PSTAT 10: Homework 2

Solution Sketches

If we find these solutions posted to the likes of Chegg, CourseHero, etc., we will cancel all future quizzes and move the corresponding weights of your final grade solely to the final exam.

Please Note:

- There are often many ways of solving each problem. Just because the solution sketches below are different than your own personal approach doesn't mean your approach is invalid!
- Do not post these solutions anywhere (e.g. Chegg, CourseHero, etc.). If we find these solutions posted online, we will change the grading structure of the course to put 100% weight on the final exam.

Instructions

- Remember to always copy the necessary files to your working directory; never edit and/or knit the files directly in the `10f22-content` folder.
- Rename your `.Rmd` file as `hw02-your-netid.Rmd`. For example, my Rmd script file will be `hw02-umaravat.Rmd`.
- Make sure to include your identifying details in the Rmd file.
- **Please keep in mind that there are 5 additional Multiple Choice questions that appear on Canvas**, in the submission portal for this homework. These questions are still considered a part of the homework, and you must complete them in order to be eligible to earn full points.
- When submitting to Canvas, follow the provided submission instructions carefully.

Exercises

Exercise 0: YAML

Update the YAML to reflect your own information (i.e. name, netid, etc.).

Exercise 0.1: Installation of Requisite Package

Before knitting this document, you will need to install the package `kableExtra`.

Exercise 1: Thinking Critically About Data

As a Data Scientist, you may find yourself needing to code data into `R` manually (as opposed to simply reading in a file). In such cases, it is important to think critically about the data, and exactly *how* you want to encode it!

As a simple example, we will consider the following dataset:

Name	Major	Midterm Score (in %)	Final Letter Grade
Brad	Statistics	99	A+
Beckett	Statistics	60	B-
Samanthan	Engineering	90	A
D'Vana	Biology	99	A+
Carol	Biology	97	A

We will work toward inputting it as a data frame, but will do so by carefully considering each variable separately.

(a) What Data Type should we use to encode the names of the individuals in the dataset? *No coding necessary for this part.*

Solutions:

It is ideal to use the data type `character` (or `string`) to encode names.

(b) Now, create a vector called `name` that stores the names of the individuals in the dataset.

Solutions:

```
name <- c("Brad", "Beckett", "Samanthan", "D'Vana", "Carol")
```

Though this wasn't required, we can check the data type using `typeof()`:

```
typeof(name)
```

```
## [1] "character"
```

(c) What Data Type should we use to encode the majors of the individuals in the dataset? *No coding necessary for this part.*

Solutions:

Because there are several people with the same major, it is best to store majors in a `factor` data *structure*, which would correspond to an `integer` data *type.*

(d) Now, create a vector called `major` that stores the majors of the individuals in the dataset.

Solutions:

```
major <- factor(c("Statistics", "Statistics", "Engineering", "Biology", "Biology"))

## check data structure:
class(major)
```

```
## [1] "factor"
```

```
## check data type:
typeof(major)
```

```
## [1] "integer"
```

(e) What Data Type should we use to encode the midterm scores of the individuals in the dataset? *No coding necessary for this part.*

Solutions:

Scores can be any numerical value, so we should use the `double` data type when storing exam scores.

(f) Now, create a vector called `mt_scores` that stores the midterm scores of the individuals in the dataset.

Solutions:

```
mt_scores <- c(99, 60, 90, 99, 97)

## check the data type:
typeof(mt_scores)
```

```
## [1] "double"
```

(g) What Data Type should we use to encode the final letter grades of the individuals in the dataset? *No coding necessary for this part.*

Solutions:

The data *structure* we should use is that of an ordered factor, which gives rise to an integer data *type*.

(h) Now, create a vector called `final_grades` that stores the final letter grades of the individuals in the dataset.

Solutions:

```
final_grades <- factor(c("A+", "B-", "A", "A+", "A" ),
                       ordered = TRUE,
                       levels = c("B-", "A", "A+"))
```

(i) Now, it's time to put everything together! Using the variables you created in the above parts, encode the full dataset into a data frame called `df_dataset`

Solutions:

```
df_dataset <- data.frame(
  name,
  major,
  mt_scores,
  final_grades
)

df_dataset
```

```
##        name       major mt_scores final_grades
## 1      Brad  Statistics        99           A+
## 2   Beckett  Statistics        60           B-
## 3 Samanthan Engineering        90            A
## 4    D'Vana     Biology        99           A+
## 5     Carol     Biology        97            A
```

(j) Suppose that Brad has purple hair; Beckett, Samanthan, and Carol have black hair; and D'Vana has green hair. Add a new column to `df_dataset` called `hair_color` containing this new information. Print `df_dataset` to ensure the column has successfully been added.

