Supplemental Materials

Elizabeth Hora

12/6/2020

Abstract

This file contains all of the extra work that would have cluttered the main project, but is too important not to mention at all.

Loading Packages

```
library(tidyverse)
library(ggplot2)
library(readxl)
library(janitor)
library(naniar)
library(visdat)
library(lubridate)
library(stringr)
library(gganimate)
library(lvplot)
```

Loading up Both Data Sets

Concord Rotary

```
# Step 1
files <- dir("../data/elm_street/", pattern = "\\.xlsx$", full.names = TRUE)
file_list <- list()

# Step 2
times <- as.character(0:23)
for (i in files){
  file_list[[i]] <- read_excel(i, skip = 10, col_names = c("day", times, "Total", "QC Status"))
%>%
    pivot_longer(c(`0`,`1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, `12`, `13`, `14
`, `15`, `16`, `17`, `18`, `19`, `20`, `21`, `22`, `23`),
                 names_to = "hour", values_to = "cars_per_hour") %>%
    select("day", "hour", "cars_per_hour")
}

# Step 3
months <- str_extract(names(file_list), "_403_[A-Z]B_\\d+") %>%
  str_sub(9) %>%
  as.double()

years <- str_extract(names(file_list), "\\d{4}") %>%
  as.double()

travel_direction <- str_extract(names(file_list), "_403_[A-Z]B") %>%
  str_sub(6)

# Step 4
for (i in 1:length(files)){
  file_list[[i]] <- file_list[[i]] %>%
    mutate(hour = as.numeric(hour)) %>%
    mutate(full_date = make_datetime(year = years[i], month = months[i], day = day, hour = hou
r)) %>%
    mutate(day_of_week = wday(full_date, label = TRUE, abbr = FALSE)) %>%
    mutate(direction = travel_direction[i])
}

# Finalizing
elm_master <- bind_rows(file_list)
```

