

The Beach BIA EDA

Exploratory Data Analysis

The Beaches BIA Project, for Dr. Johanna Carlo.

```
library(ggplot2)
library(tidyverse)
library(lubridate)
library(sf)
library(janitor)
library(skimr)
library(visdat)
```

TTC STREETCAR DATA

```
library(opendatatoronto)
library(dplyr)

# get package
package <- show_package("b68cb71b-44a7-4394-97e2-5d2f41462a5d")
package
```

```
# A tibble: 4 x 11
  title          id    topics civic_issues publisher excerpt dataset_category
  <chr>          <chr> <chr>   <chr>         <chr>    <chr>    <chr>
1 TTC Streetcar De~ b68c~ Trans~ <NA>         Toronto ~ TTC St~ Document
2 TTC Streetcar De~ b68c~ Trans~ <NA>         Toronto ~ TTC St~ Document
3 TTC Streetcar De~ b68c~ Trans~ <NA>         Toronto ~ TTC St~ Document
4 TTC Streetcar De~ b68c~ Trans~ <NA>         Toronto ~ TTC St~ Document
# i 4 more variables: num_resources <int>, formats <chr>, refresh_rate <chr>,
#   last_refreshed <date>
```

```
# get all resources for this package
ttc_resources <- list_package_resources("b68cb71b-44a7-4394-97e2-5d2f41462a5d")

ttc_resources20s <- ttc_resources |> mutate(year = str_extract(name, "202.?"))
delay_ids <- ttc_resources20s |> filter(year %in% c(2022, 2023, 2024)) |> select(id) |> pull

ttc_2025 <- get_resource("cf2fdd12-6f0e-4644-aae7-0bbfc31df031")
ttc_2024 <- get_resource("5f527714-2284-437b-958b-c02b6f21eb9d")
ttc_2023 <- get_resource("472d838d-e41a-4616-a11b-585d26d59777")
ttc_2022 <- get_resource("28547222-35fe-48b6-ac4b-ccc67d286393")
```

Streetcar 501 QUEEN: Stops in the Beaches BIA

From Coxwell Ave to Neville Park Loop

- Queen St East at Neville Park Blvd
- Queen St East at Silver Birch Ave
- Queen St East at Beech Ave
- Queen St East at Glen Manor Dr
- Queen St East at Wineva Ave
- Queen St East at Bellefair Ave
- Queen St East at Elmer Ave
- Queen St East at Woodbine Ave
- Queen St East at Lockwood Rd
- Queen St East at Kingston Rd
- Queen St East at Coxwell Ave

```
standardize_ttc_data <- function(df) {
  df |>
    clean_names() %>% # Converts 'Min Delay' and 'Min.Delay' both to 'min_delay'
    rename(
      # Unify 'station' and 'location' into 'station'
      station = any_of(c("location", "station")),
      # Unify 'code' and 'incident' into 'incident'
      incident = any_of(c("code", "incident")),
      delay = min_delay,
      gap = min_gap
    ) %>%
    mutate(
      date = as_date(date),
      line = as.character(line), # Handle numeric/char differences in 'Line'
      month = floor_date(date, "month")
    )
}
```

```

    ) %>%
    filter(str_detect(as.character(line), "501")) # Focus strictly on the Queen Streetcar
  }

ttc501_22 <- standardize_ttc_data(ttc_2022)
ttc501_23 <- standardize_ttc_data(ttc_2023)
ttc501_24 <- standardize_ttc_data(ttc_2024)
ttc501_25 <- standardize_ttc_data(ttc_2025)

# 1. Define keywords based on your target stops
# These represent the 'Beaches' specific intersections
beaches_keywords <- c("NEVILLE", "SILVER BIRCH", "BEECH", "GLEN MANOR",
                      "WINEVA", "BELLEFAIR", "ELMER", "WOODBINE",
                      "LOCKWOOD", "KINGSTON", "COXWELL")

filter_beaches_stops <- function(df) {
  df_cleaned <- df %>%
    mutate(station_clean = case_when(
      str_detect(station, "NEVILLE") ~ "Neville Park Blvd",
      str_detect(station, "SILVER BIRCH") ~ "Silver Birch Ave",
      str_detect(station, "BEECH") ~ "Beech Ave",
      str_detect(station, "GLEN MANOR") ~ "Glen Manor Dr",
      str_detect(station, "WINEVA") ~ "Wineva Ave",
      str_detect(station, "BELLEFAIR") ~ "Bellefair Ave",
      str_detect(station, "ELMER") ~ "Elmer Ave",
      str_detect(station, "WOODBINE") ~ "Woodbine Ave",
      str_detect(station, "LOCKWOOD") ~ "Lockwood Rd",
      str_detect(station, "KINGSTON") ~ "Kingston Rd",
      str_detect(station, "COXWELL") ~ "Coxwell Ave",
      TRUE ~ "Other/Outside BIA"
    )) %>%
    # Filter strictly for the neighborhood of interest
    filter(station_clean != "Other/Outside BIA") |>
    mutate(delay_final = ifelse(delay > 90, 90, delay))

  return(df_cleaned)
}

beaches_ttc_25 <- filter_beaches_stops(ttc501_25)
beaches_ttc_24 <- filter_beaches_stops(ttc501_24)
beaches_ttc_23 <- filter_beaches_stops(ttc501_23)
beaches_ttc_22 <- filter_beaches_stops(ttc501_22)

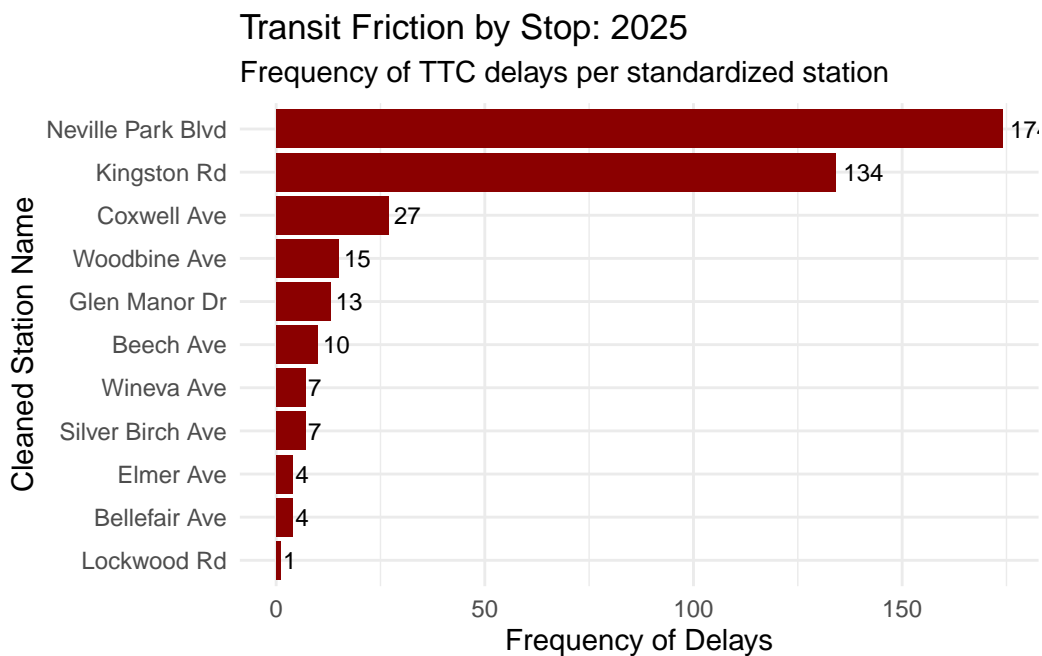
```

```

plot_beaches_friction <- function(df, year_label) {
  df %>%
    count(station_clean, sort = TRUE) %>%
    ggplot(aes(x = reorder(station_clean, n), y = n)) +
    geom_col(fill = "darkred") +
    geom_text(aes(label = n), hjust = -0.2, size = 3) + # Adds data labels for clarity
    coord_flip() +
    labs(
      title = paste("Transit Friction by Stop:", year_label),
      subtitle = "Frequency of TTC delays per standardized station",
      x = "Cleaned Station Name",
      y = "Frequency of Delays"
    ) +
    theme_minimal()
}

plot_beaches_friction(beaches_ttc_25, 2025)

```



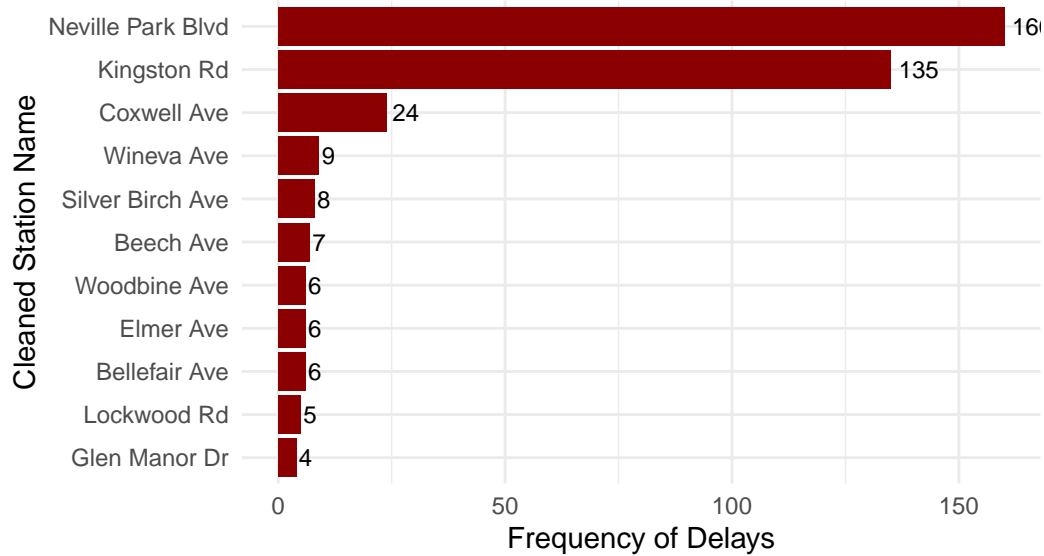
```

plot_beaches_friction(beaches_ttc_24, 2024)

```

Transit Friction by Stop: 2024

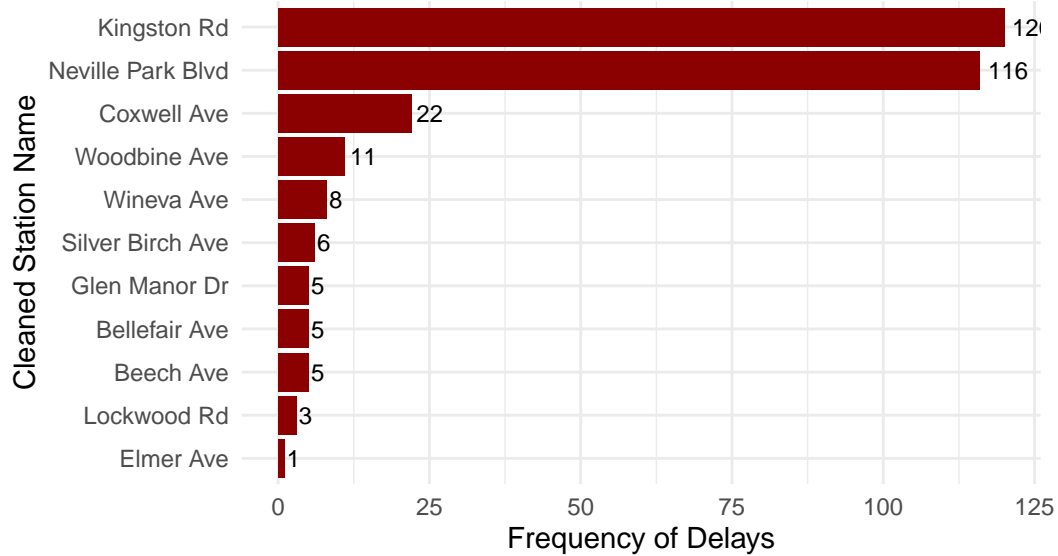
Frequency of TTC delays per standardized station



```
plot_beaches_friction(beaches_ttc_23, 2023)
```

