Parking infractions, City of Toronto take home assignment

Nima Jamshidi

Contents

In this notebook, I am going to study the most frequent parking infractions in Toronto with analyzing their locations and underlying factors. For this study, I'm using R programming language with the use of RStudio IDE.

First step is to install and load the required packages.

```
suppressPackageStartupMessages(library(opendatatoronto))
suppressPackageStartupMessages(library(ckanr))
suppressPackageStartupMessages(library(readr))
suppressPackageStartupMessages(library(vroom))
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(sf))
suppressPackageStartupMessages(library(leaflet))
suppressPackageStartupMessages(library(rjson))
suppressPackageStartupMessages(library(scales))
suppressPackageStartupMessages(library(paletteer))
```

Parking Tickets

Here I wrote a function for getting data for any year available on the website. There are a few note: - `opendatatoronto` package has problem with pulling `.csv` files. This is done manually here. - For some of the years, the `.csv` file is corrupt. The Unicode requires adjustment. This function does that. - For some of the years, the data is split into different files. This function handles that as well and combines the data into one object.

```
PaTi_metadata <- opendatatoronto::show_package("https://open.toronto.ca/dataset/parking-tickets/")
PaTi_list <- opendatatoronto::list_package_resources("https://open.toronto.ca/dataset/parking-tickets/")
PaTi_data_tot <- tibble()
Col_type <- cols(
        tag_number_masked = col_character(),
        date_of_infraction = col_double(),
        infraction_code = col_double(),
        infraction_description = col_character(),
        set_fine_amount = col_double(),
        time_of_infraction = col_character(),
        location1 = col_character(),
        location2 = col_character(),
```

```
        location3 = col_character(),
        location4 = col_character(),
        province = col_character()
      )

# https://github.com/tidyverse/vroom/issues/138
Corrupted_csv <- function(dir,PaTi_data_id,save_path,Col_type) {
  resource_dir <- fs::dir_create(paste0(dir, "/", PaTi_data_id))
  csv_files <- unzip(save_path[["path"]], exdir = resource_dir)
  con <- file(csv_files, "rb")
  x <- readBin(con, "raw", n = 500000000)
  utf_text <- iconv(
        list(x),
        from = "UTF-16LE",
        to = "UTF-8",
        toRaw = F
      )
  res <-
    vroom::vroom(
      utf_text,
      delim = ",",
      col_types = Col_type,
      col_names = names(Col_type$cols),
      skip = 1
    ) %>%
    mutate(time_of_infraction = replace_na(time_of_infraction, "0000"))
  return(res)
}

read_all_zip <- function(file, ...) {
  filenames <- unzip(file, list = TRUE)$Name
  vroom(purrr::map(filenames, ~ unz(file, .x)), ...)
}

PaTi_retrieve <- function(i){
  PaTi_data_id <- PaTi_list[[i,"id"]]
  resource <-
    resource_show(PaTi_data_id, url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/", as = "list")
  dir <- tempdir()
  resource_dir <- fs::dir_create(paste0(dir, "/", PaTi_data_id))
  save_path <-
    ckan_fetch(
      resource[["url"]],
      store = "disk",
      path = paste0(dir, "/", PaTi_data_id, "/", "res.zip")
    )
  t <- try({
    # https://github.com/tidyverse/vroom/issues/125
    filenames <- unzip(save_path$path, list = TRUE)$Name
    res <- bind_rows(purrr::map(filenames, ~ vroom(unz(save_path$path, .x),
                                                   delim = ",",
                                                   col_names = names(Col_type$cols),
                                                   col_types = Col_type,
```

```
                                                    skip = 1)))
  })
  if (inherits(t, "try-error")|ncol(res)!=11)  {
    print("corrupted CSV")
    res <- Corrupted_csv(dir,PaTi_data_id,save_path,Col_type)}
  unlink(dir)
  return(res)
}
```

Originally I pulled the data for all the available years. It had more than 30 million rows. Here you can find the related codes commented out. We will use 2020 data only.

```
# for (i in str_which(PaTi_list$name, "parking-tickets-\\d")) {
#   print(paste(str_extract(PaTi_list$name[i],"\\d+"),"started."))
#   res <- PaTi_retrieve(i)
#   print(paste(str_extract(PaTi_list$name[i],"\\d+"),"retrieval done!"))
#   PaTi_data_tot <- PaTi_data_tot %>% rbind(res)
# }
# rm(Col_type,Pati_list,res,i,Corrupted_csv)
# saveRDS(PaTi_data_tot,"PaTi_data_tot.rds")
# PaTi_data_tot <- readRDS("PaTi_data_tot.rds") %>% filter(date_of_infraction>20200000)
PaTi_data_tot <- PaTi_retrieve(13)
```

