SCM7047 assignment 1 - R programming

Steven Mitchell 401758852

2023-03-09

Contents

1 Prerequistes - Install bookdown, rmarkdown, and tidyverse packages

install.packages(c("bookdown", "rmarkdown", "tidyverse")) # check directory getwd() setwd("C:/Users/swmit/OneDrive - Queen's University Belfast/SCM7047 Scientific Programming & Statistical Computing/Assignment 1/SCM7047-R-assignment") install.packages('tinytex') tinytex::install_tinytex()

Question 1

Which parts of the following code do you expect to produce an error? Provide an explanation.
4 marks

```
# data1 <- matrix(51:190, nrow = 20)
# data2 <- Matrix(c(1:10, 20:11), byrow =T)
# data3 = data1+data2


  # data2 variable will produce error in Matrix() as the called function has an
#   uppercase M when it should be in lowercase. Therefore, the creation of data2
#   variable will result in an error. Corrected code shown below.
```

```
data1 <- matrix(51:190, nrow = 20)
data2 <- matrix(c(1:10, 20:11), byrow = TRUE)

  # data3 will produce an error as the matrix dimensions are incompatible. data1
#   20 x 7 is not directly compatible with data2 2 x 10. Matrices can only be
#   added or subtracted when they have the same dimensions, as every element
#   requires a corresponding element.
```

Question 2

Debug the following code snippets to assign the given values to objects (one object for each line). Comment out the incorrect code and leave a copy of the corrected code beneath. For each snippet provide a short explanation as a code comment.
8 marks

```
# a) variable.a <- matrix(c(rbind(1:2,), c(1, 2:7, 2, 5)), nrow = 1)

    # Creating matrix but comma is on the wrong side of the in the rbind()
#     function, which will result in an error.
    variable.a <- matrix(c(rbind(1:2), c(1, 2:7, 2, 5)), nrow = 1)

# b)study_string = as.Character("Assignment1")

    # as.Character() function requires lowercase as.character() to convert
#     string "Assignment1" to a character vector.
    study_string <- as.character("Assignment1")

#c).5z <- "string manipulation."

    # A variable name must start with a letter, if it starts with (.) it cannot
#     be followed by a digit. Also, variable names cannot begin with a number.
#     Therefore, the   variable name .5z is not valid and will result in an error,
#     a valid name would be .z5
    .z5 <- "string manipulation."

#d) Varz <= c(TRUE, FALSE, 56, "TRUE", "TRUE", TRUE, 88, 567.5)

    # Assignment operator should be <- not <= which would be less than or equal to.
    Varz <- c(TRUE, FALSE, 56, "TRUE", "TRUE", TRUE, 88, "567.5")
```

Question 3

In the given code, a user enters the following values: `var1` = 4 `var2` = 25 `var3` = 1050 expecting the return value to be 31. However, it does not return anything. Modify the code (not the input values) so that it returns the value as expected by the user. Briefly explain your changes using code comments. *12 marks*

```
#   ~BAD CODE~       var1 <- as.integer(readline("Enter a value:"))
#                    var2 <- as.integer(readline("Enter a value:"))
#                    var3 <- as.integer(readline("Enter a value:"))
#                    func.1 <- function(var1) {
```

```
#                    func.2 <- function(var2) {
#                    var2 + var3
#                    }
#                    var1 + func.2(var2)
#                    var3 = 2
#                    }
#                    func.1(var1)

#   ~ANSWER~

var1 <- as.integer(readline("Enter a value:"))
```

```
## Enter a value:
```

```
var2 <- as.integer(readline("Enter a value:"))
```

```
## Enter a value:
```

```
var3 <- as.integer(readline("Enter a value:"))
```

```
## Enter a value:
```

```
# Move re-assignment of var3 to before the definition of func.2
var3 <- 2

func.1 <- function(var1) {
  func.2 <- function(var2) {
    var2 + var3
  }
  # Return result
  return(var1 + func.2(var2))
}
# Print to console
print(func.1(var1))
```

```
## [1] NA
```

Question 4

Solutions for this question should not rely on `tidyverse` functions.

