CHANGE TITLE

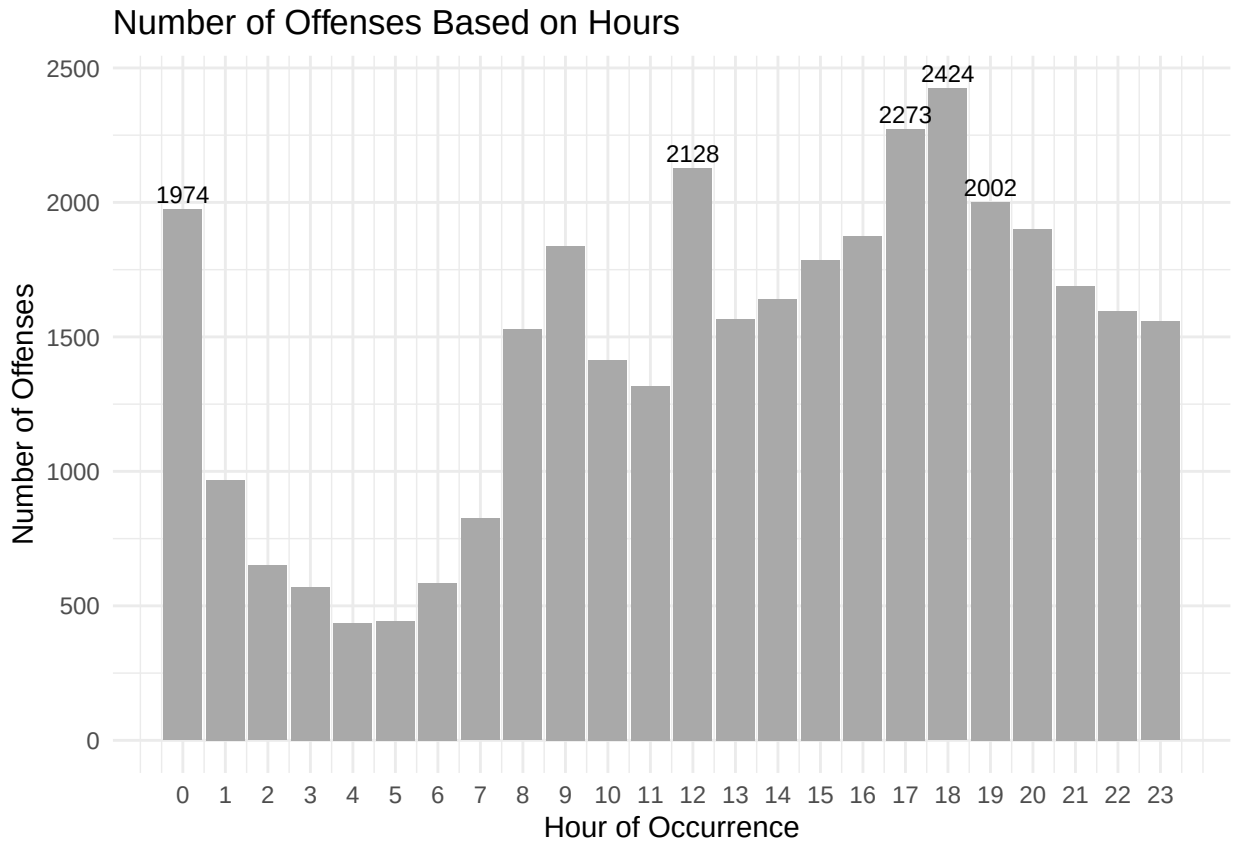
Ruibo Sun and Nixi Huang

2024-03-14 00:42:32

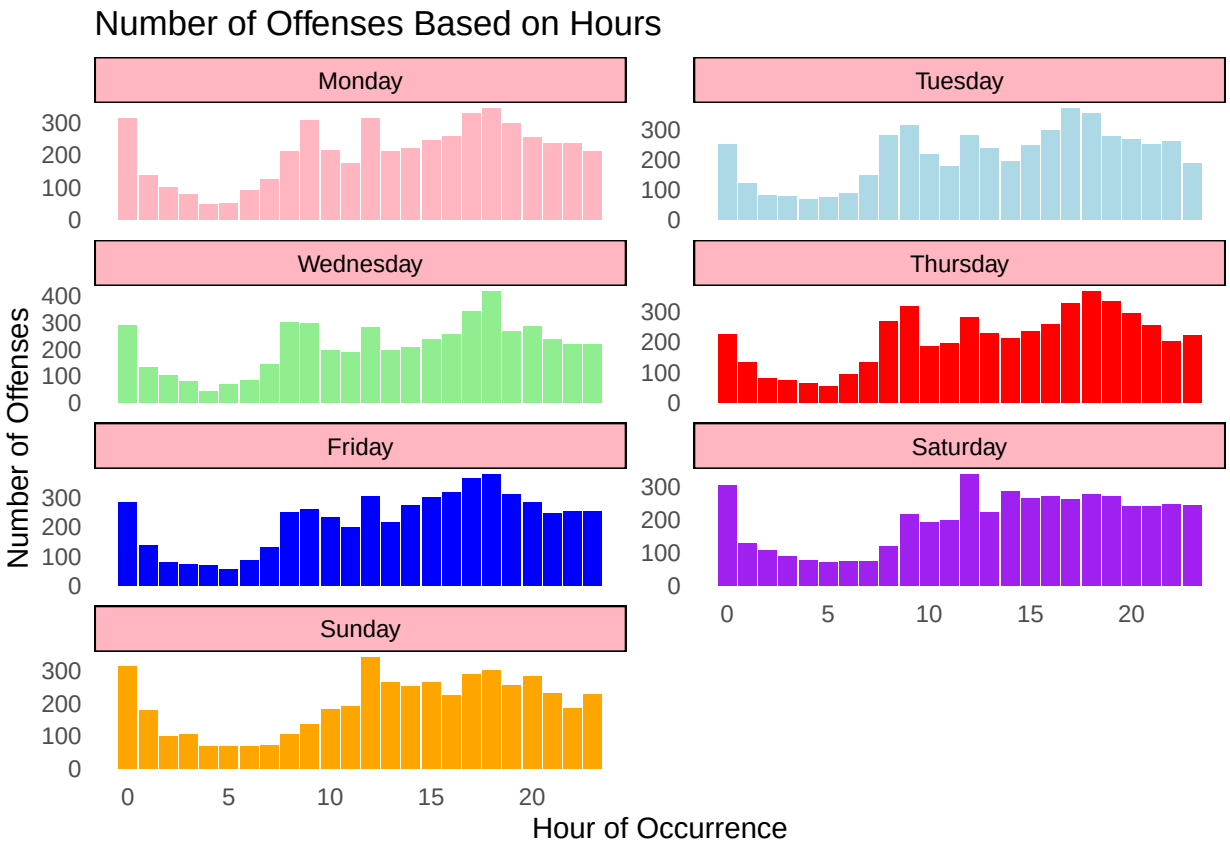
Abstract

ADD ABSTRACT

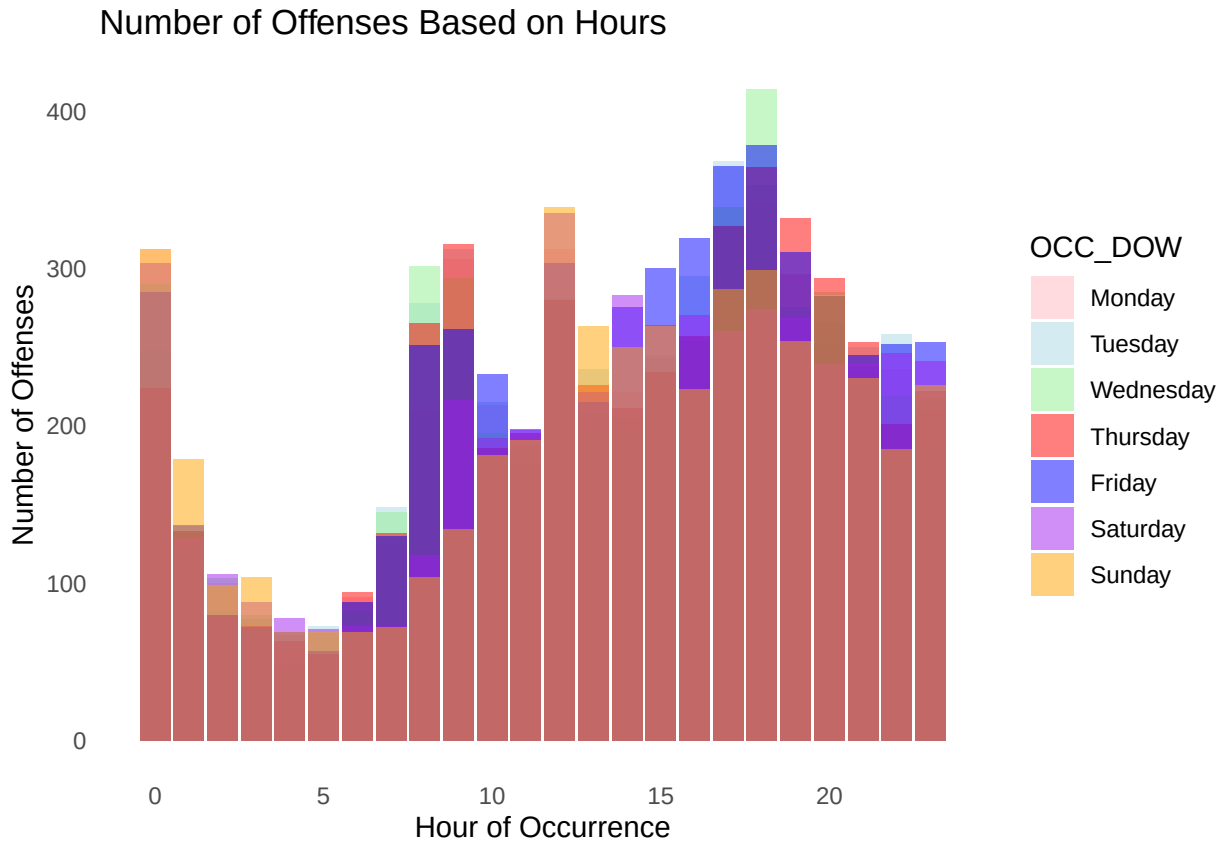
add-up to plot1



KEEP Plot 1 version 1



plot 1 version2

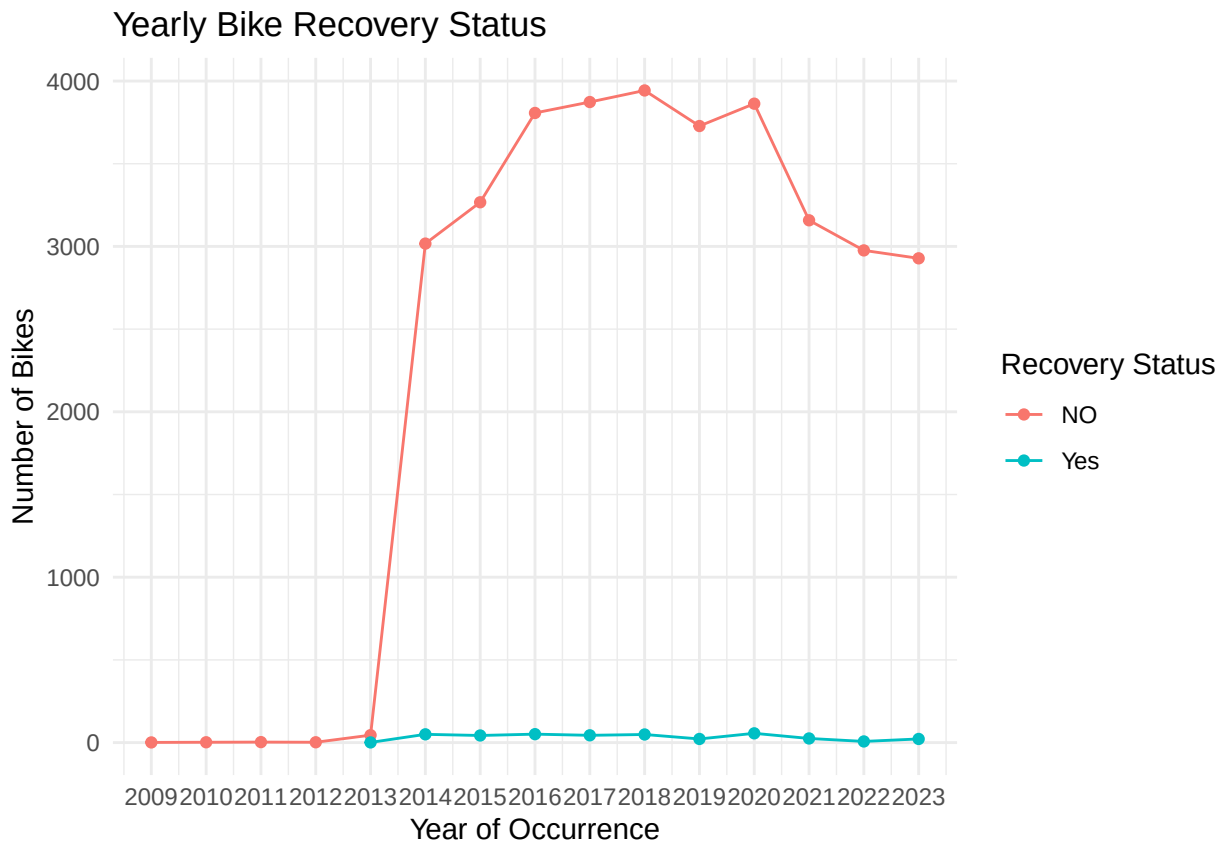


plot 2

```
# data cleaning for plot 2
#| include: false
#| warning: false
#| message: false

bike_raw$y_n <- ifelse(bike_raw$STATUS =='RECOVERED','Yes','NO')
```

```
#| echo: false
#| warning: false
#| message: false


# Assuming bike_raw has a column 'OCC_YEAR' for the year of occurrence
# and a column 'y_n' indicating if the bike was recovered ('Yes' or 'No')

# Summarize data
yearly_counts <- bike_raw %>% filter(OCC_YEAR>2000) %>%
  group_by(OCC_YEAR, y_n) %>%
  summarise(count = n(), .groups = 'drop')
```

```r
# Create the time series plot
ggplot(yearly_counts, aes(x = OCC_YEAR, y = count, color = y_n)) +
  geom_line() +  # Line plot
  geom_point() + # Add points to the lines
  labs(title = "Yearly Bike Recovery Status",
       x = "Year of Occurrence",
       y = "Number of Bikes",
       color = "Recovery Status") +
  theme_minimal() +
  scale_x_continuous(breaks = unique(yearly_counts$OCC_YEAR))  # Ensure all years are included as breaks
```



plot 3: the map!!