Transit Friction by Stop: 2023

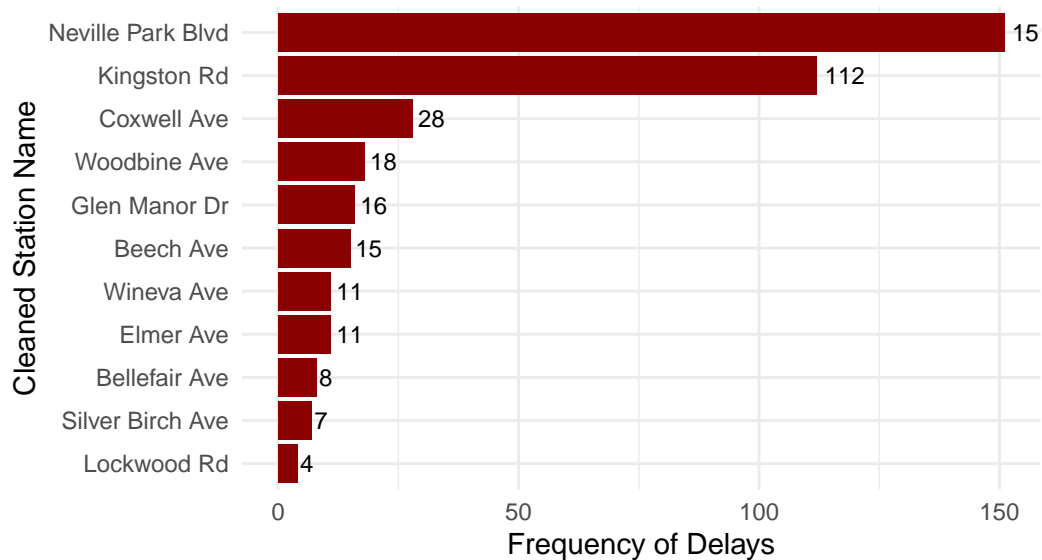
Frequency of TTC delays per standardized station



```
plot_beaches_friction(beaches_ttc_22, 2022)
```

Transit Friction by Stop: 2022

Frequency of TTC delays per standardized station



```
summarize_monthly_ttc <- function(df) {
  df %>%
    group_by(month) %>%
    summarise(
      Total_Delays = n(),
      Avg_Min_Delay = mean(delay, na.rm = TRUE),
      .groups = "drop"
    )
}

beaches_ttc_metrics25 <- summarize_monthly_ttc(beaches_ttc_25)
beaches_ttc_metrics24 <- summarize_monthly_ttc(beaches_ttc_24)
beaches_ttc_metrics23 <- summarize_monthly_ttc(beaches_ttc_23)
beaches_ttc_metrics22 <- summarize_monthly_ttc(beaches_ttc_22)

plot_monthly_frequency <- function(monthly_df, year_label) {
  ggplot(monthly_df, aes(x = month, y = Total_Delays)) +
    geom_col(fill = "steelblue") +
    geom_text(aes(label = Total_Delays), vjust = -0.5, size = 3) +
    scale_x_date(date_labels = "%b", date_breaks = "1 month") +
    labs(
      title = paste("Frequency: TTC Delays per Month -", year_label),
      subtitle = "Higher counts may indicate 'busy season' inflow/congestion",
      x = "Month",

```

```

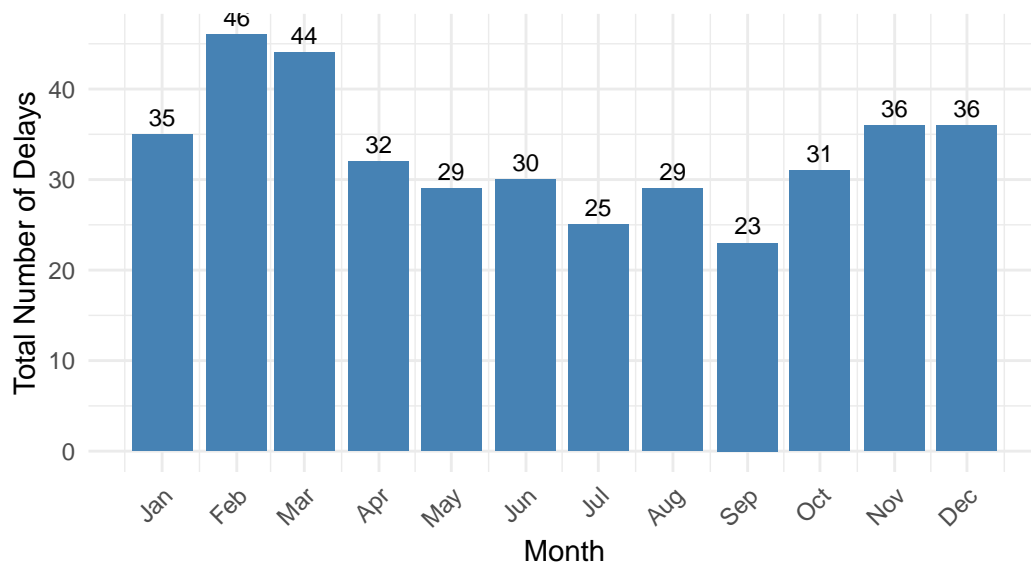
    y = "Total Number of Delays"
  ) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}

plot_monthly_frequency(beaches_ttc_metrics25, 2025)

```

Frequency: TTC Delays per Month – 2025

Higher counts may indicate 'busy season' inflow/congestion



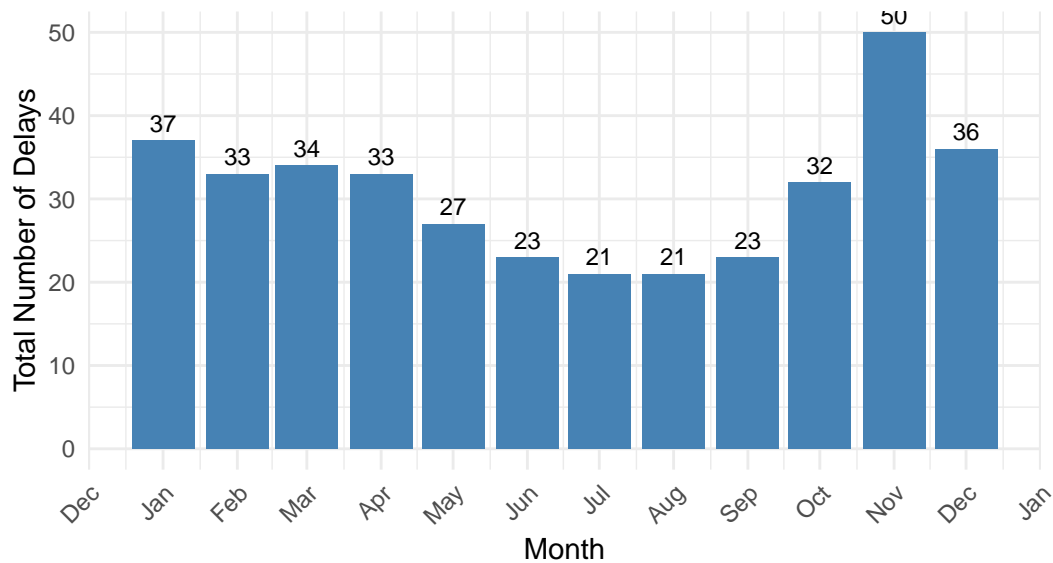
```

plot_monthly_frequency(beaches_ttc_metrics24, 2024)

```

Frequency: TTC Delays per Month – 2024

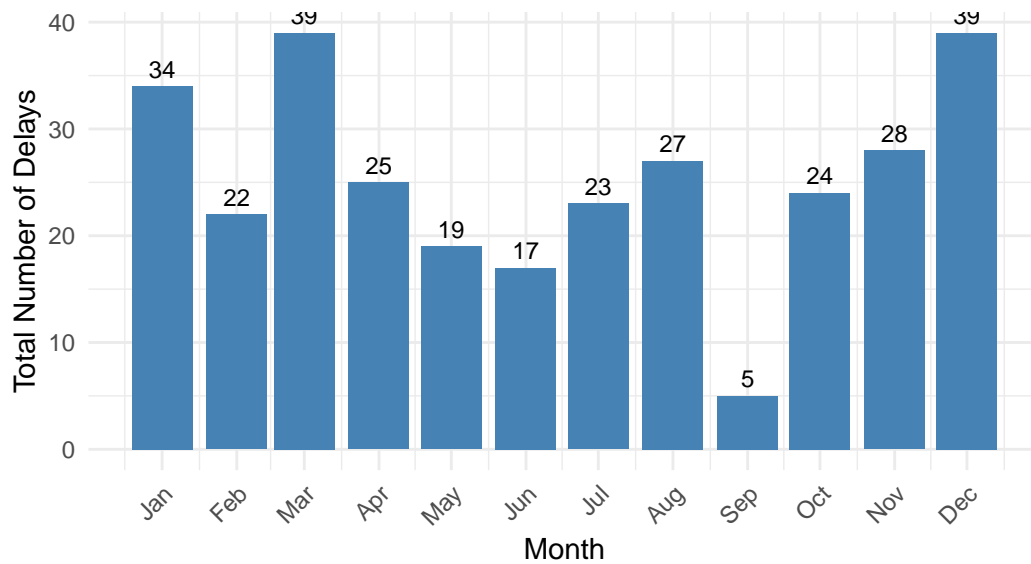
Higher counts may indicate 'busy season' inflow/congestion



```
plot_monthly_frequency(beaches_ttc_metrics23, 2023)
```

Frequency: TTC Delays per Month – 2023

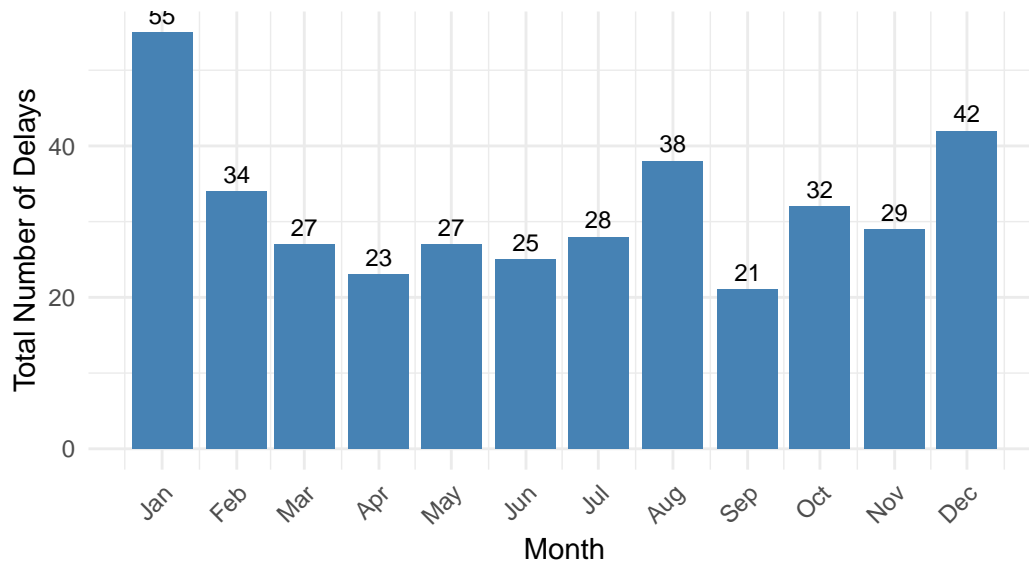
Higher counts may indicate 'busy season' inflow/congestion



```
plot_monthly_frequency(beaches_ttc_metrics22, 2022)
```

Frequency: TTC Delays per Month – 2022

Higher counts may indicate 'busy season' inflow/congestion

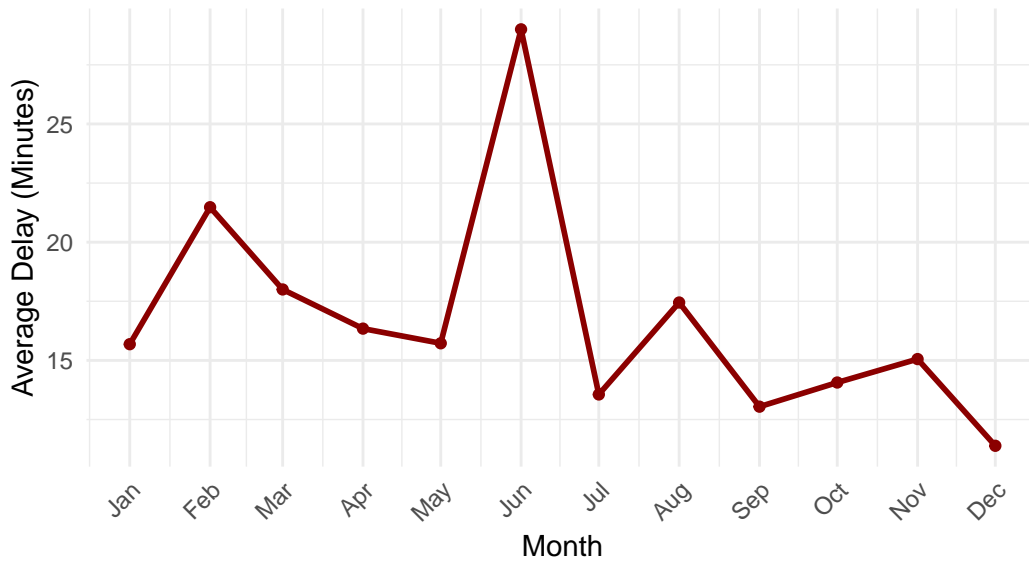


```
plot_monthly_severity <- function(monthly_df, year_label) {  
  ggplot(monthly_df, aes(x = month, y = Avg_Min_Delay)) +  
    geom_line(color = "darkred", linewidth = 1) +  
    geom_point(color = "darkred") +  
    scale_x_date(date_labels = "%b", date_breaks = "1 month") +  
    labs(  
      title = paste("Severity: Average Duration of Delays -", year_label),  
      subtitle = "Measures the 'friction' visitors face when entering the BIA",  
      x = "Month",  
      y = "Average Delay (Minutes)"  
    ) +  
    theme_minimal() +  
    theme(axis.text.x = element_text(angle = 45, hjust = 1))  
}
```

```
plot_monthly_severity(beaches_ttc_metrics25, 2025)
```

