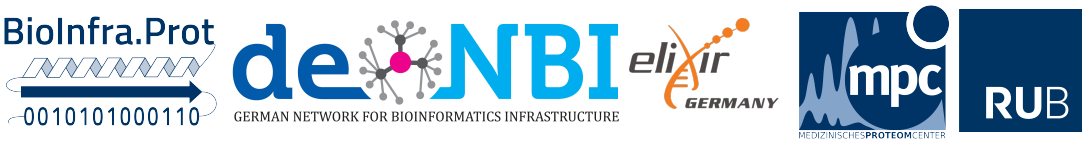
Differential analysis of quantitative proteomics data using R

Solution

Michael Turewicz, Karin Schork

November 2, 2020



Contents

*Optional advanced exercise.

1 Hands-on session Ia

General hints:

- have a look at the slides to solve the exercises
- write down your code in an editor (e.g. the upper left window of RStudio, see slide 23)
- also look at the help pages of the mentioned R functions

Exercise 1.1: Pocket calculator

Open R and use it as *"pocket calculator"* for the following computational tasks:
$10 + 5$,
$1/2$,
$(3 + 5) * 2$,
$3 + 5 * 2$,
$3 + 5^2$
and $100/0$.

```
> 10 + 5

[1] 15

> # This is a comment which is not executed.
> 1 / 2 # This is also a comment.

[1] 0.5

> (3 + 5) * 2

[1] 16

> 3 + 5 * 2

[1] 13

> 3 + 5 ^ 2

[1] 28

> 100 / 0

[1] Inf
```

Exercise 1.2: Variables

Generate a variable x with value 2 and one variable y with value 5. Then compute the following expressions: $x + y$, $log_{10}(x \cdot y)$ and y^x.

```
> x <- 2
> y <- 5

> x + y

[1] 7

> log10(x * y)

[1] 1

> y ^ x

[1] 25
```

Choose some of the functions mentioned in the script and use `?` or `??` to learn more about them. Terminate the R session correctly via `q()` without saving the workspace.

Exercise 1.3: Vectors

Create a vector x containing the numbers 5, 2, 4, 6 and 43.

```
> x <- c(5, 2, 4, 6, 43)
```

Set the third element of x to 10.

```
> x[3] <- 10
```

Replace the last element of the vector by its negative value and print the modified vector x into the console using the function `print()`.

```
> x[length(x)] <- -x[length(x)]
> print(x)

[1]   5   2  10   6 -43
```

Create the vector $y = (4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5)$ in a *"clever"* way and print the vector y into the console.

```
> y <- rep(4:5, each = 7)
> print(y)

 [1] 4 4 4 4 4 4 4 5 5 5 5 5 5 5
```

Exercise 1.4: Matrices

Create the following matrices:

$$X = \begin{bmatrix} 3 & 8 \\ 6 & 2 \end{bmatrix} \qquad Y = \begin{bmatrix} 6 & 3 \\ 9 & 2 \\ 3 & 7 \end{bmatrix}$$

Index the matrices to the following elements, rows and columns of X and Y:

- the upper right element of X
- the lower left element of Y
- the first row of X
- the second column of Y

```
> x <- c(3, 6, 8, 2)
> y <- c(6, 9, 3, 3, 2, 7)
> X <- matrix(x, ncol = 2, nrow = 2)
> Y <- matrix(y, ncol = 2, nrow = 3)
> print(X)

     [,1] [,2]
[1,]    3    8
[2,]    6    2

> print(Y)

     [,1] [,2]
[1,]    6    3
[2,]    9    2
[3,]    3    7

> X[1, 2]

[1] 8

> Y[3, 1]

[1] 3

> X[1,]

[1] 3 8

> Y[,2]

[1] 3 2 7
```

Exercise 1.5: Data frames

Use the function `data.frame()` to create a table with the following content:

Type	Diameter	Height	Age
Oak	57	15.7	29
Beech	38	12.1	18
Birch	23	8.5	10
Chestnut	63	17.3	27
Beech	41	15.1	21

```
> Z <- data.frame(
+   "Type" = c("Oak", "Beech", "Birch", "Chestnut", "Beech"),
+   "Diameter" = c(57, 38, 23, 63, 41),
+   "Height" = c(15.7, 12.1, 8.5, 17.3, 15.1),
+   "Age" = c(29, 18, 10, 27, 21)
+ )
> print(Z)

      Type Diameter Height Age
1      Oak       57   15.7  29
2    Beech       38   12.1  18
3    Birch       23    8.5  10
4 Chestnut       63   17.3  27
5    Beech       41   15.1  21
```

First, index the data in order to obtain the diameters of the trees.

```
> Z$Diameter

[1] 57 38 23 63 41

> Z[, "Diameter"]

[1] 57 38 23 63 41
```

Then, index the data in order to obtain the information on beeches.

```
> Z[Z$Type == "Beech",]

   Type Diameter Height Age
2 Beech       38   12.1  18
5 Beech       41   15.1  21

> Z[Z[, "Type"] == "Beech",]
```

```
   Type Diameter Height Age
2 Beech       38   12.1  18
5 Beech       41   15.1  21
```

