main

nwfried

2025-01-21

Load libraries

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.10.0, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(tigris)
```

```
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
library(knitr)
options(tigris_use_cache = TRUE)
```

```
filepath <- "~/Documents/congestion_pricing/mapbox_drive_times/data"
```

Initialise variables for congestion zone

```
cong_zips <-c("10036", "10038", "10280", "10282")
for (i in 10001:10022) {
  cong_zips <- append(cong_zips, as.character(i))
}
```

Pull GIS info for Manhattan

```
newyork <- tigris::counties(state = "NY", class = "sf") %>%
  st_transform(crs = "WGS84")
```

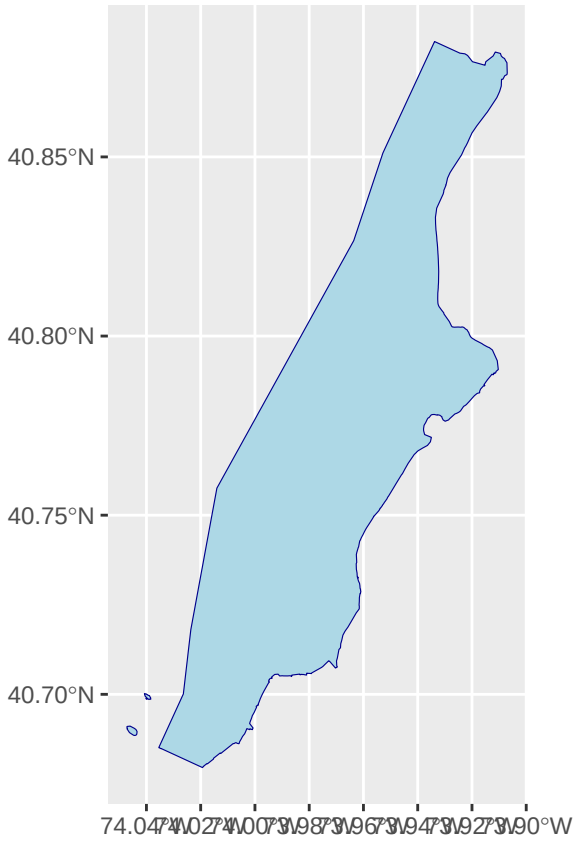
```
## Retrieving data for the year 2022
```

```
manhattan <- newyork %>% filter(NAME == "New York") %>%
  st_transform(crs = "+proj=longlat +datum=WGS84")
brooklyn <- newyork %>% filter(NAME == "Kings") %>%
  st_transform(crs = "WGS84")
bronx <- newyork %>% filter(NAME == "Bronx") %>%
  st_transform(crs = "WGS84")
staten_island <- newyork %>% filter(NAME == "Richmond") %>%
  st_transform(crs = "WGS84")
queens <- newyork %>% filter(NAME == "Queens") %>%
  st_transform(crs = "WGS84") #probably a better way to just pull in WGS84 coords lol
nyc <- newyork %>% filter(NAME == "New York" | NAME == "Kings" | NAME == "Bronx" | NAME == "Richmond" |
manhattan_planar <- manhattan %>% st_transform(32618)
```

Pull GIS info for congestion zip codes

```
cong_zone <- tigris::zctas(year = "2020", class = "sf")
cong_zone <- cong_zone %>% filter(ZCTA5CE20 %in% cong_zips) %>% st_transform(crs = "WGS84")
```

```
ggplot() + geom_sf(data = manhattan, fill = "lightblue", color = "darkblue")
```




```
make_sf <- function(df){
  df<- df %>% filter(duration!= 0) %>% left_join(jan15_less,by = c("orig_long", "orig_lat", "dest_long"
  rowwise() %>%
```

```
  filter(duration_min != 0, coordinates.x != "{}", coordinates.y != "{}") %>%
  mutate(
    geometry = list(
      st_linestring(
        as.matrix(fromJSON(coordinates))
      )
    )
  ) %>%
  st_as_sf(crs = 4326)
  return(df)
}
```

```
analyse_sf <- function(sf){
  sf <- sf %>% mutate(
  cong = sapply(st_intersects(geometry, cong_zone), function(x) length(x) > 0),
  manhattan = sapply(st_intersects(geometry, manhattan), function(x) length(x) > 0),
  brooklyn = sapply(st_intersects(geometry, brooklyn), function(x) length(x) > 0),
  queens = sapply(st_intersects(geometry, queens), function(x) length(x) > 0),
  bronx = sapply(st_intersects(geometry, bronx), function(x) length(x) > 0),
  staten_island = sapply(st_intersects(geometry, staten_island), function(x) length(x) > 0),
  nyc = sapply(st_intersects(geometry, nyc), function(x) length(x) > 0))%>%
  st_transform(32618) %>% mutate(
  crosses_manhattan = lengths(st_crosses(geometry, manhattan_planar)) > 0
)
  return(sf)
}
```