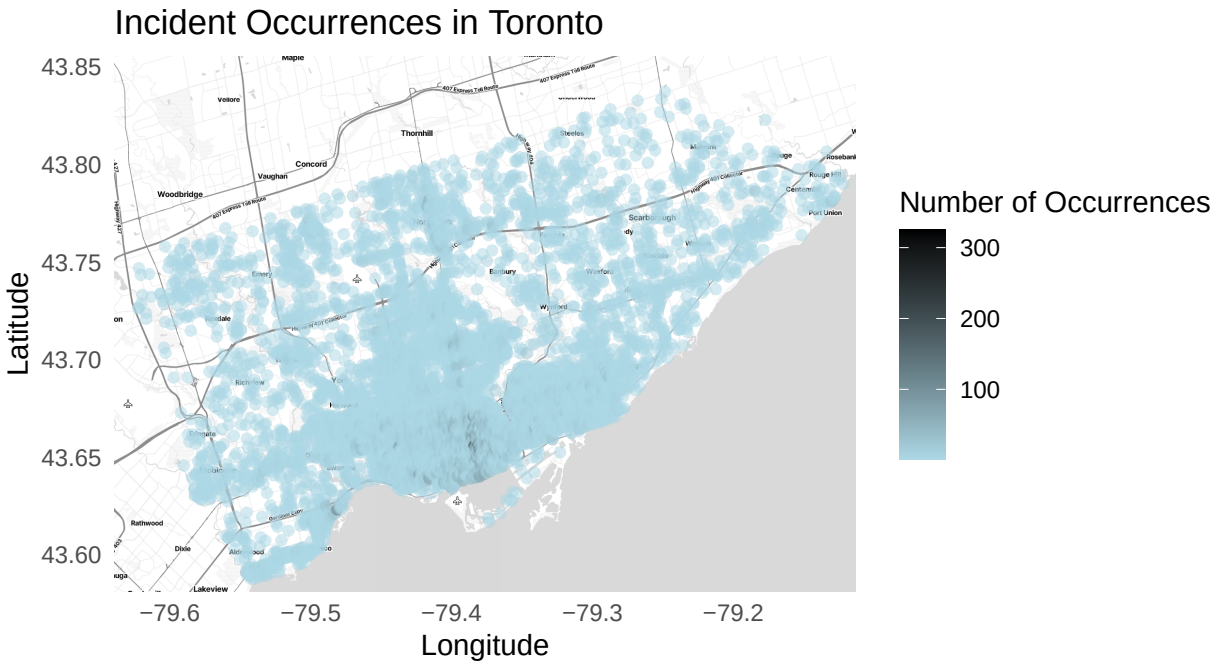
```r
register_stadiamaps(key = "ddfcfc93-13e6-497f-8336-c5fdd043f311")

toronto_map <- get_stadiamap(bbox = c(left = -79.639219, bottom = 43.581024, right = -79.113219, top = 4
                             maptype = "stamen_toner_lite",
                             zoom = 12)
```

```
## i © Stadia Maps © Stamen Design © OpenMapTiles © OpenStreetMap contributors.
```

```r
# data cleaning
bike_aggregated <- bike_raw %>%
  group_by(LONG_WGS84, LAT_WGS84) %>%
  summarise(count = n(), .groups = 'drop')
```

```r
ggmap(toronto_map) +
  geom_point(data = bike_aggregated, aes(x = LONG_WGS84, y = LAT_WGS84, color = count), alpha = 0.5) +
  scale_color_gradient(low = "lightblue", high = "black") +  # Adjust color gradient as needed
  labs(title = "Incident Occurrences in Toronto",
       x = "Longitude",
       y = "Latitude",
       color = "Number of Occurrences") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



KEEP

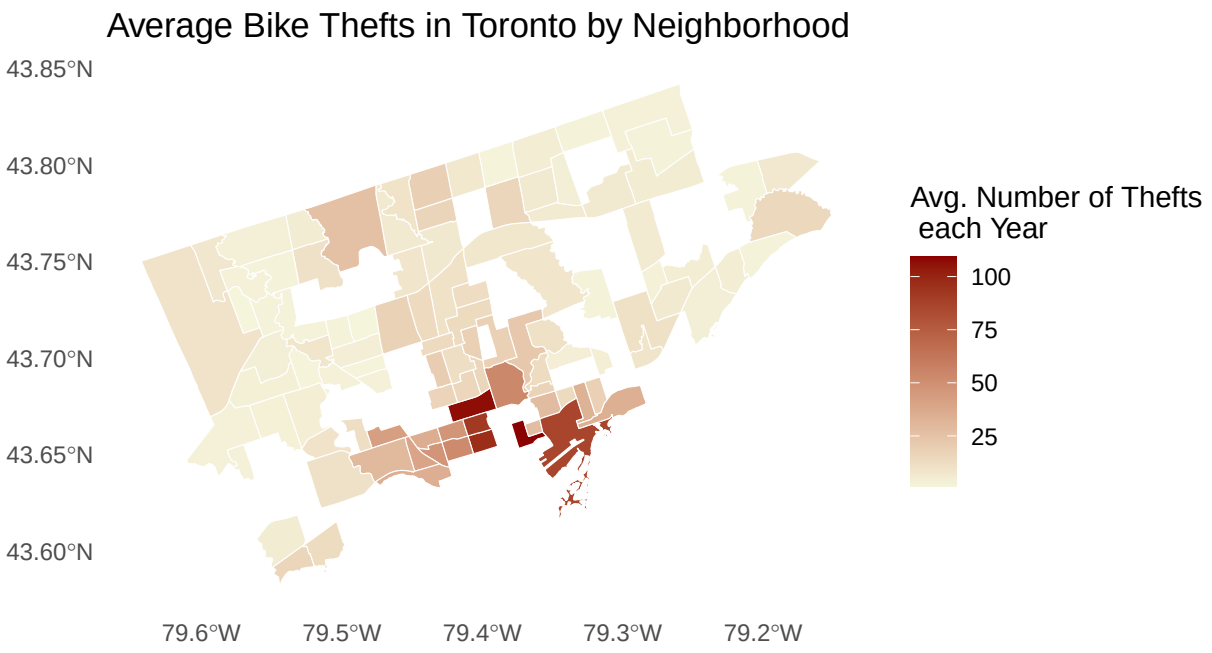
```r
# Load necessary libraries
library(ggplot2)
library(sf)
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```r
library(dplyr)

