Calgary Traffic

Siavash Fard

August 17, 2020

Introduction

Calgary is a city in the western Canadian province of Alberta. The city had a population of 1,285,711 in 2019, making it the largest Alberta's city, and third largest municipality in Canada. According to Statistics Canada, average commute to work for Calgarians is about 27 minutes making it the sixth slowest commute among all cities in Canada. The commute duration may depend on several factors such as roads conditions, urban planning, infrastructure, traffic incidents, etc. According to Statistics Canada, the average fatalities and injuries for Canadian drivers in 2017 are 7.1 and 595.6 (per 100,000 licensed drivers) respectively. Albertans fatalities and injuries in 2017 were 9.5 (fourth place after Youkan, Prince Edward, Saskatchewan, and North West Territories) and 539.4 (eighth place) respectively. Even though Calgary is usually ranked one of the best cities in Canada for driving (or least congested), congestion and traffic remain the top concerns in the city satisfaction survey. In this report, I tried to study the effect of different variables (e.g. traffic signs, signals and volume) on the traffic incidents in the city of Calgary. Unfortunately, locations of pedestrian incidents were not available on the City of Calgary database. As a result, only vehicle collisions were studied in this report. Finally, several machine learning algorithms were used to predict the incidents rate. RSME was used as our loss function to evaluate the predictions based on the validation set.

Data

The data for this report has been collected from the City of Calgary public database (https://data.calgary.ca/).

```
# Our first dataset is the pedestrains collision injuries data from the City of Calgary
# the dataset contains 76 rows and 4 columns and information about the year the incident occurred
# the count of pedestrain injuries as well as admission and injury types. Null values
# represent counts of 1-4 injuries.

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/2msm-bgzt/rows.csv?accessType=DOWNLOAD
", dl)

ped_collision <- read.csv(dl, header = TRUE, sep = ",")

# Second dataset is the traffic signs in Calgary. The dataset contains information about the
# traffic signs in Calgary such as type of sign, sign text, size, etc.
# The dataset contains 207,564 rows and 21 columns

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/u6ce-yibw/rows.csv?accessType=DOWNLOAD
```

```
", dl)

traffic_signs <- read.csv(dl, header = TRUE, sep = ",")

# Next dataset is Calgary Police Service Office Locations. The dataset only cotians 12 rows
# and 7 columns.

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/ap4r-bav3/rows.csv?accessType=DOWNLOAD
", dl)

police_locations <- read.csv(dl, header = TRUE, sep = ",")

# Out next dataset is traffic signals. The dataset contains information about traffic signals
# location, whether the signal is equipped with electrical count down timer, whether the
# signal is equipped with electrical push button, type of traffic signal, etc.
# The dataset contains 1,536 rows and 12 columns.

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/qr97-4jvx/rows.csv?accessType=DOWNLOAD
", dl)

traffic_signals <- read.csv(dl, header = TRUE, sep = ",")

# Next dataset is traffic volumne in Calgary. The dataset contains information such as shape length
# and traffic volume in that multilinestring. The dataseet contains 1,769 rows and 5 columns

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/wwf6-cpsg/rows.csv?accessType=DOWNLOAD
", dl)

traffic_volume <- read.csv(dl, header = TRUE, sep = ",")

# Traffic incidents dataset contains information such as the location, count and type of the
# incident. The dataset contains 19,994 rows and 10 columns. This dataset has missing data for
# July and August 2019.

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/35ra-9556/rows.csv?accessType=DOWNLOAD
", dl)

traffic_incidents <- read.csv(dl, header = TRUE, sep = ",")

# Last dataset is traffic camera locations. The dataset contains 119 rows and 7 columns.

dl <- tempfile()
download.file("https://data.calgary.ca/api/views/k7p9-kppz/rows.csv?accessType=DOWNLOAD
", dl)

traffic_cameras <- read.csv(dl, header = TRUE, sep = ",")

# None of these datasets contain null values that could affect the data. For example,
# the traffic_signals dataset contains 1.536 null values but none of these are the
```

```
# location or type of the signal but they are all ACCESSIBLE.PEDESTRIAN.SIGNAL. Only
# null values that could affect the data are pedestrian collision coutns which will not
# be included in our final model because the dataset does not have the location of the
# incidents occurred.
```

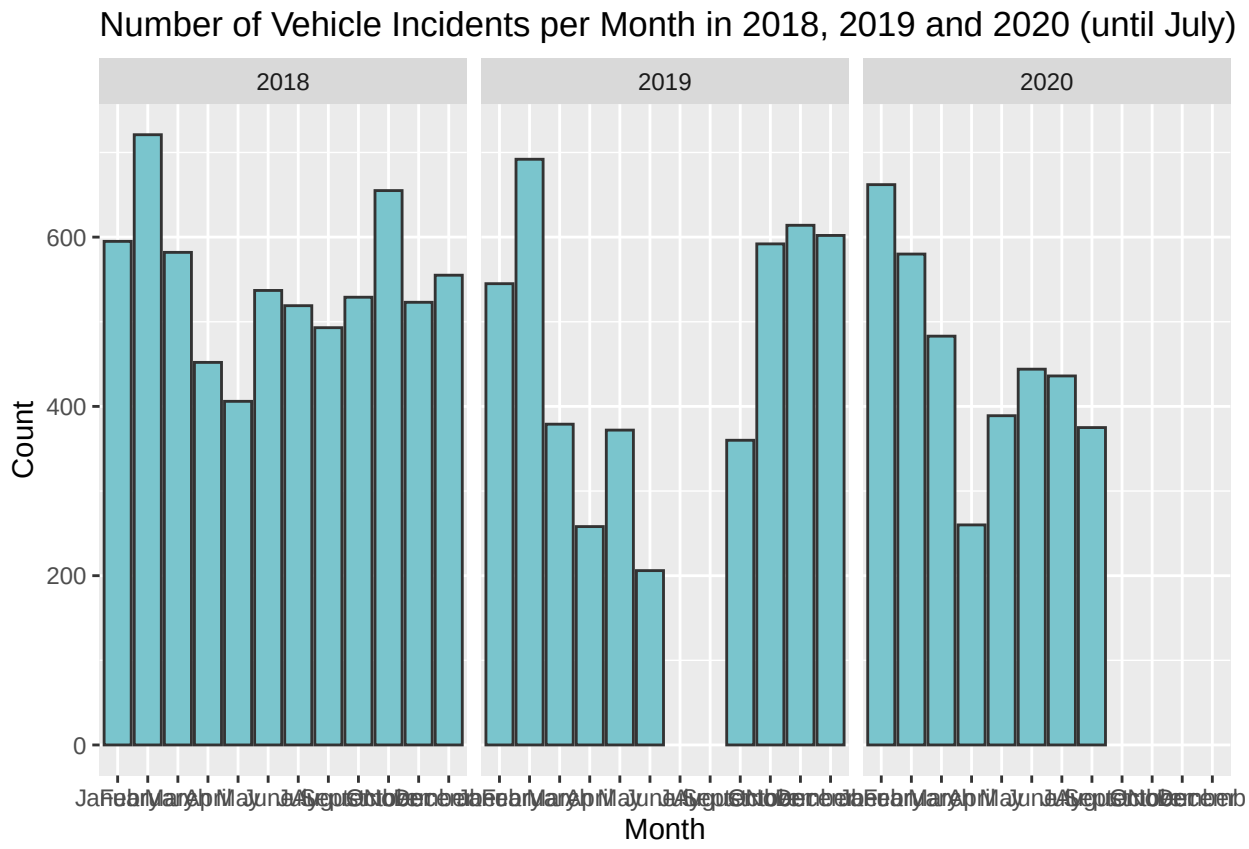
Data Cleaning and Wrangling

Since the locations of pedestrian collision are not available, this dataset has not been included in our modelling. The START_DT column in traffic incidents need to be converted to a date format. Unfortunately, there is some missing data in 2016 and 2019. As a result, only data for 2018 were included in the final dataset. There have been 6,567 incidents in 2018 out of which 1,081 involved single vehicles, and 3,739 involved two vehicles. Only signs and signals installed prior to 2018 were, therefore, considered in the final dataset. There are 75 signals and 6,048 signs installed after 2018.

```
# Extracting the longitude and latitude from the point object
traffic_signs$longitude <- as.numeric(gsub(".*?([-]*[0-9]+[.][0-9]+).*", "\\1", traffic_signs$POINT))
traffic_signs$latitude <- as.numeric(gsub(".* ([-]*[0-9]+[.][0-9]+).*", "\\1", traffic_signs$POINT))
traffic_signs$DATE <- as.Date(traffic_signs$INSTDATE)

# Extracting the dates from the incidents dataframe
traffic_incidents$DATE <- as.Date(traffic_incidents$START_DT, "%m/%d/%Y %H:%M:%S")
traffic_incidents$MONTH <- months(traffic_incidents$DATE)
traffic_incidents$MONTH <- factor(traffic_incidents$MONTH, levels=month.name)

# To incorporate the traffic volume data into our final dataset, I tried to extract the first and last
# pair of coordinates (beginning and end of the line) instead of them


long_0 <- as.numeric(gsub("\\(","",word(traffic_volume$multilinestring, 2)))
lat_0 <- as.numeric(gsub(",","",word(traffic_volume$multilinestring, 3)))
long_1<- as.numeric(word(traffic_volume$multilinestring, -2))
lat_1 <- as.numeric(gsub("\\)","",word(traffic_volume$multilinestring, -1)))

traffic_volume$long_0 <- long_0
traffic_volume$lat_0 <- lat_0
traffic_volume$long_1 <- long_1
traffic_volume$lat_1 <- lat_1
```

```
conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")
conflict_prefer("mutate", "dplyr")
conflict_prefer("year", "data.table")


# Incidents data for July and August of 2019 are missing.
traffic_incidents %>%
  group_by(MONTH, Year = year(DATE)) %>%
  summarise(Count = n()) %>%
  filter(Year %in% c(2018, 2019, 2020)) %>%
  ggplot(aes(x = MONTH ,y=Count)) +
  geom_bar(stat="identity", fill = "cadetblue3", color = "grey20") +
  ggtitle("Number of Vehicle Incidents per Month in 2018, 2019 and 2020 (until July)") +
```

