STAT 2200: Problem Set 1

Jack Ambery

Due: Thursday, 2/1 at the beginning of class

- You may discuss this assignment with other students in the class, but you may not sit down and type it up with them or show them your code. You also may not discuss the assignment with anyone who isn't in our class, nor may you look up anything online.
- You must type up your homework using R Markdown. I want to see all of your code and output, and any answers you provide that aren't code must be typed above the respective `R` code chunk.
- Make sure none of your code runs off the page, otherwise you will lose points.
- You must print and turn in a PDF of your homework. I won't accept anything else (e.g., a Word document). All pages must be stapled together.
- This assignment is worth 150 points.

Part 1

Type each expression below and print the resulting value. Do not store anything.

1. $1 - 4 + 2 - 3 + 5 - 7 + 11$

```
1 - 4 + 2 - 3 + 5 - 7 + 11
```

```
## [1] 5
```

2. $\frac{4}{9}(96 - 31)$

```
(4 / 9) * (96 - 31)
```

```
## [1] 28.88889
```

3. $\frac{21,000}{1 - (1 + 0.035)^{-36}}$

```
(21000) / (1 - ((1 + 0.035) ^ -36))
```

```
## [1] 29570.5
```

4. $2,500\left(1+\dfrac{0.043}{12}\right)^{12\times 25}$

```
(2500) * ((1 + (0.043 / 12)) ^ (12 * 25))
```

```
## [1] 7310.921
```

5. $-\dfrac{1}{3}+\ln(24)+e^3\sqrt{\dfrac{3}{5}+3\times 5^{-2}}$

```
(-1/3) + (log(24)) + ((exp(3)) * (sqrt((3 / 5) + 3 * (5 ^ -2))))
```

```
## [1] 19.88786
```

Part 2

1. Create and store a vector of values starting at 5 and ending at 73, with the values increasing in increments of 4.

```r
vec1 <- seq(from = 5, to = 73, by = 4)
```

2. Add 2 to all of the values in the vector from problem 1 and store the resulting vector. Do this by referring to the vector you made in problem 1. Do not print anything.

```r
vec2 <- vec1 + 2
```

3. Remove the seventh value from the vector in problem 2 and store the new vector. Print the result.

```r
vec3 <- vec2[-7]
vec3
```

```
##  [1]  7 11 15 19 23 27 35 39 43 47 51 55 59 63 67 71 75
```

4. Add the values 25 and 31 to the end of the vector in problem 3 and store the new vector. Print the result.

```r
vec4 <- c(vec3, 25, 31)
vec4
```

```
##  [1]  7 11 15 19 23 27 35 39 43 47 51 55 59 63 67 71 75 25 31
```

5. Suppose the fourth value in the vector from problem 4 was supposed to be a 22. Make the appropriate change in the vector and print the resulting vector.

```r
vec4[4] <- 22
vec4
```

```
##  [1]  7 11 15 22 23 27 35 39 43 47 51 55 59 63 67 71 75 25 31
```

6. Sort the vector from problem 5 so that all of the values are in increasing order. Then store the new vector. Do not print anything here.

```r
vec5 <- sort(vec4)
```

7. Using an `R` function, find the length of the vector in problem 6.

```r
length(vec5)
```

```
## [1] 19
```

8. Create a new vector by repeating the number zero 40 times. Print the result. Do not store this vector.

```r
rep(0, times = 40)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [39] 0 0
```

9. Recreate the following vector using the `rep()` function: 5, 5, 5, 1, 1, 1, 4, 4, 4, 2, 2, 2, 1, 1, 1. Also store it.

```
vec6 <- c(rep(5, times = 3), rep(1, times = 3), rep(4, times = 3),
          rep(2, times = 3), rep(1, times = 3))
```

10. Combine the vectors from problems 5 and 9, and then print the result. You do not need to store anything here.

```
c(vec4, vec6)
```

```
##  [1]  7 11 15 22 23 27 35 39 43 47 51 55 59 63 67 71 75 25 31  5  5  5  1  1  1
## [26]  4  4  4  2  2  2  1  1  1
```

Part 3

1. The following are the final high school GPAs of a group of college students: 3.63, 3.81, 3.96, 4.00, 3.12, 2.86, 3.77, 2.53, 3.26, and 2.29. Create a vector that stores these values in the order specified. Choose a variable name that's descriptive but relatively short.

```
hsGPAs <- c(3.63, 3.81, 3.96, 4.00, 3.12, 2.86, 3.77, 2.53, 3.26, 2.29)
```

2. The following are the SAT quantitative scores for the same group of students: 640, 660, 790, 770, 580, 560, 670, 440, 650, and 470, respectively. Create another vector that stores these SAT scores in the order provided.

```
SATScores <- c(640, 660, 790, 770, 580, 560, 670, 440, 650, 470)
```

