homework vi

Vivek Panchal,Rohit Kunjilikattil

2019-10-11

1. Introduction

This an an exploratory analysis of the NYC311 dataset. Firstly, the dataset is cleaned so that it is ready for use. It includes various visualization on borough against complaint types and against agencies. Another dataset, the NYC_Payroll dataset is cleaned and introduced. A join is performed on the summarized version of NYC311 and the NYC_Payroll database. Lastly, it includes visualization in line with the joins performed.

1.1 Loading and installing required libraries

```
library(ggplot2)
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(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyve

## v tibble  2.1.3     v purrr   0.3.2
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
library(tidyr)
install.packages('tidyverse', repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/vivek 14/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)

## Warning: package 'tidyverse' is in use and will not be installed
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

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

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(knitr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following object is masked from 'package:base':
##
##     date
```

```
library(magrittr)
```

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

## The following object is masked from 'package:ggmap':
##
##     inset

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

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

```r
library(data.table)
library(gender)
library(dplyr)
library(ggjoy)
```

```
## Loading required package: ggridges
```

```
##
## Attaching package: 'ggridges'
```

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

```
## The ggjoy package has been deprecated. Please switch over to the
## ggridges package, which provides the same functionality. Porting
## guidelines can be found here:
## https://github.com/clauswilke/ggjoy/blob/master/README.md
```

```r
library(ggridges)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(ggthemes)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

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

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Tidyverse is a r package that helps in making data tidy. Tidy data is defined as data which follows three basic rules : each variable must have its own column, each observation must have its own row and finally each value must have its own cell.One of the advantages of tidy data is it's easier to work with data that always has a consistent structure. Further, R's vectorised nature makes it particularly easy to transform tidy data.

1.2 Reading csv

```
nyc311<-fread("C:/Users/vivek 14/Downloads/311_Service_Requests_from_2010_to_Present.csv", nrows = 10000
options("scipen" = 100,"digits" = 4)
```

2. Cleaning NYC311Data

Cleaning NYC311 data to save processing power on bad data. As the complaint type is one of the most important columns,it is made as uniform as possible. We go about doing this byOther cleaning include removing any unwanted spaces, replacing hyphens with slashes and finally filtering the zipcodes to only have values of length 5.

```
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$`Complaint Type` = gsub('paint - plaster', 'paint/plaster', nyc311$`Complaint Type`)
nyc311$`Complaint Type` = gsub('general construction', 'construction', nyc311$`Complaint Type`)
nyc311$`Complaint Type` = gsub('nonconst', 'construction', nyc311$`Complaint Type`)
nyc311$`Complaint Type` = gsub('street sign - [[:alpha:]]+', 'street sign', nyc311$`Complaint Type`)
nyc311$`Complaint Type` = gsub('fire alarm - .+','fire alarm', nyc311$`Complaint Type`)
nyc311_clean2 = nyc311
```

2.1 Separating Columns

The data contains multiple columns that are of class character that have date and time values together in one column. The **separate** function from tidyr library helps in separating such columns into columns with date and time that are stored separately.

```
nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Created Date`, into = c("Created Date", "Created Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Closed Date`, into = c("Closed Date", "Closed Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Due Date`, into = c("Due Date", "Due Time"), sep = 10)

nyc311_clean2 <- nyc311_clean2 %>%
  separate(`Resolution Action Updated Date`, into = c("Resolution Action Updated Date", "Resolution Acti
```

3. Data Exploration of NYC311 Dataset

3.1 Exploration using Latitude and Longitude columns in data.

Location data is very helpful in visualization. The ggmap library in order **to find out which locations have complaints with status as "CLOSED".** The filtering is done so that the exploration gives us a sense of efficiency of the oraganization in different boroughs

Above Map is made with data that contains the selected four columns `Borough`,`Latitude`,`Longitude`,`Status`. The map used is "terrain".Some data contains rows with which have 'Boroughs' as 'Unspecified'. These rows are removed. For better readability, different boroughs are marked with different colors.

In order to visualize maps, library "ggmap" is used . With the help of this package, it is easy to retrieve map tiles from sources such as Google Maps, Open Street Maps and many more. They have the necessary tools which helps in smooth functioning for our routing. The two columns which are very essential to visualize any maps are `Latitude` and `Longitude`.

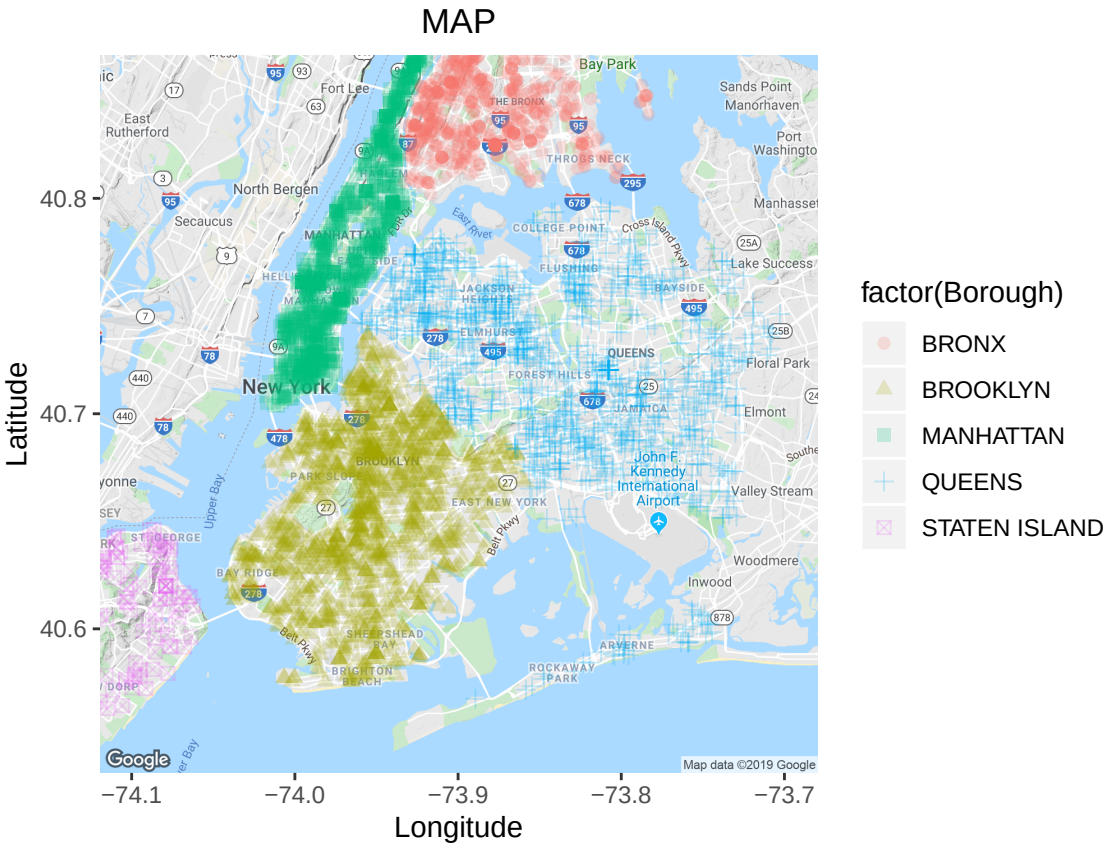
For the purpose of this visualization you need to enable google static map service. If this service was to be used before July 2017, there was no need to signup for any google map services. But now it requires setting up an API Key and enabling billing. While signing up for the service you need to copy your API key which will be added to your chunk.

