Lab 2

Alexandra Uspenskaya

2024-01-29

```
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)
```

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

V <- c("Bears", "Lions", "Dolphins", "Eagles", "Bengals") An Atomic Vector is homogeneous, meaning the items in the vector are all the same type. All items in the vector above are character strings.

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"
```

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"
```

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

```
V[-1]

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

V[c(2,3,4,5)]

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

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

K <- list(x = 3:7, “never”, 43, y = list(10,20,30)) We are storing elements of different types in the same vector. To this we create it using the list() function.

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

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

## [1] 4
```

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

```
K[4]

## $y
## $y[[1]]
## [1] 10
##
## $y[[2]]
## [1] 20
##
## $y[[3]]
## [1] 30
```

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

```
K[c(1,2,3,4)]

## $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
```

#9)

```
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
dt

## # A tibble: 6 × 4
##       x     y     w     z
##   <dbl> <dbl> <dbl> <dbl>
## 1   210   300   220   180
## 2   102   100   119   187
## 3   176   175   188   173
## 4    87    95    91    94
## 5   202   210   234   218
## 6   110   122   131   128
```

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

```
mean(dt$x)

## [1] 147.8333

mean(dt$y)

## [1] 167
```

```
mean(dt$w)

## [1] 163.8333

mean(dt$z)

## [1] 163.3333
```

#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 
```

9c) Use and show a map function that will calculate the square root of each value of each column of the data table dt

```
map_df(dt, sqrt)

## # A tibble: 6 × 4
##       x     y     w     z
##   <dbl> <dbl> <dbl> <dbl>
## 1 14.5  17.3  14.8  13.4 
## 2 10.1  10    10.9  13.7 
## 3 13.3  13.2  13.7  13.2 
## 4  9.33  9.75  9.54  9.70
## 5 14.2  14.5  15.3  14.8 
## 6 10.5  11.0  11.4  11.3 
```

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)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    87.0   104.0   143.0   147.8   195.5   210.0 

summary(dt$y)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    95.0   105.5   148.5   167.0   201.2   300.0 

summary(dt$w)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    91.0   122.0   159.5   163.8   212.0   234.0 
```

```
summary(dt$z)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    94.0   139.2   176.5   163.3   185.2   218.0
```

10)

```
x <- list(26, 32, 45, 50, 65, 77, 82)
y <- list(30, 43, 50, 58, 62, 71, 88)
```

For the lists given above, show and use R code (a map function) to iteratively find:

a) sums across the two vectors. (Use two methods)

```
map_dbl(1:7, ~x[[.x]]+y[[.x]])

## [1]  56  75  95 108 127 148 170

summ<- function(x,y)
  for (i in seq_along(x)){
    print(x[[i]]+y[[i]])
  }
summ(x,y)

## [1] 56
## [1] 75
## [1] 95
## [1] 108
## [1] 127
## [1] 148
## [1] 170
```

##b) the calculation of the square of the x value minus the square root of the y value.

```
map_dbl(1:7, ~sqrt(x[[.x]])-sqrt(y[[.x]]))

## [1] -0.3782061 -0.9005843 -0.3628639 -0.5447053  0.1882499  0.3488146 -
0.3254464
```

##c) the ratio of the common log of the x value to the natural log of the y value.

```
map_dbl(1:7, ~log(x[[.x]])/log(y[[.x]]))

## [1] 0.9579263 0.9214442 0.9730675 0.9634473 1.0114493 1.0190316 0.9842278
```