Introduction to R

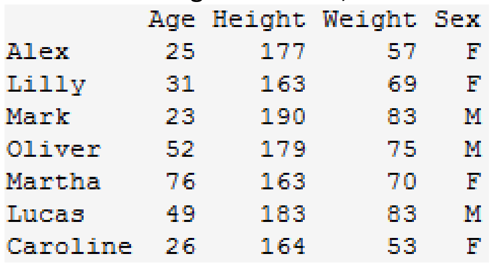
1. Display "Hello, world!" on R terminal.
2. Use R to calculate
   31 * 78
   697/41

3. Create a vector called vec1 containing the number 2 5 8 12 16
4. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.
5. In each case, what is the value of x?
   a) x <- 2 - 1 * 2
   b) x <- 6/3-2+1*0+3/3-3
   c) x <- 19%%17%%13 # compare to (19%%17)%%13 and 19%%(17%%13)
   d) x <- 2ˆ17%%17
   e) x <- 3-2%%5+3*2-4/2

6. In each case, what is the value of x?
   a. x <- sum (1:10 -5)
   b. x <- sum (1:10) -5
   c. x <- 1:10 -10:1
   d. x <- sum (1:10 -10:1)

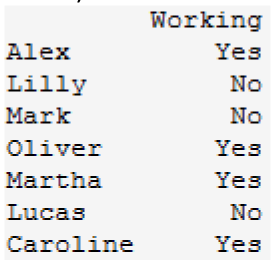
7. Write a R program to create a vector which contains 10 random integer values between -50 and +50.
8. Write a R program to find the maximum and the minimum value of a given vector.
9. Write a R program to read the .csv file and display the content.
10. Write a R program to compute sum, mean and product of a given vector elements.
11. Write a R program to create a Data Frames which contain details of 5 employees (name, gender, designation, salary) and display summary of the data.
12. Write a R program to create an empty data frame.
13. Write a R program to create a data frame from four given vectors.
14. Write a R program to get the structure of a given data frame.
15. Write a R program to get the statistical summary and nature of the data of a given data frame.
16. Write a R program to call the (built-in) dataset airquality. Check whether it is a data frame or not? Order the entire data frame by the first and second column.
17. Write a R program to add two vectors of integers type and length 3.
18. Write a R program to find Sum, Mean and Product of a Vector.
19. Write a R program to add 3 to each element in a given vector. Print the original and new vector.
20. Write a R program to test whether the value of the element of a given vector greater than 10 or not. Return TRUE or FALSE.

21. Create the following data frame, afterwards invert `Sex` for all individuals.



```
         Age Height Weight Sex
Alex      25    177     57   F
Lilly     31    163     69   F
Mark      23    190     83   M
Oliver    52    179     75   M
Martha    76    163     70   F
Lucas     49    183     83   M
Caroline  26    164     53   F
```

22. **Exercise 2**
    Create this data frame (make sure you import the variable `Working` as character and not factor).



```
         Working
Alex         Yes
Lilly         No
Mark          No
Oliver       Yes
Martha       Yes
Lucas         No
Caroline     Yes
```

Add this data frame column-wise to the previous one.
a) How many rows and columns does the new data frame have?
b) What class of data is in each column?

Data Frames exercises with solutions

5.2.1 Create the following data frame, afterwards invert Sex for all individuals.

```
Name <- c("Alex", "Lilly", "Mark", "Oliver", "Martha", "Lucas", "Caroline")
Age <- c(25, 31, 23, 52, 76, 49, 26)
Height <- c(177, 163, 190, 179, 163, 183, 164)
Weight <- c(57, 69, 83, 75, 70, 83, 53)
Sex <- as.factor(c("F", "F", "M", "M", "F", "M", "F"))
df <- data.frame (row.names = Name, Age, Height, Weight, Sex)

levels(df$Sex) # The order of levels: "F" "M". So we can reassign them
differently to invert Sex for all.
## [1] "F" "M"
levels(df$Sex) <- c("M","F")

df
##            Age Height Weight Sex
## Alex        25    177     57   M
## Lilly     31    163     69   M
## Mark      23    190     83   F
## Oliver    52    179     75   F
## Martha    76    163     70   M
## Lucas     49    183     83   F
## Caroline  26    164     53   M
```

5.2.2 Create this data frame (make sure you import the variable Working as character and not factor).

```
Working <- c("Yes","No","No","Yes","Yes","No","Yes")

df2 <- data.frame(row.names = Name,Working) #Name has been already defined in
exercise 1.

df2
##          Working
## Alex         Yes
## Lilly         No
## Mark          No
## Oliver       Yes
## Martha       Yes
## Lucas         No
## Caroline     Yes
```

Add this data frame column-wise to the previous one.
a) How many rows and columns does the new data frame have?
b) What class of data is in each column?

```
df <- cbind(df,df2)

dim(df) #The output is [1] nrow ncolumn
## [1] 7 5
str(df)
## 'data.frame':    7 obs. of  5 variables:
##  $ Age    : num  25 31 23 52 76 49 26
##  $ Height : num  177 163 190 179 163 183 164
##  $ Weight : num  57 69 83 75 70 83 53
##  $ Sex    : Factor w/ 2 levels "M","F": 1 1 2 2 1 2 1
##  $ Working: Factor w/ 2 levels "No","Yes": 2 1 1 2 2 1 2
```