Finally, index the data in order to find trees that are 20 years or older.

```
> Z[Z$Age >= 20,]

      Type Diameter Height Age
1      Oak       57   15.7  29
4 Chestnut       63   17.3  27
5    Beech       41   15.1  21

> index <- Z$Age >= 20
> Z[index,]

      Type Diameter Height Age
1      Oak       57   15.7  29
4 Chestnut       63   17.3  27
5    Beech       41   15.1  21

> Z[Z[, "Age"] >= 20,]

      Type Diameter Height Age
1      Oak       57   15.7  29
4 Chestnut       63   17.3  27
5    Beech       41   15.1  21
```

2 Hands-on session Ib

Exercise 1.6: t-test

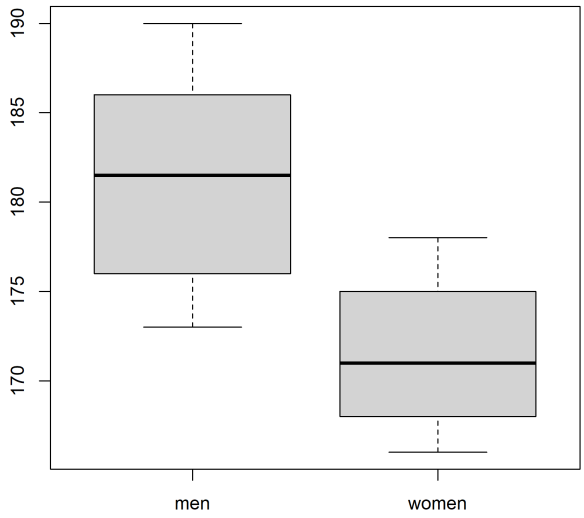
The following table lists the body height of six men and six women in cm. Create two vectors named `men` and `women` with the respective values. Then conduct a t-test (function `t.test()`) on the data to investigate the difference between men and women. Finally plot a box plot using the command

```
boxplot(men, women, names = c("men", "women"))
```

to visualize this difference.

men	186	176	182	173	190	181
women	172	168	178	166	175	170

```
> men <- c(186, 176, 182, 173, 190, 181)
> women <- c(172, 168, 178, 166, 175, 170)
> print(men)

[1] 186 176 182 173 190 181

> print(women)

[1] 172 168 178 166 175 170

> t.test(men, women)

        Welch Two Sample t-test

data:  men and women
t = 3.1367, df = 9.0444, p-value = 0.01192
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.746939 16.919727
sample estimates:
mean of x mean of y
 181.3333  171.5000

> test.results <- t.test(men, women)
> p <- test.results$p.value
> print(p)

[1] 0.01191992

> boxplot(men, women, names = c("men", "women"))
```



Exercise 1.7: ANOVA*

The following table gives body weight in kg of different dogs. Investigate the difference between breeds (T terrier, S sausage dog, P pekinese) by means of ANOVA. Pass the data as a `data.frame()` to the function `aov()`. In case of rejection conduct the post-hoc test *"Tukey's honest significant difference"* using the function `TukeyHSD()`.

breed	T	T	T	S	S	S	P	P	P
weight	7.6	8.1	8.7	5.7	5.6	4.8	6.2	5.9	6.5

```
> breed <- c("T", "T", "T", "S", "S", "S", "P", "P", "P")
> weight <- c(7.6, 8.1, 8.7, 5.7, 5.6, 4.8, 6.2, 5.9, 6.5)
> print(breed)

[1] "T" "T" "T" "S" "S" "S" "P" "P" "P"

> print(weight)

[1] 7.6 8.1 8.7 5.7 5.6 4.8 6.2 5.9 6.5

> dogs <- data.frame("Breed"=breed, "Weight"=weight)
> aov.results <- aov(Weight ~ Breed, data=dogs)
> summary(aov.results)

            Df Sum Sq Mean Sq F value   Pr(>F)
Breed        2 12.087   6.043   28.48 0.000866 ***
```

```
Residuals    6  1.273   0.212
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> tukey <- TukeyHSD(aov.results)
> tukey

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Weight ~ Breed, data = dogs)

$Breed
          diff        lwr       upr     p adj
S-P -0.8333333 -1.9874348 0.3207682 0.1467621
T-P  1.9333333  0.7792318 3.0874348 0.0051290
T-S  2.7666667  1.6125652 3.9207682 0.0007901
```

3 Hands-on session II

After considering the basics of R, the following exercises will be directly related to the differential analysis of your quantitative proteomics data. Thus, in the following you may also imagine a scenario where hundreds or thousands of proteins have been quantified (e.g., with label-free or targeted MS-based proteomics) in a set of samples from two or more groups of interest. E.g., in a preclinical study aiming at the detection of biomarker candidates these groups may be "diseased" and "healthy controls". Now, we will train the skills to detect the best biomarker candidates by computing p-values and fold changes (i.e., the ratios of means of the considered groups) using R.

Exercise 2.1: Data import

The folder *"data"* on the USB drive contains several files that are part of exercise 2.1. Import the files `example_import.txt`, `r_workshop_final_data1.txt`, `r_workshop_final_data1.csv`, `r_workshop_final_data2.txt` and `r_workshop_final_data2.csv`.
(hint: inspect the help pages of the R functions `read.table()` and `read.csv()`, see also slides 42-44).
It is always a good idea to have a short look into the data first (e.g., in order to see whether there is a header and which column separator is used).

