citadel_datathon_practice1

May 12, 2018

```
In [6]: library(ggplot2) # Data visualization
        library(readr) # CSV file I/O, e.g. the read_csv function
        library(plyr)
        library(dplyr)
        library(reshape2)
```

Tried to download raw data from NYC open data's 311 requests https://nycopendata.socrata.com/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9#

The data size is extremly gigantic, with 17.1 M rows and 41 columns (from 2010 to present) Even for a timeperiod of one year, there are 2.1 M rows. (To get a perspective, the data that'd supplied in Datathon has 1 M rows, 19 columns, sized ~425 MB) Now it makes sense on why we've to rely on organizer's Dataset instead of downloading from source.

I found some simple versions, (reduced number of columns, and limited time frame.)

```
In [4]: NYC311 = read.csv('2014_NYC.csv', header=T)
```

DATA Preparation * Making all lower case, removing punctuation * Grouping similar complaints * Some zipcodes are 9 digits, we will consider just first 5 digits * Aggregate by complaint type and zipcode

```
In [7]: #Clean data and make complaints uniform
        NYC311$Complaint.Type = tolower(NYC311$Complaint.Type)
        NYC311$Complaint.Type = gsub('s$', '', NYC311$Complaint.Type)
        NYC311$Incident.Zip = gsub('-[[:digit:]]{4}$', '', NYC311$Incident.Zip)
        NYC311$Complaint.Type = gsub('paint - plaster', 'paint/plaster', NYC311$Com
        NYC311$Complaint.Type = gsub('general construction', 'construction', NYC311
        NYC311$Complaint.Type = gsub('nonconst', 'construction', NYC311$Complaint.T
        NYC311$Complaint.Type = gsub('street sign - [[:alpha:]]+', 'street sign', N
        NYC311$Complaint.Type = gsub('fire alarm - .+','fire alarm', NYC311$Complai
        idx = grepl('[[:digit:]]{5}', NYC311$Incident.Zip)
        NYC311clean = NYC311[idx,]
```

```
In [8]: #Counts of each complaint by zipcode
        NYC311byZip = ddply(NYC311clean, .(Incident.Zip, Complaint.Type), count)
```

Data Exploration * Using Factor analysis, trying to explore if the underlying structure of the data set

In [13]:
```
library(tidyr) #prepare data for pca
raw = spread(NYC311byZip, Complaint.Type, n)
raw[is.na(raw)] = 0
counts = which(colSums(raw[,-1]) < 10)
zipcodes = raw[,1]
raw = raw[,-1]
raw = raw[,-counts]
processed = scale(raw, center=T, scale=T)

library(psych)
pca = principal(processed, nfactor=6, covar=F)
```

