Practice Session 1

Part 1: Quarto

Quarto documents allow one to create reproducible analyses that include both written explanations as well as R code. We will use Quarto documents for all the class code and for the homework. For more information on Quarto see https://quarto.org/

1.1 Rendering Quarto documents

We can render a Quarto document to a pdf document by pressing the **Render button** which is located at the top of the Quarto file in RStudio (the button has an arrow symbol). This will generate a pdf that includes both content as well as the output of any embedded R code chunks within the document.

Note: When working on the homework, please be sure to render the Quarto document often so that you can tell right away when you make a mistake. Otherwise, it will be very difficulty to debug your code at the end when you try to submit the homework.

1.2 LaTeX symbols

We can include LaTeX symbols in our Quarto documents. For example, we can write the Greek letter alpha using the code α.

We can also write subscripts using underscores. For example, $x_{original}$.

Sometimes we will want to put marks above characters such as a hat above a letter. For example, we can write $\hat{y}$ to get a hat above the letter y. We can also write $\bar{y}$ to get a bar above it.

Exercises: Try writing the following:

- x-bar subscript 20
- Pi subscript 1
- β-hat subscript 0

Part 2: Introduction to R

Let's now discuss some of the basics of R!

2.1 Running R code

We can run R code on the console or inside of R chunks. If we press the green "play" button at the top of the R chunk, this is the same as running the code in the console.

```
1000 + 9999 # Addition
```

```
[1] 10999
```

```
10 * 4 # Multiplying 2 numbers
```

```
[1] 40
```

```
7 * 7 * 20 # Multiplying 3 numbers
```

```
[1] 980
```

```
3^4 # Exponents (i.e., 3 to the fourth power)
```

```
[1] 81
```

```
132 / 11 # Division
```

```
[1] 12
```

Exercises: Compute the area of the following shapes:

- Square with side length 4
- Circle with radius 3 (you can use 3.14 for π)
- Triangle with base 5 and height 6

2.2 Assigning values to objects

We can assign values to objects using the assignment operator `<-`.

Note, there cannot be any spaces in between the `<` and `-` symbols.

Any object that is placed on a line by itself will print the value stored in that object. This is how you will "show your work" on the homework; i.e., show that you have the correct answer.

```
a <- 45
b <- 55

z <- a + b

z
```

```
[1] 100
```

```
fruit <- "apple"
vegetable <- "carrot"
```

Exercises: Assign the areas that you computed above to corresponding variables. You can name the variables based on the shape (e.g., square <- ...).

Area of :

1. Square : $A = s^2$
2. Circle : $A = \pi r^2$
3. Triangle: $A = \frac{1}{2}bh$

```
# your code here #
```

2.3 Functions

Functions are used to perform specific tasks. Usually they take input values and return an output value.

Function use syntax that ends with parenthesis: functionName(x)

Any ideas what the functions below do?

1. round(49.209, 2)
2. toupper("panama!")

To get information about a function, use the ? before the function name.

For example: ?round

Exercises: Find the help page for the `toupper` function.

2.4 Vectors

Vectors are used to store multiple values in a single object. We can create vectors using the c() function.

We can access values in a vector using square brackets [].

Example:

```
v <- c(1, 3, 5, 7, 9) # defining a vector called v

v
```

```
[1] 1 3 5 7 9
```

```
states <- c("Alabama", "Alaska", "Arizona", "Arkansas")

states[4] # indexing or selecting the fourth element in the vector
```

```
[1] "Arkansas"
```

```
w = 1:10 # Sequence of numbers from 1 through 10

w
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Exercises: Answer the following using the vector `v` that we defined above.

- Find the sum of `v` above using the `sum` function. Save the result to a vector called `v_sum`
- Take the square root of `v` using the `sqrt` function. Save the result to a vector called `v_sqrt`
- Subtract `v_sqrt` from `v_sum`, and round it to 2 decimal places

```
# your code here #
```

Part 2: Quick review of categorical data analysis in R

Let's quickly review how to analyze categorical data in R using a fictional dataset of voters' opinions of president Trump.

2.1 Getting a simulated sample of voters

We can use the SDS1000 function `get_approval_sample()` to get a vector of random sprinkle colors.

```
# Load the SDS1000 package
library(SDS1000)

# Setting the seed so that we all get the same "random sample"
set.seed(1000)


# Get a random sample of 1000 fictional voter's opinion of president Trump
# your code here #
```

```
# Show the first 10 voters in the sample
# your code here #
```

2.2 Get the proportion:

Example: Get the proportion of voters that appropove of trump

We can use the `table()` and `prop.table()` functions to create a frequency and relative frequency table of approval ratings.

We can also use the SDS1000 `get_proprortion()` function to get the proportion of voters that approve of president Trump.

```
# Create a table of the approval ratings
# your code here #


# Create a relative frequency table of the approval ratings
# your code here #


# Get the proportion that approves
# your code here #
```

2.3 Bar plots and pie charts

We can use the `barplot()` and `pie()` functions to visualize the data.

Note: Both functions take a frequency table as input, and **not** the original vector of data.

```
# Create a bar plot of the sprinkle colors
# your code here #


# Create a pie chart of the sprinkle colors
# your code here #
```

Can you figure out how to change the color of the segments in the pie chart?

Part 3: Analyzing Quantitative Data

Mean, Median, and histogram:

Generate histograms for each of the following data sets. Use the `$` command to access the individual data sets.

1. For each histogram, add the **mean** and the **median** to the plot using `abline()`.
2. Describe the shape of the histogram, the position of the mean, and the position of median.
3. Do you see any potential outliers?

```
mydata <- data.frame(
  dat1 = -rchisq(1000, df = 1),
  dat2 = rchisq(1000, df = 1),
  dat3 = runif(1000),
  dat4 = rnorm(1000),
  dat5 = sample(c(rnorm(1000, mean = 2), rnorm(1000, mean = 10)), size = 1000)
)
```

Answers:

```
# your code here #
```