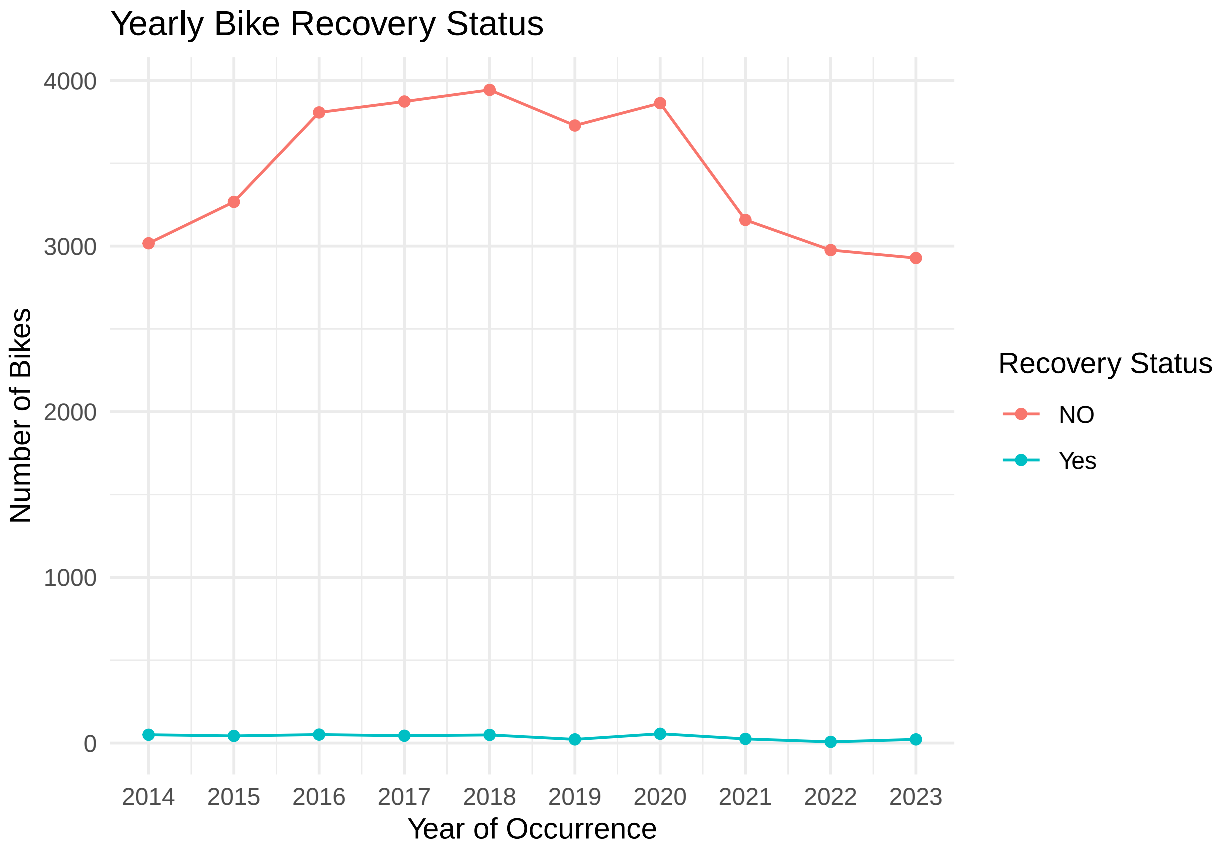
CHANGE TITLE

Ruibo Sun and Nixi Huang

2024-03-15 12:02:29

Abstract

ADD ABSTRACT



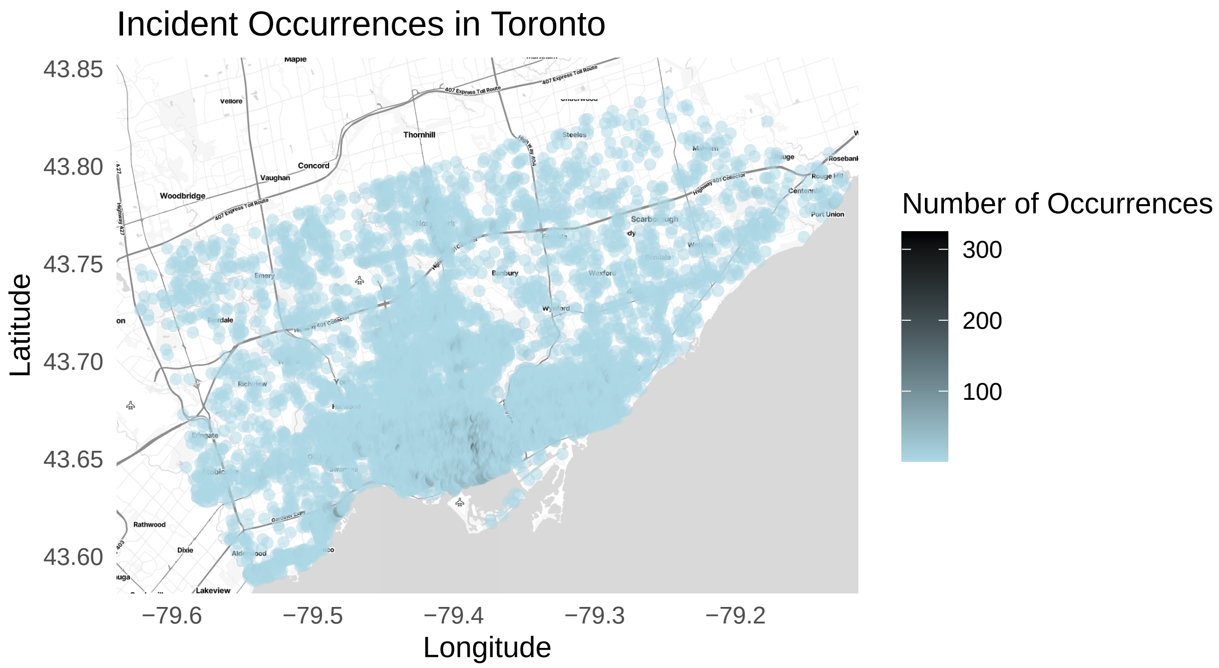
```r
register_stadiamaps(key = "ddfcfc93-13e6-497f-8336-c5fdd043f311")

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()‘).
```

