Parking infractions, City of Toronto take home assignment

Nima Jamshidi

Contents

Parking Tickets 1
Green P parking 10
Demographics 16

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))

options(dplyr.summarise.inform = FALSE)
```

Parking Tickets

Here I wrote a function for getting data for any year available on the webpage. There are a few notes:

- `opendatatoronto` package has problems with pulling `.csv` files. This is done manually here.
- For some of the years, the `.csv` file is corrupt. Its Unicode requires adjustment. This function address 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 of 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 in the table below.

```
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))

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-toronto-one-address-repository/")

GeoC_list <-
  opendatatoronto::list_package_resources(
    "https://open.toronto.ca/dataset/address-points-municipal-toronto-one-address-repository/")

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")],quiet = T)
}

GeoC_shp <- GeoC_retrieve()
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))?"),
#       "[^[:punct:]]+"),
#          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)))
```

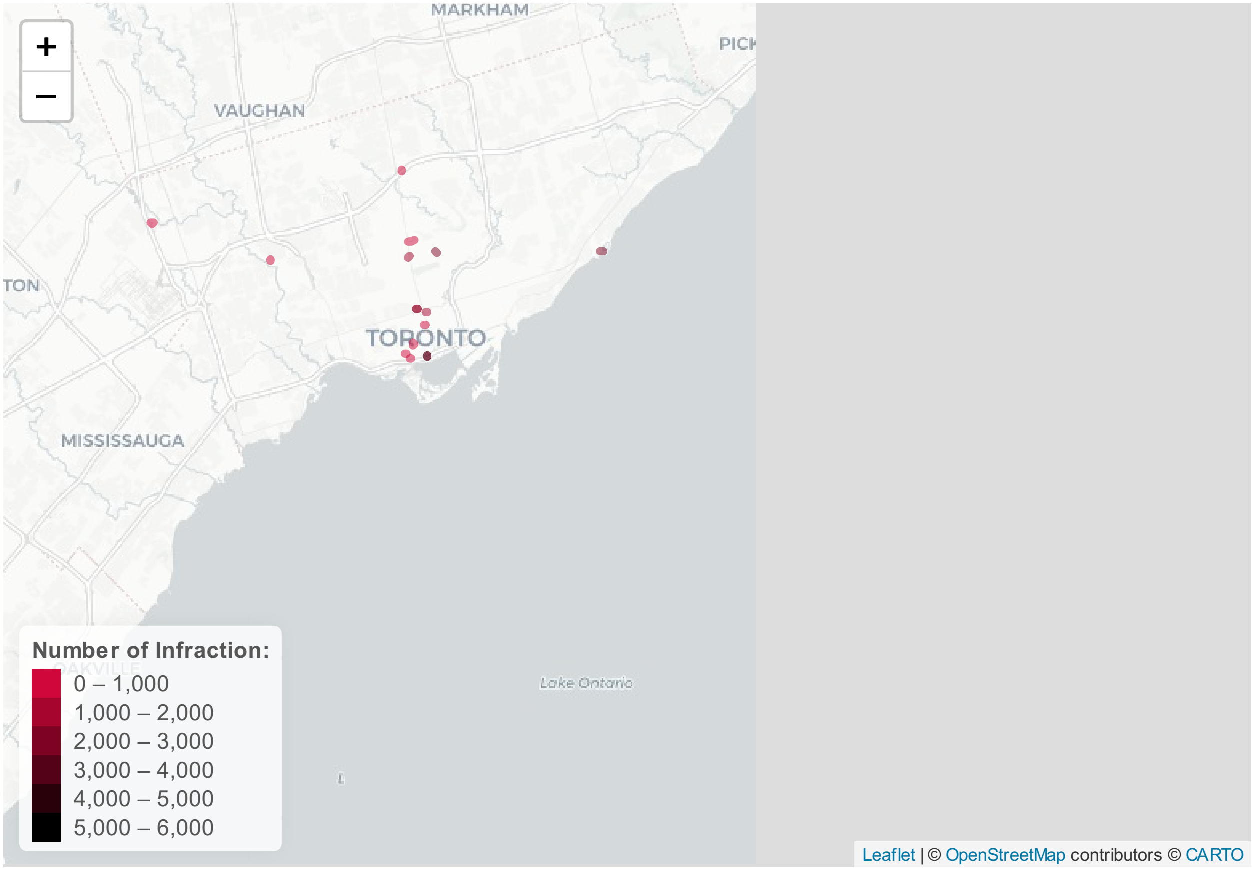
There are occasions in the ticket data that the address is written as an intersection of two roads rather than a postal address. This is a challenge since the One Address Repository does not have such information. I designed an algorithm that would search the One Address Repository for each of the roads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

