```



For each hypothesis, one can visually determine its weight by identifying its stratum and the fold it was assigned to. In the example above, we see that the general trend is driven by the covariate (stratum) and not as much by the fold. Recall that IHW assumes that the "optimal" weights should be a function of the covariate (and hence the stratum) only. Therefore, the weight functions calculated on random (overlapping) splits of the data should behave similarly, while there should be no trend driven by the folds. Also as expected, genes with very low baseMean count get assigned a weight of 0, while genes with high baseMean count get prioritized.

As a further convenience for further work, a ihwResult object can be converted to a data.frame as follows:

```
ihw_res_df <- as.data.frame(ihw_res)
colnames(ihw_res_df)
```

```
   [1] "pvalue"          "adj_pvalue"      "weight"          "weighted_pvalue"
   [5] "group"           "covariate"       "fold"
```

11 Answers to exercises

Exercise: Normal Model for a gene

Suppose that the distribution of the expression values for a gene is distributed according to $N(1.6, 0.42)$.

1. Compute the probability that the expression values are less than 1.2. 2. What is the probability that the expression values are between 1.2 and 2.0? 3. What is the probability that the expression values are between 0.8 and 2.4? 4.

Compute the exact values for the quantiles $x_{0.025}$ and $x_{0.975}$.

5. Use `rnorm` to draw a sample of size 1000 from the population and compare the sample mean and standard deviation to that of the population.

Solution: Normal Model for a gene

```
# a
#########################################################
1 - pnorm(1.2, 1.6, 0.42)

# b
#########################################################
pnorm(2, 1.6, 0.42) - pnorm(1.2, 1.6, 0.42)

# c
#########################################################
pnorm(2.4, 1.6, 0.42)  - pnorm(0.8, 1.6, 0.42)


# d
#########################################################
qnorm(0.025, 1.6, 0.42)
qnorm(0.975, 1.6, 0.42)

# e
#########################################################
test.sample = rnorm(1000, 1.6, 0.42)
mean(test.sample)
sd(test.sample)
```

Exercise: Rocky mountain spotted fever

In 747 cases of "Rocky Mountain spotted fever" from the western United States, 210 patients died. Out of 661 cases from the eastern United States, 122 died. Is the difference statistically significant? Use a prop–test as well as a Fisher–test.

Solution: Rocky mountain spotted fever

```
deaths <-c(210,122)
tot.cases <-c(747,661)

prop.test(deaths, tot.cases)

chisq.test(rbind(deaths, tot.cases-deaths))
fisher.test(rbind(deaths, tot.cases-deaths))
```

Exercise: Group comparison for gene GYPC

The gene GYPC plays an important role in regulating the mechanical stability of red cells. I can be found in line 8197 of the expALL data set. (Try `grep("GYPC", anno_fusALL$SYMBOL))`.

Test for the equality of the means by an appropriate t–test. Is the experimental effect very strong? Also, try testing the hypothesis using a moderated t–test and a wilcoxon test.

Solution: Group comparison for gene GYPC

```
t.test(expALL[8197,] ~ groupsALL, var.equal=FALSE)

### strong difference between groups ...
### confirmed by wilcoxon test
wilcox.test(expALL[8197,] ~ groupsALL)

### the moderated t-test is also significant
modt.stat(t(expALL), groupsALL)[8197]
### p-value using the normal distribution
2 - 2*pnorm(abs(modt.stat(t(expALL), groupsALL)[8197]))
```

Exercise: Multiple testing for the ALL data

Try other multiple testing procedures like q–values on the p–values obtained from the shrinkage t statistics. Can you gain power? Produce a p–value histogram where significant statistics are indicated by color-fill.

Solution: Multiple testing for the ALL data

```
golub.qval <- Qvalue(pval.st)
significant <- as.factor(golub.qval$qValues < 0.05)
levels(significant) <- ifelse(levels(significant) , "yes", "no")

table(significant)

ggplot2::qplot(pval.st , xlab = "p-values of shrinkage t statistics",
               main = "Histogram of p-values,
       golub data, q-value < 0.05",
       fill = significant)
```

```
  `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```

```
significant <- as.factor(pv.st.BH < 0.05)
levels(significant) <- ifelse(levels(significant) , "yes", "no")

(ggplot2::qplot(pval.st , xlab = "p-values of shrinkage t statistics",
                main = "Histogram of p-values,
       golub data, BH adjusted p-value < 0.05",
       fill = significant)  + scale_fill_brewer(type = "qual", palette = 8))
```

```
  `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```