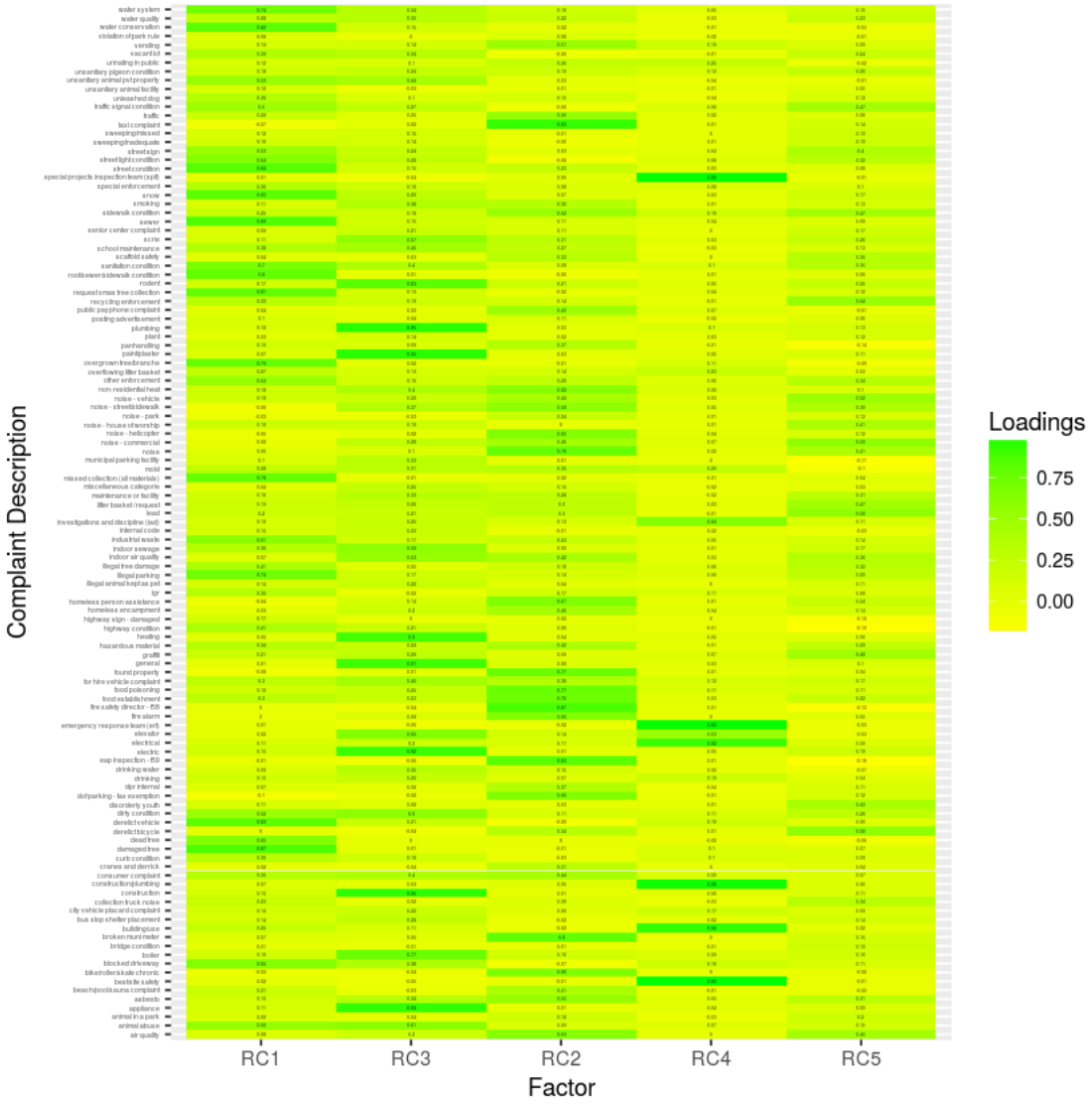
```
Attaching package: 'psych'

The following objects are masked from 'package:ggplot2':

    %+%, alpha
```

In [19]:
```
#Visualize EFA
loadings = as.data.frame(pca$loadings[,1:5])
loadings$complaint.type = rownames(loadings)
loadings_m = melt(loadings, id='complaint.type')

ggplot(loadings_m, aes(x=variable, y=complaint.type, label = round(value,2
      geom_tile()+xlab('Factor')+ylab('Complaint Description')+geom_text(s
      scale_fill_continuous(low='yellow', high='green', name='Loadings')+
      theme(axis.text.y = element_text(size=3))
```