Solutions:

The easiest way to add a new column, along with its name, to a data frame is to simply use the `$` operator and pass in the column name you want, followed by `=` and then the values you want stored.

```
df_dataset$hair_color <- factor(c("purple", "black", "black", "green", "black"))
df_dataset
```

```
##         name       major mt_scores final_grades hair_color
## 1       Brad  Statistics        99           A+     purple
## 2    Beckett  Statistics        60           B-      black
## 3 Samanthan Engineering        90            A      black
## 4     D'Vana     Biology        99           A+      green
## 5      Carol     Biology        97            A      black
```

Exercise 2: Hairy Situ-eye-tion

Consider the data in the following table:

		Eye Color			
		BROWN	**BLUE**	**HAZEL**	**GREEN**
Hair Color	**BLACK**	32	11	10	3
Hair Color	**BROWN**	53	50	25	15
Hair Color	**BLONDE**	3	30	5	8

(a) Construct a matrix using the data given in the table above, and assign it to the variable `hair_eye`. Fill in the matrix by rows.

Solutions:

```
value <- c(32, 11, 10, 3, 53, 50, 25, 15, 3, 30, 5, 8)
hair_eye <- matrix(value, 3, 4)
hair_eye
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   32    3   25   30
## [2,]   11   53   15    5
## [3,]   10   50    3    8
```

(b) Add appropriate row names and column names, and assign this named matrix to the variable `hair_eye_named`.

Solutions:

```
hair_eye_named <- matrix(value, 3, 4,
             dimnames = list(c("BLACK", "BROWN", "BLONDE"),
                             c("BROWN", "BLUE", "HAZEL", "GREEN")))
hair_eye_named
```

```
##        BROWN BLUE HAZEL GREEN
## BLACK     32    3    25    30
## BROWN     11   53    15     5
## BLONDE    10   50     3     8
```

(c) Determine how many people have blonde hair and blue eyes

Solutions:

```
hair_eye_named["BLONDE", "BLUE"]
```

```
## [1] 50
```

(d) Compute the row-sums and column-sums of `hair_eye_named` , (*Hint:* Check out the `rowSums` function in `R`) and interpret your results. **In words**, what do the row-sums tell you? What about the column-sumns?

Solutions:

```
rowSums(hair_eye_named)
```

```
##  BLACK  BROWN BLONDE
##     90     84     71
```

```
colSums(hair_eye_named)
```

```
## BROWN  BLUE HAZEL GREEN
##    53   106    43    43
```

The row-sums tell us the total number of people possessing each hair color; i.e. 90 people have black hair, 84 have brown, and 71 have blonde. Similarly, the column-sums tell us the total number of people possessing each eye color; i.e. 53 people have brown eyes, 106 have blue, 43 have hazel, and 43 have green.

(e) Remove the column of data containing the hair colors of people with blue eyes. (After doing so, `hair_eye_named` should have one fewer columns)

Solutions:

```
hair_eye_named <- hair_eye_named[ ,-2]
hair_eye_named
```

```
##        BROWN HAZEL GREEN
## BLACK     32    25    30
## BROWN     11    15     5
## BLONDE    10     3     8
```

(f) It is found that there was a mistake when the data was collected: of the individuals with green eyes, 5 of them had black hair, 20 of them had brown hair, and 10 of then had blonde hair. Update your matrix `hair_eye` to reflect the corrected information.

Solutions:

```
hair_eye_named[ ,3] <- c(5, 20, 10)
hair_eye_named
```

```
##        BROWN HAZEL GREEN
## BLACK     32    25     5
## BROWN     11    15    20
## BLONDE    10     3    10
```

Exercise 3: Homework *can* be Mean

In PSTAT 3029 (which doesn't exist… yet), the 10 homework scores for 3 students (named "Student 1", "Student 2", and "Student 3") are given as:

```
hw_score <- list(
                  s1 = runif(10, 50, 100),
                  s2 = runif(10, 50, 100),
                  s3 = runif(10, 50, 100))
hw_score
```

```
## $s1
##  [1] 88.54378 53.21510 58.07685 65.36196 75.48536 52.28632 78.61486 57.72672
##  [9] 80.93167 78.75078
##
## $s2
##  [1] 80.51773 65.78335 69.05191 50.77567 56.50593 68.02399 64.76794 64.93893
##  [9] 70.85745 85.80403
##
## $s3
##  [1] 92.12253 97.66393 78.79373 99.23355 73.20642 66.93426 54.73380 92.93503
##  [9] 67.65334 87.67131
```

