Prep5S24

Bilal Tariq

2024-02-29

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Qmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope before the deadline (Sunday night, 3/3/24, by midnight).

Reading

The associated reading for the week is Chapter 19 and Section 12.1. The topics are text as data and clustering. Each of these topics are potential topics that could be incorporated into your final (not shiny) projects. Clustering is explored further in Stat 240 (Multivariate Data Analysis), and we have a text analytics elective (Stat 325).

1 - Scraping Poems

For our first prep problem, we want to return to Lab 5a_Scraping, where we played around with scraping the poems of Emily Dickinson from Wikipedia pages. In the Lab folder, open scrape-poems.R, the provided R script for the Lab solution. We want to look at the script before we tackle the data in our next Lab, combining our scraping and text analysis skills. The goal here is to work to understand the script and how it is doing the scraping, as this may be useful for your projects (shiny and/or final projects).

Note: If you decide to run the script in its entirety yourself, the process will take a decent length of time (15-30 minutes). However, you don't need to (and don't have to) run the code to answer the questions that follow. Just look at it.

> part a - The plyr::ldply command is new. What do you learn about this function in the help menu?

Solution: It takes each element of a list and applies a function to it then it combines it to a data frame.

> part b - Why does the loop have tryCatch? What is this meant to do?

Solution: It seems as though that loop constantly searches each url and sees if it has the div p in its html and searches for the poem. If it doesn't find it, then it just shows an error.

> part c - What is reported as the poem text if an error occurs while the function is working through the loop?

Solution: It returns the string "Missing".

> part d - Finally, why is it important to end the file with a save_RDS command? I.E. how is this useful? (You can look up what the function does!)

Solution: It saves a sort of connection to another file but doesn't actually save the file itself. This can be useful when we want to tie an R script to a file in r that has our data. Since we don't want to change our R script, we can use a saveRDS command.

2 - Text as Data

```
# set up text data provided
data(Macbeth_raw)

# str_split returns a list: we only want the first element
macbeth <- Macbeth_raw %>%
  str_split("\r\n") %>%
  pluck(1)
```

In Section 19.1.1, the `str_subset()`, `str_detect()`, and `str_which()` functions are introduced for detecting a pattern in a character vector (like finding a needle in a haystack). This section also introduces *regular expressions*.

> part a - Explain what the length values and 6 returned records tell us about what each function does below:

```
length(str_subset(macbeth, "  MACBETH"))
```

```
[1] 147
```

```
str_subset(macbeth, "  MACBETH") %>% head()
```

```
[1] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[2] "  MACBETH. So foul and fair a day I have not seen."
[3] "  MACBETH. Speak, if you can. What are you?"
[4] "  MACBETH. Stay, you imperfect speakers, tell me more."
[5] "  MACBETH. Into the air, and what seem'd corporal melted"
[6] "  MACBETH. Your children shall be kings."
```

```
length(str_which(macbeth, "  MACBETH"))
```

```
[1] 147
```

```
str_which(macbeth, "  MACBETH") %>% head()
```

```
[1] 228 433 443 466 478 483
```

```
length(str_detect(macbeth, "  MACBETH"))
```

```
[1] 3194
```

```
str_detect(macbeth, "  MACBETH") %>% head()
```

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

Solution: Str_subset returns the values or strings in this case where Macbeth is found. The first result returns the total results. As for str_which, it firstly returns the total results in the dataset. Then it returns the indices of where the search result was found. Finally, str_detect returns a logical vector in which the string was found. So it will always have the same length as the dataset.

part b - Why do the two lines below differ in their results?

```
str_subset(macbeth, "MACBETH\\.") %>% head()
```

```
[1] "  MACBETH. So foul and fair a day I have not seen."
[2] "  MACBETH. Speak, if you can. What are you?"
[3] "  MACBETH. Stay, you imperfect speakers, tell me more."
[4] "  MACBETH. Into the air, and what seem'd corporal melted"
[5] "  MACBETH. Your children shall be kings."
[6] "  MACBETH. And Thane of Cawdor too. Went it not so?"
```

```
str_subset(macbeth, "MACBETH.") %>% head()
```

```
[1] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[2] "  LADY MACBETH, his wife"
[3] "  MACBETH. So foul and fair a day I have not seen."
[4] "  MACBETH. Speak, if you can. What are you?"
[5] "  MACBETH. Stay, you imperfect speakers, tell me more."
[6] "  MACBETH. Into the air, and what seem'd corporal melted"
```

Solution: The period is known as a Mechanator. To search for the literal term of the period we need to use a double backslash.