```
nyc311_map<-nyc311[sample(nrow(nyc311),10000),]
nyc_map_data <- nyc311_map %>% select(24,50,51,20)

nyc_map_data<- nyc_map_data %>% filter(!str_detect(Borough, "Unspecified"))
nyc_map_data<-na.omit(nyc_map_data)
xtab<-dplyr::filter(nyc_map_data,Status == "Closed")

#library(devtools)
#devtools::install_github("fresques/ggmap")
# devtools::install_github("dkahle/ggmap")

key<-'AIzaSyD8Zu8HBooWB2MP9yHqbXOA5ERj5oTQ-mA'
register_google(key ="AIzaSyD8Zu8HBooWB2MP9yHqbXOA5ERj5oTQ-mA")
nyc_map = get_map(location = c(lon = -73.9, lat = 40.7),
                  zoom = 11, maptype="terrain")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=40.7,-73.9&zoom=11&size=640x640&scale=
```

```
map<-ggmap(nyc_map) + geom_point(data=xtab,aes(x=xtab$Longitude,y=xtab$Latitude,shape =factor(Borough),
map
```

```
## Warning: Removed 391 rows containing missing values (geom_point).
```



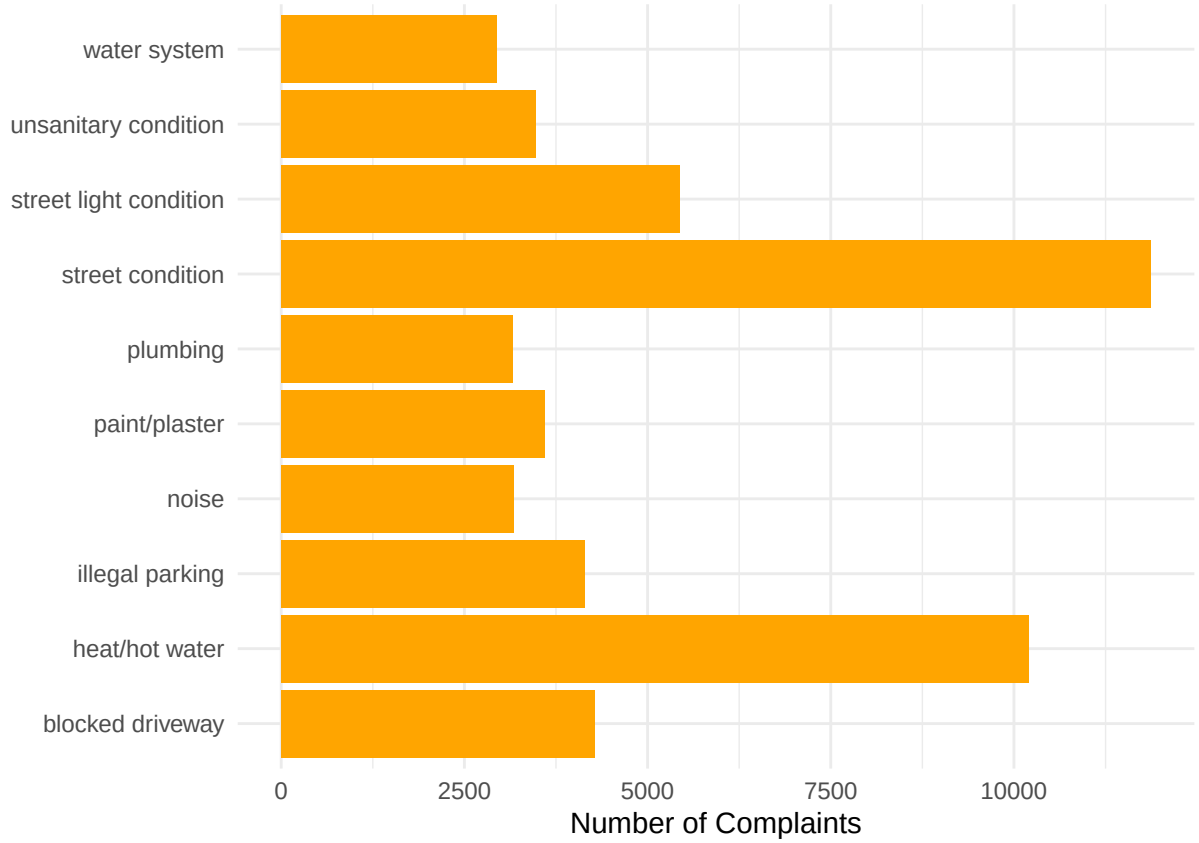
3.2 Exploring relationship between complaints in different Borough

Next, an exploration to demonstrate the relationship between top complaints and the boroughs. This would give an insight about what complaints are registered more frequently in which boroughs.