The data is wrangled. you can find the Infraction Codes sorted with the highest occurrence (`n_tot`) at top.

```
Pati_rank <- PaTi_data_tot %>%
  group_by(infraction_code, infraction_description) %>%
  summarise(n = n()) %>%
  mutate(n_tot = sum(n),
         infraction_description2 = infraction_description[which.max(nchar(infraction_description))],
         row_num = row_number()) %>%
  filter(row_num == 1) %>%
  select(-n, -infraction_description2,-row_num) %>%
  arrange(desc(n_tot))
```

```
## `summarise()` has grouped output by 'infraction_code'. You can override using
## the `.groups` argument.
```

```
Pati_rank
```

```
## # A tibble: 156 x 3
## # Groups:   infraction_code [156]
##    infraction_code infraction_description          n_tot
##              <dbl> <chr>                           <int>
##  1               3 PARK ON PRIV PROP NO CONSENT   338199
##  2             207 PARK MACHINE-REQD FEE NOT PAID 215036
##  3               5 PARK-HWY DRNG PROH TIMES/DAYS  204658
##  4              29 PARK-SIGNED HWY-PROHIBIT DY/TM 161727
##  5               9 STOP-HWY-PROHIBITED TIMES/DAYS  69285
##  6               2 PARK - LONGER THAN 3 HOURS      67395
##  7               8 PARK-SIGNED HWY-PROHIBIT DY/TM  43255
##  8               6 PARK-SIGNED HWY-EXC PERMT TIME  39764
```

```
##  9               4 PARK ON MUN PROP NO CONSENT     38870
## 10             406 PARK-VEH. W/O VALID ONT PLATE   27665
## # ... with 146 more rows
```

I use the suggested One Address Repository for geocoding.

```
GeoC_metadata <- opendatatoronto::show_package("https://open.toronto.ca/dataset/address-points-municipal
GeoC_list <- opendatatoronto::list_package_resources("https://open.toronto.ca/dataset/address-points-mu
GeoC_retrieve <- function() {
  GeoC_data_id <- GeoC_list[[3, "id"]]
  resource <-
    resource_show(GeoC_data_id, url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/", as = "list")
  dir <- tempdir()
  resource_dir <- fs::dir_create(paste0(dir, "/", GeoC_data_id))
  save_path <-
    ckan_fetch(
      resource[["url"]],
      store = "disk",
      path = paste0(dir, "/", GeoC_data_id, "/", "res.zip")
    )
  GeoC_files <- unzip(save_path[["path"]], exdir = resource_dir)
  GeoC_shp <- st_read(GeoC_files[str_ends(GeoC_files, ".shp")])
}

GeoC_shp <- GeoC_retrieve()
```

```
## Reading layer `ADDRESS_POINT_WGS84' from data source
##   `C:\Users\nimad\AppData\Local\Temp\RtmpeOx2XL\eba07dba-8645-45f8-950c-0381a0dcaa1b\ADDRESS_POINT_W
##   using driver `ESRI Shapefile'
## Simple feature collection with 525353 features and 22 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -8865288 ymin: 5401672 xmax: -8807754 ymax: 5442900
## Projected CRS: WGS 84 / Pseudo-Mercator
```

```
GeoC_shp <- GeoC_shp %>% mutate(LFNAME_lower = tolower(LFNAME))
```

I write the code required to geocode.

```
PaTi_data_top20_split <- PaTi_data_tot %>%
  filter(infraction_code %in% Pati_rank$infraction_code[1:20]) %>%
  mutate(group = grepl('^[[:punct:]]?\\d', location2))

# PaTi_data_top20_1 <-  PaTi_data_top20_split[PaTi_data_top20_split$group,] %>%
#   group_by(infraction_code,location2) %>%
#   summarise(n=n()) %>%
#   rowwise() %>%
#   mutate(ADDRESS = str_extract(str_extract(location2,"^[:punct:]?\\d*((?<=\\d)[:alpha:](?=\\s))?"),"[
#          LFNAME_lower = tolower(str_trim(sub(ADDRESS,"",location2)))) %>%
#   left_join((GeoC_shp %>% select(GEO_ID,LINK,LFNAME_lower,ADDRESS)),by = c("LFNAME_lower","ADDRESS"))
#   group_by(infraction_code, LINK) %>%
#   summarise(n = sum(n,na.rm = TRUE)) %>%
```

```
#   mutate(accurate_type = T,
#          reference_type = "LINK") %>%
#   rename(reference = LINK)
#
# saveRDS(PaTi_data_top20_1,"PaTi_data_top20_1.rds")

PaTi_data_top20_1 <- readRDS("PaTi_data_top20_1.rds")
PaTi_data_top20_2 <- PaTi_data_top20_split[!PaTi_data_top20_split$group,] %>%
  group_by(infraction_code,location2,location4) %>%
  summarise(n=n()) %>%
  rowwise() %>%
  mutate(location2 = tolower(str_trim(location2)),
         location4 = tolower(str_trim(location4)))
```

```
## `summarise()` has grouped output by 'infraction_code', 'location2'. You can
## override using the `.groups` argument.
```