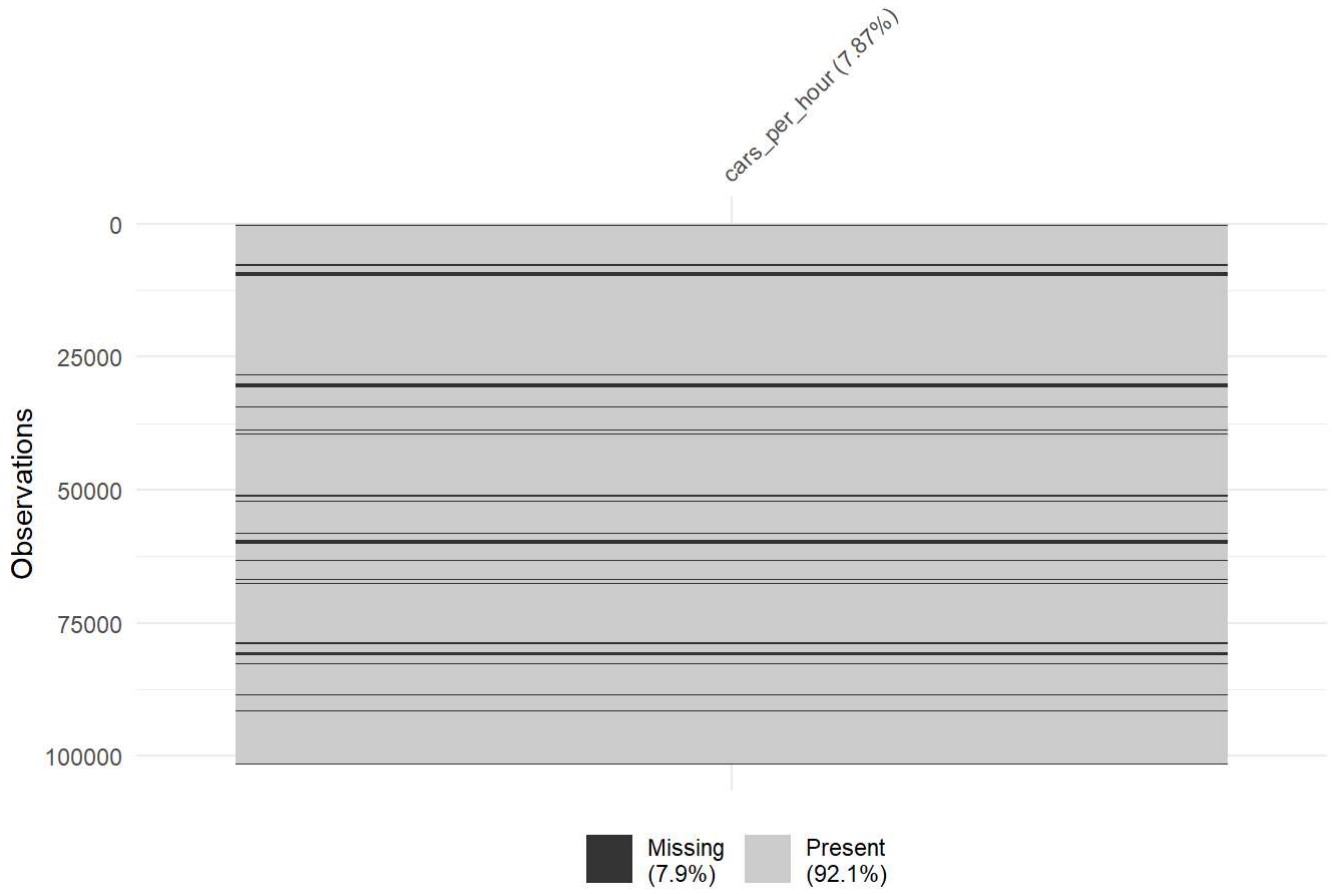
Mass Pike

```
# Step 1
paths_to_open <- c("../data/mass_pike/framingham",
                   "../data/mass_pike/cochituate",
                   "../data/mass_pike/newton",
                   "../data/mass_pike/pre_allston",
                   "../data/mass_pike/post_allston"
                   )
files <- dir(paths_to_open, pattern = "\\.xlsx$", full.names = TRUE)
pike_file_list <- list()

# Step 2
for (i in files){
  pike_file_list[[i]] <- read_excel(i, skip = 10, col_names = c("day", times, "Total", "QC Statu
s")) %>%
    pivot_longer(c(`0`,`1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, `12`, `13`, `14
`, `15`, `16`, `17`, `18`, `19`, `20`, `21`, `22`, `23`),
                 names_to = "hour", values_to = "cars_per_hour") %>%
    select("day", "hour", "cars_per_hour", "Total")
}

# Step 3
months <- str_extract(names(pike_file_list), "_[A-Z]B_\\d+") %>%
  str_sub(start = 5) %>%
  as.double()
years <- str_extract(names(pike_file_list), "\\d{4}") %>%
  as.double()
travel_direction <- str_extract(names(pike_file_list), "_[A-Z]B_") %>%
  str_sub(start = 2L, end = -2L)
location <- str_extract(names(pike_file_list), "/mass_pike/..") %>%
  str_sub(start = 12L) %>%
  as.character()

# Step 4
for (i in 1:length(files)){
  pike_file_list[[i]] <- pike_file_list[[i]] %>%
    mutate(hour = as.numeric(hour)) %>%
    mutate(full_date = make_datetime(year = years[i], month = months[i], day = day, hour = hou
r)) %>%
    mutate(day_of_week = wday(full_date, label = TRUE, abbr = FALSE)) %>%
    mutate(direction = travel_direction[i]) %>%
    mutate(location = location[i])
}

# Finalizing
pike_master <- bind_rows(pike_file_list) %>%
  mutate(daily_percent = cars_per_hour / Total * 100)
```