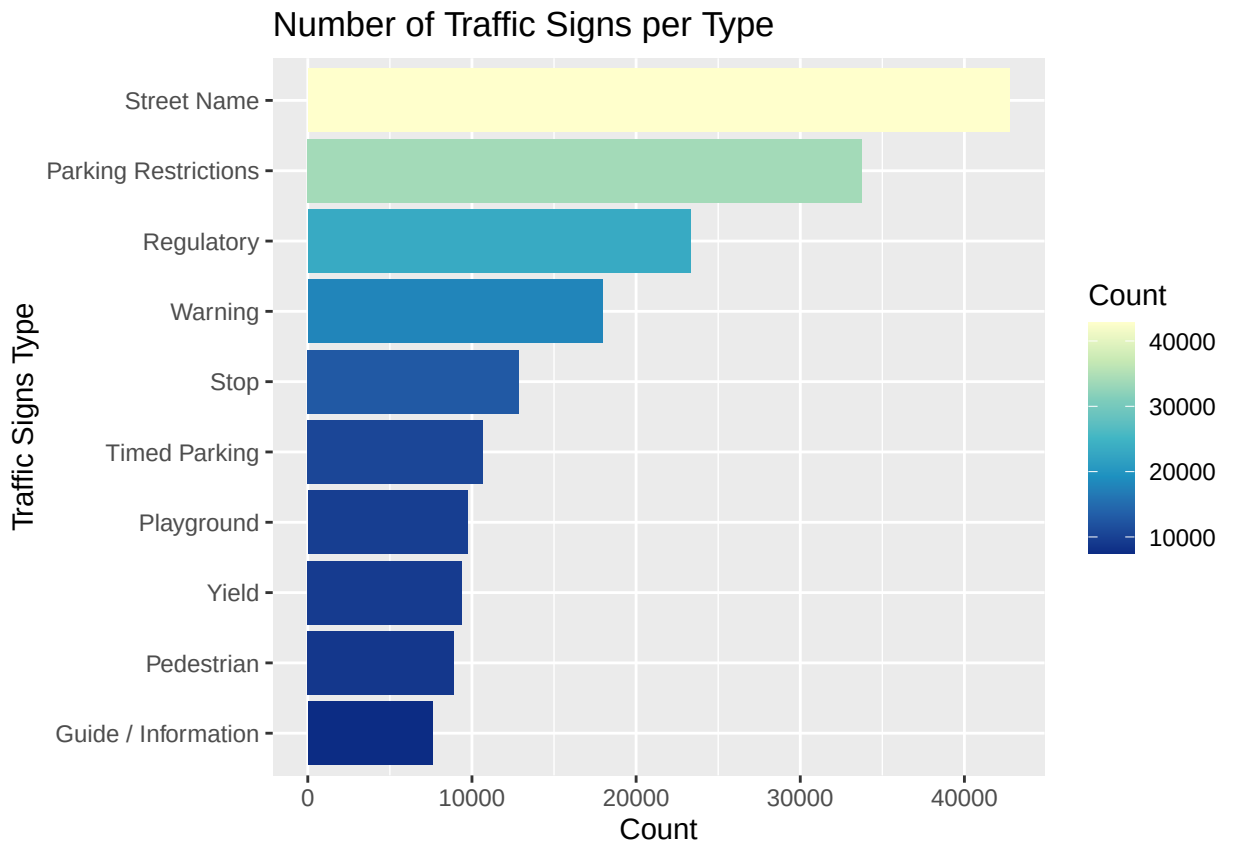
```
  # theme_ipsum() +
  xlab("Month") +
  ylab('Count') +
  # theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  facet_wrap(~Year)
```



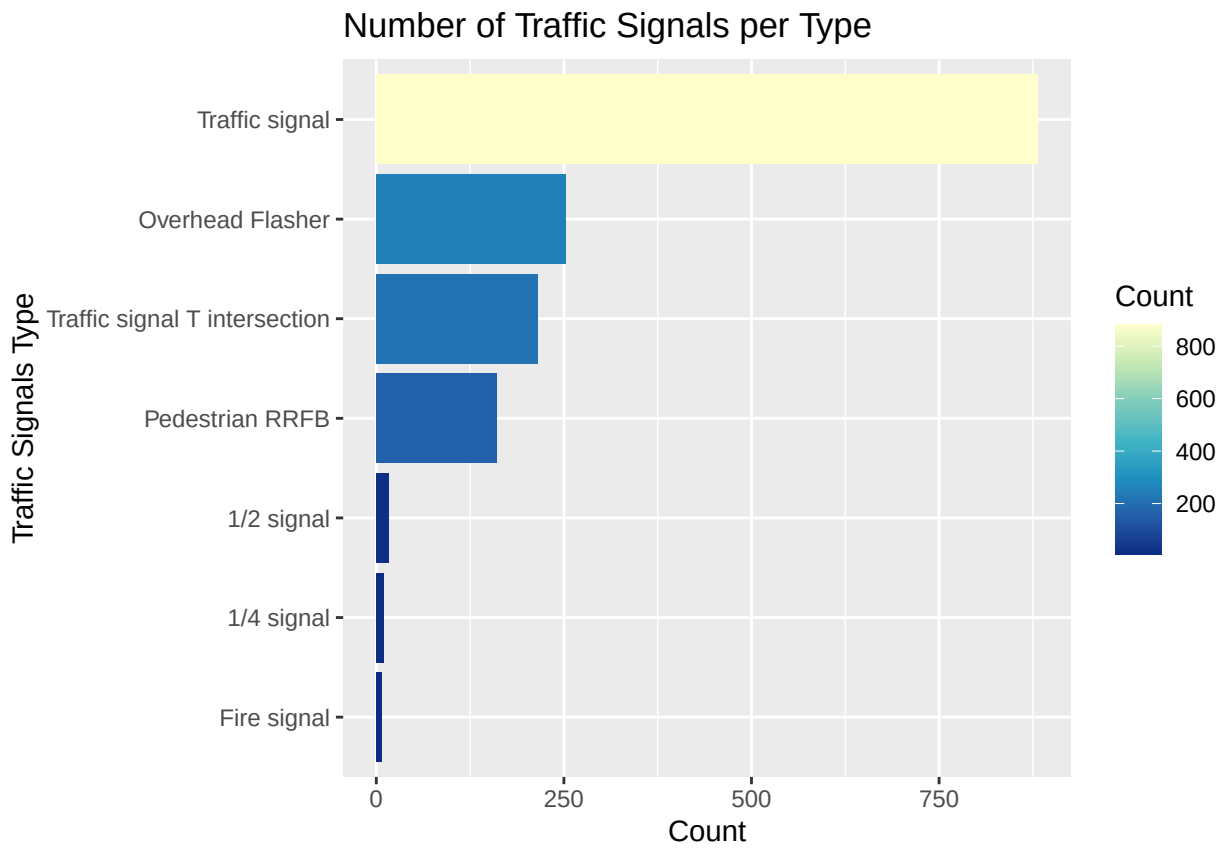
The DESCRIPTION column in the traffic incident contains a brief description of the type of incident. With some simple text mining techniques, the incident description has been grouped into incidents that involve single vehicle, two vehicles and multi vehicles. Longitude and Latitude of traffic signals need to be extracted from the Point object in the dataset. There are 22 unique blade types in the dataset. For the final dataset, only traffic related signs (i.e. Regulatory, Warning, Playground, Speed, Yield, Stop and School) were included. Figure 1 shows the number of traffic signs per type. Since street name and parking regulations comprise most of the traffic signs in Calgary but were not included in the final dataset . There are 6,048 signs installed after 2018. In addition to traffic signs, signals with timer were also extracted from the signals' dataset.

```
traffic_signs %>%
  group_by(BLADE_TYPE) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  head(10) %>%
  ggplot(aes(x = reorder(BLADE_TYPE, Count), Count, fill = Count)) +
  geom_bar(stat="identity") +
  coord_flip() +
  scale_fill_distiller(palette = "YlGnBu") +
  ggtitle("Number of Traffic Signs per Type") +
```

```
  xlab("Traffic Signs Type") +
  ylab('Count')
```



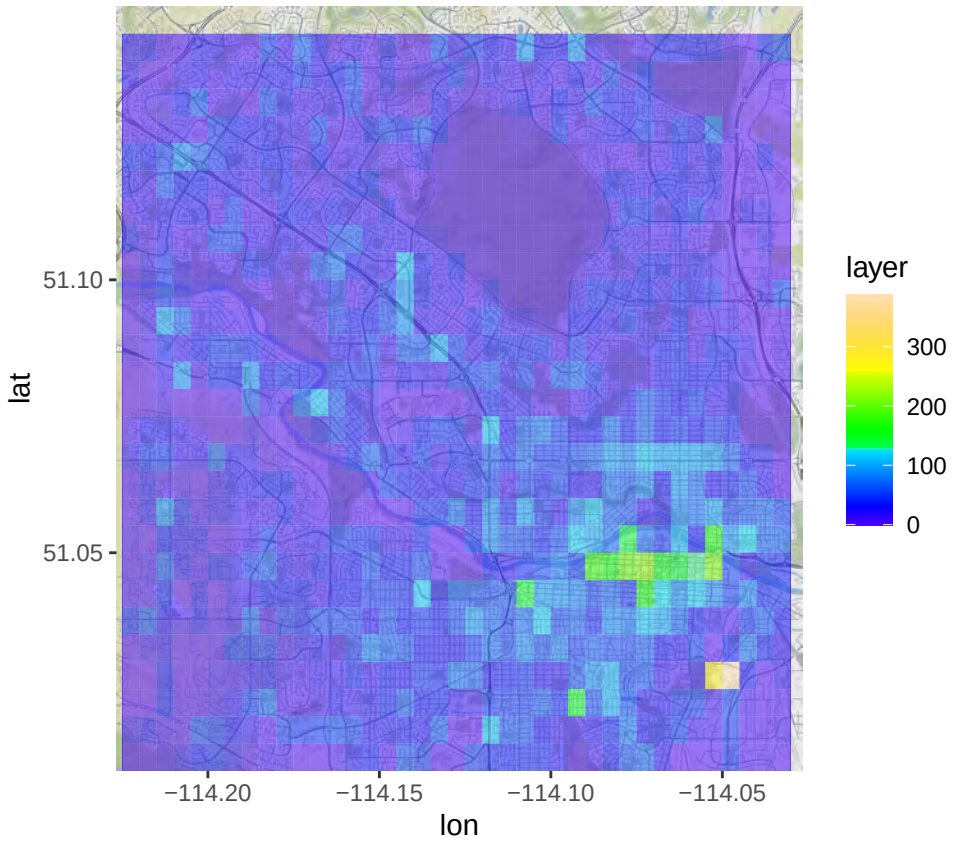

```
traffic_signals %>%
  group_by(INT_TYPE) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
  head(10) %>%
  ggplot(aes(x = reorder(INT_TYPE, Count), Count, fill = Count)) +
  geom_bar(stat="identity") +
  coord_flip() +
  scale_fill_distiller(palette = "YlGnBu") +
  ggtitle("Number of Traffic Signals per Type") +
  xlab("Traffic Signals Type") +
  ylab('Count')
```




```
# Only keeping traffic related signals and signs
signals <- c("Traffic signal T intersection", "Traffic signal", "1/2 signal", "1/4 signal")
signs <- c("Regulatory", "Warning", "Playground", "Speed", "Yield", "Stop", "School")

# I filtered all columns containing single vehicle, two and multi vehicle incidents

single_vehicles_incidents <- traffic_incidents[grep("Single vehicle incident",
                                                    traffic_incidents$DESCRIPTION), ] %>%
  mutate(TYPE = 'Single vehicle incident')
two_vehicles_incidents <- traffic_incidents[grep("Two vehicle incident",
                                                    traffic_incidents$DESCRIPTION), ] %>%
  mutate(TYPE = 'Two vehicle incident')

multi_vehicles_incidents <- traffic_incidents[grep("Multi-vehicle incident",
                                                   traffic_incidents$DESCRIPTION), ] %>%
  mutate(TYPE = 'Multi vehicle incident')

incident_type <- rbind(single_vehicles_incidents, two_vehicles_incidents, multi_vehicles_incidents)
```