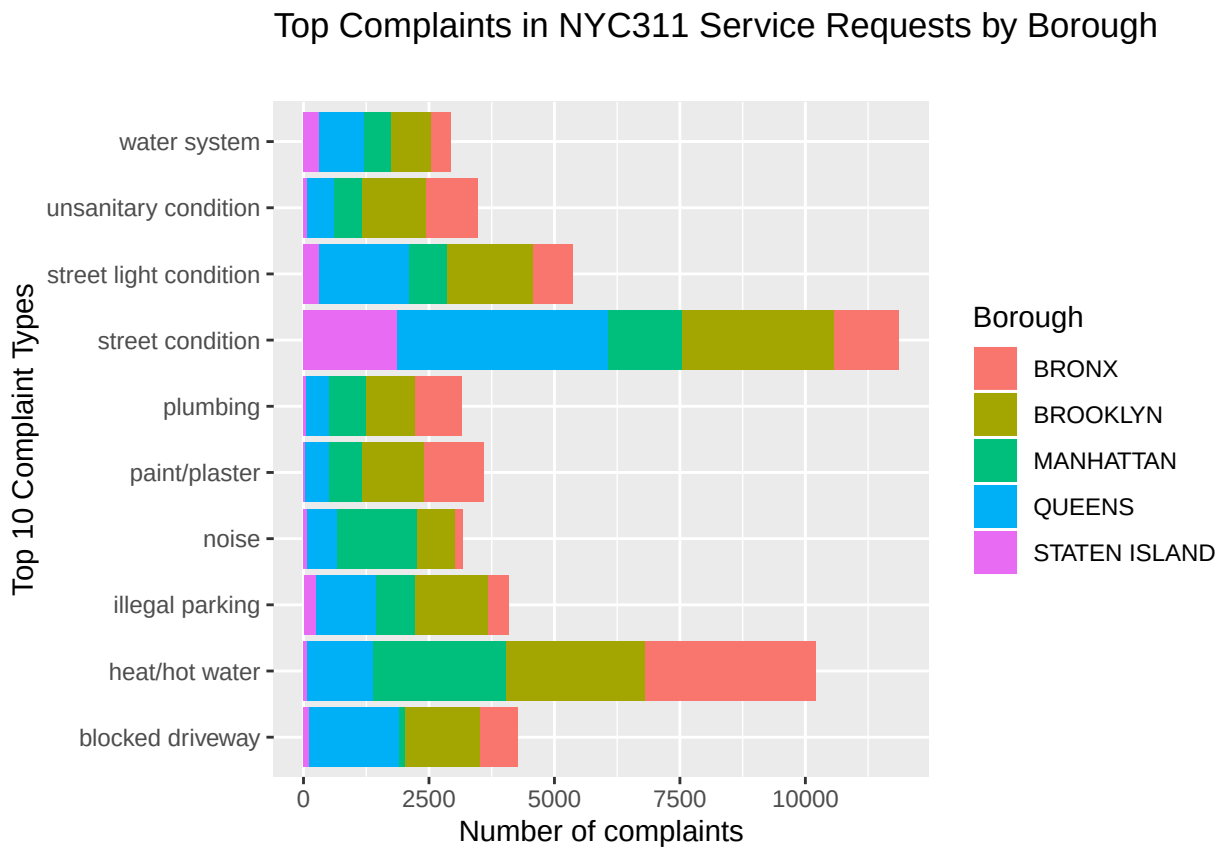
The first figure is barplot of the top complaints to showcase the most registered complaints in the nyc311 data. The second graph is an an overview of distribution of complaints across different boroughs, while the third graph is a closer look at the situation in each borough.The third graph is divided by the color scheme based on agencies. This can help in showing which agency is responsible for handling which type of complaint.

```
all_complaints <-  nyc311 %>% select(2, 6, 24)

all_complaints$`Complaint Type`[grepl("^Noise.*", all_complaints$Complaint.Type)] <-
"Noise"

all_complaints_temp <- all_complaints %>%
group_by(`Complaint Type`) %>%
summarise(count = n()) %>%
arrange(desc(count))
top10_complaints <- top_n(all_complaints_temp, 10, count)
ggplot(top10_complaints) + geom_bar(
aes(x = top10_complaints$`Complaint Type` , y = top10_complaints$count),
fill = "Orange",
stat = "identity"
```

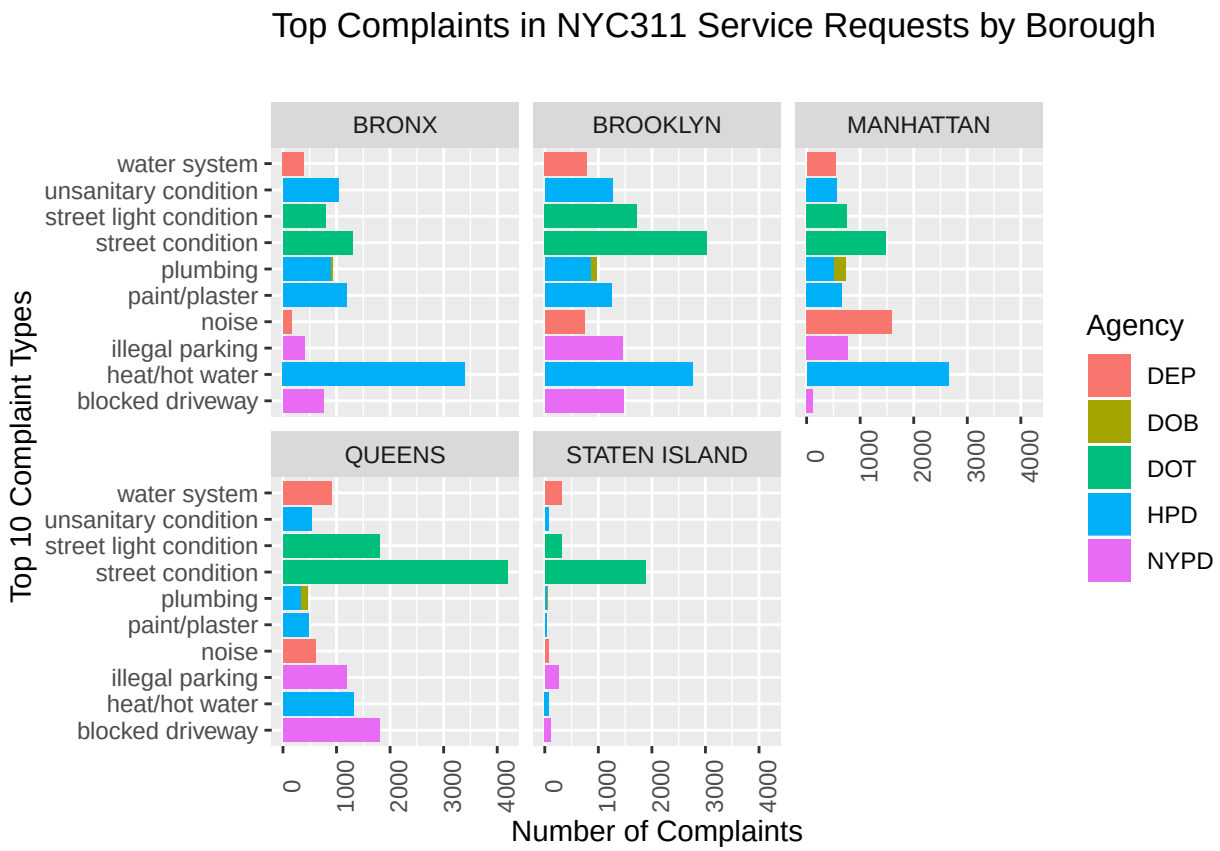
```
) + theme_minimal() +
xlab("") + ylab("Number of Complaints") + coord_flip()
```




