Homework 3

Aaron Coates

5/13/2019

```
setwd("~/Documents/GitHub/MMSS_311_2")

library(tidytext)
library(tm)
```

```
## Loading required package: NLP
```

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

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

```
library(proxy)
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.2-2 (2019-03-07) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
```

```
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```
library(mixtools)
```

```
## mixtools package, version 1.1.0, Released 2017-03-10
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518

##
## Attaching package: 'mixtools'

## The following object is masked from 'package:grid':
##
##     depth
```

1.1

```
manifestos <- read_csv("/Users/aaroncoates/Downloads/manifestos.csv")
```

```
## Parsed with column specification:
## cols(
##   doc_id = col_character(),
##   text = col_character()
## )
```

```
tidymani <- manifestos %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  mutate(word = removeNumbers(word)) %>%
  mutate(word = stripWhitespace(word)) %>%
  mutate(word = removePunctuation(word)) %>%
  mutate(word = stemDocument(word)) %>%
  group_by(doc_id) %>%
  count(word) %>%
  subset(word != '') %>%
  cast_dtm(doc_id, word, n) %>%
  removeSparseTerms(0.99) %>%
  as.matrix()
```

```
## Joining, by = "word"
```

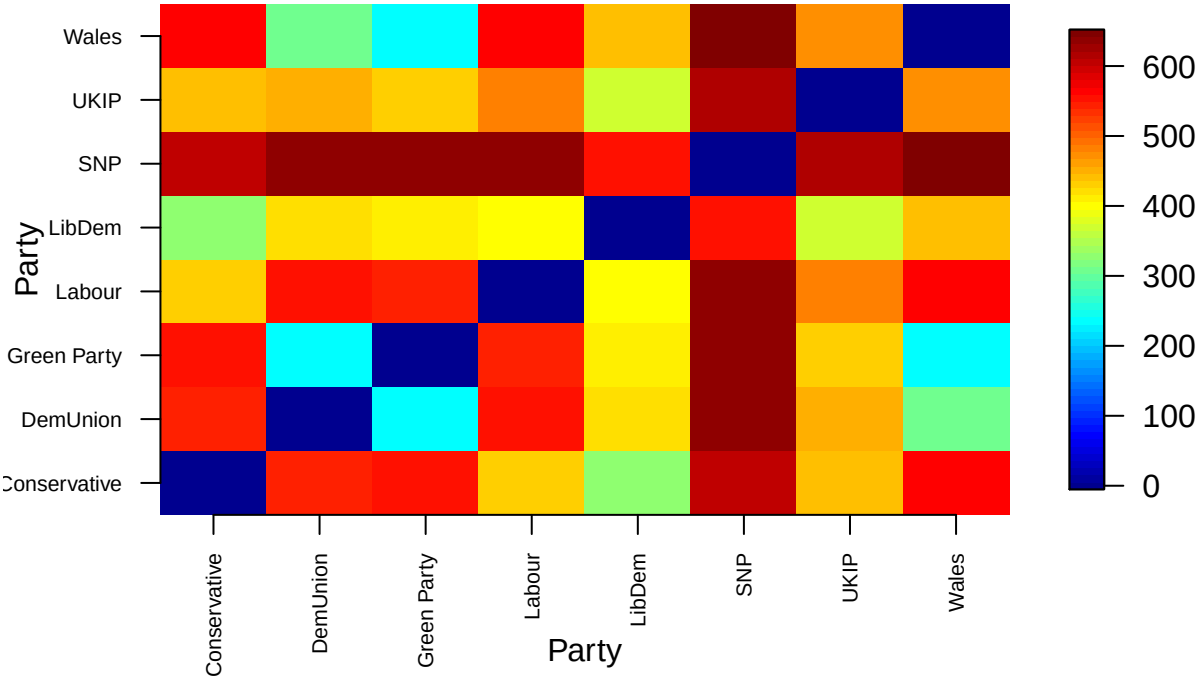
1.2

```
eucdist.mani <- dist(tidymani) %>%
  as.matrix()
eucdist.mani
```

```
##              Conservative DemUnion Green Party   Labour   LibDem      SNP
```

```
## Conservative       0.0000 540.9251    553.1763 430.9872 331.1148 605.1454
## DemUnion         540.9251   0.0000    233.8846 550.0636 416.3352 636.9356
## Green Party      553.1763 233.8846      0.0000 545.5199 407.2677 637.9647
## Labour           430.9872 550.0636    545.5199   0.0000 400.8628 637.7907
## LibDem           331.1148 416.3352    407.2677 400.8628   0.0000 554.7901
## SNP              605.1454 636.9356    637.9647 637.7907 554.7901   0.0000
## UKIP             437.0595 452.5030    433.9850 484.7422 372.9048 620.1887
## Wales            566.1042 312.4644    233.4266 564.4201 438.1153 646.8995
##                      UKIP    Wales
## Conservative 437.0595 566.1042
## DemUnion     452.5030 312.4644
## Green Party  433.9850 233.4266
## Labour       484.7422 564.4201
## LibDem       372.9048 438.1153
## SNP          620.1887 646.8995
## UKIP           0.0000 467.4345
## Wales        467.4345   0.0000
```

```
manifestosabc <- manifestos[order(manifestos$doc_id),]
```