After assigning it to the variable `dat1` inspect the imported file `r_workshop_final_data1.txt` using the functions `print()`, `dim()`, `head()` and `tail()`. In order to use `dat1` for the following exercises transform it into a matrix using the command `dat1 <- as.matrix(dat1)`.

```
> ex.imp <- read.table(file = "C:/USB_drive/data/example_import.txt",
+            header = TRUE, sep = "\t", skip = 1, dec = ",")
> dat1 <- read.table(file = "C:/USB_drive/data/r_workshop_final_data1.txt",
+           header = TRUE, sep = "\t")
> dat1 <- as.matrix(dat1)
> dat2 <- read.table(file = "C:/USB_drive/data/r_workshop_final_data2.txt",
+           header = TRUE, sep = "\t")
> # read csv files
> dat1_csv <- read.csv(file = "C:/USB_drive/data/r_workshop_final_data1.csv",
+               header = TRUE, sep = ",")
> dat2_csv <- read.csv(file = "C:/USB_drive/data/r_workshop_final_data2.csv",
+               header = TRUE, sep = ",")

> dim(dat1)

[1] 1000   10

> head(dat1)
```

```
           D1        D2         D3        D4         D5        C1        C2
[1,]  56.22302 250.23056 116.351115  60.43167   2.858646  45.68594  16.64358
[2,]  46.04548 898.77619  77.288900  44.52311  74.491848  22.89094  20.66956
[3,] 177.39347  29.87137 211.799695 130.09407  48.632016  25.46085 410.42190
[4,]  74.08954  65.19188   9.852425 167.25783 123.486250 117.47252  34.64786
[5,]  79.11307  95.39491  80.270471  57.08211  73.378632  32.51832  61.40129
[6,] 182.37052  77.76701  71.987875 155.26117  17.509283  34.20217  64.88913
            C3        C4        C5
[1,]   98.67317 111.61690  79.98854
[2,]  142.10068  53.73327  57.16795
[3,] 7821.05165  94.34441 402.20339
[4,]  114.48980  49.67095  95.10256
[5,]   13.12541 340.16019  10.54165
[6,]  117.50140  39.46954  66.53234
```

```
> tail(dat1)
```

```
              D1        D2        D3        D4        D5        C1        C2
 [995,]  31.16891 208.51977  44.19517  21.01304 110.67876  21.31686 204.06733
 [996,] 108.65741 391.15793 134.23431 105.06943 139.59064  42.27218 108.23129
 [997,]  21.02006  42.31329  41.07080  26.76687 313.67813 179.88242  26.52937
 [998,]  47.32198  37.17189 106.88043 160.81656  46.41185  33.42306 207.53380
 [999,] 116.68821 179.91666  10.40108  33.44296  56.48293  35.43963 146.90769
[1000,] 310.96568  87.52385  26.65298 115.94185  77.91967 111.06444 493.24191
              C3        C4        C5
 [995,]  63.23266 183.09222  45.23825
 [996,]  19.85858  39.47642  23.36415
 [997,] 505.67621  70.51979 109.32380
 [998,]  80.41132 103.60585 392.81379
 [999,] 134.52777  14.87456 485.68065
[1000,]  40.45813 495.29092  23.38209
```

Exercise 2.2: Descriptive statistics

Please use row 581 from `dat1` (from exercise 2.1) and compute characteristics like mean, variance, and extrema for group *"D"* and group *"C"*. To this end, create a vector of group membership using the command `groups <- substr(colnames(dat1), 1, 1)`. Then save the values of group *"D"* (i.e., `dat1[581, groups == "D"]`) in a vector `d` and the last five values (i.e., `dat1[581, groups == "C"]`) in a vector `c`.

Additionally, check the output of the function `summary()`. Finally draw a boxplot of the log2-transformed values using `boxplot(log2(d), log2(c))`.

```
> groups <- substr(colnames(dat1), 1, 1)
> print(groups)
```

```
 [1] "D" "D" "D" "D" "D" "C" "C" "C" "C" "C"
```

```
> print(groups == "D")
```

```
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
> d <- dat1[581, groups == "D"]
> c <- dat1[581, groups == "C"]
> print(d)
```

```
       D1        D2        D3        D4        D5
340.50259  81.37332  64.85623 606.51813  74.24439
```

```
> print(c)
```

```
       C1        C2        C3        C4        C5
8981.5332 3760.2228 9655.9759 4229.3835  900.8078
```

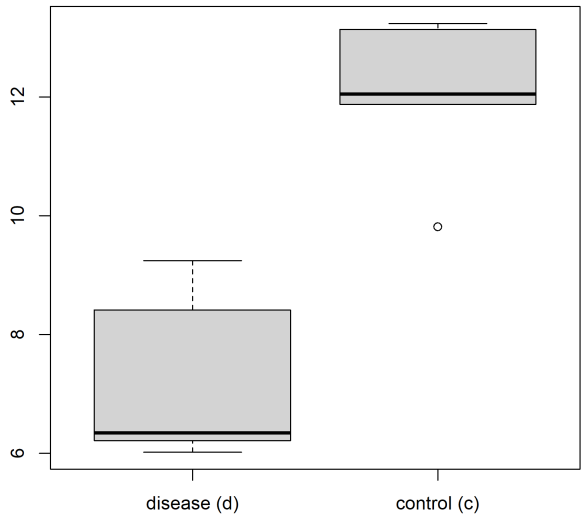
```
> summary(d)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  64.86   74.24   81.37  233.50  340.50  606.52
```

```
> summary(c)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  900.8  3760.2  4229.4  5505.6  8981.5  9656.0
```

```
> boxplot(log2(d), log2(c), names = c("disease (d)", "control (c)"))
```