```
nyc311_subset <- subset(nyc311,`Complaint Type` %in% count(nyc311,`Complaint Type`,sort = T)[1:10,]$`Co
nyc311_subset <- nyc311_subset %>% select(`Complaint Type`,Borough,Status,Agency)  %>% group_by(Borough
nyc311_subset <- nyc311_subset %>% filter(!str_detect(Borough, "Unspecified"))
ggplot(nyc311_subset, aes(`Complaint Type`)) + geom_bar(stat = "count")  + labs(x = "Top 10 Complaint Ty
                              y = "Number of complaints",
                              title = "Top Complaints in NYC311 Service Requests by Borough",
                              subtitle = "")  +coord_flip()  + aes(fill=Borough)
```




```
ggplot(nyc311_subset, aes(`Complaint Type`)) +
  geom_bar(stat = "count") +
  facet_wrap(~Borough) + labs(x = "Top 10 Complaint Types",
                              y = "Number of Complaints",
                              title = "Top Complaints in NYC311 Service Requests by Borough ",
                              subtitle = "") + coord_flip()  + aes(fill=Agency) + theme(axis.text.x = el
```



4. Reading a dataset from a url

Next, the NYC_Payroll data is read directly from the url to work on it without having to download the csv file. This dataset contians the salary, pay rate and total compensation of every New York City employees from year 2014 till 2017.This dataset provides columns for fiscal year, agency they work for, borough they are working in. Some useful explorations as who gets paid how much and for what are done. The source dataset thatis being extracted from url follows the rules of tidydata.

```
nyc_payroll <- fread("https://data.cityofnewyork.us/api/views/k397-673e/rows.csv?accessType=DOWNLOAD")
nyc_payroll<- nyc_payroll[which (nyc_payroll$`Work Location Borough` == 'MANHATTAN' |nyc_payroll$`Work
nycpayroll_nodups<-distinct(nyc_payroll)
isTRUE(all.equal(nycpayroll_nodups,nyc_payroll))
```

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

5. Exploring NYCPayroll Dataset

There are 3 types of pay basis i.e Per hour, Per day and Per annum. Exploration of it is done by finding the total count of such employees working in New York city.

```
nyc_payroll %>% dplyr::group_by(`Pay Basis`) %>% dplyr::summarise(n = n())
```

```
## # A tibble: 4 x 2
##   `Pay Basis`            n
##   <chr>              <int>
## 1 per Annum        1579636
## 2 per Day           607849
## 3 per Hour          507036
## 4 Prorated Annual    12986
```

The total count of records by year and displaying first five records of the result set.

```
nyc_payroll %>% dplyr::group_by(`Fiscal Year`) %>% dplyr::summarise(n = n())
```

```
## # A tibble: 5 x 2
##   `Fiscal Year`      n
##           <int>  <int>
## 1          2015 547677
## 2          2016 520583
## 3          2017 541957
## 4          2018 525778
## 5          2019 571512
```

```
#Showing first five records of the nyc_payroll dataset.
nyc_payroll %>%
  head(5)%>%
  kable()
```

Fiscal Year	Payroll Number	Agency Name	Last Name	First Name	Mid Init	Agency S
2019	67	ADMIN FOR CHILDREN'S SVCS	SIMMONS	DONALD		07/04/20
2019	67	ADMIN FOR CHILDREN'S SVCS	MOHAMMED	KATHIE	S	10/24/20
2019	67	ADMIN FOR CHILDREN'S SVCS	MCRAE	TANESIA	M	09/11/20
2019	67	ADMIN FOR CHILDREN'S SVCS	ROZON	GINNETTE		08/14/20
2019	67	ADMIN FOR CHILDREN'S SVCS	LOPEZ	RAFAEL		01/17/20

The dataset is split into two parts, agencies that have income less than 10k and over 10k.For representation of agencies that have **Base Salary** more than 10K use of kable() of knitr library is done to display data in concise manner