Severity: Average Duration of Delays – 2025

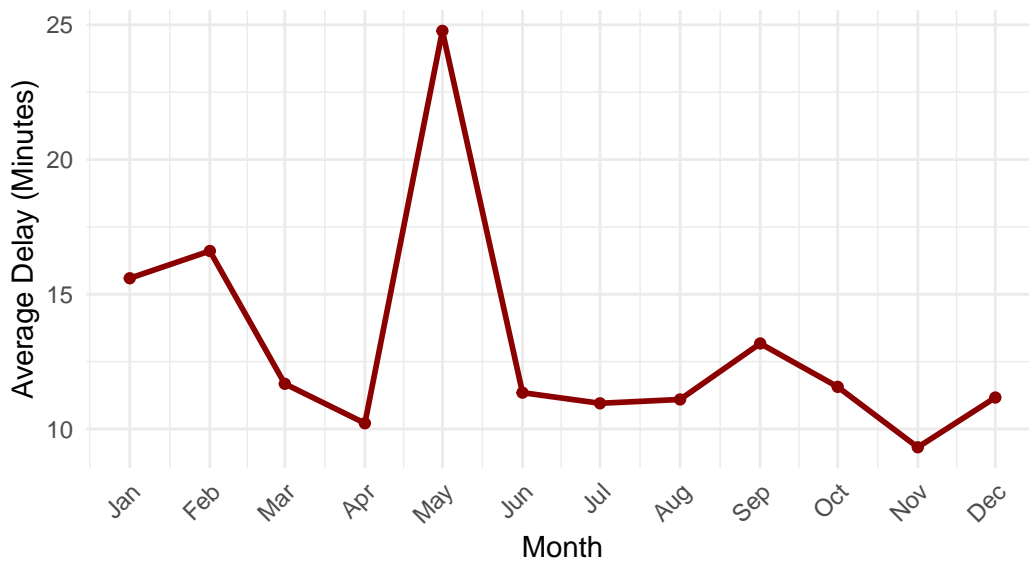
Measures the 'friction' visitors face when entering the BIA



```
plot_monthly_severity(beaches_ttc_metrics24, 2024)
```

Severity: Average Duration of Delays – 2024

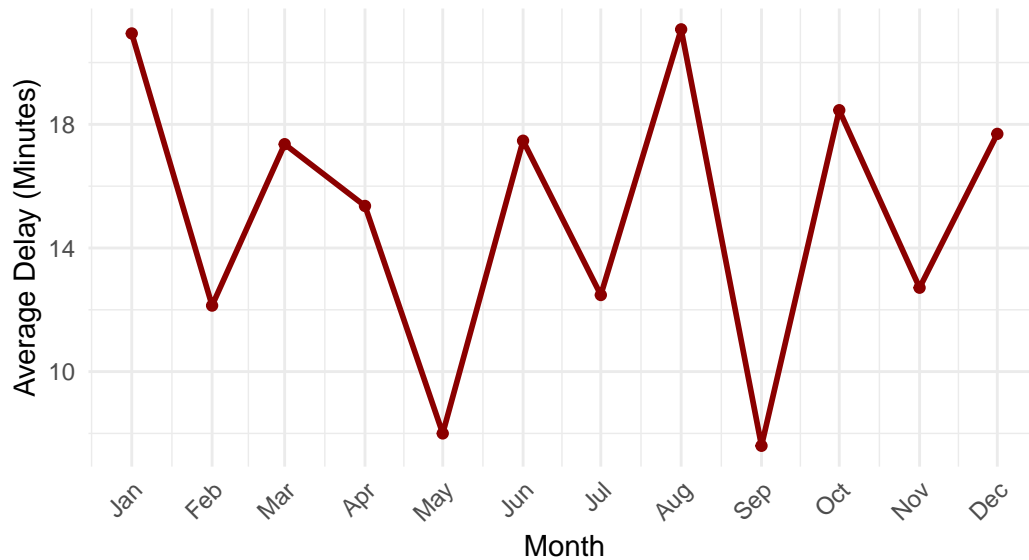
Measures the 'friction' visitors face when entering the BIA



```
plot_monthly_severity(beaches_ttc_metrics23, 2023)
```

Severity: Average Duration of Delays – 2023

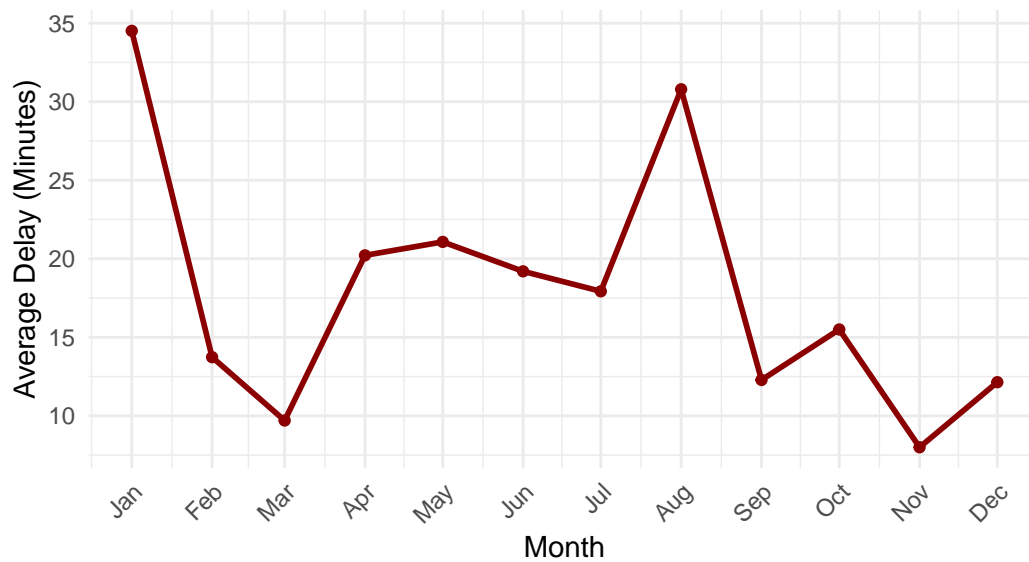
Measures the 'friction' visitors face when entering the BIA



```
plot_monthly_severity(beaches_ttc_metrics22, 2022)
```

Severity: Average Duration of Delays – 2022

Measures the 'friction' visitors face when entering the BIA



```
delay_codes <- get_resource("06d13be4-a8b7-4365-ac68-2169de8e0630")
```

TTC TRAFFIC DATA (can't find for BEACHES roads)

```
# get package
package <- show_package("7a0ac637-43da-4e42-a79c-6d8279e21d85")
package
```

```
# A tibble: 5 x 11
  title          id    topics civic_issues publisher excerpt dataset_category
  <chr>          <chr> <chr>   <chr>         <chr>    <chr>   <chr>
1 Traffic Volumes ~ 7a0a~ Trans~ Mobility      Transpor~ This d~ Table
2 Traffic Volumes ~ 7a0a~ Trans~ Mobility      Transpor~ This d~ Table
3 Traffic Volumes ~ 7a0a~ Trans~ Mobility      Transpor~ This d~ Table
4 Traffic Volumes ~ 7a0a~ Trans~ Mobility      Transpor~ This d~ Table
5 Traffic Volumes ~ 7a0a~ Trans~ Mobility      Transpor~ This d~ Table
# i 4 more variables: num_resources <int>, formats <chr>, refresh_rate <chr>,
#   last_refreshed <date>
```

```
traffic_res <- list_package_resources("7a0ac637-43da-4e42-a79c-6d8279e21d85")

traffic <- get_resource("2aef8448-455c-4d61-8480-4fa38b5acf9a")

glimpse(traffic)
```

```
Rows: 17,568
Columns: 10
$ id          <int> 8926341, 8926342, 8926343, 8926344, 8926345, 8926346, 89~
$ count_id    <int> 2856386, 2856386, 2856386, 2856386, 2856386, 2856386, 28~
$ location_name <chr> "Bayview Ave to Corktown Common Trl", "Bayview Ave to Co~
$ longitude    <dbl> -79.35313, -79.35313, -79.35313, -79.35313, -79.35313, -~
~
$ latitude     <dbl> 43.65564, 43.65564, 43.65564, 43.65564, 43.65564, 43.655~
$ centreline_id <int> 30061072, 30061072, 30061072, 30061072, 30061072, 300610~
$ time_start   <chr> "2021-04-27 22:00:00", "2021-04-27 22:15:00", "2021-04-~
2~
$ time_end     <chr> "2021-04-27 22:15:00", "2021-04-27 22:30:00", "2021-04-~
2~
$ direction    <chr> "EB", "EB", "EB", "EB", "EB", "EB", "EB", "EB", "EB", "E~
$ volume_15min <int> 9, 11, 6, 5, 7, 7, 6, 7, 2, 6, 10, 3, 2, 4, 3, 1, 2, 2, ~
```

```
traffic
```

```
# A tibble: 17,568 x 10
```

	id	count_id	location_name	longitude	latitude	centreline_id	time_start
	<int>	<int>	<chr>	<dbl>	<dbl>	<int>	<chr>
1	8926341	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
2	8926342	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
3	8926343	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
4	8926344	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
5	8926345	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
6	8926346	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
7	8926347	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
8	8926348	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-2~
9	8926349	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-3~
10	8926350	2856386	Bayview Ave to ~	-79.4	43.7	30061072	2021-04-3~

```
# i 17,558 more rows
```

```
# i 3 more variables: time_end <chr>, direction <chr>, volume_15min <int>
```

```
library(tidyverse)
```

```
library(sf)
```

```
filter_beaches_coords <- function(df) {
```

```
  # Define the Beaches BIA Bounding Box
```

```
  lat_min <- 43.6658
```

```
  lat_max <- 43.67417
```

```
  lon_min <- -79.31663
```

```
  lon_max <- -79.28164
```

```
  df %>%
```

```
    filter(
```

```
      latitude >= lat_min & latitude <= lat_max,
```

```
      longitude >= lon_min & longitude <= lon_max
```

```

    )
}

# Apply to your traffic data
colnames(traffic)

[1] "id"          "count_id"      "location_name" "longitude"
[5] "latitude"     "centreline_id" "time_start"     "time_end"
[9] "direction"    "volume_15min"

```

```

beaches_traffic_geo <- filter_beaches_coords(traffic)
beaches_traffic_geo

```

```

# A tibble: 0 x 10
# i 10 variables: id <int>, count_id <int>, location_name <chr>,
#   longitude <dbl>, latitude <dbl>, centreline_id <int>, time_start <chr>,
#   time_end <chr>, direction <chr>, volume_15min <int>

```

```

lat_min <- 43.6658
lat_max <- 43.67417
lon_min <- -79.31663
lon_max <- -79.28164

# Define the Beaches Bounding Box
# W: -79.320, E: -79.280, S: 43.660, N: 43.678
lat_min <- 43.660
lat_max <- 43.678
lon_min <- -79.320
lon_max <- -79.280

# Filter the raw traffic data
traffic_beaches <- traffic %>%
  filter(
    latitude >= lat_min,
  )
traffic_beaches

```

```

# A tibble: 1,152 x 10
      id count_id location_name longitude latitude centreline_id time_start
  <int>   <int>   <chr>         <dbl>    <dbl>         <int> <chr>

```

```

 1 8927925 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 2 8927885 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 3 8927886 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 4 8927887 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 5 8927888 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 6 8927889 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 7 8927890 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 8 8927891 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
 9 8927892 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
10 8927893 2863434 Codsell Ave: Li~ -79.4 43.8 14073490 2021-
10-0~
# i 1,142 more rows
# i 3 more variables: time_end <chr>, direction <chr>, volume_15min <int>

```

```

traffic %>%
  filter(as.numeric(longitude) > -79.4 & as.numeric(longitude) < -79.280) |>
  filter(as.numeric(latitude) > 43.66 & as.numeric(latitude) < 43.69)

```

```

# A tibble: 576 x 10
      id count_id location_name longitude latitude centreline_id time_start
  <int>   <int>   <chr>         <dbl>   <dbl>         <int> <chr>
1 8931053 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~
2 8931054 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~
3 8931055 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~
4 8931056 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~
5 8931057 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~
6 8931058 2871638 Birch Ave: Lion~ -79.4    43.7         30105718 2022-
08-1~

