DS413613 HOMEWORK 3 KEY

James Dickens

1/28/2022

```
# DS 413/613
# HOMEWORK 3  Vectors, Lists, Functions
# KEY     53 total points

# coding answers may vary slightly. If codint is similar
# and the output is correct consider granting full credit

Vector1 <- (c( 10, 19, 121, 83, 63, 7, 77, 61, 51, 97,
            123, 41))
Vector1
```

```
##  [1]  10  19 121  83  63   7  77  61  51  97 123  41
```

```
# 1) For the vector given above, use and show two methods
# of R coding to extract the first element and the last
# element.  6 points

# method 1

Vector1[c(1,12)]
```

```
## [1] 10 41
```

```
# method 2

Vector1[-c(2:11)]
```

```
## [1] 10 41
```

```
# 2) For the vector given above, use and show two methods
# of R coding to extract all of the elements that are less
# than 60.           6 points

# possible and suggested methods
# method 1
Vector1[Vector1 < 60]
```

```
## [1] 10 19  7 51 41
```

```
# method 2
Vector1[!(Vector1 >= 60)]
```

```
## [1] 10 19  7 51 41
```

```r
# method 3
Vector1[c(1,2,6,9,12)]
```

```
## [1] 10 19  7 51 41
```

```r
# 3) For the vector given above, use and show two
# methods of R coding to extract all numbers that are
# not divisible by 2 or 3.    6 points

# The numbers from the vector are not divisible by 2 or
# the numbers are not divisible by 3  (all numbers !!)

# method 1
Vector1[!(Vector1 %% 2 == 0) | !(Vector1 %% 3 == 0)]
```

```
##  [1]  10  19 121  83  63   7  77  61  51  97 123  41
```

```r
# method 2
Vector1[]
```

```
##  [1]  10  19 121  83  63   7  77  61  51  97 123  41
```

```r
# 4) Use and show two R coding methods to confirm that
# Vector1 does not have missing values   6 points

# method 1
is.na(Vector1) # confirming that all elements evaluated
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```r
               # for missing is FALSE
# method 2
(Vector1[ ]) # confirming every element in Vector1. Note
```

```
##  [1]  10  19 121  83  63   7  77  61  51  97 123  41
```

```r
             # that there are no missing elements

# Use the list above for problems 5 - 7.

myList <- list(TRUE, 12.35, "pear",48, c = 3:8,
               list(23, "team"))
myList
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] 12.35
##
## [[3]]
```

```
## [1] "pear"
##
## [[4]]
## [1] 48
##
## $c
## [1] 3 4 5 6 7 8
##
## [[6]]
## [[6]][[1]]
## [1] 23
##
## [[6]][[2]]
## [1] "team"
```

```
# (note: it is better to type the list into R studio or
# R markdown. Do not copy and paste)

# 5) For the list given above, use and show R coding to
#  confirm that "pear" is a character element.  4 points

str(myList)
```

```
## List of 6
##  $  : logi TRUE
##  $  : num 12.3
##  $  : chr "pear"
##  $  : num 48
##  $ c: int [1:6] 3 4 5 6 7 8
##  $  :List of 2
##   ..$ : num 23
##   ..$ : chr "team"
```

```
# 6) For the list given above, use and show R coding to
#    extract the first three elements of the list.
#  4 points

myList[1:3]
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] 12.35
##
## [[3]]
## [1] "pear"
```

```
# 7) Use the $ operator to extract the element "pear"
# from your list. Be sure to use and show required R code
# to produce the requested output.
```

```r
# students will be expected to assign the character
# element to a variable and then apply $ to the variable.

# 4 points

myList <- list(TRUE, 12.35, k = "pear",48, c = 3:8,
               list(23, "team"))
myList
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] 12.35
##
## $k
## [1] "pear"
##
## [[4]]
## [1] 48
##
## $c
## [1] 3 4 5 6 7 8
##
## [[6]]
## [[6]][[1]]
## [1] 23
##
## [[6]][[2]]
## [1] "team"
```

```r
myList$k
```

```
## [1] "pear"
```

```r
# 8) Use and show R code to write a function to solve the
# following quadratic equations by using the quadratic
# formula. (all equations have two real number solutions)

# a) x2 - 3x - 28 = 0
# b) x2 + x  - 30 = 0
# c) 3x2+ 14x + 8 = 0
# d) 2x2+11x = 6

# 7 points


QuadFormula <- function(a,b,c){
  answer1 <- (-b - sqrt(b^2 - (4*a*c)))/(2*a)
```

```
  answer2<-  (-b + sqrt(b^2 - (4*a*c)))/(2*a)
  return(c(answer1 = answer1, answer2 = answer2))

}

QuadFormula(1,-3,-28)

## answer1 answer2
##       -4       7

QuadFormula(1,1,-30)

## answer1 answer2
##       -6       5

QuadFormula(3,14,8)

##    answer1    answer2
## -4.0000000 -0.6666667

QuadFormula(2,11,-6)

## answer1 answer2
##    -6.0     0.5

# 9) In your book (towards the end of chapter 16) a
# special set of vectors are defined as Augmented
# Vectors.  One such augmented vector is a Tibble.
# Use and show R code that will produce the Tibble
# shown below.  Do not simply type or copy and paste.
# You must show and use R coding that will output the
# tibble.

#  6 points

# suggested method:

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

tibble(x = 1:10, y = 10:1, z = .5*y)

## # A tibble: 10 x 3
##        x     y     z
##    <int> <int> <dbl>
##  1     1    10   5
##  2     2     9   4.5
##  3     3     8   4
##  4     4     7   3.5
##  5     5     6   3
##  6     6     5   2.5
##  7     7     4   2
##  8     8     3   1.5
##  9     9     2   1
## 10    10     1   0.5

# 10 In statistics, the Interquartile Range is the
# difference between Q3 and Q1.  Now show and use map
# function coding to find the Interquartile Range for
# each column of the tibble from number 9.

# suggested solution coding

# 4 points


tibble(x = 1:10, y = 10:1, z = .5*y) -> anyvariable
anyvariable

## # A tibble: 10 x 3
##        x     y     z
##    <int> <int> <dbl>
##  1     1    10   5
##  2     2     9   4.5
##  3     3     8   4
##  4     4     7   3.5
##  5     5     6   3
##  6     6     5   2.5
##  7     7     4   2
##  8     8     3   1.5
##  9     9     2   1
## 10    10     1   0.5
```

```
map_dbl(anyvariable, IQR)

##    x    y    z
## 4.50 4.50 2.25
```