# Step 1: Load the geographic data
toronto_neighborhoods <- st_read(file.path("input/toneighshape/Neighbourhoods v2_region.shp"))
```

```
## Reading layer ‘Neighbourhoods v2_region’ from data source
##   ‘/Users/xiuzh/Desktop/313/input/toneighshape/Neighbourhoods v2_region.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 140 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 609589.4 ymin: 4826145 xmax: 651617.9 ymax: 4857224
## Projected CRS: unnamed
```

```r
# Step 2: Aggregate bike theft data to calculate the average occurrence over the years
# Replace `neighborhood_name` and `year` with the actual column names from your bike data
# Assume you have a column `YEAR` for the year of each theft
bike_thefts_avg <- bike_raw %>%
  group_by(NEIGHBOURHOOD_158, OCC_YEAR) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(NEIGHBOURHOOD_158) %>%
  summarise(avg_count = mean(count), .groups = 'drop')

# Step 3: Merge the geographic and theft data
# Replace `NAME` with the neighborhood name column from the shapefile
toronto_thefts_geo <- merge(toronto_neighborhoods, bike_thefts_avg,
                            by.x = "NAME", by.y = "NEIGHBOURHOOD_158")

# Step 4: Create the map showing average occurrences
ggplot(toronto_thefts_geo) +
  geom_sf(aes(fill = avg_count), color = "white", size = 0.2) +
  scale_fill_gradient(low = "beige", high = "darkred") +  # Adjust color scale as needed
  labs(title = "Average Bike Thefts in Toronto by Neighborhood",
       fill = "Avg. Number of Thefts \n each Year") +
  theme_minimal() +
  theme(legend.position = "right", panel.grid.major = element_blank(), panel.grid.minor = element_blank
```



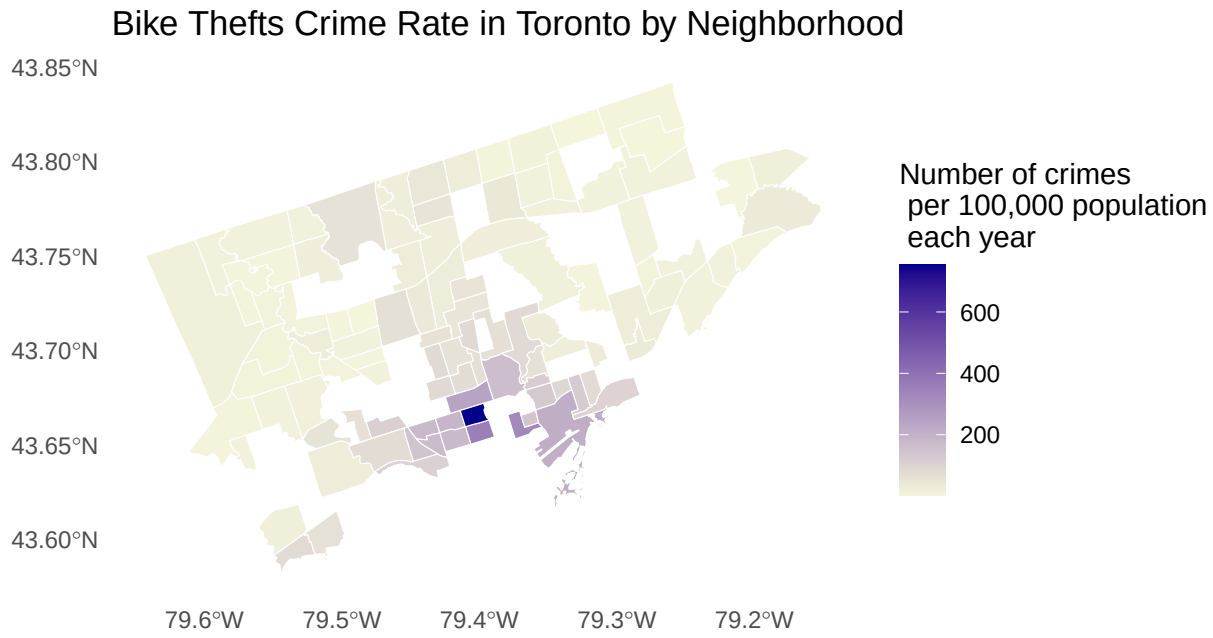

```
# Step 2: Aggregate bike theft data
# Replace `neighborhood_name` with the actual column name from your bike data
neighbourhood_data <- read_csv("input/Neighbourhood_Crime_Rates_Open_Data.csv")
```