```
jan15 <- read_delim(file.path(filepath, "mapbox_output_2025-01-15_HH08.csv"))
```

```
## New names:
## Rows: 2131 Columns: 22
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (6): weight_name, coordinates, type, waypoints, code, uuid dbl (15): ...1, X,
## inputzip, zip_commute, orig_long, orig_lat, dest_long, d... dttm (1): timestamp
## 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`
```

```
jan15_sf <- jan15 %>% rowwise() %>%
  filter(duration_min != 0) %>%
  mutate(
    geometry = list(
      st_linestring(
        as.matrix(fromJSON(coordinates))
      )
    )
  ) %>%
  st_as_sf(crs = 4326)
```

turn this into function

```
jan15_less <- jan15 %>% select(orig_long, orig_lat, dest_long, dest_lat, coordinates) %>% rowwise() %>%
  filter(coordinates != '{}') %>%
  mutate(
    geometry = list(
```

```
      st_linestring(
        as.matrix(fromJSON(coordinates))
      )
    )
  ) %>%
  st_as_sf(crs = 4326)
jan15_less <- analyse_sf(jan15_less)
jan15_less <- jan15_less %>% st_drop_geometry()
```

```
cong_info <- function(df) {
  mean <- df %>% st_drop_geometry %>%
    ungroup() %>%
    filter(cong == TRUE)%>%
    summarise(mean_duration = mean(duration_min), median = median(duration_min), n = n())
  return(mean)
}
```

```
trip_info <- function(df) {
  mean <- df%>%st_drop_geometry%>%
    ungroup()%>%
  summarise(mean_duration = mean(duration_min), median = median(duration_min), n = n())
  return(mean)
}
control_info <- function(df) {
  mean <- df%>%st_drop_geometry%>% ungroup()%>% filter(nyc == FALSE) %>%
    summarise(mean_duration = mean(duration_min), median = median(duration_min), n = n())
  return(mean)
}
```

```
extract_date_hour <- function(file_name) {
  match <- str_match(file_name, "mapbox_output_(\\d{4}-\\d{2}-\\d{2})_HH(\\d{2})")
  list(
    date = match[2],  # Extracted date (YYYY-MM-DD)
    hour = match[3]   # Extracted hour (HH)
  )
}
```

```
# Read all CSV files in a folder
process_csv_files <- function(folder_path) {
  # List all CSV files in the folder
  csv_files <- list.files(path = folder_path, pattern = "\\.csv$", full.names = TRUE)
  # Empty dataframe to store results
 results_df <- data.frame(
    date = character(),
    hour = character(),
    cong_mean = numeric(),
    trip_mean = numeric(),
    control_mean = numeric(),
    stringsAsFactors = FALSE
  )

