Assignment 4

Chiu-Yuan Wu

10/8/2020

Problem 1 (50 pts) This problem will involve the nycflights13 dataset (including tables airlines, airports, planes and weather), which we saw in class. It is available in both R and Python, however R is recommended for at least the visualization portion of the question. Start by installing and importing the dataset to your chosen platform. We will first use joins to search and manipulate the dataset, then we will produce a flightpath visualization.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.3.0

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(dplyr)
library(nycflights13)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

a. (10 pts) Filter the dataset (using a left join) to display the tail number, year, month, day, hour, origin, and humidity for all flights heading to Tampa International Airport (TPA) on the afternoon of November 1, 2013.

```r
flights = nycflights13::flights
weather = nycflights13::weather
airports = nycflights13::airports
airlines = nycflights13::airlines
planes = nycflights13::planes

numeric_time = flights$time_hour
numeric_time <- gsub("[: -]", "", numeric_time, perl = TRUE)
flights = cbind(flights, numeric_time)
numeric_time = weather$time_hour
numeric_time <- gsub("[: -]", "", numeric_time, perl = TRUE)
weather = cbind(weather, numeric_time)

flights_join <- left_join(flights, weather, by = "time_hour", all.x = true)
flights_join_sub = filter(flights_join, month.x == 11, day.x == 1, dest == "TPA",
    numeric_time.x > 20131101120000)

tailnum = flights_join_sub$tailnum
year = flights_join_sub$year.x
month = flights_join_sub$month.x

filtered_flights <- cbind.data.frame(flights_join_sub$tailnum, flights_join_sub$year.x) %>%
    cbind.data.frame(flights_join_sub$month.x) %>% cbind.data.frame(flights_join_sub$day.x) %>%
    cbind.data.frame(flights_join_sub$hour.x) %>% cbind.data.frame(flights_join_sub$origin.x) %>%
    cbind.data.frame(flights_join_sub$humid)

setNames(filtered_flights, c("Tail number", "Year", "Month", "Day", "Hour", "Origin",
    "Humidity"))
```

```
##    Tail number Year Month Day Hour Origin Humidity
## 1       N580JB 2013    11   1   14    JFK    52.95
## 2       N580JB 2013    11   1   14    JFK    63.08
## 3       N580JB 2013    11   1   14    JFK    56.51
## 4       N337NB 2013    11   1   14    LGA    52.95
## 5       N337NB 2013    11   1   14    LGA    63.08
## 6       N337NB 2013    11   1   14    LGA    56.51
## 7       N567UA 2013    11   1   15    EWR    52.80
## 8       N567UA 2013    11   1   15    EWR    65.07
## 9       N567UA 2013    11   1   15    EWR    50.60
## 10      N515MQ 2013    11   1   14    JFK    52.95
## 11      N515MQ 2013    11   1   14    JFK    63.08
## 12      N515MQ 2013    11   1   14    JFK    56.51
## 13      N779JB 2013    11   1   15    EWR    52.80
## 14      N779JB 2013    11   1   15    EWR    65.07
## 15      N779JB 2013    11   1   15    EWR    50.60
## 16      N561JB 2013    11   1   16    LGA    52.80
## 17      N561JB 2013    11   1   16    LGA    67.57
## 18      N561JB 2013    11   1   16    LGA    50.60
## 19      N974DL 2013    11   1   18    JFK    65.07
## 20      N974DL 2013    11   1   18    JFK    74.75
## 21      N974DL 2013    11   1   18    JFK    60.51
## 22      N319NB 2013    11   1   19    LGA    72.53
## 23      N319NB 2013    11   1   19    LGA    83.54
```

```
## 24      N319NB 2013    11   1   19    LGA    60.51
## 25      N76265 2013    11   1   19    EWR    72.53
## 26      N76265 2013    11   1   19    EWR    83.54
## 27      N76265 2013    11   1   19    EWR    60.51
## 28      N768JB 2013    11   1   19    JFK    72.53
## 29      N768JB 2013    11   1   19    JFK    83.54
## 30      N768JB 2013    11   1   19    JFK    60.51
```

b. (10 pts) What is the difference between the following two joins? anti_join(flights, airports, by = c("dest" = "faa")) anti_join(airports, flights, by = c("faa" = "dest"))

For the first join, we can find the flights that has a destination that is not recorded in the airports data. For the second join, we can find the airports that has no flight fly to.

c. (10 pts) Select the origin and destination airports and their latitude and longitude for all fights in the dataset (using one or more inner joins). Hint: There should be 329,174 flights if you've done this correctly.

```
Airports_Flights = inner_join(airports, flights, by = c(faa = "dest"))

filtered_flights_1c <- cbind.data.frame(Airports_Flights$origin, Airports_Flights$faa) %>%
    cbind.data.frame(Airports_Flights$lat) %>% cbind.data.frame(Airports_Flights$lon)

nrow(filtered_flights_1c)
```

```
## [1] 329174
```

d. (10 pts) Use group_by and count to get the number of flights to each unique origin/destination combination. Hint: There should be 217 of these total.

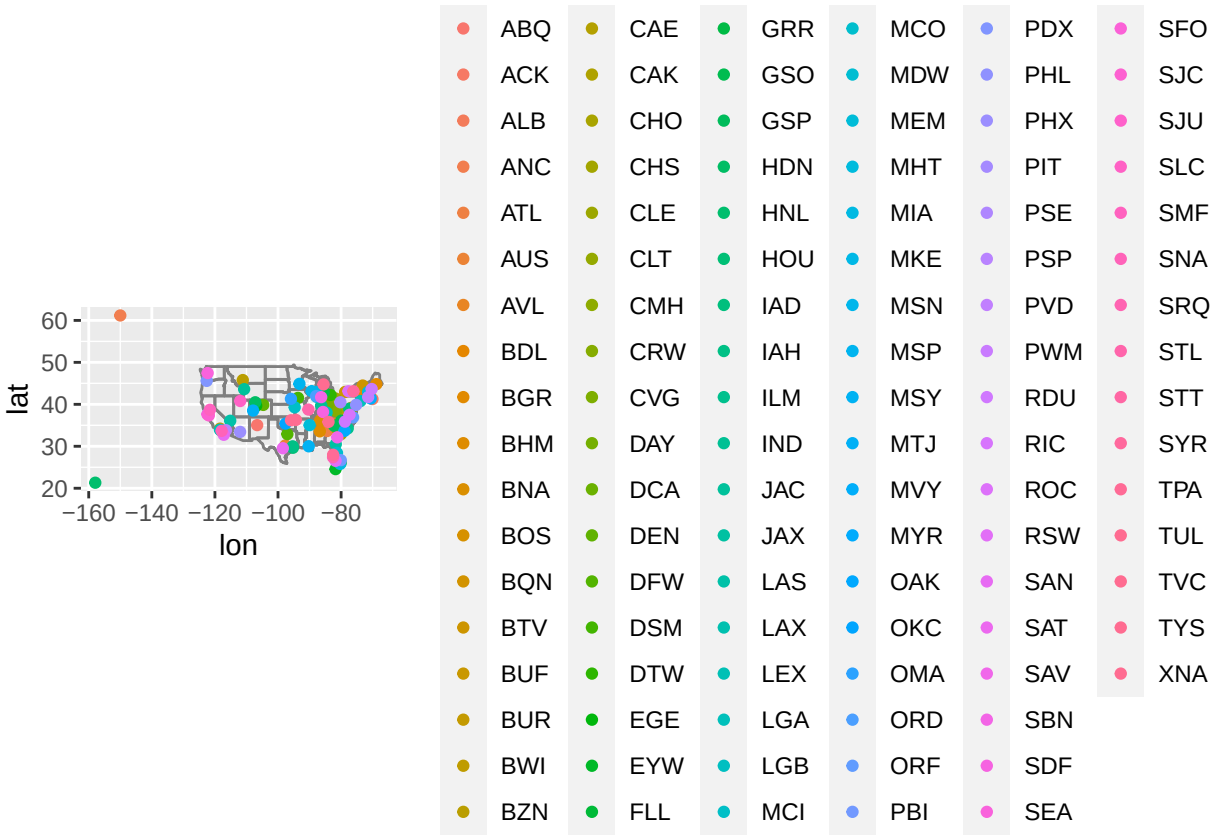
```
unique_flights <- group_by(flights, origin, dest) %>% count
nrow(unique_flights)
```

```
## [1] 224
```

e. (10 pts) Produce a map that colors each destination airport by the average air time of its incoming flights. Here is a code snippet to draw a map of all flight destinations, which you can use as a starting point. You may need to install the maps packages if you have not already. Adjust the title, axis labels and aesthetics to make this visualization as clear as possible. Hint: You may find it useful to use a different type of join in your solution than the one in the snippet.

```
filtered_airports = right_join(airports, flights, c(faa = "dest"))
filtered_airports %>% select(faa, air_time, lat, lon) %>% group_by(faa, lat, lon) %>%
    summarise_at(vars(air_time), list(AverageAirTime = mean), na.rm = TRUE) %>% ggplot(aes(lon,
    lat)) + borders("state") + geom_point(aes(colour = factor(faa))) + coord_quickmap()
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```




```
ave_time = filtered_airports %>% select(faa, air_time, lat, lon) %>% group_by(faa,
    lat, lon) %>% summarise_at(vars(air_time), list(AverageAirTime = mean), na.rm = TRUE)
```

Problem 2 (30 pts) You may recall from on the lecture on Friday, Sep 25 (when we had Dr. Ofer Amram as a guest speaker), the warm-up question that day was to type in the city and state (or city and country) where you grew up. The result of that warm-up question is summarized in a plaintext file posted alongside this assignment. The task you have for this problem is to visualize that list on a world map, indicating in some way the cities. Try to make your visualization as nice looking and visually informative as possible.