```

```

 7 8931059 2871638 Birch Ave: Lion~ -79.4 43.7 30105718 2022-
08-1~
 8 8931060 2871638 Birch Ave: Lion~ -79.4 43.7 30105718 2022-
08-1~
 9 8931061 2871638 Birch Ave: Lion~ -79.4 43.7 30105718 2022-
08-1~
10 8931062 2871638 Birch Ave: Lion~ -79.4 43.7 30105718 2022-
08-1~
# i 566 more rows
# i 3 more variables: time_end <chr>, direction <chr>, volume_15min <int>

```

GEOJSON FILE

```
geo_data <- st_read("BIA.geojson")
```

```

Reading layer `Business Improvement Areas Data - 4326' from data source
`/Users/arusansurendiran/Home/GitHub (R)/TheBeachBIA/BIA.geojson'
using driver `GeoJSON'
Simple feature collection with 85 features and 10 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:   xmin: -79.57099 ymin: 43.59387 xmax: -79.22213 ymax: 43.79714
Geodetic CRS:   WGS 84

```

```

# Take a look at the attribute table (the non-spatial data)
head(geo_data)

```

```

Simple feature collection with 6 features and 10 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:   xmin: -79.44661 ymin: 43.63812 xmax: -79.35628 ymax: 43.73624
Geodetic CRS:   WGS 84

```

	X_id	AREA_ID	DATE_EFFECTIVE	AREA_ATTR_ID	PARENT_AREA_ID	AREA_SHORT_CODE
1	1	2763038	2026-01-07 14:12:05	26262344	<NA>	087-00
2	2	2763037	2026-01-07 14:12:05	26262343	<NA>	082-00
3	3	2763016	2026-01-07 14:12:05	26262322	<NA>	006-00

```

4      4 2763015 2026-01-07 14:12:05      26262321      <NA>      040-
00
5      5 2763061 2026-01-07 14:12:05      26262367      <NA>      060-
02
6      6 2763060 2026-01-07 14:12:05      26262366      <NA>      070-
01

```

	AREA_LONG_CODE	AREA_NAME	AREA_DESC	OBJECTID
1	087-00	Ossington Avenue	Ossington Avenue	21664625
2	082-00	Oakwood Village	Oakwood Village	21664641
3	006-00	Parkdale Village	Parkdale Village	21664657
4	040-00	Yonge Lawrence Village	Yonge Lawrence Village	21664673
5	060-02	Uptown Yonge	Uptown Yonge	21664689
6	070-01	Historic Queen East	Historic Queen East	21664705

```

geometry
1 MULTIPOLYGON (((-79.42008 4...
2 MULTIPOLYGON (((-79.44055 4...
3 MULTIPOLYGON (((-79.44574 4...
4 MULTIPOLYGON (((-79.40306 4...
5 MULTIPOLYGON (((-79.39841 4...
6 MULTIPOLYGON (((-79.36416 4...

```

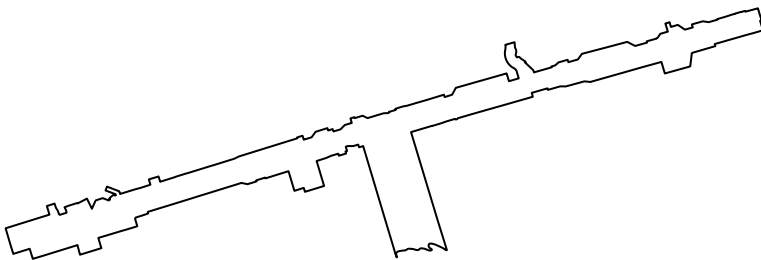
```
geo_data$AREA_NAME
```

[1] "Ossington Avenue"	"Oakwood Village"
[3] "Parkdale Village"	"Yonge Lawrence Village"
[5] "Uptown Yonge"	"Historic Queen East"
[7] "Leslieville"	"Baby Point Gates"
[9] "Greektown on the Danforth"	"Cedarbrae Markham Lawrence Village "
[11] "Mirvish Village"	"Shop The Queensway"
[13] "Downtown Yonge"	"Wilson Village"
[15] "Little Portugal Toronto"	"MarkeTO District"
[17] "Toronto Downtown West"	"Bloordale Village"
[19] "The Eglinton Way"	"Hillcrest Village"
[21] "Financial District"	"York-Eglinton "
[23] "Mount Pleasant Village"	"The Junction"
[25] "Bloor West Village"	"Village of Islington"
[27] "The Waterfront"	"Little Italy"
[29] "Korea Town"	"The Kingsway"
[31] "Church-Wellesley Village"	"Bayview Leaside"
[33] "Midtown Yonge"	"West Queen West"
[35] "Bloor Annex"	"Dovercourt Village"
[37] "Long Branch"	"Yonge & St. Clair"

[39] "Wexford Heights"	"Emery Village"
[41] "Kennedy Road"	"Bloorcourt Village"
[43] "Lawrence Ingram Keele"	"College Promenade"
[45] "Mount Dennis"	"Bloor By The Park"
[47] "The Beach"	"Roncesvalles Village"
[49] "Trinity-Bellwoods"	"Rogers Road"
[51] "Forest Hill Village"	"Cabbagetown"
[53] "Lakeshore Village"	"Pape Village"
[55] "Harbord Street"	"Danforth Village"
[57] "Chinatown"	"Regal Heights Village"
[59] "College West"	"Upper Village"
[61] "Sheppard East Village"	"Corso Italia"
[63] "Danforth Mosaic"	"Eglinton Hill"
[65] "Wychwood Heights"	"Fairbank Village"
[67] "Crossroads of the Danforth"	"Weston Village"
[69] "Mimico Village"	"DuKe Heights"
[71] "CityPlace and Fort York"	"Mimico By The Lake"
[73] "Dupont by the Castle"	"Rosedale Main Street"
[75] "Broadview Danforth"	"Albion Islington Square"
[77] "Queen Street West"	"Old Town Toronto"
[79] "Riverside District"	"Gerrard India Bazaar"
[81] "Liberty Village"	"Kensington Market"
[83] "Yonge North York"	"St. Clair Gardens"
[85] "Bloor-Yorkville"	

```
my_area <- geo_data %>%
  filter(AREA_NAME == "The Beach")

plot(st_geometry(my_area))
```



```
my_area
```

Simple feature collection with 1 feature and 10 fields

```

Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:   xmin: -79.31663 ymin: 43.6658 xmax: -79.28164 ymax: 43.67417
Geodetic CRS:  WGS 84
  X_id AREA_ID      DATE_EFFECTIVE AREA_ATTR_ID PARENT_AREA_ID AREA_SHORT_CODE
1   47 2762996 2026-01-07 14:12:05      26262302          <NA>          053-
00
  AREA_LONG_CODE AREA_NAME AREA_DESC OBJECTID                      geometry
1              053-00 The Beach The Beach 21665361 MULTIPOLYGON (((-79.30957 4...

```

TORONTO WEATHER

```
climate_daily <- read_csv("climate-daily.csv")
```

```
Rows: 1479 Columns: 36
```

```
-- Column specification -----
```

```
Delimiter: ","
```

```
chr  (13): STATION_NAME, ID, PROVINCE_CODE, MEAN_TEMPERATURE_FLAG, MIN_TEMPE...
```

```
dbl  (16): x, y, CLIMATE_IDENTIFIER, LOCAL_YEAR, LOCAL_MONTH, LOCAL_DAY, MEA...
```

```
lgl   (6): TOTAL_RAIN, TOTAL_RAIN_FLAG, TOTAL_SNOW, TOTAL_SNOW_FLAG, SNOW_ON...
```

```
dtm   (1): LOCAL_DATE
```

```
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.
```

```
# 1. Cleaning and Preparation
```

```
climate_cleaned <- climate_daily %>%
```

```
  mutate(
```

```
    Date = as.Date(LOCAL_DATE),
```

```
    Year = as.factor(LOCAL_YEAR),
```

```
    # Create an ordered month factor for better plotting
```

```
    Month_Label = factor(month.abb[LOCAL_MONTH], levels = month.abb)
```

```
  ) %>%
```

```
  # Filter for years we want to focus on (2022 to current)
```

```
  filter(LOCAL_YEAR >= 2022)
```

```
# 2. Plot: Combined Time Series (The "Together" Plot)
```

```
# Visualizes the continuous trend of temperature over the years
```

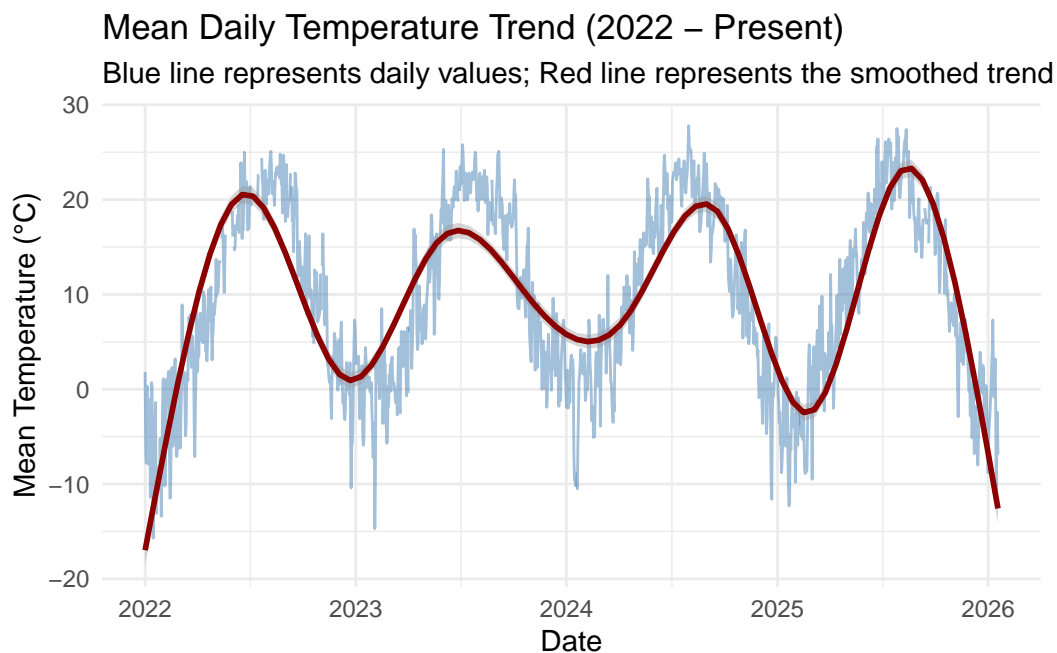
```
ggplot(climate_cleaned, aes(x = Date, y = MEAN_TEMPERATURE)) +
```

```
  geom_line(color = "steelblue", alpha = 0.5) +
```

```
geom_smooth(color = "darkred", method = "gam", span = 0.1) +
labs(
  title = "Mean Daily Temperature Trend (2022 – Present)",
  subtitle = "Blue line represents daily values; Red line represents the smoothed trend",
  x = "Date",
  y = "Mean Temperature (°C)"
) +
theme_minimal()
```

`geom_smooth()` using formula = 'y ~ s(x, bs = "cs")'

Warning: Removed 44 rows containing non-finite outside the scale range
(`stat_smooth()`).



```
# 3. Plot: Individual Yearly Comparison (Faceted)
# Allows you to see year-over-year variability in a single view
ggplot(climate_cleaned, aes(x = yday(Date), y = MEAN_TEMPERATURE, color = Year)) +
  geom_line() +
  facet_wrap(~Year, ncol = 1) +
  labs(
    title = "Yearly Temperature Profiles",
    subtitle = "Comparing the progression of the year across 2022-2025",
```

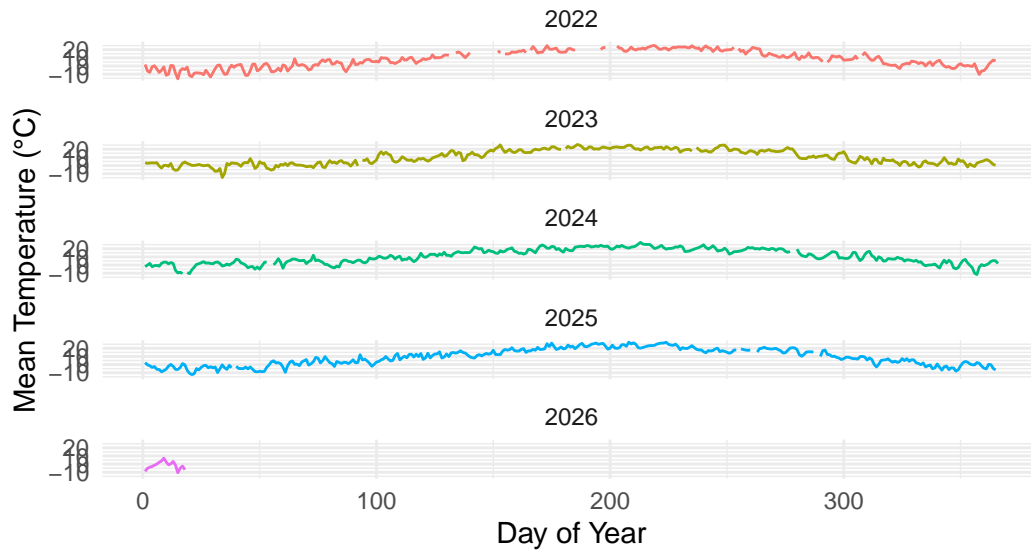
```

x = "Day of Year",
y = "Mean Temperature (°C)"
) +
theme_minimal() +
theme(legend.position = "none")