  # Loop over all files
  for (file in csv_files) {
    extracted <- extract_date_hour(basename(file))
    date_hour <- paste0(extracted$date, "_", extracted$hour)  # Combine date and hour for unique naming
```

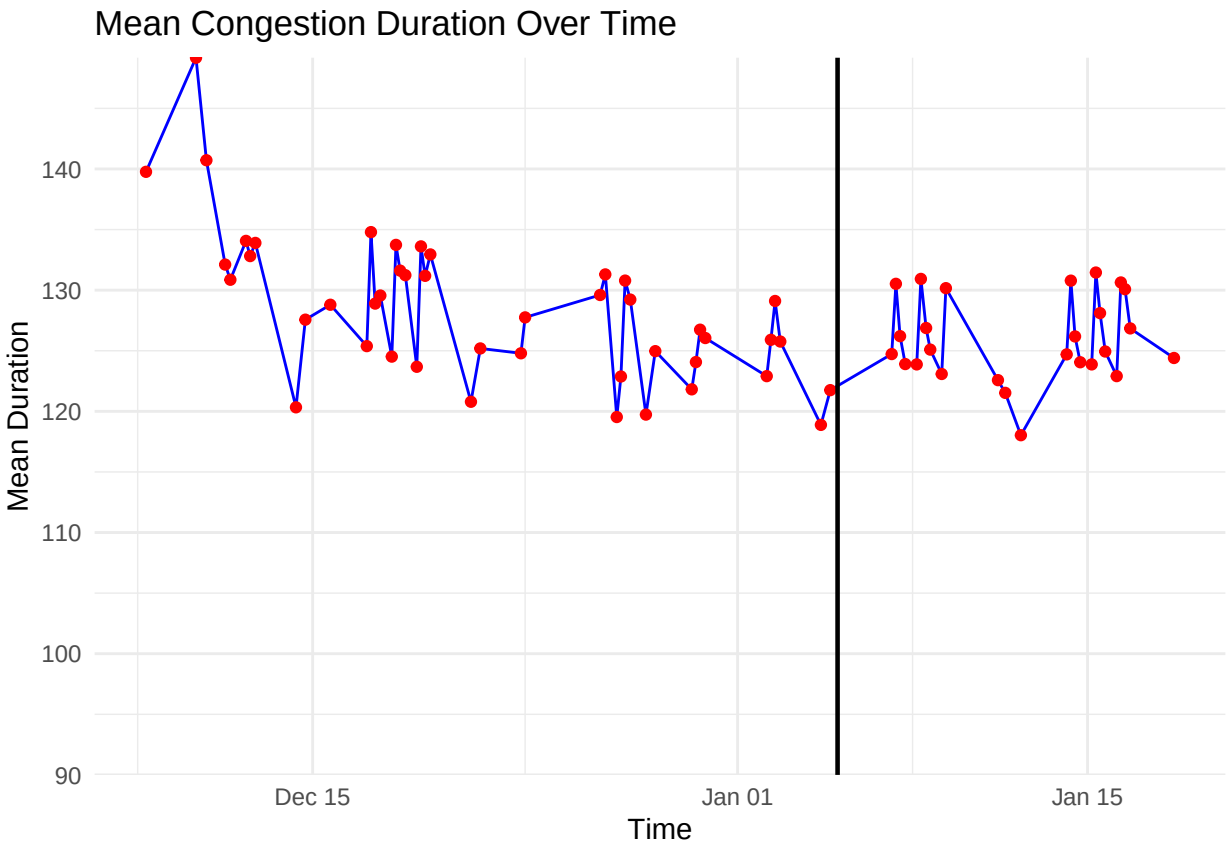
```
    # Read the CSV file
    data <- read.csv(file)

    # Apply the functions
    sf_data <- left_join(data, jan15_less, by = c("orig_long", "orig_lat", "dest_long", "dest_lat"), rel
    congestion_result <- cong_info(sf_data)
    trip_result <- trip_info(sf_data)
    control_result <- control_info(sf_data)
    # Store the result in results dataframe
   results_df <- rbind(
      results_df,
      data.frame(
        date = extracted$date,
        hour = extracted$hour,
        cong_mean = congestion_result,
        trip_mean = trip_result,
        control_mean = control_result,
        stringsAsFactors = FALSE
      )
    )

  }

  return(results_df)
}
```

```
results <- process_csv_files(filepath)
```

Mean trip duration with the congestion zone against time; line indicates date congestion relief begins.

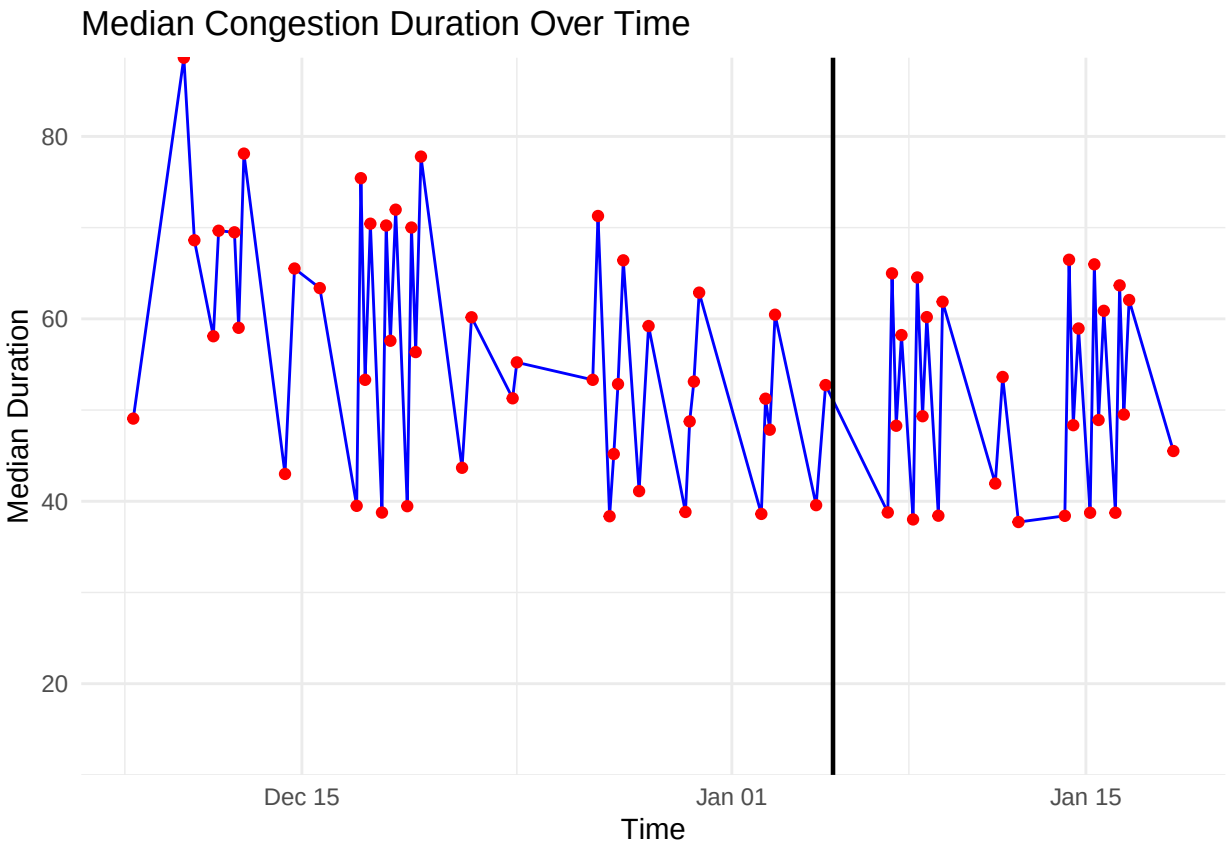
```
ggplot(results, aes(x = as.POSIXct(paste(date, hour), format = "%Y-%m-%d %H"), y = cong_mean.mean_durati
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.POSIXct("2025-01-05 00:00", format = "%Y-%m-%d %H"), linetype = "solid", co
  labs(
    x = "Time",
    y = "Mean Duration",
    title = "Mean Congestion Duration Over Time"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(90, NA)) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



