Reading Set 7

Kauila

Due by 10pm ET on Monday

Reading Set Information

A more thorough reading and light practice of the textbook reading prior to class allows us to jump into things more quickly in class and dive deeper into topics. As you actively read the textbook, you will work through the Reading Sets to help you engage with the new concepts and skills, often by replicating on your own the examples covered in the book.

These should be completed on your own without help from your peers. 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!

GitHub Workflow

1. Before editing this file, verify you are working on the copy saved in *your* repo for the course (check the filepath and the project name in the top right corner).
2. Before editing this file, make an initial commit of the file to your repo to add your copy of the problem set.
3. Change your name at the top of the file and get started!
4. You should *save, knit, and commit* the .Rmd file each time you've finished a question, if not more often.
5. You should occasionally *push* the updated version of the .Rmd file back onto GitHub. When you are ready to push, you can click on the Git pane and then click **Push**. You can also do this after each commit in RStudio by clicking **Push** in the top right of the *Commit* pop-up window.
6. When you think you are done with the assignment, save the pdf as "*Name_thisfilename_date*.pdf" (it's okay to leave out the date if you don't need it) before committing and pushing (this is generally good practice but also helps me in those times where I need to download all student homework files).

Gradescope Upload

For each question (e.g., 3.1), allocate all pages associated with the specific question. If your work for a question runs onto a page that you did not select, you may not get credit for the work. If you do not allocate *any* pages when you upload your pdf, you may get a zero for the assignment.

You can resubmit your work as many times as you want before the deadline, so you should not wait until the last minute to submit some version of your work. Unexpected delays/crises that occur on the day the assignment is due do not warrant extensions (please submit whatever you have done to receive partial credit).

Problem 1 **k-means clustering** Section 12.1.2 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 (longitude and latitude coordinates). The textbook code is modified below to parse out some of the steps for using the `kmeans()` function of the **mclust** package and to add the centroid locations to a reproduction of Figure 12.4. *Note*: the data for this example comes from the **mdsr** package.

Problem 2 Run the code below to implement the k-means algorithm. *Note*: The k-means clustering algorithm starts by choosing k points at random as initial guesses of where the cluster centroids might be. The `nstart` option specifies how many different configurations of initial guesses we want the algorithm to try, and the algorithm reports the results from the best initial configuration. It is generally recommended to specify `nstart` with a large value (e.g., 20 or 50). Take a look at the resulting output for each object in the code chunk in some way (print in the console, `glimps()`, or `head()`).

```
data(world_cities)

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

big_cities2 <- world_cities %>%
  arrange(desc(population)) %>%
  head(4000) %>%
  select(longitude, latitude)

big_cities3 <- world_cities %>%
  arrange(desc(population)) %>%
  head(4000) %>%
  select(longitude, latitude)

# Make sure to set seed for reproducibility!
set.seed(15)
city_kmeans_results <- big_cities %>%
  kmeans(centers = 6, nstart = 30)

set.seed(15)
city_kmeans_results2 <- big_cities2 %>%
  kmeans(centers = 3, nstart = 30)

set.seed(15)
city_kmeans_results3 <- big_cities3 %>%
  kmeans(centers = 15, nstart = 30)
```

2.1 The textbook code skips over much of the important output we get from running the `kmeans()` function (*note*: it's not clear that the fitted clusters are worth grabbing—we can get assigned clusters from the `kmeans()` output already!). The code below reports on the class of the `city_kmeans_results` and the named elements we can refer to and select from that class. What type of object is `city_kmeans_results`? What named elements does the object contain and what information does each element give us? *Hint*: the **Value** section of the help documention will help you understand what each named element is.

The class is kmeans.

"kmeans returns an object of class"kmeans" which has a print and a fitted method. It is a list with at least the following components:

cluster A vector of integers (from 1:k) indicating the cluster to which each point is allocated.

centers A matrix of cluster centres.

totss
The total sum of squares.

withinss
Vector of within-cluster sum of squares, one component per cluster.

tot.withinss
Total within-cluster sum of squares, i.e. sum(withinss).

betweenss
The between-cluster sum of squares, i.e. totss-tot.withinss.

size
The number of points in each cluster.

iter
The number of (outer) iterations.

ifault
integer: indicator of a possible algorithm problem – for experts." - taken directly from `?kmeans` in rstudio.

```
# Check object class or type
class(city_kmeans_results)
```

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

```
# Check named elements of object
names(city_kmeans_results)
```

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

2.2 Where are the estimated centroids of the 6 clusters? How many cities were assigned to the first cluster? Which cluster contains the most cities?