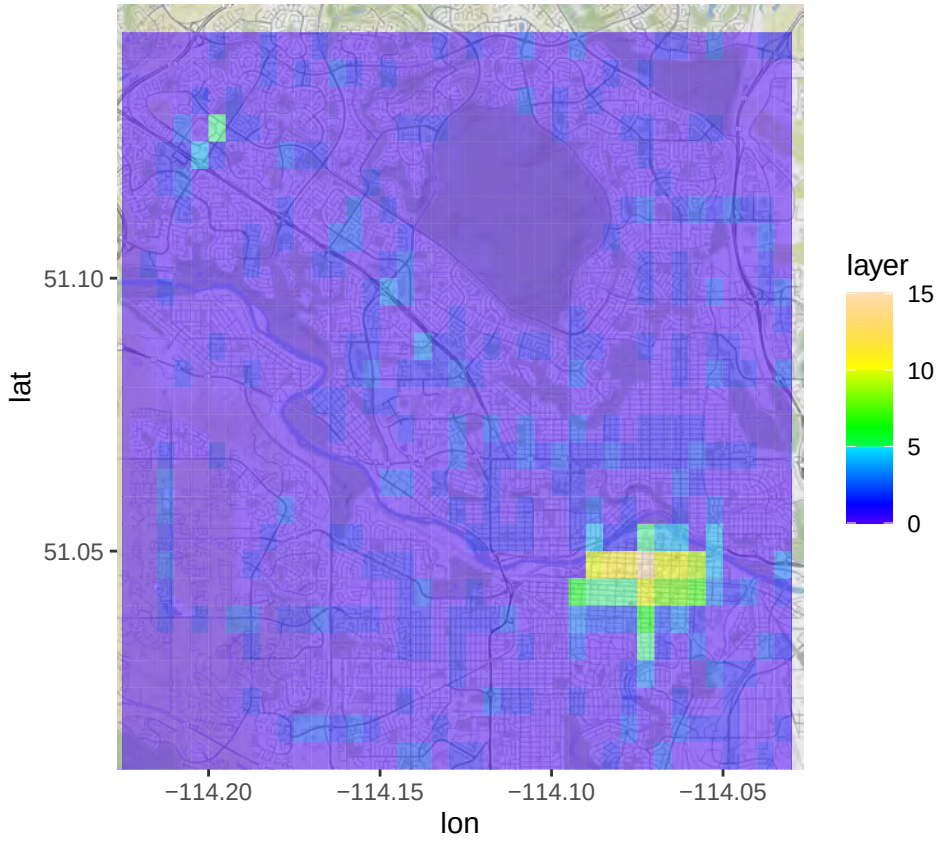
Traffic volume dataset contains multistring lines of roads in Calgary with their volume in 2018. Longitude and latitude of the beginning and end of each line were extracted and assigned with the traffic volume. I believe this was one of the few ways to incorporate the traffic volume data into the final dataset (grid cells). Finally, to create the final dataset, a raster layer from -113.8 to -114.4 longitude and 50.9 to 51.5 latitude with the resolution of 0.005 was created. Each dataset (i.e. signs, signals, police stations, incidents, cameras) was converted to a sf spatial object and the number of points which fall into each grid cell were determined using the code below:

```r
# First, we need to convert the dataframe to the spatial data frame.
traffic_signs_sf <- st_as_sf(x = traffic_signs %>% filter((year(DATE)<2018) & (BLADE_TYPE%in% signs)),
                             coords = c("longitude", "latitude"),
                             crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")

# Now create a raster object for the grids with 0.005 resolution.
r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0

tab <- table(cellFromXY(r, as(traffic_signs_sf, "Spatial")))

r[as.numeric(names(tab))] <- tab

# Now we create a dataframe and

d <- data.frame(coordinates(r), sings_count=r[])
coordinates(d) <- cbind(d$x , d$y)
signs_sf <- st_as_sf(d)

# To visualize the grids, I converted polygons to raster and then plotted the layers

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

rtpFort <- fortify(rtp, data = rtp@data)
rtpFortMer <- merge(rtpFort, rtp@data, by.x = 'id', by.y = 'id')  # join data

# Before we plot the map, we need to register our API key

# ggmap::register_google(key = "Enter Your Google APi key Here")

p <- get_map(location = c(-114.226666,51.009999
                          ,-114.026666,51.149999),
                         maptype = "roadmap",
                         source = "google",
                         zoom = 14)

ggmap(p) + geom_polygon(data = rtpFortMer,
                        aes(x = long, y = lat, group = group, fill = layer),
                        alpha = 0.5,
                        size = 0) +  ## size = 0 to remove the polygon outlines
  scale_fill_gradientn(colours = topo.colors(255))
```




```
# Now we create a dataframe called joined to merge all these dataframes into one

joined <- rtp

# Same steps for traffic signals

traffic_signals_sf <- st_as_sf(x = traffic_signals %>% filter((INT_TYPE %in% signals) &
                                (year(INSTDATE)<2018)),
                            coords = c("longitude", "latitude"),
                            crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")

# Now create a raster object for the grids with 0.01 resolution.
r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(traffic_signals_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

# To visualize the grids, I converted polygons to raster and then plotted the layers

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

rtpFort <- fortify(rtp, data = rtp@data)
rtpFortMer <- merge(rtpFort, rtp@data, by.x = 'id', by.y = 'id')  # join data
```

```
ggmap(p) + geom_polygon(data = rtpFortMer,
                        aes(x = long, y = lat, group = group, fill = layer),
                        alpha = 0.5,
                        size = 0) +  ## size = 0 to remove the polygon outlines
  scale_fill_gradientn(colours = topo.colors(255))
```



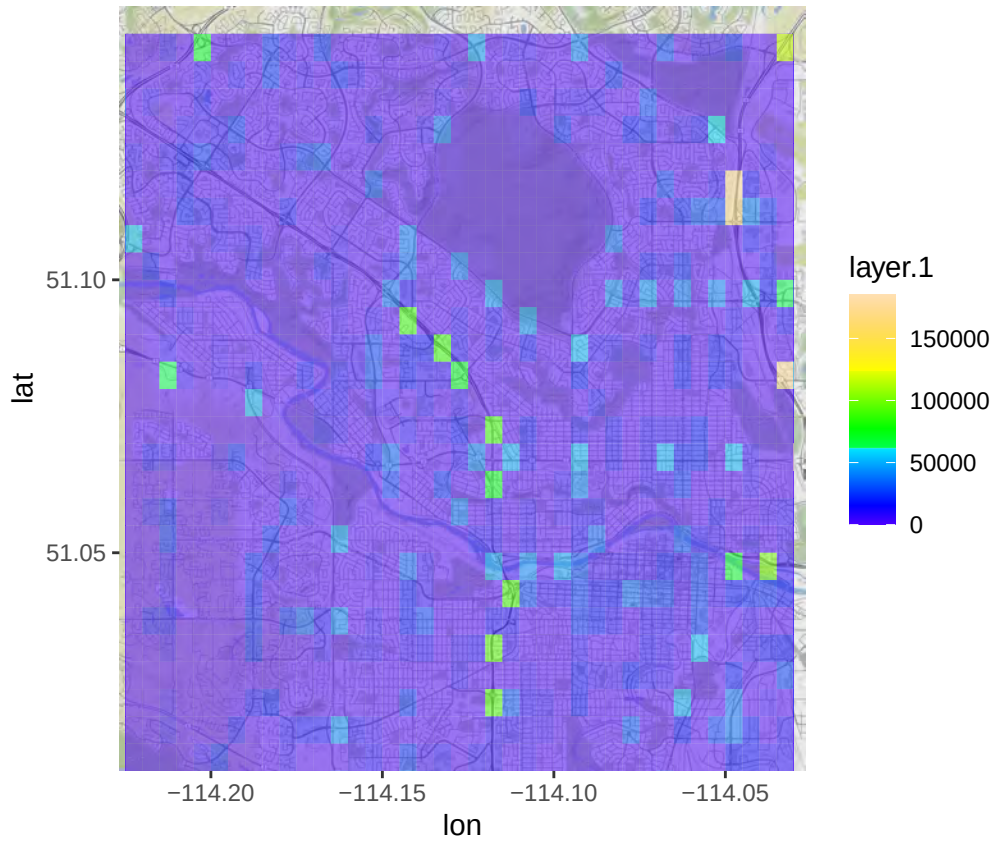

```
joined <- cbind(joined, rtp)

# Repeating all the steps for incidents

traffic_incidents_sf <- st_as_sf(x = traffic_incidents %>% filter((year(DATE) == 2018)),
                                 coords = c("Longitude", "Latitude"),
                                 crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(traffic_incidents_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)    # add id column for join
```

```
joined <- cbind(joined, rtp)

# Let's add single, two and multi vehicle incidents to the joined dataset

single_incidents_sf <- st_as_sf(x = incident_type %>% filter(TYPE == 'Single vehicle incident'),
                                coords = c("Longitude", "Latitude"),
                                crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(single_incidents_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

joined <- cbind(joined, rtp)

# Two vehicle incident

Two_incidents_sf <- st_as_sf(x = incident_type %>% filter(TYPE == 'Two vehicle incident'),
                             coords = c("Longitude", "Latitude"),
                             crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(Two_incidents_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

joined <- cbind(joined, rtp)

# Multi vehicle incidents

Multi_incidents_sf <- st_as_sf(x = incident_type %>% filter(TYPE == 'Multi vehicle incident'),
                               coords = c("Longitude", "Latitude"),
                               crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(Multi_incidents_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab
```

```
rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

joined <- cbind(joined, rtp)

# Repeating all the steps for cameras

traffic_cameras_sf <- st_as_sf(x = traffic_cameras,
                               coords = c("longitude", "latitude"),
                               crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(traffic_cameras_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

joined <- cbind(joined, rtp)

# Repeating all the steps for police stations

police_locations_sf <- st_as_sf(x = police_locations,
                               coords = c("longitude", "latitude"),
                               crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")


r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(police_locations_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

rtp <- rasterToPolygons(r)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join

joined <- cbind(joined, rtp)

# For the traffic volume, we want to know if the coordinates are within the gridcell
# and then want to associate the traffic volume to it.

volume_points_0 <- as.data.frame(traffic_volume) %>% select(long_0, lat_0, VOLUME)
volume_points_1 <- as.data.frame(traffic_volume) %>% select(long_1, lat_1, VOLUME)


coordinates(volume_points_0) = ~long_0+lat_0
coordinates(volume_points_1) = ~long_1+lat_1
```