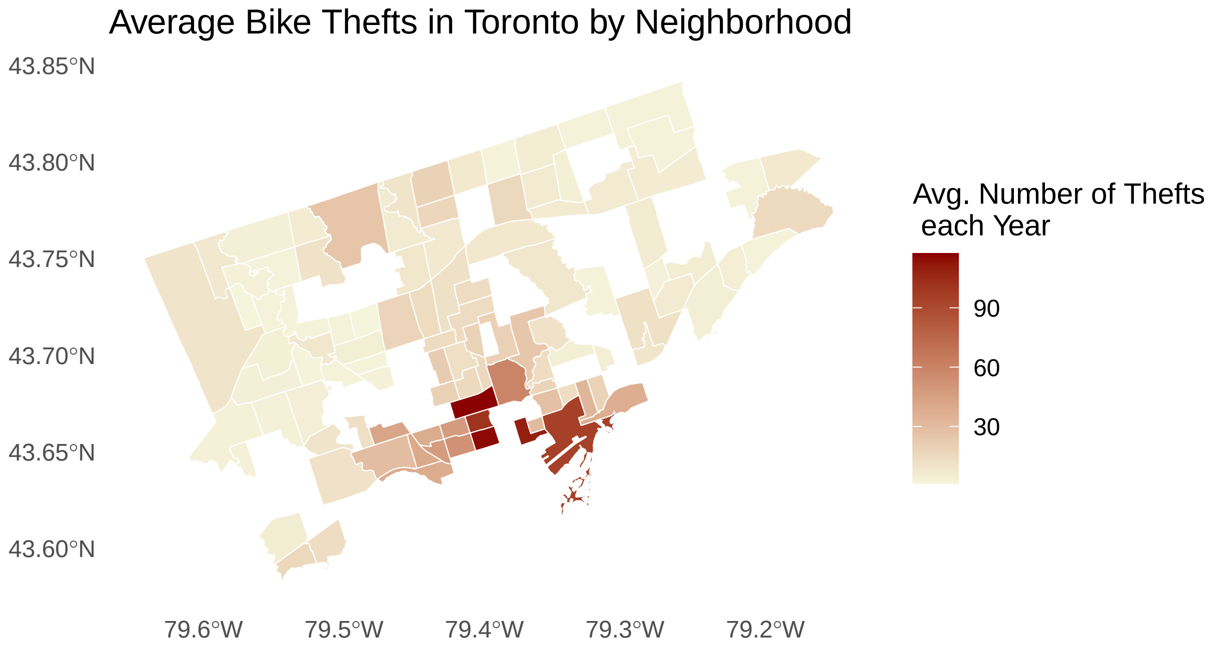


KEEP

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



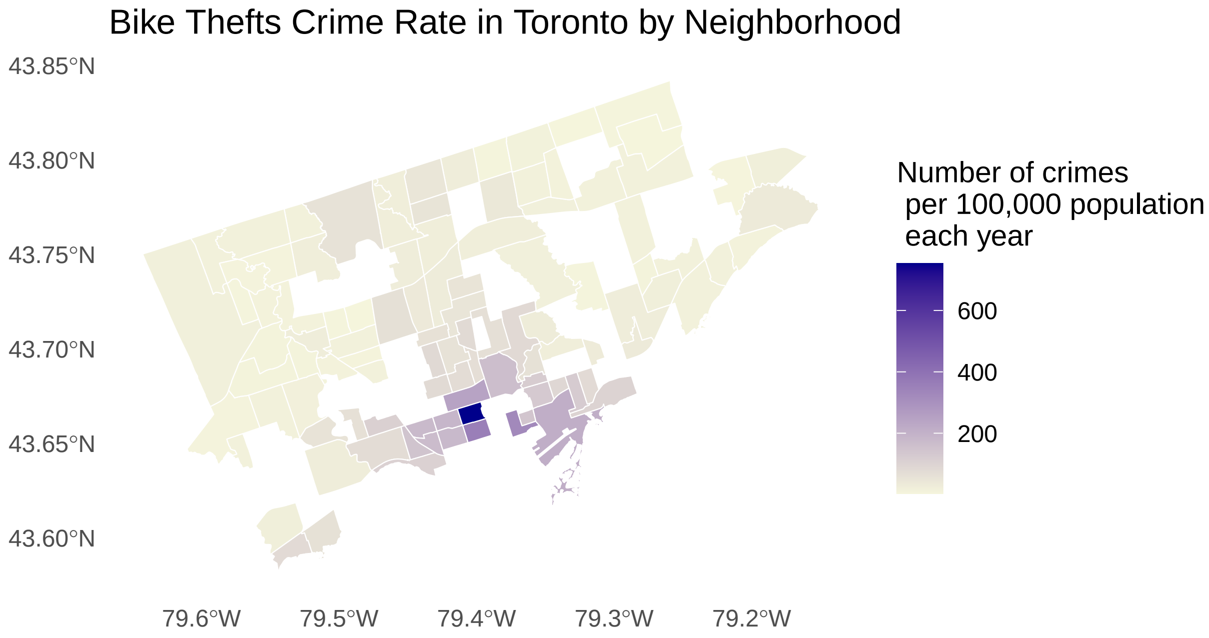
```
# Step 2: Aggregate bike theft data
# Replace `neighborhood_name` with the actual column name from your bike data
neighbourhood_data <- read_csv("input/Neighbourhood_Crime_Rates_Open_Data.csv")
```

```
## 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()
```