There are occasions in the ticket data that the address is written as an intersection of two roads rather than an postal address. This is a challenge since the One Address Repository does not have such information. I designed a algorithm that would search One Address Repository for each of the road in an intersection, look for the closest pair of points among them and choose that as a geolocation of the intersection. This part takes a few minutes to compute.

```
Intersection_geocoder <- function(location2,location4){
  shp1 <- GeoC_shp %>%
    filter(LFNAME_lower == location2)
  if (nrow(shp1)==0){return(NA)}
  shp1 <- shp1 %>% group_by(LINK) %>%
    filter(DISTANCE %in% c(min(DISTANCE),max(DISTANCE)))

  shp2 <- GeoC_shp %>%
    filter(LFNAME_lower == location4)
  if (nrow(shp2)==0){return(NA)}
  shp2 <- shp2 %>% group_by(LINK) %>%
    filter(DISTANCE %in% c(min(DISTANCE),max(DISTANCE)))

  dist = st_distance(shp1,shp2)
  ind = which(dist == min(dist), arr.ind = TRUE)
  location = shp1[ind[1],][["GEO_ID"]]
  return(location)
}

GeoC_shp_intersection <- PaTi_data_top20_2 %>% ungroup() %>% select(location2,location4) %>% distinct()
  mutate(GEO_ID = purrr::map2_dbl(location2,location2,Intersection_geocoder))

saveRDS(GeoC_shp_intersection,"GeoC_shp_intersection.rds")
```

```
GeoC_shp_intersection <- readRDS("GeoC_shp_intersection.rds")


PaTi_data_top20_2 <- PaTi_data_top20_2 %>% left_join(GeoC_shp_intersection,by = c("location2","location
  group_by(infraction_code, GEO_ID) %>%
```

```
  summarise(n = sum(n,na.rm = TRUE)) %>%
  mutate(accurate_type = F,
         reference_type = "GEO_ID") %>%
  rename(reference = GEO_ID)
```

```
## `summarise()` has grouped output by 'infraction_code'. You can override using
## the `.groups` argument.
```

```
PaTi_data_top20 <- PaTi_data_top20_1 %>%
  rbind(PaTi_data_top20_2) %>%
  filter(!is.na(reference)) %>%
  group_by(infraction_code) %>%
  filter(n == max(n)) %>%
  arrange(desc(n))
```

The One Address Repository has grouped addresses in LINKs that are street blocks. I use these LINKs to calculate the most common location for the top Infraction codes. Here you can see the map of the most common location for top Infraction codes. On an html-based veiwer you can hover over the map and click for more information about each locatio and infraction code.

```
fine_data <- PaTi_data_tot %>% group_by(infraction_code,set_fine_amount) %>% summarise(n=n()) %>% filte
```

```
## `summarise()` has grouped output by 'infraction_code'. You can override using
## the `.groups` argument.
```

```
PaTi_polygon_1 <- GeoC_shp %>%
  inner_join((PaTi_data_top20 %>%
                filter(reference_type == "LINK") %>%
                select(infraction_code,reference,no_infraction = n)),
              by = c("LINK"="reference")) %>%
  left_join((Pati_rank %>% select(starts_with("infraction"))),by = "infraction_code") %>%
  group_by(infraction_code, no_infraction, infraction_description, LFNAME) %>%
  summarise(n_link=n())
```