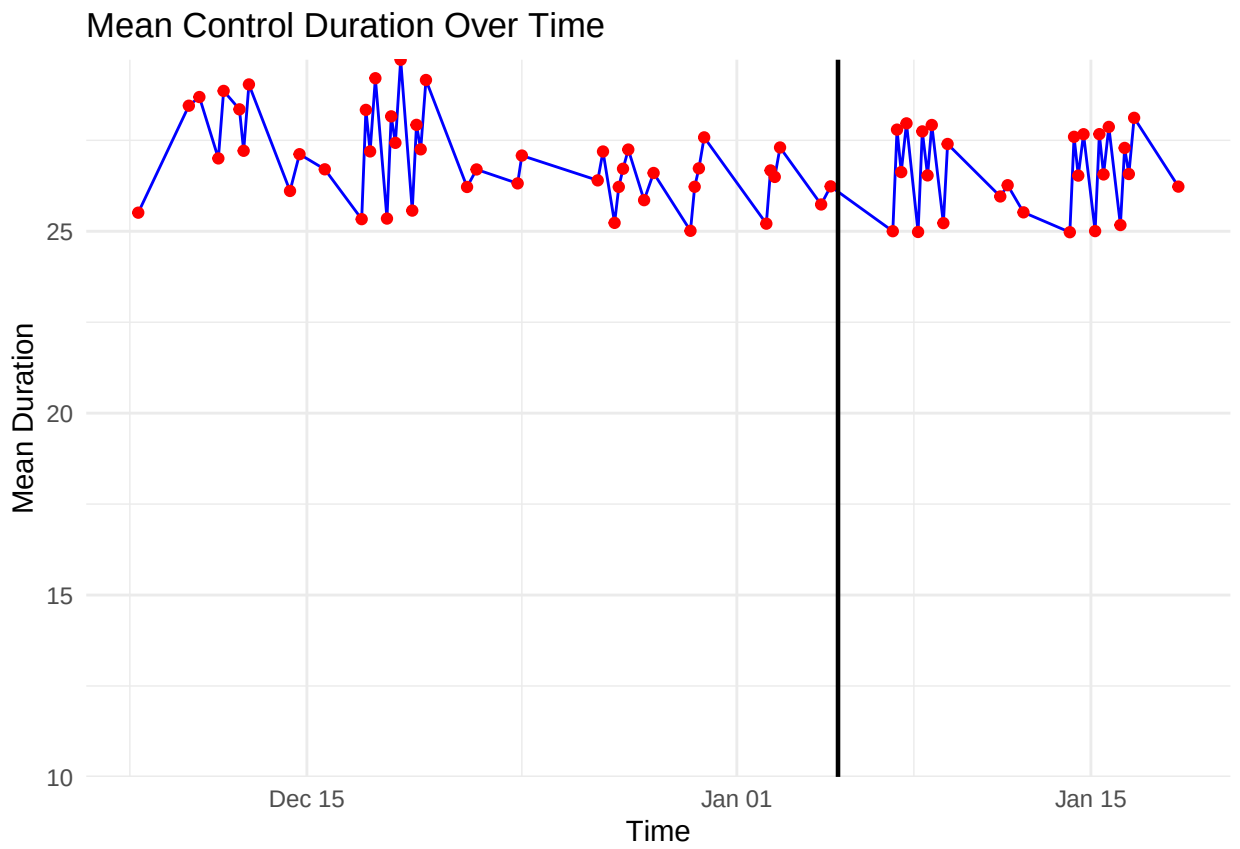
Median trip duration with the congestion zone against time; line indicates date congestion relief begins.

```
ggplot(results, aes(x = as.POSIXct(paste(date, hour), format = "%Y-%m-%d %H"), y = cong_mean.median)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.POSIXct("2025-01-05 00:00", format = "%Y-%m-%d %H"), linetype = "solid", c
  labs(
    x = "Time",
    y = "Median Duration",
    title = "Median Congestion Duration Over Time"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(10, NA)) +
  theme_minimal()
```