```
byagencysub10<- nyc_payroll %>% count(nyc_payroll$`Agency Name`, nyc_payroll$`Fiscal Year`) %>% ungroup
kable(head(byagencysub10,10))
```

nyc_payroll$Agency Name	nyc_payroll$Fiscal Year	n
DEPT OF ED PER SESSION TEACHER	2019	104676
DEPT OF ED PEDAGOGICAL	2019	100050
DEPT OF ED PEDAGOGICAL	2018	96539
DEPT OF ED PER SESSION TEACHER	2015	93342
DEPT OF ED PEDAGOGICAL	2017	93202
DEPT OF ED PEDAGOGICAL	2016	90895
DEPT OF ED PEDAGOGICAL	2015	90475
DEPT OF ED PER SESSION TEACHER	2018	77471

nyc_payroll$Agency Name	nyc_payroll$Fiscal Year	n
DEPT OF ED PER SESSION TEACHER	2017	76326
DEPT OF ED PER SESSION TEACHER	2016	74699

```
#Agency representation over 10k

byunion10plus<- nyc_payroll %>% count(nyc_payroll$`Agency Name`,nyc_payroll$`Title Description`,nyc_pay
kable(head(byunion10plus,10))
```

nyc_payroll$Agency Name	nyc_payroll$Title Description	nyc_payroll$Base Salary	n
DEPT OF ED PER SESSION TEACHER	TEACHER- PER SESSION	33.18	414765
BOARD OF ELECTION POLL WORKERS	ELECTION WORKER	1.00	168769
DEPT OF ED PER DIEM TEACHERS	TEACHER-GENERAL ED	110.29	54286
POLICE DEPARTMENT	POLICE OFFICER	85292.00	40273
POLICE DEPARTMENT	POLICE OFFICER	78026.00	16421
FIRE DEPARTMENT	FIREFIGHTER	85292.00	15991
POLICE DEPARTMENT	POLICE OFFICER	76488.00	13998
DEPARTMENT OF CORRECTION	CORRECTION OFFICER	85292.00	9980
DEPT OF ED PARA PROFESSIONALS	SUBSTITUTE ED PARA	154.77	8609
DEPT OF ED PARA PROFESSIONALS	SUBSTITUTE ED PARA	157.87	8380

Calculation of the average salary of each agency is done in terms of `Regular Gross Paid`. For this analysis, dplyr functions are used. Group by is done on the `Fiscal year` column of nycpayroll dataset.Mutate() helped in creating a new variable from the data. The summarise_at() affects variables selected with character vector or vars().vars() is nothing but a character vector of column names, a numeric vector of column positions.

```
salbyagency <- nyc_payroll %>%
  group_by(`Fiscal Year`) %>%
  mutate(Count = n()) %>%
  group_by(`Agency Name`, `Count`) %>%
  summarise_at(vars(`Regular Gross Paid`), funs(mean(., na.rm = TRUE))) %>%
  arrange(desc(`Regular Gross Paid`))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
kable(head(salbyagency, 10))
```

Agency Name	Count	Regular Gross Paid
DOE CUSTODIAL PAYROL	520583	140067
DOE CUSTODIAL PAYROL	547677	112818
BRONX COMMUNITY BOARD #1	571512	111942
BRONX COMMUNITY BOARD #3	571512	110221
BRONX COMMUNITY BOARD #3	541957	109132
BRONX COMMUNITY BOARD #3	525778	106822
BRONX COMMUNITY BOARD #1	541957	106387
BRONX COMMUNITY BOARD #1	525778	106382
FINANCIAL INFO SVCS AGENCY	571512	104313
BRONX COMMUNITY BOARD #3	547677	103340

In our dataset we have a column `Agency Start Date` so what we did is we created a new column `yrs` from it which has the year of that particular person when he started working. We made use of POSIXct for this purpose. Further to calculate experience we took out present date and from that we subtracted agency start date.We performed group by on Boroughs so that we get average years of experience of each Borough.

```
nyc_payroll$`yrs` <- year(as.POSIXct(nyc_payroll$`Agency Start Date`, format="%d/%m/%Y"))
today <- format(Sys.Date(), "%Y")
nyc_payroll$exp <- as.numeric(today)-as.numeric(nyc_payroll$yrs)
salbyexp<- nyc_payroll%>%
  group_by(`Work Location Borough`) %>%
  select(`Work Location Borough`,exp) %>%
  dplyr::summarise_at(vars(`exp`), funs(mean(., na.rm=TRUE)))
kable(head(salbyexp,10))
```

Work Location Borough	exp
BRONX	12.06
BROOKLYN	12.67
MANHATTAN	12.98
QUEENS	12.08

As we are going to join our nyc payroll dataset with nyc311 open data set we have common columns such as Boroughs and Agency for that purpose we renamed our column name of `Work Location Borough` of nyc_payroll dataset to `Borough` so that we can perform merge operation.

```
nyc_pr_br<-as.data.table(nyc_payroll) %>% {setnames(., old = "Work Location Borough", new = "Borough")[

nyc_payroll %>% filter(`Pay Basis`=='per Annum') %>% filter(`Work Location Borough`!="") %>% group_by(`
```

```
## # A tibble: 4 x 2
##   `Work Location Borough`   count
##   <chr>                     <int>
## 1 MANHATTAN                1027685
## 2 QUEENS                    232771
## 3 BROOKLYN                  209057
## 4 BRONX                     110123
```

6. Gender Prediction

R has a library gender which helps in determining gender of a person from the first name. It takes in arguments such as name, years and method. The "ssa" method takes up name from SSN administration of new born baby data.

```
nyc_payroll<-as.data.frame(nyc_payroll)
all_names <- nyc_payroll %>% select(`First Name`) %>% sapply(as.character) %>% as.vector()
min_year<-rep(1950,length(all_names))
max_year<-rep(2012,length(all_names))
temp_df <- data_frame(first_names = all_names,min_year = min_year,max_year = max_year)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