Write a function-based program in R which performs the following tasks based on data it reads from the CSV file `Q4-CervicalCancer.csv`. The program should be called using a *single* line of code in which the user can specify *three* arguments: `var_x` and `var_y` each representing a different variable in the dataset and `output_type` indicating the type of file in which to save summary statistics.

a) Produce a data.frame containing summary statistics for `var_x` and `var_y`: mean, standard deviation, range, and number of missing values. *8 marks*

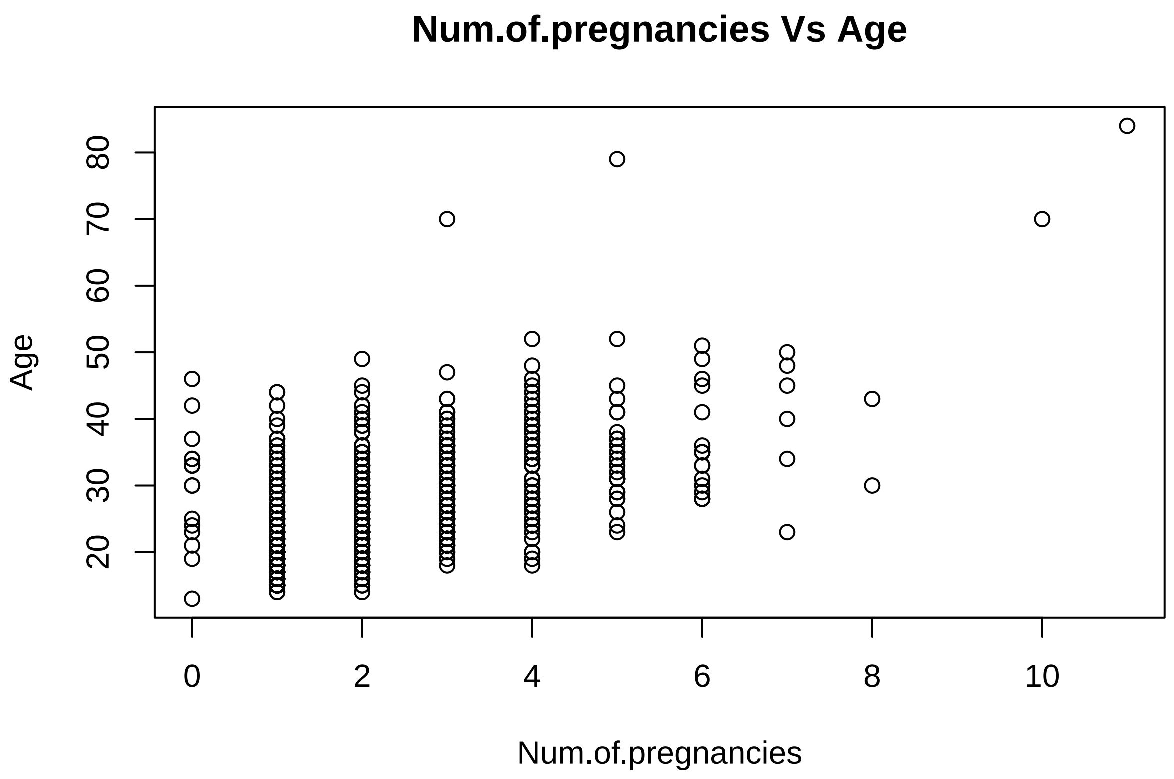
b) Create a scatter plot between `var_x` and `var_y`. The plot should have X and Y axis labels and a plot title based on the selected attributes (e.g. – Biopsy Vs Age). *12 marks*

c) Your program should give the user an option to save the summary results as either a .csv or a .txt file, named `Q4-summary-statistics`. *6 marks*

About the sample file – This is a cervical cancer dataset from Hospital Universitario de Caracas, Venezuela. To learn more about the dataset, refer to the UCI Repository.

```r
analysis_function <- function(var_x, var_y, output_type) {

  # Read in data
  data <- read.csv("Q4-CervicalCancer.csv", header = TRUE, stringsAsFactors = FALSE)

  # Check if the variables are numeric, if not, convert to numeric
  if (!is.numeric(data[[var_x]])) {
    data[[var_x]] <- as.numeric(data[[var_x]])
  }
  if (!is.numeric(data[[var_y]])) {
    data[[var_y]] <- as.numeric(data[[var_y]])
  }

  # Get the summary statistics for var_x and var_y
  summary_stats <- data.frame(
    variable = c(var_x, var_y),
    mean = c(mean(data[, var_x], na.rm = TRUE), mean(data[, var_y], na.rm = TRUE)),
    sd = c(sd(data[, var_x], na.rm = TRUE), sd(data[, var_y], na.rm = TRUE)),
    range = c(paste0(min(data[, var_x], na.rm = TRUE), "-", max(data[, var_x],
                                                            na.rm = TRUE)),
              paste0(min(data[, var_y], na.rm = TRUE), "-", max(data[, var_y],
                                                            na.rm = TRUE))),
    missing = c(sum(is.na(data[, var_x])), sum(is.na(data[, var_y])))
  )

  # Print summary statistics
  print(summary_stats)

  # Create scatter plot of var_x vs. var_y
  plot(data[, var_x], data[, var_y], main = paste(var_x, "Vs", var_y),
       xlab = var_x, ylab = var_y)

  # Save summary statistics
  if (output_type == "csv") {
    write.csv(summary_stats, file = "Q4-summary-statistics.csv", row.names = FALSE)
  } else if (output_type == "txt") {
    write.table(summary_stats, file = "Q4-summary-statistics.txt", sep = "\t",
                row.names = FALSE)
  } else {
    cat("Error: Output type must be 'csv' or 'txt'")
  }
}

analysis_function("Num.of.pregnancies", "Age", "csv")
```

```
## Warning in analysis_function("Num.of.pregnancies", "Age", "csv"): NAs introduced
## by coercion

##             variable      mean       sd range missing
## 1 Num.of.pregnancies  2.275561 1.447414  0-11      56
## 2                Age 26.820513 8.497948 13-84       0
```



