Solutions to Practice Problems

### 1 R Programming Basics

##### 1.1 Practice: BMI calculation

\*\* Note: feel free to use fake values for the following problem.  
Body Mass Index (BMI) can be used to screen for weight categories that may lead to health problems. The formula is BMI = weight(kg)/height(m)^2. BMI ranges are underweight (under 18.5 kg/m2), normal weight (18.5 to 25), overweight (25 to 30), and obese (over 30).  
a) Define a variable and assign the value of your weight to it. (Note: kg = lb/2.205)  
b) Define a variable and assign the value of your height to it. (Note: m = inches/39.37)  
c) Calculate your BMI. Assign the value to a variable.  
d) Print it with 3 significant digits.  
e) In 2013, a new formula for BMI that accounts for the distortions of the traditional BMI formula for shorter and taller individuals was proposed by Nick Trefethen, Professor of numerical analysis at Oxford University. (source: [link](https://people.maths.ox.ac.uk/trefethen/bmi.html)). The new formula is BMI = 1.3\*weight(kg)/height(m)^2.5. What is your BMI now?  
f) Print the statement "My BMI is \_\_\_." to the console.  
g) Use a relational operator to check whether your BMI is not underweight? h) Use relational operators AND logical operators to check whether your BMI is in the “normal” range.

Solution:

# 1.1 BMI practice solution  
# a)  
my\_weight <- 65 #kg  
# b)  
my\_height <- 1.80 #m  
# c)   
my\_BMI <- my\_weight/my\_height^2  
my\_BMI

## [1] 20.06173

# d)   
print(my\_BMI, digits = 3)

## [1] 20.1

# e)  
my\_new\_BMI <- 1.3\*my\_weight/my\_height^2.5  
# f)  
statement <- sprintf("My BMI is %s.", round(my\_BMI,digits = 1)) #round to 1 decimal place  
print(statement)

## [1] "My BMI is 20.1."

# g)  
my\_BMI >= 18.5 # am i not underweight? T/F

## [1] TRUE

# h)  
# usually we'd do 18.5 <= my\_BMI <= 25 but you can't have multiple relational operators on the same line of code without Boolean ones  
my\_BMI >= 18.5 & my\_BMI <= 25

## [1] TRUE

### 2 Data structures - Vectors and dataframes

##### Practice 2.1 - My family

You are gathering information about your family members (alternatively, your friends or coworkers). a) Make a vector of their names.  
b) Who is the first person you wrote down? (i.e. Get the first element)  
c) Make a vector of their ages (in same order as part a)).  
d) Make a vector if they’re a kid or not (TRUE/FALSE).  
e) Make a data frame of your family members with column names: Name, Age, Is\_Kid.  
f) Sort their names by alphabetical order. The output should be saved to a data frame variable. (Look this up if you don’t know how to!)  
g) Subset your data frame so only rows of the members that are children shows (do not save as variable).  
h) Subset your data frame so only rows of the members that are older than 20 years old show (do not save as variable).  
i) Add 1 to the ages of all your members in one line of code.  
j) Remove the Is\_Kid column.

Solution:

# a) Make a vector using c()   
names <- c("Tinky Winky", "Dipsy", "Laa Laa", "Po")  
# b) Use positive indexing  
names[1]

## [1] "Tinky Winky"

# c) Make a vector using c()  
ages <- c(21, 10, 10, 2)  
# d) Make a logical vector using c()  
is\_kid <- c(F,T,T,T)  
# e) Make a data frame using data.frame()  
family <- data.frame(Name = names, Age = ages, Is\_Kid = is\_kid)  
family

## Name Age Is\_Kid  
## 1 Tinky Winky 21 FALSE  
## 2 Dipsy 10 TRUE  
## 3 Laa Laa 10 TRUE  
## 4 Po 2 TRUE

# f) https://www.r-bloggers.com/r-sorting-a-data-frame-by-the-contents-of-a-column/  
sorted\_family <- family[order(family$Name),]   
sorted\_family

## Name Age Is\_Kid  
## 2 Dipsy 10 TRUE  
## 3 Laa Laa 10 TRUE  
## 4 Po 2 TRUE  
## 1 Tinky Winky 21 FALSE

# g) Logical vector or subset()  
sorted\_family[sorted\_family$Is\_Kid,]

## Name Age Is\_Kid  
## 2 Dipsy 10 TRUE  
## 3 Laa Laa 10 TRUE  
## 4 Po 2 TRUE

subset(sorted\_family, subset = Is\_Kid)