Missing Data

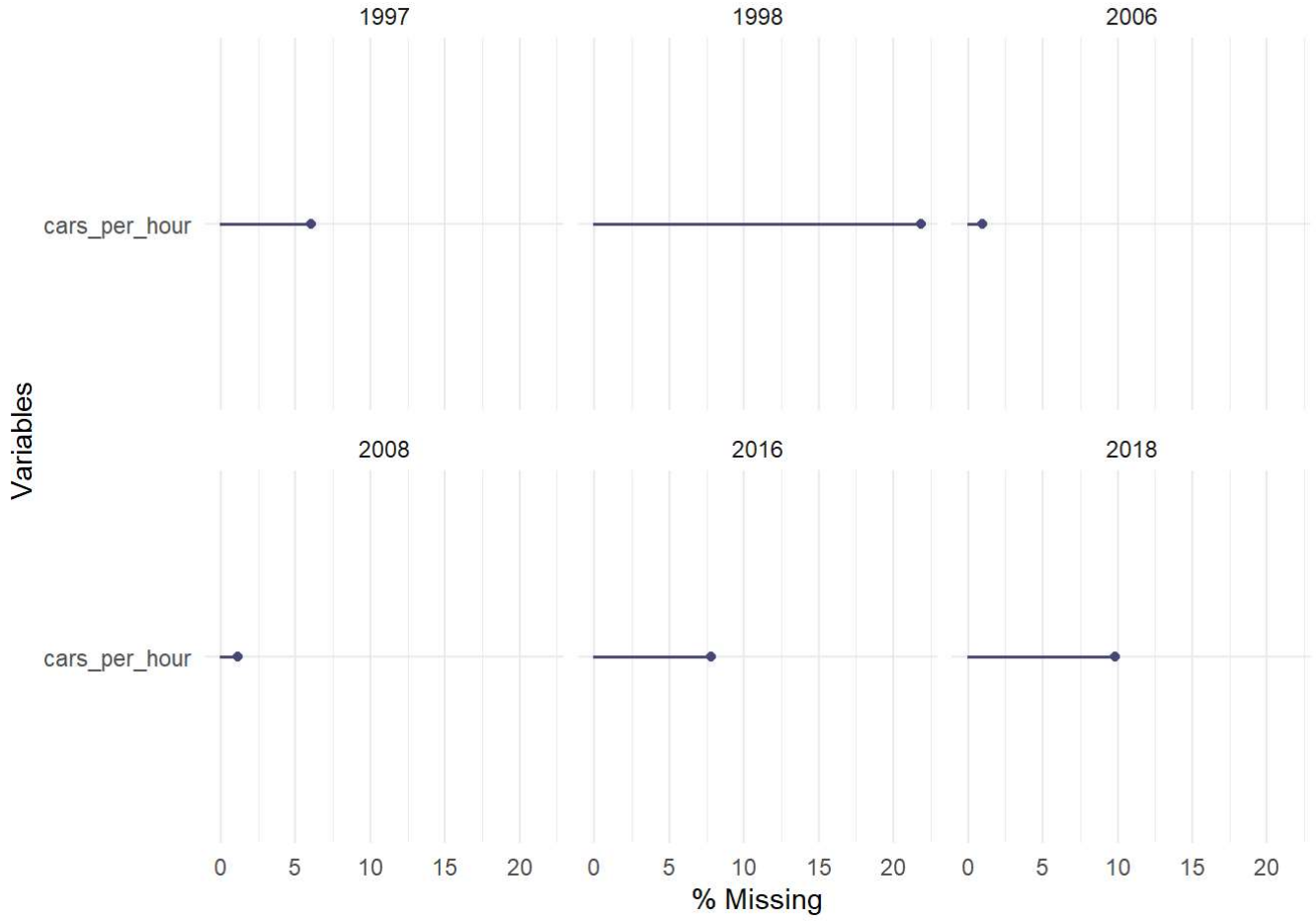
Concord Rotary

```
elm_master %>%
  mutate(year = year(full_date)) %>%
  select(cars_per_hour) %>%
  vis_miss()
```



Looking at the observations as a whole, only `7.9%` of the data are missing.