Question 5

Solutions using either base R or `tidyverse` functions will be accepted for this question.

As a member of a Data Science team in a Bio-analytics company, you have been assigned the following tasks:

a) Develop a function-based R program which reads a dataset specified by the user and performs the following operations. The program should be called using a *single* line of code in which the user can specify an appropriate number of arguments.

i. Provide the number of dimensions and the names of variables for a user-specified dataset *6 marks*

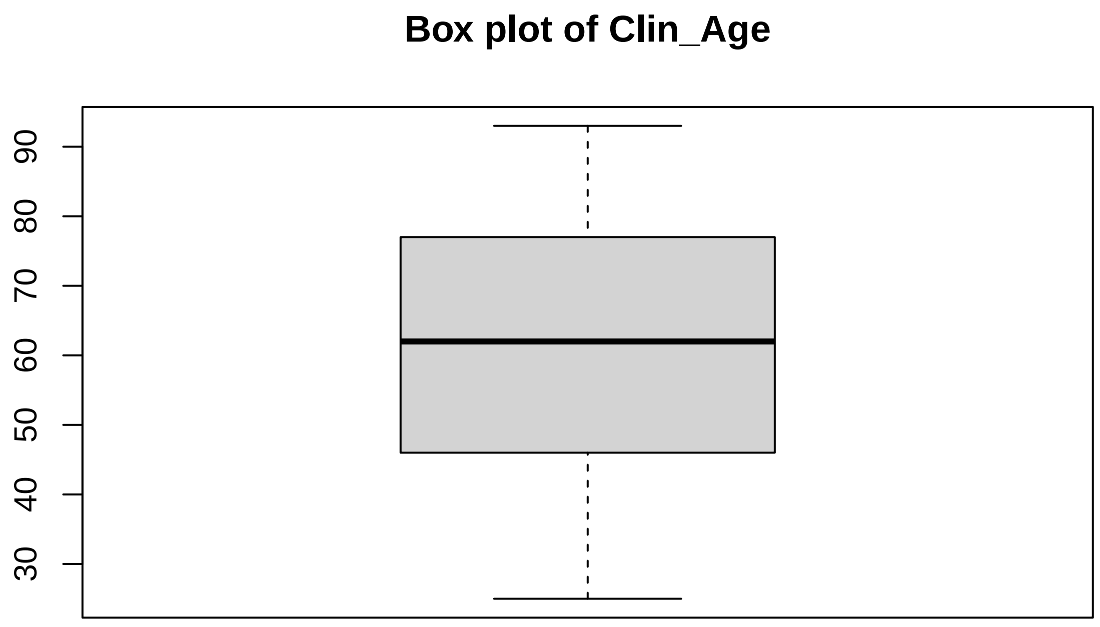
ii. Replace any missing values with NA. *2 marks*

iii. Outputs a box plot of the distribution for at least one user-specified continuous variable. *6 marks*

Two files are provided, `Q5-File1.csv` contains clinical data and `Q5-File2.csv` contains protein expression data. The program should work for both of these files (called separately for each).

```r
# Define function
boxplot_function <- function(file_name, var_names) {

  # Read dataset
  data <- read.csv(file_name, header = TRUE)

  # Get the number of dimensions and variable names
  n_dims <- dim(data)[2]
  names <- colnames(data)

  # Print the number of dimensions and variable names
  cat("Number of dimensions:", n_dims, "\n")
  cat("Variable names:", paste(names, collapse = ", "), "\n")

  # Replace missing values with NA
  data[is.na(data)] <- NA

  # Output a box plot for the first specified continuous variable
  var <- var_names[1]
  if (is.numeric(data[[var]])) {
    boxplot(data[[var]], main = paste("Box plot of", var))
  } else {
    cat(paste("Variable", var, "is not continuous"))
  }
}

boxplot_function ("Q5-File1.csv", c("Clin_Age"))
```

```
## Number of dimensions: 10
## Variable names: PatientID, Clin_Age, Clin_gender, Clin_Stage, Clin_Histology, Clin_BiomarkerAPreSurg
```



