R-Solution

----------Lab 1----------

```
1. How it works
3 + 4
6 + 12


2. Documenting your code
# Calculate 3 + 4
3 + 4

# Calculate 6 + 12
6 + 12

3. Little arithmetics with R
# Addition
5 + 5

# Subtraction
5 - 5

# Multiplication
3 * 5

 # Division
(5 + 5) / 2

# Exponentiation
2^5

# Modulo
28 %% 6

4. R's pros and cons

statements (1) and (5) are correct; the others are false.

5. Variable assignment

# Assign the value 42 to x
x <- 42

# Print out the value of the variable x
x

6. Variable assignment (2)
# Assign the value 5 to the variable called my_apples
my_apples = 5
```

```
# Print out the value of the variable my_apples
my_apples

7. Variable assignment (3)

# Assign a value to the variables my_apples and my_oranges
my_apples <- 5
my_oranges<- 6

# Add these two variables together and print the result
my_apples+my_oranges

# Create the variable my_fruit
my_fruit

8.The workspace

# Clear the entire workspace
rm(list = ls())

# List the contents of your workspace
ls()

# Create the variable horses
horses <- 3

# Create the variable dogs

dogs <- 7
# Create the variable animals
animals <- horses + dogs

# Inspect the contents of the workspace again

ls()
# Remove dogs from the workspace

rm(dogs)
# Inspect the objects in your workspace once more
ls()

9. Build and destroy your workspace

# Create the variables r and R

r=2
R=6
```

```
# Calculate the volume of the donut: vol_donut

vol_donut=2*pi*pi*r*r*R
# Remove all intermediary variables that you've used with rm()

rm(r,R)
# List the elements in your workspace
ls()

--------------------------------------------------Lab
2----------------------------------------------

1. Basic Data Types
# What is the answer to the universe?
my_numeric <- 42
#my_numeric=42
# The quotation marks indicate that the variable is of type character
my_character <- "forty-two"
#my_character="forty-two"
# Change the value of my_logical
my_logical <- FALSE

2. Back to Apples and Oranges

# Assign a value to the variable called my_apples
my_apples <- 5

# Print out the value of my_apples
my_apples

# Assign a value to the variable my_oranges and print it out
my_oranges <- "six"
my_oranges =6

# New variable that contains the total amount of fruit
my_fruit <- my_apples + my_oranges
my_fruit

3. What's that data type?

a's class is numeric, b is a character, c is a logical.


4. Coercion: Taming your data

# Create variables var1, var2 and var3
var1 <- TRUE
```

```
var2 <- 0.3
var3 <- "i"

# Convert var1 to a character: var1_char
var1_char=as.character(var1)

# See whether var1_char is a character
is.character(var1_char)

# Convert var2 to a logical: var2_log
var2_log=as.logical(var2)

# Inspect the class of var2_log
class(var2_log)

# Coerce var3 to a numeric: var3_num
var3_num=as.numeric(var3)

 5. Coercion for the sake of cleaning

 # Convert age to numeric: age_clean
age_clean=as.numeric(age)

# Convert employed to logical: employed_clean
employed_clean=as.logical(employed)

# Convert salary to numeric: salary_clean
salary_clean=as.numeric(salary)


----------------------------------------------------Lab
3-----------------------------------------------

1. Create a vector (1)

numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")

# Create boolean_vector
boolean_vector <- c(TRUE, FALSE ,TRUE)


2. Create a vector (2)

# Poker winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)

# Roulette winnings from Monday to Friday: roulette_vector
```

```
roulette_vector <- c(-24, -50, 100, -350, 10)

3. Naming a vector (1)

# Poker winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)

# Roulette winnings from Monday to Friday
roulette_vector <- c(-24, -50, 100, -350, 10)

# Add names to both poker_vector and roulette_vector
days <- c( "Monday", "Tuesday", "Wednesday", "Thursday" , "Friday")
names(poker_vector) <-days
names(roulette_vector) <-days
#poker_vector

4. Naming a vector (2)

# Poker winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)

# Roulette winnings from Monday to Friday
roulette_vector <- c(-24, -50, 100, -350, 10)

# Create the variable days_vector
days_vector <- c("Monday","Tuesday","Wednesday","Thursday","Friday")

# Assign the names of the day to roulette_vector and poker_vector
names(poker_vector)  <-  days_vector
names(roulette_vector)  <- days_vector


5.Different ways to create and name vectors

poker_vector1 and roulette_vector1 have the same names, while poker_vector2 and
roulette_vector2 show a names mismatch.

--------------------------------------------------Lab
4-----------------------------------------------
1. Summing and subtracting vectors
# A_vector and B_vector have already been defined for you
A_vector <- c(1, 2, 3)
B_vector <- c(4, 5, 6)

# Take the sum of A_vector and B_vector: total_vector
total_vector =  A_vector + B_vector

# Print total_vector
```

```
total_vector

# Calculate the difference between A_vector and B_vector: diff_vector
diff_vector =  A_vector - B_vector

# Print diff_vector
diff_vector

2. Calculate your earnings

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Calculate your daily earnings: total_daily
total_daily = poker_vector + roulette_vector

3. Calculate total winnings: sum()

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Total winnings with poker: total_poker
total_poker = sum(poker_vector)

# Total winnings with roulette: total_roulette
total_roulette = sum(roulette_vector)

# Total winnings overall: total_week
total_week = sum(poker_vector+roulette_vector)

# Print total_week
total_week

4. Comparing total winnings

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
```

```
names(roulette_vector) <- days_vector

# Calculate poker_better
poker_better <- poker_vector > roulette_vector

# Calculate total_poker and total_roulette, as before
total_poker <- sum(poker_vector)
total_roulette <- sum(roulette_vector)

# Calculate choose_poker
choose_poker <- total_poker > total_roulette

# Print choose_poker
choose_poker

5. First steps in rational gambling
# Calculate total gains for your entire past week: total_past

total_past <- sum(poker_past + roulette_past)
# Difference of past to present performance: diff_poker
diff_poker  <- poker_present - poker_past


----------------------------------------------------Lab
5-----------------------------------------------
1. Selection by index (1)

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Poker results of Wednesday: poker_wednesday
poker_wednesday = c(poker_vector["Wednesday"])

# Roulette results of Friday: roulette_friday
roulette_friday = c(roulette_vector["Friday"])

2. Selection by index (2)

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector
```

```
# Mid-week poker results: poker_midweek
poker_midweek = c(poker_vector[c(2, 3 ,4) ])

# End-of-week roulette results: roulette_endweek
roulette_endweek = c(roulette_vector[c(4 ,5)])


3. Vector selection: the good times (3)

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Roulette results for Tuesday to Friday inclusive: roulette_subset
roulette_subset  <- roulette_vector[c(2:5)]

# Print roulette_subset
roulette_subset

4. Selection by name (1)

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Select Thursday's roulette gains: roulette_thursday

roulette_thursday =  roulette_vector[c(4)]
# Select Tuesday's poker gains: poker_tuesday
poker_tuesday = poker_vector[c(2)]

5. Selection by name (2)

# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Select the first three elements from poker_vector: poker_start
```

```
poker_start <- poker_vector[c(1:3)]

# Calculate the average poker gains during the first three days: avg_poker_start
avg_poker_start = mean(poker_start)
avg_poker_start
```