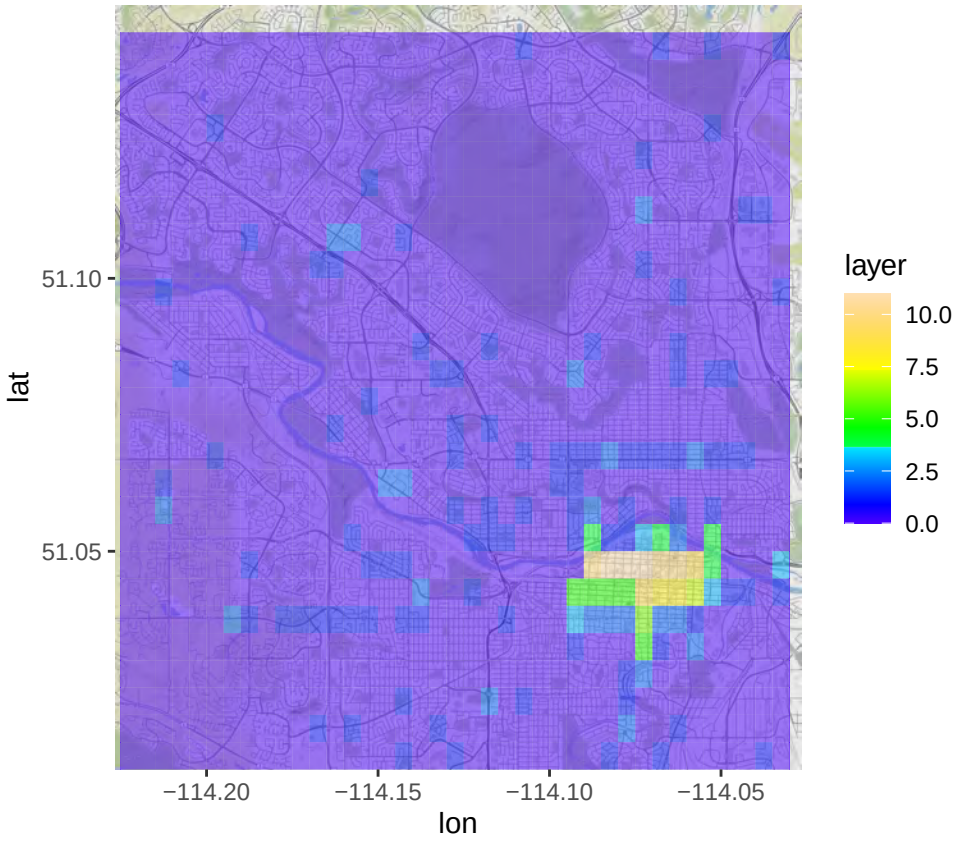
```
traffic_volume_points_0_sf <- st_as_sf(x = volume_points_0,
                                       coords = c("long_0", "lat_0"),
                                       crs = "+datum=WGS84 +no_defs +towgs84=0,0,0")

traffic_volume_points_1_sf <- st_as_sf(x = volume_points_1,
                                        coords = c("long_1", "lat_1"),
                                        crs = "+datum=WGS84 +no_defs +towgs84=0,0,0")

r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r_1 <- raster(xmn=-114.40, ymn=50.90,
             xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
r_1[] <- 0

tab <- table(cellFromXY(r, as(traffic_volume_points_0_sf, "Spatial")))
tab_1 <- table(cellFromXY(r_1, as(traffic_volume_points_1_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab
r_1[as.numeric(names(tab_1))] <- tab_1


st_crs(traffic_volume_points_0_sf) <- st_crs(r)
st_crs(traffic_volume_points_1_sf) <- st_crs(r_1)


volume_points_0 <- as(traffic_volume_points_0_sf, "Spatial")
volume_points_1 <- as(traffic_volume_points_1_sf, "Spatial")

rtp <- rasterToPolygons(r)
rtp_1 <- rasterToPolygons(r_1)
rtp@data$id <- 1:nrow(rtp@data)   # add id column for join
rtp_1@data$id <- 1:nrow(rtp_1@data)   # add id column for join

joined_volume_0 <- st_join(traffic_volume_points_0_sf, st_as_sf(rtp), left = FALSE)
joined_volume_1 <- st_join(traffic_volume_points_1_sf, st_as_sf(rtp_1), left = FALSE)

rtp@data$layer.1 <- 0

rtp@data$layer.1[joined_volume_0$id] <- joined_volume_0$VOLUME
rtp@data$layer.1[joined_volume_1$id] <- joined_volume_1$VOLUME


# Visualizing the traffic volume grid cellls

rtpFort <- fortify(rtp, data = rtp@data)
rtpFortMer <- merge(rtpFort, rtp@data, by.x = 'id', by.y = 'id')  # join data

ggmap(p) + geom_polygon(data = rtpFortMer,
                        aes(x = long, y = lat, group = group, fill = layer.1),
                        alpha = 0.5,
                        size = 0) +  ## size = 0 to remove the polygon outlines
```

```
  scale_fill_gradientn(colours = topo.colors(255))
```




```
# Merge it with the joined dataframe

joined <- cbind(joined, rtp)

# Let's also incorporate the signals with pedestrain timer

traffic_signals_w_timers_sf <- st_as_sf(x = traffic_signals %>% filter((INT_TYPE %in% signals) &
                                                                       (year(INSTDATE)<2018) &
                                                                       (PED_TIMER == 'Yes')),
                                coords = c("longitude", "latitude"),
                                crs = "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0")

# Now create a raster object for the grids with 0.01 resolution.
r <- raster(xmn=-114.40, ymn=50.90,
            xmx=-113.80, ymx=51.50, res=0.005)

r[] <- 0
tab <- table(cellFromXY(r, as(traffic_signals_w_timers_sf, "Spatial")))
r[as.numeric(names(tab))] <- tab

# To visualize the grids, I converted polygons to raster and then plotted the layers

rtp <- rasterToPolygons(r)
```

```
rtp@data$id <- 1:nrow(rtp@data)    # add id column for join

rtpFort <- fortify(rtp, data = rtp@data)
rtpFortMer <- merge(rtpFort, rtp@data, by.x = 'id', by.y = 'id')  # join data

ggmap(p) + geom_polygon(data = rtpFortMer,
                        aes(x = long, y = lat, group = group, fill = layer),
                        alpha = 0.5,
                        size = 0) +  ## size = 0 to remove the polygon outlines
  scale_fill_gradientn(colours = topo.colors(255))
```




```
joined <- cbind(joined, rtp)

# Let's also incorporate the type of incidents

# All these sf dataframes now have 14,400 rows (or gridcells). We can merge all the counts
# by the gridcells and create our final dataset.

joined_traffic <- as.data.frame(joined) %>% select(layer,layer.1,layer.2,layer.3,layer.4,
                                                   layer.5, layer.6, layer.7, layer.8,
                                                   layer.1.1, layer.9) %>%
    rename(signs_count = layer, signals_count = layer.1, incidents_count = layer.2,
           single_incidents = layer.3, two_incidents = layer.4, multi_incidents = layer.5,
           cameras_count = layer.6, pstations_count = layer.7, volume_count = layer.8,
           volume = layer.1.1, timer_count = layer.9)
```

```
# There are some discrepencies between the counts in each grid cell and the original data
# For example, the total rows of the signals dataset is 1,058 but the sum of counts in the
# final dataset is 1,030. Or the total rows of the signs dataset is 69,780 but the sum of
# signs count for the final dataset is only 65,734!
```

The traffic volume points were merged with the raster layer with sf_join function. The final dataset joined_traffic contains 14,400 rows and 11 columns. The final dataset does not have any missing data.

Only grids with traffic volume of greater than 500 were considered for the modelling. There are 998 grids with traffic volume of greater than 500.

Methodology

The dataset (joined_traffic_vol) was partitioned into a train (90%) and test set (10%). After splitting the dataset, the train and validation sets have 897 and 101 rows respectively (both have six variables or predictors). LOOCV and K-fold cross validation were used for modelling the training set. Linear regression, Random Forest, Conditional Random Forest and Conditional Inference Trees were used to model and predict the incidents count in each grid. RSME was used as our loss function to evaluate the predictions based on the validation set.

Results and Discussion

Figure 4 shows the number of pedestrian incidents from 2014 to 2018. Majority of the collisions was caused as a result of car or pickup trucks. Because the pedestrian dataset did not include the location of the collision, they were not considered in the final dataset.