(a) Retrieve the exam scores of Student 2 in two ways

Solutions:

```
hw_score$s2
```

```
##  [1] 80.51773 65.78335 69.05191 50.77567 56.50593 68.02399 64.76794 64.93893
##  [9] 70.85745 85.80403
```

```
hw_score[[2]]
```

```
##  [1] 80.51773 65.78335 69.05191 50.77567 56.50593 68.02399 64.76794 64.93893
##  [9] 70.85745 85.80403
```

(b) Describe the type of object returned by `hw_score[1]`

Solutions:

```
hw_score[1]
```

```
## $s1
##  [1] 88.54378 53.21510 58.07685 65.36196 75.48536 52.28632 78.61486 57.72672
##  [9] 80.93167 78.75078
```

```
typeof(hw_score[1])
```

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

(c) Create a list `hw_score_average` that holds each students average hw score as its members. Name the averages as `ave_s1` , `ave_s2` , `ave_s3` .

Solutions:

```
hw_average <- list(ave_s1 = mean(hw_score[[1]]),
                    ave_s2 = mean(hw_score$s2),
                    ave_s3 = mean(hw_score$s3))
hw_average
```

```
## $ave_s1
## [1] 68.89934
##
## $ave_s2
## [1] 67.7027
##
## $ave_s3
## [1] 81.09479
```

(d) What is the overall average hw score?

Solutions:

```
mean(c(hw_average[[1]],hw_average[[2]],hw_average[[3]]))
```

```
## [1] 72.56561
```

Exercise 4: Splitting Hairs… Again!

Recall that in homework 1 we examined the `strsplit()` function. In this problem, we investigate it a bit further. Additionally, for this problem it is important you perform the parts in order.

(a) Create a vector `x` and assign it the value `c("hair", "follicle")`.

Solutions:

```
x <- c("hair", "follicle")
```

(b) What is the data type of `x`?

Solutions:

```
typeof(x)
```

```
## [1] "character"
```

(c) Define `y <- strsplit(x, "i")`. What data structure does `x` possess? **Correction:** This was meant to ask about the data structure of `y`.

Solutions:

```
y <- strsplit(x, "i")
class(y)
```

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

(d) What data structure does `y[[1]]` possess?

Solutions:

```
class(y[[1]])
```

```
## [1] "character"
```

(e) Use subsetting on `y` to write a single line of code that returns the string `"foll"`. Don't just print `"foll"`!

Solutions:

```
y[[2]][1]
```

```
## [1] "foll"
```

Exercise 5: More Palm-y Weather!

Recall the `palm.csv` dataset from homework 1.

(a) Copy the dataset from your HW01 `data` folder into a new folder (also called `data`) in your HW02 subdirectory.

Solutions:

```
# Done
```

(b) Load the data into your workspace, and store the resulting data in a variable called `palm_2`.

Solutions:

```
palm_2 <- read.csv("./data/palm.csv")
```

(c) Some of you noticed (on HW01) that there was no unique city block that contains the maximum number of King Palms. Run code to list **all** of the city blocks that contain the maximum number of King Palms represented in the dataset.

Solutions:

```
which(palm_2$KP == max(palm_2$KP))
```

```
##  [1]   3   6  10  14  16  30  31  33  36  37  47  53  54  64  65  74  77  82  84
## [20]  89  98 109 118 124 126 134 144 152 153 179 187 218
```

(d) Do any city blocks contain more than 8 Foxtail Palms? **Answer this question in TWO ways, using two DIFFERENT lines of code.**

Solutions:

```
any(palm_2$FP > 8)
```

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

```
which(palm_2$FP > 8) # this is nonempty, so the answer is 'yes'
```

```
##  [1]  10  14  18  21  23  25  28  30  34  37  39  42  44  51  54  55  62  66  67
## [20]  68  69  72  79  87  89  92  93  95  96 100 102 107 118 123 127 130 136 144
## [39] 152 157 162 179 185 186 189 191 192 193 212 213 215 226 231 232
```