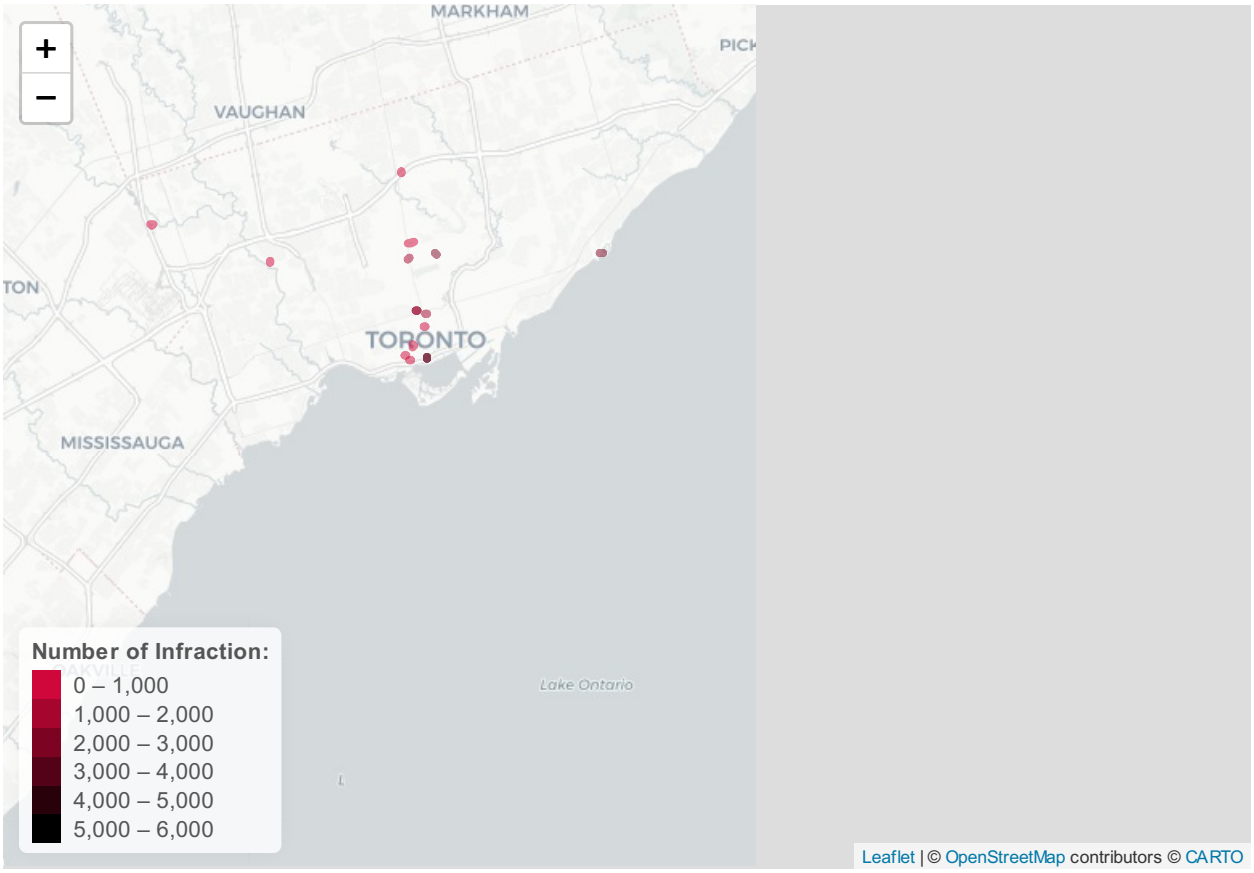
```
## `summarise()` has grouped output by 'infraction_code', 'no_infraction',
## 'infraction_description'. You can override using the `.groups` argument.
```

```
PaTi_polygon_2 <- GeoC_shp %>%
  inner_join((PaTi_data_top20 %>%
                filter(reference_type == "GEO_ID") %>%
                select(infraction_code,reference,no_infraction = n)),
              by = c("GEO_ID"="reference")) %>%
  left_join((Pati_rank %>% select(starts_with("infraction"))),by = "infraction_code") %>%
  group_by(infraction_code, no_infraction, infraction_description, LFNAME) %>%
  summarise(n_link=n())
```

```
## `summarise()` has grouped output by 'infraction_code', 'no_infraction',
## 'infraction_description'. You can override using the `.groups` argument.
```

```r
PaTi_polygon_make <- function(data) {
  if (data$n_link[1] == 1) {
    data %>%
      # summarise(n_link = n()) %>%
      # filter(n_link == 1) %>%
      st_transform(2952) %>%
      st_buffer(dist = 10) %>%
      st_transform(4326) %>%
      st_cast("POLYGON")
  } else if (data$n_link[1] == 2) {
    data %>%
      # summarise(n_link = n()) %>%
      st_cast("LINESTRING") %>%
      st_transform(2952) %>%
      st_buffer(dist = 10) %>%
      st_transform(4326)
  } else if (data$n_link[1] == 3) {
    temp = data %>% st_coordinates()
    st_geometry(data) = st_geometry(st_as_sf(as.data.frame(temp[c(1,1,2,3),]),coords = c("X","Y"),crs =
    data %>%
      st_cast("POLYGON") %>%
      st_convex_hull() %>%
      st_transform(4326)
  } else {
    data %>%
      st_cast("POLYGON") %>%
      st_convex_hull() %>%
      st_transform(4326)
  }
}

## For some reason sf package does not let me have poygons made from point buffers be in the same df tha

PaTi_polygon_pl <- PaTi_polygon_1 %>%
  rbind(PaTi_polygon_2) %>%
  filter(n_link > 1) %>%
  PaTi_polygon_make() %>%
  left_join(fine_data,by = "infraction_code")

PaTi_polygon_pt <- PaTi_polygon_1 %>%
  rbind(PaTi_polygon_2) %>%
  filter(n_link == 1) %>%
  PaTi_polygon_make() %>%
  left_join(fine_data,by = "infraction_code")


# leaf_sf(tt[1,] %>% st_transform(4326)) %>%
#   addPolygons(data = tt2[1,],
#               popup=paste("Stratum:",tt[1,]$infraction_code,"<br>"),
#               label = ~ paste0("Total Income: ", infraction_code))
#
# leaflet() %>%
#   addProviderTiles(
```

```
#     "OpenStreetMap",
#     # give the layer a name
#     group = "OpenStreetMap"
#   ) %>%
#   addPolygons(data = tt2[1,],
#               popup=paste("Stratum:",tt[1,]$infraction_code,"<br>"),
#               label = ~ paste0("Total Income: ", infraction_code))
pal <- colorBin(colorRampPalette(c("#d0073a","#000000"))(20),
                domain = PaTi_polygon_pl$no_infraction)

PaTi_plot <- leaflet() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addPolygons(data = PaTi_polygon_pl,
              popup=paste(#"<b>Infraction Code</b>",PaTi_polygon_pl$infraction_code,"<br>",
                          "<b>Description:</b>",PaTi_polygon_pl$infraction_description,"<br>",
                          "<b>Location:</b>",PaTi_polygon_pl$LFNAME,"<br>",
                          "<b>Number of Infraction:</b>",PaTi_polygon_pl$no_infraction,"<br>",
                          "<b>Infraction Fine:</b>",dollar_format()(PaTi_polygon_pl$set_fine_amount),"<
              label = ~ paste0("Infraction Code ", infraction_code),
              color = #colorRampPalette(c("#d0073a","#000000")))
                ~ pal(no_infraction)) %>%
  addPolygons(data = PaTi_polygon_pt,
              popup=paste(
                          "<b>Description:</b>",PaTi_polygon_pt$infraction_description,"<br>",
                          "<b>Location:</b>",PaTi_polygon_pt$LFNAME,"<br>",
                          "<b>Number of Infraction:</b>",PaTi_polygon_pt$no_infraction,"<br>",
                          "<b>Infraction Fine:</b>",dollar_format()(PaTi_polygon_pt$set_fine_amount),"<
              label = ~ paste0("Infraction Code ", infraction_code),
              color = ~ pal(no_infraction)) %>%
  addLegend(
    data = PaTi_polygon_pl,
    pal = pal,
    values = ~no_infraction,
    position = "bottomleft",
    title = "Number of Infraction:",
    opacity = 1
  )

PaTi_plot
```



Green P parking

I pull the json file from the suggested link for Green P parkings.