## Name Age Is\_Kid  
## 2 Dipsy 10 TRUE  
## 3 Laa Laa 10 TRUE  
## 4 Po 2 TRUE

# h) subset using logical vector  
sorted\_family[sorted\_family$Age > 20,]

## Name Age Is\_Kid  
## 1 Tinky Winky 21 FALSE

subset(sorted\_family, subset = Age > 20)

## Name Age Is\_Kid  
## 1 Tinky Winky 21 FALSE

# i) modify the Age column only  
sorted\_family$Age <- sorted\_family$Age + 1  
# j) Using negative indexing or assigning column to NULL  
sorted\_family$Is\_Kid <- NULL  
# Or sorted\_family <- sorted\_family[, - which (colnames(sorted\_family) == "Is\_Kid")]

### 3 Import, Export and base R Graphs

### Practice 3.1 American Women

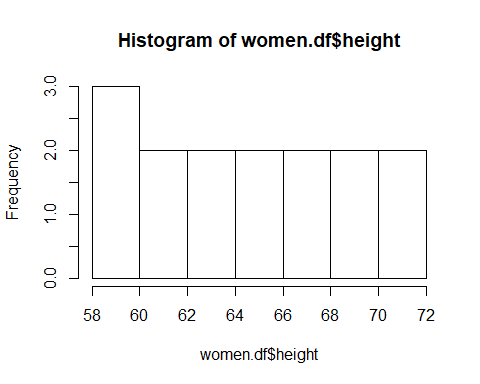
The “women” data set in R gives the average heights and weights for American women aged 30–39.  
a) Print the first 15 rows to the console. (Hint: use the “n” argument in head() function)  
b) Create a folder called “Data Sets” in your current working directory.  
c) Write the women data frame as a csv file to the Data Sets folder (exclude row names).  
d) Read this file back into R and assign it to a variable called “women.df”.  
e) Plot a histogram of the heights column.  
f) Find the mean and standard deviation of heights (Recall: vectors tutorial)  
g) print the variables from f) in a statement ""The mean and standard deviation of the heights is \_\_ and \_\_" (Hint: use sprintf() or paste() )  
h) Plot a scatter plot, where x = height and y = weight. Relabel x and y axes to “Height (in)” and “Weight (lbs)” respectively.  
i) Save f) to a jpeg file.

Solution

data("women")  
# a) Print using head()  
head(women, n = 15)

## height weight  
## 1 58 115  
## 2 59 117  
## 3 60 120  
## 4 61 123  
## 5 62 126  
## 6 63 129  
## 7 64 132  
## 8 65 135  
## 9 66 139  
## 10 67 142  
## 11 68 146  
## 12 69 150  
## 13 70 154  
## 14 71 159  
## 15 72 164

# b) Create a folder using dir.create()  
dir.create("Data Sets")  
# c) Write to csv using write.csv()  
write.csv(x = women, file = "Data sets/women.csv", row.names = F)  
# d) Read using read.csv()  
women.df <- read.csv(file = "Data sets/women.csv")  
# e) Plot histogram using hist()  
hist(women.df$height)



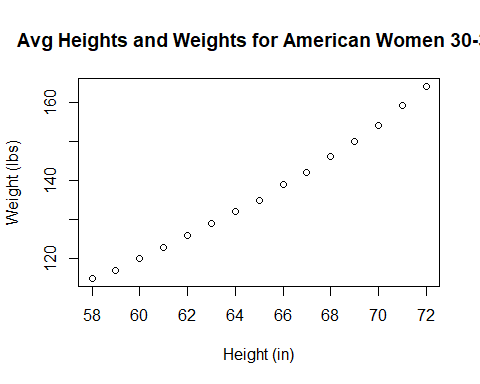
# f) Find mean using mean() and standard deviation using sd()  
mean.hts <- mean(women.df$height)  
sd.hts <- sd(women.df$weight)  
# g) Print  
sprintf("The mean and standard deviation of the heights is %s and %s", mean.hts, sd.hts)

## [1] "The mean and standard deviation of the heights is 65 and 15.4986942614378"

paste("The mean of the heights is ", mean.hts, " and ", sd.hts, sep = "")

## [1] "The mean of the heights is 65 and 15.4986942614378"

# h) Plot using plot()  
plot(x=women.df$height, y=women.df$weight, xlab = "Height (in)", ylab = "Weight (lbs)", main = "Avg Heights and Weights for American Women 30-39")



# i) Save as jpeg  
jpeg(filename = "heights\_vs\_weights.jpeg")  
plot(x=women.df$height, y=women.df$weight, xlab = "Height (in)", ylab = "Weight (lbs)", main = "Avg Heights and Weights for American Women 30-39")  
dev.off()