```
## Rows: 158 Columns: 186
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): AREA_NAME
## dbl (185): OBJECTID_1, HOOD_ID, ASSAULT_2014, ASSAULT_2015, ASSAULT_2016, AS...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
average_bike_thefts <- neighbourhood_data %>%
  # Select the HOOD_ID and BIKETHEFT columns
  select(AREA_NAME, starts_with("BIKETHEFT_")) %>%
  # Gather the year columns into key-value pairs
  gather(key = "year", value = "bike_thefts", -AREA_NAME) %>%
  # Extract the year from the column names
  mutate(year = as.numeric(sub("BIKETHEFT_", "", year))) %>%
  # Calculate the average thefts per neighborhood
  group_by(AREA_NAME) %>%
  summarise(average_bike_thefts = mean(bike_thefts, na.rm = TRUE)) %>%
  ungroup()
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `year = as.numeric(sub("BIKETHEFT_", "", year))`.
## Caused by warning:
## ! NAs introduced by coercion
```

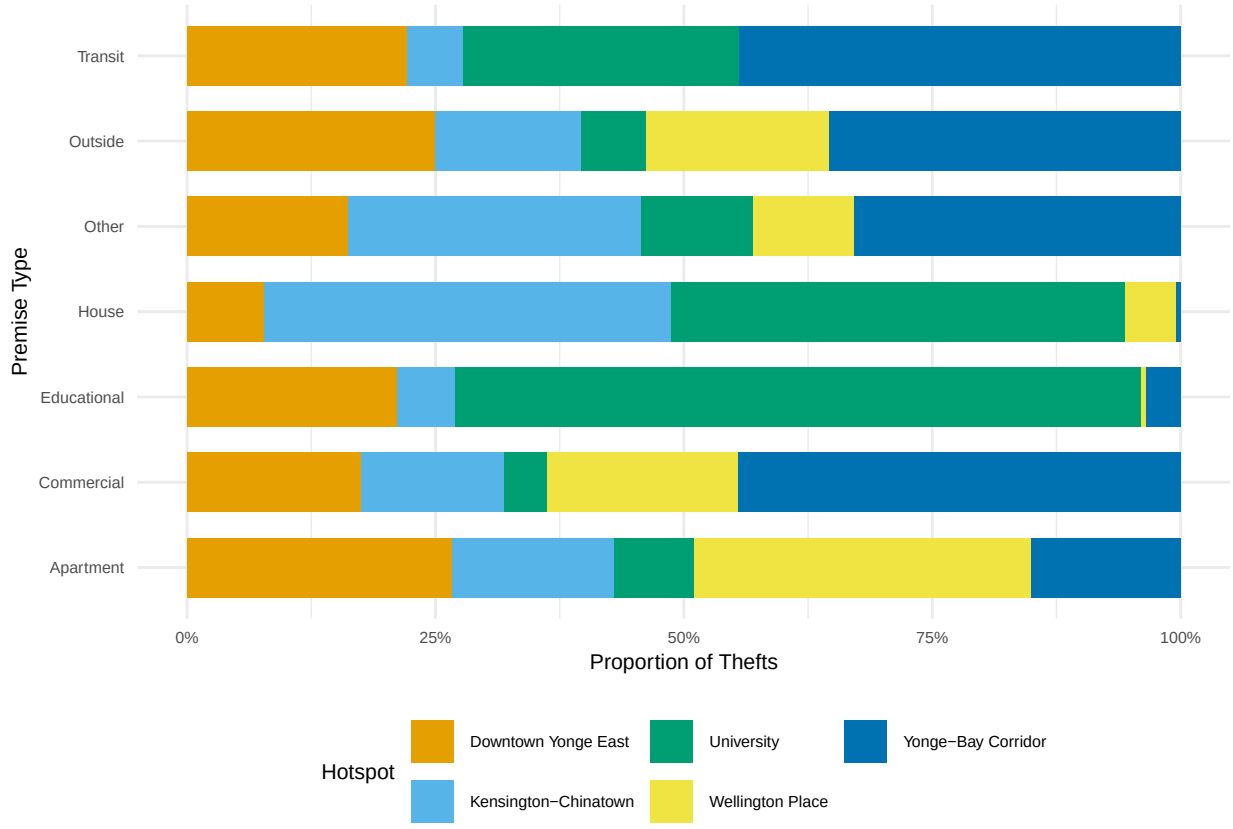
```r
# Step 3: Merge the geographic and theft data
# Replace `NAME` with the neighborhood name column from the shapefile
toronto_thefts_rate <- merge(toronto_neighborhoods, average_bike_thefts,
                             by.x = "NAME", by.y = "AREA_NAME")

# Step 4: Create the map
ggplot(toronto_thefts_rate) +
  geom_sf(aes(fill = average_bike_thefts), color = "white", size = 0.2) +
  scale_fill_gradient(low = "beige", high = "darkblue") +  # Adjust color scale as needed
  labs(title = "Bike Thefts Crime Rate in Toronto by Neighborhood",
       fill = "Number of crimes \n per 100,000 population \n each year") +
  theme_minimal() +
  theme(legend.position = "right", panel.grid.major = element_blank(), panel.grid.minor = element_blank
```



KEEP prop plot