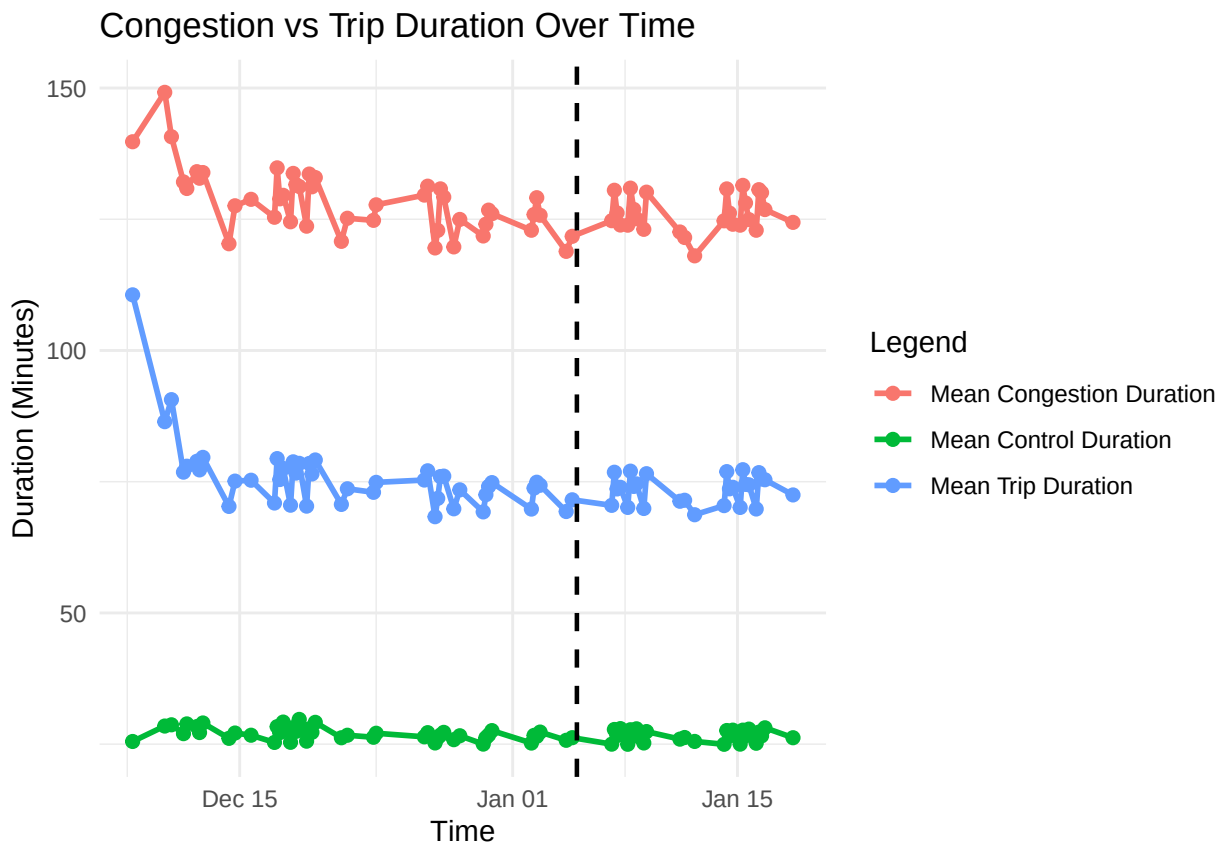


Mean control trip duration over time (trips outside of NYC):

```
ggplot(results, aes(x = as.POSIXct(paste(date, hour), format = "%Y-%m-%d %H"), y = control_mean.mean_du
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.POSIXct("2025-01-05 00:00", format = "%Y-%m-%d %H"), linetype = "solid", c
  labs(
    x = "Time",
    y = "Mean Duration",
    title = "Mean Control Duration Over Time"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(10, NA)) +
  theme_minimal()
```




```
results$datetime <- as.POSIXct(paste(results$date, results$hour), format = "%Y-%m-%d %H")
ggplot(results, aes(x = datetime)) +
  geom_line(aes(y = cong_mean.mean_duration, color = "Mean Congestion Duration"), size = 1) +