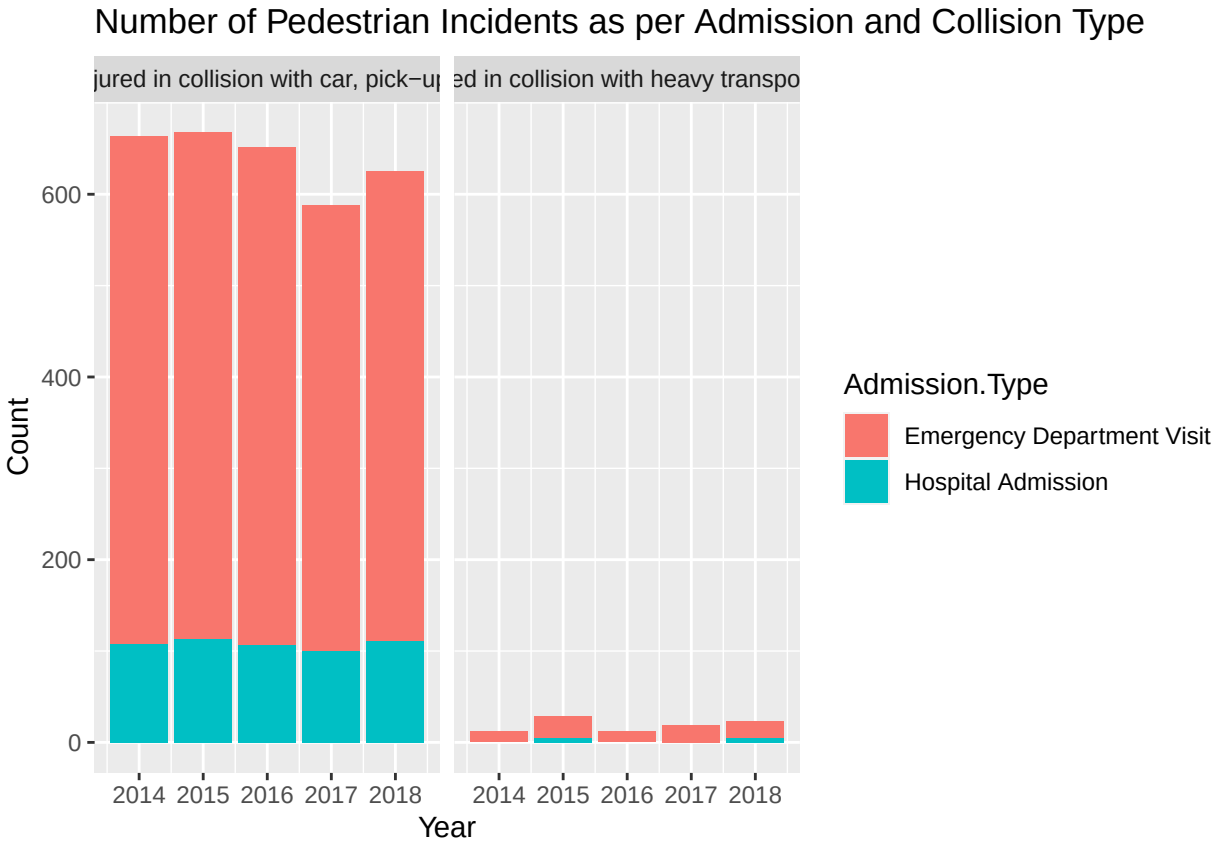
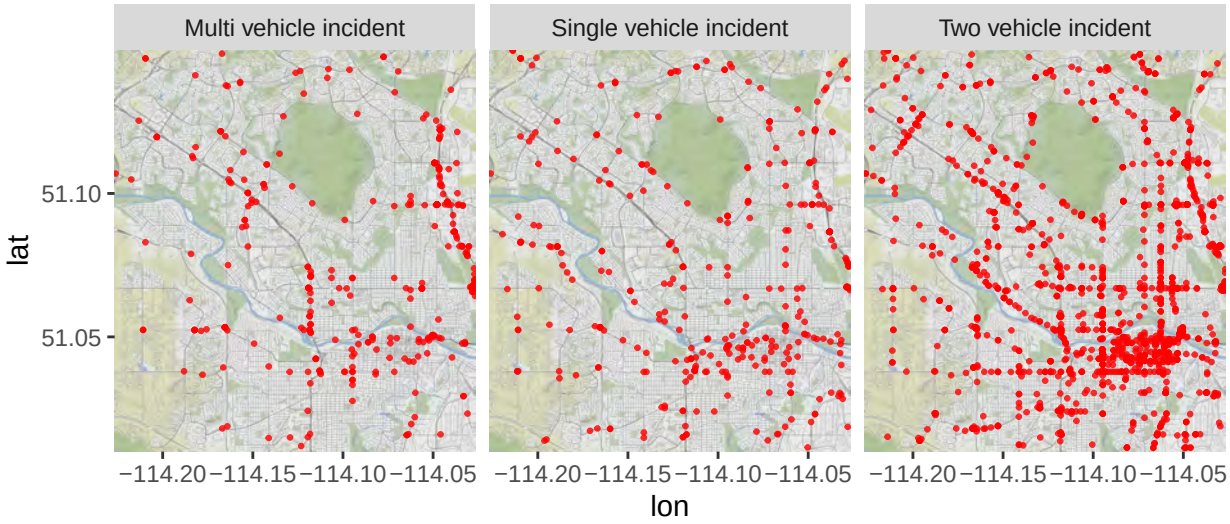
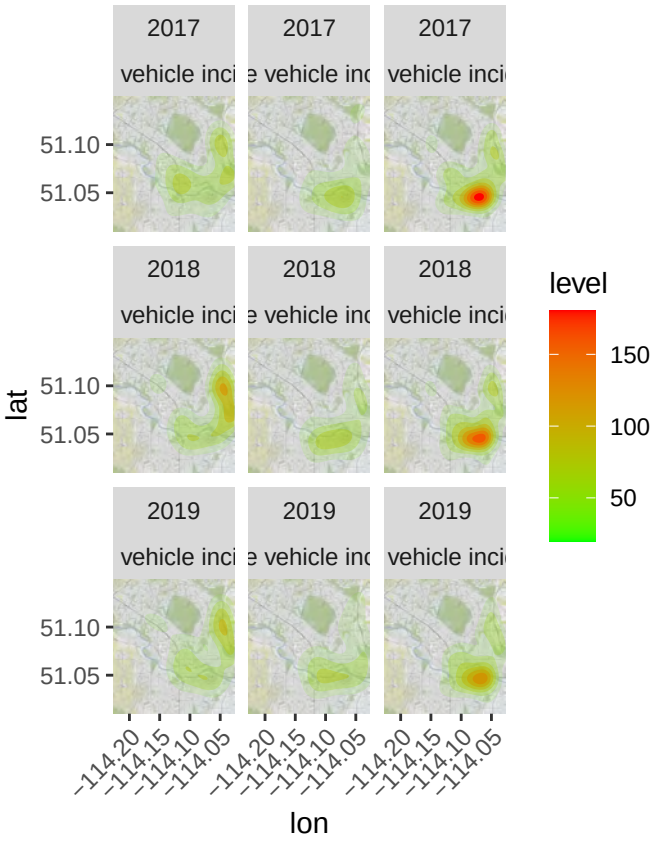
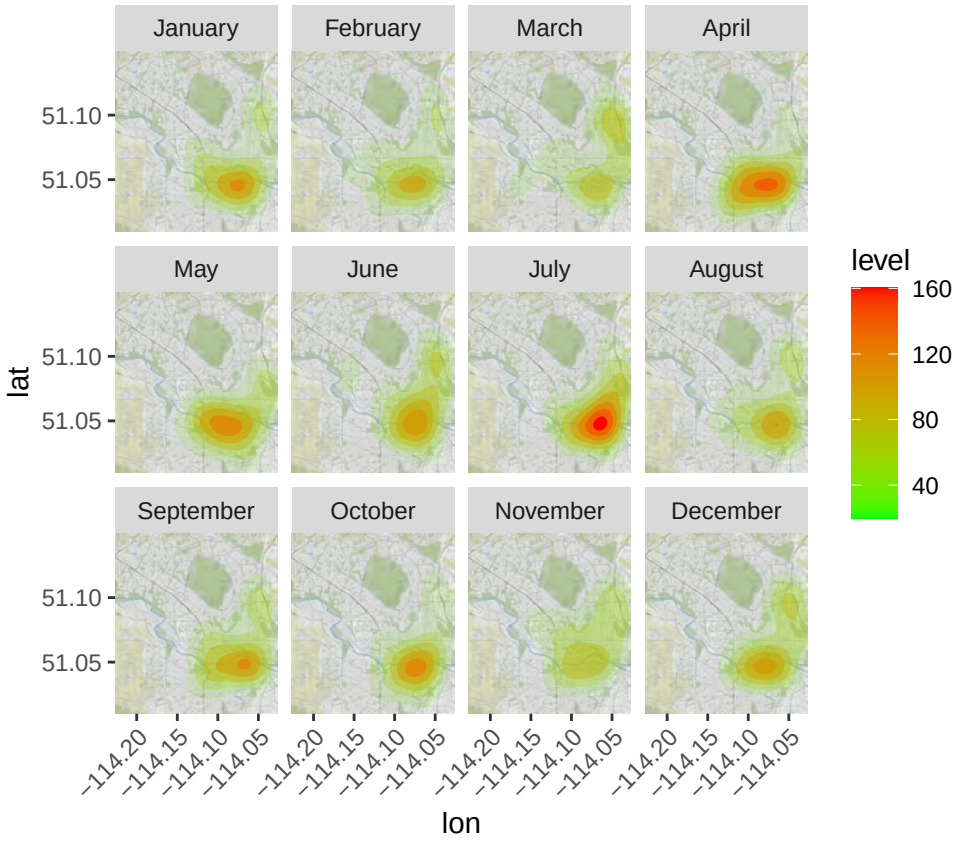
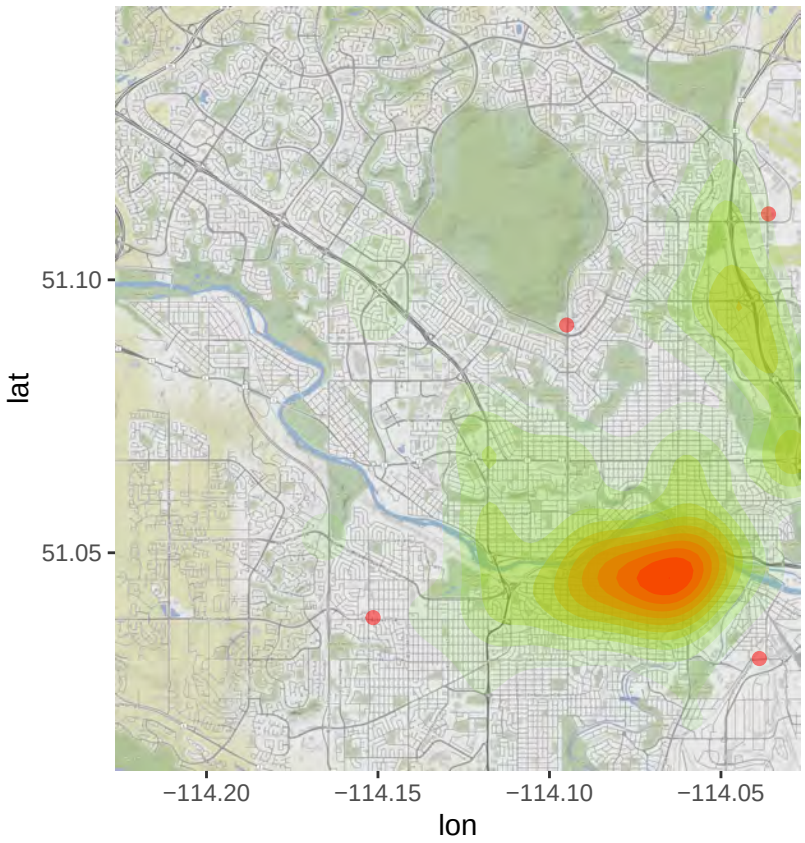


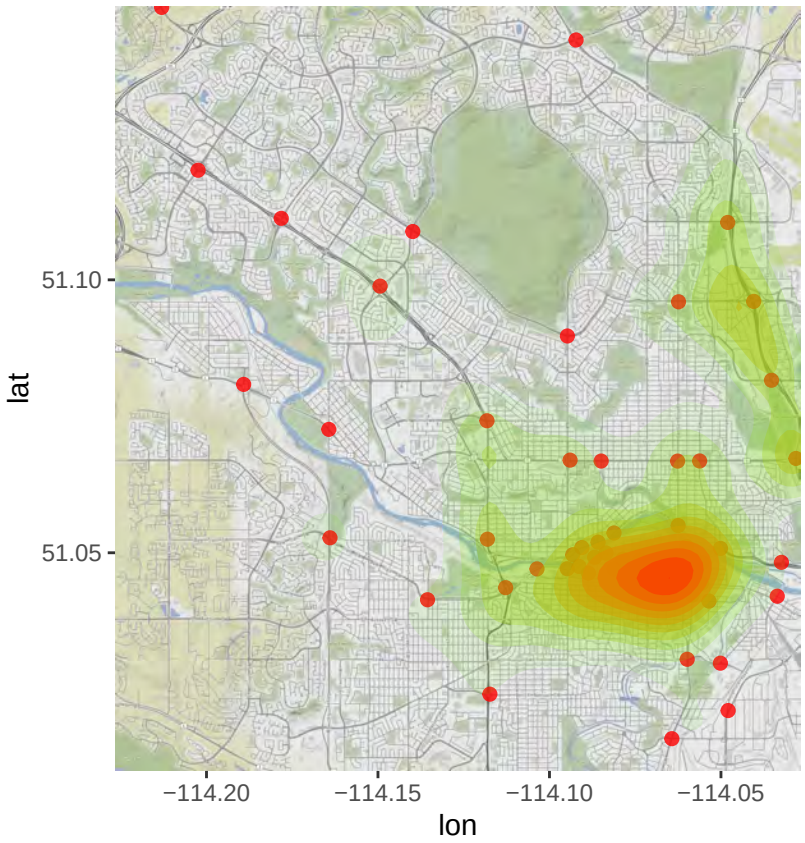
Figure 5 shows the distribution of traffic incidents (per type) in 2018. There are more single and two vehicles incidents in the downtown area but multi vehicles incidents are more likely to happen in Deer foot trail and roads with higher traffic volumes.


