BMI 713/BMIF Problem Set 1 Solution

Niklas Rindtorff

2018-09-12

Overview:

I load a package collection in the background.

Question 1 (4 pts)

1.2 Getting Help (2 pts)

Please state what the arguments to the "sum" function in R?

It may be useful to use some of R's built in help functionality.

```
# Get help on the function sum

# ANSWER 1.2
# The arguments of the sum function are:
# * A vector with either numeric, logical (True/False = 1/0) or complex numbers
# * An na.rm argument. This argument is a logical and decides if vectors with a
#missing value are going to produce a missing value as overall result or if NAs
#are dropped during summation.
```

1.3 Using Help (2 pts)

```
vector_example <- c(5,6,7,8)
vector_example
```

```
## [1] 5 6 7 8
```

Using the R documentation for help if needed, please write one to two lines of code to create a vector of numbers ranging from 50 to 100 (inclusive) and report the standard deviation of this vector. Credit will **NOT** be given if the entire vector is manually entered like the example below:

```
my_vector <- c(50:100)
sd(my_vector)
```

```
## [1] 14.86607
```

```
# ANSWER 1.3
# The vector can be generated using the colon operator
# The standard deviation is about 14.87
```

```
set.seed(42)
# Create random vector
random_vector <- runif(50, min=0, max=50)
# Get standard deviation of vector
sd(random_vector)
```

```
## [1] 15.18244
```

Question 2 (12 pts)

Basic R syntax

2.1 Indexing (1 pts)

Each element in our vector, vector_example is stored in a position that can be addressed by its index. Different programming languages use different starting indices. Write the command that will index the first value in vector_example.

```
# Index first element
vector_example[1]
```

```
## [1] 5
```

```
# ANSWER 2.1
#R starts indexing at 1 (different from many other programming languages like Python)
```

2.2 Booleans & subsetting (4 pts)

Now write a code that will:

1) Create variable *c* which stores whether each number in vector_example is even or odd (*hint* logical vector)
2) Using vector *c*, create two separate vectors, Evens_example and Odds_example, which store the even and odd values of vector_example, respectively.

Your answer should make use of of an ifelse statement and appropriate subsetting.

```
# create vector C
# elegant alternative: c <- as.logical(vector_example%%2)
c <- ifelse(vector_example%%2, 1, 0)

# create Evens_example
Evens_example <- vector_example[!c]

# create Odds_example
Odds_example <- vector_example[c]

#ANSWER 2.2
#I calculate the modulo 2 for each number in the vector.
#Uneven elements have a residual of one.
#By using a logical transformation, I turn the residual value of 1 into a TRUE statement.
#This statement hence indicates an uneven number at its respective index.
#Next I use c as an index-vector to return even and uneven numbers.
#To return even numbers I use the logical negating variable "!".
```

2.3 Understanding Syntax (2 pts)

In our first lecture, we worked through of data normalization in which a random 5x20 example matrix needed to be normalized by column (scaling values so that they are between 0 and 1). Now write a similar normalization function from so that it normalizes *by rows* instead of columns.

Remember: check your logic for the normalization procedure as we corrected in the example in class.

```
# Create row-wise normalization function
normalize_row_wise <- function(input_matrix){
```

```r
require(magrittr)
require(dplyr)

input_matrix %>% t() %>%
  #I use my function and insert two transpositions, here
  as.data.frame() %>%
  mutate_all(funs(.-min(.))) %>%
  # I subtract the minimal value from all values (thereby creating a 0 value field per column)
  mutate_all(funs(./max(.))) %>%
  # Afterwards I scale all values by the maximum value, making the latter equal to 1.
  as.matrix() %>% t() %>% #and here
    return()
}

#It was not clear to me wether we have to generate a "classic" matrix normalization
#or if we should scale every row between 0 and 1. English is not my first language.
# normalize_simple <- function(input_matrix){
#   return((input_matrix-min(input_matrix))/max(input_matrix-min(input_matrix)))
# }
```

2.4 Adding conditionals (5 pts)

Write a normalization function *(column-wise)* that takes an input matrix as an argument and returns nothing if the normalized matrix is equal to the input matrix but otherwise returns normalized matrix.