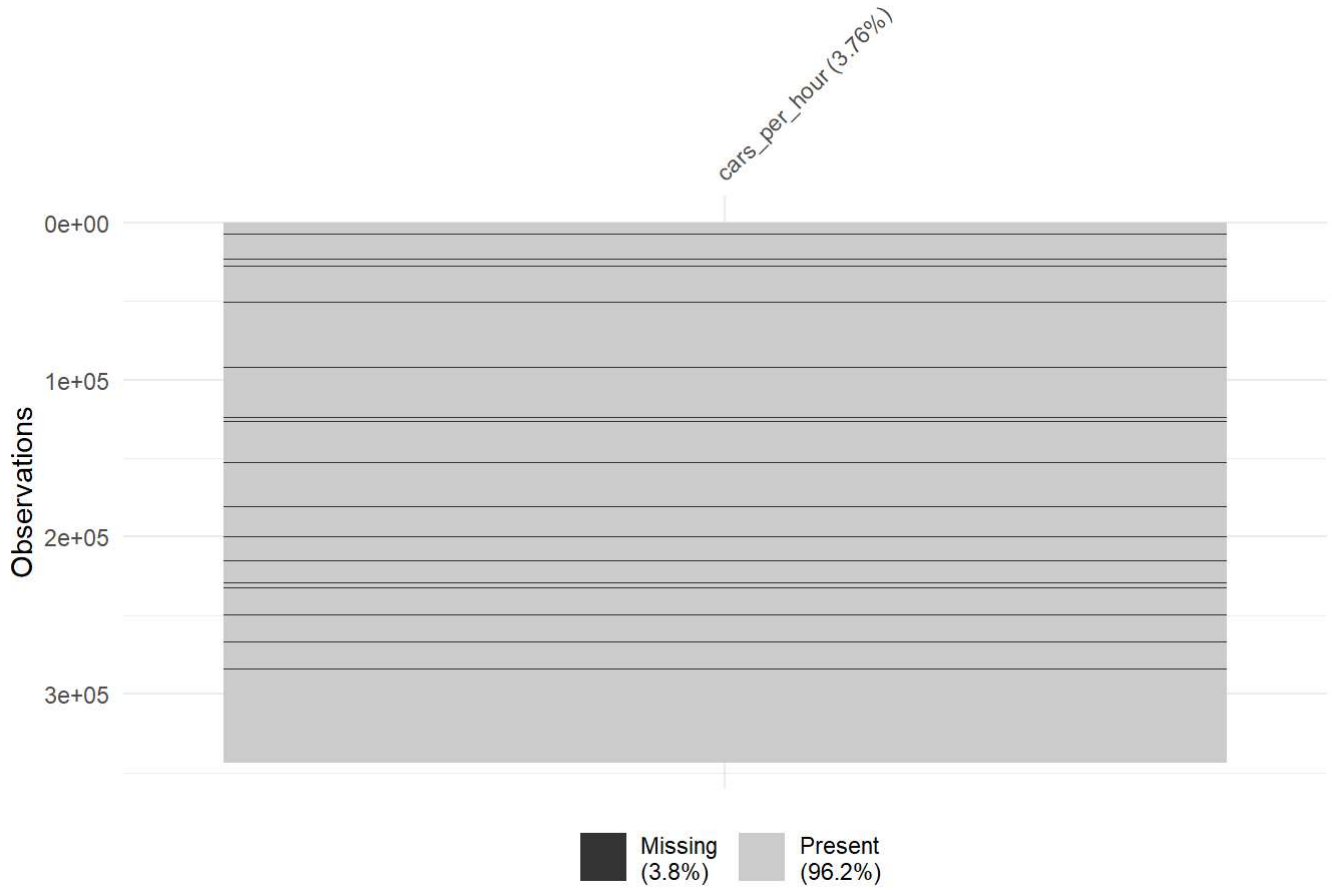
```
elm_master %>%
  mutate(month = month(full_date)) %>%
  mutate(year = year(full_date)) %>%
  select(cars_per_hour, year) %>%
  gg_miss_var(year, show_pct = TRUE)
```



Looking at each year individually, there does not seem to be any obvious patterns that would warrant hesitation when working with the data within `elm_master`. Although `1998` has the highest percentage of missing data with around `20%`, the rest of the years are fairly covered.

