Homework 2

1. Make sure to include your name in the file.
2. Imagine a factor that contains "categories" **c,a,z,s**.

- How many levels does this factor have? Answer: 4.
- Make an object of class factor named **f** containing the **c,a,z,s** (as factor!).

```
f <- as.factor(c("c","a","z","s"))
```

- Now make an object of class character named **c** that contains the same letters.

```
c <- c("c","a","z","s")
```

- Have a look at the two objects. Why is the order different in the levels of **f** in comparison with **c**? Answer: because factor levels are ordered alphabetically.
- Change the levels of the factor **f** so that the order is **c,a,z,s**.

```
f <- factor(f, levels=c("c","a","z","s","p"))
levels(f)
```

```
## [1] "c" "a" "z" "s" "p"
```

3. Create a for loop for a range between 1 and 8, it has to print each element of the range + 5.

```
for (i in 1:8){
  print(i +5)
}
```

```
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
```

4. Use one of the two functions in R that help you find out what these two functions do: cbind() rbind()

```
?cbind
help(rbind)
```

5. Create a new vector containing the name of 3 of your friends.

```
names <- c("Mary", "John", "Joe")
```

6. Create another vector containing their ages.

```
age <- c(28,32,34)
```

7. Join the two previous vectors and create a new object named "friends1". Do that by using the correct function: cbind or rbind?

```
friends1 <- cbind(names,age)
```

8. Now create a new object named "friends2" with the vectors from ex. 5 and 6 but this time use the function data.frame() and give names to your columns, i.e., "names" and "age".

```
friends2 <- data.frame(age=age, names=names)
```

9. Change the class of the second column of friends2 to **factor**.

```
friends2$names <- as.factor(friends2$names)
```

10. Which is the difference between a matrix and a data frame? A matrix can only contain one type of data, whereas dataframes can combine multiple types of data.

11. With the variables created below, type the code to find out whether: a is greater or equal than c; b is equal to a; c is less than b. Is c smaller than b? why? (hint: check the class of the variables)

```
a <- 20
b <- "20"
c <- 10

a >= c
```

```
## [1] TRUE
```

```
b == a
```

```
## [1] TRUE
```

```
c < b
```

```
## [1] TRUE
```

Object b is of class character. It should be converted to numeric if we want to use logical operations properly.

12. **If statements:** create a variable 'x' and assign the value '6' to it. Then create an if statement: If x is equal or bigger than 0, create a new variable called 'y' that is equal to 'x' multiplied by 10 (and print y). Else, the variable 'y' will be equal to x multiplied by 5 (and print y). Then run the code.

```r
x <- 6
if (x >= 0){
  y <- x*10
  print(y)
}else{
  y <- x*5
  print(y)
}
```

```
## [1] 60
```

13.

a) Create a data frame with two columns. The first column, named condition, has 10 observations corresponding to condition **a**. The second column, called group, contains two factors: L1 and L2. The group column contains 10 observations, 10 for each group

```r
mydata <- data.frame(condition=rep(c("a","b"),each=5),
                     group=rep(c("L1","L2"),5))
```

b) add a new column called grp_contrast. If the column group has value L1, the column grp_contrast should have -1, otherwise 1. Use the ifelse function.

```r
mydata$grp_contrast <- ifelse(mydata$group=="L1",-1,1)
```

c) add a new column called grp_cond1. If condition is a and group is L1, values of grp_cond1 should be 1, otherwise 0 (hint: the ifelse expression can be as long as needed; you can add more than one condition by joining them using logical operators).

```r
mydata$grp_cond1 <- ifelse(mydata$group=="L1" & mydata$condition=="a",1,0)
```

c) add a new column called grp_cond2. If condition is not a and group is not L1, values of grp_cond1 should be 1, otherwise 0.

```r
mydata$grp_cond2 <- ifelse(mydata$group!="L1" & mydata$condition!="a",1,0)
```

d) add a new column called grp_cnd3. If condition is a or b, and group is L2, values of grp_cnd3 should be 1, otherwise 0. (Use the logical operators discussed in class.)

```r
mydata$grp_cond3 <- ifelse(mydata$group=="L2" & mydata$condition=="a"|
                             mydata$group=="L2" & mydata$condition=="b",1,0)
```

e) print the content of the data frame to check that you did the previous exercises correclty.

```r
print(mydata)
```

```
##    condition group grp_contrast grp_cond1 grp_cond2 grp_cond3
## 1          a    L1           -1         1         0         0
## 2          a    L2            1         0         0         1
## 3          a    L1           -1         1         0         0
## 4          a    L2            1         0         0         1
## 5          a    L1           -1         1         0         0
## 6          b    L2            1         0         1         1
## 7          b    L1           -1         0         0         0
## 8          b    L2            1         0         1         1
## 9          b    L1           -1         0         0         0
## 10         b    L2            1         0         1         1
```

14.

a) Using the previous data frame, create a new data frame that contains only the observations corresponding to the L1 group.

```
mydata2 <- droplevels(subset(mydata,group=="L1"))
```

b) create a new column called RT that contains data points drawn from a lognormal distribution with location 6 and scale 1. (hint: rlnorm). Remember that the number of data points needs to be the same as the number of observations (or rows) in the data frame.

```
mydata2$RT <- rlnorm(nrow(mydata2),6,1)
```

c) Compute the mean and the sd of the RT column, and round it up.

```
mean(mydata2$RT)
```

```
## [1] 523.2527
```

```
sd(mydata2$RT)
```

```
## [1] 374.7542
```

d) change all the column names to whatever names you like.

```
colnames(mydata2) <- c("col1", "col2", "col3", "col4","col5", "col6")
```

15. In the range 1 to N, where N=100, sample 50 data points from a normal distribution with mean 40 and sd 10. Then take the mean of each sampled distribution, and store it in a previously created vector with length equal to N (see the slides for an example of this). Finally, print the content of the vector, which should contain 100 means.

```
N <- 100
sample.means <- rep(NA,N)

for(i in 1:N){
sample.50 <- rnorm(50,mean=40,sd=10)
sample.means[i] <- mean(sample.50)
}

print(sample.means)
```

```
##   [1] 38.17383 39.02336 39.40950 40.33859 42.35096 39.99569 41.35706 40.79221
##   [9] 42.82126 40.96957 41.18262 39.54134 37.40566 39.80375 38.49639 38.29946
##  [17] 37.86075 41.27789 41.76117 41.54925 40.03723 40.03950 39.70275 40.68601
##  [25] 36.04775 42.83697 41.60690 41.35877 39.93386 38.81163 41.64262 38.72788
##  [33] 38.85307 41.79831 39.62565 40.16196 42.70529 38.71184 42.97084 41.19230
##  [41] 39.56486 40.40103 41.09122 38.77752 42.52904 40.39811 39.57500 40.47388
##  [49] 41.67163 40.71095 40.10008 39.64438 38.63538 41.08729 39.37685 38.92671
##  [57] 38.88156 40.40932 37.79919 39.13989 41.42323 39.32020 38.43969 41.18815
##  [65] 41.87033 41.97311 39.86139 41.15885 40.22396 40.08264 39.21461 42.19344
##  [73] 42.81390 38.00123 38.94717 41.29894 41.15242 41.69151 40.70881 40.92452
##  [81] 41.14960 41.15402 37.23669 40.54518 38.00362 40.75620 40.65475 37.33736
##  [89] 37.61204 39.08059 42.27213 39.83254 42.96184 36.95191 41.28695 39.94048
##  [97] 39.51903 40.38733 41.43396 38.16540
```