Homework 2

Sasha Uspenskaya

2024-01-30

```
library(tidyverse)

## Warning: package 'tidyr' was built under R version 4.3.2

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors

library(dplyr)
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.

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

## [1] 10 41

Vector1[c(-2,-3,-4,-5,-6,-7,-8,-9,-10,-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.

```
Vector1[c(1,2,6,9,12)]

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

```
Vector1[Vector1<60]

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

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.

```
Vector1[Vector1 %% 2 != 0 & Vector1 %% 3 != 0]

## [1]  19 121  83   7  77  61  97  41

for (i in 1:12){
  if (Vector1[i]%% 2 != 0 & Vector1[i]%% 3 != 0){
    print(Vector1[i])
  }
}

## [1] 19
## [1] 121
## [1] 83
## [1] 7
## [1] 77
## [1] 61
## [1] 97
## [1] 41
```

4) Use and show two R coding methods to confirm that Vector1 does not have missing values.

```
!any(is.na(Vector1))

## [1] TRUE

sum(is.na(Vector1)) == 0

## [1] TRUE

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"
```

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

```
is.character(myList[[3]])

## [1] TRUE
```

6) For the list given above, use and show R coding to extract the first three elements of the list

```
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.

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

## [1] "pear"
```

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) $x^2 - 3x - 28 = 0$

```
quad <- function(a, b, c) {
  soln <- b^2-4*a*c
  root <- (-b + sqrt(soln)) / (2 * a)
  root1 <- (-b - sqrt(soln)) / (2 * a)
  return(c(root, root1))
}
quad(1,-3,-28)

## [1]  7 -4
```

##b) $x^2 + x - 30 = 0$

```
quad(1,1,-30)

## [1]  5 -6
```

c)

$$3x^2 + 14x + 8 = 0$$

```
quad(3,14,8)

## [1] -0.6666667 -4.0000000
```

$$d)2x^2 + 11x = 6$$

```
quad(2,11,-6)

## [1]  0.5 -6.0
```

9)

```
tribble(~x, ~y, ~z,
1, 10, 5,
2, 9, 4.5,
3, 8, 4,
4, 7, 3.5,
5, 6, 3,
6, 5, 2.5,
7, 4, 2,
8, 3, 1.5,
9, 2, 1,
10, 1, 0.5
) -> df
df
```

```
## # A tibble: 10 × 3
##        x     y     z
##    <dbl> <dbl> <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.

```
map_df(df, IQR)
```

```
## # A tibble: 1 × 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1   4.5   4.5  2.25
```