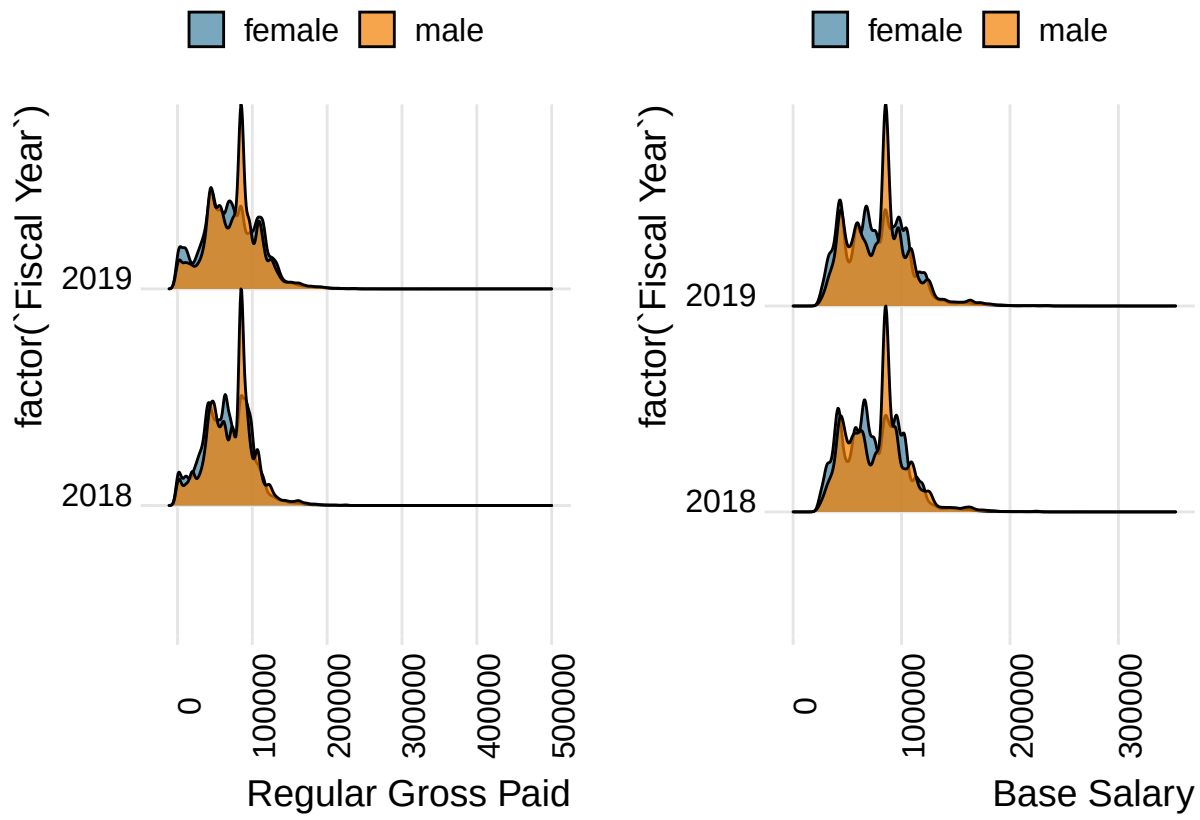
```
results_df <- gender_df(temp_df, name_col = "first_names", year_col = c("min_year", "max_year"), method

#bind input and output
res<-temp_df %>% left_join(results_df, by = c("first_names" = "name", "min_year" = "year_min", "max_yea

nyc_payroll<-cbind(nyc_payroll, res)
```

6.1 Salary Evolution over Years

As the dataset is large,it is hard for data to be distributed normally, thus taking the mean or median of a distribution may lead to different resultsets.

It is observed that over the years, the distribution hasn't changed much. Hence the median value may be more appropriate to describe the salary feature.

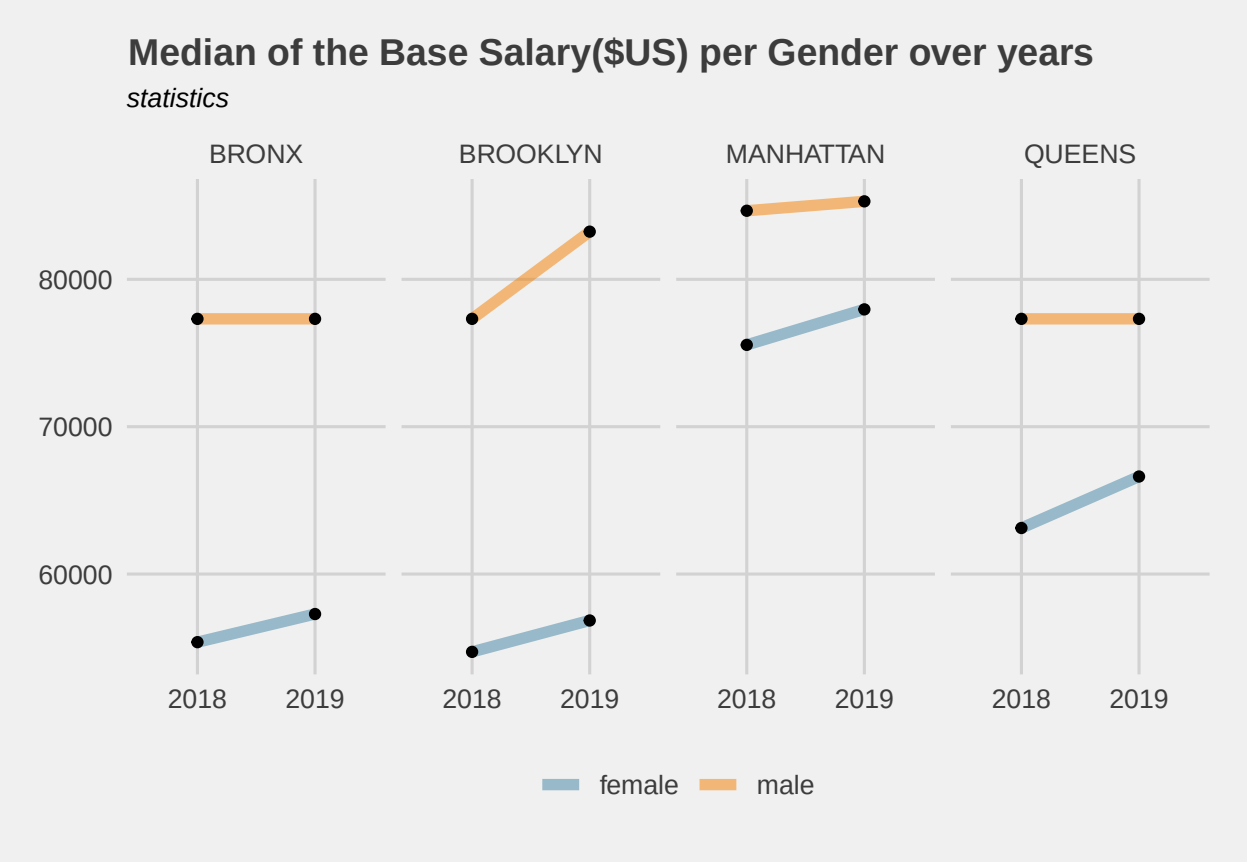
```
g1<-nyc_payroll %>% dplyr::filter(`Pay Basis`=='per Annum') %>% na.omit() %>% ggplot(aes(x= `Regular Gr

g2<-nyc_payroll %>% dplyr::filter(`Pay Basis`=='per Annum') %>% na.omit() %>% ggplot(aes(x= `Base Salary

grid.arrange(g1,g2,ncol=2)
```