```

Yearly Temperature Profiles

Comparing the progression of the year across 2022–2025



```

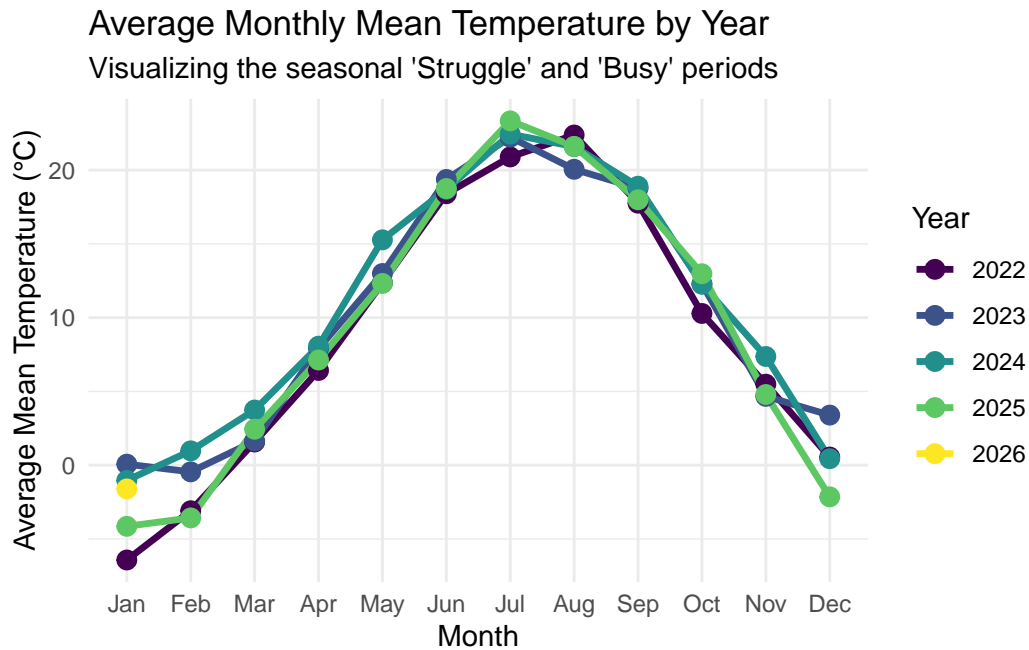
# 4. Plot: Average Monthly Temperatures (Seasonality)
# Aggregates daily data to show the 'typical' temperature for each month per year
monthly_avg_temp <- climate_cleaned %>%
  group_by(Year, Month_Label) %>%
  summarise(Avg_Monthly_Temp = mean(MEAN_TEMPERATURE, na.rm = TRUE), .groups = "drop")

ggplot(monthly_avg_temp, aes(x = Month_Label, y = Avg_Monthly_Temp, group = Year, color = Year)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  scale_color_viridis_d() +
  labs(
    title = "Average Monthly Mean Temperature by Year",
    subtitle = "Visualizing the seasonal 'Struggle' and 'Busy' periods",
    x = "Month",
    y = "Average Mean Temperature (°C)",
    color = "Year"
  )

```

```
) +  
theme_minimal()
```

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.



TORONTO PARKING TICKETS DATA

```
# get package  
package <- show_package("8c233bc2-1879-44ff-a0e4-9b69a9032c54")  
  
tickets_res <- list_package_resources("8c233bc2-1879-44ff-a0e4-9b69a9032c54")  
  
tickets_data <- get_resource("3263cbd6-39f8-46c9-8ca6-bd8ffc730157")  
  
tickets24_data1 <- tickets_data[[1]]  
tickets24_data10 <- tickets_data[[2]]  
tickets24_data11 <- tickets_data[[3]]  
tickets24_data12 <- tickets_data[[4]]  
tickets24_data2 <- tickets_data[[5]]
```

```

tickets24_data3 <- tickets_data[[6]]
tickets24_data4 <- tickets_data[[7]]
tickets24_data5 <- tickets_data[[8]]
tickets24_data6 <- tickets_data[[9]]
tickets24_data7 <- tickets_data[[10]]
tickets24_data8 <- tickets_data[[11]]
tickets24_data9 <- tickets_data[[12]]

```

```

filter_beaches_parking_addresses <- function(df) {
  df %>%
    mutate(
      # 1. Clean the string to uppercase
      location_clean = toupper(location2),

      # 2. Extract the first set of numbers found in the address
      address_num = as.numeric(str_extract(location_clean, "\\d+")),

    ) %>%
    filter(
      # Logic: (Street Name matches) AND (Number is within BIA bounds)
      (str_detect(location_clean, "QUEEN ST E| QUEEN STREET EAST") & address_num >= 1590 & address_num <= 1599) |

      #(str_detect(location_clean, "WOODBINE")      & address_num <= 150) |
      #(str_detect(location_clean, "COXWELL")       & address_num <= 100) |
      #(str_detect(location_clean, "LEE AVE")        & address_num <= 80) |
      #(str_detect(location_clean, "WINEVA")         & address_num <= 80) |
      #(str_detect(location_clean, "NEVILLE")      & address_num <= 50)

    )
  }

# Apply to your data
beaches24_parking_1 <- filter_beaches_parking_addresses(tickets24_data1)
beaches24_parking_2 <- filter_beaches_parking_addresses(tickets24_data2)
beaches24_parking_3 <- filter_beaches_parking_addresses(tickets24_data3)
beaches24_parking_4 <- filter_beaches_parking_addresses(tickets24_data4)
beaches24_parking_5 <- filter_beaches_parking_addresses(tickets24_data5)
beaches24_parking_6 <- filter_beaches_parking_addresses(tickets24_data6)
beaches24_parking_7 <- filter_beaches_parking_addresses(tickets24_data7)
beaches24_parking_8 <- filter_beaches_parking_addresses(tickets24_data8)
beaches24_parking_9 <- filter_beaches_parking_addresses(tickets24_data9)
beaches24_parking_10 <- filter_beaches_parking_addresses(tickets24_data10)
beaches24_parking_11 <- filter_beaches_parking_addresses(tickets24_data11)

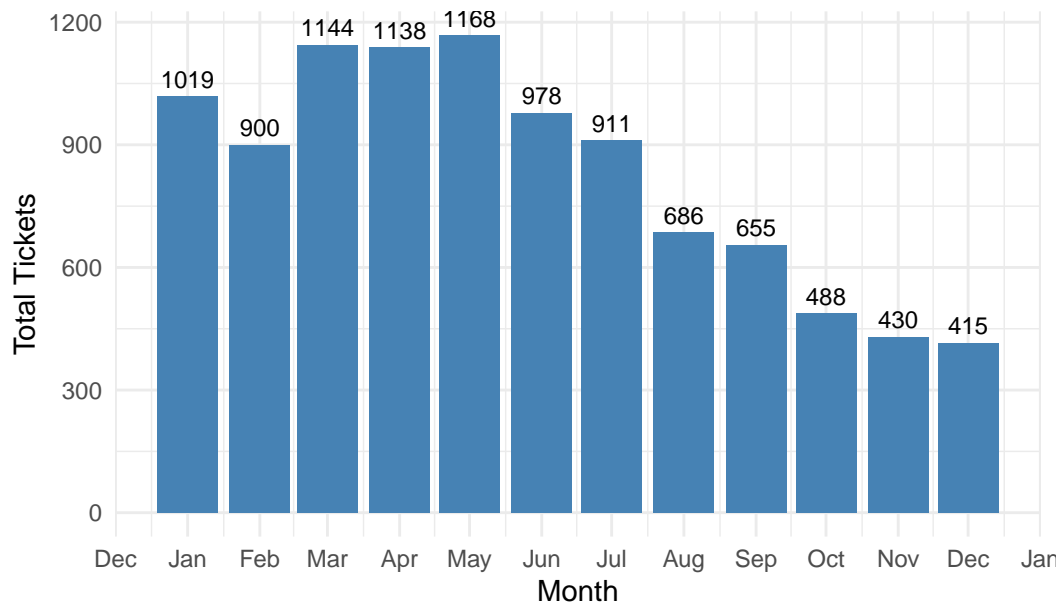
```

```
beaches24_parking_12 <- filter_beaches_parking_addresses(tickets24_data12)
```

```
# Combine all 12 months into one dataframe
beaches24_all_parking <- bind_rows(
  beaches24_parking_1, beaches24_parking_2, beaches24_parking_3, beaches24_parking_4,
  beaches24_parking_5, beaches24_parking_6, beaches24_parking_7, beaches24_parking_8,
  beaches24_parking_9, beaches24_parking_10, beaches24_parking_11, beaches24_parking_12
) %>%
  mutate(
    # Convert integer date (20240118) to real Date object
    date_clean = ymd(as.character(date_of_infraction)),
    month = floor_date(date_clean, "month"),
    day_of_week = wday(date_clean, label = TRUE, abbr = TRUE),
    # Convert time (e.g., 1437) to hour (14)
    hour = floor(time_of_infraction / 100)
  )
```

```
beaches24_all_parking %>%
  count(month) %>%
  ggplot(aes(x = month, y = n)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = n), vjust = -0.5, size = 3) +
  scale_x_date(date_labels = "%b", date_breaks = "1 month") +
  labs(
    title = "Monthly Ticket Volume: Seasonality Analysis",
    x = "Month",
    y = "Total Tickets"
  ) +
  theme_minimal()
```

Monthly Ticket Volume: Seasonality Analysis



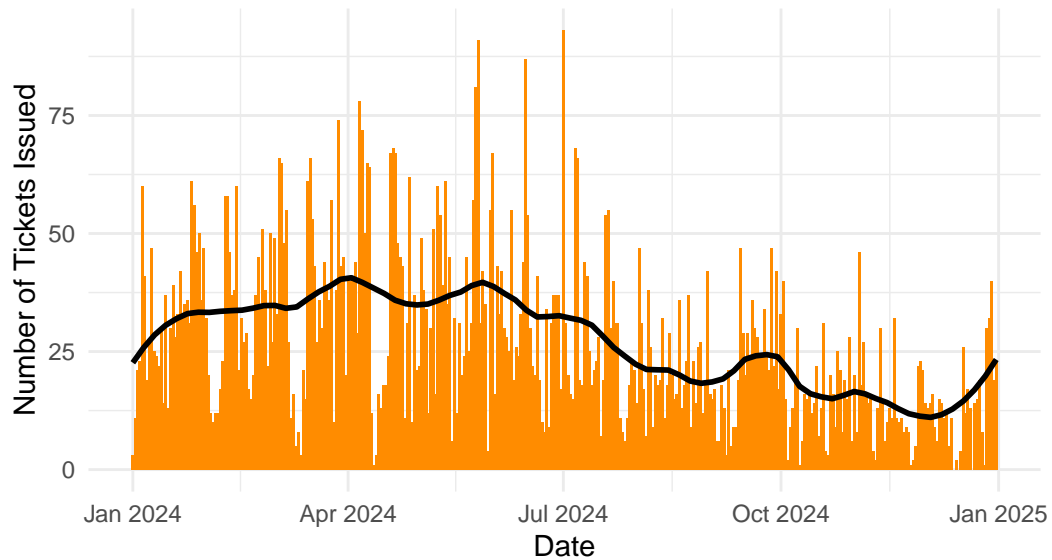
```
# 1. Create a daily summary first
daily_parking_counts <- beaches24_all_parking %>%
  group_by(date_clean) %>%
  summarise(ticket_count = n())

# 2. Now plot using the new 'ticket_count' column as Y
ggplot(daily_parking_counts, aes(x = date_clean, y = ticket_count)) +
  geom_col(fill = "darkorange") + # Using geom_col instead of geom_bar for pre-summed data
  geom_smooth(method = "loess", span = 0.2, color = "black", se = FALSE) +
  labs(
    title = "Daily Parking Tickets: The Beaches BIA (2024)",
    subtitle = "Black line shows the smoothed trend across seasons",
    x = "Date",
    y = "Number of Tickets Issued"
  ) +
  theme_minimal()
```

`geom_smooth()` using formula = 'y ~ x'

Daily Parking Tickets: The Beaches BIA (2024)