  geom_line(aes(y = trip_mean.mean_duration, color = "Mean Trip Duration"), size = 1) +
  geom_point(aes(y = cong_mean.mean_duration, color = "Mean Congestion Duration"), size = 2) +
  geom_point(aes(y = trip_mean.mean_duration, color = "Mean Trip Duration"), size = 2) +
  geom_line(aes(y = control_mean.mean_duration, color = "Mean Control Duration"), size = 1) +
  geom_point(aes(y = control_mean.mean_duration, color = "Mean Control Duration"), size = 2) +
  geom_vline(
    xintercept = as.POSIXct("2025-01-05 00:00", format = "%Y-%m-%d %H"),
    linetype = "dashed", color = "black", size = 0.8
  ) +
  labs(
    x = "Time",
    y = "Duration (Minutes)",
    title = "Congestion vs Trip Duration Over Time",
    color = "Legend"
  ) +
  theme_minimal()
```

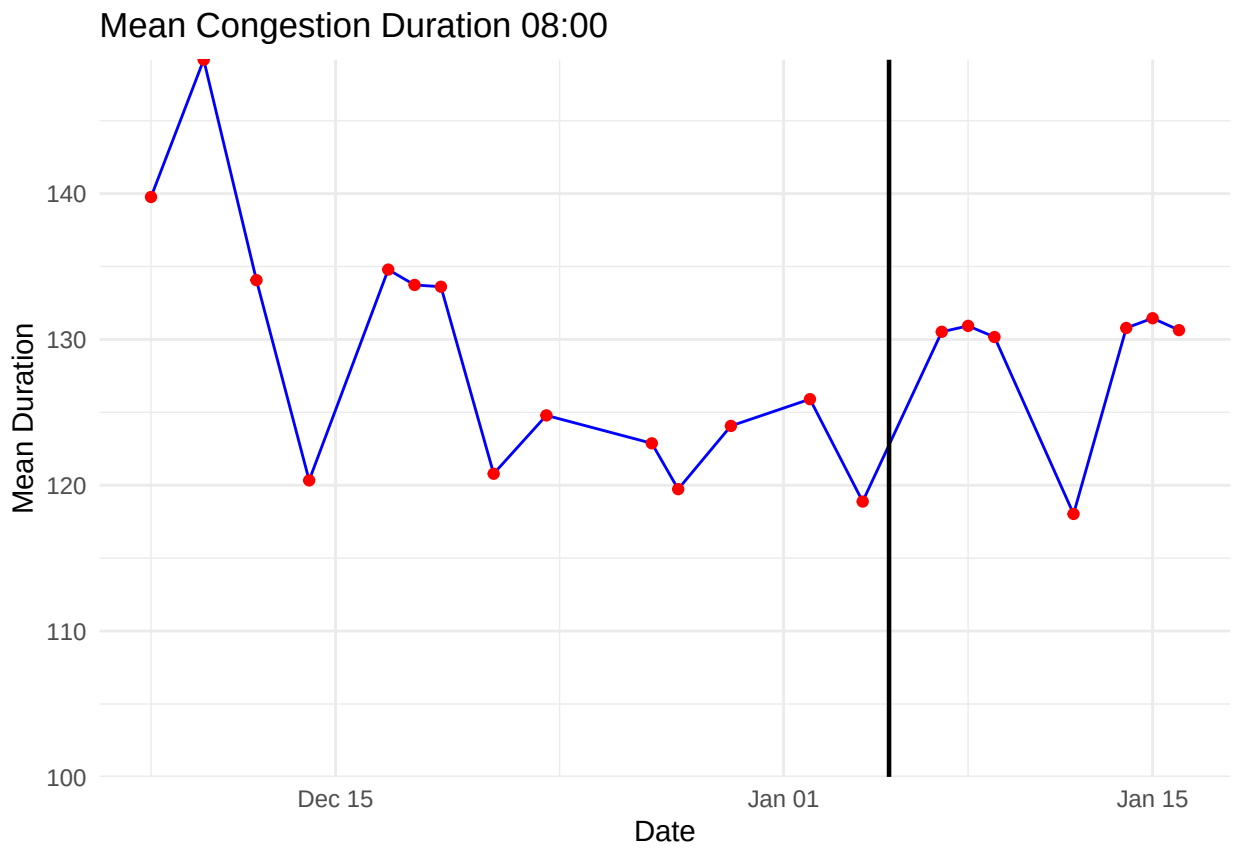


Filter by hour.