## png   
## 2

### Statistical tests

### Practice 5.1 American women

The “women” data set in R gives the average heights and weights for American women aged 30–39.  
\* Significance level is p<0.05.  
a) Print the first 10 rows to the console. (Hint: use the “n” argument in head() function)  
b) What is the data type of the height column? (Hint: use str() or class())  
c) Are the height and weight variables normally distributed? (Hint: use Shapiro’s test for each)  
d) Should we use t-test or wilcoxin test on this data? Why?  
e) Compare the heights to an estimated mean of 66.2 using a one-sample t-test. Is there a significant difference in means?  
f) Compare the first 6 weights recorded (ie. 1 to 6) to the next 6 (ie. 7 to 12) using a t-test. Is there a significant difference in means?

Solution

# Load data (you can still use it without this step)  
data("women")  
# a) Print using head()  
head(women, n = 10)

## height weight  
## 1 58 115  
## 2 59 117  
## 3 60 120  
## 4 61 123  
## 5 62 126  
## 6 63 129  
## 7 64 132  
## 8 65 135  
## 9 66 139  
## 10 67 142

# b) Use class() to get data type   
class(women$height) # ANSWER: numeric

## [1] "numeric"

# c) use Shapiro's test to test for normality. If p > 0.05, normally distributed  
shapiro.test(women$height) # p-value = 0.7545

##   
## Shapiro-Wilk normality test  
##   
## data: women$height  
## W = 0.96359, p-value = 0.7545

shapiro.test(women$weight) # p-value = 0.6986

##   
## Shapiro-Wilk normality test  
##   
## data: women$weight  
## W = 0.96036, p-value = 0.6986

# ANSWER: Yes, since p > 0.05 for both variables, the data is normally distributed  
  
# d) ANSWER: We could use t-tests because parametric statistical tests is used on normally distributed data.  
  
# e) use t.test(), where mu = 66.2  
t.test(women$height, mu = 66.2) # p-value = 0.3163

##   
## One Sample t-test  
##   
## data: women$height  
## t = -1.0392, df = 14, p-value = 0.3163  
## alternative hypothesis: true mean is not equal to 66.2  
## 95 percent confidence interval:  
## 62.52341 67.47659  
## sample estimates:  
## mean of x   
## 65

# ANSWER: No, since p > 0.05, there is no significant difference, so the mean of heights is close to 66.2.  
  
# f) use t.test() with x,y (two samples)  
t.test(x = women$weight[1:6], y = women$weight[7:12]) # p-value 0.0003556

##   
## Welch Two Sample t-test  
##   
## data: women$weight[1:6] and women$weight[7:12]  
## t = -5.4053, df = 9.5115, p-value = 0.0003556  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -26.88688 -11.11312  
## sample estimates:  
## mean of x mean of y   
## 121.6667 140.6667

# ANSWER: Yes, since p < 0.05, there is a significant difference

### 7 Heatmaps

### Practice 7.1 Motor trend car road tests (mtcars)

Fuel consumption and 10 additional aspects (variables) of automobile design tested in 1974 are given in an in-built R dataset called “mtcars”. a) Look at the first rows of the data frame (using head()) b) Look at the structure, are all the data numeric? (use str()) c) Convert the data frame to a data matrix into a new variable. d) Add 1 to all values and apply log2 transformation. e) Make a heatmap (using the pheatmap package). i) Scale by column, ii) use clustering method “ward”, iii) set cell width to 10, iv) pick 9 colours from RColorBrewer color palette “RdPu” and make a palette with 250 colors for the heatmap, and v) set the title to “my heatmap” vi) save to a jpeg file called “mtcars\_heatmap”

Solution

# a) Use head()   
head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

# b) Use str() to see the structure  
str(mtcars) # ANSWER: Yes, all variables are numeric

## 'data.frame': 32 obs. of 11 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

# c) convert to matrix using as.matrix  
mtcars\_mat <- as.matrix(mtcars)  
# d) add 1 and log2  
mtcars\_mat <- mtcars\_mat + 1  
mtcars\_mat <- log2(mtcars\_mat)  
# equivalent: mtcars\_mat <- log2(mtcars\_mat + 1)  
# e) plot heatmap  
library(pheatmap)  
library(RColorBrewer)  
pheatmap(mtcars\_mat, #e  
 scale = "column", #i  
 clustering\_method = "ward.D", #ii  
 cellwidth = 10, #iii  
 col = colorRampPalette(brewer.pal(9, "RdPu")) (256), #iv  
 main = "my\_heatmap", #v  
 filename = "mtcars\_heatmap.jpeg" #vi  
 )