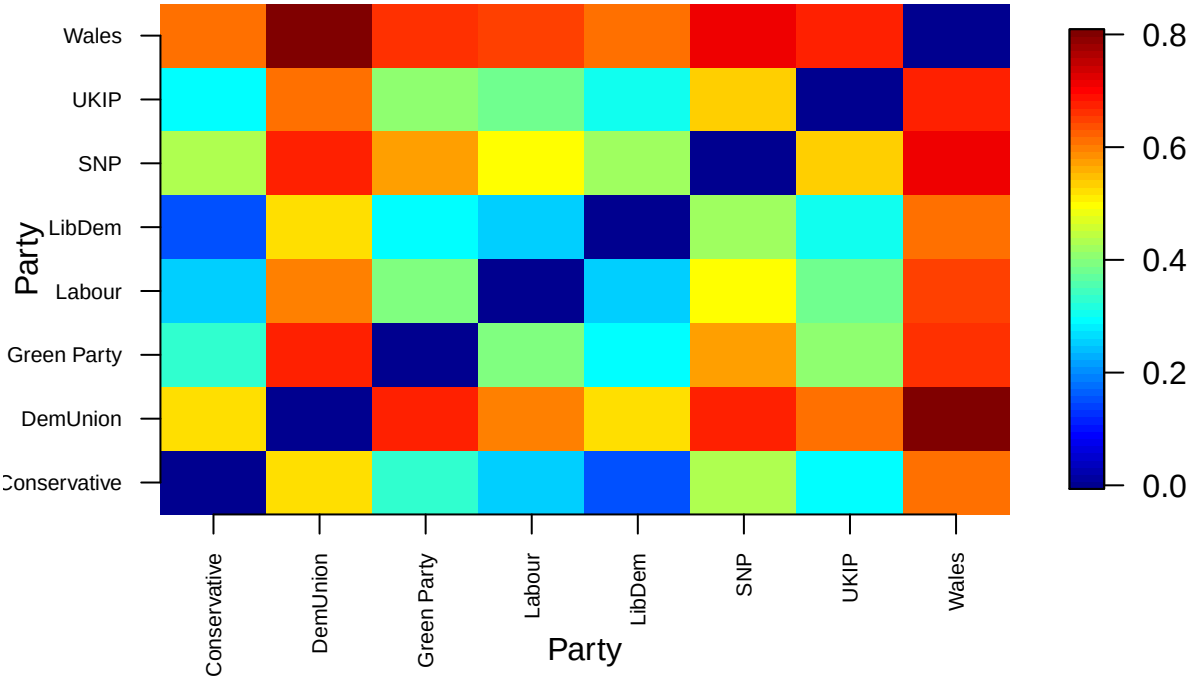
Using Euclidean distance, it seems as though the Green Party and the DemUnion have very similar manifestos. The Wales and the Green Party are also just as similar. In addition, LibDem and the Conservative Party are somewhat similar, as are Wales and DemUnion.

1.3

```
cosdist.mani <- dist(tidymani, method="cosine") %>%
  as.matrix()
cosdist.mani
```

```
##              Conservative  DemUnion Green Party    Labour    LibDem
## Conservative    0.0000000 0.5191131   0.3314567 0.2517155 0.1540386
## DemUnion        0.5191131 0.0000000   0.6764975 0.6009285 0.5270453
## Green Party     0.3314567 0.6764975   0.0000000 0.3931298 0.2874942
```

```
## Labour          0.2517155 0.6009285   0.3931298 0.0000000 0.2549242
## LibDem          0.1540386 0.5270453   0.2874942 0.2549242 0.0000000
## SNP             0.4381332 0.6721036   0.5751741 0.4965629 0.4200896
## UKIP            0.2923578 0.6171294   0.4055833 0.3809932 0.3031847
## Wales           0.6111722 0.8028051   0.6676606 0.6549316 0.6068421
##                    SNP      UKIP     Wales
## Conservative 0.4381332 0.2923578 0.6111722
## DemUnion     0.6721036 0.6171294 0.8028051
## Green Party  0.5751741 0.4055833 0.6676606
## Labour       0.4965629 0.3809932 0.6549316
## LibDem       0.4200896 0.3031847 0.6068421
## SNP          0.0000000 0.5339088 0.7104875
## UKIP         0.5339088 0.0000000 0.6741885
## Wales        0.7104875 0.6741885 0.0000000
```



Using cosine distance, LibDem and Conservative are the closest two parties. In addition, the Labour and Conservative parties are similar, as are the LibDem and Labour parties.