Multi vehicle incident
Single vehicle incident
Two vehicle incident
lat
51.10
51.05
−114.20
−114.15
−114.10
−114.05
lon

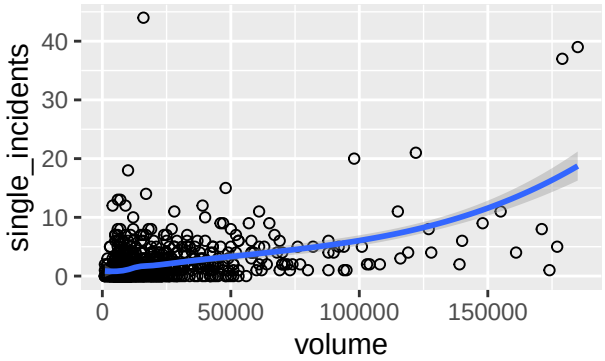

2017
2018
2019
vehicle inci
51.10
51.05
lat
level
150
100
50
−114.20
−114.15
−114.10
−114.05
lon


January
February
March
April
May
June
July
August
September
October
November
December
lat
51.10
51.05
lon
−114.20
−114.15
−114.10
−114.05
level
160
120
80
40

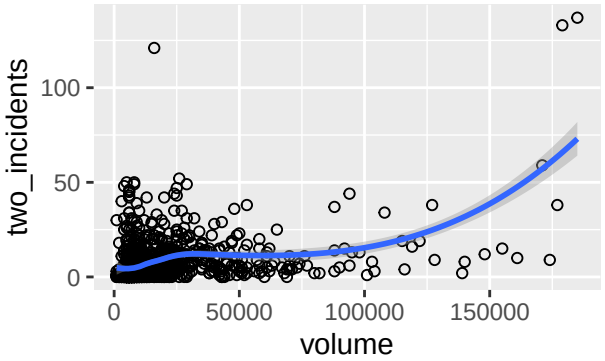

51.10
51.05
lat
−114.20
−114.15
−114.10
−114.05
lon


51.10
51.05
lat
−114.20
−114.15
−114.10
−114.05
lon

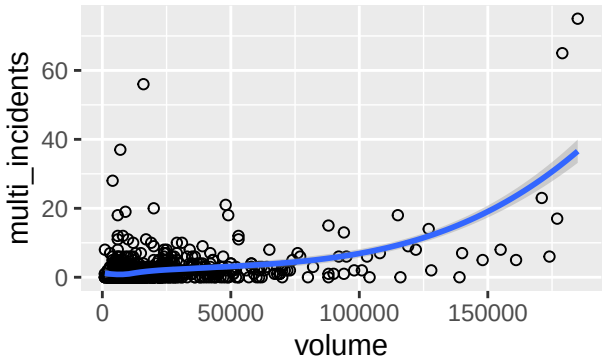
atter plot of single vehicle incidents and tr



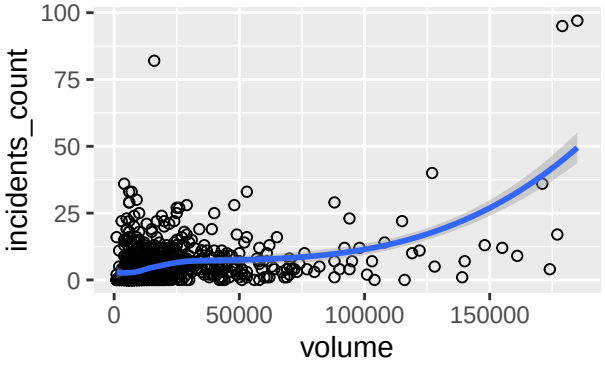
Scatter plot of two vehicle incidents and tra



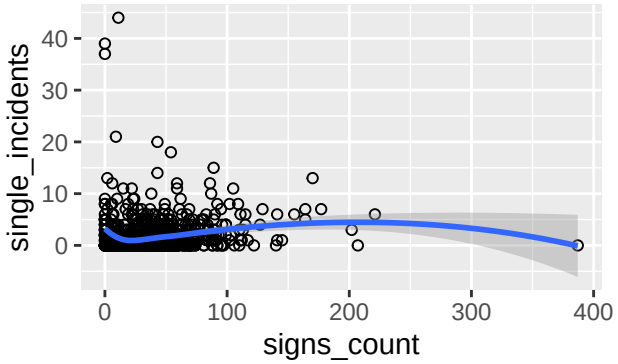
atter plot of multi vehicle incidents and tr



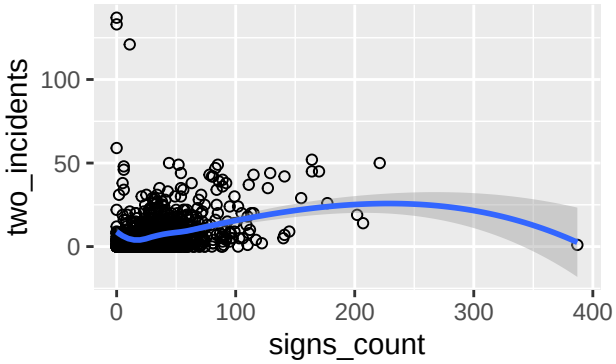
Scatter plot of total vehicle incidents and tra



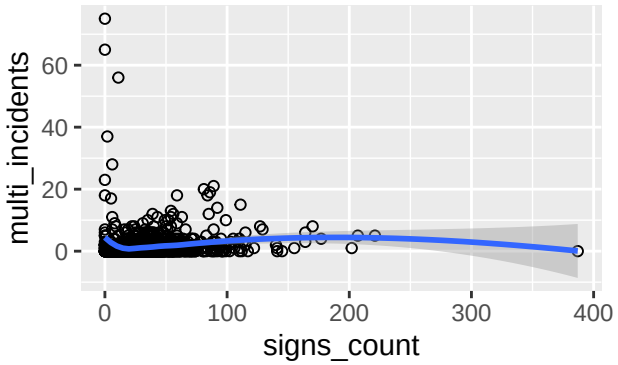
atter plot of single vehicle incidents and S



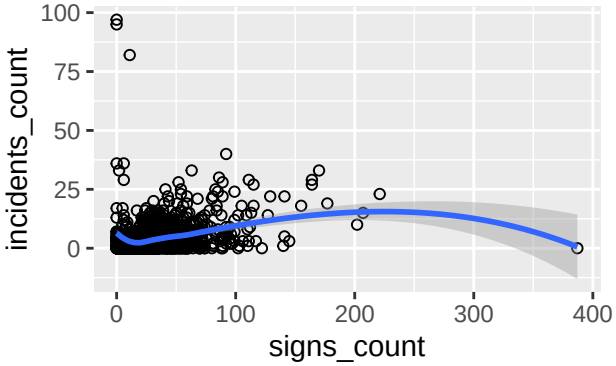
Scatter plot of two vehicle incidents and si



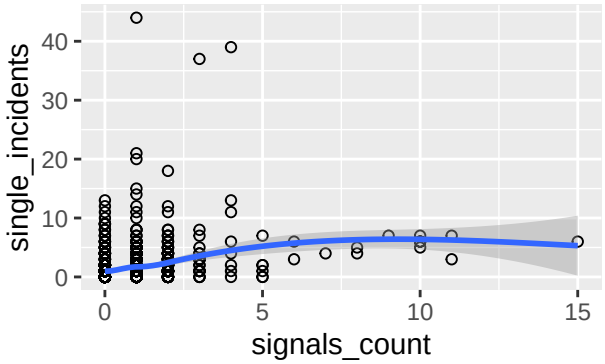
catter plot of multi vehicle incidents and Si



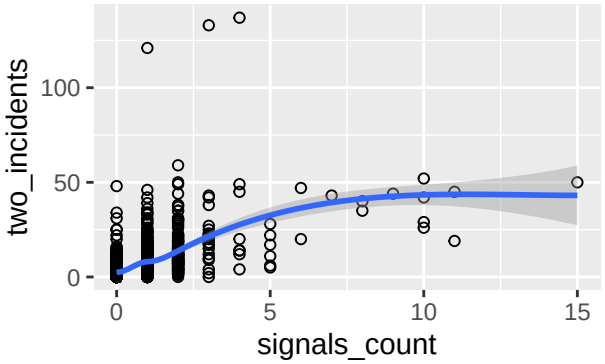
Scatter plot of total vehicle incidents and s



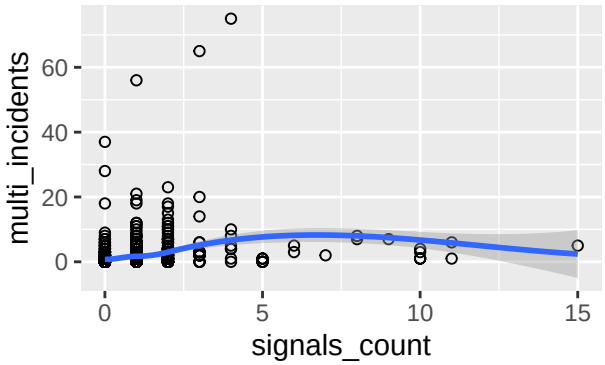
atter plot of single vehicle incidents and Sc



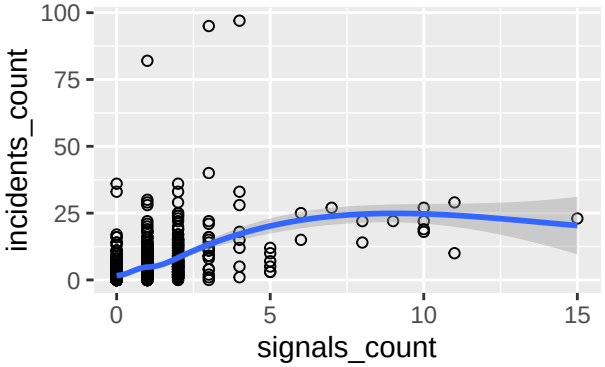
Scatter plot of two vehicle incidents and sig