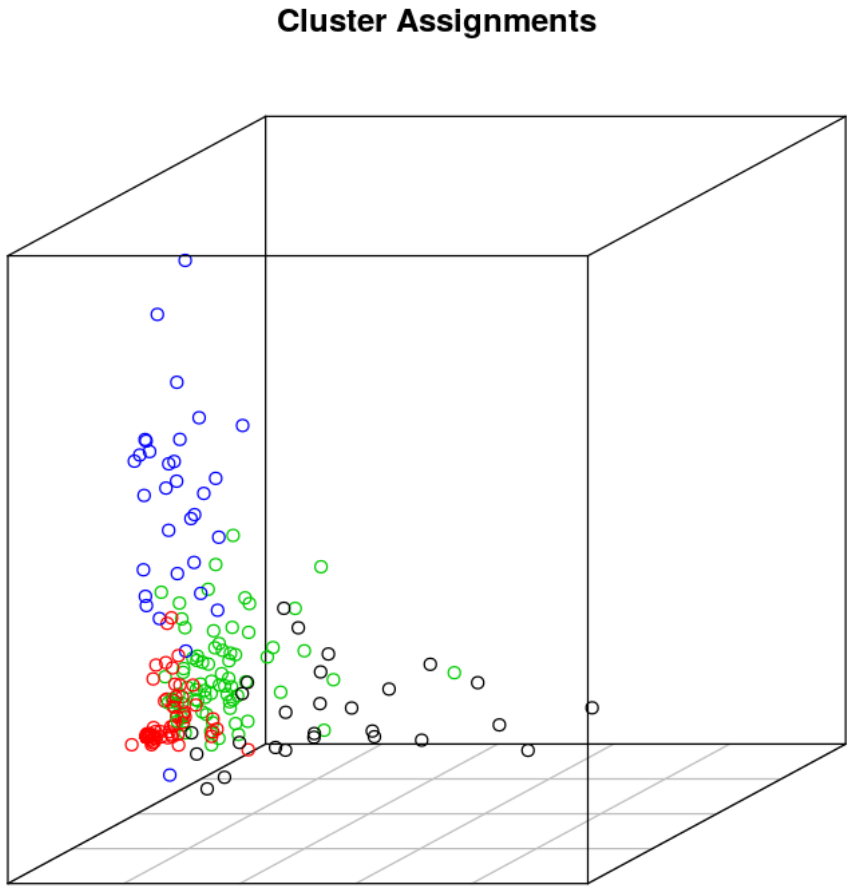


Clustering Zipcodes Given that there are four factors driving the variation in the data the complaints were clustered with four centers. Then the cluster assignments were visualized in Eigenspace to inspect the results. As shown below, the clusters are fairly well separated and the cluster assignments appear reasonable

```
In [15]: #Cluster data
         set.seed(400)
         cluster=kmeans(processed, 4)

         #Visualize cluster results
         library(scatterplot3d)
         library(rgl)
         NYCPCs = pca$scores
         scatterplot3d(NYCPCs[,3], NYCPCs[,1], NYCPCs[,2], color=cluster$cluster, x
```

```
                    tick.marks=FALSE, main='Cluster Assignments')
```



In [16]:
```
library(maptools)
library(RColorBrewer)

#Assign cluster colors to zipcodes
NYC = readShapePoly('ZIP_CODE_040114/ZIP_CODE_040114.shp')

zipcolors = data.frame(zip = NYC$ZIPCODE, color = NA)
for(i in 1:nrow(zipcolors)){
  if(zipcolors[i,1] %in% zipcodes){
    zipcolors[i,2] = cluster$cluster[which(zipcodes == zipcolors[i,1])]
  }
```

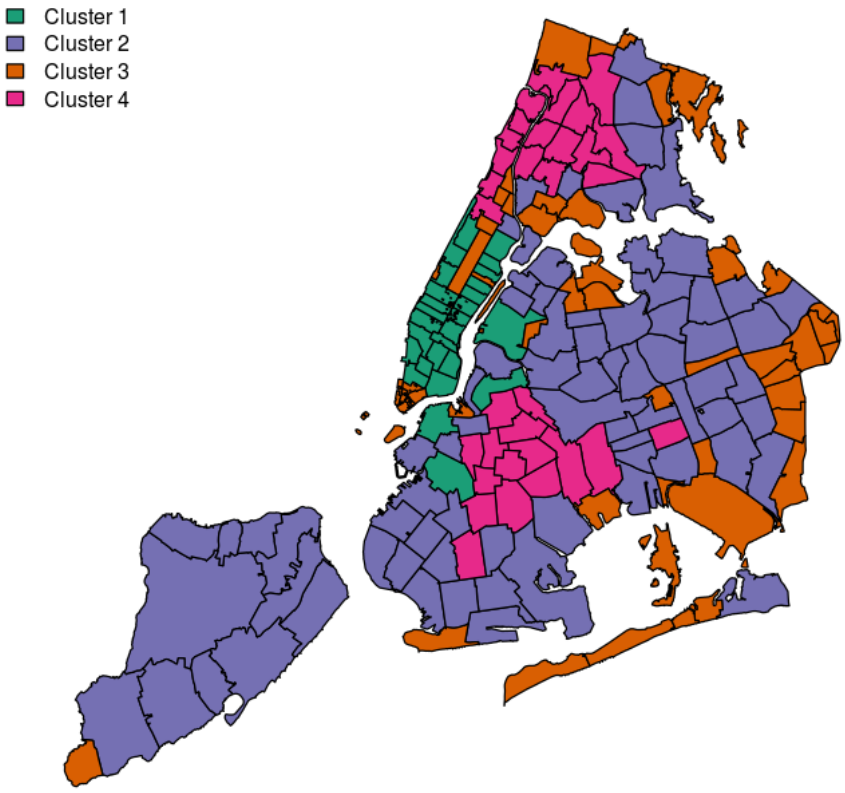
```
        }
        zipcolors$clusters = ifelse(zipcolors$color == 'NA', NA, paste0('Cluster '

        sum(is.na(zipcolors$clusters))

        #Visualize clusters on NYC map
        colors = brewer.pal(4, 'Dark2')
        plot(NYC, col=colors[zipcolors$color])
        title("NYC, by Complaints")
        legend('topleft', legend=names(table(zipcolors$clusters)), fill = names(ta
```

```
Loading required package: sp
Checking rgeos availability: FALSE
        Note: when rgeos is not available, polygon geometry          computations i
        which has a restricted licence. It is disabled by default;
        to enable gpclib, type gpclibPermit()
Warning message:
“use rgdal::readOGR or sf::st_read”
```