part c - The three commands below look similar, but return different results. In words, explain what overall pattern is being searched for in each of the three cases (i.e., what do the patterns `MAC[B-Z]`, `MAC[B|Z]`, and `^MAC[B-Z]` indicate?)?

```
str_subset(macbeth, "MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
[3] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
[4] "THE TRAGEDY OF MACBETH"
[5] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[6] "  LADY MACBETH, his wife"
```

```
str_subset(macbeth,"MAC[B|Z]") %>% head()
```

```
[1] "THE TRAGEDY OF MACBETH"
[2] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[3] "  LADY MACBETH, his wife"
[4] "  MACBETH. So foul and fair a day I have not seen."
[5] "  MACBETH. Speak, if you can. What are you?"
[6] "  MACBETH. Stay, you imperfect speakers, tell me more."
```

```
str_subset(macbeth, "^MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
```

Solution: The first asks to search MAC followed by any character that may start from B till Z. The second does something similar, but it only allows for B OR Z. The third one uses a carrot to represent an anchor of the pattern to the beginning of the text.

Optional - Explore these other patterns to figure out what they do.

```
str_subset(macbeth, ".*MAC[B-Z]") %>% head()
```

```
[1] "MACHINE READABLE COPIES MAY BE DISTRIBUTED SO LONG AS SUCH COPIES"
[2] "MACHINE READABLE COPIES OF THIS ETEXT, SO LONG AS SUCH COPIES"
[3] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
[4] "THE TRAGEDY OF MACBETH"
[5] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[6] "  LADY MACBETH, his wife"
```

```r
str_subset(macbeth, ".MAC[B-Z]") %>% head()
```

```
[1] "WITH PERMISSION.  ELECTRONIC AND MACHINE READABLE COPIES MAY BE"
[2] "THE TRAGEDY OF MACBETH"
[3] "  MACBETH, Thane of Glamis and Cawdor, a general in the King's"
[4] "  LADY MACBETH, his wife"
[5] "  MACDUFF, Thane of Fife, a nobleman of Scotland"
[6] "  LADY MACDUFF, his wife"
```

```r
str_subset(macbeth, "more$") %>% head()
```

```
[1] "    Who, almost dead for breath, had scarcely more"
[2] "    Hath left you unattended. [Knocking within.] Hark, more"
[3] "more"
```

```r
# Code below should return character(0) (i.e., nothing)
str_subset(macbeth, "^MAC[B|Z]") %>% head()
```

```
character(0)
```

Notes:

3 - 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. The data for this example comes from the **mdsr** package.

> part a - Run the code below to implement the k-means algorithm. Take a look at the resulting output for each object in the code chunk in some way (print in the console, `glimpse()`, or `head()`).

Solution:

```
data(world_cities)

# Identify the 4,000 biggest cities in the world
big_cities <- 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)

glimpse(city_kmeans_results)
```

```
List of 9
 $ cluster     : int [1:4000] 3 6 4 1 2 3 1 3 3 1 ...
 $ centers     : num [1:6, 1:2] 75.1 -94.5 120.4 -57.1 18.9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "longitude" "latitude"
 $ totss       : num 24082261
 $ withinss    : num [1:6] 180486 203611 481501 150292 171681 ...
 $ tot.withinss: num 1527888
 $ betweenss   : num 22554373
 $ size        : int [1:6] 726 554 984 392 356 988
 $ iter        : int 3
```

```
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

part b - 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 documentation will help you understand what each named element is.

```
# 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"
```

Solution: It has a class of type kmeans and the values of the objects contain information of the character vector which is up to or around the same length. The named elements contained by the object are in the second line.

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