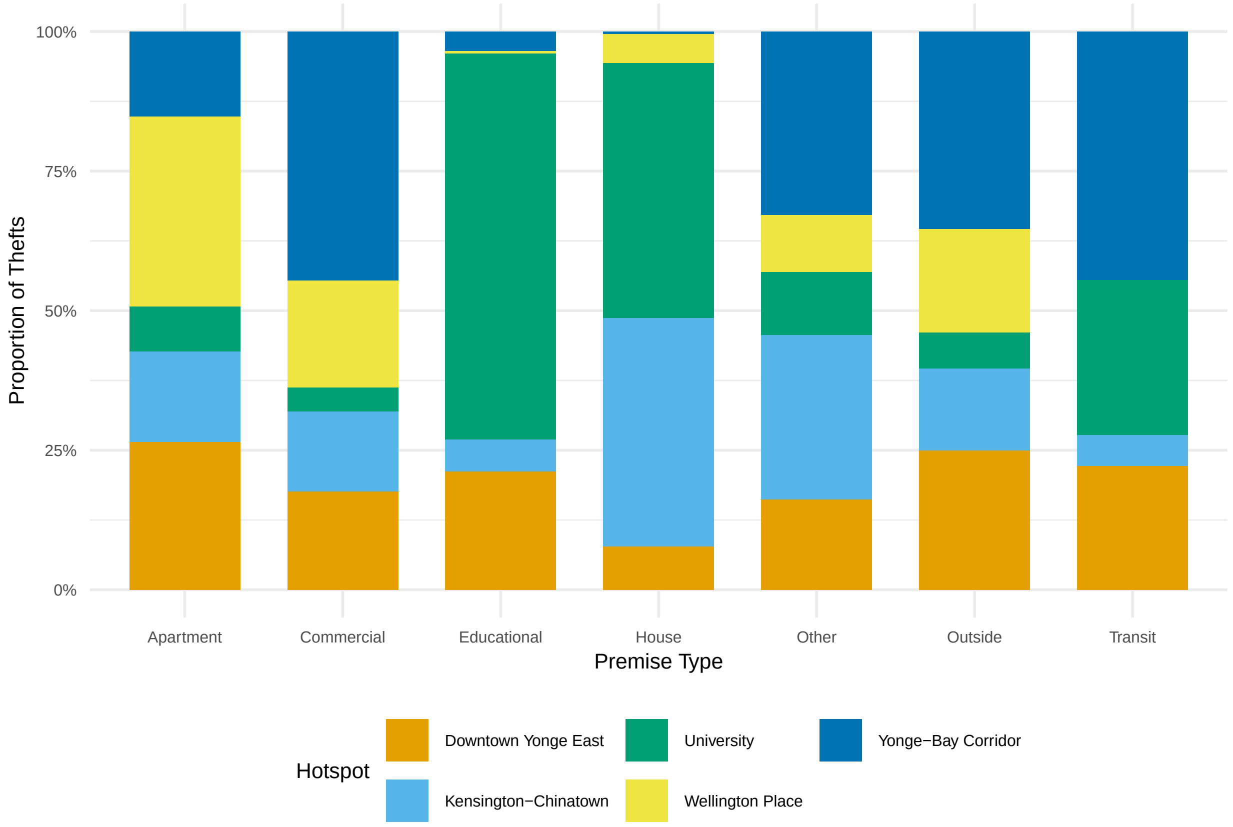
```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘year = as.numeric(sub("BIKETHEFT_", "", year))‘.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# 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
```