1.4

In principle, cosine distance is the most accurate measure for this particular application. This is because cosine distance controls for varying lengths of each manifesto and is able to produce a similarity index that is independent of manifesto length.

2.1

```
sanny <- read_csv("/Users/aaroncoates/Downloads/311_sanitation_requests_2019.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   `Unique Key` = col_double(),
##   `Incident Zip` = col_double(),
##   `Intersection Street 1` = col_logical(),
##   `Intersection Street 2` = col_logical(),
```

```
##   Landmark = col_logical(),
##   BBL = col_double(),
##   `X Coordinate (State Plane)` = col_double(),
##   `Y Coordinate (State Plane)` = col_double(),
##   `Vehicle Type` = col_logical(),
##   `Taxi Company Borough` = col_logical(),
##   `Taxi Pick Up Location` = col_logical(),
##   `Bridge Highway Name` = col_logical(),
##   `Bridge Highway Direction` = col_logical(),
##   `Road Ramp` = col_logical(),
##   `Bridge Highway Segment` = col_logical(),
##   Latitude = col_double(),
##   Longitude = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 2 parsing failures.
##   row                   col           expected        actual
## 31141 Intersection Street 1 1/0/T/F/TRUE/FALSE THAMES STREET '/Users/aaroncoates/Downloads/311_sanita
## 31141 Intersection Street 2 1/0/T/F/TRUE/FALSE PORTER AVENUE '/Users/aaroncoates/Downloads/311_sanita
```

```
sanny <- sanny[!is.na(sanny$Latitude), ]
sanny <- sanny[!is.na(sanny$Longitude), ]

set.seed(80)
kmeanmat<- kmeans(select(sanny, c(Latitude, Longitude)), 2, nstart=50)
clusty <- kmeanmat$cluster
sanny$cluster <- as.factor(clusty)
```

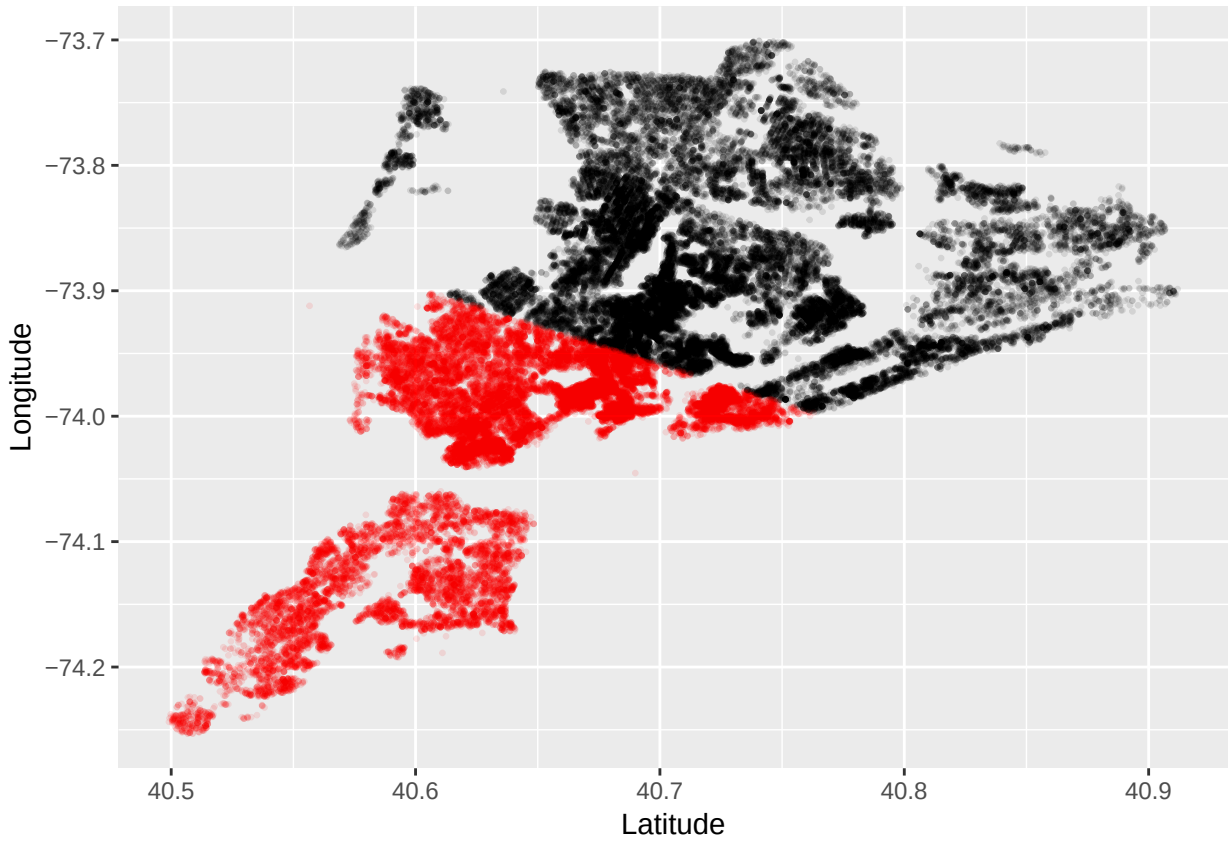
2.2

```
table(sanny$`Borough`, sanny$cluster)
```

```
##
##                     1     2
##   BRONX          5045     0
##   BROOKLYN      11248 22234
##   MANHATTAN      4868  3669
##   QUEENS        27448     1
##   STATEN ISLAND     0 10049
```