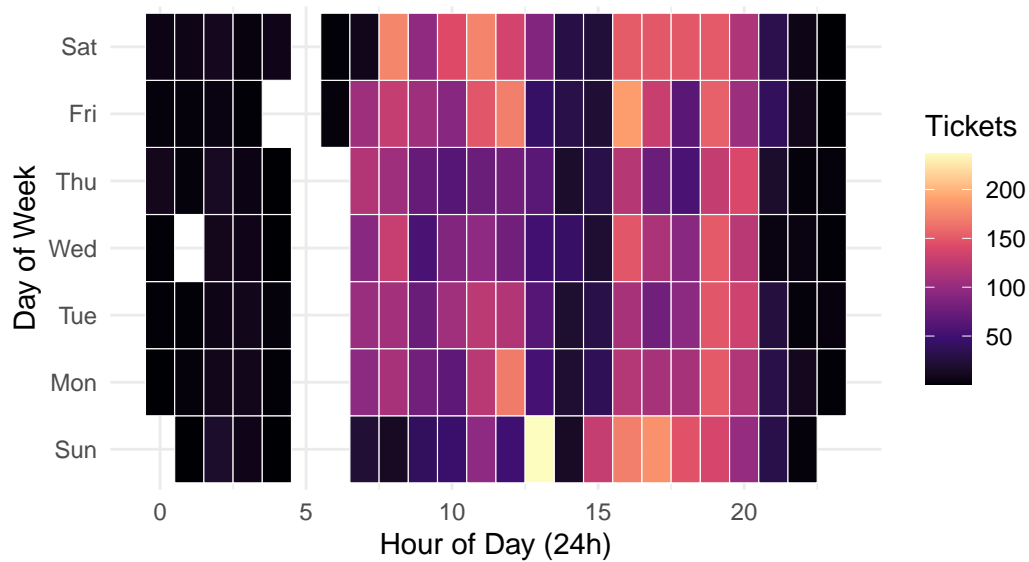
Black line shows the smoothed trend across seasons



```
beaches24_all_parking %>%
  group_by(day_of_week, hour) %>%
  summarise(ticket_count = n(), .groups = "drop") %>%
  ggplot(aes(x = hour, y = day_of_week, fill = ticket_count)) +
  geom_tile(color = "white") +
  scale_fill_viridis_c(option = "magma") +
  labs(
    title = "Parking Enforcement Heatmap: Day vs. Hour",
    subtitle = "Highlights when visitors face the most 'friction'",
    x = "Hour of Day (24h)",
    y = "Day of Week",
    fill = "Tickets"
  ) +
  theme_minimal()
```

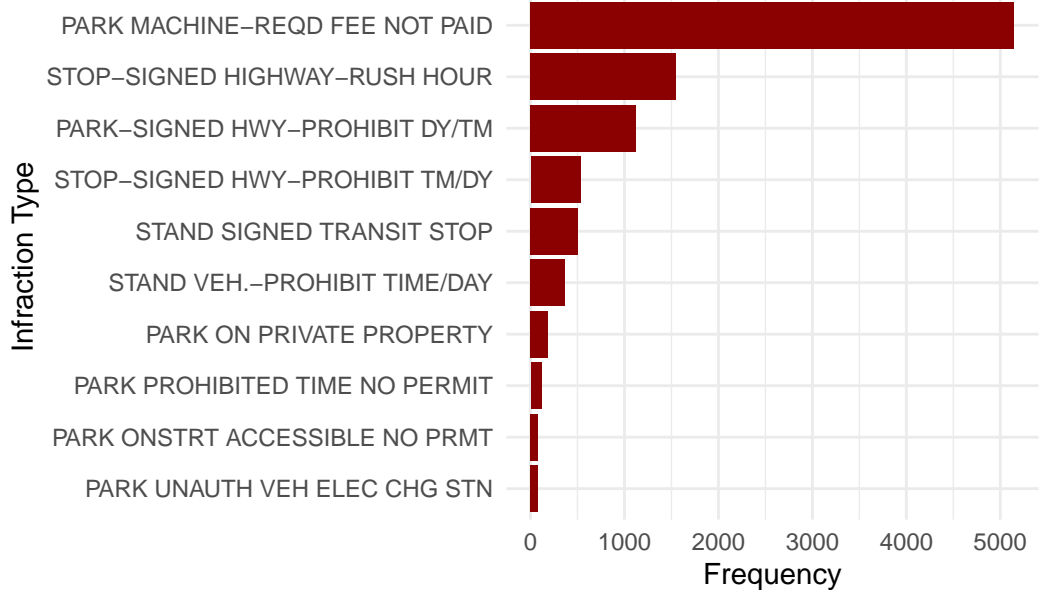
Parking Enforcement Heatmap: Day vs. Hour

Highlights when visitors face the most 'friction'



```
beaches24_all_parking %>%
  count(infraction_description, sort = TRUE) %>%
  head(10) %>%
  ggplot(aes(x = reorder(infraction_description, n), y = n)) +
  geom_col(fill = "darkred") +
  coord_flip() +
  labs(
    title = "Top 10 Parking Infractions in The Beaches",
    x = "Infraction Type",
    y = "Frequency"
  ) +
  theme_minimal()
```

Top 10 Parking Infractions in The Be

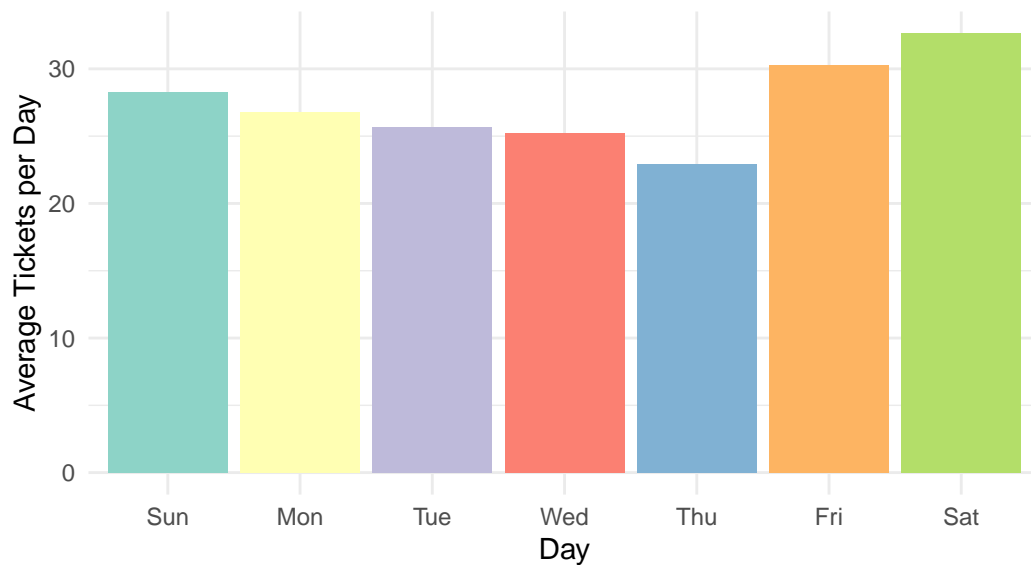


```
# Calculate average tickets per specific day of the week
dow_summary <- beaches24_all_parking %>%
  count(date_clean, day_of_week) %>%
  group_by(day_of_week) %>%
  summarise(avg_tickets = mean(n), .groups = "drop")

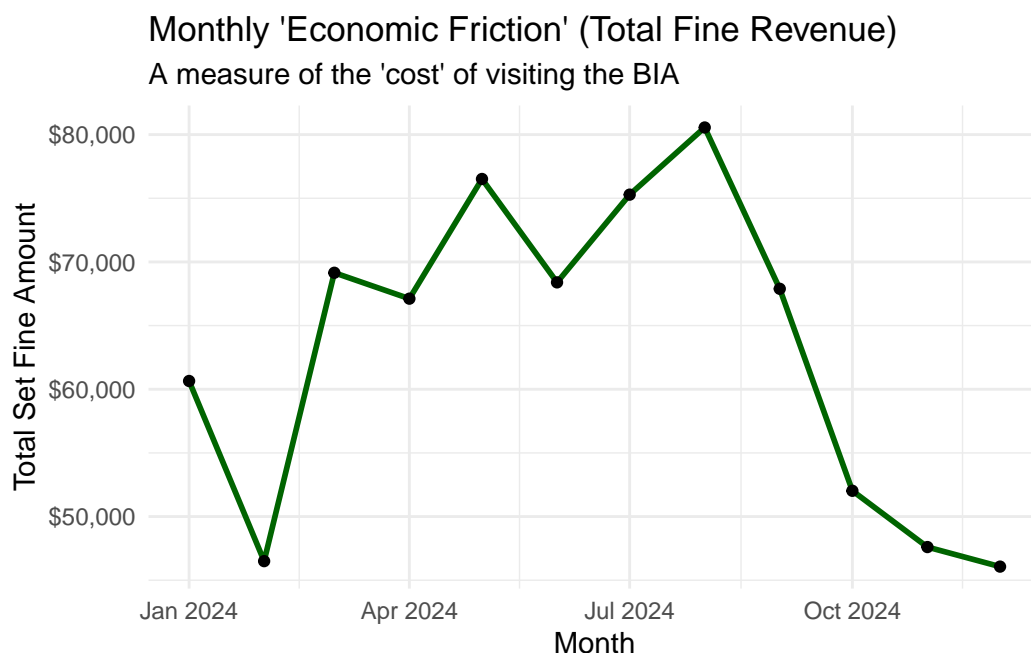
ggplot(dow_summary, aes(x = day_of_week, y = avg_tickets, fill = day_of_week)) +
  geom_col() +
  scale_fill_brewer(palette = "Set3") +
  labs(
    title = "Average Parking Tickets by Day of the Week",
    subtitle = "Higher weekend averages suggest leisure-driven visitor inflow",
    x = "Day",
    y = "Average Tickets per Day"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

Average Parking Tickets by Day of the Week

Higher weekend averages suggest leisure-driven visitor inflow



```
beaches24_all_parking %>%
  group_by(month) %>%
  summarise(total_revenue = sum(set_fine_amount)) %>%
  ggplot(aes(x = month, y = total_revenue)) +
  geom_line(size = 1, color = "darkgreen") +
  geom_point() +
  scale_y_continuous(labels = scales::dollar) +
  labs(
    title = "Monthly 'Economic Friction' (Total Fine Revenue)",
    subtitle = "A measure of the 'cost' of visiting the BIA",
    x = "Month",
    y = "Total Set Fine Amount"
  ) +
  theme_minimal()
```



THE BEACHES BEACH QUALITY

```
# get all resources for this package
resources <- list_package_resources("c8ce3e2f-935d-4ef6-ade2-27e08df239f2")

# load the first datastore resource as a sample
beach_quality_data <- get_resource("42551dd8-a3c1-45fa-9e8b-674c43b4d3b7")

kewbeach_quality_data <- beach_quality_data |> filter(beachName == "Kew Balmy Beach")
```

```
kew_cleaned <- kewbeach_quality_data %>%
  mutate(
    Date = as.Date(dataCollectionDate),
    Year = year(Date),
    Month = month(Date, label = TRUE),
    # Clean waterTemp: Water in Lake Ontario doesn't exceed ~28°C
    waterTemp = ifelse(waterTemp > 35, NA, waterTemp)
  )
```

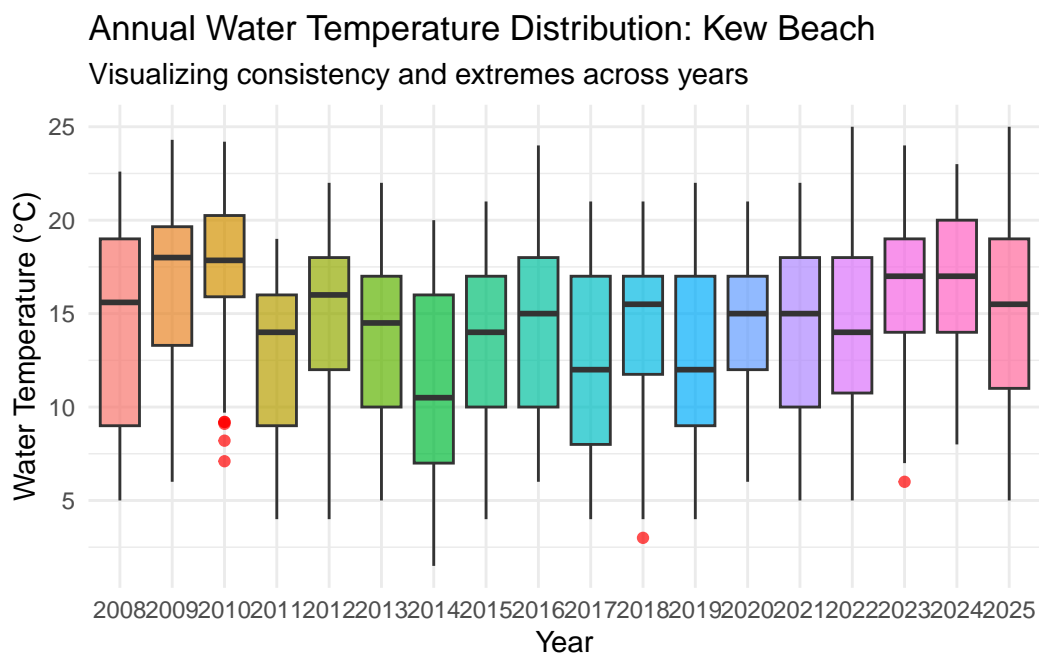
```
ggplot(kew_cleaned, aes(x = factor(Year), y = waterTemp, fill = factor(Year))) +
  geom_boxplot(outlier.color = "red", alpha = 0.7) +
```

```

labs(
  title = "Annual Water Temperature Distribution: Kew Beach",
  subtitle = "Visualizing consistency and extremes across years",
  x = "Year",
  y = "Water Temperature (°C)"
) +
theme_minimal() +
theme(legend.position = "none")

```

Warning: Removed 120 rows containing non-finite outside the scale range (`stat_boxplot()`).