```
# 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
```

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

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

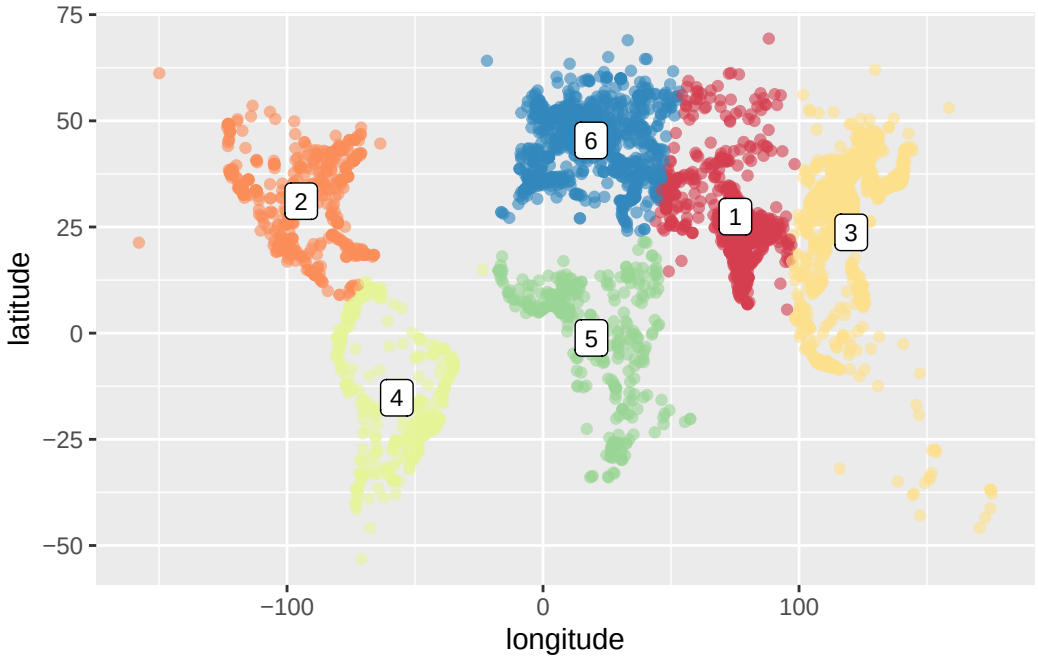
Solution: The estimated centroids of the 6 clusters are given above with the longitude and latitude coordinates. The cluster that contains the most cities seems to cluster 6 as it has 988 data points.

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

```r
# 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")
```



Solution: Cluster 2 represents the major cities centered or around North America.

> part e - In k-means clustering, the analyst specifies the number of clusters to create. Reproduce the k-means analysis above but update the `center` argument within the `kmeans()` function to identify 3 clusters instead of 6. (Show the code for this - copy/paste/edit as needed.) Then, 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?

Solution: There are 1589 cities in cluster one. Cluster 2 seems to now represent the continents of north america and south america.

```
# be sure to set a seed for reproducibility before k-means runs

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

# Make sure to set seed for reproducibility!
set.seed(12)
```

```r
city_kmeans_results <- big_cities %>%
  kmeans(centers = 3, nstart = 30)

# Centroids
city_kmeans_results$centers
```

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

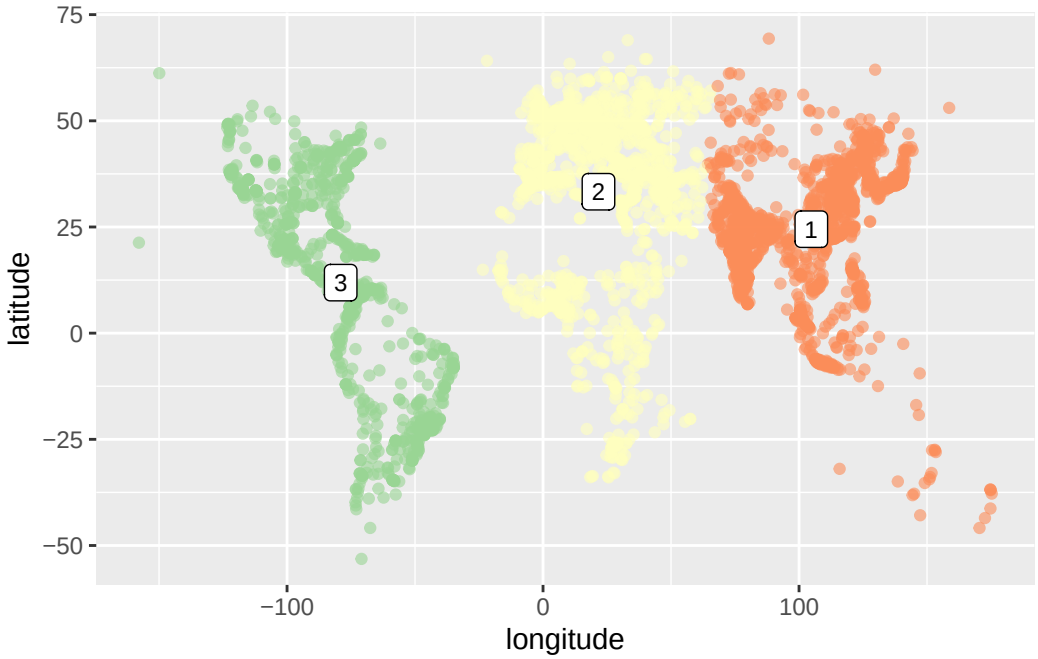
```r
# Cluster sizes
city_kmeans_results$size
```

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

```r
# 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 = 1:3)

# 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")
```


75
50
25
0
−25
−50
latitude
−100
0
100
longitude
1
2
3