atter plot of multi vehicle incidents and Sc



Scatter plot of total vehicle incidents and sig



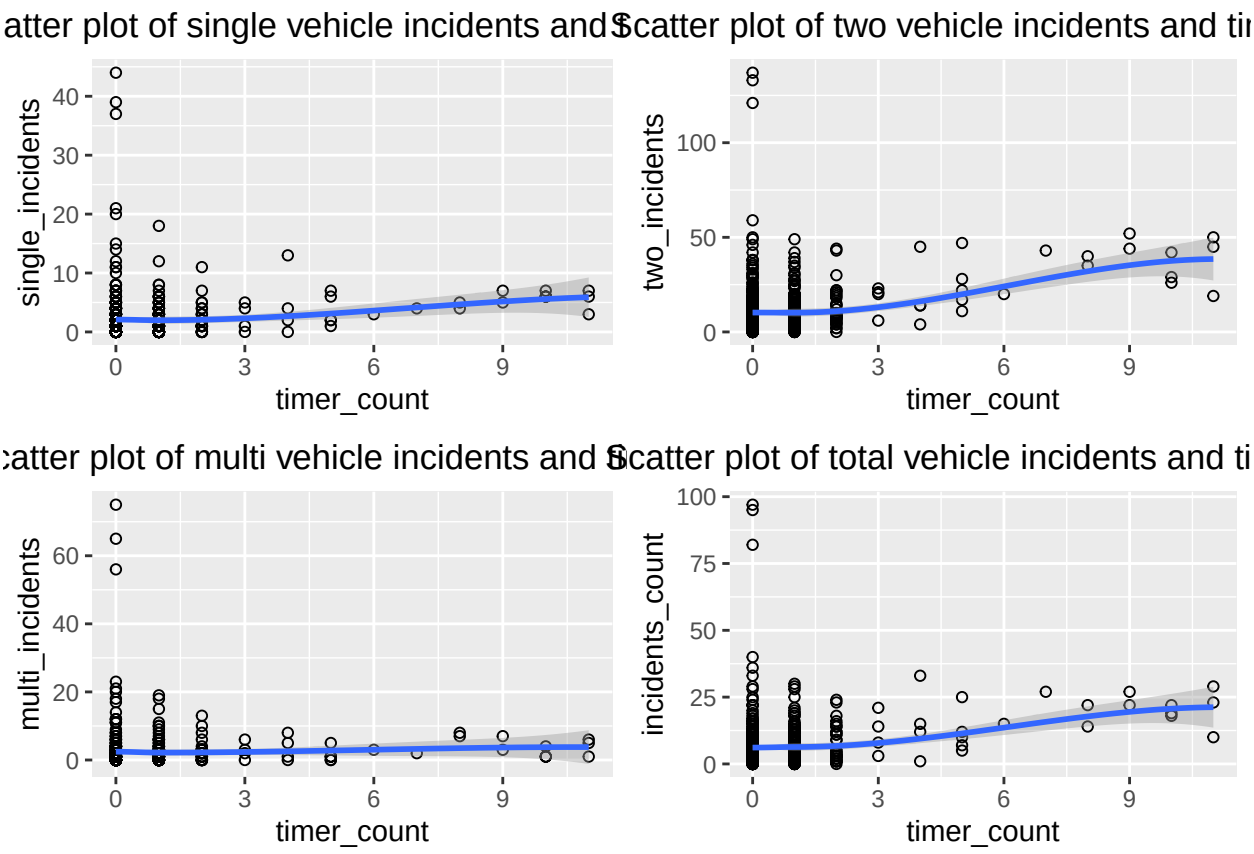
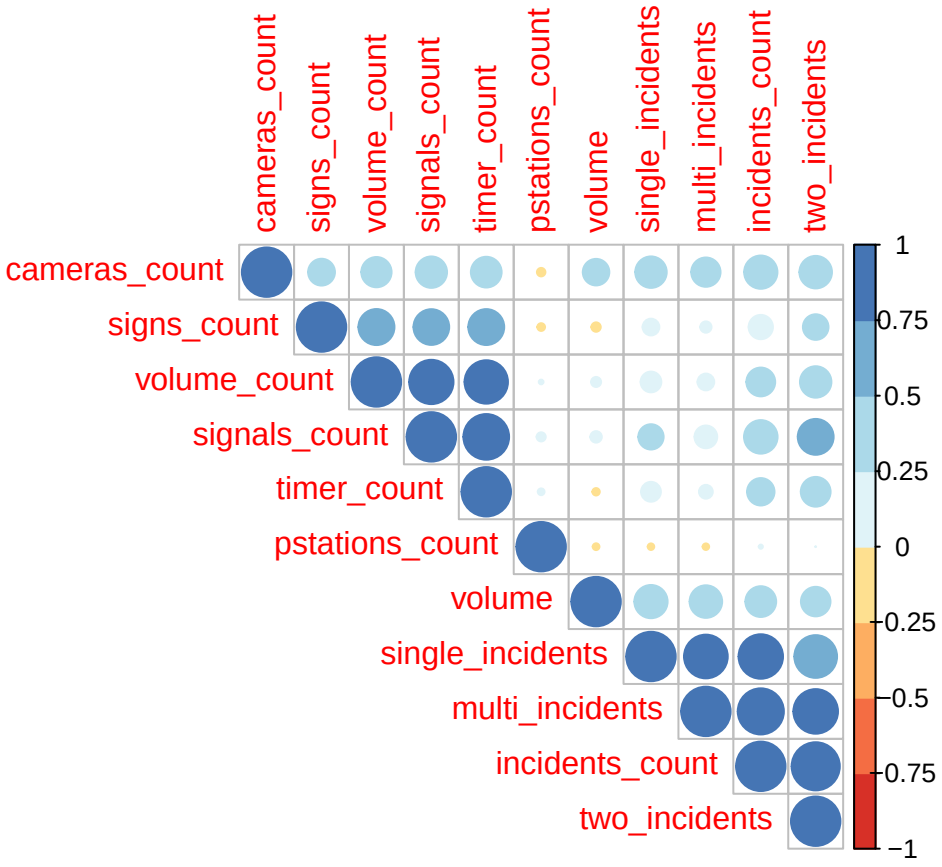


Figure 15 shows the correlations between different variables in the traffic dataset. Because single, two and multi incidents are not independent of the total vehicle incidents, they were removed from our final dataset. Signals count have the strongest correlation with the incidents count with the correlation factor of 0.46 followed by camera counts (0.45). It should be noted that association does not imply causation. For example, there may be more signals or traffic cameras as a result of more traffic incidents, and not the other way around.

```
# Let's see what the correlations look like between the variables

corrplot(cor(joined_traffic_vol), type="upper", order="hclust",
         col=brewer.pal(n=8, name="RdYlBu"))
```



Model Performance

```
#RMSE function

RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Normalization funtion
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

# Normalization of the final dataset excluding the single, two and multi vehicle incidents

traffic_model <- joined_traffic_vol[,traffic_var]

columns <- 1:6
for (col in columns){
  traffic_model[,col] <- normalize(traffic_model[,col])
}

# Partitioning the data
```

```
# Validation set will be 10% of MovieLens data
set.seed(1)

test_index <- createDataPartition(y = traffic_model$incidents_count, times = 1, p = 0.1, list = FALSE)
train_set <- traffic_model[-test_index,]
test_set <- traffic_model[test_index,]
```

Linear Regression

RMSE for the linear regression model on the validation set (without cross validation) was 5.101977.

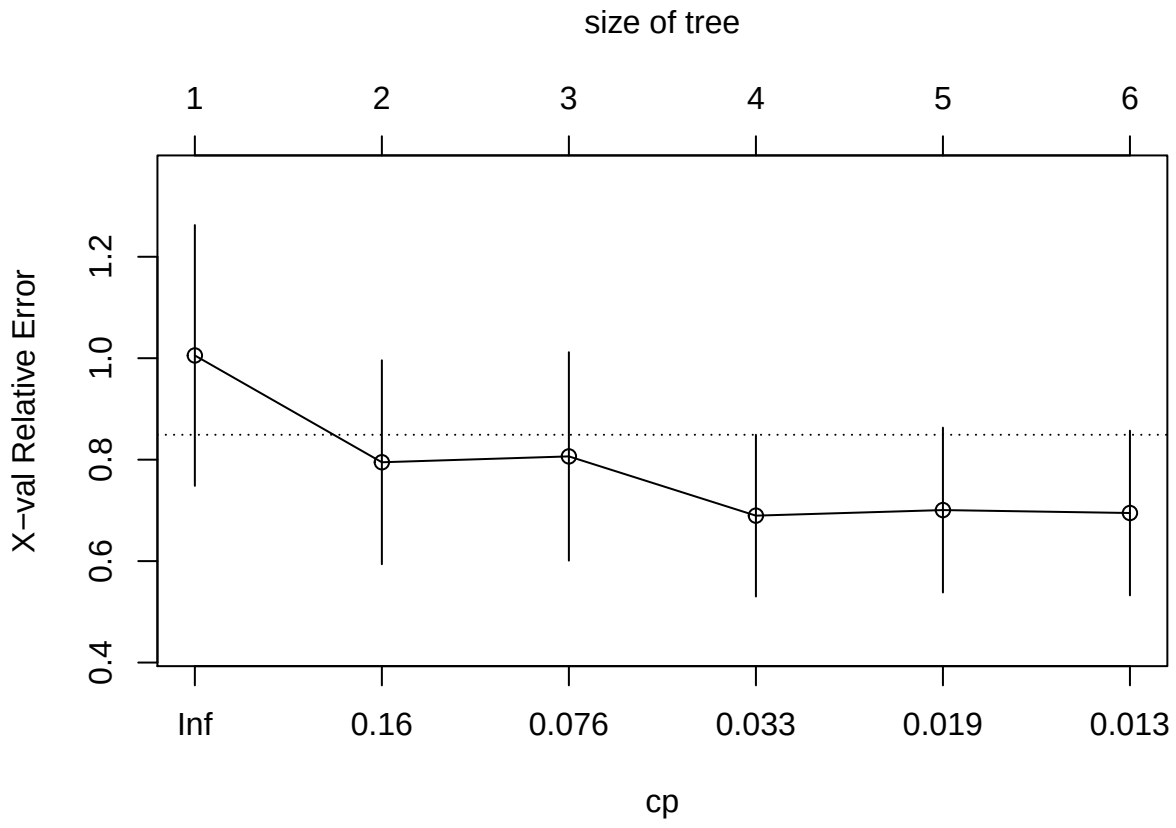
```
mode_lm <- lm(incidents_count~., data = train_set)
#use predict() to make prediction on a new set
pred_lm <- predict(mode_lm, test_set, type = "response")
residuals <- test_set$incidents_count - pred_lm
pred_lm_df <- data.frame("Predicted" = pred_lm, "Actual" = test_set$incidents_count, "Residual" = residu

RMSE(pred_lm, test_set$incidents_count)
```

```
## [1] 5.778998
```

ctree

RMSE for the ctree model with cp of 0.01 on the validation set was 5.430917. Figure 16 shows the size of tree and cp versus the relative error of the model.

```
model_ctree <- rpart(incidents_count~.,
                     data = train_set,control = rpart.control(cp = 0.01))

plotcp(model_ctree)
```




```
printcp(model_ctree)
```

```
##
## Regression tree:
## rpart(formula = incidents_count ~ ., data = train_set, control = rpart.control(cp = 0.01))
##
## Variables actually used in tree construction:
## [1] cameras_count signals_count signs_count   volume
##
## Root node error: 52880/897 = 58.952
##
## n= 897
##
##         CP nsplit rel error  xerror    xstd
## 1 0.213531      0   1.00000 1.00545 0.25704
## 2 0.117067      1   0.78647 0.79488 0.20101
## 3 0.049365      2   0.66940 0.80642 0.20544
## 4 0.021660      3   0.62004 0.68964 0.15944
## 5 0.016059      4   0.59838 0.70062 0.16244
## 6 0.010000      5   0.58232 0.69482 0.16234
```