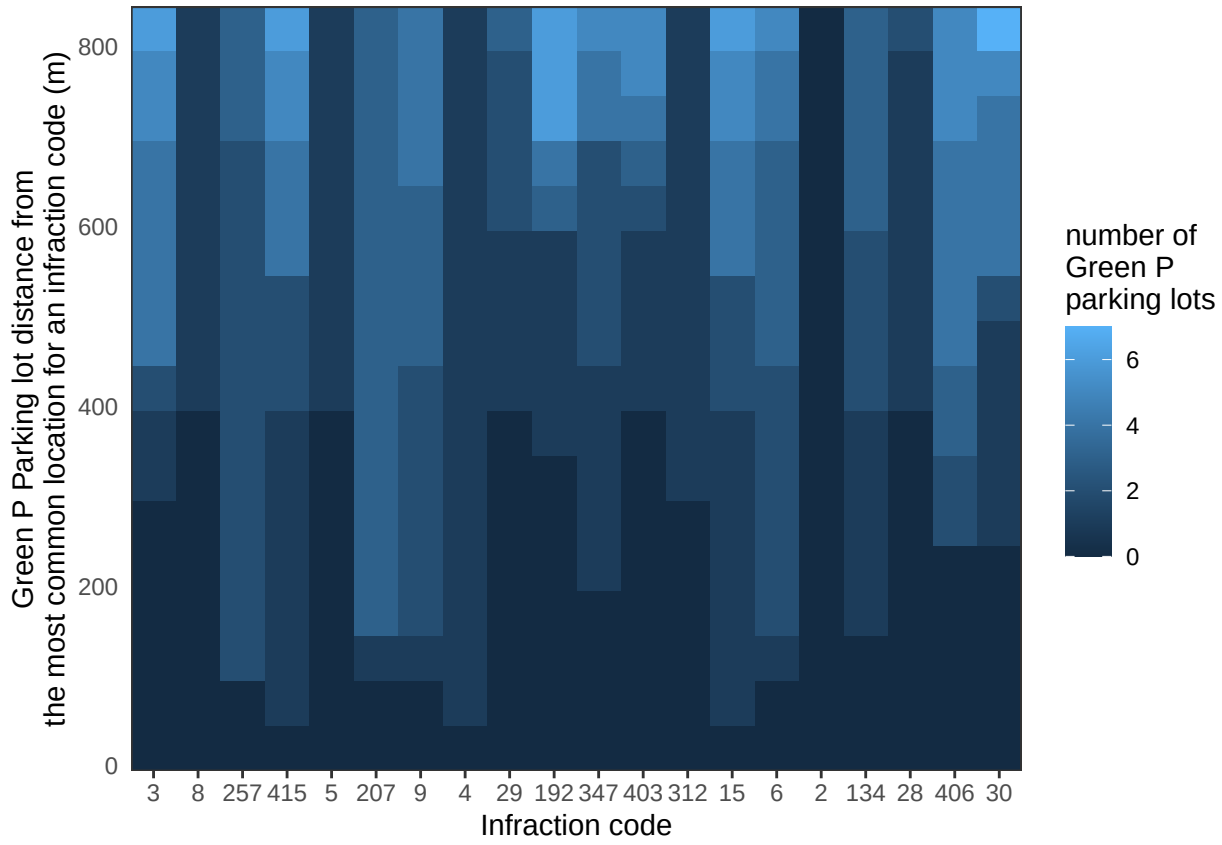
```
GP_metadata <- opendatatoronto::show_package("https://open.toronto.ca/dataset/green-p-parking/")
GP_list <- opendatatoronto::list_package_resources("https://open.toronto.ca/dataset/green-p-parking/")
GP_retrieve <- function(URL,No,format) {
  list <- opendatatoronto::list_package_resources(URL)
  data_id <- list[[No, "id"]]
  resource <-
    resource_show(data_id, url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/", as = "list")
  dir <- tempdir()
  resource_dir <- fs::dir_create(paste0(dir, "/", data_id))
  save_path <-
    ckan_fetch(
      resource[["url"]],
      store = "disk",
      path = paste0(dir, "/", data_id, "/", "res.",format)
    )
  GP_json <- fromJSON(file = save_path[["path"]])
}

GP_json <- GP_retrieve("https://open.toronto.ca/dataset/green-p-parking/",1,"JSON")

GP_data <- tibble(id = 1:length(GP_json[[1]])) %>%
```

```
  mutate(address = map_chr(id,~ GP_json[[1]][[.x]]$address),
         lat = as.double(map_chr(id,~ GP_json[[1]][[.x]]$lat)),
         lng = as.double(map_chr(id,~ GP_json[[1]][[.x]]$lng)),
         rate = map_chr(id,~ GP_json[[1]][[.x]]$rate))
GP_sf <- GP_data %>%
  st_as_sf(coords = c("lng", "lat"),
           crs = st_crs(4326))
```