5.2.3 Check what class of data is the (built-in data set) state.center and convert it to data frame.

```
class(state.center) #state.center is a list of 2 numeric vectors.
## [1] "list"
data.state <- as.data.frame(state.center) # Now it is a data frame
```

5.2.4 Create a simple data frame from 3 vectors. Order the entire data frame by the first column.

```
a <- (rnorm(10))
b <- letters[4:13]
c <- c("yes","no","no","no","no","yes","no","yes","yes","no")
```

```
df3 <- data.frame(a,b,c)

df3[with (df3, order(a)),]
##              a b   c
## 5  -1.4874427 h  no
## 6  -0.9261379 i yes
## 8  -0.8676770 k yes
## 10 -0.4776941 m  no
## 2  -0.3444281 e  no
## 9  -0.1275440 l yes
## 4   0.4226900 g  no
## 1   0.4440771 d yes
## 3   0.4682689 f  no
## 7   0.9045589 j  no
```

5.2.5 Create a data frame from a matrix of your choice, change the row names so every row says `id_i` (where i is the row number) and change the column names to `variable_i` (where i is the column number). I.e., for column 1 it will say `variable_1`, and for row 2 will say `id_2` and so on.

```
matrix.data <- matrix(1:40, nrow = 10, ncol = 4)

df <- as.data.frame(matrix.data)

colnames(df) <- paste("variable_", 1:ncol(df))

rownames(df) <- paste("id_", 1:nrow(df))

df
##        variable_ 1 variable_ 2 variable_ 3 variable_ 4
## id_ 1            1          11          21          31
## id_ 2            2          12          22          32
## id_ 3            3          13          23          33
## id_ 4            4          14          24          34
## id_ 5            5          15          25          35
## id_ 6            6          16          26          36
## id_ 7            7          17          27          37
## id_ 8            8          18          28          38
## id_ 9            9          19          29          39
## id_ 10          10          20          30          40
```

5.2.6 For this exercise, we’ll use the (built-in) dataset `VADeaths`.

a. Make sure the object is a data frame, if not change it to a data frame.
b. Create a new variable, named Total, which is the sum of each row.
c. Change the order of the columns so total is the first variable.

```
class(VADeaths) #Which shows that it is a matrix
## [1] "matrix"
df <- data.frame(VADeaths)

df$Total <- rowSums(df)
```

```
df <- df[,c(5,1,2,3,4)]

df
##        Total Rural.Male Rural.Female Urban.Male Urban.Female
## 50-54   44.2       11.7          8.7       15.4          8.4
## 55-59   67.7       18.1         11.7       24.3         13.6
## 60-64  103.5       26.9         20.3       37.0         19.3
## 65-69  161.6       41.0         30.9       54.6         35.1
## 70-74  241.4       66.0         54.3       71.1         50.0
```

5.2.7 For this exercise we'll use the (built-in) dataset `state.x77`.

a. Make sure the object is a data frame, if not change it to a data frame.

```
class(state.x77)
## [1] "matrix"
df2 <- data.frame(state.x77)
```

b. Find out how many states have an income of less than 4300.

```
nrow(df2[df2$Income < 4300,]) # We are finding the number of rows in the
table which is filtered as Income < 4300
## [1] 20
#length(df2$Income[df2$Income < 4300]) # So there are several ways.
#count(df2$Income < 4300) # You can see a frequency table with this statement
```

c. Find out which is the state with the highest income.

```
row.names(df2[which.max(df2$Income),]) # When I need to find a max value,
"which.max" statement comes to my mind in a moment but you can find the
solution with different ways.
## [1] "Alaska"
```

5.2.8 With the dataset swiss, create a data frame of only the rows 1, 2, 3, 10, 11, 12 and 13, and only the variables Examination, Education and Infant.Mortality.

```
df3 <- data.frame(swiss[c(1,2,3,10,11,12,13),c("Examination", "Education",
"Infant.Mortality")])
```

a. The infant mortality of Sarine is wrong, it should be a NA, change it.

```
df3$Infant.Mortality[4] <- NA
```

b. Create a row that will be the total sum of the columns, name it Total.

```
Total <- colSums(df3, na.rm = TRUE) #We have a NA value in 3rd column. That's
why it's safe to ignore NA values.

df4 <- rbind(df3,Total)

rownames(df4) = c(rownames(df3),"Total")
```

```
df4
##              Examination Education Infant.Mortality
## Courtelary            15        12             22.2
## Delemont               6         9             22.2
## Franches-Mnt           5         5             20.2
## Sarine                16        13               NA
## Veveyse               14         6             24.5
## Aigle                 21        12             16.5
## Aubonne               14         7             19.1
## Total                 91        64            124.7
```