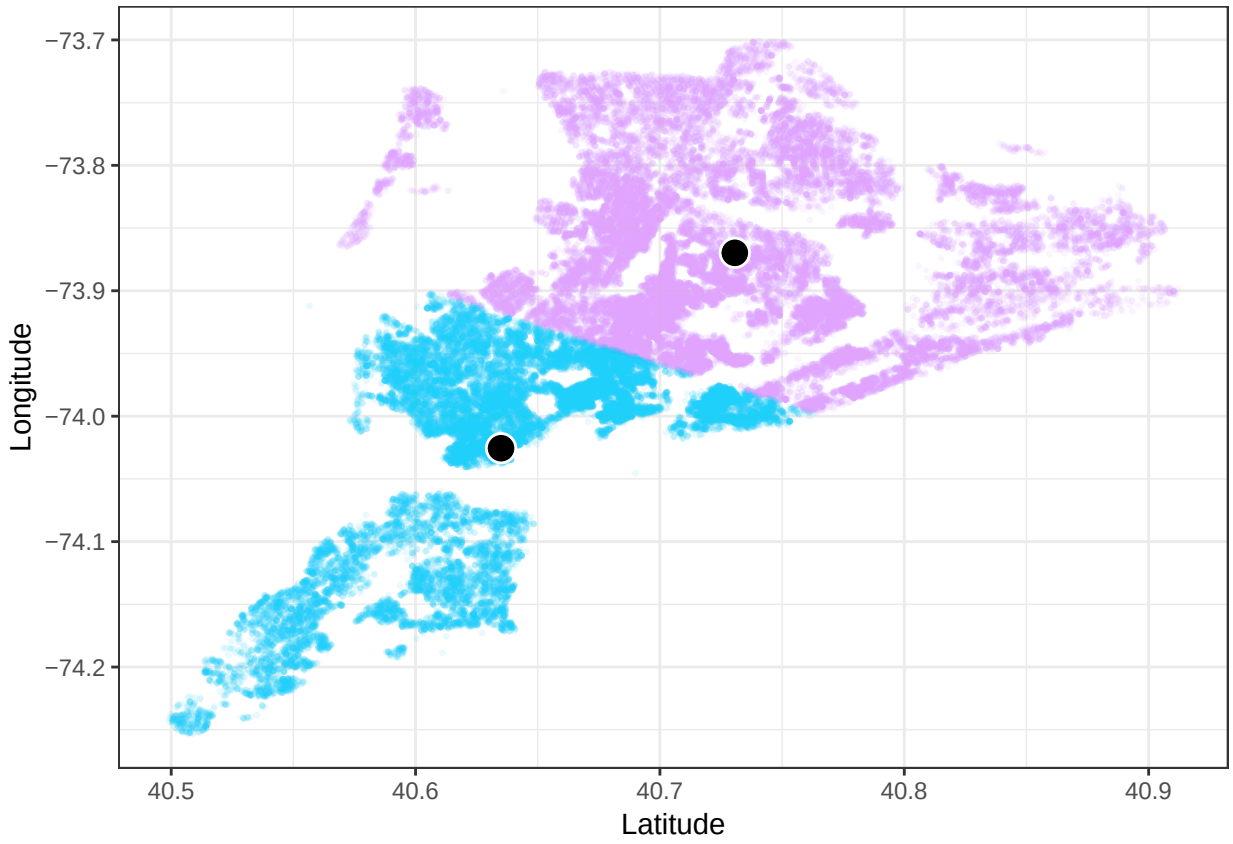
The clusters match the political boundaries somewhat well. Queens, the Bronx, and Staten Island are each encompassed in only one cluster. Brooklyn and Manhattan are split up.

2.3



2.4

```
centroidz <- as.data.frame(kmeanmat$centers)
```

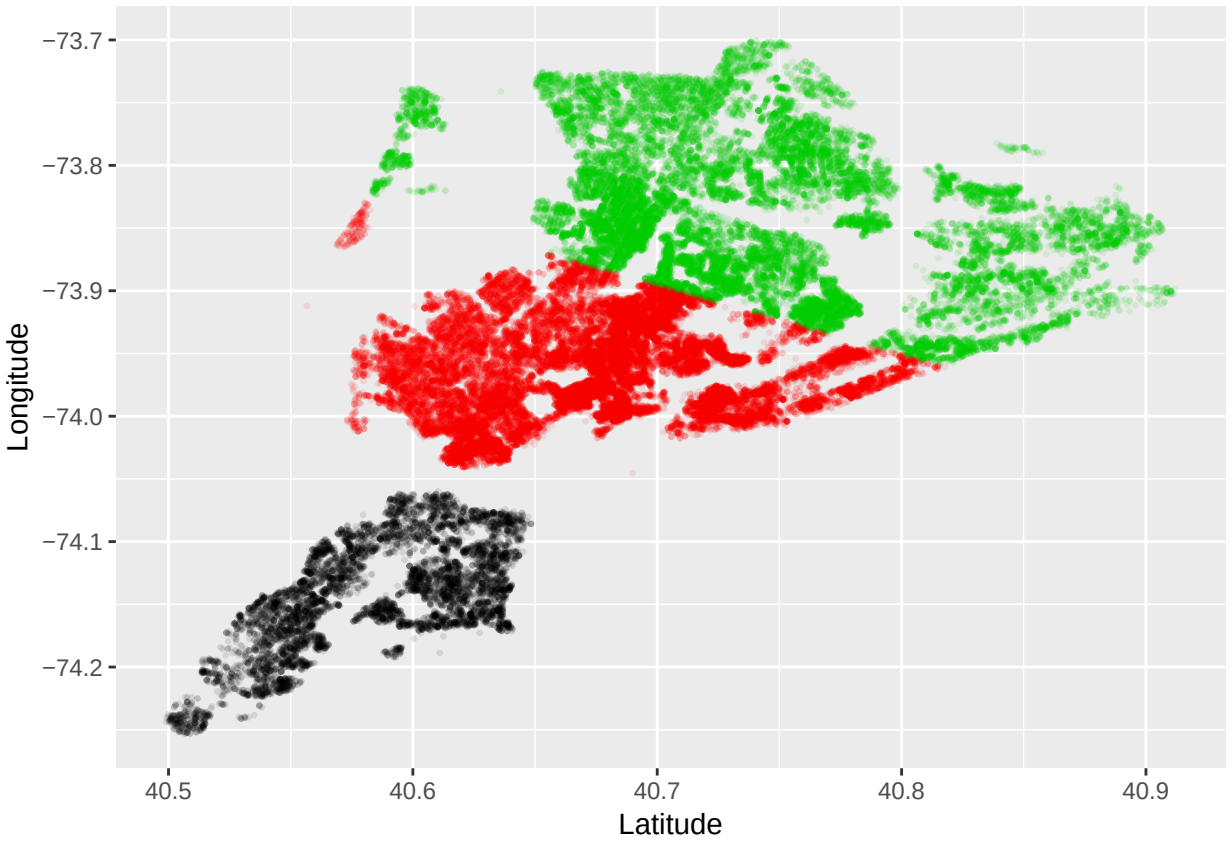


2.5

```
kmeanmat3 <- kmeans(select(sanny, c(Latitude, Longitude)), 3, nstart=50)
clusty3 <- kmeanmat3$cluster
sanny$cluster3 <- as.factor(clusty3)
table(sanny$`Borough`, sanny$cluster3)
```

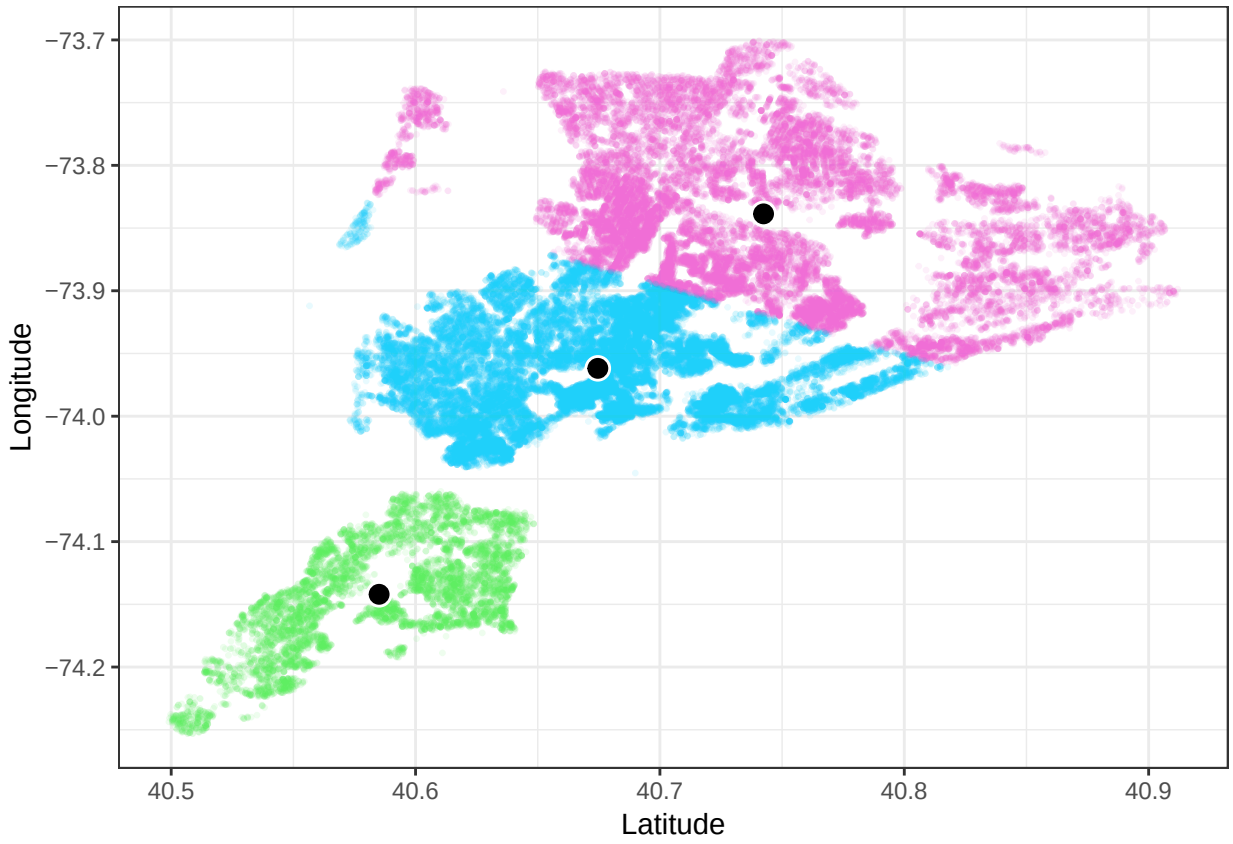
```
##
##                     1     2     3
##   BRONX             0     0  5045
##   BROOKLYN          0 32668   814
##   MANHATTAN         0  6968  1569
##   QUEENS            0  2422 25027
##   STATEN ISLAND 10049     0     0
```

First, plot without centers.



Then, plot with centers.

```
centroidz3 <- as.data.frame(kmeanmat3$centers)
```



2.6

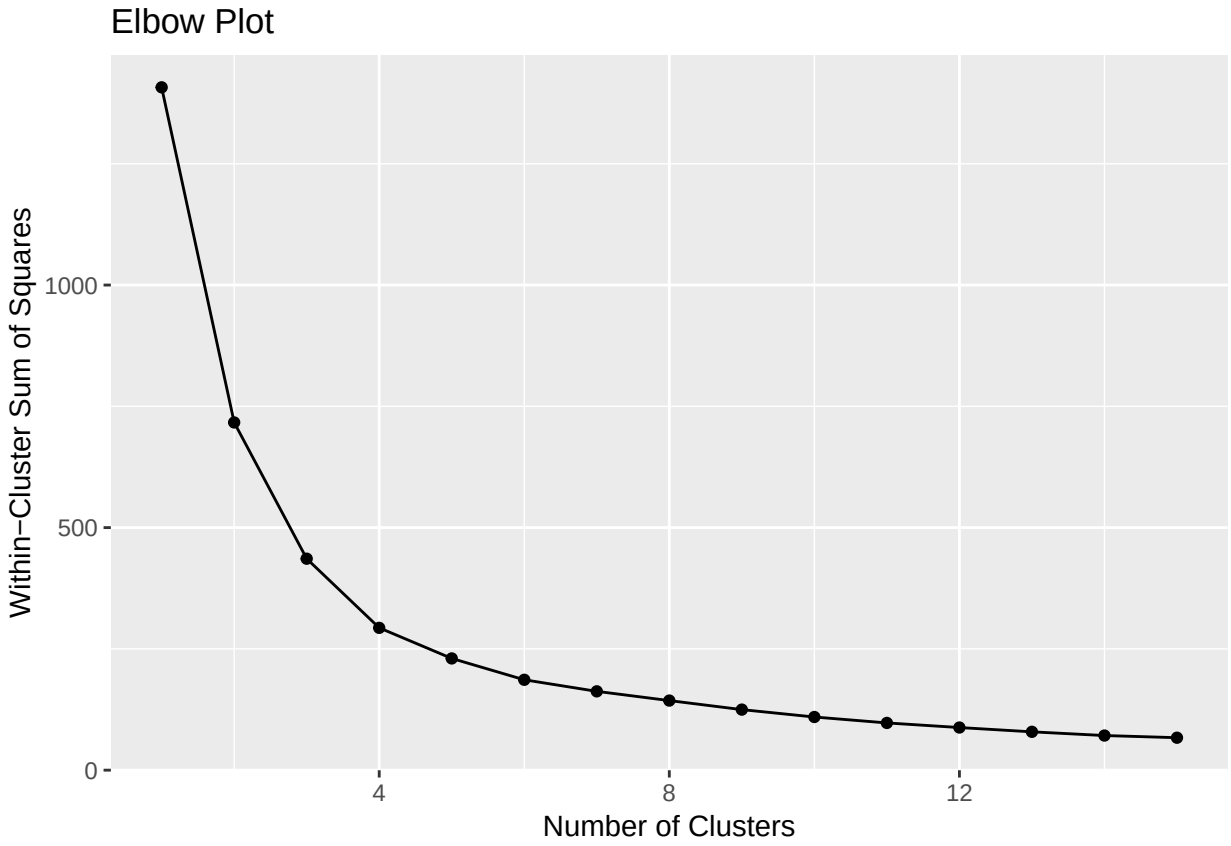
```
sumsq <- numeric(15)
for (i in 1:15){
  km <- kmeans(select(sanny, c(Latitude, Longitude)), i, nstart=15)
  sanny[[paste0("k", i)]] <- as.factor(km$cluster)
  sumsq[i] <- km$tot.withinss
}
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 4228100)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 4228100)
```

```
sumsq<- as.data.frame(sumsq)
sumsq$k <- c(1:15)
```

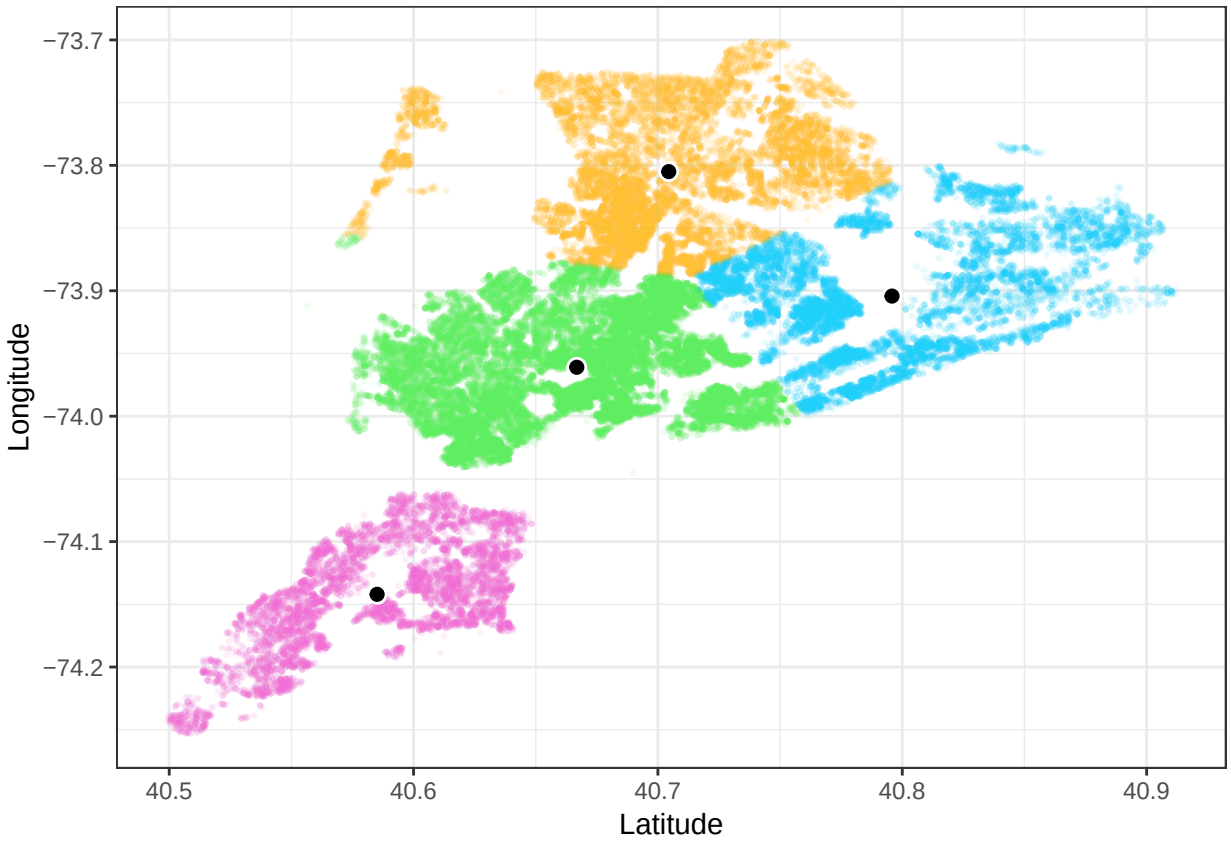
2.7



Since the new facilities are very expensive to create, it seems as though using four facilities would fit our results the most well. Increasing the number of centers from four would only slightly decrease Sum of Squared Distance.

2.8

```
kmeanmat4 <- kmeans(select(sanny, c(Latitude, Longitude)), 4, nstart=50)
clusty4 <- kmeanmat4$cluster
sanny$cluster4 <- as.factor(clusty4)
centroidz4 <- as.data.frame(kmeanmat4$centers)
```



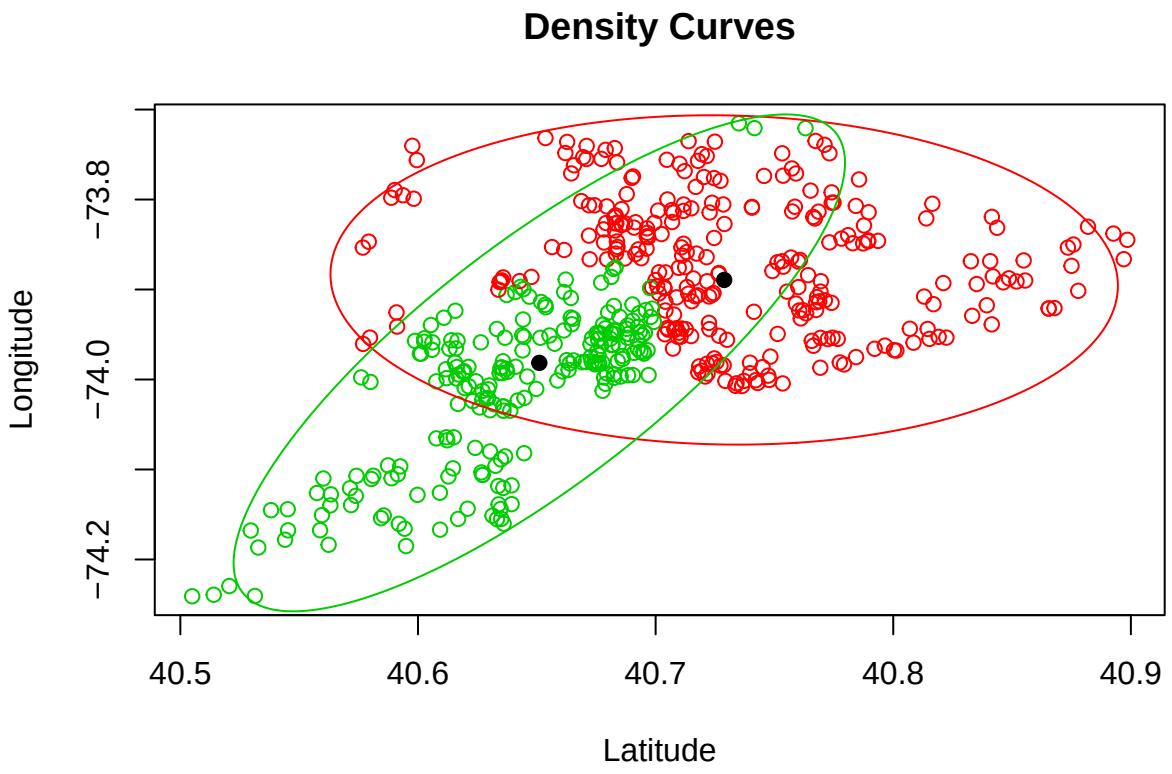
2.9

For this application, Euclidean distance finds clusters of calls that are the closest based solely on latitude and longitude. However, this will not take into account the specific geography of New York City. For instance, perhaps there is a water barrier that significantly increases the travel from a facility to a customer call. Euclidean distance would not be able to take this into account when forming clusters.

3.1-3.2

```
final <- sanny[sample(1:nrow(sanny), 500, replace=FALSE),]
```

```
## number of iterations= 113
```



3.3

```
## number of iterations= 357
```