b) Develop a function-based R program, which matches patient IDs (variable `PatientID`) from the first dataset with reference IDs (`Reference_ID`) from the second dataset. `Reference ID` in the second file is a combination of patient ID with sample ID (hint – try strsplit() or tidyverse separate()). For each patient, produce a data.frame listing the arithmetic mean of expression values for each protein. Thereafter, based on a user specified threshold value, produce a data.frame listing patients with a mean protein expression value for any protein above this threshold. The program should be called using a *single* line of code in which the user can specify the protein threshold as an argument.

36 marks

```r
library(tidyverse)

# Define function
match_patients <- function(threshold) {

  # Load data from Q5-File1.csv and Q5-File2.csv
  data1 <- read.csv("Q5-File1.csv")
  data2 <- read.csv("Q5-File2.csv")

  # Extract patient IDs from Reference IDs in data2
  data2 <- separate(data2, Reference_ID, into=c("PatientID", "SampleID"), sep="_")

  # Change to integer data
  data1$PatientID <- as.numeric(data1$PatientID)
  data2$PatientID <- as.numeric(data2$PatientID)
  # Convert any non-numeric data to NA
  data2[data2 == "?"] <- NA
  data2[data2 == ""] <- NA
  data1[data1 == "?"] <- NA
  data1[data1 == ""] <- NA

  # Convert Protein columns to numeric
  data2[, 6:15] <- apply(data2[, 6:15], 2, function(x) as.numeric(as.character(x)))

  # Compute the mean expression of each protein for each patient
  mean_protein_by_patient <- data.frame(aggregate
                                        (data2[,6:15], by = list
                                          (PatientID = data2$PatientID),
                                          FUN = function(x) mean(x, na.rm = TRUE)))

  # Merge data1 and mean_protein_by_patient based on PatientID
  merged_data <- merge(mean_protein_by_patient, data1, by = "PatientID")

  # Filter patients by mean protein values greater than threshold
  filtered_data <- subset(merged_data, rowMeans(merged_data[, 2:11],
                                              na.rm = TRUE) > threshold)

  # Return filtered data.frame
  return(filtered_data)
}

# Call match_patients function with threshold of 1500
match_patients(1500)
```

```
##     PatientID Protein.1 Protein.2 Protein.3 Protein.4 Protein.5 Protein.6
## 10         10  3396.000  3201.667  1218.333  793.6667  312.1667  581.0000
## 17         17  2755.000  3805.667  1831.000 1338.0000  504.0000  947.3333
## 25         25  5589.667  7462.500  3031.333 2011.6667  787.3333 1527.3333
## 77         77  5749.500  7388.500  3169.000 2140.5000  920.6667 1705.3333
## 97         97  4250.333  5582.667  2286.167 1575.6667  657.8333 1159.8333
## 99         99  2128.000  2778.000  1520.667 1019.3333  354.6667  740.0000
## 114       114  3680.833  4632.333  1018.167  630.8333  164.3333  308.8333
## 141       141  2342.000  3168.667  1426.500  937.1667  409.3333  791.5000
##     Protein.7 Protein.8 Protein.9 Protein.10 Clin_Age Clin_gender Clin_Stage
## 10  1012.1667  1691.333  2026.667   1647.167       68           M          2
## 17  1123.3333  1832.000  3138.333   2578.333       48           M          2
## 25  2226.3333  3605.667  5177.667   4275.000       38           F          1
## 77  2370.3333  3797.667  5270.500   4345.333       63           F          2
## 97  1706.0000  2785.500  3600.667   3030.000       41           M          2
## 99   843.3333  1312.667  2367.667   2023.000       38           F          1
## 114 1379.6667  2348.667  1689.333   1348.000       72           M          2
## 141  932.8333  1544.167  2298.333   1886.167       54           F          2
##     Clin_Histology Clin_BiomarkerAPreSurgery Clin_TreatmentType
## 10               A                        67                  1
## 17               A                        47                  1
## 25               A                        26                  1
## 77               C                        59                  1
## 97               C                        29                  1
## 99              C                       27                 1
## 114             D                       72                 1
## 141             E                       70                 0
##     Clin_PerformanceScore Clin_BiomarkerAPostSurgery Clin_SurgeryType
## 10                     30                         43                0
## 17                     40                         58                0
## 25                     40                         69                1
## 77                     70                         65                0
## 97                     40                         67                0
## 99                     60                         62                0
## 114                    60                         69                0
## 141                    30                         52                1
```