c. Create a new variable that will be the proportion of Examination (Examination / Total)

```
Prop <- df4$Examination / df4["Total","Examination"]

Prop
## [1] 0.16483516 0.06593407 0.05494505 0.17582418 0.15384615 0.23076923
## [7] 0.15384615 1.00000000
df4 <- cbind(df4,Prop)

df4
##              Examination Education Infant.Mortality       Prop
## Courtelary            15        12             22.2 0.16483516
## Delemont               6         9             22.2 0.06593407
## Franches-Mnt           5         5             20.2 0.05494505
## Sarine                16        13               NA 0.17582418
## Veveyse               14         6             24.5 0.15384615
## Aigle                 21        12             16.5 0.23076923
## Aubonne               14         7             19.1 0.15384615
## Total                 91        64            124.7 1.00000000
```

5.2.9 Create a data frame with the datasets state.abb, state.area, state.division, state.name, state.region. The row names should be the names of the states.

```
df <- data.frame(state.abb, state.area, state.division, state.region,
row.names = state.name)
```

a. Rename the column names so only the first 3 letters after the full stop appear (e.g. States.abb will be abb).

```
colnames(df) <- substr(colnames(df), 7, 9)
```

5.2.10 Add the previous data frame column-wise to state.x77

```
new.df <- cbind(state.x77,df)
```

a. Remove the variable div.

```
new.df$div <- NULL
```

b. Also remove the variables Life Exp, HS Grad, Frost, abb, and are.

```
#subsetting is easier when you need to remove more columns

new.df <- subset(new.df, ,-c(4, 6, 7, 9, 10))
```

c. Add a variable to the data frame which should categorize the level of illiteracy: [0,1) is low, [1,2) is some, [2, inf) is high.

```
# For such categorization, ifelse statement is very very useful.

new.df$Illiteracy.Levels <- ifelse(new.df$Illiteracy >= 0 & new.df$Illiteracy
< 1, "Low",
                                   ifelse(new.df$Illiteracy >= 1 &
new.df$Illiteracy < 2, "Some",
                                          "High"))
```

d. Find out which state from the west, with low illiteracy, has the highest income, and what that income is.

```
x <- subset(new.df, reg == "West" & Illiteracy.Levels == "Low")

row.names(x[which.max(x$Income),])
## [1] "Nevada"
max(x$Income)
## [1] 5149
```

R Exercises - Loops (For Loop, Which Loop, Repeat Loop), If and Ifelse Statements in R

1. Simple `ifelse` statement

Create the data frame `‘student.df’` with the data provided below:

```
student.df = data.frame( name = c("Sue", "Eva", "Henry", "Jan"),

                         sex = c("f", "f", "m", "m"),

                         years = c(21,31,29,19)); student.df
```

Use a simple `‘ifelse’` statement to add a new column `‘male.teen’` to the data frame. This is a boolean column, indicating T if the observation is a male younger than 20 years.

```
#expected result

name  sex years male.teen

Sue    f     21           F

Eva    f     31           F

Henry m     29           F
```

---

2. `For` loop combined with `if` statement

Write a `for` loop that prints the D*isplacement* (`‘disp’`) of the `‘mtcars’` dataset.
a. This loop will only print observations of 160 or higher in `‘disp’`.
b. This loop will stop as soon as an observation is smaller than 160 in `‘disp’`.

```
#expected result


#a

[1] 160
```

```
[1] 160
[1] 258
[1] 360
[1] 225
[1] 360
[1] 167.6
[1] 167.6
[...]
```

```
#b
[1] 160
[1] 160
```

---

3. Adding a new column to a `data.frame` – NA handling

a. You have the data.frame `'mydf'` with four columns like below.

```
a = c(3,7,NA, 9)
b = c(2,NA,9,3)
f = c(5,2,5,6)
d = c(NA,3,4,NA)

mydf = data.frame(a=a,b=b,f=f,d=d);
mydf
```

b. You want to add another column `'5'`:

- the 5th column contains the value of col 2 if col 1 is NA;
- the 5th column contains the value of col 4 if col 2 is NA;
- the 5th column contains the value of col 3 in all other cases.

```
#expected result
```

```
   a  b f  d V5
1  3  2 5 NA 5
2  7 NA 2  3  3
3 NA 9 5  4  9
4  9  3 6 NA 6
```

---

4. Simple `while` loop

Write a `while` loop starting with x = 0. The loop prints all numbers up to 35 but it skips number 7.

```
#expected result

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 8
[1] 9
[1] 10
[1] 11
[...]
```

---

5. River classifications

Use the *'rivers'* dataset to write a `for` loop. The loop prints the dataset:

- rivers shorter than 500 are a *'short river'*;
- rivers longer than 2000 are a *'long river'*;
- and rivers in the middle range are printed in their *original numbers*.

```
#expected result

[1] 735
[1] "short river"
[1] "short river"
[1] "short river"
[1] 524
[1] "short river"
[1] 1459
[1] "short river"
[1] "short river"
[1] 600
[1] "short river"
[...
```