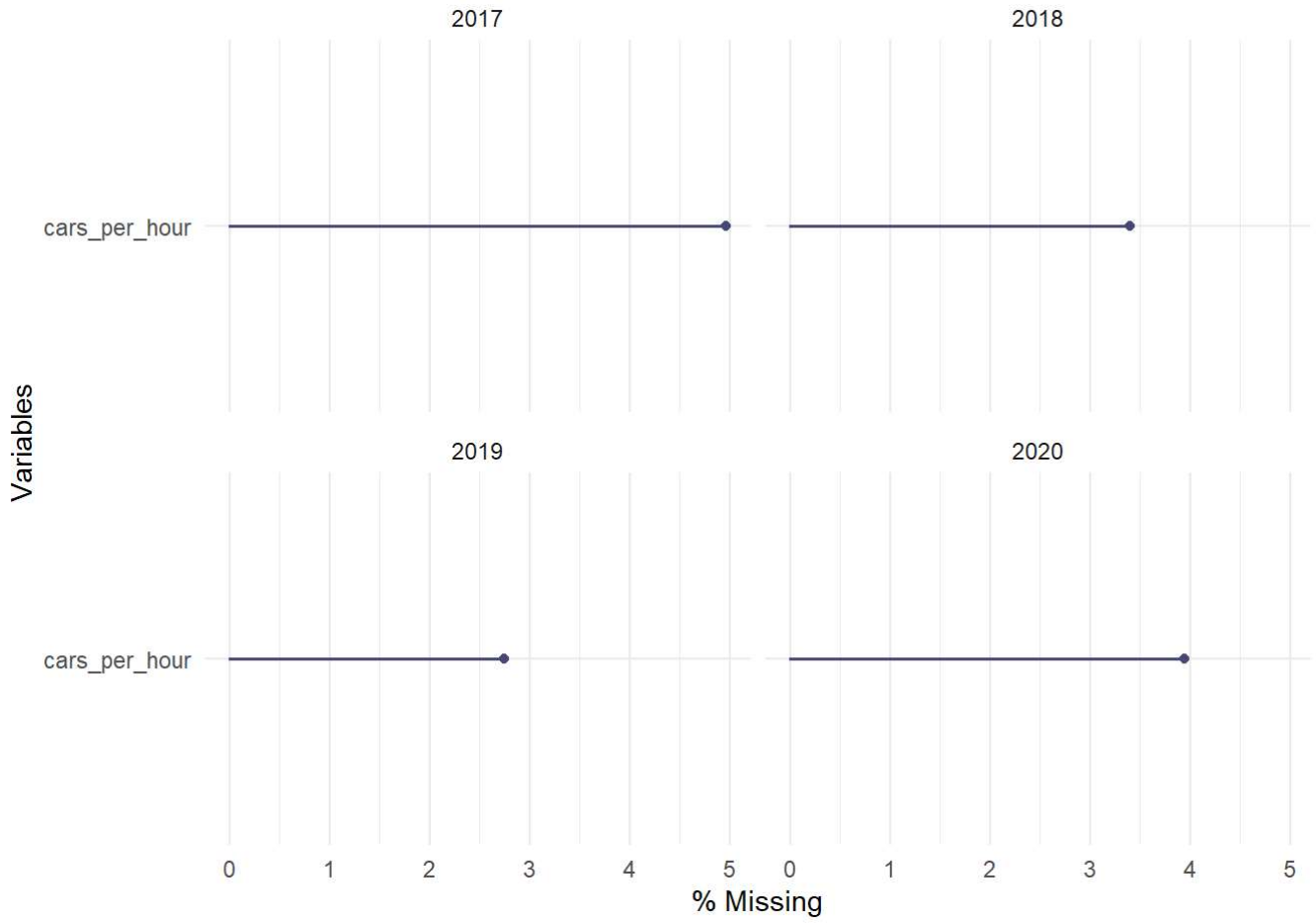
Mass Pike

```
pike_master %>%
  mutate(year = year(full_date)) %>%
  select(cars_per_hour) %>%
  vis_miss()
```



Looking at the observations as a whole, only `3.8%` of the data are missing.

```
pike_master %>%
  mutate(month = month(full_date)) %>%
  mutate(year = year(full_date)) %>%
  select(cars_per_hour, year) %>%
  gg_miss_var(year, show_pct = TRUE) %>%
  print
```