6.2 Breakdown by Borough

Some Boroughs shows data for only 1 gender, the reason is because of the low statisitic for these Boroughs. Manhattan being the Borough with most of the data, it proves to be the most trustable.

```
nyc_payroll %>% filter(`Pay Basis`=='per Annum') %>%
  na.omit() %>%
  group_by(`Fiscal Year`, `gender`, `Work Location Borough`) %>%
  summarize(count = n(),
            medianSalary = median(`Base Salary`)) %>%
  ggplot(aes(x=factor(`Fiscal Year`),y=`medianSalary`,group=`gender`)) +
  geom_line(aes(color=`gender`),size=2,alpha=.5) + geom_point(size=1.5) +
  scale_color_manual(name="",values = c("#3f83a3","#f48000")) + theme_fivethirtyeight() +
  theme(plot.title=element_text(face="bold",hjust=.012,vjust=.8,colour="#3C3C3C",size=14),
        plot.subtitle=element_text(size=10, hjust=0, face="italic", color="black")) +
        labs(
        title="Median of the Base Salary($US) per Gender over years",
        subtitle="statistics") + facet_wrap(~`Work Location Borough`,ncol = 5)
```



7. Joining Dataset

Join is performed between a summarized version of nyc311, summarized on borough and nyc_pr_exp, which was summarized based on average experience in Boroughs. The final result of this join is a table containing the total types of complaint and the total average experience in that borough. This can help in visualizing if the boroughs with a variety of complaints have the required experience to handle it. So more the type of complaints, more should be the average experience.

```
nyc_pr_exp<-as.data.table(salbyexp) %>% {setnames(., old = "Work Location Borough", new = "Borough")[]}
nyc_ct <- nyc311_clean2[,c(8,28)]
nyc_ct <-  nyc_ct%>% dplyr::group_by(`Complaint Type`) %>% dplyr::summarise(n = n())
nyc_br <- nyc311_clean2 %>% dplyr::group_by(`Borough`) %>% dplyr::summarise(n = n())
merged<- inner_join(all_complaints_temp,all_complaints, by ='Complaint Type')
merged<- merged %>% filter(!str_detect(Borough, "Unspecified"))

merged <- merged %>% group_by(Borough, `Complaint Type`) %>% summarise(n = mean(count))
merged <- merged %>% group_by(`Complaint Type`) %>% filter(n()>1)
final_merge <- inner_join(nyc_pr_exp,merged, by='Borough')
final_merge <- final_merge %>% group_by(Borough, `Complaint Type`) %>% summarise(n = mean(exp))
final_merge2 <- final_merge %>% group_by(`Complaint Type`) %>% summarise(n = n())
final_join <- inner_join(final_merge,final_merge2, by ='Complaint Type')

final_join2 <- final_join %>% group_by(`Borough`) %>% summarise(Total_Avg_Exp = mean(n.x), Total_Comp_Ty
```

```
merged <- distinct(merged,x)
```

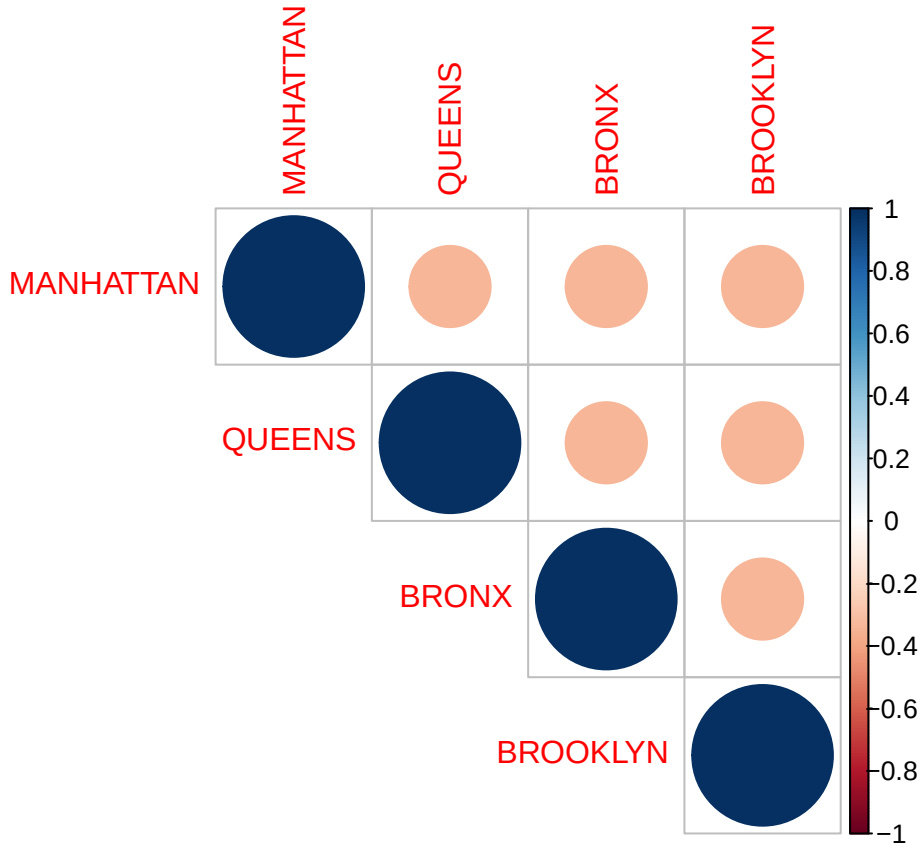
```
## Warning: Trying to compute distinct() for variables not found in the data:
## - `x`
## This is an error, but only a warning is raised for compatibility reasons.
## The operation will return the input unchanged.
```

```
merged <- na.omit(merged)

head(final_join2)
```

```
## # A tibble: 4 x 3
##   Borough   Total_Avg_Exp Total_Comp_Types
##   <chr>             <dbl>            <dbl>
## 1 BRONX              12.1                4
## 2 BROOKLYN           12.7                4
## 3 MANHATTAN          13.0                4
## 4 QUEENS             12.1                4
```