6. Selection by logicals (1)

```
# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Roulette results for day 1, 3 and 5: roulette_subset
 roulette_subset = roulette_vector[c(1,3,5)]
# Poker results for first three days: poker_start
poker_start =poker_vector[c(1:3)]
```

7. Selection by logicals (2)

```
# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Create logical vector corresponding to profitable poker days: selection_vector
selection_vector <- poker_vector > 0

# Select amounts for profitable poker days: poker_profits
poker_profits <- poker_vector[selection_vector]
```

8. Selection by logicals (3)

```
# Casino winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)
roulette_vector <- c(-24, -50, 100, -350, 10)
days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
names(poker_vector) <- days_vector
names(roulette_vector) <- days_vector

# Select amounts for profitable roulette days: roulette_profits
roulette_profits <- roulette_vector[roulette_vector > 0]

# Sum of the profitable roulette days: roulette_total_profit
```

```
roulette_total_profit <- sum(roulette_profits)

# Number of profitable roulette days: num_profitable_days
num_profitable_days <- sum(roulette_vector > 0)

9. Vectors: place your bets

# Select the player's score for the third game: player_third
player_third = player[c(3)]

# Select the scores where player exceeds house: winning_scores
winning_scores <- player[player > house]

# Count number of times player < 18: n_low_score
n_low_score <- sum(player < 18)

--------------------------------------------------Lab
6-----------------------------------------------
1. Analyzing matrices, you shall
# Star Wars box office in millions (!)
box <- c(460.998, 314.4, 290.475, 247.900, 309.306, 165.8)

# Create star_wars_matrix
star_wars_matrix <- matrix(box,nrow=3,byrow=TRUE)
star_wars_matrix

2. Analyzing matrices, you shall (2)
# Star Wars box office in millions (!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)

# Create star_wars_matrix
star_wars_matrix = rbind(new_hope,empire_strikes,return_jedi)

star_wars_matrix

3. Naming a matrix
# Star Wars box office in millions (!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)
star_wars_matrix <- rbind(new_hope, empire_strikes, return_jedi)

# Name the columns and rows of star_wars_matrix
col_names_vector <- c("US","non-US")
row_names_vector <- c("A New Hope", "The Empire Strikes Back","Return of the Jedi")
```

```
colnames(star_wars_matrix) <- col_names_vector
rownames(star_wars_matrix) <- row_names_vector

4. D
5. Calculating the worldwide box office
# Star Wars box office in millions (!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)
star_wars_matrix <- rbind(new_hope, empire_strikes, return_jedi)
colnames(star_wars_matrix) <- c("US", "non-US")
rownames(star_wars_matrix) <- c("A New Hope", "The Empire Strikes Back", "Return of
the Jedi")

# Calculate the worldwide box office: worldwide_vector
worldwide_vector = rowSums(star_wars_matrix)
worldwide_vector


6. Adding a column for the Worldwide box office
# Star Wars box office in millions (!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)
star_wars_matrix <- rbind(new_hope, empire_strikes, return_jedi)
colnames(star_wars_matrix) <- c("US", "non-US")
rownames(star_wars_matrix) <- c("A New Hope", "The Empire Strikes Back", "Return of
the Jedi")

# The worldwide box office figures
worldwide_vector <- rowSums(star_wars_matrix)

# Bind the new variable worldwide_vector as a column to star_wars_matrix:
star_wars_ext
star_wars_ext = cbind(star_wars_matrix,worldwide_vector)

7. Adding a row
# Matrix that contains the first trilogy box office
star_wars_matrix

# Matrix that contains the second trilogy box office
star_wars_matrix2

# Combine both Star Wars trilogies in one matrix: all_wars_matrix
all_wars_matrix = rbind(star_wars_matrix,star_wars_matrix2)
all_wars_matrix

8. The total box office revenue for the entire saga
```

```
# Print box office Star Wars
all_wars_matrix

# Total revenue for US and non-US: total_revenue_vector
total_revenue_vector <- colSums(all_wars_matrix)

9. Matrices: Move it up a notch

theater <- rbind(first_row, second_row, third_row, fourth_row)
row_scores <- rowSums(theater)
scores <- cbind(theater, row_scores)
rownames(scores) <- c("row1", "row2", "row3", "row4")
colnames(scores) <- c("c1", "c2", "c3","c4", "c5", "c6", "total")


----------------------------------------------------Lab
7-----------------------------------------------

1. Select elements
# star_wars_matrix is already defined in your workspace
star_wars_matrix
# US box office revenue for "The Empire Strikes Back"
star_wars_matrix[2,1]

# non-US box office revenue for "A New Hope"
star_wars_matrix[1,2]

2. Select rows and columns
# star_wars_matrix is already defined in your workspace

# Select all US box office revenue
star_wars_matrix[,1]

# Select revenue for "A New Hope"
star_wars_matrix[1,]

# Average non-US revenue per movie: non_us_all
non_us_all = mean(star_wars_matrix[,2])

# Average non-US revenue of first two movies: non_us_some
non_us_some = mean(star_wars_matrix[c(1,2),2])

3. Create sub matrices
# star_wars_matrix is already defined in your workspace

# All figures for "A New Hope" and "Return of the Jedi"
star_wars_matrix[c(1,3), c(1,2)]
star_wars_matrix[c(1,3), ]
```

4.Alternative ways of sub setting

```
# star_wars_matrix is already defined in your workspace

# Select the US revenues for "A New Hope" and "The Empire Strikes Back"
star_wars_matrix[c("A New Hope", "The Empire Strikes Back"), "US"]

# Select the last two rows and both columns
star_wars_matrix[c(FALSE, TRUE, TRUE), c(TRUE, TRUE)]

# Select the non-US revenue for "The Empire Strikes Back"
star_wars_matrix ["The Empire Strikes Back" ,c(FALSE,TRUE)]
```

5. Be selective

3

6. Subsetting: The final challenge

```
view_count_all <- cbind(view_count_1, view_count_2)
view_count_loud <- view_count_all[,c(3,6,7)]
total_views_loud <- colSums(view_count_loud)
```

----------Lab 8----------

1. Arithmetic with matrices

```
# Star Wars box office in millions (!)
new_hope <- c(460.998, 314.4)
empire_strikes <- c(290.475, 247.900)
return_jedi <- c(309.306, 165.8)
star_wars_matrix <- rbind(new_hope, empire_strikes, return_jedi)
colnames(star_wars_matrix) <- c("US", "non-US")
rownames(star_wars_matrix) <- c("A New Hope", "The Empire Strikes Back", "Return of
the Jedi")

# Estimation of visitors
visitors <- star_wars_matrix / 5

# Print the estimate to the console
visitors
```

2.Arithmetic with matrices (2)

```
# Star Wars box office in millions (!)
box_office_all <- c(461, 314.4, 290.5, 247.9, 309.3, 165.8)
movie_names <- c("A New Hope","The Empire Strikes Back","Return of the Jedi")
col_titles <- c("US","non-US")
star_wars_matrix <- matrix(box_office_all, nrow = 3, byrow = TRUE, dimnames =
list(movie_names, col_titles))
```

```
# Definition of ticket_prices_matrix
ticket_prices_matrix <- matrix(c(5, 5, 6, 6, 7, 7), nrow = 3, byrow = TRUE, dimnames
= list(movie_names, col_titles))

# Estimated number of visitors
visitors <- star_wars_matrix / ticket_prices_matrix

# Average number of US visitors
average_us_visitors <- mean(visitors[ ,1])


# Average number of non-US visitors
average_non_us_visitors <- mean(visitors[ ,2])

3. May the matrix force be with you!

# Calculate the money that remains after subtracting the commission: remaining
remaining = star_wars_matrix-commission_rates*star_wars_matrix

# Calculate income per film: remaining_tot
remaining_tot = rowSums(remaining)

# Calculate profit
profit = remaining_tot -budget
--------------------------------------------------Lab
9------------------------------------------------

1. Vector to factor

# Definition of hand_vector
hand_vector <- c("Right", "Left", "Left", "Right", "Left")

# Convert hand_vector to a factor: hand_factor
hand_factor <- factor(hand_vector)

# Display the structure of hand_factor
str(hand_factor)

2. Factor levels

# Definition of survey_vector
survey_vector <- c("R", "L", "L", "R", "R")

# Encode survey_vector as a factor with the correct names: survey_factor
survey_factor <- factor(survey_vector, levels = c("R", "L"), labels = c("Right",
"Left"))
survey_factor_2 <- factor(survey_vector, levels = c("L", "R"), labels = c("Left",
```

```
"Right")) # also possible

# Print survey_factor
survey_factor

3. Summarizing a factor

# Defintion of survey_vector and survey_factor
survey_vector <- c("R", "L", "L", "R", "R")
survey_factor <- factor(survey_vector, levels = c("R", "L"), labels = c("Right",
"Left"))

# Summarize survey_vector
summary(survey_vector)

# Summarize survey_factor
summary(survey_factor)

4. Nominal versus Ordinal, Unordered versus Ordered
# Definition of animal_vector and temperature_vector
animal_vector <- c("Elephant", "Giraffe", "Donkey", "Horse")
temperature_vector <- c("High", "Low", "High", "Low", "Medium")

# Convert animal_vector to a factor: animal_factor
animal_factor <- factor(animal_vector)

# Encode temperature_vector as a factor: temperature_factor
temperature_factor <- factor(temperature_vector, order = TRUE, levels = c("Low",
"Medium", "High"))

# Print out animal_factor and temperature_factor
animal_factor
temperature_factor

5. Left better than Right
# Definition of survey_vector and survey_factor
survey_vector <- c("R", "L", "L", "R", "R")
survey_factor <- factor(survey_vector, levels = c("R", "L"), labels = c("Right",
"Left"))

# First element from survey_factor: right
right <- survey_factor[1]

# Second element from survey_factor: left
left <- survey_factor[2]

# Right 'greater than' left?
right > left
```

```
6. Ordered factors
# Create speed_vector
speed_vector <- c("OK", "Slow", "Slow", "OK", "Fast")

7.Ordered factors (2)
# Create speed_vector
speed_vector <- c("OK", "Slow", "Slow", "OK", "Fast")

# Convert speed_vector to ordered speed_factor
speed_factor <- factor(speed_vector, ordered = TRUE, levels= c("Slow", "OK",
"Fast"))

# Print speed_factor
speed_factor

# Summarize speed_factor
summary(speed_factor)

8.Comparing ordered factors

# Definition of speed_vector and speed_factor
speed_vector <- c("Fast", "Slow", "Slow", "Fast", "Ultra-fast")
speed_factor <- factor(speed_vector, ordered = TRUE, levels = c("Slow", "Fast",
"Ultra-fast"))

# Compare DA2 with DA5: compare_them
compare_them <- factor_speed_vector[2] > factor_speed_vector[5]

# Print compare_them: Is DA2 faster than DA5?
compare_them

9. Flying high in factor space
# Prespecification of levels and labels
lvls <- c("eco", "bus", "fir")
lbls <- c("economy", "business", "first")

# Encode fly_class as a factor, with the appropriate names and ordering

fly_class_factor <- factor(fly_class,
                           levels = c("eco", "bus", "fir"),
                           ordered = TRUE,
                           labels = c("economy", "business", "first"))

--------------------------------------------------Lab
10-----------------------------------------------
1. Create a list
```

```
# Numeric vector: 1 up to 10
my_vector <- 1:10

# Numeric matrix: 1 up to 9
my_matrix <- matrix(1:9, ncol = 3)

# Factor of sizes
my_factor <- factor(c("M","S","L","L","M"), ordered = TRUE, levels = c("S","M","L"))

# Construct my_list with these different elements
my_list = list(my_vector,my_matrix,my_factor)

2. Listception: lists in lists

# Numeric vector: 1 up to 10
my_vector <- 1:10

# Numeric matrix: 1 up to 9
my_matrix <- matrix(1:9, ncol = 3)

# Factor of sizes
my_factor <- factor(c("M","S","L","L","M"), ordered = TRUE, levels = c("S","M","L"))

# List containing vector, matrix and factor
my_list <- list(my_vector, my_matrix, my_factor)

# Construct my_super_list with the four data structures above
my_super_list = list(my_vector,my_matrix,my_factor,my_list)

# Display structure of my_super_list
str(my_super_list)

3.Create a named list

# Numeric vector: 1 up to 10
my_vector <- 1:10

# Numeric matrix: 1 up to 9
my_matrix <- matrix(1:9, ncol = 3)

# Factor of sizes
my_factor <- factor(c("M","S","L","L","M"), ordered = TRUE, levels = c("S","M","L"))

# Construct my_list with these different elements
my_list <- list(my_vector, my_matrix, my_factor)

# Print my_list to the console
names(my_list) = c("vec","mat","fac")
```

```
my_list

4.Create a named list (2)

# Create actors and reviews
actors_vector <- c("Jack Nicholson","Shelley Duvall","Danny Lloyd","Scatman
Crothers","Barry Nelson")
reviews_factor <- factor(c("Good", "OK", "Good", "Perfect", "Bad", "Perfect",
"Good"),
                  ordered = TRUE, levels = c("Bad", "OK", "Good", "Perfect"))

# Create shining_list
# Create the list 'shining_list'
shining_list <- list(title = "The Shining", actors = actors_vector, reviews =
reviews_factor)

5. List your skills
# Create the list lst

lst <- list(top[c(5)],prop[,4])
lst
# Create the list skills
skills <- list(topics=top,context=cont,properties=prop,list_info=lst)

# Display the structure of skills
str(skills)
--------------------------------------------------Lab
11-----------------------------------------------

1. Selecting elements from a list
# shining_list is already defined in the workspace

# Actors from shining_list: act
act = shining_list$actors

# List containing title and reviews from shining_list: sublist
sublist = list(shining_list$title,shining_list$reviews)

# Display structure of sublist
str(sublist)

2.Chaining your selections
# shining_list is already defined in the workspace

# Select the last actor: last_actor
last_actor <- shining_list$actors[length(shining_list$actors)]

# Select the second review: second_review
```

```
second_review <- shining_list$reviews[2]

3.Vector Subsetting vs. List Subsetting
A and C (2)

4. Extending a list (1)

# shining_list is already defined in the workspace

# Add the release year to shining_list
shining_list$year <- 1980

# Add the director to shining_list
shining_list$director <- "Stanley Kubrick"

# Inspect the structure of shining_list
str(shining_list)

5. Extending a list (2)
# shining_list is already defined in the workspace

# Add both the year and director to shining_list: shining_list_ext
shining_list_ext = c(shining_list,list(year=1980,director="Stanley Kubrick"))

# Have a look at the structure of shining_list_ext
str(shining_list_ext)

6. List your skills (2)

skills$list_info

# Create the list key_skills
key_skills <- list(skills$topics[2], skills$context[2], skills$list_info[[2]][4])

--------------------------------------------------Lab
12----------------------------------------------

1. Have a look at your data set

# Print the first observations of mtcars

head(mtcars)
# Print the last observations of mtcars
tail(mtcars)

# Print the dimensions of mtcars
dim(mtcars)
```