```r
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")
```

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


PaTi_data_top20_2 <- PaTi_data_top20_2 %>%
  left_join(GeoC_shp_intersection,
            by = c("location2","location4")) %>%
  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)

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 a feature called `LINK` that is a representation of a street block. the postal addresses from the same block in this repo has the same LINK. I use these `LINK`s to calculate the most common location for the top Infraction codes (please view the graph below). On an html-based viewer you can hover over the map and click for more information about each location and infraction code.

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

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())

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())

PaTi_polygon_make <- function(data) {
  if (data$n_link[1] == 1) {
    data %>%
      st_transform(2952) %>%
      st_buffer(dist = 10) %>%
      st_transform(4326) %>%
      st_cast("POLYGON")
  } else if (data$n_link[1] == 2) {
    data %>%
      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 = st_crs(data)
    ) %>% summarise())
    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 that contains
# regular polygons, so we split them:

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")


pal <- colorBin(
  colorRampPalette(c("#d0073a", "#000000"))(20),
  domain = PaTi_polygon_pl$no_infraction)

PaTi_plot <- leaflet() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addPolygons(
    data = PaTi_polygon_pl,
    popup = paste(
      "<b>Description:</b>",
      PaTi_polygon_pl$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$set_fine_amount),
      "<br>"
    ),
    label = ~ paste0("Infraction Code ", infraction_code),
    color =
      ~ pal(no_infraction)
  ) %>%
  addPolygons(
    data = PaTi_polygon_pt,
    popup = paste(
      "<b>Description:</b>",
      PaTi_polygon_pt$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),
      "<br>"
    ),
    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
```