```

kew_recent <- kew_cleaned %>%
  filter(Year >= 2022)

# Monthly Average Statistics for Water Temp
kew_recent_summary <- kew_recent %>%
  group_by(Year, Month) %>%
  summarise(
    Avg_Water_Temp = mean(waterTemp, na.rm = TRUE),
    Avg_Air_Temp = mean(airTemp, na.rm = TRUE),
    Avg_WaterFowl = mean(waterFowl, na.rm = TRUE),

```

```

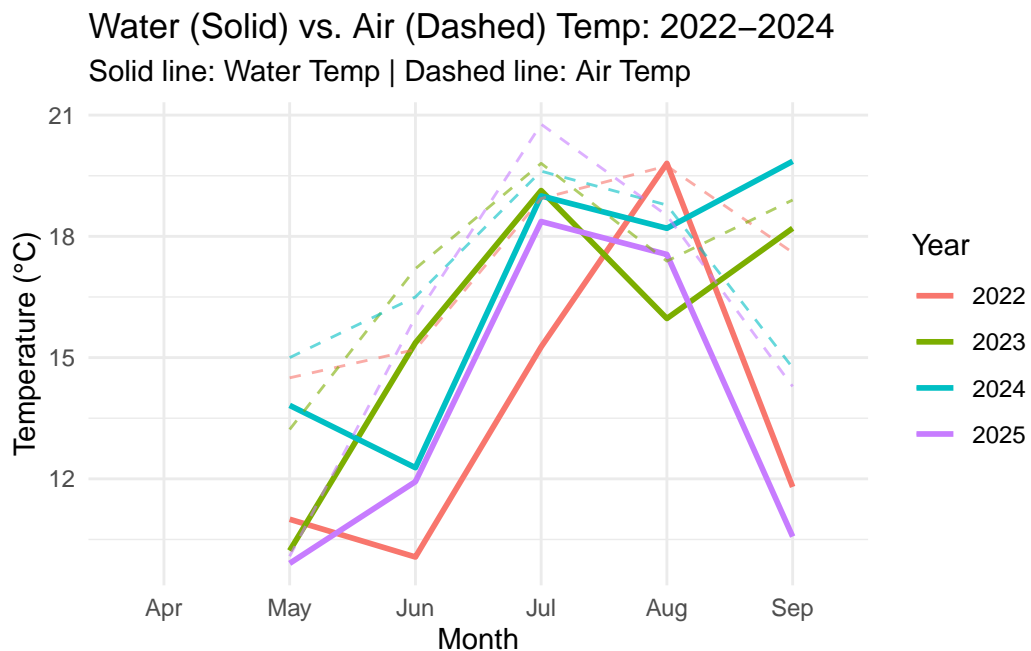
    .groups = "drop"
  )

# Plot: Water vs Air Seasonality
ggplot(kew_recent_summary, aes(x = Month, group = Year, color = factor(Year))) +
  geom_line(aes(y = Avg_Water_Temp), size = 1) +
  geom_line(aes(y = Avg_Air_Temp), linetype = "dashed", alpha = 0.6) +
  labs(
    title = "Water (Solid) vs. Air (Dashed) Temp: 2022-2024",
    subtitle = "Solid line: Water Temp | Dashed line: Air Temp",
    x = "Month",
    y = "Temperature (°C)",
    color = "Year"
  ) +
  theme_minimal()

```

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

Removed 1 row containing missing values or values outside the scale range (`geom_line()`).



```

# 1. Clean and Prepare the Daily Data
kew_daily_recent <- kewbeach_quality_data %>%
  mutate(
    Date = as.Date(dataCollectionDate),
    Year = year(Date),
    # Handle the 201°C outlier in water temp
    waterTemp = ifelse(waterTemp > 50, NA, waterTemp),
    # Create a 'Day of Season' to overlap lines if needed later
    Day_of_Year = yday(Date)
  ) %>%
  filter(Year >= 2022)

# 2. Daily Water Temperature Trends (Faceted by Year)
ggplot(kew_daily_recent, aes(x = Date, y = waterTemp)) +
  geom_line(color = "steelblue", size = 0.8) +
  geom_point(color = "steelblue", size = 1, alpha = 0.5) +
  # Scales = "free_x" ensures each year's panel only shows its specific season
  facet_wrap(~Year, scales = "free_x", ncol = 1) +
  labs(
    title = "Daily Water Temperature Trends: Kew Beach",
    subtitle = "Tracking the onset and duration of the swimmable season",
    x = "Date",
    y = "Water Temperature (°C)"
  ) +
  theme_minimal()

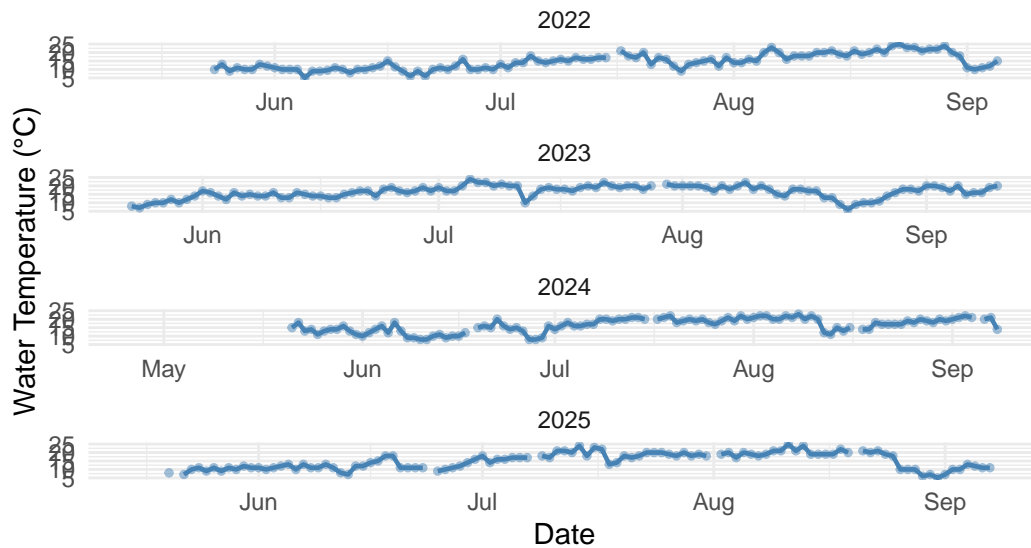
```

Warning: Removed 7 rows containing missing values or values outside the scale range (``geom_line()``).

Warning: Removed 18 rows containing missing values or values outside the scale range (``geom_point()``).

Daily Water Temperature Trends: Kew Beach

Tracking the onset and duration of the swimmable season



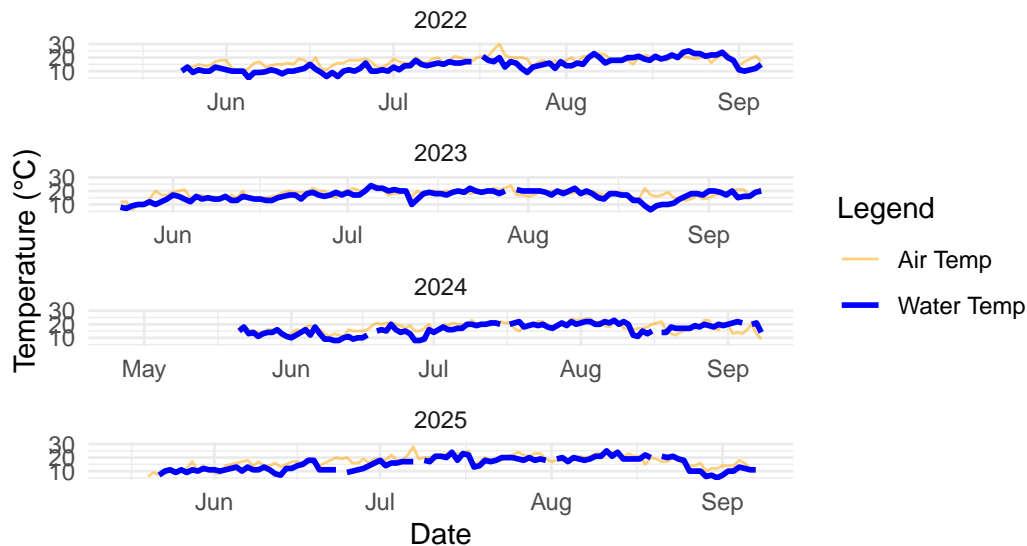
```
# 3. Daily Air vs. Water Temperature Comparison
ggplot(kew_daily_recent, aes(x = Date)) +
  geom_line(aes(y = airTemp, color = "Air Temp"), alpha = 0.5) +
  geom_line(aes(y = waterTemp, color = "Water Temp"), size = 1) +
  facet_wrap(~Year, scales = "free_x", ncol = 1) +
  scale_color_manual(values = c("Air Temp" = "orange", "Water Temp" = "blue")) +
  labs(
    title = "Daily Atmosphere vs. Lake Interaction",
    subtitle = "Note how water temp lags behind or drops suddenly (upwelling)",
    x = "Date",
    y = "Temperature (°C)",
    color = "Legend"
  ) +
  theme_minimal()
```

Warning: Removed 7 rows containing missing values or values outside the scale range (`geom_line()`).

Warning: Removed 7 rows containing missing values or values outside the scale range (`geom_line()`).

Daily Atmosphere vs. Lake Interaction

Note how water temp lags behind or drops suddenly (upwelling)



TORONTO BEACHES ECOLI

```
# get package
package <- show_package("92b0de8f-1ada-44a7-84cf-adc04868e990")

# get all resources for this package
resources <- list_package_resources("92b0de8f-1ada-44a7-84cf-adc04868e990")

beach_ecoli_data <- get_resource("fa96223e-ccf8-4c0a-817b-6f5039311287")
kewbeach_ecoli_data <- beach_ecoli_data |> filter(beachName == "Kew Balmy Beach")

ecoli_cleaned <- kewbeach_ecoli_data %>%
  mutate(
    Date = as.Date(collectionDate),
    Year = year(Date),
    Month = month(Date, label = TRUE)
  ) %>%
  filter(Year >= 2022) %>%
  # Handle missing values and group by day to average the different test sites
  group_by(Date, Year, Month) %>%
  summarise(Daily_Avg_Ecoli = mean(eColi, na.rm = TRUE), .groups = "drop")
```

Toronto follows Ontario's criteria and will close a beach when E.coli levels exceed 100 E.coli per 100 millilitres of water

```
ggplot(ecoli_cleaned, aes(x = Date, y = Daily_Avg_Ecoli)) +
  geom_line(color = "darkgreen", alpha = 0.6) +
  geom_point(aes(color = Daily_Avg_Ecoli > 100), size = 1.5) +
  # The "Safe" threshold line
  geom_hline(yintercept = 100, linetype = "dashed", color = "red") +
  annotate("text", x = Inf, y = 110, label = "Safety Limit (100)",
          hjust = 1.1, color = "red", fontface = "bold") +
  scale_color_manual(values = c("FALSE" = "darkgreen", "TRUE" = "red")) +
  facet_wrap(~Year, scales = "free_x", ncol = 1) +
  scale_y_log10() + # Log scale helps see variations when there are massive spikes
  labs(
    title = "Daily E. coli Levels: Kew Beach",
    subtitle = "Spikes above 100 CFU indicate beach closures and reduced 'Third Space' utilization",
    x = "Date",
    y = "E. coli Count (CFU/100mL) - Log Scale",
    color = "Unsafe for Swimming"
  ) +
  theme_minimal()
```

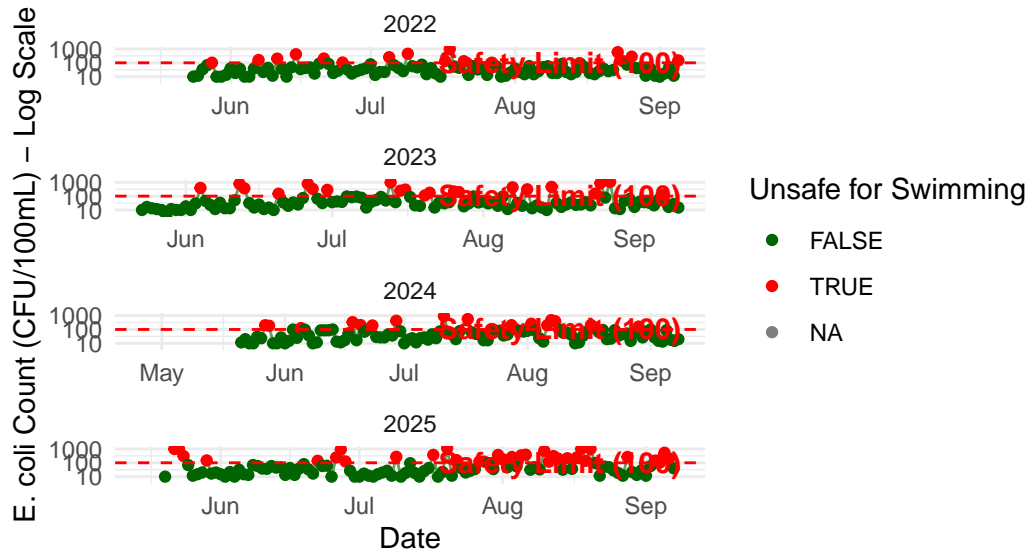
Warning in scale_x_date(): A <numeric> value was passed to a Date scale.
i The value was converted to a <Date> object.

Warning: Removed 7 rows containing missing values or values outside the scale range (``geom_line()``).

Warning: Removed 9 rows containing missing values or values outside the scale range (``geom_point()``).

Daily E. coli Levels: Kew Beach

Spikes above 100 CFU indicate beach closures and reduced 'Third Spac