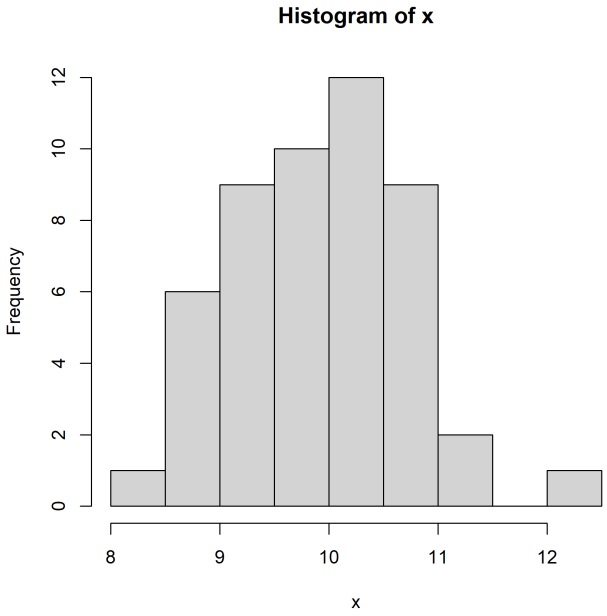


Exercise 2.3: Create plots

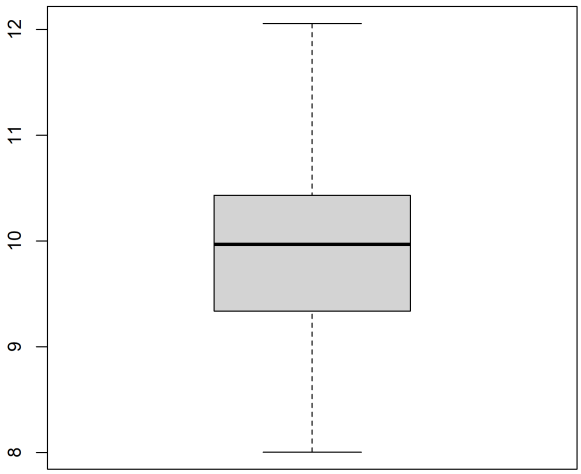
By using the function `rnorm(n, mean, sd)` one can draw `n` random numbers from a normal distribution with mean `mean` and standard deviation `sd`. Create a vector x containing 50 random numbers from a normal distribution with mean 10 and standard deviation 1. Then create a vector y containing 50 random numbers from a normal distribution with mean 0 and standard deviation 0.5. Sum the vectors and store them in a vector z. Create a histogram as well as a boxplot of the vector x. Create a scatterplot of vector z versus vector x. Create a barplot of the first 10 entries of y. (Hint: For the basic plot functions take a look at slides 54 and 57.)

```
> set.seed(211116)
> # With the function set.seed() you can specify a seed for the random
> # number calculation (i. e. you will always get the same random
> # numbers if you use this specific seed, which makes your code
> # reproducible). As you certainly worked with different random
> # numbers, your plots will look slightly different from the plots
> # in the example solution.
>
> x <- rnorm(n = 50, mean = 10, sd = 1)
> print(x)

 [1]  9.188176  9.711714  9.088691 10.431763 12.055489 10.342903  8.003182
 [8] 10.107873 10.676410 10.178412 10.556958  9.993561 10.003564  8.843506
[15]  9.335883  9.718007  9.248407  9.433064  9.654865  9.674087 10.291494
[22] 10.777486  8.903243  9.787190  8.601405  8.651310 10.016429 10.163946
[29] 10.626269  8.830325  9.632519 10.284691 10.168481  9.945029  9.431301
[36] 11.115149 10.326968  8.783820 10.579430  9.832321  9.678545 10.795904
[43] 10.815745  9.104587 10.691120 10.366245  9.269076 11.320779  9.439527
[50] 10.806473

> y <- rnorm(n = 50, mean = 0, sd = 0.5)
> print(y)

 [1] -0.591038175 -0.639241106 -1.026783129  0.181217617  1.717525869
 [6] -0.099967899 -0.584836321 -0.143269192  0.673445597  0.118847185
[11]  0.771763651 -0.788336991 -0.159976926 -0.230984982 -1.227528176
[16] -0.711447731 -0.265371098 -0.470712652  0.215286433  0.393475517
[21] -1.233882963  0.394246137 -0.238119730  0.585161750 -0.205366634
[26] -0.247993310  0.159828613  0.318514841 -0.531705679  0.203429090
[31]  1.243484431  0.310482606 -0.178411472 -0.137556560  0.757707971
[36]  0.778313792 -0.060868604  0.449421754 -0.789757745 -0.390074144
[41] -0.457054501 -0.420473851 -0.360321621 -0.158818866 -1.073220198
[46]  0.006143631 -0.300869848  0.133380290  0.478024509 -0.988588333
```