2. Have a look at the structure

```
# Investigate the structure of the mtcars data set
str(mtcars)
```

3. Creating a data frame

```
# Definition of vectors
planets <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
"Neptune")
type <- c("Terrestrial planet", "Terrestrial planet", "Terrestrial planet",
          "Terrestrial planet", "Gas giant", "Gas giant", "Gas giant", "Gas giant")
diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE)

# Create a data frame: planets_df
planets_df = data.frame(planets,type,diameter,rotation,rings)


# Display the structure of planets_df
str(planets_df)
```

4. Creating a data frame (2)

```
# Definition of vectors
planets <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
"Neptune")
type <- c("Terrestrial planet", "Terrestrial planet", "Terrestrial planet",
          "Terrestrial planet", "Gas giant", "Gas giant", "Gas giant", "Gas giant")
diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE)

# Encode type as a factor: type_factor
type_factor <- factor(type)

# Construct planets_df: strings are not converted to factors!
planets_df <- data.frame(planets, type_factor, diameter, rotation, rings,
stringsAsFactors = FALSE)

# Display the structure of planets_df
str(planets_df)
```

5. Rename the data frame columns

```
# Construct improved planets_df
planets <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
```

```
"Neptune")
type <- c("Terrestrial planet", "Terrestrial planet", "Terrestrial planet",
          "Terrestrial planet", "Gas giant", "Gas giant", "Gas giant", "Gas giant")
diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE)
type_factor <- factor(type)
planets_df <- data.frame(planets, type_factor, diameter, rotation, rings,
stringsAsFactors = FALSE)

# Improve the names of planets_df
names(planets_df) <- c("name","type","diameter","rotation","has_rings")
planets_df

6.Rule the world!

continents_factor <- factor(continents)
countries_df <- data.frame(name = countries, cont = continents_factor, GDP = gdp,
HDI = hdi, has_president = president, stringsAsFactors=FALSE)
str(countries_df)

--------------------------------------------------Lab
13-----------------------------------------------
1. Selection of data frame elements

# planets_df is pre-loaded

# The type of Mars: mars_type
mars_type <- planets_df[4, 2]

# Entire rotation column: rotation
rotation <- planets_df[ ,4]

# First three planets: closest_planets_df
closest_planets_df <- planets_df[1:3, ]

# Last three planets: furthest_planets_df
furthest_planets_df <- planets_df[6:8, ]


2. Selection of data frame elements (2)

# planets_df is pre-loaded
planets_df
# Diameter and rotation for Earth: earth_data
earth_data <- planets_df[3,3:4]
earth_data
# Diameter for the last six rows: furthest_planets_diameter
```

```
furthest_planets_diameter <- planets_df[3:8,"diameter"]
# Print furthest_planets_diameter
furthest_planets_diameter
```