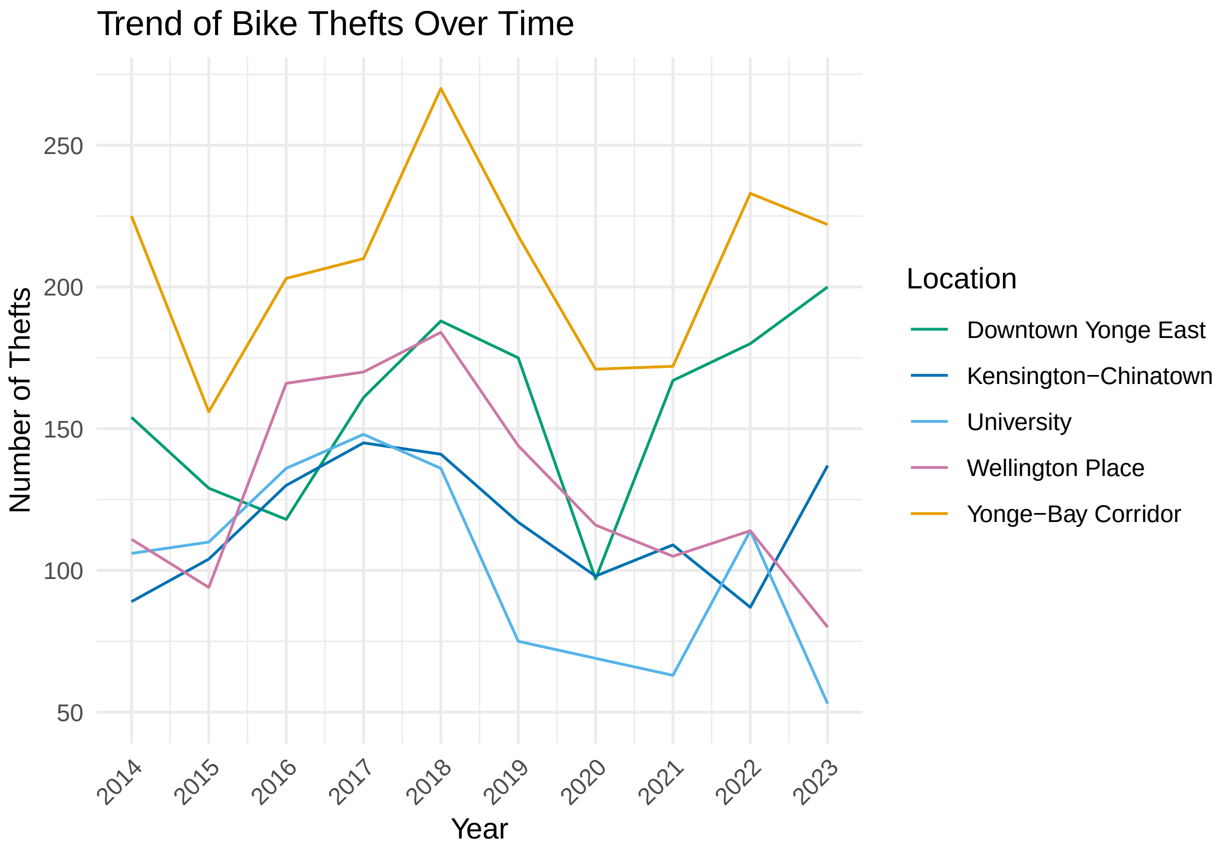
KEEP prop plot

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

# 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") +
  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

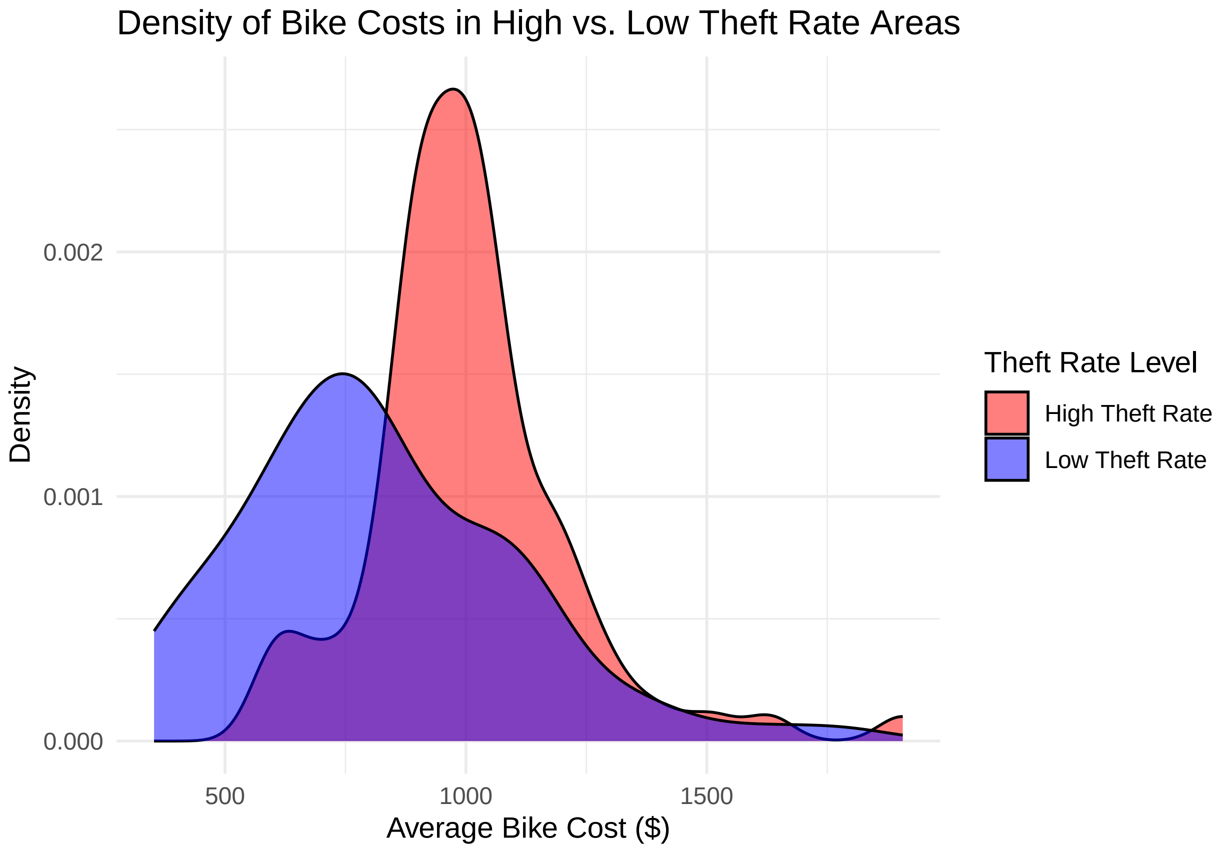
```
library(ggplot2)
library(dplyr)

# 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")
```



```
# data cleaning

bike_p <- bike_raw %>%
  mutate(cost_level = case_when(
    BIKE_COST >= 1000 ~ "Expensive",
    BIKE_COST < 1000 ~ "Affordable"
  ))

bike_p$cost_level <- factor(bike_p$cost_level, levels = c("Affordable", "Expensive"))
```