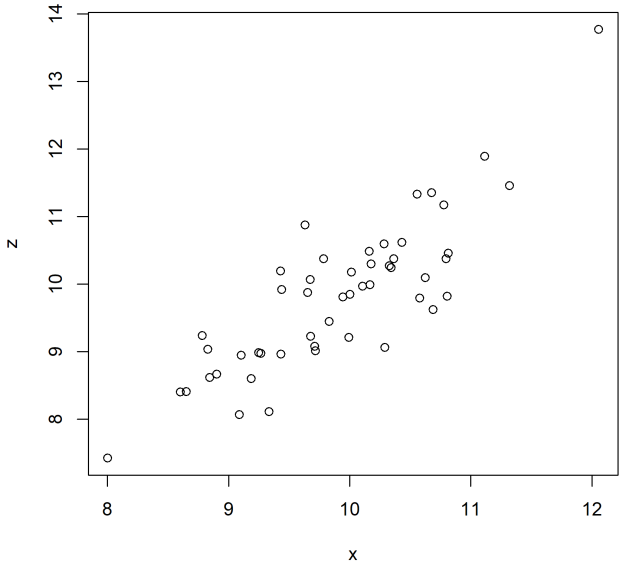
```
> z <- x + y
> print(z)

 [1]  8.597138  9.072473  8.061908 10.612980 13.773015 10.242935  7.418346
 [8]  9.964604 11.349856 10.297259 11.328722  9.205224  9.843587  8.612521
[15]  8.108354  9.006559  8.983036  8.962351  9.870151 10.067562  9.057611
[22] 11.171732  8.665124 10.372352  8.396038  8.403317 10.176258 10.482461
[29] 10.094563  9.033754 10.876003 10.595173  9.990069  9.807472 10.189009
[36] 11.893462 10.266100  9.233241  9.789672  9.442247  9.221490 10.375430
[43] 10.455423  8.945768  9.617900 10.372388  8.968206 11.454160  9.917552
[50]  9.817885

> hist(x)
```



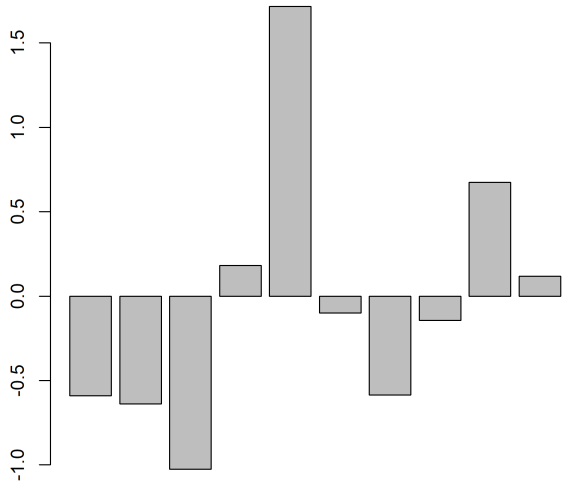

```
> boxplot(x)
```




```
> plot(x,z)
```




```
> barplot(y[1:10])
```

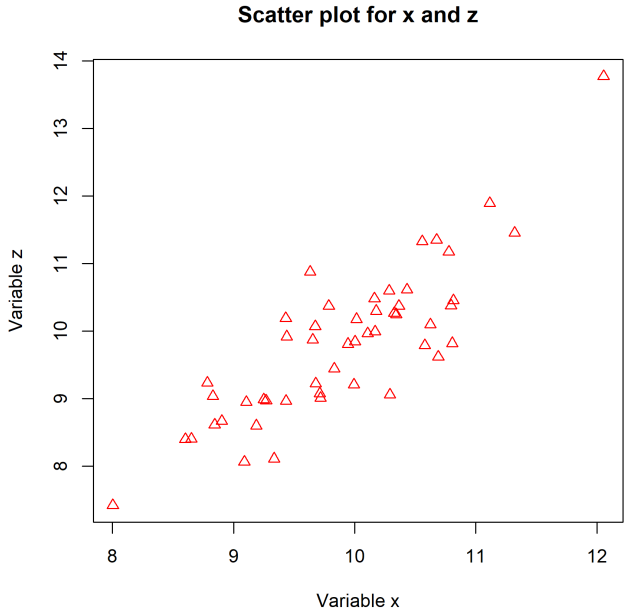


Exercise 2.4: Graphics parameters

Recreate the scatter plot of exercise 2.3. Add an appropriate title and label the axes with "Variable x" and "Variable z". Instead of the default black circles use red triangles. Finally, save the graphic as png file into the results file of your USB drive or on the computer.