```
ecoli_impact_stats <- ecoli_cleaned %>%
  group_by(Year) %>%
  summarise(
    Total_Testing_Days = n(),
    Unsafe_Days = sum(Daily_Avg_Ecoli > 100, na.rm = TRUE),
    Pct_Days_Closed = (Unsafe_Days / Total_Testing_Days) * 100,
    Max_Contamination = max(Daily_Avg_Ecoli, na.rm = TRUE)
  )

print(ecoli_impact_stats)
```

A tibble: 4 x 5

	Year	Total_Testing_Days	Unsafe_Days	Pct_Days_Closed	Max_Contamination
	<dbl>	<int>	<int>	<dbl>	<dbl>
1	2022	108	16	14.8	1000
2	2023	111	21	18.9	1058.
3	2024	113	19	16.8	922.
4	2025	113	33	29.2	1150

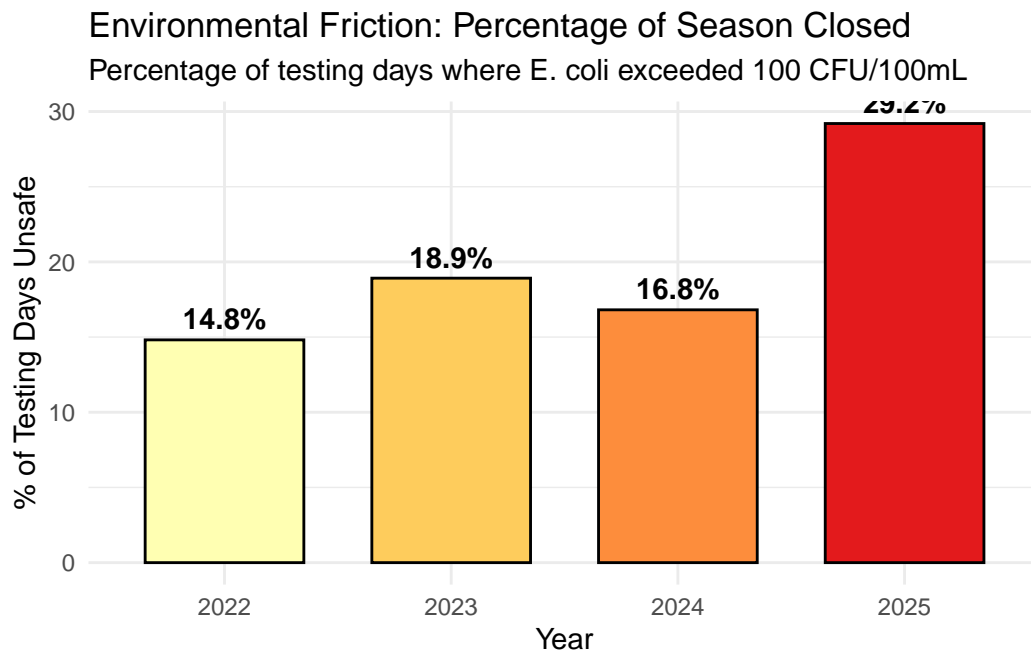
Plot: Percentage of Days the Beach was 'Closed' (Unsafe)

```
ggplot(ecoli_impact_stats, aes(x = factor(Year), y = Pct_Days_Closed, fill = factor(Year))) +
  geom_col(color = "black", width = 0.7) +
  geom_text(aes(label = paste0(round(Pct_Days_Closed, 1), "%")),
```

```

      vjust = -0.5, fontface = "bold") +
scale_fill_brewer(palette = "YlOrRd") +
labs(
  title = "Environmental Friction: Percentage of Season Closed",
  subtitle = "Percentage of testing days where E. coli exceeded 100 CFU/100mL",
  x = "Year",
  y = "% of Testing Days Unsafe",
  fill = "Year"
) +
theme_minimal() +
theme(legend.position = "none")

```



BIKESHARE DATA

```

library(opendatatoronto)
library(dplyr)

# get package
package <- show_package("7e876c24-177c-4605-9cef-e50dd74c617f")
package

```

```
# A tibble: 2 x 11
  title          id    topics civic_issues publisher excerpt dataset_category
  <chr>          <chr> <chr>  <chr>          <chr>    <chr>    <chr>
1 Bike Share Toron~ 7e87~ Trans~ Mobility    Toronto ~ "Histo~ Document
2 Bike Share Toron~ 7e87~ Trans~ Mobility    Toronto ~ "Histo~ Document
# i 4 more variables: num_resources <int>, formats <chr>, refresh_rate <chr>,
#   last_refreshed <date>
```

```
bike_resources <- list_package_resources("7e876c24-177c-4605-9cef-e50dd74c617f")
bike_resources
```

```
# A tibble: 11 x 4
  name          id          format last_modified
  <chr>          <chr>          <chr>    <date>
1 bikeshare-ridership-readme ad78f9f7-d4f2-42a1-9a1c-6~ XLSX 2022-04-12
2 bikeshare-ridership-2014-2015 85326868-508c-497e-b139-b~ XLSX 2022-04-11
3 bikeshare-ridership-2016 6814e3b2-9e57-4df6-915a-a~ XLSX 2022-04-11
4 bikeshare-ridership-2017 98b63ba7-24ba-41da-a788-1~ ZIP 2022-04-11
5 bikeshare-ridership-2018 f4e91cc6-b925-46bb-bb89-a~ ZIP 2022-04-11
6 bikeshare-ridership-2019 f725ec3b-4f5c-4013-8a39-a~ ZIP 2022-04-11
7 bikeshare-ridership-2020 e534141d-92c6-4dd7-8ba1-b~ ZIP 2022-04-11
8 bikeshare-ridership-2021 ddc039f6-07fa-47a3-a707-0~ ZIP 2022-04-11
9 bikeshare-ridership-2022 db10a7b1-2702-481c-b7f0-0~ ZIP 2023-01-18
10 bikeshare-ridership-2023 f0fa6a67-4571-4dd6-9d5a-d~ ZIP 2024-01-09
11 bikeshare-ridership-2024 9a9a0163-8114-447c-bf66-7~ ZIP 2024-10-25
```

```
bike_data <- get_resource("9a9a0163-8114-447c-bf66-790b1a92da51")
bike24_data_1 <- bike_data[[7]]
```

EMDAABIIMOK AVE / LAKE SHORE BLVD E ORCHARD PARK 1651 QUEEN ST E
 QUEEN ST. E / EASTERN AVE NORTHERN DANCER BLVD / LAKE SHORE BLVD
 E QUEEN ST E / JOSEPH DUGGAN RD WOODBINE AVE / LAKE SHORE BLVD E
 KEWBEACH AVE / KENILWORTH AVE 85 LEE AVE HUBBARD BLVD / GLEN MANOR
 DR HUBBARD BLVD / BALSAM AV QUEEN ST E / HAMMERSMITH AVE QUEEN ST.
 E / SPRUCE HILL RD. QUEEN ST E / NURSEWOOD RD

```
# 1. Prepare the Target List
targets <- tibble(original = c(
  "EMDAABIIMOK AVE / LAKE SHORE BLVD E", "ORCHARD PARK", "1651 QUEEN",
  "QUEEN ST. E / EASTERN AVE", "NORTHERN DANCER BLVD / LAKE SHORE BLVD E",
  "QUEEN ST E / JOSEPH DUGGAN RD", "WOODBINE AVE / LAKE SHORE BLVD E",
```

```

"KEWBEACH AVE / KENILWORTH AVE", "85 LEE AVE", "HUBBARD BLVD / GLEN MANOR DR",
"HUBBARD BLVD / BALSAM AV", "QUEEN ST E / HAMMERSMITH AVE",
"QUEEN ST. E / SPRUCE HILL RD.", "QUEEN ST E / NURSEWOOD RD"
))

# 2. Get the unique stations from your 200k row dataset
dataset_stations <- unique(bike24_data_1$End.Station.Name)

# 3. Create a Custom Matcher
find_beaches_stations <- function(target, station_list) {
  # Clean the target: remove dots, extra spaces, make uppercase
  clean_target <- str_replace_all(toupper(target), "\\.", "") %>% str_squish()

  # Split into parts if there is a slash
  parts <- str_split(clean_target, " / ")[[1]]

  # Search the station_list
  matches <- station_list[apply(station_list, function(s) {
    s_clean <- str_replace_all(toupper(s), "\\.", "") %>% str_squish()
    # Logic: All parts of our target must be found in the station name
    all(apply(parts, function(p) str_detect(s_clean, fixed(p))))
  })]

  # Return the first match found, or NA
  if(length(matches) > 0) return(matches[1]) else return(NA_character_)
}

# 4. Generate the Map
station_map <- targets %>%
  mutate(Actual_Station_Name = map_chr(original, ~find_beaches_stations(.x, dataset_stations))

# Check the results
print(station_map)

```

```
# A tibble: 14 x 2
```

	original	Actual_Station_Name
	<chr>	<chr>
1	EMDAABIIMOK AVE / LAKE SHORE BLVD E	<NA>
2	ORCHARD PARK	Orchard Park
3	1651 QUEEN	<NA>
4	QUEEN ST. E / EASTERN AVE	Queen St. E / Eastern Ave
5	NORTHERN DANCER BLVD / LAKE SHORE BLVD E	Northern Dancer Blvd / Lake Shore B~

6 QUEEN ST E / JOSEPH DUGGAN RD
 7 WOODBINE AVE / LAKE SHORE BLVD E
 8 KEWBEACH AVE / KENILWORTH AVE
 9 85 LEE AVE
 10 HUBBARD BLVD / GLEN MANOR DR
 11 HUBBARD BLVD / BALSAM AV
 12 QUEEN ST E / HAMMERSMITH AVE
 13 QUEEN ST. E / SPRUCE HILL RD.
 14 QUEEN ST E / NURSEWOOD RD

Queen St E / Joseph Duggan Rd
 Woodbine Ave / Lake Shore Blvd E
 Kewbeach Ave / Kenilworth Ave
 85 Lee Ave
 Hubbard Blvd / Glen Manor Dr
 Hubbard Blvd / Balsam Av
 Queen St E / Hammersmith Ave
 Queen St. E / Spruce Hill Rd.
 Queen St E / Nursewood Rd

BIKESHARE JSON (FILES NOT FOUND)

```
bikeshare <- list_package_resources("2b44db0d-eea9-442d-b038-79335368ad5a")

list_package_resources("2b44db0d-eea9-442d-b038-79335368ad5a")
```

```
# A tibble: 4 x 4
  name                                id    format last_modified
  <chr>                             <chr> <chr>    <date>
1 bike-share-json                   5c1c~ JSON    2019-07-23
2 bike-share-gbfs-general-bikeshare-feed-specificati~ b698~ JSON    2019-07-23
3 gbfs-documentation                2eb3~ WEB      2019-07-23
4 bike-share-stations-readme        5ee3~ TXT      2019-07-23
```

```
get_resource("b69873a1-c180-4ccd-a970-514e434b4971")
```

```
$last_updated
# A tibble: 1 x 1
  value
  <int>
1 1563373889
```

```
$ttl
# A tibble: 1 x 1
  value
  <int>
1    19
```

```
$data
```

```

$data$en
# A tibble: 5 x 1
  feeds$name      $url
  <chr>          <chr>
1 system regions https://tor.publicbikesystem.net/ube/gbfs/v1/en/system_r~
2 system_information https://tor.publicbikesystem.net/ube/gbfs/v1/en/system_i~
3 station_information https://tor.publicbikesystem.net/ube/gbfs/v1/en/station_~
4 station_status    https://tor.publicbikesystem.net/ube/gbfs/v1/en/station_~
5 system_pricing_plans https://tor.publicbikesystem.net/ube/gbfs/v1/en/system_p~

bikeshare_json <- get_resource("5c1c2c06-d27f-47b7-ae82-926a6d23d76f")

```