Untitled

STAT613\_InCLassExercise021021

2/10/2021

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2 v purrr 0.3.4  
## v tibble 3.0.1 v dplyr 1.0.0  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(dplyr)

#QUES 1):

V <- c("Bears", "Lions", "Dolphins", "Eagles", "Bengals")

#Why is the vector show is an atomic vector? (Explain using two #or three sentences)

#Answer 1):Atomic vector is a collection tool. In the above example, the #vector contains objects which are all characters qualifying as Atomic vector #since the vector contains objects of the same type whether it is all #numeric, all characters, all logical etc.

#QUES 2): Use and show R code that will extract “Dolphins” from the vector #shown above.

V <- c("Bears", "Lions", "Dolphins", "Eagles", "Bengals")  
  
V[3]

## [1] "Dolphins"

#QUES 3): Use and show Rcode that will extract “Bears” , “Dolphins” and #“Bengals” from the vector shown above.

V[c(1,3,5)]

## [1] "Bears" "Dolphins" "Bengals"

#QUES 4): Use and show two Rcoding methods that will show all objects of #the vector given above except “Bears”. #METHOD 1):

V[!V == "Bears"]

## [1] "Lions" "Dolphins" "Eagles" "Bengals"

#METHOD 2):

V[-1]

## [1] "Lions" "Dolphins" "Eagles" "Bengals"

#QUES 5):

K <- list( x = 3:7, "never", 43, y = list(10,20,30))

#Why is the vector given above called a list? (Explain in two or #three sentences) If the vector is a list, identify the type of each object in #the list.

#Answer 5): Unlike Atomic vectors, list vectors can contain different types #or combination of objects. It can also contain “lists” within the vector. #This is also an example of heterogenous list. #In the above example, we have a combination of the following: #List of 4: #list: x=3:7 and y=list(10,20,30) #character: “never” #numeric/integer strings: 43

str(K)

## List of 4  
## $ x: int [1:5] 3 4 5 6 7  
## $ : chr "never"  
## $ : num 43  
## $ y:List of 3  
## ..$ : num 10  
## ..$ : num 20  
## ..$ : num 30

#Ques 6): Use and show R code that will give the length of the vector shown #above.

length(K)

## [1] 4

#Ques 7): Use and show R code that will output the fourth object in the vector #shown above. #Method 1:

K[[4]]

## [[1]]  
## [1] 10  
##   
## [[2]]  
## [1] 20  
##   
## [[3]]  
## [1] 30

#Method 2:

K$y

## [[1]]  
## [1] 10  
##   
## [[2]]  
## [1] 20  
##   
## [[3]]  
## [1] 30

#Ques 8): Use and show R code that will show all objects in the vector (list) #given above.

K[]

## $x  
## [1] 3 4 5 6 7  
##   
## [[2]]  
## [1] "never"  
##   
## [[3]]  
## [1] 43  
##   
## $y  
## $y[[1]]  
## [1] 10  
##   
## $y[[2]]  
## [1] 20  
##   
## $y[[3]]  
## [1] 30

#Ques 9): #Copy paste and run the tribble given below.

tribble( ~x, ~y, ~w, ~z,  
 210, 300, 220, 180,  
 102, 100, 119, 187,  
 176, 175, 188, 173,  
 87, 95, 91, 94,  
 202, 210, 234, 218,  
 110, 122, 131, 128,  
) -> dt

#Ques 9a): Use and show a map function to find the mean of each column of #the dt data table

map\_dbl(dt, mean)

## x y w z   
## 147.8333 167.0000 163.8333 163.3333

#Ques 9b): Use and show a map function to find the standard deviation of #each column of the dt data table.

map\_dbl(dt, sd)

## x y w z   
## 54.45151 79.12016 58.40348 44.66617

#Ques 9c): Use and show a map function that will calculate the square root #of each value of each column of the data table dt. #Using MAP FUNCTION:

map(dt, sqrt)

## $x  
## [1] 14.491377 10.099505 13.266499 9.327379 14.212670 10.488088  
##   
## $y  
## [1] 17.320508 10.000000 13.228757 9.746794 14.491377 11.045361  
##   
## $w  
## [1] 14.832397 10.908712 13.711309 9.539392 15.297059 11.445523  
##   
## $z  
## [1] 13.41641 13.67479 13.15295 9.69536 14.76482 11.31371

#Using PIPING METHOD:

dt%>%  
 map(sqrt)

## $x  
## [1] 14.491377 10.099505 13.266499 9.327379 14.212670 10.488088  
##   
## $y  
## [1] 17.320508 10.000000 13.228757 9.746794 14.491377 11.045361  
##   
## $w  
## [1] 14.832397 10.908712 13.711309 9.539392 15.297059 11.445523  
##   
## $z  
## [1] 13.41641 13.67479 13.15295 9.69536 14.76482 11.31371

#Ques 9d): Use R code to find the mean, max, 1st Quartile, 2nd Quartile, #Median, and Mean for each column of the dt data table. (Hint: You do #not have to use a map function).

summary(dt)

## x y w z   
## Min. : 87.0 Min. : 95.0 Min. : 91.0 Min. : 94.0   
## 1st Qu.:104.0 1st Qu.:105.5 1st Qu.:122.0 1st Qu.:139.2   
## Median :143.0 Median :148.5 Median :159.5 Median :176.5   
## Mean :147.8 Mean :167.0 Mean :163.8 Mean :163.3   
## 3rd Qu.:195.5 3rd Qu.:201.2 3rd Qu.:212.0 3rd Qu.:185.2   
## Max. :210.0 Max. :300.0 Max. :234.0 Max. :218.0