Hints: Take a look at slide 55 and the example on slide 58. You can get a list of predefined colours using the command `colors()`. Possible plot symbols are shown on the help page of the function argument `pch`. How to save a graphic is explained on slide 59.

```
> plot(x, z, main = "Scatter plot for x and z", xlab = "Variable x",
+     ylab = "Variable z", col = "red", pch = 2)
```




```
> png(filename = "C:/USB_drive/results/scatterplot.png")
>     plot(x, z, main = "Scatter plot for x and z", xlab = "Variable x",
+     ylab = "Variable z", col = "red", pch = 2)
> dev.off()
```

Exercise 2.5: For-loop

Use a for-loop to calculate and print the first ten square numbers $(1^2, 2^2, ..., 10^2)$. Save the calculated numbers in a vector called `square`.

```
> square <- rep(NA, 10)
> for (i in 1:10) {
+   n <- i^2
+   print(n)
+   square[i] <- n
+ }

[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
```

```
[1] 81
[1] 100

> print(square)

 [1]   1   4   9  16  25  36  49  64  81 100
```

Exercise 2.6: Calculation of p-values and fold changes

In this exercise we will use the dataset `dat1` that you read into `R` in exercise 2.1. Make sure that you have transformed `dat1` into a matrix (`is.matrix(dat1)` should give the result `TRUE`).

Construct a for-loop which iterates over all 1000 rows of `dat1` and computes row-wise p-values (t-test) and mean ratios (fold changes) between the two groups. The t-test should be performed on the log2-transformed values (function `log2()`), while the mean ratios should be calculated on the untransformed data. Create a vector of p-values (`p`) and a vector of mean ratios (`fc`) that contain the p-values resp. mean ratios for all rows. Finally, to manage the multiple testing problem adjust the p-values using the function `p.adjust()` with the method `fdr`.

Hints: Take a look at slides 46-47. Construct the for-loop in a text editor (e.g. the upper left window of RStudio) before you test it in the R console.

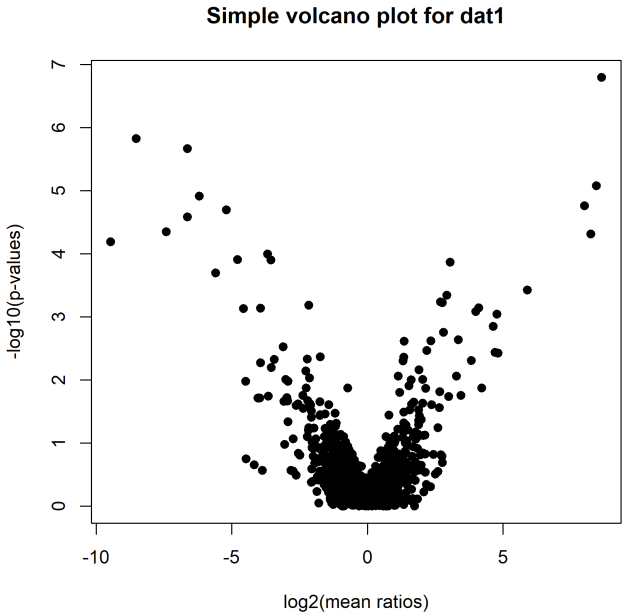
```
> row.number <- 1000
> p.value <- rep(NA, row.number)
> mean.ratio <- rep(NA, row.number)
> for(i in 1:row.number) {
+   x <- dat1[i, groups == "D"]
+   y <- dat1[i, groups == "C"]
+   p <- t.test(log2(x), log2(y), var.equal = TRUE)$p.value
+   p.value[i] <- p
+
+   mr <- mean(dat1[i, groups == "D"]) / mean(dat1[i, groups == "C"])
+   mean.ratio[i] <- mr
+ }
> fdr.p.value <- p.adjust(p.value, method = "fdr")
```

Exercise 2.7: Simple volcano plot

Draw a simple volcano plot using the results from exercise 2.6. To this end, compute the `-log10` of the (not adjusted) p-values and the `log2` of the mean ratios. Then plot the resulting log-mean-ratios vs the resulting log-p-values. Also add a plot title and axis labels.

```
> log.p <- -log10(p.value)
> log.mr <- log2(mean.ratio)
```

```
> plot(log.mr, log.p, main = "Simple volcano plot for dat1",
+      xlab = "log2(mean ratios)", ylab = "-log10(p-values)", pch = 19)
```



Exercise 2.8: Data export

Combine the results of exercise 2.6 with the original data `dat1` in order to obtain the summary of a simple differential analysis. To this end, use the function `cbind()` to combine `dat1` with the vectors containing the p-values, adjusted p-values and mean ratios. Inspect the resulting table via `head()`. Then export the resulting table using the function `write.table()` as txt file into the results folder of your USB drive or on the computer. Finally, open the resulting file in excel.

```
> output <- cbind(dat1, mean.ratio = mean.ratio, p.value = p.value,
+   fdr.p.value = fdr.p.value)
> head(output)
```

```
           D1        D2         D3        D4         D5        C1        C2
[1,]  56.22302 250.23056 116.351115  60.43167   2.858646  45.68594  16.64358
[2,]  46.04548 898.77619  77.288900  44.52311  74.491848  22.89094  20.66956
[3,] 177.39347  29.87137 211.799695 130.09407  48.632016  25.46085 410.42190
[4,]  74.08954  65.19188   9.852425 167.25783 123.486250 117.47252  34.64786
[5,]  79.11307  95.39491  80.270471  57.08211  73.378632  32.51832  61.40129
[6,] 182.37052  77.76701  71.987875 155.26117  17.509283  34.20217  64.88913
            C3        C4        C5 mean.ratio   p.value fdr.p.value
[1,]  98.67317 111.61690  79.98854 1.37857008 0.8416710   0.9710049
```

```
[2,]  142.10068  53.73327  57.16795 3.84784280 0.2665390   0.8769424
[3,] 7821.05165  94.34441 402.20339 0.06829175 0.2709575   0.8769424
[4,]  114.48980  49.67095  95.10256 1.06926434 0.7773270   0.9646280
[5,]   13.12541 340.16019  10.54165 0.84159877 0.3268096   0.8820571
[6,]  117.50140  39.46954  66.53234 1.56510957 0.5667536   0.9428277
```

```
> file.out <- "C:/USB_drive/results/exercise_2_8.txt"
> write.table(x = output, file = file.out, sep = "\t", row.names = FALSE,
+   col.names = TRUE)
```