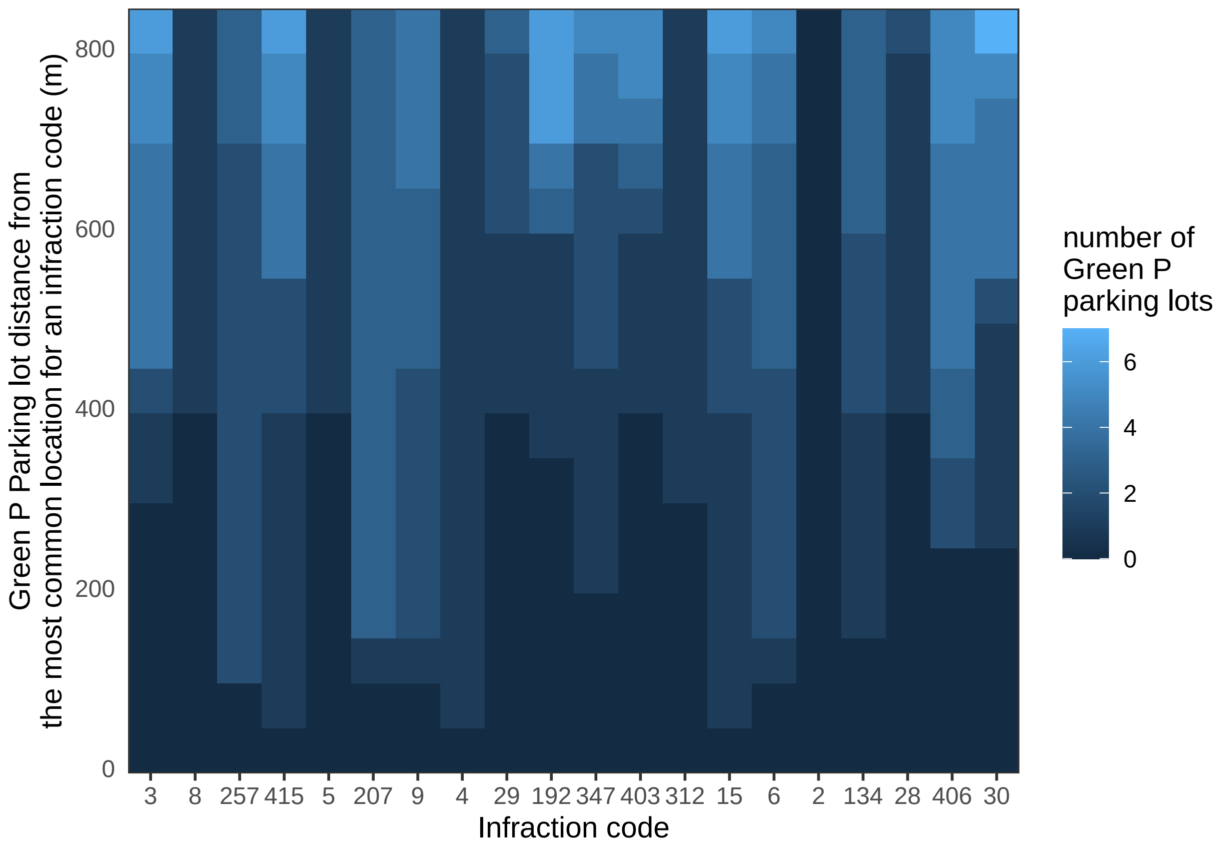
Green P parking

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))
```

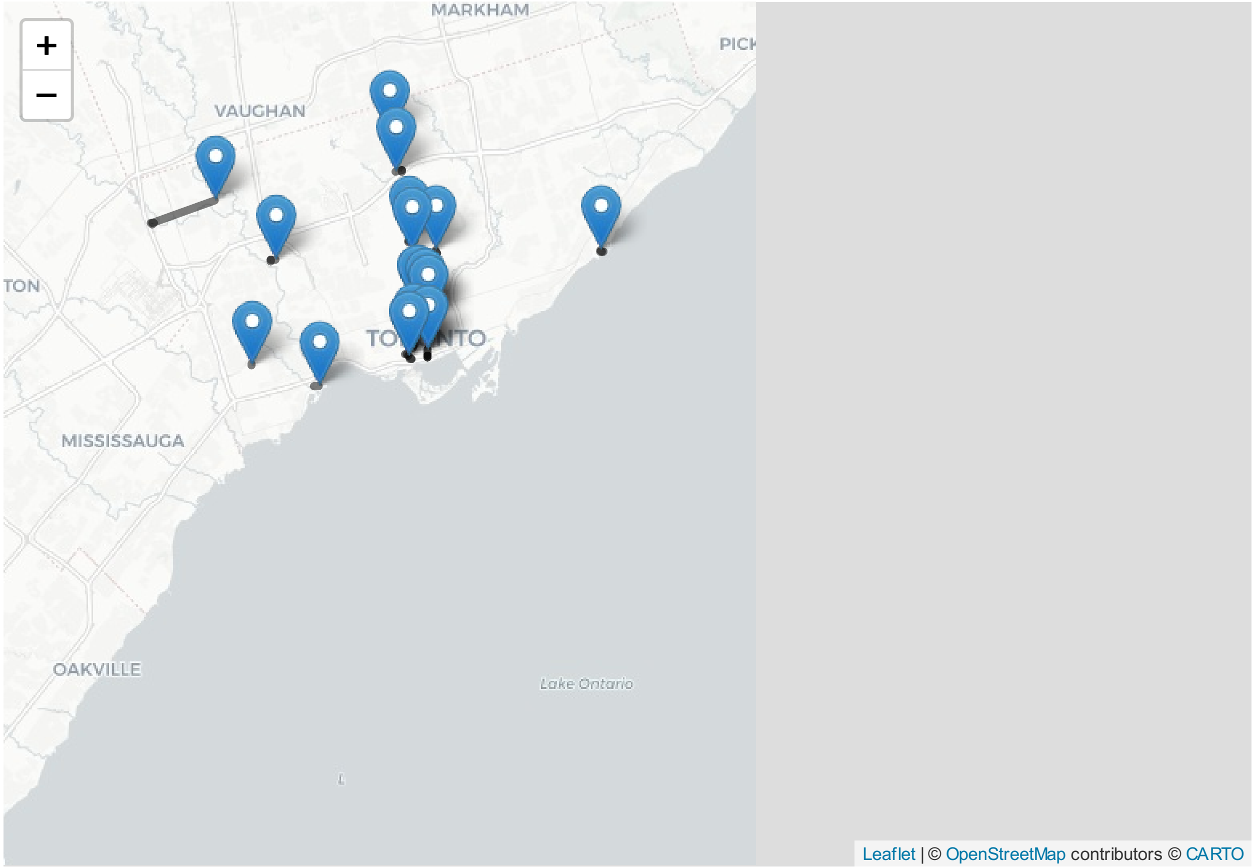
Not all Green P locations are close to the common infraction locations. Here you can see a graph that provides information about the distribution of Green P off-street parkings close to (up to 800 meters) each common infraction location.