The estimaded centroids of the 6 clusters are the continents. 726 cities were assigned to the first cluster, but the 6th cluster contains the most cities with 988.

```
# Centroids
city_kmeans_results$centers
```

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

```
# Cluster sizes
city_kmeans_results$size
```

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

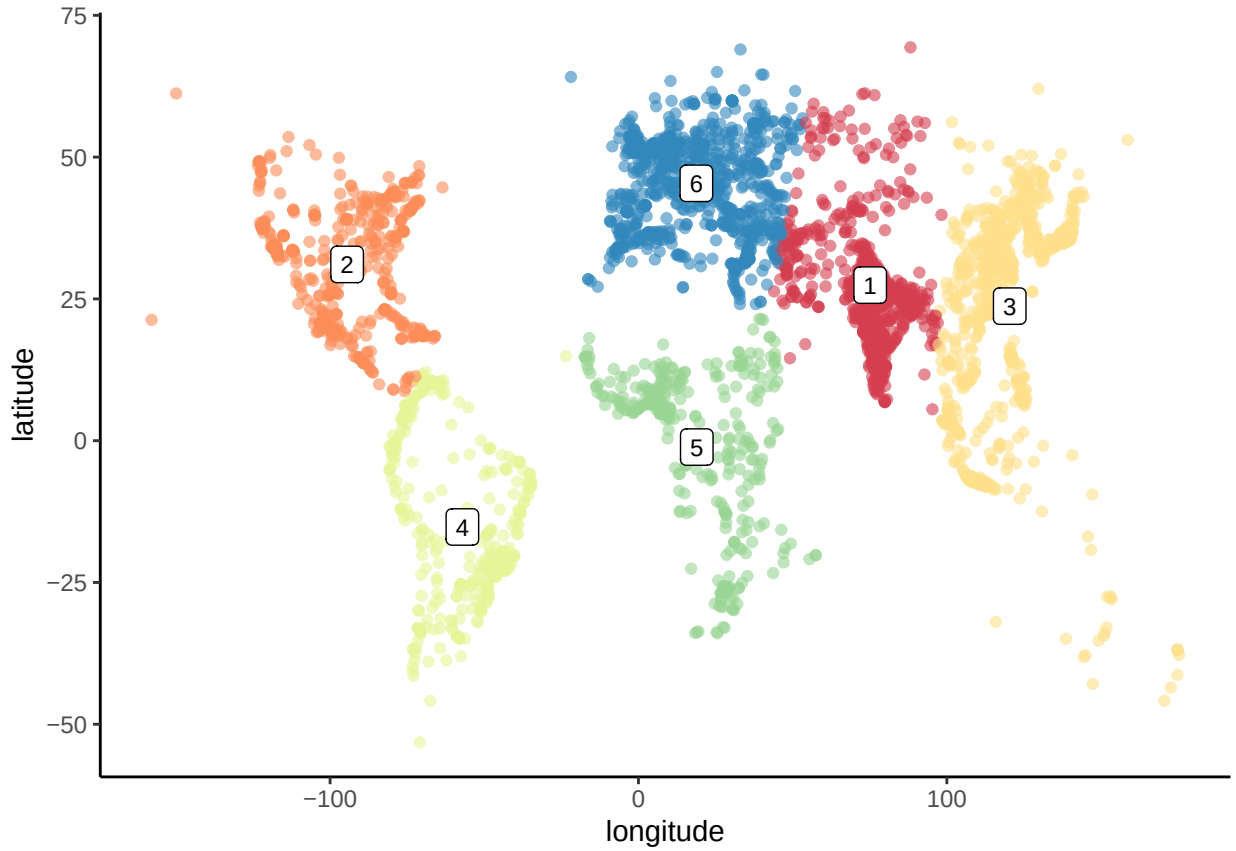
2.3 Run the code below to create a plot of the assigned clusters. What do you think Cluster 2 represents?

Cluster 2 represents North and Central America.

```
# Join cluster assignments with original dataset
big_cities <- big_cities %>%
  add_column(cluster = factor(city_kmeans_results$cluster))

# Make dataframe out of estimated cluster centroids
city_cluster_centers <- city_kmeans_results$centers %>%
  data.frame() %>%
  add_column(cluster_number = 1:6)

# Plot cluster assignments and centroids
ggplot(big_cities, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster), alpha = 0.6)  +
  scale_color_brewer(palette = "Spectral") +
  geom_label(data = city_cluster_centers,
             aes(label = cluster_number),
             size = 3) +
  theme(legend.position = "none")
```



2.4 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 2 represent now?