Not all Green P locations are close to the common infraction locations. Here you can see a graph that provides information about the distribution of Green P off-street parkings close to (up to 800 meters) each common infraction location.

```
PaTi_GP <- tibble()
PaTi_GP_over <- function(r) {
  GP_sf_buf <- GP_sf %>%
    st_transform(2952) %>%
    st_buffer(dist = r) %>%
    st_transform(4326)

  sparse_pl <-
    PaTi_polygon_pl %>% st_intersects(GP_sf_buf, sparse = T)
  over_pl <-
    tibble(infraction_code = PaTi_polygon_pl$infraction_code) %>%
    mutate(
      row_num = row_number(),
      GP_distance = r,
      no_GP = map_dbl(row_num,  ~ length(sparse_pl[[.x]]))
    )

  sparse_pt <-
    PaTi_polygon_pt %>% st_intersects(GP_sf_buf, sparse = T)
  over_pt <-
    tibble(infraction_code = PaTi_polygon_pt$infraction_code) %>%
    mutate(
      row_num = row_number(),
      GP_distance = r,
      no_GP = map_dbl(row_num,  ~ length(sparse_pt[[.x]]))
    )

  res <- over_pl %>% rbind(over_pt)
}

for (r in seq(0,800,by=50)) {
  PaTi_GP <- PaTi_GP %>% rbind(PaTi_GP_over(r))
}


PaTi_GP_tile <- PaTi_GP %>%
  ggplot()+
  geom_raster(aes(x=factor(infraction_code,levels = PaTi_data_top20$infraction_code),y=GP_distance,fill=
  theme_bw()+
  theme(axis.text.y = element_text(vjust = 1.5),
        axis.ticks.y = element_blank()#,
```

```
        # panel.grid = element_blank(),
        # panel.border = element_blank()
        )+
  scale_x_discrete(name = "Infraction code",expand=c(0,0))+
  scale_y_continuous(name = "Green P Parking lot distance from\nthe most common location for an infracti
  scale_fill_continuous(name = "number of\nGreen P\nparking lots")

plot(PaTi_GP_tile)
```



Here you can see the map of common infraction locations with their closest Green P parking connected to them with a line. On an html-based veiwer you can hover and click on the line, Green P Parking to gain more information.