```
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 = no_GP
  )) +
  theme_bw() +
  theme(axis.text.y = element_text(vjust = 1.5),
        axis.ticks.y = 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
                       c(0, 0)) +
  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 viewer 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())

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,
             infraction_code, color) %>% st_drop_geometry()
  )

PaTi_polygon_pt <-
  PaTi_polygon_pt %>% left_join(
    PaTi_polygon_center %>%
      select(closest_GP,
             GP_dist,
             infraction_code, color) %>% st_drop_geometry()
  )

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

PaTi_GP_map <- leaflet() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addPolygons(
    data = PaTi_polygon_pl,
    popup = paste(
      "<b>Description:</b>",
      PaTi_polygon_pl$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$set_fine_amount),
      "<br>"
    ),
    label = ~ paste0("Infraction Code ", infraction_code),
    color = ~ color
  ) %>%
  addPolygons(
    data = PaTi_polygon_pt,
    popup = paste(
      "<b>Description:</b>",
      PaTi_polygon_pt$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$set_fine_amount),
      "<br>"
    ),
    label = ~ paste0("Infraction Code ", infraction_code),
    color = ~ color
  ) %>%
  addMarkers(
    data = GP_sf_closest,
    popup = paste(
      "<b>Address:</b>",
      GP_sf_closest$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")], quiet = T)
  res_names <- read.csv(files[str_ends(files, ".csv")])
  colnames(res) <- res_names$name
  return(res)
}

ngbh_shp <-
  ngbh_retrieve(
    "https://open.toronto.ca/dataset/neighbourhoods/",
    5,
    "zip")
ngbh_prof_metadata <-
  opendatatoronto::show_package(
    "https://open.toronto.ca/dataset/neighbourhood-profiles/")
ngbh_prof_list <-
  opendatatoronto::list_package_resources(
    "https://open.toronto.ca/dataset/neighbourhood-profiles/")
ngbh_prof_data <- ngbh_prof_list[2, ] %>% get_resource()
```

Here I pull information about socio-economic features of the neighbourhoods and compare the ones with High Infraction or not. The table below provides information about the average of the whole City of Toronto alongside the High infraction neighbourhoods. The features used are described below:

- `pop_density_standardized`: Population density per square kilometre devided by the maximum among the neighbourhoods.
- `Immigration_rate`: Rate of Immigration and citizenship Immigrants
- ‘rent_rate“: Fraction of population as Housing Renter
- `young_University_rate`: fraction of people between 15 to 24 with a university certificate, diploma or degree at bachelor level or above
- `middle_University_rate`: fraction of people between 25 to 64 with a university certificate, diploma or degree at bachelor level or above
- `driver_rate`:Fraction of commuters that drive a car, truck or etc. to work.
- `newly_moved_rate`: Fraction of people that moved to the neighbourhood in the past year.
- `High_infraction`: If one or more of the common locations for each of the top Infraction codes is in the neighbourhood.

