Exercise 1.1:

If X to be normally distributed with some values for mean and variance, say, $X{\sim}N(\mu,\sigma^2)$, is the mean of $Y = e^X$ then

$a)$ $e^{\mu+\sigma^2/2}$ or

$b)$ $e^{\mu-\sigma^2/2}$?

Use the `rnorm()` function and the other functions from the first slide in the introduction session to figure it out.

Exercise 1.2:

Consider the second order polynomial equation

$$X^2 + 3X + 1 = 0 \qquad (a)$$

The general polynomial equation

$$AX^2 + BX + C = 0$$

has the solutions

$$\frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

1. Construct a vector of length 2 that contains the solutions to the equation (a), and display it on the screen with 1 decimal point.
2. Work out how much error you make (in percent) by referring to the solutions with just one decimal. The result will be different for the two solutions.

Exercise 1.3:

Construct the object x as

```
x <- rnorm(100, mean=.5, sd=.3)
```

Perform the following task (check out the help system as needed):

1. Calculate mean and standard deviation of x.

2. Plot a histogram of x.

3. Put the second axis on the right side of the histogram plot instead of on the left.

Exercise 2.1:

a) Study the function paste() via its help page and use it to paste together the string ”R session” and the number 1. What type of object is returned?

b) What happens if we try to add a number to a data frame containing only numbers?

c) What happens if we try to add a number to a data frame with both number columns and character columns?

Exercise 2.2:

a) Create a function that takes a vector of numbers and returns the standardized values; ie. the data with the mean subtracted and rescaled so that the variance is 1.

b) Describe what happens if you apply your function to a vector of length 1, and why.

Exercise 2.3:

Consider the function

$$f(x) = 3\sin\left(\frac{x}{2}\right) + x$$

a) Define a similar function *f* in R. Calculate $f(0)$, $f(-1)$ and $f(\pi)$.

b) Write a script that plots the function *f* over the interval from -7 to 7. Save the script on your system and close it. Then, execute the script in R with the source() command.

Exercise 3.1a:

a) Write a loop that investigates the contents of a matrix my.data. For each row starting from the first, the loop should write a line in the output window which differs according to the contents of the ith row of my.data:

   i. If the ith row of my.data do not contain any negative values, the line should contain the sentence: “The mean of row number i is”, where i should be the row number, and then the mean of the ith row of my.data.
   ii. If the ith row contains negative values, but no more than 3 lines with negative numbers have been encountered, the line should contain the sentence “<Row i contains negative values>”, where again i is the row number.
   iii. If the ith row contains negative values, and at least 3 lines with negative numbers have been encountered, the line should contain the sentence “Too many negative values”.

   If the line “Too many negative values” have been written, the loop should stop writing out lines immediately, otherwise it should stop when the last line of my.data have been investigated an a line has been written.

Exercise 3.1b:

b) Construct a dataset in the following way:

```
set.seed(1786)
data.exercise.3.1<-exp(matrix(rnorm(2000),nrow=100))
index1.temp<-sample(1:100,10)
index2.temp<-sample(1:20,10)
for(i in 1:10){
 data.exercise.3.1[index1.temp[i],index2.temp[i]]<--1
}
```

Assign the dataset data.exercise.3.1 to my.data and check that your algorithm works. You should write out 14 lines.

Exercise 3.2:

a) The function `Sys.time()` returns the current time. In the slide where we compared run times of elementwise squaring a matrix y containing elements generated with the `rnorm()` function, and doing it using the object-oriented form of R. We looked at a huge matrix with 1.000.000 elements. What is the effect for small matrices? Redo the runtime exercise with a matrix of dimension 10x10 instead of 1000x1000, and calculate the runtime increase factor. Comment.

b) Now, let k be the number of rows in the quadratic matrix y. calculate runtime increase factors as above for k=10, 20, 50, 100, 200, 500, 800 and 1000, and plot the runtime increase factors as a function of the number of elements in y.

Exercise 3.3:

1. Create a data frame with 10 rows and 100.000 columns, and fill it with random numbers generated with `rnorm()`. We want to calculate the mean of all 100.000 columns, and place it in a vector.

2. Calculate the runtime factor from looping over the 100.000 columns and using the mean function, in contrast to using `sapply()`.