THE PLOT of COST

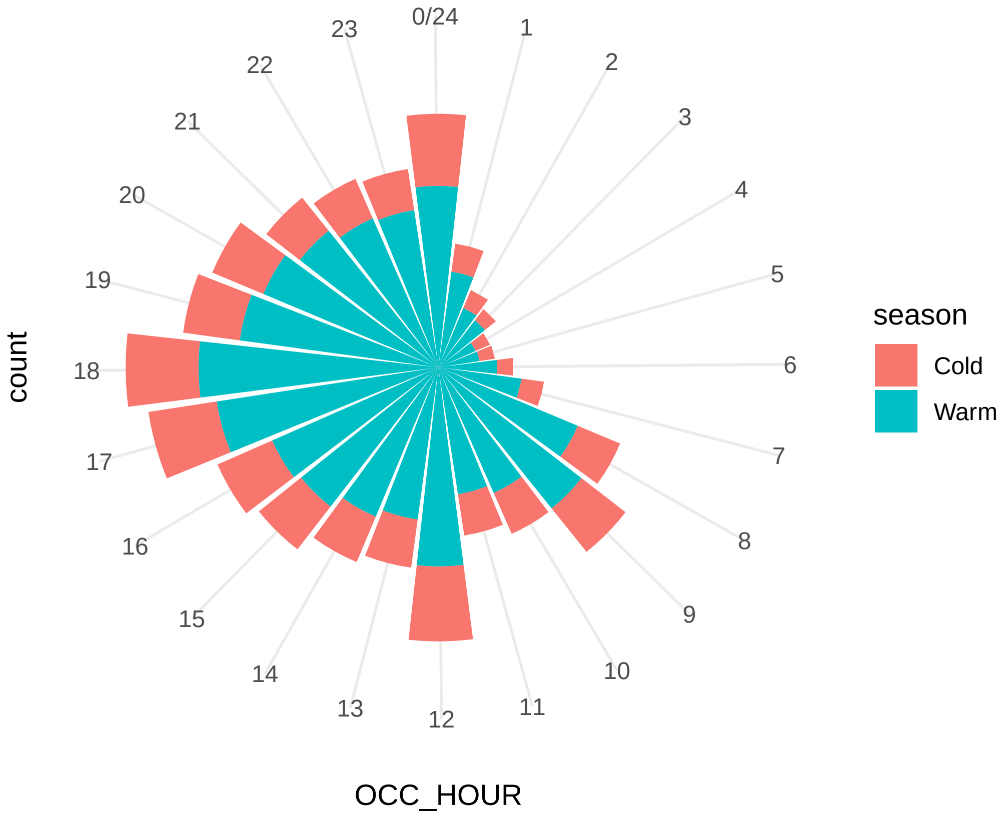
```
# First, classify areas based on theft rate threshold
threshold <- median(average_bike_thefts$average_bike_thefts)

average_bike_thefts <- average_bike_thefts %>%
  mutate(Theft_Level = if_else(average_bike_thefts >= threshold, "High Theft Rate", "Low Theft Rate"))


average_bike_cost <- bike_raw %>%
  group_by(NEIGHBOURHOOD_158) %>%  # Replace with the correct neighborhood column name
  summarise(Average_Bike_Cost = mean(BIKE_COST, na.rm = TRUE))


combined_data <- average_bike_thefts %>%
  left_join(average_bike_cost, by = 'NEIGHBOURHOOD_158')

# Now, plot the density of bike costs for high-theft-rate areas versus low-theft-rate areas
ggplot(combined_data, aes(x = Average_Bike_Cost, fill = Theft_Level)) +
  geom_density(alpha = 0.5) +  # Set transparency to see overlapping areas
  scale_fill_manual(values = c("High Theft Rate" = "red", "Low Theft Rate" = "blue")) +
  labs(title = "Density of Bike Costs in High vs. Low Theft Rate Areas",
       x = "Average Bike Cost ($)",
       y = "Density",
       fill = "Theft Rate Level") +
  theme_minimal() +
  theme(legend.position = "right")
```



after the cost, hotpsot vs hours of a day

```
hour_plot <- left_join(bike_raw, average_bike_thefts, by = 'NEIGHBOURHOOD_158')

hour_plot$OCC_HOUR <- as.factor(hour_plot$OCC_HOUR)
if("0" %in% levels(hour_plot$OCC_HOUR)) {
  levels(hour_plot$OCC_HOUR)[levels(hour_plot$OCC_HOUR) == "0"] <- "0/24"
}
```

```
# Use scale_x_discrete() to include all factor levels
hour_plot <- hour_plot %>%
  mutate(season = if_else(OCC_MONTH %in% c("December", "January", "February", "March", "April", "Novembe

degrees_to_rotate <- 8
radians_to_rotate <- degrees_to_rotate * (pi / 180)

ggplot(hour_plot, aes(x = OCC_HOUR, fill=season)) +
  geom_bar() +
  scale_x_discrete(drop = FALSE) + # Include all levels, even if some have no data
  coord_polar(start = 0 - radians_to_rotate)+
  theme_minimal()+
  theme(
    axis.text.y = element_blank(),  # This will remove the radial axis labels
    panel.grid.major.y = element_blank(),  # This will remove the major grid lines (optional)
    panel.grid.minor.y = element_blank()  # This will remove the minor grid lines (optional)
  )
```


0/24
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
count
OCC_HOUR
season
Cold
Warm