3. Suppose I made a mistake when recording the SAT quantitative score for the fifth student, whose actually score was a 680. Make this change using the vector you created in problem 2 and then print the resulting vector.

```
SATScores[5] <- 680
SATScores
```

```
##  [1] 640 660 790 770 680 560 670 440 650 470
```

Part 4

According to the International Monetary Fund (IMF), the countries with the ten highest gross domestic products (GDPs) in 2018, from highest to lowest, were the United States, China, Japan, Germany, the United Kingdom, France, India, Italy, Brazil, and Canada. Their GDPs (in millions of US dollars) were 20,513,000, 13,457,267, 5,070,626, 4,029,140, 2,808,899, 2,794,696, 2,689,992, 2,086,911, 1,909,386, and 1,733,706, respectively.

1. Create a variable that consists of the GDPs specified above and then print it. Use an appropriate name when storing, and be careful with commas!

```r
gdps = c(20513000, 13457267, 5070626, 4029140, 2808899, 2794696, 2689992, 2086911,
1909386, 1733706)
gdps
```

```
##  [1] 20513000 13457267  5070626  4029140  2808899  2794696  2689992  2086911
##  [9]  1909386  1733706
```

2. Create a new variable that measures the GDPs in trillions of US dollars by converting the GDPs you stored in problem 1. You must use `R` to convert the values for you (recall: `R` is a vectorized language). Then print the resulting vector.

```r
gdpsTrill = gdps / 1000000000
gdpsTrill
```

```
##  [1] 0.020513000 0.013457267 0.005070626 0.004029140 0.002808899 0.002794696
##  [7] 0.002689992 0.002086911 0.001909386 0.001733706
```

3. What were Italy's and Brazil's GDPs in 2018? Use one of the vectors you already created and subset it to print these values. You must do this in one line of code! Then type your answers (in a full sentence) above the `R` code chunk to practice typing text outside of `R` code chunks.

The GDP's of Italy and Brazil in 2018 were $2,086,911 and $1,909,386, respectively.

```r
gdps[8:9]
```

```
## [1] 2086911 1909386
```

4. What was the mean GDP (in 2018) of the ten countries listed above? Calculate this value and then type a full sentence that answers the question. This is meant to give you more practice typing text outside of `R` code chunks.

The mean GDP of the 10 largest countries in 2018 was $5,709,362.

```r
mean(gdps)
```

```
## [1] 5709362
```

5. What was the total (combined) GDP (in 2018) of the ten countries listed above? Calculate this value and then type a full sentence that answers the question. This is meant to give you even more practice typing text outside of `R` code chunks.

The total combined GDPs of the 10 largest countries in 2018 (in millions) was $57,093,623.

```r
sum(gdps)
```

```
## [1] 57093623
```

Part 5

1. Create and store a vector that consists of the following movie lengths (in minutes): 94, 109, 110, 123, 125, 108, 92, 106, 84, 119, 110, and 140.

```
durations <- c(94, 109, 110, 123, 125, 108, 92, 106, 84, 119, 110, 140)
```

2. How many movie lengths are in the vector from problem 1? Use an appropriate `R` function to determine this, and then type your answer above your code (outside of the `R` code chunk).

The given vector has the lengths of 12 movies.

```
length(durations)
```

```
## [1] 12
```

3. For each movie, determine if the movie is at least 90 minutes long. Your result should be a vector of logicals.

```
durations >= 90
```

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

4. Determine if any movies are over two hours long. Your result should involve a logical.

Three movies have a duration that is over 2 hours.

```
sum(durations > 120)
```

```
## [1] 3
```

5. Determine if all of the movies are shorter than 90 minutes. Your result should involve a logical.

```
sum(durations < 90) == length(durations)
```

```
## [1] FALSE
```

6. Print the movie lengths that are between 90 and 120 minutes, inclusive.

```
durations[durations <= 120 & durations >= 90]
```

```
## [1]  94 109 110 108  92 106 119 110
```

7. Print the indices of the movie lengths that are between 95 and 115 minutes, inclusive.

```
which(durations <= 115 & durations >= 95)
```

```
## [1]  2  3  6  8 11
```

8. Print the movie lengths that are at least 90 minutes but not equal to 110 minutes.

```
durations[durations >= 90 & durations != 110]
```

```
## [1]  94 109 123 125 108  92 106 119 140
```

9. Print the movie lengths that are either under 90 minutes or over 120 minutes.

```r
durations[durations < 90 | durations > 120]
```

```
## [1] 123 125  84 140
```

10. Print the indices of the movie lengths that are exactly either 123 or 110 minutes.

```r
which(durations == 123 | durations == 110)
```

```
## [1]  3  4 11
```