3.Only planets with rings

```
# planets_df is pre-loaded in your workspace

# Create rings_vector
rings_vector= planets_df$has_rings

# Print rings_vector
rings_vector
```

4.Only planets with rings (2)

```
# planets_df pre-loaded in your workspace

# Create rings_vector
rings_vector <- planets_df$has_rings

# Select the information on planets with rings: planets_with_rings_df
planets_with_rings_df <- planets_df[rings_vector,]

# Print planets_with_rings_df
planets_with_rings_df
```

5.Only planets with rings but shorter

```
# planets_df is pre-loaded in your workspace

# Planets that are smaller than planet Earth: small_planets_df
small_planets_df <- planets_df[planets_df$diameter < 1, ]


# Planets that rotate faster than planet Earth: fast_planets_df
slow_planets_df <- planets_df[abs(planets_df$rotation) > 1, ]
```

6. Add variable/column

```
 # planets_df is already pre-loaded in your workspace

# Definition of moons and masses
moons <- c(0, 0, 1, 2, 67, 62, 27, 14)
masses <- c(0.06, 0.82, 1.00, 0.11, 317.8, 95.2, 14.6, 17.2)

# Add moons to planets_df under the name "moon"
planets_df$moon <- moons
```

```
# Add masses to planets_df under the name "mass"
planets_df$mass <- masses
```

7. Add observations

```
# planets_df is pre-loaded (without the columns moon and mass)

# Name pluto correctly
pluto <- data.frame(name = "Pluto", type = "Terrestrial planet", diameter = 0.18,
rotation = -6.38, has_rings = FALSE)

# Bind planets_df and pluto together: planets_df_ext
planets_df_ext <- rbind(planets_df, pluto)

# Print out planets_df_ext
planets_df_ext
```

8. Sorting

```
# Just play around with the order function in the console to see how it works!
a <- c(100,9,101)
order(a)
```

9.Sorting your data frame

```
# planets_df is pre-loaded in your workspace

# Create a desired ordering for planets_df: positions
positions <- order(planets_df$diameter,decreasing=TRUE)

# Create a new, ordered data frame: largest_first_df
largest_first_df <- planets_df[positions, ]

# Print largest_first_df
largest_first_df
```

10. Rule the world: part II

```
# Remove economic variables and add population.
countries_df_dem <- countries_df[ , c(1, 2, 5)]
countries_df_dem$population <- population

# Add brazil
names(brazil) <- c("name", "continent", "has_president", "population")
```

```
countries_df2 <- rbind(countries_df_dem,brazil)

# Sort by population
countries_df2[order(countries_df2$population,decreasing=TRUE), ]

----------------------------------------------------Lab
14-----------------------------------------------
1. Plotting factors

# movies is already pre-loaded

# Display the structure of movies
str(movies)

# Plot the genre variable of movies
plot(movies$genre)

# Plot the genre variable against the rating variable
plot(movies$genre,movies$rating)

2. Plotting numerics

# movies is already pre-loaded

# Plot the runtime variable of movies

 plot(movies$runtime)
# Plot rating (x) against runtime (y)
plot(movies$rating,movies$runtime)

3. Create a Histogram

# movies is already pre-loaded

# Create a histogram for rating

hist(movies$rating)
# Create a histogram for rating, with 20 bins
hist(movies$rating,breaks=20)

4. Other graphics functions

# movies is already pre-loaded

# Create a boxplot of the runtime variable
boxplot(movies$runtime)

# Subset the dateframe and plot it entirely
```

```
plot(movies[,c("rating", "votes", "runtime")])

# Create a pie chart of the table of counts of the genres
pie(table(movies$genre))

5. How does your salary compare?

salaries_educ <- salaries[salaries$degree == 3, ]
hist(salaries_educ$salary, breaks = 10)
```