Is a return statement needed in an R function (look back to the "normalize" functions from lecture and in 2.4)?

```r
# Create column-wise normalization function
column_wise <- function(input_matrix){
require(magrittr)
require(dplyr)

output_matrix <- input_matrix %>%
  as.data.frame() %>%
  mutate_all(funs(.-min(.))) %>%
  # I subtract the minimal value from all values (thereby creating a 0 value field per column)
  mutate_all(funs(./max(.))) %>%
  # Afterwards I scale all columns by their maximum value, making the latter equal to 1.
  as.matrix() %>%
  unname() #helper to remove all names that have been introduced during conversions

if(!identical(input_matrix, output_matrix)){return(output_matrix)} else{return()}
}

# ANSWER 2.4
# The function is very similar to the function above with the only difference
#that I did not introduce a transposition twice. The output of the function is controlled
#by a conditional statement which depends on the result of identical(). Since identical() is
#very strict, I have to remove all object attributes such as name from the
#output matrix before comparison with unname().
#A return statement is not necessary in R functions, as R will always
#return the last modified object in the function envionment. In general, a return()
#argument is considered good style. In my implementation it is necessary to control the output.
```

Question 3 (5 pts)

a) What is one reason why using the "assign" function in R is stylistically a bad choice?

b) Which two other operators are most commonly utilized for variable assignment?

c) Which of these two operators is better to use stylistically?

Write a short piece of code that utilizes both possible operators *most commonly* utilized for variable assignment at least once where swapping all occurence of each operator for the other results in a different output or generate an error.

```
# Using Operator 1
y = 5
plot(x = 1,y = 3)
y

# Replace w/ Operator 2 and get error/different output
y <- 5
plot(x <-  1,y <- 3)
y

# Using Operator 2
x <- 3
c(mean(x <- 1:15), x)

# Replace w/ Operator 1 and get error/different output
x = 3
c(mean(x = 1:15), x)


# ANSWER 3a
#Assign sets "a value to a name in an envionment" (R Documentation).
#This environment, however, is most of the time not specified.
#This leads to an assignment of the name and value pair in the general envionment,
#even if assign() is used in a seperate function or loop.
#Therefore, if you set x <- 3 in your script and you would run an unrelated third-party
#function afterwards that uses the name "x" and assign() to assign a value,
#you could not expect x==3 to be TRUE.
#Moreover, assign breaks the magrittr pipe operator %>% which constructs
#object-overwriting helper functions to proccess a pipe.

# ANSWER 3b
# Values can be assigned with <- and =

# ANSWER 3c
# <- should be preferred acc. to the Google Style Guide for R
```

Question 4 (10 pts)

For Loops and Apply

Find the drive time for each car in the **cars** dataset. (Hint: Time = Distance/Speed)

```
# Cars is a built-in dataset
# You can use head() to view the dataset
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
times_elegant <- cars %>%
  mutate(time = dist/speed)

head(times_elegant, 20)
```

```
##    speed dist      time
## 1      4    2 0.5000000
## 2      4   10 2.5000000
## 3      7    4 0.5714286
## 4      7   22 3.1428571
## 5      8   16 2.0000000
## 6      9   10 1.1111111
## 7     10   18 1.8000000
## 8     10   26 2.6000000
## 9     10   34 3.4000000
## 10    11   17 1.5454545
## 11    11   28 2.5454545
## 12    12   14 1.1666667
## 13    12   20 1.6666667
## 14    12   24 2.0000000
## 15    12   28 2.3333333
## 16    13   26 2.0000000
## 17    13   34 2.6153846
## 18    13   34 2.6153846
## 19    13   46 3.5384615
## 20    14   26 1.8571429
```

```
#ANSWER
#I use the mutate() function from the dplyr package to introduce
#a rowwise operation to the dataframe
```

4a. Using For (3)