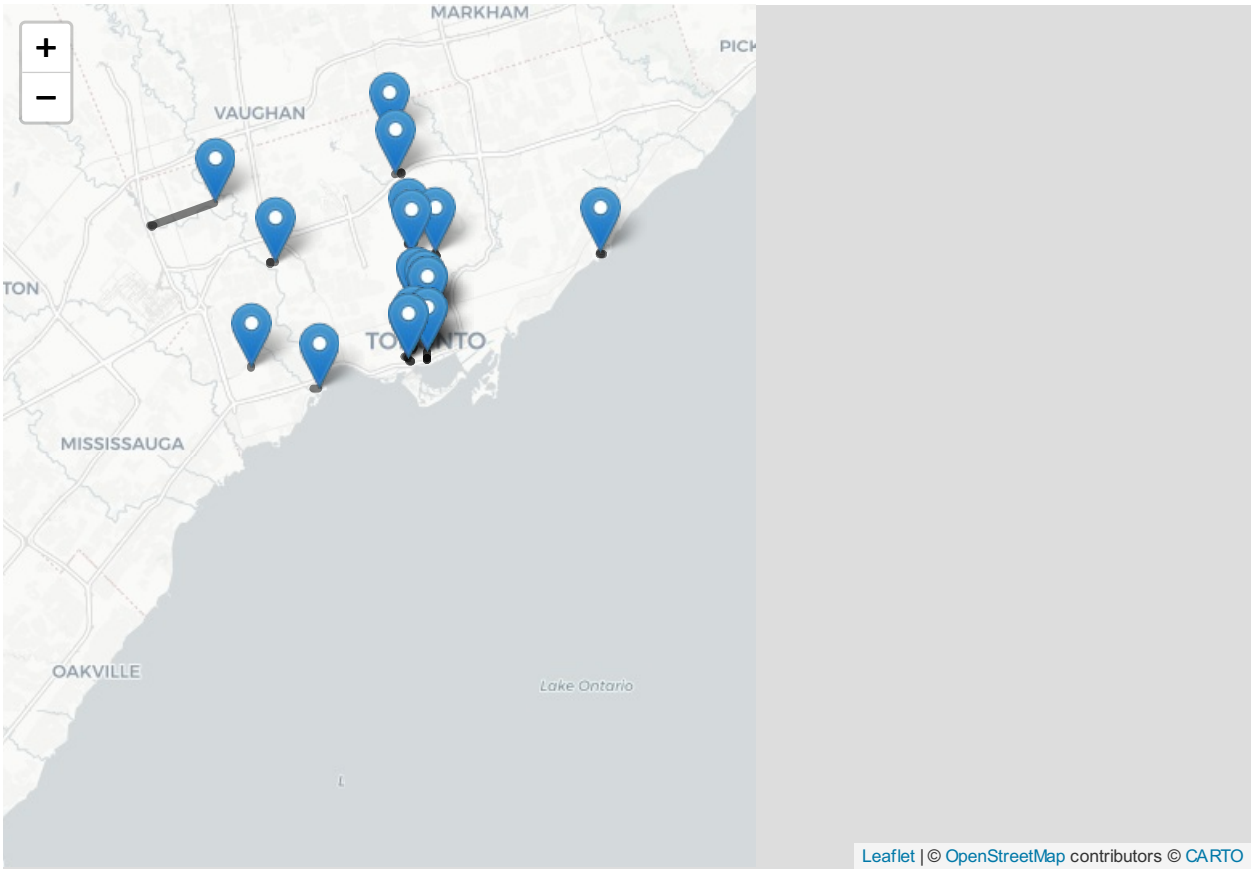
```
PaTi_polygon_center <- PaTi_polygon_pl %>% st_centroid() %>% rbind(PaTi_polygon_pt %>% st_centroid())
```

```
## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x

## Warning in st_centroid.sf(.): st_centroid assumes attributes are constant over
## geometries of x
```

```
dist_mat <- GP_sf %>% st_distance(PaTi_polygon_center)

PaTi_polygon_center <- PaTi_polygon_center %>%
  ungroup() %>%
```

```
  mutate(row_num = row_number()) %>%
  mutate(
    closest_GP = map_dbl(row_num, ~ which.min(dist_mat[,.x])),
    GP_dist = map_dbl(row_num, ~ min(dist_mat[,.x])),
    GP_lon = map_dbl(closest_GP, ~ GP_data[GP_data$id==.x,]$lng),
    GP_lat = map_dbl(closest_GP, ~ GP_data[GP_data$id==.x,]$lat))

PaTi_polygon_center <- PaTi_polygon_center %>%
  cbind(PaTi_polygon_center %>%
          st_coordinates() %>%
          as_tibble()) %>%
  mutate(color = paletteer_c("grDevices::rainbow", 20))

PaTi_polygon_pl <- PaTi_polygon_pl %>% left_join(PaTi_polygon_center %>% select(closest_GP,GP_dist,infra


## Joining, by = "infraction_code"

PaTi_polygon_pt <- PaTi_polygon_pt %>% left_join(PaTi_polygon_center %>% select(closest_GP,GP_dist,infra


## Joining, by = "infraction_code"

GP_sf_closest <- GP_sf %>% inner_join(PaTi_polygon_center %>% st_drop_geometry(),by = c("id"="closest_GP

PaTi_GP_map <- leaflet() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addPolygons(data = PaTi_polygon_pl,
              popup=paste(#"<b>Infraction Code</b>",PaTi_polygon_pl$infraction_code,"<br>",
                          "<b>Description:</b>",PaTi_polygon_pl$infraction_description,"<br>",
                          "<b>Location:</b>",PaTi_polygon_pl$LFNAME,"<br>",
                          "<b>Number of Infraction:</b>",PaTi_polygon_pl$no_infraction,"<br>",
                          "<b>Infraction Fine:</b>",dollar_format()(PaTi_polygon_pl$set_fine_amount),"<b
              label = ~ paste0("Infraction Code ", infraction_code),
              color = ~ color) %>%
  addPolygons(data = PaTi_polygon_pt,
              popup=paste(
                          "<b>Description:</b>",PaTi_polygon_pt$infraction_description,"<br>",
                          "<b>Location:</b>",PaTi_polygon_pt$LFNAME,"<br>",
                          "<b>Number of Infraction:</b>",PaTi_polygon_pt$no_infraction,"<br>",
                          "<b>Infraction Fine:</b>",dollar_format()(PaTi_polygon_pt$set_fine_amount),"<b
              label = ~ paste0("Infraction Code ", infraction_code),
              color = ~ color) %>%
  # addPolygons(data = GP_sf_closest_buf,
  #             popup=paste(
  #                         "<b>Address:</b>",GP_sf_closest_buf$address,"<br>",
  #                         "<b>Rate:</b>",GP_sf_closest_buf$rate),
  #             label = ~ paste0("Green P id: ", id),
  #             color = ~ color) %>%
  addMarkers(data = GP_sf_closest,
             # icon = makeIcon("https://www.iconpacks.net/icons/2/free-location-icon-2955-thumb.png", i
              #   icon = makeIcon("my-icon.png", iconWidth = 16, iconHeight = 16),
              popup=paste(
                          "<b>Address:</b>",GP_sf_closest$address,"<br>",
```

```
                          "<b>Rate:</b>",GP_sf_closest$rate),
              label = ~ paste0("Green P id: ", id))


for (i in 1:nrow(PaTi_polygon_center)){
PaTi_GP_map <- PaTi_GP_map %>%
  addPolylines(
      lng = c(PaTi_polygon_center$X[i],PaTi_polygon_center$GP_lon[i]),
      lat = c(PaTi_polygon_center$Y[i],PaTi_polygon_center$GP_lat[i]),
      color = PaTi_polygon_center$color[i],
      popup=paste("<b>Distance:</b>",round(PaTi_polygon_center$GP_dist[i]),"m","<br>"))
}

PaTi_GP_map
```



Demographics

Here, I pull the data about the neighbourhoods.

```
ngbh_metadata <- opendatatoronto::show_package("https://open.toronto.ca/dataset/neighbourhoods/")
ngbh_list <- opendatatoronto::list_package_resources("https://open.toronto.ca/dataset/neighbourhoods/")
ngbh_retrieve <- function(URL,No,format) {
  list <- opendatatoronto::list_package_resources(URL)
  data_id <- list[[No, "id"]]
  resource <-
    resource_show(data_id, url = "https://ckan0.cf.opendata.inter.prod-toronto.ca/", as = "list")
  dir <- tempdir()
```

```
  resource_dir <- fs::dir_create(paste0(dir, "/", data_id))
  save_path <-
    ckan_fetch(
      resource[["url"]],
      store = "disk",
      path = paste0(dir, "/", data_id, "/", "res.",format)
    )
  files <- unzip(save_path[["path"]], exdir = resource_dir)
  res <- st_read(files[str_ends(files, ".shp")])
  res_names <- read.csv(files[str_ends(files, ".csv")])
  colnames(res) <- res_names$name
  return(res)
}

URL = "https://open.toronto.ca/dataset/neighbourhoods/"
No = 5
format = "zip"

ngbh_shp <- ngbh_retrieve("https://open.toronto.ca/dataset/neighbourhoods/",5,"zip")
```

```
## Reading layer `Neighbourhoods - historical 140' from data source
##   `C:\Users\nimad\AppData\Local\Temp\RtmpeOx2XL\d4b0aa31-19a7-47ff-a828-6893e5c17eec\Neighbourhoods
##   using driver `ESRI Shapefile'
## Simple feature collection with 140 features and 11 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -79.63926 ymin: 43.581 xmax: -79.11527 ymax: 43.85546
## Geodetic CRS:  WGS 84
```

```
ngbh_prof_metadata <- opendatatoronto::show_package("https://open.toronto.ca/dataset/neighbourhood-profi
ngbh_prof_list <- opendatatoronto::list_package_resources("https://open.toronto.ca/dataset/neighbourhood
ngbh_prof_data <- ngbh_prof_list[2,] %>% get_resource()
```