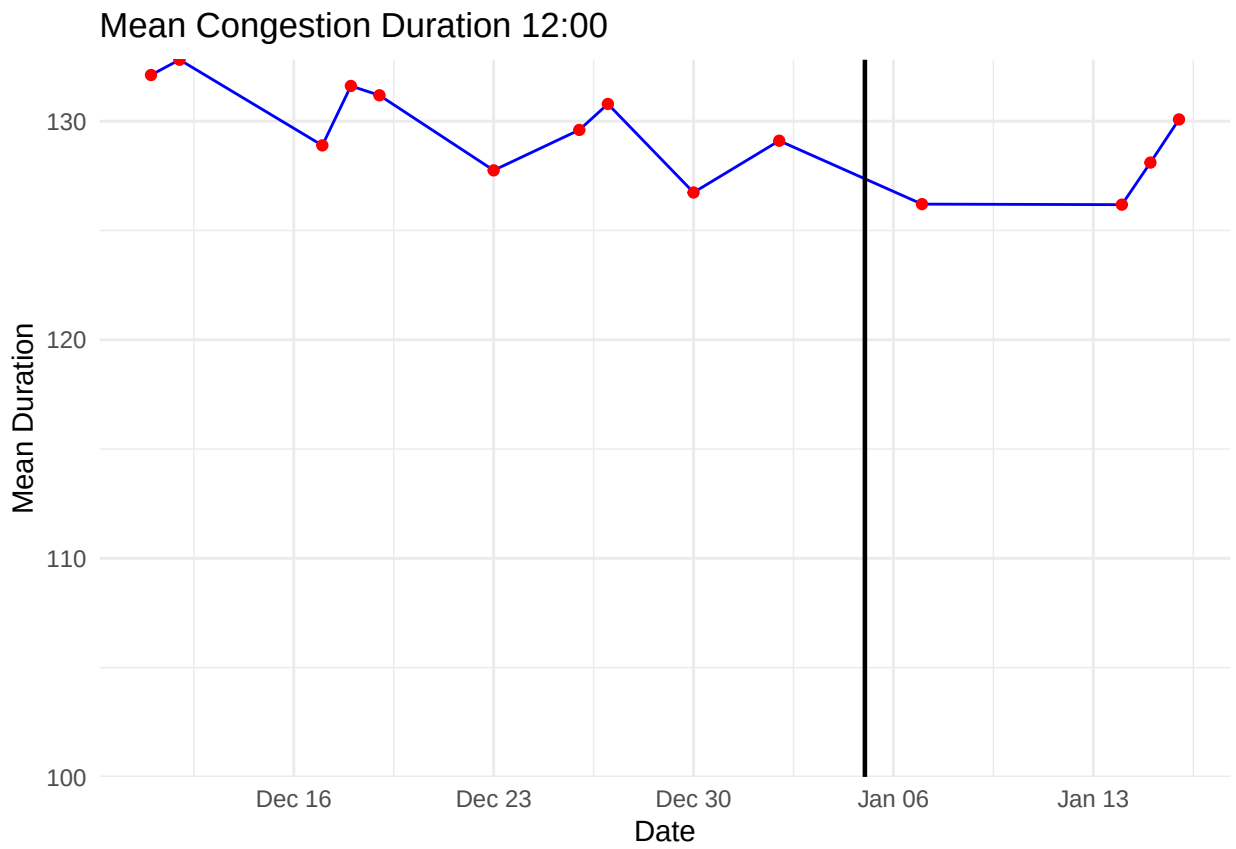
```
results_08 <- results %>% filter(hour == '08') %>% select(-hour)
results_12 <- results %>% filter(hour == '12') %>% select(-hour)
results_17 <- results %>% filter(hour == '17') %>% select(-hour)
```

Plot:

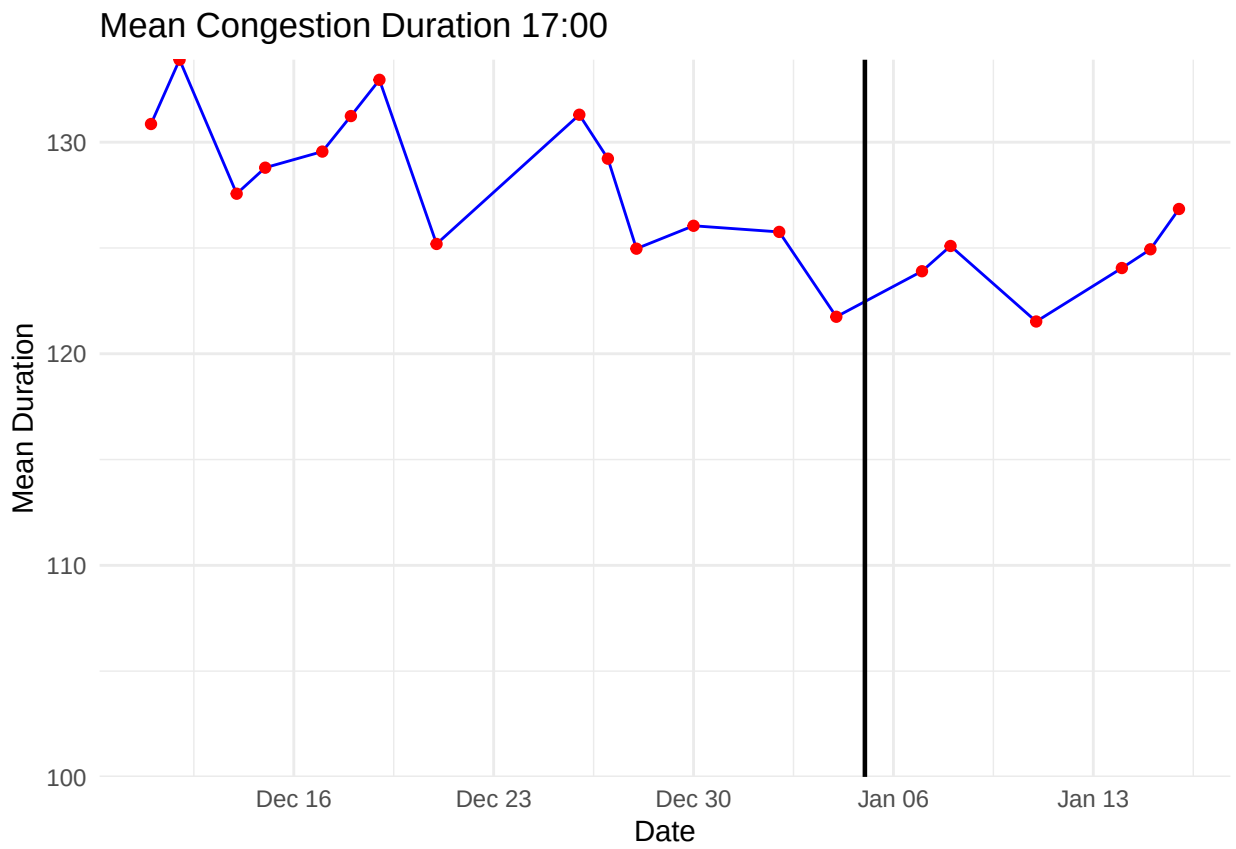
```
ggplot(results_08, aes(x = as.Date(date), y = cong_mean.mean_duration)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.Date("2025-01-05"), linetype = "solid", color = "black", size = 0.8) +
  labs(
    x = "Date",
    y = "Mean Duration",
    title = "Mean Congestion Duration 08:00"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(100, NA)) +
  theme_minimal()
```




```
ggplot(results_12, aes(x = as.Date(date), y = cong_mean.mean_duration)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.Date("2025-01-05"), linetype = "solid", color = "black", size = 0.8) +
  labs(
    x = "Date",
    y = "Mean Duration",
    title = "Mean Congestion Duration 12:00"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(100, NA)) +  # Ensure the y-axis starts at 100
  theme_minimal()
```




```
ggplot(results_17, aes(x = as.Date(date), y = cong_mean.mean_duration)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  geom_vline(xintercept = as.Date("2025-01-05"), linetype = "solid", color = "black", size = 0.8) +
  labs(
    x = "Date",
    y = "Mean Duration",
    title = "Mean Congestion Duration 17:00"
  ) + scale_y_continuous(expand = c(0, 0), limits = c(100, NA)) +  # Ensure the y-axis starts at 100
  theme_minimal()
```



Summary statistics:

```
cong_stats <- results[-c(1:45), ] %>%
  summarise(mean_cong = mean(cong_mean.mean_duration), mean_trip = mean(trip_mean.mean_duration), media
before_cong <- results[1:45, ] %>%
  summarise(mean_cong = mean(cong_mean.mean_duration), mean_trip = mean(trip_mean.mean_duration), media
```

```
cong_stats
```

```
##   mean_cong mean_trip median_cong median_trip mean_control
## 1  126.0175    73.449    49.41452    31.11211     26.62304
```

```
before_cong
```

```
##   mean_cong mean_trip median_cong median_trip mean_control
## 1   128.461  75.89977     56.3552    33.81175      26.9924
```