This corrplot is the visulaization of the relationship between the Boroughs and the average experience in the boroughs.

```
nyc311_corr <- final_join %>% select( 1,3)
new_table2 <- table(melt(nyc311_corr, id.var="Borough"))
new_table2<- as.data.table(new_table2)
nyc311_corr2<-new_table2 %>% select( 1, 4)
wide_table <- spread(new_table2,'Borough',N)
wide_table <- wide_table[,3:6]
resultfinal <- cor(wide_table, use = "complete.obs")
p <- corrplot(resultfinal, type="upper", order="hclust")
```

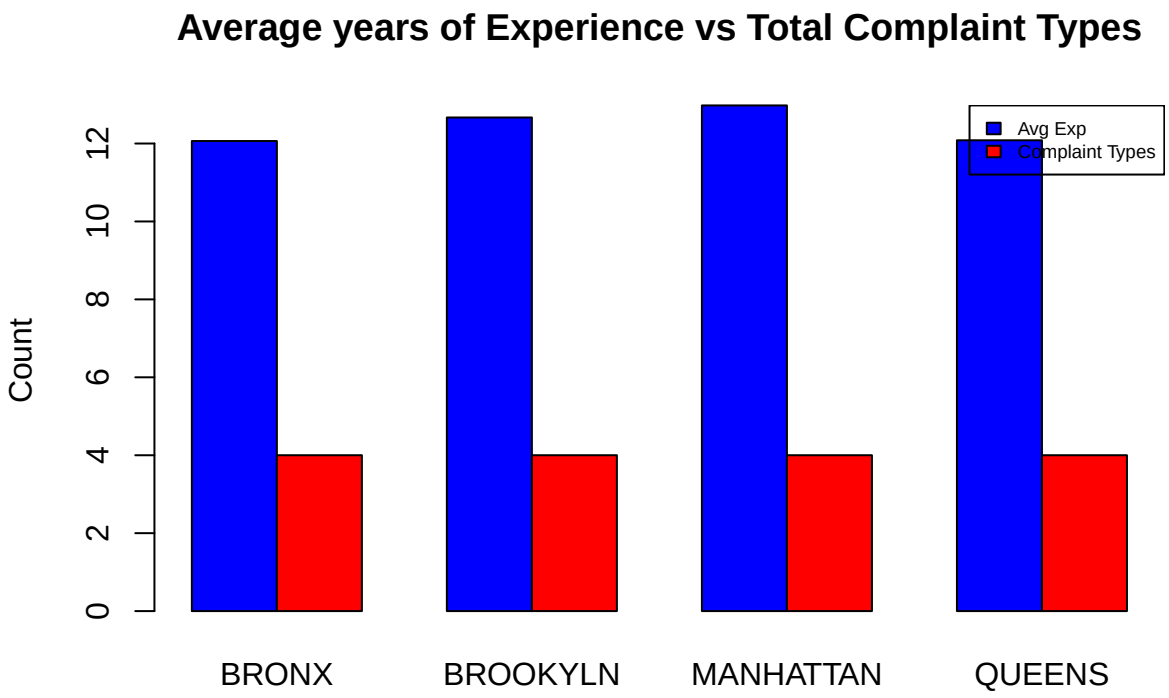



```
p
```

```
##           MANHATTAN  QUEENS   BRONX BROOKLYN
## MANHATTAN    1.0000 -0.3333 -0.3333  -0.3333
## QUEENS      -0.3333  1.0000 -0.3333  -0.3333
## BRONX       -0.3333 -0.3333  1.0000  -0.3333
## BROOKLYN    -0.3333 -0.3333 -0.3333   1.0000
```

This is a barplot showcasing the different average experience and the total number of complaint types in each borough. It can help to identify the borough which have a variety of complaints and yet have low average experience. The government can take special care to make sure that the complaints in such boroughs are handled efficiently.

```
test <- rbind(final_join2$Total_Avg_Exp,final_join2$Total_Comp_Types)
bnames <- c("BRONX","BROOKYLN","MANHATTAN","QUEENS")
plot<-barplot(test, beside=TRUE, axisnames=TRUE,
              main = 'Average years of Experience vs Total Complaint Types',
              ylab = 'Count',
              col=c('blue', 'red'),
              names.arg=bnames)

legend("topright", cex=0.6,  c('Avg Exp' , 'Complaint Types'), fill=c('blue', 'red'))
```



8. Conclusion

This report is an overview of various exploratory analysis performed on the NYC311 dataset. The analysis shows the differenct relations between aspects of the dataset like, relation between boroughs and complaint types, between complaints and location data and many others. Further, there is an exploration of a relatable dataset, the NYCPayroll dataset. This dataset is summarized into a dataset with borough and the total average experience across the boroughs. It is then joined to a summarized version of the NYC311 dataset containing boroughs and total complaint types. Finally, the report contains some visualizations on the joined data.

9. Appendix

1. Fiscal Year - It indicates the tax year in which the job was obtained.
2. Agency Name - The payroll agency the employee works for.
3. Work Location Borough - It indicates employee's work location.
4. Base Salary - Base salary the employee obtains.
5. Regular Gross Paid - Amount paid to employee during fiscal year.
6. Agency Start Date - Date on which employee started working for agency.
7. exp - The average experience of employees of the specified Borough.