50



In [33]:
```
#library(tidyverse)
library(stringr)
library(lubridate)
library(leaflet)
library(DT)
library(forecast)
```

```
Error in library(forecast): there is no package called 'forecast'
Traceback:

1. library(forecast)
```

```
        2. stop(txt, domain = NA)
```

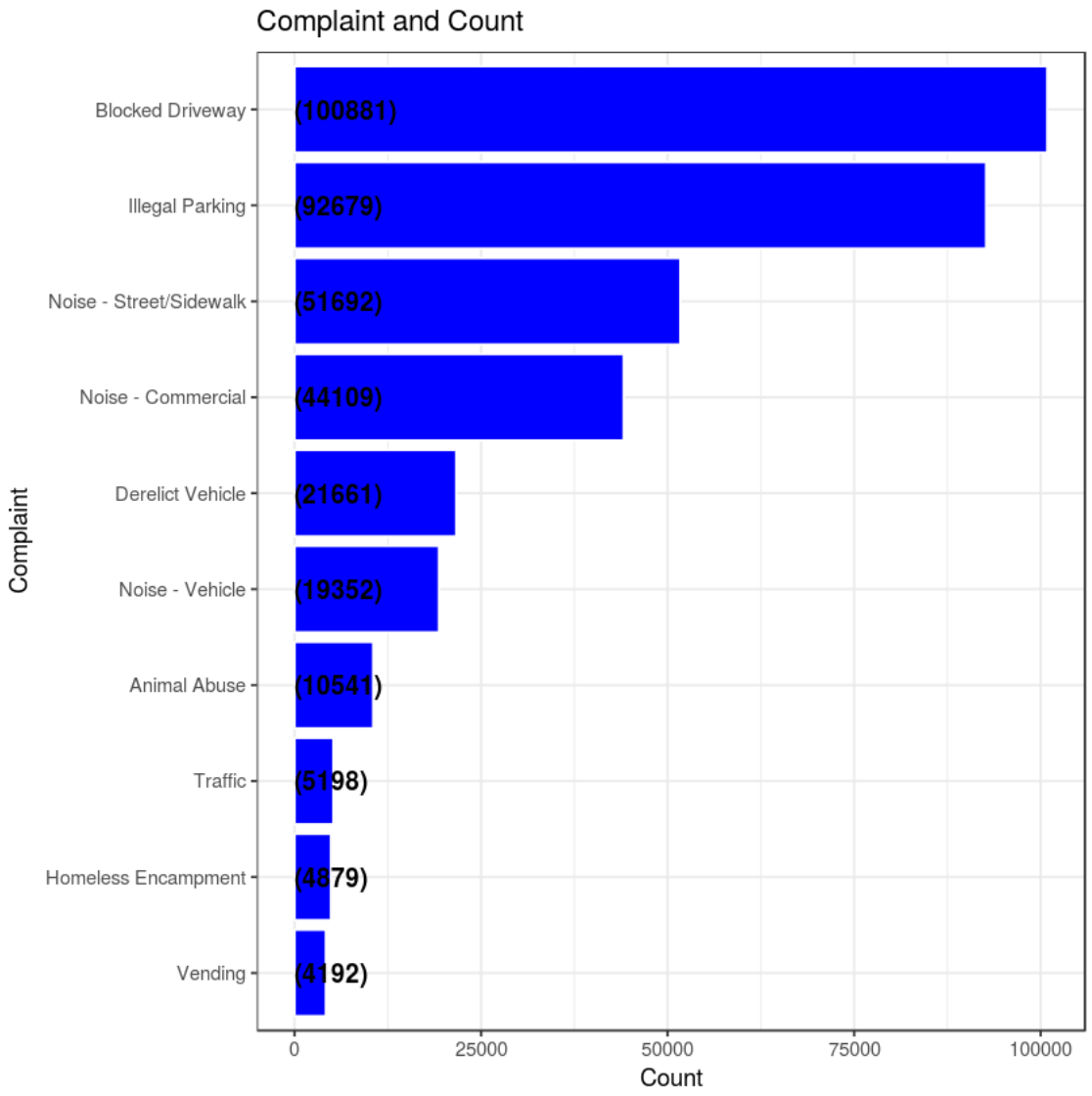
```
In [21]: NYC311 = read_csv("311.csv")

Parsed with column specification:
cols(
  .default = col_character(),
  `Unique Key` = col_integer(),
  `Incident Zip` = col_integer(),
  `X Coordinate (State Plane)` = col_integer(),
  `Y Coordinate (State Plane)` = col_integer(),
  Latitude = col_double(),
  Longitude = col_double()
)
See spec(...) for full column specifications.
Warning message in rbind(names(probs), probs_f):
"number of columns of result is not a multiple of vector length (arg 2)"Warning mes
"1 parsing failure.
row # A tibble: 1 x 5 col      row col          expected   actual file
"
```

```
In [24]: NYC311 = NYC311 %>%
           rename(ComplaintType = `Complaint Type`) %>%
           rename(CreatedDate = `Created Date`)
```

```
In [27]: NYC311 %>%
           group_by(ComplaintType) %>%
           summarise(Count = n()) %>%
           ungroup() %>%
           mutate(ComplaintType = reorder(ComplaintType,Count)) %>%
           arrange(desc(Count)) %>%
           head(10) %>%

           ggplot(aes(x = ComplaintType,y = Count)) +
           geom_bar(stat='identity',colour="white", fill = 'blue') +
           geom_text(aes(x = ComplaintType, y = 1, label = paste0("(",Count,")",sep
                     hjust=0, vjust=.5, size = 4, colour = 'black',
                     fontface = 'bold') +
           labs(x = 'Complaint',
                y = 'Count',
                title = 'Complaint and Count') +
           coord_flip() +
           theme_bw()
```