Using a For Loop, find and report the time for each car.

```
# Calculate drive time for each car using for loop
times_loop <- c()
for(ii in 1:nrow(cars)){
  times_loop[[ii]] <- cars$dist[ii]/cars$speed[ii]
}

# Display the drive times
times_loop
```

```
##  [1] 0.5000000 2.5000000 0.5714286 3.1428571 2.0000000 1.1111111 1.8000000
##  [8] 2.6000000 3.4000000 1.5454545 2.5454545 1.1666667 1.6666667 2.0000000
## [15] 2.3333333 2.0000000 2.6153846 2.6153846 3.5384615 1.8571429 2.5714286
## [22] 4.2857143 5.7142857 1.3333333 1.7333333 3.6000000 2.0000000 2.5000000
## [29] 1.8823529 2.3529412 2.9411765 2.3333333 3.1111111 4.2222222 4.6666667
## [36] 1.8947368 2.4210526 3.5789474 1.6000000 2.4000000 2.6000000 2.8000000
## [43] 3.2000000 3.0000000 2.3478261 2.9166667 3.8333333 3.8750000 5.0000000
## [50] 3.4000000
```

4b. Now using apply (3)

Using Apply, find and report the time for each car.

```
# Calculate drive time using apply()
times_apply <- apply(cars, 1, function(x){x[2]/x[1]})
# Display the drive times
times_apply
```

```
##  [1] 0.5000000 2.5000000 0.5714286 3.1428571 2.0000000 1.1111111 1.8000000
##  [8] 2.6000000 3.4000000 1.5454545 2.5454545 1.1666667 1.6666667 2.0000000
## [15] 2.3333333 2.0000000 2.6153846 2.6153846 3.5384615 1.8571429 2.5714286
## [22] 4.2857143 5.7142857 1.3333333 1.7333333 3.6000000 2.0000000 2.5000000
## [29] 1.8823529 2.3529412 2.9411765 2.3333333 3.1111111 4.2222222 4.6666667
## [36] 1.8947368 2.4210526 3.5789474 1.6000000 2.4000000 2.6000000 2.8000000
## [43] 3.2000000 3.0000000 2.3478261 2.9166667 3.8333333 3.8750000 5.0000000
## [50] 3.4000000
```

Question 5 (7 pts)

5a. Scope Example

If the below block of code were to be executed, what are the resulting values of a and b?

```
a <- c(3,6,2,3,6,4,1,7,6,3,9)
b <- 0.5
function_1 <- function(a,b){
  b <- lapply(a,function(x) x/b)
  b
}
a = b <- function_1(a,b)
```

```
a <- c(3,6,2,3,6,4,1,7,6,3,9)
b <- 0.5
function_1 <- function(a,b){
  b <- lapply(a,function(x) x/b)
  b
}
a = b <- function_1(a,b)

identical(a,b)

# ANSWER 5a
# a and b are identical lists of 11 numeric elements, representing the
#initial values of a*2. The initial values in "a" are coerced to a list by
#list-apply (aka lappy), which divides the values in "a" by 0.5 and stores them
#in a list with the name "b". "b" then is assigned to "a".
```

5b. A Questions On Scope

a) How does variable assignment within a function affect a variable that has already been previously assigned a value outside a function (look back to question 5a and to lecture)?

```
# ANSWER 5b.
# It does not effect the value outside the function in general (if assign() was not used).
#The assigned objects outside the function are usually part of the global enviromment
#while objects inside the function are located in a separate envionment.
```

Question 6 (12 pts)

Conditionals

We explored 2 types of conditionals: if/else and switches.

Use the iris dataset for these exercises.

```
# Iris is a built-in dataset
# You can use head() to view the dataset
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

6a. If/else

i. Define a function using *if/else* and *apply()* that will return sepal length as long as petal length / petal width > 5, otherwise return NA. Use apply to run this on the iris dataset and report your results. (3)

```
# Create function using if/else
f_ifelse <- function(row_var){
  if(row_var[3]/row_var[4] > 5){return(row_var[1])}
  else{return(NA)}
}

# Call function using apply() and report results
apply(iris[,1:4], 1, f_ifelse)
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0  NA  NA 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8  NA  NA
##  [18]  NA 5.7  NA 5.4  NA  NA  NA 4.8 5.0  NA 5.2 5.2 4.7 4.8  NA 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1  NA  NA 4.4  NA  NA  NA 5.1 4.6 5.3 5.0  NA
##  [52]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [69]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [86]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [103]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [120]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [137]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

```
#ANSWER
#The function is based on an if() and else() element
```

ii. Instead of using apply, do the same thing using the *ifelse()* function. Run using the iris dataset, and report your results. (2)

```
# Use ifelse() and report results
iris[ifelse(iris$Petal.Length/iris$Petal.Width > 5, TRUE, NA),1]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0  NA  NA 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8  NA  NA
##  [18]  NA 5.7  NA 5.4  NA  NA  NA 4.8 5.0  NA 5.2 5.2 4.7 4.8  NA 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1  NA  NA 4.4  NA  NA  NA 5.1 4.6 5.3 5.0  NA
##  [52]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [69]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [86]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [103]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [120]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## [137]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

6b. Switches (3)

Using *switch*, define a function that will return sepal dimensions (Length and width) for setosa flowers and return Petal demensions (Length and width) otherwise. Run this on iris and report your results.