Exercise 4.1:

a) Explain what happens in the following matrix construction:

```
matrix(1:6,nrow=3,ncol=3)
```

b) What happens if we do not give any input vector at all when we define a matrix?

Exercise 4.2:

a) Explain what goes on here:

```
x<-matrix(1:12,4)
x
x[cbind(c(1,3,2),c(3,3,2))]
```

b) Calculate the following and explain the difference with (a)

```
x[c(1,3,2),c(3,3,2)]
```

Exercise 4.3

Create matrices row, column and A in the following way:

```
row<-matrix(rep(1:100,100),nrow=100)
column<-matrix(rep(1:100,100),nrow=100,byrow=T)
A<-3*column^3/(1+row*column)
```

a) Find an easy way to calculate the sum

$$\sum_{i=1}^{100}\sum_{j=1}^{100}\frac{3i^3}{1+ij}$$

b) Next, find an easy way to calculate

$$\sum_{i=1}^{100}\sum_{j=1}^{i}\frac{3i^3}{1+ij}$$

Exercise 5.1

1. Read the data from the file data.exercise5.1.dat into a data frame in R. If you have problems, study the help file accessed with `?read.table`

2. Describe `read.table()`'s convention for columns without a name.

Exercise 5.2

1. Read the data from the file data.exercise5.2.dat into a data frame in R. If you have problems, study the help file accessed with `?read.table`.

2. Which items does R interpret as missing here?

3. What would have happened if the blank space had been in column w rather than in column y?

Exercise 5.3

Read the contents of the file Exercise 5.3.xlsx into an R data frame.

Exercise 5.4

1. Read the data in the file Exercise 5.4a.txt into a data frame.

2. Read the data in the file Exercise 5.4b.txt into a list.

Exercise 6.1

Create two data frames with the commands

```
set.seed(9007)
my.data<-data.frame(x=rnorm(10),y=rnorm(10)+5,z=rchisq(10,1))
additional.data<-data.frame(x=rnorm(3),y=rnorm(3)+5,z=rchisq(3,1))
```

1. write my.data to a file called "Exercise 6.txt" (preferably in a Data subdirectory), without row and column names.

2. You realize that you have more data that you need to write to the file. Consult `?read.table`, and figure out a way to add the contents of the dataset additional.data to the contents of the file "Exercise 6.txt".

Exercise 6.2

Create the data my.data as

```
set.seed(45)
my.data<-data.frame(x=rnorm(10),y=rnorm(10),z=rnorm(10))
```

1. Write the data to a csv file with write.csv2. If your regional settings determines that write.csv is compatible with Excel standards rather than `write.csv2`, use `write.csv` instead.

2. Open the file with Excel, save it as an xlsx workbook, and confirm that the file export has happened as it should.

Exercise 6.3

1. Make R write the following data:

```
"a";"b"
"A";1
"B";2
"C";3
"D";4
"E";5
```

2. Write the following data file:

```
TITLE extra line
2 3 5 7
11 13 17
One more line
```

Exercise 6.4

Create the data my.data as Exercise 6.2

```
set.seed(45)
my.data<-data.frame(x=rnorm(10),y=rnorm(10),z=rnorm(10))
```

1. Save it to your system with the `save()` function.

2. Remove my.data from the R memory with the `rm()` function.

3. Then, read `my.data` back into the system, and display `head(my.data)` on the screen.

Exercise 7.1

1. Find the column names and types of the table ’sentiment’ from the SQL server `msedxeus.database.windows.net` described on the slides, without fetching the entire table.

2. Find the number of rows in ’sentiment’ without fetching the entire table.

Server Name: msedxeus.database.windows.net
Database Name: DAT209x01
User ID: Rlogin
Password: P@ssw0rd

Exercise 7.2

Construct a dataset based on the table ‘sentiment’ from Exercise 7.1 with average score by Date for the State ’WA’. The dataset should also contain the column ’Date’.

Server Name: msedxeus.database.windows.net
Database Name: DAT209x01
User ID: Rlogin
Password: P@ssw0rd

Exercise 8.1

Create the two data frames

```
data.frame.x<-data.frame(names=c("Gretha","Robert","John","Heather"),
                         age=c(30,18,25,70),
                         nickname=c("Quicksilver","The Man","Nifty","Starlight"))
data.frame.y<-data.frame("Person_name"=c("William","Nancy","Charlotte","Henry),
                         age=c(15,75,32,51),
                         "pet_dog"=c("King","Whity","Captain Vom","Doggie"))
```

1. What happens if you merge `data.frame.x` and `data.frame.y`, without specifying a ‘by’ option? What happens if you merge `data.frame.y` and `data.frame.x`, without specifying a ‘by’ option?

2. Merge this information into the data frame `data.frame.z`, which should contain the 4 columns `“Person_name”`, `”age”`, `”pet_dog”` and `“nickname”.` Display the data frame.

Exercise 8.2

Create a subset of the built-in data set `iris`, `setosa.data`, containing the data for the species setosa, where the sepal length is less than the median for all data. `setosa.data` should of course not contain a column with the species name.

Exercise 8.3

Create a text objects as

```
my.text<-“Over the last decade, bluetongue virus have
spread northwards from the mediterranean area. Initially
this was ascribed to climate changes, but it has since
been realized that a major contributing factor has been
new transmitting vectors, culicoides obsoletus and
culicoides pulicaris, which have the ability to aquire
and transmit the disease. Recently, schmallenberg virus
has emerged in northern europe, transmitted by biting
midges as well.”
```

Change the object my.text so that the words bluetongue, culicoides, europe, mediterranean, northern and schmallenberg all starts with a capital letter.

Exercise 8.4

Create as set of date-time objects (POSIXct):

```
set.seed(885)
my.posixct<-as.POSIXct(sample((60*60*24*365*50):(60*60*24*365*55),20),
                      origin = as.Date("1960-01-01"))
```

Display my.posixct. Create a new set of `date.time` objects, `my.posixct2`, by adding 2 hours, 30 minutes and 10 seconds to the date-time objects in `my.posixct`. Display the head of a data frame with `my.posixct` and `my.posixct2`.

Exercise 9.1

The built in presidents dataset contains quarterly approval ratings for American presidents from 1945 to 1974. Use the `cycle()` function and `tapply()` to obtain a vector of quarterly approval ratings. Consider how to handle missing data.

Exercise 9.2

Sometimes you may wish to group data according to a continuous variable. Study the `cut()` function, and use it to construct 10 intervals that covers the values `airquality$Wind`, and work out group means of `airquality$Solar.R` according to grouped values of wind with `tapply()`.

Exercise 9.3

Sometimes you may wish to calculate means according to groupings of more than 1 variable. Use the `cut()` function on the built-in dataset `swiss` to cut the variables `Agriculture` and `Catholic` into 10 intervals, and use this to construct a matrix of mean values of the `Fertility` variable, according to the cross-classification by `Agriculture` and `Catholic`.

Exercise 10.1

With low sample sizes, it can be hard to detect if data indeed follow a normal distribution. Construct a frame for plotting 9 plots on you graphics device with the `par()` function given the option `mfrow=c(3,3)`. Then plot 9 histograms of 25 simulated standard normally distributed numbers. Add the density of the standard normal to the plot with the `curve()` function. Study the `par()` function to find options that sets the line color to red and the line width to 3, and incorporate this. Look at the histograms. Do they resemble a normal distribution?

Exercise 10.2

Cleanse the ozone variable in the airquality data for missings and remove the the value which attains the minimum 1:

```
my.ozone<-airquality$Ozone[!is.na(airquality$Ozone) & airquality$Ozone>1]
```

a) If my.ozone should be normally distributed, the best guess of the mean and standard deviation would be:

   ```
   mean.1<-mean(my.ozone)
   sd.1<-sd(my.ozone)
   ```

   Simulate a number of normally distributed numbers with mean mean.1 and standard deviation sd.1, equal to the amount of data in my.ozone. Compare the two sets of points in a qqplot with the qqplot() function. Add a line corresponding to the identity of the two sets with the lines() function. Comment on the plot.

b) Now, consider a log-transform of the data. If the log-transformed data should be normally distributed, a best guess on mean and standard deviation would be:

   ```
   mean.2<-mean(log(my.ozone))
   sd.2<-sd(log(my.ozone))
   ```

   Simulate a number of normally distributed numbers with mean `mean.2` and standard deviation `sd.2`, equal to the amount of data in `my.ozone`. Compare the exponential to the simulated points with `my.ozone` in a qqplot, with the `qqplot()` function. Add a line corresponding to the identity of the two sets with the `lines()` function. Comment on the plot.

Exercise 10.3

Consider the following set up: You roll two dice, and count the number x of eyes on the dice. You then roll a number of dice corresponding to x, and record the number y of eyes on the dice. Simulate the process 1.000 times, and make a histogram of the values of y.

Exercise 11.1

Consider the ozone variable from the `airquality` data, which we investigated in Exercise 10.2.

a) Construct a lm object where you regress log(ozone) on the variables solar radiation, wind and temperature. As in Exercise 10.2, leave out the data line where Ozone is equal to 1.

b) Compare the residuals of the lm object from a) with the normal distribution in a plot using the `qqnorm()` function. Add a line to the plot, with no intercept and a slope equal to the standard deviation of the model residuals. As in Exercise 10.2, give the line a width of 3 and color it red. Comment on the plot.

Exercise 11.2

Expand the liner model from Exercise 11.1 with pairwise interactions between solar radiation, wind and temperature. Reduce the model with the `drop1()` function, removing insignificant factors sequentially, always removing the one with the highest p-value (above 0.05). Are there any indications of interactions in the model?

Exercise 11.3

Install the package `glm2`, and load it with the library statement. Look at the `crabs` data:

```
> head(crabs)
  Satellites Width Dark GoodSpine Rep1 Rep2
1          8  28.3   no        no    2    2
2          0  22.5  yes        no    4    5
3          9  26.0   no       yes    5    6
4          0  24.8  yes        no    6    6
5          4  26.0  yes        no    6    8
6          0  23.8   no        no    8    8
```

The variable ‘Satellites’ describes a number of hopeful male horseshoe crabs that, in addition to the sexual partner, attach themselves to the females during mating. We shall look at the event that the female horseshoe crab has a satellite, and thus look at the data in the following form:

```
crab.data<-data.frame(satellite=1*(crabs$Satellites>0),width=crabs$Width)
```

a) Perform a logistic regression of the probability of having a satellite on the width of the female crab. Does the probability of a satellite increase or decrease with the width of the crab?

b) Find the linear predictors with the `predict()` function, and their standard errors. See `predict.glm` for how to use `predict()` on glm objects. Construct a data frame `my.linear.predictor` that contains the linear predictors, the lower confidence limit (linear predictor minus 1.96 times the standard error) and the upper confidence limit.

c) Order the rows of `my.linear.predictor` according to `crab.data$width`, and transform `my.linear.predictor` with the logistic function. Use this to plot the predicted values of the probability of a satellite as a function of width, with dashed lines indicating upper and lower confidence limits.

d) Use the `cut()` and `tapply()` functions to create 5 grouped means of satellite by `width`. Plot them on the graph with the `lines()` function, where the option `type="p"` (for points) is specified, together with the option `pch=16` (filled circle). Comment.

Exercise 11.4

Recreate the diabetes analysis from the slides with the data my.diabetes data, by loading the datasets `diabetes.data` and `new.diabetes.data`:

```
diabetes.data<-read.csv2("Data/my.diabetes.data.csv")
new.diabetes.data<-read.csv2("Data/my.new.diabetes.data.csv")
```

You should remove the first column which contains line numbers, to make the code fit to the slides:

```
diabetes.data<-diabetes.data[,-1]
new.diabetes.data<-new.diabetes.data[,-1]
```

Repeat the model creation from the slides, and store the analysis of `diabetes.data` in `my.analysis`. Obtain the linear predictors for `new.diabetes.data`, and transform them with the logistic function to obtain the model predictors.

Create a density plot for the model predictors for which the variable `readmi_class` is equal to `YES`. Set the option `ylim=c(0,4)` in the graph creation. Add a density curve with the `lines()` function for the model predictions for which `readmi_class` is equal to `NO`. Use a dashed line. Comment.

Exercise 12.1

Create a plot of depth versus price from the diamonds data with `ggplot()`. Add a point geom and a 2d density geom. Display the plot.

Exercise 12.2

Consider the diamonds data, and cut the depth variable into 5 groups with the `cut()` function. Then create a density plot with price grouped after plot, with filled density curves. Display the plot, and comment.

Exercise 12.3

Download map data for the USA map on the slides. Access the built-in dataset state.x77 to obtain knowledge on the murder rate in the individual states per 100.000 (as of 1976). Overlay the map with circles according to the murder rate for each state.