```r
# Assuming average_bike_thefts is your precomputed dataframe with average thefts by neighborhood
# and it contains a 'neighborhood' column and an 'avg_theft' column.
```

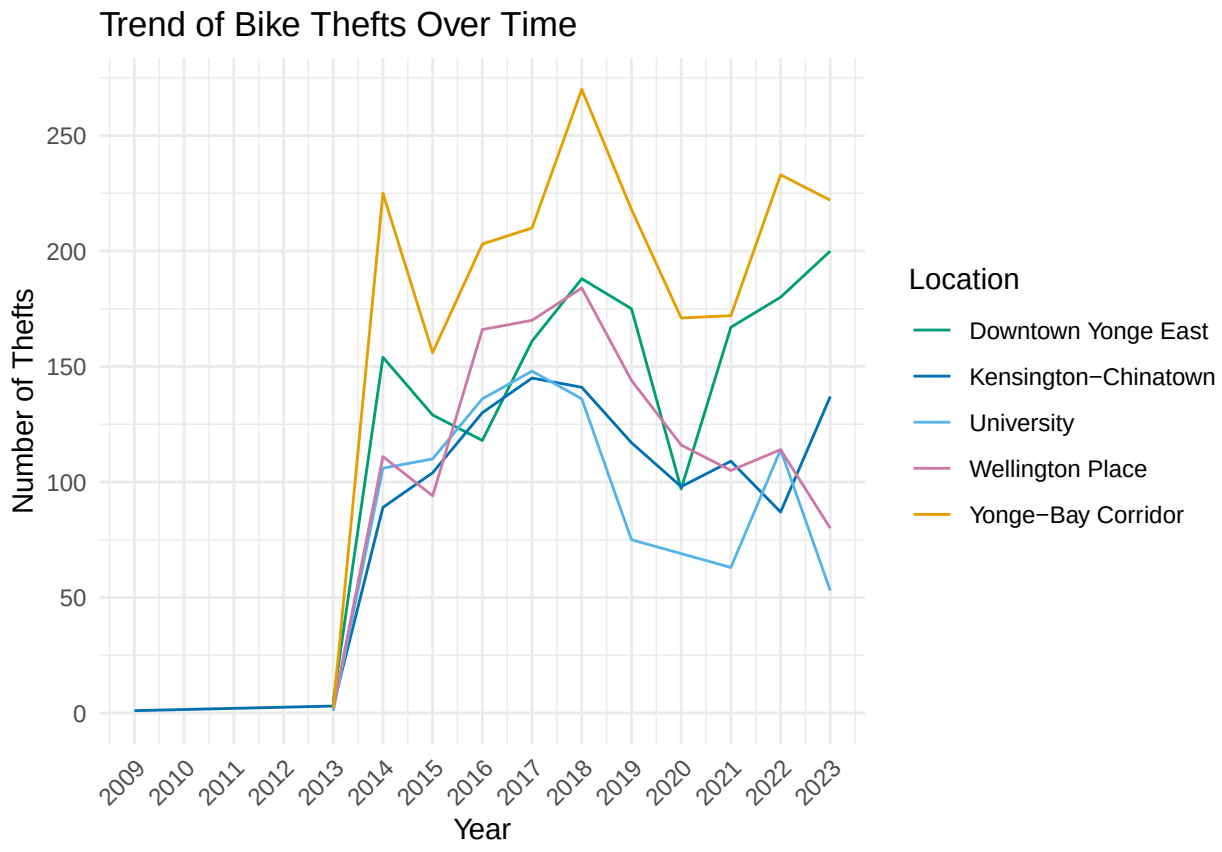
```r
# First, identify the top 10 hotspots
top_neighborhoods <- average_bike_thefts %>%
  arrange(desc(average_bike_thefts)) %>%
  top_n(5, average_bike_thefts)

# Now, filter your main dataset to only include records from these hotspots
# and calculate the proportion of thefts by premises type within these hotspots
data_premise <- bike_raw %>%
  filter(NEIGHBOURHOOD_158 %in% top_neighborhoods$AREA_NAME) %>% # Replace 'NEIGHBORHOOD' with the actu
  group_by(PREMISES_TYPE) %>% # Replace 'PREMISE_TYPE' with the actual column name
  count(HOTSPOT = NEIGHBOURHOOD_158) %>% # Count occurrences by premise type within each hotspot
  ungroup() %>%
  group_by(PREMISES_TYPE) %>%
  mutate(total = sum(n)) %>%
  ungroup() %>%
  mutate(prop = n / total)

# ... [earlier code for preparing the data] ...

unique_hotspots <- unique(data_premise$HOTSPOT)

# Create a named vector of colors for each hotspot
my_colors <- setNames(c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"), u

