

Homework 1

Daniel Carpenter

August 2022

Table of contents

| | | |
|---|------------------------------|---|
| 1 | Using R: Vectors | 2 |
| 2 | Using R: Some missing values | 8 |

1 Using R: Vectors

1.1 (a)

Create a vector with 10 numbers (3, 12, 6, -5, 0, 8, 15, 1, -10, 7) and assign it to x.

```
x <- c(3, 12, 6, -5, 0, 8, 15, 1, -10, 7)
```

1.2 (b)

Using the commands seq, min, and max with one line of code create a new vector y with 10 elements ranging from the minimum value of x to the maximum value of x.

```
# Min, max of x vector
xMin = min(x)
xMax = max(x)
INCREMENT = (xMax - xMin) / (length(x) - 1) # size of each step

# Create the vector from min of x to max of x while maintaining length of 10
y = seq(min(x), max(x), INCREMENT)
y # display
```

```
[1] -10.000000 -7.222222 -4.444444 -1.666667  1.111111  3.888889
[7]  6.666667  9.444444 12.222222 15.000000
```

```
# Prove is 10 elements:
paste('The length of y is:', length(y) )
```

```
[1] "The length of y is: 10"
```

1.3 (c)

1.3.1 Compute the sum, mean, standard deviation, variance, mean absolute deviation, quartiles

```
# Combine x and y into single object
xAndY <- cbind(x, y)

# This is what x and y look like
head(xAndY)
```

```
      x      y
[1,]  3 -10.000000
[2,] 12  -7.222222
[3,]  6  -4.444444
[4,] -5  -1.666667
[5,]  0   1.111111
[6,]  8   3.888889
```

```
# List of all the function to calculate on x and y (excluding quintile)
vectorOfFuns <- c('sum', 'mean', 'sd', 'var', 'mad', 'quantile')
vectorOfFunsNames <- c('sum', 'mean', 'standard deviation', 'variance', # function full na
                        'mean absolute deviation', 'quartiles')

for (funIdx in 1:length(vectorOfFuns)) {

  # Print result of the function calculation output, for x and y
  print( paste( vectorOfFunsNames[funIdx], 'of x and y:' ) )

  # Calculate using a function from `vectorOfFuns`
  output <- apply(xAndY,                                # Using X and Y
                  ncol(xAndY),                          # There are n cols in x & y
                  paste0(vectorOfFuns[funIdx]) ) # Retrieve/apply function

  print(output)
}
```

```
[1] "sum of x and y:"
   x  y
37 25
```

```

[1] "mean of x and y:"
      x      y
3.7 2.5
[1] "standard deviation of x and y:"
      x      y
7.572611 8.410140
[1] "variance of x and y:"
      x      y
57.34444 70.73045
[1] "mean absolute deviation of x and y:"
      x      y
5.93040 10.29583
[1] "quartiles of x and y:"
      x      y
0%   -10.00 -10.00
25%    0.25  -3.75
50%    4.50   2.50
75%    7.75   8.75
100%   15.00  15.00

```

1.3.2 Compute the Quintiles

```

# Quintile for x
quantile(x, probs = seq(0, 1, 0.20))

```

```

      0%    20%    40%    60%    80%   100%
-10.0  -1.0    2.2    6.4    8.8   15.0

```

```

# Quintile for y
quantile(y, probs = seq(0, 1, 0.20))

```

```

      0%          20%          40%          60%          80%
-1.000000e+01 -5.000000e+00 -1.665335e-15  5.000000e+00  1.000000e+01
      100%
 1.500000e+01

```

1.4 (d)

Use `sample()` to create a new 7 element vector `z` by using R to randomly sample from `x` with replacement.

```
z = sample(x, size=7)

z # show value
```

```
[1]  1  8 15 -10  0 -5  7
```

1.5 (e)

Use `t.test()` to compute a statistical test for differences in means between the vectors `x` and `y`.

```
t.test(x, y)
```

```
Welch Two Sample t-test
```

```
data:  x and y
t = 0.33531, df = 17.805, p-value = 0.7413
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -6.324578  8.724578
sample estimates:
mean of x mean of y
    3.7      2.5
```

Are the differences in means significant?

No, they are not statistically significant (90%, 95%, or 99%), given

- t-value is $< \sim 1.96$
- p-value is $> 0.10, 0.05, 0.01$

1.6 (f)

To sort a data frame in R, use the `order()` function. Sort the vector `x` and re-run the t-test as a paired t-test.

```
# Sort x
xSorted = x[order(x)]

# Run as paired t test
t.test(x, y, paired = TRUE)
```

Paired t-test

```
data: x and y
t = 0.30858, df = 9, p-value = 0.7647
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -7.596943  9.996943
sample estimates:
mean difference
      1.2
```

1.7 (g)

Create a logical vector that identifies which numbers in x are negative.

```
xThatAreNeg = x < 0

x # reminder of what x looks like

[1] 3 12 6 -5 0 8 15 1 -10 7
```

```
xThatAreNeg # display
```

```
[1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE
```

1.8 (h)

Use this logical vector to remove all entries with negative numbers from x. (Make sure to overwrite the vector x so that the new vector x has 8 elements!)

```
x = subset(x, !xThatAreNeg)
```

```
x # display
```

```
[1]  3 12  6  0  8 15  1  7
```

2 Using R: Some missing values

2.1 (a)

2.1.1 Use the code below to create the dataframe X and

```
col1 <- c(1,2,3,NA,5)
col2 <- c(4,5,6,89,101)
col3 <- c(45,NA,66,121,201)
col4 <- c(14,NA,13,NA,27)
X <- rbind (col1,col2,col3,col4)
```

2.1.2 Then write code to display all rows in X with missing values.

```
# All rows of X that have NAs (e.g., not complete)
X[!complete.cases(X), ]
```

| | [,1] | [,2] | [,3] | [,4] | [,5] |
|------|------|------|------|------|------|
| col1 | 1 | 2 | 3 | NA | 5 |
| col3 | 45 | NA | 66 | 121 | 201 |
| col4 | 14 | NA | 13 | NA | 27 |

2.2 (b)

2.2.1 Use the following vector y for this part:

```
y <- c(3,12,99,99,7,99,21)
```

- Some statistical applications and older systems sometimes code missing values with a number, e.g., 99. In order to let R know that is a missing value you need to recode it as 'NA'. Please write a line of code that will replace any 99's in the vector y with 'NA'.
- With the updated vector y, write code that will count the number of missing values in it.