> There are 1,466 cities in cluster 1 now. Cluster 2 now represents North, Central and South America.

```
city_kmeans_results2$size
```

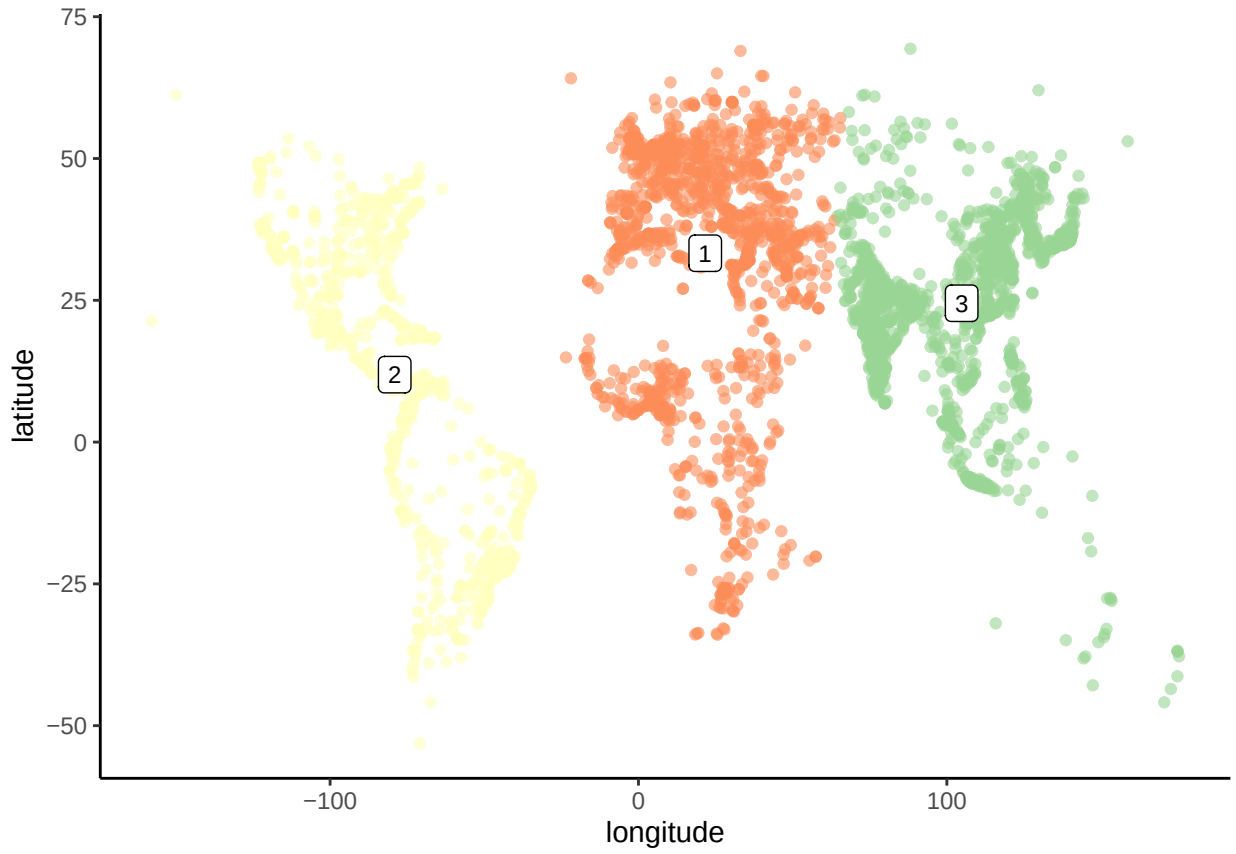
```
[1] 1466  945 1589
```

```
# Join cluster assignments with original dataset

big_cities2 <- big_cities %>%
  add_column(cluster1 = factor(city_kmeans_results2$cluster))

# Make dataframe out of estimated cluster centroids
city_cluster_centers2 <- city_kmeans_results2$centers %>%
  data.frame() %>%
  add_column(cluster_number = 1:3)
```

```
# Plot cluster assignments and centroids
ggplot(big_cities2, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster1), alpha = 0.6)  +
  scale_color_brewer(palette = "Spectral") +
  geom_label(data = city_cluster_centers2,
             aes(label = cluster_number),
             size = 3) +
  theme(legend.position = "none")
```



2.5 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 2 represent now?

> There are now 593 cities in cluster 1 now. Cluster 2 now represents either the UK, Italy, France, Spain, or even perhaps Greenland. It is hard to tell.

```
city_kmeans_results3$size
```

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

```
big_cities3 <- big_cities %>%
  add_column(cluster2 = factor(city_kmeans_results3$cluster))

# Make dataframe out of estimated cluster centroids
city_cluster_centers3 <- city_kmeans_results3$centers %>%
  data.frame() %>%
  add_column(cluster_number = 1:15)

# Plot cluster assignments and centroids
ggplot(big_cities3, aes(x = longitude, y = latitude)) +
  geom_point(aes(color = cluster2), alpha = 0.6)  +
  scale_color_brewer(palette = "Spectral") +
  geom_label(data = city_cluster_centers3,
             aes(label = cluster_number),
             size = 3) +
  theme(legend.position = "none")
```