Assignment #4

MSAN 593 - Summer 2018

***DUE**: Tuesday, August 14, 2018, 23:59*

Instructions

Upload **both** your `*.Rmd` file as well as the knitted `pdf` to Canvas by the due date and time. Late submissions will receive a grade of zero.

1. This homework is intended to be completed and submitted individually.
2. All code should be commented in a neat, concise fashion, explaining the objective(s) of individual lines of code.
3. When making reference(s) to *summary* results, include all relevant output in text of the deliverable where it is being discussed, not in an appendix at the back of the deliverable.
4. Do not include a copy of the raw data in the body of the deliverable unless there is a compelling reason.
5. `R` can generate hundreds of graphs and statistical output extremely easily. Only include *relevant* graphs and output in the deliverable. All graphs and statistical output included in the deliverable should be referenced in the text of the deliverable.
6. **There should be no orphaned figures or graphs**. Everything should be orderly and easy for a grader to read.
7. All code should be visible in the `pdf` version of the homework, i.e., for each code chunk, be sure to set `echo = TRUE`
8. Homework **may not be emailed to the instructor**.

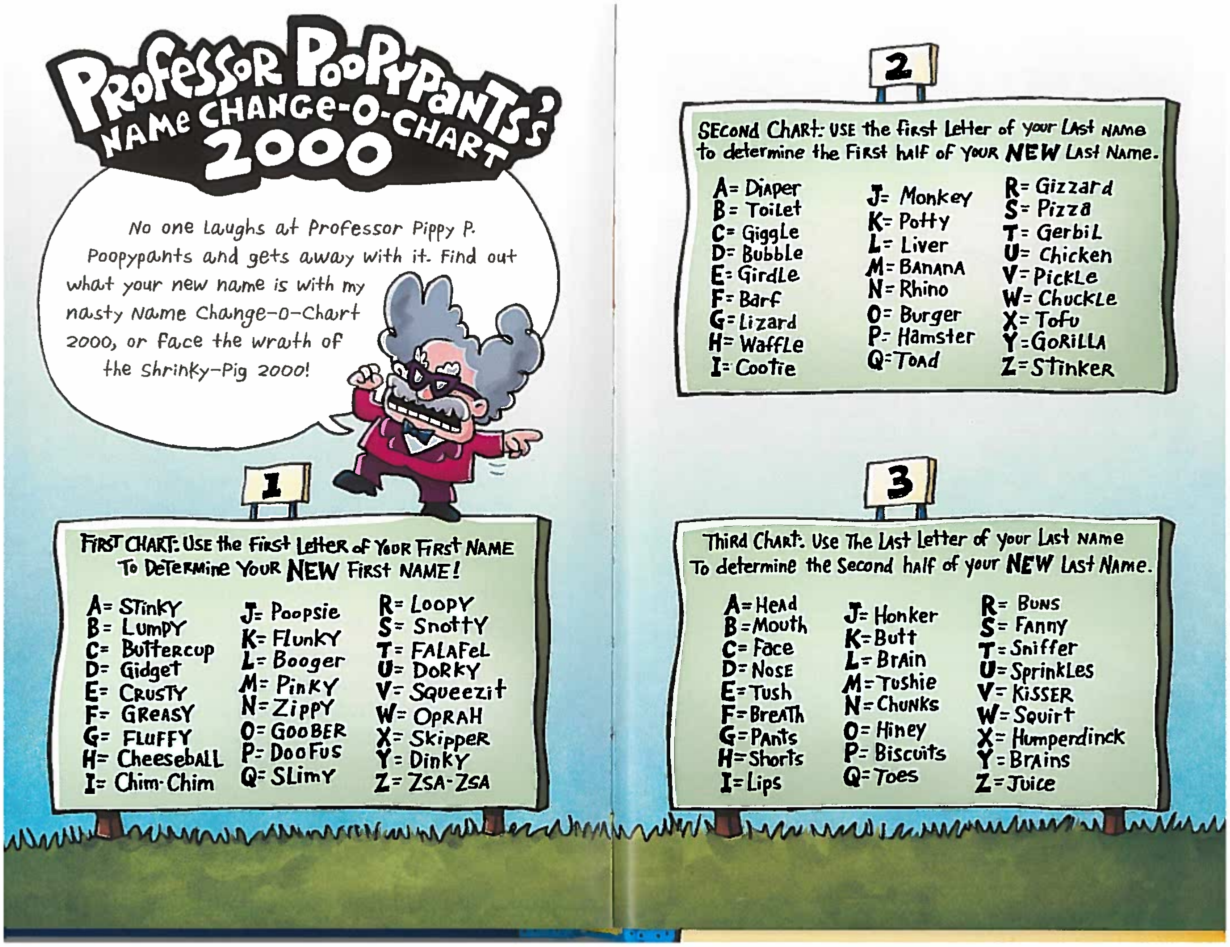
Question 1

Volume Four of the Captain Underpants series of children's books, written by Dav Pilkey, features an evil villain named Professor Pippy P. Poopypants from New Swissland. Although, in Volume 9, he ultimately changes his name to Tippy Tinkletrowsers to avoid being ridiculed (unsucesfully), his first attempt to keep people from making fun of his name is to force eveyone to change their name according to his Name Change-O-Chart 2000 (see Figure 1).

Your job is to be one of Professor Pippy P. Poopypants's minions, and code up a function which will automatically execute the name change. Your function should take a single string as input. The singular string should include only a first name and last name separated by *at least one space*. The output should print the new name to the console. Your function should be robust enough to parse through input which does not conform, e.g., including a middle (third) name, names beginning or ending with non-alpha characters, etc., and spit out a *generic* error message when the input is non-conforming.

Once your function is ready, print your function output in the submitted homework by passing the following arguments:

1. `<your name>`
2. Paul Intrevado
3. David Uminksy
4. Terence Parr


Professor Poopypants's Name Change-O-Chart 2000
No one laughs at Professor Pippy P. Poopypants and gets away with it. Find out what your new name is with my nasty Name Change-O-Chart 2000, or face the wrath of the Shrinky-Pig 2000!
1
2
3

First Chart: Use the first letter of your first name to determine your NEW first name!

A= Stinky
B= Lumpy
C= Buttercup
D= Gidget
E= Crusty
F= Greasy
G= Fluffy
H= Cheeseball
I= Chim-Chim
J= Poopsie
K= Flunky
L= Booger
M= Pinky
N= Zippy
O= Goober
P= Doofus
Q= Slimy
R= Loopy
S= Snotty
T= Falafel
U= Dorky
V= Squeezit
W= Oprah
X= Skipper
Y= Dinky
Z= Zsa-Zsa

Second Chart: Use the first letter of your last name to determine the first half of your NEW last name.

A= Diaper
B= Toilet
C= Giggle
D= Bubble
E= Girdle
F= Barf
G= Lizard
H= Waffle
I= Cootie
J= Monkey
K= Potty
L= Liver
M= Banana
N= Rhino
O= Burger
P= Hamster
Q= Toad
R= Gizzard
S= Pizza
T= Gerbil
U= Chicken
V= Pickle
W= Chuckle
X= Tofu
Y= Gorilla
Z= Stinker

Third Chart: Use the last letter of your last name to determine the second half of your NEW last name.

A= Head
B= Mouth
C= Face
D= Nose
E= Tush
F= Breath
G= Pants
H= Shorts
I= Lips
J= Honker
K= Butt
L= Brain
M= Tushie
N= Chunks
O= Hiney
P= Biscuits
Q= Toes
R= Buns
S= Fanny
T= Sniffer
U= Sprinkles
V= Kisser
W= Squirt
X= Humperdinck
Y= Brains
Z= Juice

Figure 1: Figure 1: Name Change-O-Chart 2000

5. Jeff Hamrick
6. paul intrevado
7. Intrevado, Paul
8. Intrevad0 Paul
9. Queen Elizabeth II
10. Queen Elizabeth 2nd
11. Queen Elizabeth 2
12. John Paul Euclid Rumpel
13. britishDudeThatSitsInTheBackOfTheClass

Question 2

Your objective is to code a k-means algorithm from scratch. Your function will take the following arguments:

- `myScatterInput`: a data frame with `n` rows and `m` columns, where all entries will be real numbers
  - `myScatterInput` $\in \mathbb{R}^m$
- `myClusterNum`: an integer value greater than or equal to 2 but less than or equal to n
  - $2 <$ `myClusterNum` $< n$: `myClusterNum` $\in \mathbb{Z}$

Your function should:

1. Randomly assign each of the points in `myScatterInput` to some value between 1 and `myClusterNum`. This is the initial, random assignment of points to each cluster.

2. Compute the *cluster centroid* for each cluster 1 to `myClusterNum`.

3. Compute the Euclidean distance from each *cluster centroid* to **each data point**.

4. Assign each point to the *cluster centroid* which minimizes the Euclidean distance.

5. Repeat steps 2, 3 and 4 until one of two stopping conditions are met

   - subsequent cluster assignments are unchanged
   - you have repeated steps 2, 3 and 4 `maxIter` number of times

6. Once you have reached a terminating condition, compute the sum of all Euclidean distances from each point to their respective centroids.

7. Repeat steps 1-6 100 times.

8. Identify the replication with the lowest sum of Euclidean distances from points to centroids as your best result and print the value to the console.

9. **IF** the data frame provided to you has two dimensions, generate a 2-dimensional scatter plot of the data, plotting and coloring all points based on the cluster they are in, i.e., all points associated with a certain cluster should all be the same color in the scatter plot.

n.b.

- You are not permitted to use any base `R` functions or packages to call an existing k-means algorithm

- You may use base `R` functions or packages to compute distances

- Use can use subsets of following data frame to test your code (use smaller subsets of both rows and columns, and grow to the full data frame, varying k):

```r
set.seed(101)
myDF <- as.data.frame(matrix(rnorm(10000), ncol = 5))
myDF[2] <- myDF[2] + 5
myDF[3] <- myDF[3] - 0.1
myDF[4] <- myDF[4] + 1
myDF[5] <- myDF[5] - 3
```

- At 17:00 on Monday, August 13th, I will post code on the Slack channel to generate various data sets, which I want you to test your code on and report back processing time in your homework. Students with the fatest processing time for each individual test data set, as well as the student with the fatest aggregate processing time will receive bonus marks on the homework.