Here I pull information about socio-economical features of the neighbourhoods and compare the ones with High Infraction or not. I use Logistic regression. The result shows that these features needs refinement. Perhaps better features should be obtained.

```
sparse_neigh <- PaTi_polygon_center %>% st_intersects(ngbh_shp)
is.na(sparse_neigh) <- lengths(sparse_neigh) == 0
PaTi_polygon_center <- PaTi_polygon_center %>% ungroup() %>% mutate(neighbourhood = ngbh_shp[unlist(spa
PaTi_polygon_center <- PaTi_polygon_center %>% mutate(neighbourhood = str_remove(neighbourhood,"\\(\\d+
                                neighbourhood = str_replace_all(neighbourhood,"([:alpha:]+)","\\1\\."),
                                neighbourhood = str_remove_all(neighbourhood,"[^a-zA-Z\\.]"),
                                neighbourhood = str_remove_all(neighbourhood,"\\.$"),
                                )
#8 Population density per square kilometre
#947  Median total income in 2015 among recipients ($)
#960 Average total income in 2015 among recipients ($)
#1019 Income Median total income of households in 2015 ($)
# 1152  Immigration and citizenship Non-immigrants
# 1153  Immigration and citizenship Immigrants
#1629   Housing Owner
```

```
# 1630  Housing Renter

#1703   Education   Total - Highest certificate, diploma or degree for the population aged 15 years an
#1710   Education   University certificate, diploma or degree at bachelor level or above

#1715   Education   Total - Highest certificate, diploma or degree for the population aged 25 to 64 ye
#1725   Education   University certificate, diploma or degree at bachelor level or above



#1966   Journey to work Total - Main mode of commuting for the employed labour force aged 15 years and



# 2368  Mobility Total - Mobility status 1 year ago - 25% sample data
# 2369  Mobility Non-movers
# 2370  Mobility Movers



ngbh_filtered_data <- ngbh_prof_data %>% filter(X_id %in% c(8,1152,1153,1629,1630,1703,1710,1715,1725,19
  pivot_longer(cols = City.of.Toronto:Yorkdale.Glen.Park,
                                values_to = "value",
                                names_to = "neighbourhood")

ngbh_filtered_data_wide <- ngbh_filtered_data %>%
  select(X_id,neighbourhood,value) %>%
  mutate(value = as.numeric(gsub(",", "", value))) %>%
  pivot_wider(id_cols = c("neighbourhood"),
              names_from = "X_id",
              values_from = "value")
ngbh_filtered_data_wide <- ngbh_filtered_data_wide %>%
  mutate(pop_density = `8`/max(`8`),
    Immigration_rate = `1153`/(`1153`+`1152`),
         rent_rate = `1630`/(`1630`+`1629`),
         young_University_rate = `1710`/(`1703`),
         middle_University_rate = `1725`/(`1715`),
         driver_rate = `1967`/(`1966`),
         newly_moved_rate = `2370`/(`2370`+`2369`)) %>%
  select(-c(`8`,`1152`,`1153`,`1629`,`1630`,`1703`,`1710`,`1715`,`1725`,`1966`,`1967`,`2368`,`2369`,`23

ngbh_filtered_data_wide <- ngbh_filtered_data_wide %>% mutate(High_infraction = neighbourhood %in% PaTi

logistic_model <- glm(High_infraction ~ ., family = binomial(), ngbh_filtered_data_wide %>% select(-neig

summary(logistic_model)
```

```
##
## Call:
## glm(formula = High_infraction ~ ., family = binomial(), data = ngbh_filtered_data_wide %>%
##     select(-neighbourhood))
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3520  -0.3383  -0.2305  -0.1625   2.8970
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.3266     4.4678  -2.088   0.0368 *
## pop_density               0.5105     4.3052   0.119   0.9056
## Immigration_rate         -0.3452     3.4298  -0.101   0.9198
## rent_rate                -0.5878     3.9615  -0.148   0.8820
## young_University_rate     5.9963    19.6821   0.305   0.7606
## middle_University_rate   -0.8542    16.7041  -0.051   0.9592
## driver_rate               3.9795     4.8668   0.818   0.4135
## newly_moved_rate         21.3222    10.3165   2.067   0.0388 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 82.081  on 140  degrees of freedom
## Residual deviance: 59.815  on 133  degrees of freedom
## AIC: 75.815
##
## Number of Fisher Scoring iterations: 6
```