Midterm

Instructions

- Read all of these instructions closely.
- The midterm is due Monday, March 6, 2023 at 4:15pm EST.
- Submit files electronically to me via:
  - Github or
  - Slack or
  - Email
- You may use class resources and online resources during the exam. You may not use ChatGPT or any other software the write the code for you. You also may not message each other during the exam.

Question 1

For question 1, we'll use the model output from a simple regression.

```
X <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
Y <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
lm_output <- lm(Y ~ X)
```

1a

`lm_output` is a list, containing regression model output. Print only the first element of this list.

Answer: This is how we learned in class to access each element of the list.

```
#answer here
lm_output[[1]]
```

```
## (Intercept)           X
##   7.7957139  -0.6229559
```

```
# or
lm_output$coefficients
```

```
## (Intercept)           X
##   7.7957139  -0.6229559
```

This question assesses if you know how to access elements of a list, which as we discussed in class, is different than accessing elements of a vector. With a vector, we can use single brackets `[]`. However, with lists, we need to use double brackets `[[]]`.

Using only the single brackets `[]` is not appropriate for getting *only* the first element of `lm_output` because they don't remove the element from the structure of the list, which we can see if we use the `class()` or `is.list()` functions. Even though can see the first element of the list is a length 2 numeric vector, these functions show we didn't quite access just that element, because they show we're still working with a list.

```
lm_output[1]
```

```
## $coefficients
## (Intercept)           X
##   7.7957139  -0.6229559
```

```
is.list(lm_output[1])
```

```
## [1] TRUE
```

```
class(lm_output[1])
```

```
## [1] "list"
```

1b

In words, what is the **class,** or type of data, that you printed in 1a? It is not required, but you can also use code to justify your answer.

Answer:

It is of class numeric. We can see that when we print the data in 1a (it is numbers), or we can use the `class()` or `is.numeric()` functions. We could even use the `str` function, like we did in class, that describes each element of the list. The first element is named `coefficients` and contains a length 2 numeric vector: `num [1:2] 7.796 -0.623`

```
class(lm_output[[1]])
```

```
## [1] "numeric"
```

```
is.numeric(lm_output[[1]])
```

```
## [1] TRUE
```

```
str(lm_output)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 7.796 -0.623
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "X"
##  $ residuals    : Named num [1:10] -0.388 -0.15 -0.159 -0.399 0.878 ...
##   ..- attr(*, "names")= chr [1:10] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:10] -14.7394 1.0897 -0.0576 -0.2773 0.9635 ...
##   ..- attr(*, "names")= chr [1:10] "(Intercept)" "X" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:10] 5.2 4.32 4.57 3.99 4.99 ...
##   ..- attr(*, "names")= chr [1:10] "1" "2" "3" "4" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:10, 1:2] -3.162 0.316 0.316 0.316 0.316 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "X"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.32 1.43
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 8
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = Y ~ X)
##  $ terms        :Classes 'terms', 'formula'  language Y ~ X
##   .. ..- attr(*, "variables")= language list(Y, X)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "Y" "X"
##   .. .. .. ..$ : chr "X"
##   .. ..- attr(*, "term.labels")= chr "X"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(Y, X)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "Y" "X"
##  $ model        :'data.frame':   10 obs. of  2 variables:
##   ..$ Y: num [1:10] 4.81 4.17 4.41 3.59 5.87 3.83 6.03 4.89 4.32 4.69
##   ..$ X: num [1:10] 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language Y ~ X
##   .. .. ..- attr(*, "variables")= language list(Y, X)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Y" "X"
##   .. .. .. .. ..$ : chr "X"
##   .. .. ..- attr(*, "term.labels")= chr "X"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Y, X)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Y" "X"
##  - attr(*, "class")= chr "lm"
```

1c

In words, what is the **data structure** that you printed in 1a? It is not required, but you can also use code to justify your answer.

Answer:

It is a vector. We can also see it is a vector using the `str` function, showing that is is a length 2 vector: `num [1:2]`.

```
is.vector(lm_output[[1]])
```

```
## [1] TRUE
```

```
str(lm_output)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 7.796 -0.623
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "X"
##  $ residuals    : Named num [1:10] -0.388 -0.15 -0.159 -0.399 0.878 ...
##   ..- attr(*, "names")= chr [1:10] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:10] -14.7394 1.0897 -0.0576 -0.2773 0.9635 ...
##   ..- attr(*, "names")= chr [1:10] "(Intercept)" "X" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:10] 5.2 4.32 4.57 3.99 4.99 ...
##   ..- attr(*, "names")= chr [1:10] "1" "2" "3" "4" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:10, 1:2] -3.162 0.316 0.316 0.316 0.316 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:10] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "X"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.32 1.43
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 8
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = Y ~ X)
##  $ terms        :Classes 'terms', 'formula'  language Y ~ X
##   .. ..- attr(*, "variables")= language list(Y, X)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "Y" "X"
##   .. .. .. ..$ : chr "X"
##   .. ..- attr(*, "term.labels")= chr "X"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(Y, X)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "Y" "X"
##  $ model        :'data.frame':   10 obs. of  2 variables:
##   ..$ Y: num [1:10] 4.81 4.17 4.41 3.59 5.87 3.83 6.03 4.89 4.32 4.69
##   ..$ X: num [1:10] 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language Y ~ X
##   .. .. ..- attr(*, "variables")= language list(Y, X)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "Y" "X"
##   .. .. .. .. ..$ : chr "X"
##   .. .. ..- attr(*, "term.labels")= chr "X"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(Y, X)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "Y" "X"
##  - attr(*, "class")= chr "lm"
```

Question 2

In words, describe what the following code is doing. Note the `cars` data has 50 observations and two columns: `speed` which is the stopping speed in mph, and `dist` which is the stopping distance in feet. The data were recorded in the 1920s so the cars were pretty slow!

```
data(cars)
for(i in 1:nrow(cars)){
  if(cars$speed[i] == min(cars$speed)){
    print("Stopping distance (in feet) for slowest car(s) in the dataset")
    print(cars$dist[i])
  }
}
```

```
## [1] "Stopping distance (in feet) for slowest car(s) in the dataset"
## [1] 2
## [1] "Stopping distance (in feet) for slowest car(s) in the dataset"
## [1] 10
```

Answer:

The for loop iterates over 1, 2, 3, … 50, each time checking the speed variable in the ith position to assess if it equal to the minimum speed in the dataset. If so, it prints the stopping distance in the ith position as well.

Question 3

3a

Write a function that will take a vector vec as the only input and returns the number of elements that are NAin vec.

Answer:

```
#answer here
na_function <- function(vec){
  return(sum(is.na(vec)))
}
```

3b

Use your function to print the number of elements that are NA in the following vector.

Answer:

```
vec <- c(1, 10, NA, NA, 12)
#answer here
na_function(vec)
```

```
## [1] 2
```