Exercise 2.9: Improved volcano plot*

Draw an improved volcano plot (like on slide 51) with the following features:

- A horizontal dashed line indicates the p-value threshold of 0.05.
- Two vertical dashed lines indicate the respective mean ratio threshold on each side of the volcano plot.
- All differentially expressed proteins (e.g., p-value < 0.05 and mean ratio > 2 or $< \frac{1}{2}$) are highlighted in a specific color.
- The proteins that are significant after the multiple testing correction are highlighted in another color.

Hints: For drawing the lines the function `abline()` is useful. You can use the function `which()` (see slide 19) to select the differential and very best features.

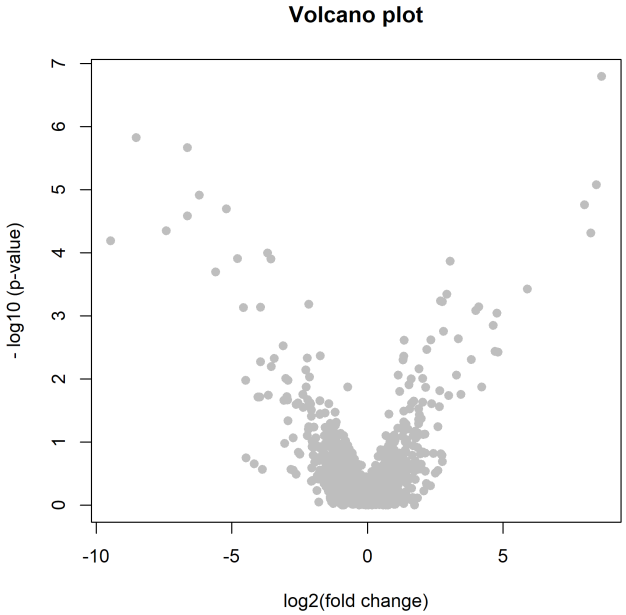
```
> dim(dat1)
```

```
[1] 1000   10
```

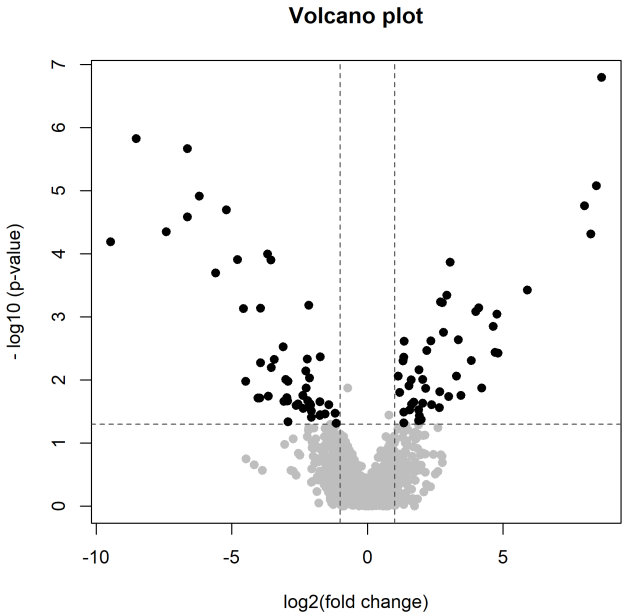
```
> head(dat1)
```

```
            D1        D2         D3        D4         D5        C1        C2
[1,]  56.22302 250.23056 116.351115  60.43167   2.858646  45.68594  16.64358
[2,]  46.04548 898.77619  77.288900  44.52311  74.491848  22.89094  20.66956
[3,] 177.39347  29.87137 211.799695 130.09407  48.632016  25.46085 410.42190
[4,]  74.08954  65.19188   9.852425 167.25783 123.486250 117.47252  34.64786
[5,]  79.11307  95.39491  80.270471  57.08211  73.378632  32.51832  61.40129
[6,] 182.37052  77.76701  71.987875 155.26117  17.509283  34.20217  64.88913
            C3        C4        C5
[1,]   98.67317 111.61690  79.98854
[2,]  142.10068  53.73327  57.16795
```

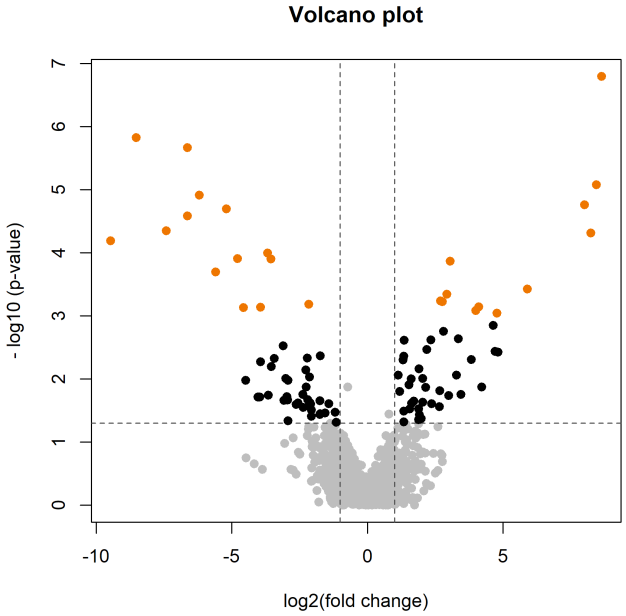
```
[3,] 7821.05165  94.34441 402.20339
[4,]  114.48980  49.67095  95.10256
[5,]   13.12541 340.16019  10.54165
[6,]  117.50140  39.46954  66.53234

> # In case there are additional columns containing information on accession or
> # the like make sure to select the actual subtable of values.
>
> # Check for on-off proteins (only present in one group)
> sum(!is.finite(mean.ratio))

[1] 0

> sum(mean.ratio == 0)

[1] 0

> # find differential features
> differential.mr <- (mean.ratio >  2 | mean.ratio < 1/2)
> differential.p <- (p.value < 0.05)
> differential <- (differential.mr & differential.p)
> which.differential <- which(differential)
> # number of differential features
> length(which.differential)

[1] 96

> # data to plot
> plot.data <- cbind(log.mr = log2(mean.ratio), log.p = -log10(p.value))
> # plot parameters
> main <- "Volcano plot"
> xlab <- "log2(fold change)"
> ylab <- "- log10 (p-value)"
> plot(plot.data, col = "grey", main = main, xlab = xlab, ylab = ylab, pch = 19)
```




```
> plot(plot.data, col = "grey", main = main, xlab = xlab, ylab = ylab, pch = 19)
> abline(h = -log10(0.05), lty = 2, col = "grey30",
+   v = log2(c(2, 1/2)))
> points(plot.data[differential,], col = "black", pch = 19)
```



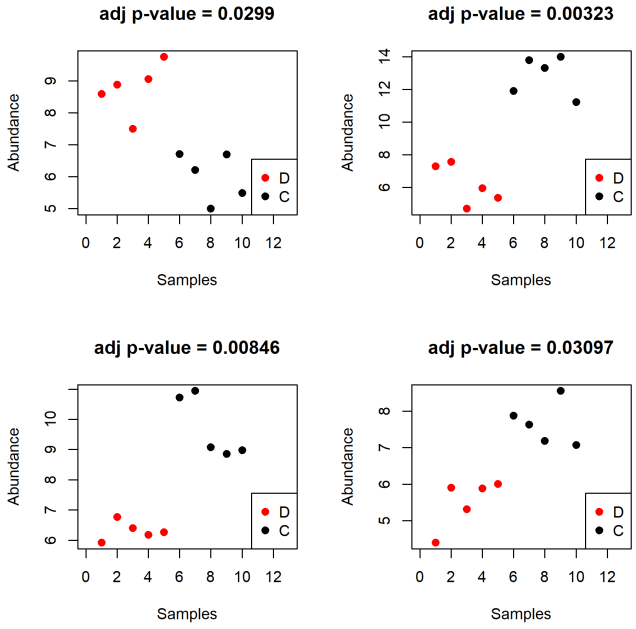
```
> plot(plot.data, col = "grey", main = main, xlab = xlab, ylab = ylab, pch = 19)
> abline(h = -log10(0.05), lty = 2, col = "grey30",
+   v = log2(c(2, 1/2)))
> points(plot.data[differential,], col = "black", pch = 19)
> # selection of proteins that show a significant difference between the
> # two groups after the multiple testing correction
> best.proteins <- which(fdr.p.value < 0.05)
> points(plot.data[best.proteins,], col = "darkorange2", pch = 19)
```



Exercise 2.10: Abundance plot*

Plot the abundance profiles of a few (e.g., 2 or 4) interesting differentially expressed proteins (like on slide 52). Try to include some numeric information of the differential analysis in the plots (e.g., the adjusted p-values).
Hint: You can use the function `paste0()` to print the value of an object to use it for example in the plot title or within the function `text()`.

```
> par(mfrow = c(2,2))
> for(b in 1:4){
+   plot(log2(as.numeric(dat1[best.proteins[b], ])), col = rep(c("red", "black"),
+     each = 5), pch = 19, ylab = "Abundance", xlab = "Samples", xlim = c(0, 13),
+     main = paste0("adj p-value = " ,
+       format(round(fdr.p.value[best.proteins[b]], 5), scientific = FALSE)))
+   legend("bottomright", legend = c("D", "C"), col = c("red", "black"), pch = 19)
+ }
```


adj p-value = 0.0299
adj p-value = 0.00323
adj p-value = 0.00846
adj p-value = 0.03097
Abundance
Samples
D
C