```r
# Create function using switch
# elegant_switch <- function(df, switcher){
#   switch(switcher,
#          sepal = df %>% as.tibble %>%
#            filter(Species == "setosa") %>%
#            select(Sepal.Length, Sepal.Width) %>%
#            as.data.frame(),
#          df %>% as.tibble %>%
#            filter(Species != "setosa") %>%
#            select(Petal.Length, Petal.Width) %>%
#            as.data.frame())
# }

#Create another function using switch that needs apply to go over the df.
apply_switch <- function(row_var, switcher){

  switch (switcher,
    sepal = if(row_var[5] == "setosa"){return(c(row_var[1], row_var[2]) %>% unlist())}
    else{return(c(NA, NA))},
    if(row_var[5] != "setosa"){return(c(row_var[3], row_var[4]) %>% unlist())}
    else{return(c(NA, NA))}
  )
}

# Call function using apply() and report results
switch_result <- apply(iris, 1, apply_switch, switcher = "sepal") %>% t() %>% as.tibble() %>%
  magrittr::set_colnames(c("my Variable 1", "my Variable 2"))
head(switch_result)
```

```
## # A tibble: 6 x 2
##   `my Variable 1` `my Variable 2`
##   <chr>           <chr>
## 1 5.1             3.5
## 2 4.9             3.0
## 3 4.7             3.2
## 4 4.6             3.1
## 5 5.0             3.6
## 6 5.4             3.9
```

```r
switch_result <- apply(iris, 1, apply_switch, switcher = "petal") %>% t() %>% as.tibble() %>%
  magrittr::set_colnames(c("my Variable 1", "my Variable 2"))
head(switch_result)
```

```
## # A tibble: 6 x 2
##   `my Variable 1` `my Variable 2`
##   <chr>           <chr>
## 1 <NA>            <NA>
## 2 <NA>            <NA>
## 3 <NA>            <NA>
## 4 <NA>            <NA>
## 5 <NA>            <NA>
## 6 <NA>            <NA>
```

Question 7 (1 pt)

How long did this problem set take to complete?

Please report this number in hours. **4 hours**

Session Info

```r
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] bindrcpp_0.2.2  knitr_1.20      forcats_0.3.0   stringr_1.3.1
##  [5] dplyr_0.7.6     purrr_0.2.5     readr_1.1.1     tidyr_0.8.1
##  [9] tibble_1.4.2    ggplot2_3.0.0   tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.18     cellranger_1.1.0 pillar_1.3.0     compiler_3.5.1
##  [5] plyr_1.8.4       bindr_0.1.1      tools_3.5.1      digest_0.6.16
##  [9] lubridate_1.7.4  jsonlite_1.5     evaluate_0.11    nlme_3.1-137
## [13] gtable_0.2.0     lattice_0.20-35  pkgconfig_2.0.2  rlang_0.2.2
## [17] cli_1.0.0        rstudioapi_0.7   yaml_2.2.0       haven_1.1.2
## [21] withr_2.1.2      xml2_1.2.0       httr_1.3.1       hms_0.4.2
## [25] rprojroot_1.3-2  grid_3.5.1       tidyselect_0.2.4 glue_1.3.0
## [29] R6_2.2.2         fansi_0.3.0      readxl_1.1.0     rmarkdown_1.10
## [33] modelr_0.1.2     magrittr_1.5     backports_1.1.2  scales_1.0.0
## [37] htmltools_0.3.6  rvest_0.3.2      assertthat_0.2.0 colorspace_1.3-2
## [41] utf8_1.1.4       stringi_1.2.4    lazyeval_0.2.1   munsell_0.5.0
## [45] broom_0.5.0      crayon_1.3.4
```