```
library(ggmap)

register_google(key = "AIzaSyBw5ru_ASNNuxomeLOdUA1X2ZqyjLCn39I")

city = readLines("WarmupData.txt")
citydf <- data.frame(city)
city_ggmap <- geocode(location = city, output = "more", source = "google")
```

```
# library(sf)

city_tibble <- as_tibble(city_ggmap)
# I tried an entire day and still could not get the sf package work
```

Problem 3 (20 pts). Create a word cloud for an existing document (relatively short, say a couple pages) of your choice. Examples of suitable documents include: summary of a recent project you are working or have worked on; your own recent Statement of Purpose or Research Statement; an essay or article on some topic you find interesting or some other similar document.

```r
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```r
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
cloud = readLines("cpts591_finalreport.txt")
cloud <- Corpus(VectorSource(cloud))

cloud <- tm_map(cloud, removeWords, "the")
```

```
## Warning in tm_map.SimpleCorpus(cloud, removeWords, "the"): transformation drops
## documents
```

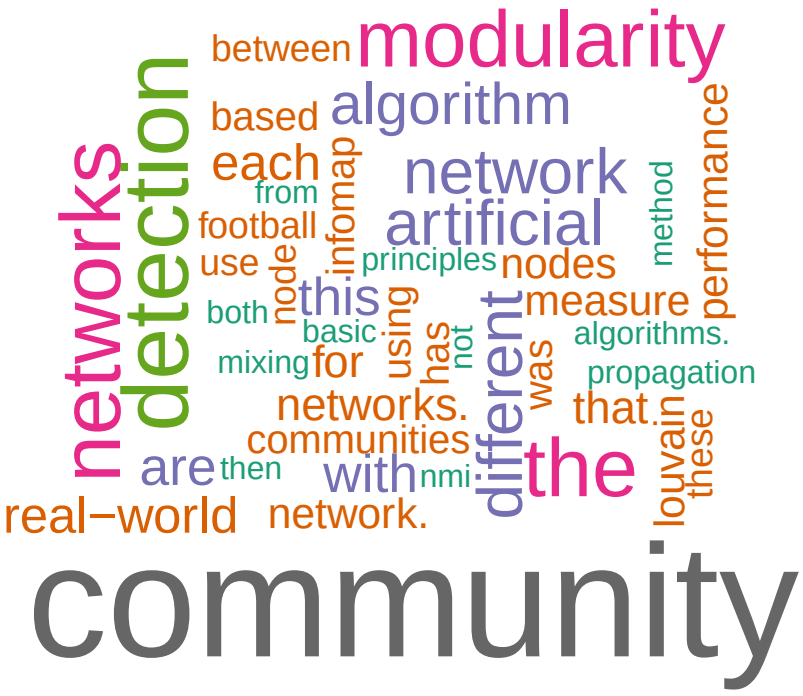
```r
cloud <- tm_map(cloud, removeWords, "and")
```

```
## Warning in tm_map.SimpleCorpus(cloud, removeWords, "and"): transformation drops
## documents
```

```r
cloud_tdm <- TermDocumentMatrix(cloud)
tdm <- as.matrix(cloud_tdm)
sorted_tdm = sort(rowSums(tdm), decreasing = TRUE)

wordcloud(cloud, scale = c(5, 0.5), max.words = Inf, min.freq = 5, random.order = TRUE,
    rot.per = 0.35, use.r.layout = FALSE, colors = brewer.pal(8, "Dark2"))
```

```
## Warning in wordcloud(cloud, scale = c(5, 0.5), max.words = Inf, min.freq = 5, :
## algorithms could not be fit on page. It will not be plotted.
```


community
modularity
detection
networks
algorithm
network
artificial
performance
between
based
each
from
football
infomap
method
use
node
principles
nodes
this
measure
both
basic
using
has
not
different
was
algorithms.
mixing
for
propagation
networks.
that
louvain
these
communities
the
are
then
with
nmi
real−world
network.