----------Lab 15----------

```
1. Title and axis labels

# movies is pre-loaded in your workspace

# Create a customized plot
# Create a customized plot
plot(movies$votes, movies$runtime,
     main = "Votes versus Runtime",
     xlab = "Number of votes [-]",
     ylab = "Runtime [s]",
     sub = "No clear correlation")

2. Colors and shapes

# movies is pre-loaded in your workspace

# Customize the plot further
plot(movies$votes, movies$runtime,
     main = "Votes versus Runtime",
     xlab = "Number of votes [-]",
     ylab = "Runtime [s]",
     sub = "No clear correlation",
     pch = 9,
     col = "#dd2d2d",
     col.main = 604)

3. Customize Everything!

# movies is pre-loaded in your workspace

# Customize the plot further
plot(movies$votes, movies$year,
     main = "Are recent movies voted more on?",
     xlab = "Number of votes [-]",
     ylab = "Year [-]",
     col = "orange",
```

```
     cex.axis = 0.8,
     pch = 19)


4. Customizing Histograms

# Build a customized histogram
hist(movies$runtime,
     breaks = 20,
     xlim = c(90, 220),
     main = "Distribution of Runtime",
     xlab = "Runtime [-]",
     col = "cyan",
     border = "red")

5. Does work experience influence your salary?

# Add the exp vector as a column experience to salaries
salaries$experience <- exp

# Filter salaries: only keep degree == 3: salaries_educ
salaries_educ <- salaries[salaries$degree == 3, ]

# Create plot with many customizations


plot(salaries_educ$experience, salaries_educ$salary,
     main = "Does experience matter?" ,
     xlab = "Work experience",
     ylab = "Salary",
     col = "blue",
     col.main = "red",
     cex.axis = 1.2)
----------------------------------------------------Lab
16-------------------------------------------------
 1. Multiple plots with par()

 # movies is pre-loaded in your workspace

# List all the graphical parameters
par()

# Specify the mfrow parameter
par(mfrow = c(2,1))

# Build two plots
plot(movies$votes, movies$rating)
hist(movies$votes)
```

2. Complex layouts!

```
# movies is pre-loaded in your workspace

# Build the grid matrix
grid <- matrix(c(1, 2, 3, 3), nrow = 2)

# Specify the layout
layout(grid)

# Build three plots
plot(movies$rating, movies$runtime)
plot(movies$votes, movies$runtime)
boxplot(movies$runtime)
```

3. Complex layouts with customized plots

```
# movies is pre-loaded in your workspace

# Build the grid matrix
grid <- matrix(c(1, 2, 3, 3), nrow = 2)

# Specify the layout
layout(grid)

# Customize the three plots
plot(movies$rating, movies$runtime, xlab = "Rating", ylab = "Runtime", pch = 4)
plot(movies$votes, movies$runtime, xlab = "Number of Votes", ylab = "Runtime", col =
"blue")
boxplot(movies$runtime, main = "Boxplot of Runtime", border = "darkgray")
```

4. Plot a linear regression

```
# movies is pre-loaded in your workspace

# Fit a linear regression: movies_lm
movies_lm <- lm(movies$rating ~ movies$votes)

# Build a scatterplot: rating versus votes
plot(movies$votes, movies$rating)

# Add straight line to scatterplot
abline(coef(movies_lm))
```

5. Customize your linear regression plot

R-Solution

```
# movies is pre-loaded in your workspace

# Fit a linear regression (don't change)
movies_lm <- lm(movies$rating ~ movies$votes)

# Customize scatterplot
plot(movies$votes, movies$rating,
     main = "Analysis of IMDb data",
     xlab = "Number of Votes",
     ylab = "Rating",
     col = "darkorange",
     pch = 15,
     cex = 0.7)

# Customize straight line
abline(coef(movies_lm), lwd = 2, col = "red")

# Add text
xco <- 7e5
yco <- 7
text(xco, yco, label = "More votes? Higher rating!")
```

6. Multiple plots with different layers

B