In [28]:
```
keyWordComplaint  ="Noise"

PlotComplaintCategory = function (keyWordComplaint)
{
  NYC311 %>%
  filter(str_detect(trimws(ComplaintType),keyWordComplaint)) %>%
  group_by(ComplaintType) %>%
  summarise(Count = n()) %>%
  ungroup() %>%
  mutate(ComplaintType = reorder(ComplaintType,Count)) %>%
  arrange(desc(Count)) %>%
  head(10) %>%
```

```
    ggplot(aes(x = ComplaintType,y = Count)) +
    geom_bar(stat='identity',colour="white", fill = 'blue') +
    geom_text(aes(x = ComplaintType, y = 1, label = paste0("(",Count,")",sep
              hjust=0, vjust=.5, size = 4, colour = 'black',
              fontface = 'bold') +
    labs(x = 'Complaint',
         y = 'Count',
         title = 'Complaint and Count') +
    coord_flip() +
    theme_bw()

  }

  PlotComplaintCategory(keyWordComplaint)
```

```
    Error in filter_impl(.data, quo): Evaluation error: could not find functior
Traceback:


    1. PlotComplaintCategory(keyWordComplaint)

    2. NYC311 %>% filter(str_detect(trimws(ComplaintType), keyWordComplaint)) %
.      group_by(ComplaintType) %>% summarise(Count = n()) %>% ungroup() %>%
.      mutate(ComplaintType = reorder(ComplaintType, Count)) %>%
.      arrange(desc(Count)) %>% head(10) %>% ggplot(aes(x = ComplaintType,
.      y = Count))   # at line 5-23 of file <text>

    3. withVisible(eval(quote(`_fseq`(`_lhs`)), env, env))

    4. eval(quote(`_fseq`(`_lhs`)), env, env)

    5. eval(quote(`_fseq`(`_lhs`)), env, env)

    6. `_fseq`(`_lhs`)

    7. freduce(value, `_function_list`)

    8. function_list[[i]](value)

    9. filter(., str_detect(trimws(ComplaintType), keyWordComplaint))

    10. filter.tbl_df(., str_detect(trimws(ComplaintType), keyWordComplaint))

    11. filter_impl(.data, quo)
```

In [ ]: