STAT 231: Problem Set 8A

Jack Dove

due by 5 PM on Monday, November 2

In order to most effectively digest the textbook chapter readings – and the new R commands each presents – series A homework assignments are designed to encourage you to read the textbook chapters actively and in line with the textbook's Prop Tip of page 33:

"**Pro Tip**: If you want to learn how to use a particular command, we highly recommend running the example code on your own"

A more thorough reading and light practice of the textbook chapter prior to class allows us to dive quicker and deeper into the topics and commands during class. Furthermore, learning a programming lanugage is like learning any other language – practice, practice, practice is the key to fluency. By having two assignments each week, I hope to encourage practice throughout the week. A little coding each day will take you a long way!

Series A assignments are intended to be completed individually. While most of our work in this class will be collaborative, it is important each individual completes the active readings. The problems should be straightforward based on the textbook readings, but if you have any questions, feel free to ask me!

Steps to proceed:

1. In RStudio, go to File > Open Project, navigate to the folder with the course-content repo, select the course-content project (course-content.Rproj), and click "Open"
2. Pull the course-content repo (e.g. using the blue-ish down arrow in the Git tab in upper right window)
3. Copy ps8A.Rmd from the course repo to your repo (see page 6 of the GitHub Classroom Guide for Stat231 if needed)
4. Close the course-content repo project in RStudio
5. Open YOUR repo project in RStudio
6. In the ps8A.Rmd file in YOUR repo, replace "YOUR NAME HERE" with your name
7. Add in your responses, committing and pushing to YOUR repo in appropriate places along the way
8. Run "Knit PDF"
9. Upload the pdf to Gradescope. Don't forget to select which of your pages are associated with each problem. *You will not get credit for work on unassigned pages (e.g., if you only selected the first page but your solution spans two pages, you would lose points for any part on the second page that the grader can't see).*

1. k-means clustering

Section 9.1.2 walks through an example of how k-means clustering can identify genuine patterns in data – in this case, clustering cities into continental groups merely based on city location (latitutde and longitude coordinates). The code below functions similarly to the code provided in this section of the textbook, but uses slightly different syntax. It also adds the centroid locations to a reproduction of Figure 9.4.

(a) Walk through the code and comments below to understand what the code is doing. How many cities were identified as belonging to cluster 1? What does cluster 1 represent?

ANSWER: 726 cities are in cluster 1, which represents a group of cities with similar longitudes and latitudes (in this case, near India and Central Asia). (Longitude is about 40-100 degrees, Latitude is about 0-60 degrees).

```r
# the code in the textbook loads WorldCities, but the dataset name has since
# been updated to world_cities
# (world_cities is a dataset provided in the mdsr package)
library(mdsr)
data(world_cities)

# identify the 4,000 biggest cities in the world
BigCities <- world_cities %>%
  arrange(desc(population)) %>%
  head(4000) %>%
  select(latitude, longitude)

# apply the k-means algorithm (set the seed to make the results reproducible)
set.seed(15)
km_out <- kmeans(BigCities, centers = 6, nstart = 20)

# km_out is of class "kmeans"; it's a kmeans object
class(km_out)
```

```
## [1] "kmeans"
```

```r
# you can use the syntax km_out$name to refer to any of the named elements below
names(km_out)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# for example, let's see what the cluster sizes are (how many cities are in each cluster)
km_out$size
```

```
## [1] 726 554 984 392 356 988
```

```r
# for another example, let's see where the centroids are
km_out$centers
```

```
##    latitude longitude
## 1  27.43226  75.14407
## 2  31.07927 -94.47442
## 3  23.72534 120.38618
## 4 -15.21344 -57.10048
## 5  -1.12440  18.91076
## 6  45.37853  18.77544
```

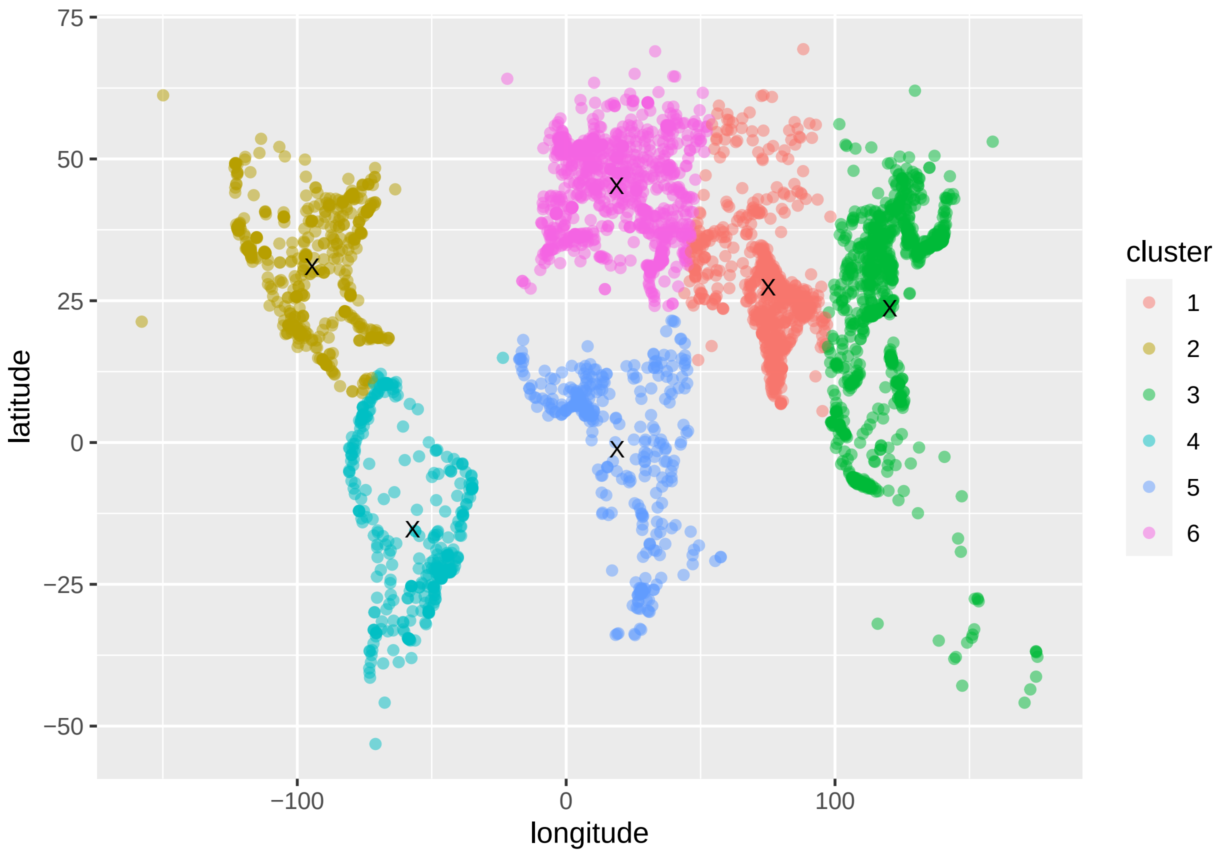
```
BigCities_km <- BigCities %>%
  mutate(cluster = as.character(km_out$cluster))

mosaic::tally(~cluster, data=BigCities_km)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
## cluster
##   1   2   3   4   5   6
## 726 554 984 392 356 988
```

```
ggplot(data = BigCities_km, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster), alpha = 0.5) +
  # add centroids to plot
  geom_point(data = as.data.frame(km_out$centers), shape = "X", size = 3)
```



(b) In k-means clustering, the analyst specifies the number of clusters to create. Update the `center` argument within the `kmeans` function to identify 3 clusters instead of 6. Create a plot like the one above, but coloring the points by these new cluster assignments. How many cities are in cluster 1 now? What does cluster 1 represent now?

ANSWER: 1466 cities fall in cluster 1, which now represents cities with low magnitudes of longitude (very central on the world map, falling typically in Africa and Europe). (Longitude is about -25-75 degrees, Latitude is about -30-70 degrees).

```
set.seed(15)
km_out <- kmeans(BigCities, centers = 3, nstart = 20)

# km_out is of class "kmeans"; it's a kmeans object
class(km_out)
```

```
## [1] "kmeans"
```

```
# you can use the syntax km_out$name to refer to any of the named elements below
names(km_out)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# for example, let's see what the cluster sizes are (how many cities are in each cluster)
km_out$size
```

```
## [1] 1466  945 1589
```

```
# for another example, let's see where the centroids are
km_out$centers
```

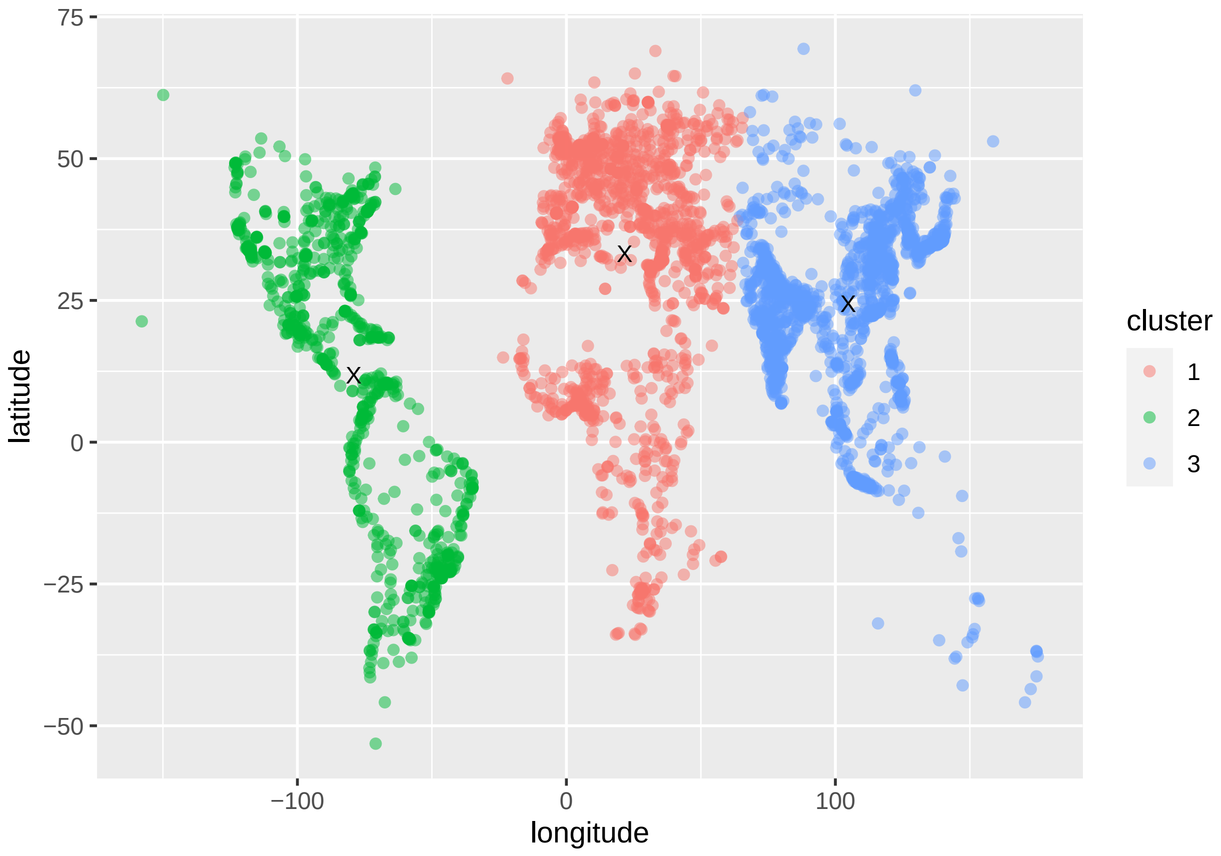
```
##   latitude longitude
## 1 33.29923  21.65145
## 2 11.89345 -79.04625
## 3 24.47673 104.80329
```

```
BigCities_km <- BigCities %>%
  mutate(cluster = as.character(km_out$cluster))

mosaic::tally(~cluster, data=BigCities_km)
```

```
## cluster
##    1    2    3
## 1466  945 1589
```

```
ggplot(data = BigCities_km, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster), alpha = 0.5) +
  # add centroids to plot
  geom_point(data = as.data.frame(km_out$centers), shape = "X", size = 3)
```



(c) Lastly, update the `center` argument within the `kmeans` function to identify 15 clusters. Create a plot like the one above, but coloring the points by these new cluster assignments. How many cities are in cluster 1 now? What does cluster 1 represent now?

ANSWER: 593 cities fall in this new cluster 1, which represents cities in India and a few parts of Southern-Central Asia. (Longitude is about 50-100 degrees, Latitude is about 0-60 degrees).

```r
set.seed(15)
km_out <- kmeans(BigCities, centers = 15, nstart = 20)

# km_out is of class "kmeans"; it's a kmeans object
class(km_out)
```

```
## [1] "kmeans"
```

```r
# you can use the syntax km_out$name to refer to any of the named elements below
names(km_out)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# for example, let's see what the cluster sizes are (how many cities are in each cluster)
km_out$size
```

```
##  [1] 593 274 225 284 280 322 274 121 192 326 195 410 186 176 142
```

```
# for another example, let's see where the centroids are
km_out$centers
```

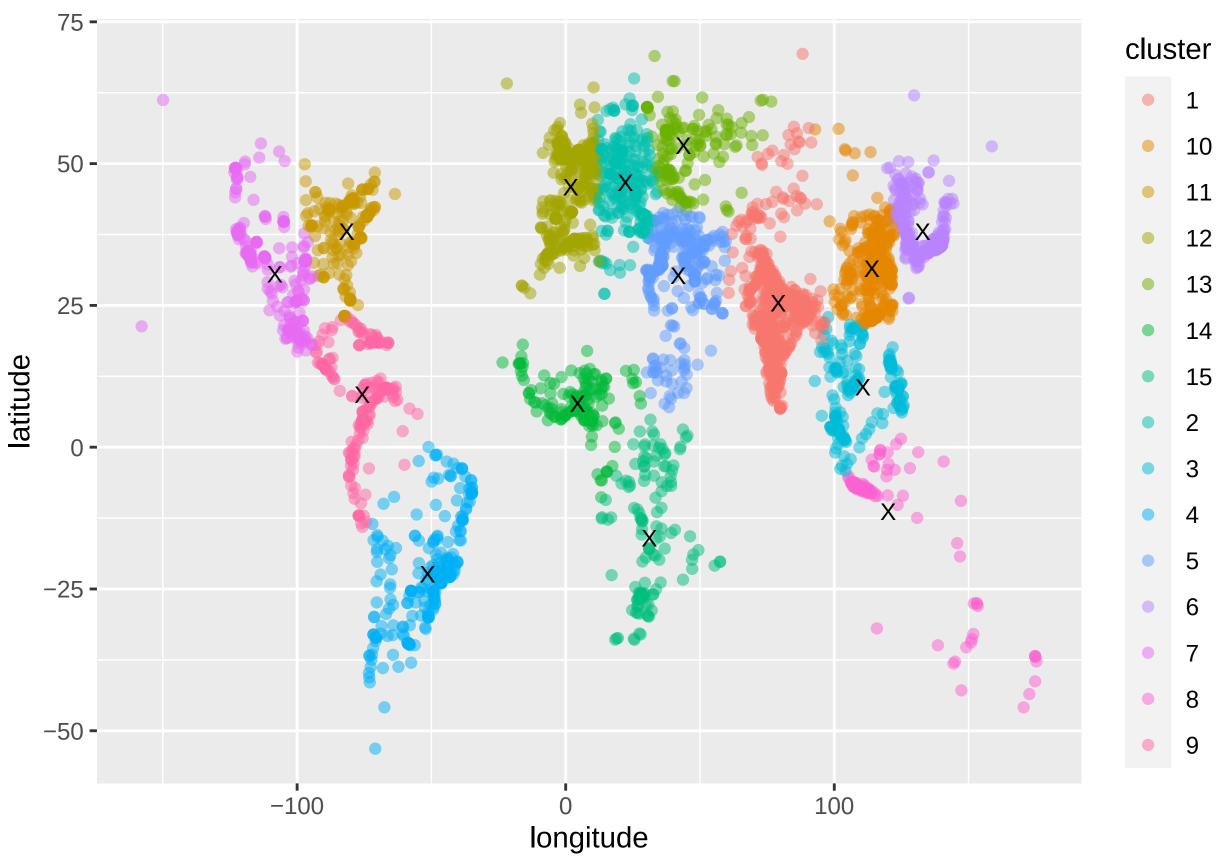
```
##      latitude   longitude
## 1   25.421427   79.122413
## 2   46.685612   22.213228
## 3   10.605780  110.662439
## 4  -22.343210  -51.401047
## 5   30.283121   41.882381
## 6   38.076731  132.953864
## 7   30.551731 -108.242820
## 8  -11.298083  120.102443
## 9    9.324475  -75.745889
## 10  31.558098  114.026512
## 11  38.068270  -81.533665
## 12  45.911377    1.860960
## 13  53.200804   43.854819
## 14   7.688174    4.437656
## 15 -16.027181   31.155775
```

```
BigCities_km <- BigCities %>%
  mutate(cluster = as.character(km_out$cluster))

mosaic::tally(~cluster, data=BigCities_km)
```

```
## cluster
##   1  10  11  12  13  14  15   2   3   4   5   6   7   8   9
## 593 326 195 410 186 176 142 274 225 284 280 322 274 121 192
```

```
ggplot(data = BigCities_km, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster), alpha = 0.5) +
  # add centroids to plot
  geom_point(data = as.data.frame(km_out$centers), shape = "X", size = 3)
```



75
50
25
0
−25
−50
latitude
−100
0
100
longitude
cluster
1
10
11
12
13
14
15
2
3
4
5
6
7
8
9