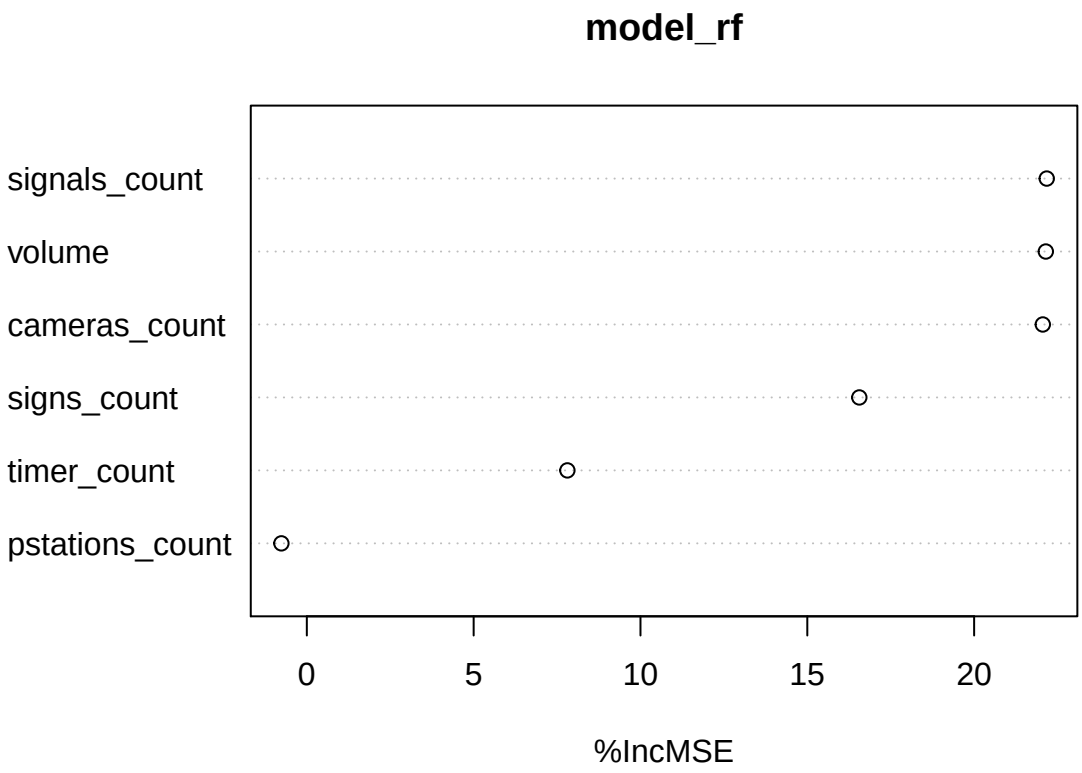
```
pred_model_ctree <- predict(model_ctree, test_set[,1:6])
RMSE(pred_model_ctree, test_set$incidents_count)
```

```
## [1] 6.815359
```

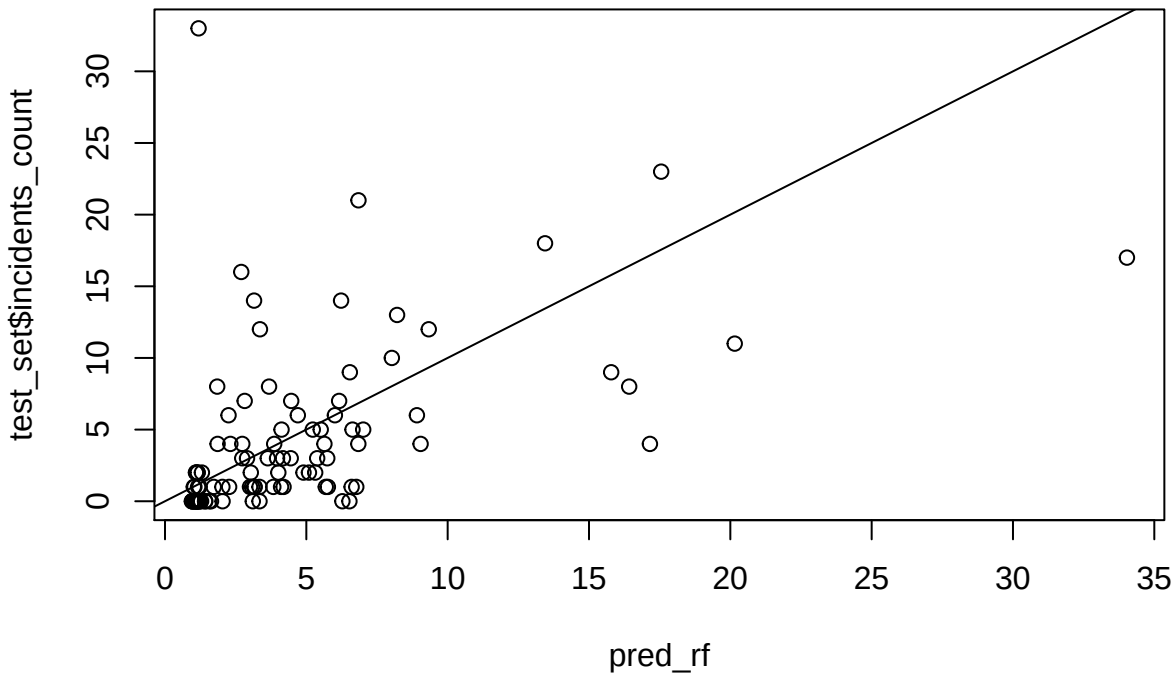
Random Forest

Table 2 summarizes the RMSE values for different Random Forest parameters. 15-fold cross validation and ntree of 1000 showed the smallest RMSE among all models. As a result, the model with 1000 ntree and 15-fold cross validation were ran on the validation dataset.

```
model_rf <- randomForest(incidents_count ~., data = train_set,
                         importance =TRUE,ntree=1000,nodesize=7, na.action=na.roughfix)

options(repr.plot.width=9, repr.plot.height=6)
varImpPlot(model_rf, type=1)
```




```
pred_rf <- predict(model_rf, newdata=test_set )

plot(pred_rf, test_set$incidents_count, main = "")
abline(0,1)
```




```
RMSE(pred_rf, test_set$incidents_count)
```

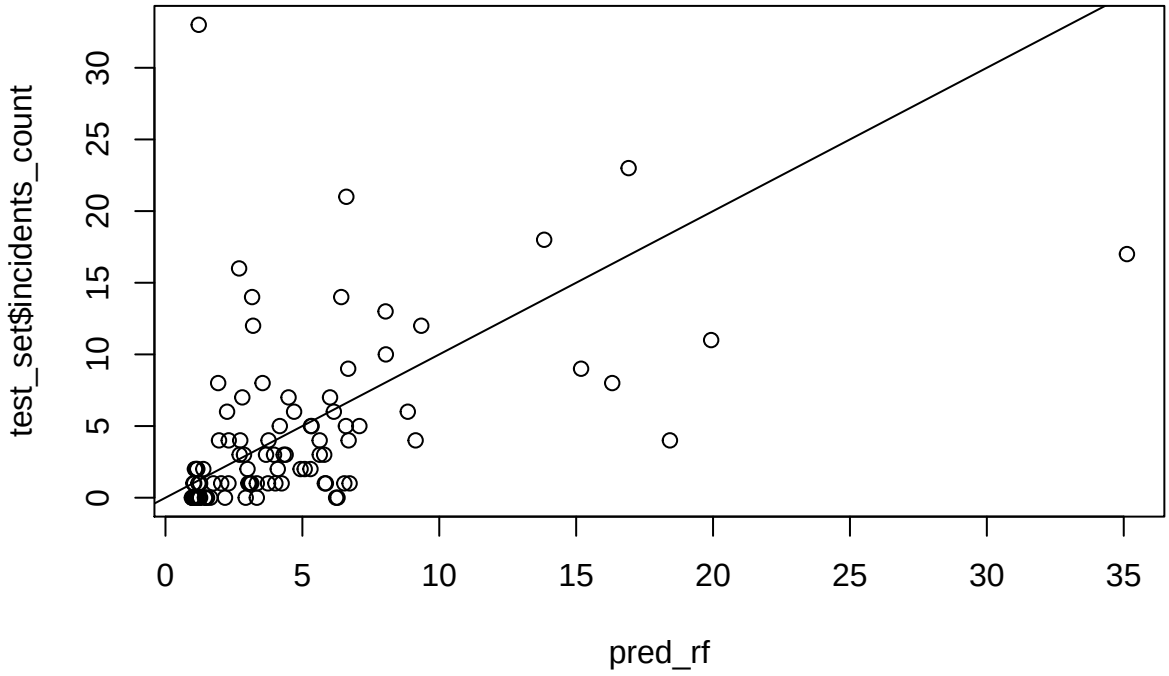
```
## [1] 5.368562
```

```
# Let's tune some of the Random Forest parameters

k <- c(5,7,10,15)
modellist <- list()

set.seed(1)

for (n in k) {
  control <- trainControl(method="repeatedcv", number= n, repeats=3, search="grid")
  tunegrid <- expand.grid(.mtry=c(sqrt(ncol(train_set)-1)))
  for (ntree in c(500, 1000, 1500, 2000)) {
    set.seed(1)
    fit <- train(incidents_count~., data=train_set, method="rf", tuneGrid=tunegrid,
                 trControl=control, ntree=ntree)
    key <- paste0('ntree of ', ntree, sep = ', k-fold of ', n)
    modellist[[key]] <- fit
  }
  }

# 15 folds and 1000 ntree showed the smallest RMSE. Let's find the RMSE of the validation
# set with these parameters.
```

```
control <- trainControl(method="repeatedcv", number= 15, repeats=3, search="grid")
set.seed(1)
fit <- train(incidents_count~., data=train_set, method="rf", tuneGrid=tunegrid,
             trControl=control, ntree=2000)
pred_rf <- predict(fit, newdata=test_set )
RMSE(pred_rf, test_set$incidents_count)
```

```
## [1] 5.430444
```

```
plot(pred_rf, test_set$incidents_count, main = "")
abline(0,1)
```



SVM Parameter Turning

Neither polynomial nor radial method improved the RMSE of the SVM method. The lowest RMSE was observed with linear method (4.865885).

```
model_svm1 <- svm(incidents_count~.,data = train_set,kernel="linear")
pred_svm1 <- predict(model_svm1, test_set)
RMSE(pred_svm1, test_set$incidents_count)
```

```
## [1] 4.92255
```

```
model_svm2 <- svm(incidents_count~.,data = train_set,kernel="poly",
                  degree=2,gamma=500,cost=0.5)
pred_svm2 <- predict(model_svm2, test_set)
RMSE(pred_svm2, test_set$incidents_count)
```

```
## [1] 179.5038
```

```
model_svm3 <- svm(incidents_count~.,data = train_set,kernel="radial",
                  gamma=1,cost=0.5)
pred_svm3 <- predict(model_svm3, test_set)
RMSE(pred_svm3, test_set$incidents_count)
```

```
## [1] 5.178252
```

Conclusion

In this report, the effect of different variables such as traffic infrastructure and volume were studied on the total traffic incidents. Out of these variables, traffic signals count, camera counts, and volume had the strongest correlation with the traffic incidents. Four machine learning algorithms (linear, ctree, Random Forest and SVM) were used to predict the vehicle incidents. Among these four models, SVM showed the smallest RMSE. Furthermore, k-fold cross validation did not improve the RMSE of the validation set.