# Now, create the plot using this color scheme
ggplot(data_premise, aes(x = PREMISES_TYPE, y = prop, fill = HOTSPOT)) +
  geom_bar(stat = "identity", position = position_fill(reverse = TRUE), width = 0.7) +
  # geom_text(aes(label = scales::percent(prop, accuracy = 1)),
  #           position = position_fill(vjust = 0.5), size = 2.5, color = "white") +
  coord_flip() +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  scale_fill_manual(values = my_colors) +
  labs(x = "Premise Type", y = "Proportion of Thefts", fill = "Hotspot") +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.direction = "horizontal",
        legend.box = "vertical",
        text = element_text(size = 8),
        axis.text.x = element_text(size = 6),
        legend.text = element_text(size = 6)) +
  guides(fill = guide_legend(nrow = 2))
```



KEEP trend

```
library(ggplot2)
library(dplyr)

# Then, prepare the data by counting thefts over time for these locations
theft_trends <- bike_raw %>%
  filter(NEIGHBOURHOOD_158 %in% top_neighborhoods$AREA_NAME) %>%
  group_by(NEIGHBOURHOOD_158, OCC_YEAR) %>%
  summarise(Thefts = n(), .groups = 'drop') %>%
  arrange(NEIGHBOURHOOD_158, OCC_YEAR)

custom_colors <- c("Yonge-Bay Corridor" = "#E69F00", "University" = "#56B4E9",
                   "Downtown Yonge East" = "#009E73", "Wellington Place" = "#CC79A7",
                   "Kensington-Chinatown" = "#0072B2")

# Plotting the theft trends with custom colors
ggplot(theft_trends, aes(x = OCC_YEAR, y = Thefts, group = NEIGHBOURHOOD_158, color = NEIGHBOURHOOD_158
  geom_line() +
  scale_color_manual(values = custom_colors) +
  scale_x_continuous(breaks = seq(min(theft_trends$OCC_YEAR), max(theft_trends$OCC_YEAR), by = 1)) +
  scale_y_continuous(limits = c(floor(min(theft_trends$Thefts)/10)*10, ceiling(max(theft_trends$Thefts)
                     breaks = seq(floor(min(theft_trends$Thefts)/10)*10, ceiling(max(theft_trends$Theft
  theme_minimal() +
```

```
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  labs(title = "Trend of Bike Thefts Over Time",
       x = "Year",
       y = "Number of Thefts",
       color = "Location")
```




```
 library(dplyr)
library(tidyr)
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

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

```
# Assuming bike_raw is already loaded into the workspace
# And you've already created the 'Slot' column with case_when as previously discussed.

# Process the dataset to categorize hours into slots and summarize the count
thefts_by_hour_day <- bike_raw %>%
  mutate(OCC_HOUR = as.numeric(as.character(OCC_HOUR)), # Ensure OCC_HOUR is numeric
```

```
        Slot = case_when(
          OCC_HOUR >= 0 & OCC_HOUR < 6   ~ 'Night',       # From 12 AM to 5:59 AM
          OCC_HOUR >= 6 & OCC_HOUR < 12  ~ 'Morning',     # From 6 AM to 11:59 AM
          OCC_HOUR >= 12 & OCC_HOUR < 18 ~ 'Afternoon',   # From 12 PM to 5:59 PM
          OCC_HOUR >= 18 & OCC_HOUR <= 23 ~ 'Evening',     # From 6 PM to 11:59 PM
          TRUE ~ NA_character_ # Should not happen, but including for completeness
        )) %>%
  group_by(OCC_DOW, Slot) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  pivot_wider(names_from = Slot, values_from = Count, values_fill = list(Count = 0)) %>%
  select(OCC_DOW, Night, Morning, Afternoon, Evening) # Reorder the columns

# Add totals for each day (row) and for each hour (column)
thefts_by_hour_day <- thefts_by_hour_day %>%
  rowwise() %>%
  mutate(Total = sum(c_across(c(Night, Morning, Afternoon, Evening)), na.rm = TRUE))

# Convert OCC_DOW to a character to avoid factor level issues when adding "Total"
thefts_by_hour_day$OCC_DOW <- as.character(thefts_by_hour_day$OCC_DOW)

# Arrange by day of the week
ordered_days <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
thefts_by_hour_day$OCC_DOW <- factor(thefts_by_hour_day$OCC_DOW, levels = ordered_days)
thefts_by_hour_day <- arrange(thefts_by_hour_day, OCC_DOW) %>% select(-Total)

# Create the table with knitr
kable(thefts_by_hour_day, caption = "Bicycle Thefts by Time Slot and Day of Week", align = 'c') %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Table 1: Bicycle Thefts by Time Slot and Day of Week

OCC_DOW	Night	Morning	Afternoon	Evening
Sunday	832	751	1625	1477
Monday	724	1119	1568	1570
Tuesday	668	1216	1617	1590
Wednesday	715	1204	1511	1640
Thursday	628	1187	1535	1666
Friday	700	1161	1777	1720
Saturday	775	869	1633	1507

plot 5 (proportion with %)

```
# data cleaning

bike_p <- bike_raw %>% mutate(cost_level = case_when(
  BIKE_COST <= 500 ~ "0-500",
  (BIKE_COST > 500 & BIKE_COST <= 1000) ~ "501-1000",
  (BIKE_COST > 1001 & BIKE_COST <= 1500) ~ "1001-1500",
  BIKE_COST > 1501 ~ "1500+"
```

```
))
bike_p$cost_level <- factor(bike_p$cost_level, levels = c("0-500", "501-1000", "10001-1500", "1500+"))
```