```
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(sparse_neigh), ]$AREA_NAME)
PaTi_polygon_center <-
  PaTi_polygon_center %>% mutate(
    neighbourhood =
      str_remove(neighbourhood, "\\(\\d+\\)"),
    neighbourhood =
      str_replace_all(
        neighbourhood, "([:alpha:]+)", "\\1\\."),
    neighbourhood =
      str_remove_all(
        neighbourhood, "[^a-zA-Z\\.]"),
    neighbourhood =
      str_remove_all(neighbourhood, "\\.$"),
  )
#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,
    1966,
    1967,
    2368,
    2369,
    2370
  )) %>%
  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_standardized = `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`,
      `2370`
    )
  )

ngbh_filtered_data_wide <-
  ngbh_filtered_data_wide %>%
  mutate(High_infraction =
           neighbourhood %in% PaTi_polygon_center$neighbourhood)

ngbh_filtered_data_wide %>%
  filter(neighbourhood == "City.of.Toronto" |
                                   High_infraction == T)
```

```
## # A tibble: 13 x 9
## neighbourhood pop_d~1 Immig~2 rent_~3 young~4 middl~5
drive~6 newly~7 High_~8
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>
## 1 City.of.Toro~ 0.0978 0.487 0.472 0.364 0.441 0.460
0.142 FALSE
## 2 Annex 0.245 0.303 0.620 0.631 0.706 0.221 0.225 TRUE
## 3 Bay.Street.C~ 0.318 0.511 0.679 0.630 0.790 0.151
0.349 TRUE
## 4 Church.Yonge~ 0.520 0.396 0.705 0.582 0.639 0.149
0.271 TRUE
## 5 Islington.Ci~ 0.0612 0.466 0.420 0.405 0.481 0.534
0.151 TRUE
## 6 Leaside.Benn~ 0.0811 0.204 0.280 0.574 0.691 0.604
0.0852 TRUE
## 7 Mount.Pleasa~ 0.122 0.275 0.434 0.578 0.687 0.462
0.127 TRUE
## 8 Waterfront.C~ 0.202 0.391 0.587 0.673 0.719 0.208
0.316 TRUE
## 9 West.Humber.~ 0.0252 0.610 0.318 0.222 0.283 0.606
0.124 TRUE
## 10 Weston 0.162 0.506 0.631 0.164 0.196 0.514 0.129 TRUE
## 11 Willowdale.E~ 0.228 0.687 0.381 0.566 0.678 0.454
0.247 TRUE
## 12 Willowdale.W~ 0.131 0.611 0.364 0.506 0.635 0.472
0.214 TRUE
## 13 Yonge.Eglint~ 0.162 0.282 0.613 0.626 0.706 0.332
0.198 TRUE
## # ... with abbreviated variable names 1:
pop_density_standardized,
## # 2: Immigration_rate, 3: rent_rate, 4:
young_University_rate,
## # 5: middle_University_rate, 6: driver_rate, 7:
newly_moved_rate,
## # 8: High_infraction
```

I use Logistic regression to compare neighbourhoods with high infraction with others. The result shows that these features need refinement. Perhaps better features should be obtained.

```r
logistic_model <-
  glm(
    High_infraction ~ .,
    family = binomial(),
    ngbh_filtered_data_wide %>%
      filter(neighbourhood != "City.of.Toronto") %>%
      select(-neighbourhood)
  )

summary(logistic_model)
```

```
##
## Call:
## glm(formula = High_infraction ~ ., family = binomial(),
data = ngbh_filtered_data_wide %>%
## filter(neighbourhood != "City.of.Toronto") %>%
select(-neighbourhood))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -1.3514 -0.3421 -0.2319 -0.1637 2.8911
```

```
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.2797 4.4563 -2.082 0.0373 *
## pop_density_standardized 0.4636 4.3136 0.107 0.9144
## Immigration_rate -0.3325 3.4218 -0.097 0.9226
## rent_rate -0.5814 3.9473 -0.147 0.8829
## young_University_rate 5.9958 19.6469 0.305 0.7602
## middle_University_rate -0.8818 16.6740 -0.053 0.9578
## driver_rate 3.9404 4.8566 0.811 0.4172
## newly_moved_rate 21.2698 10.3007 2.065 0.0389 *
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 81.902 on 139 degrees of freedom
## Residual deviance: 59.722 on 132 degrees of freedom
## AIC: 75.722
##
## Number of Fisher Scoring iterations: 6
```