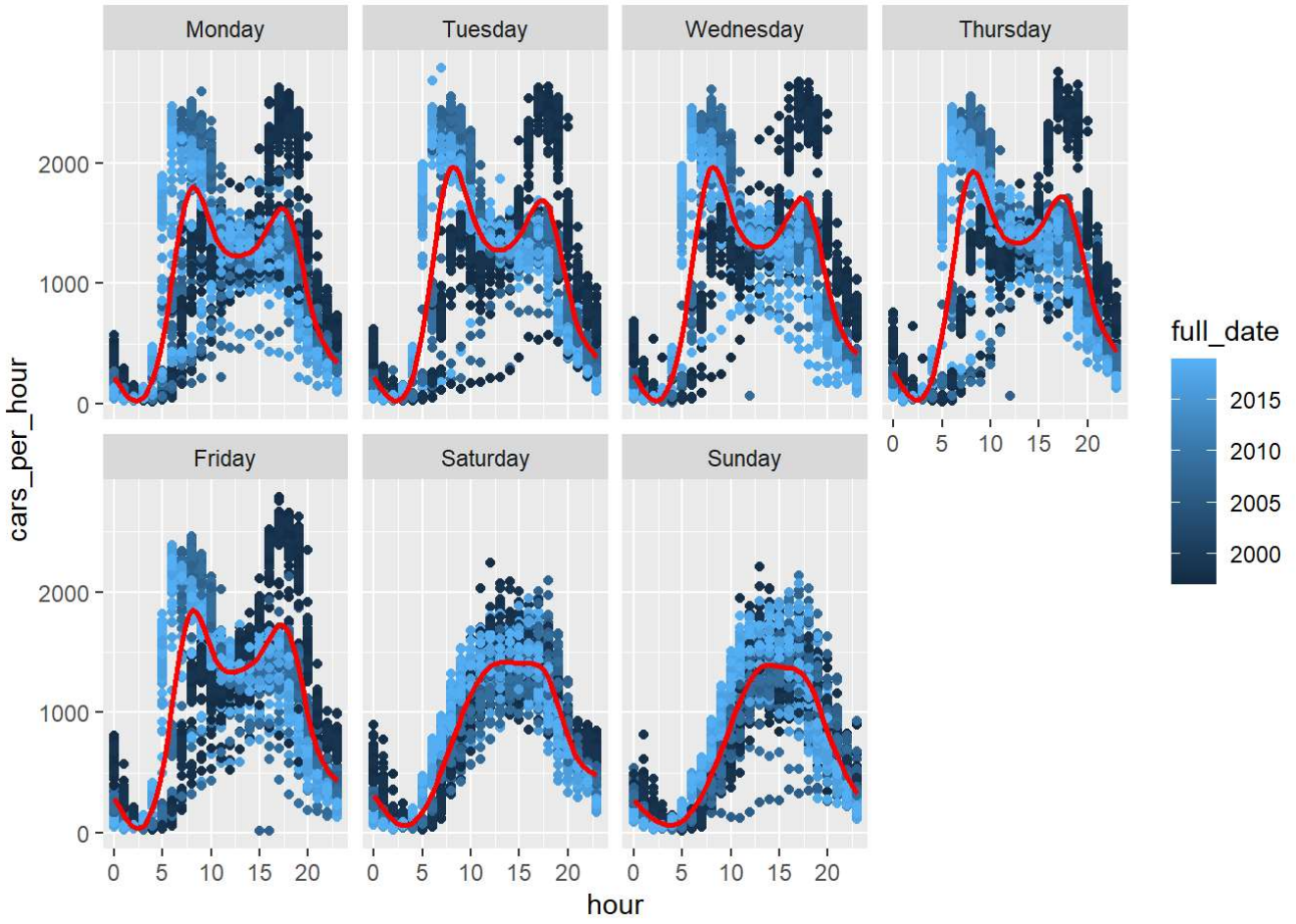


There is less than `5%` missing data in each of the four years selected from `2017-2020` . The data are reliable.

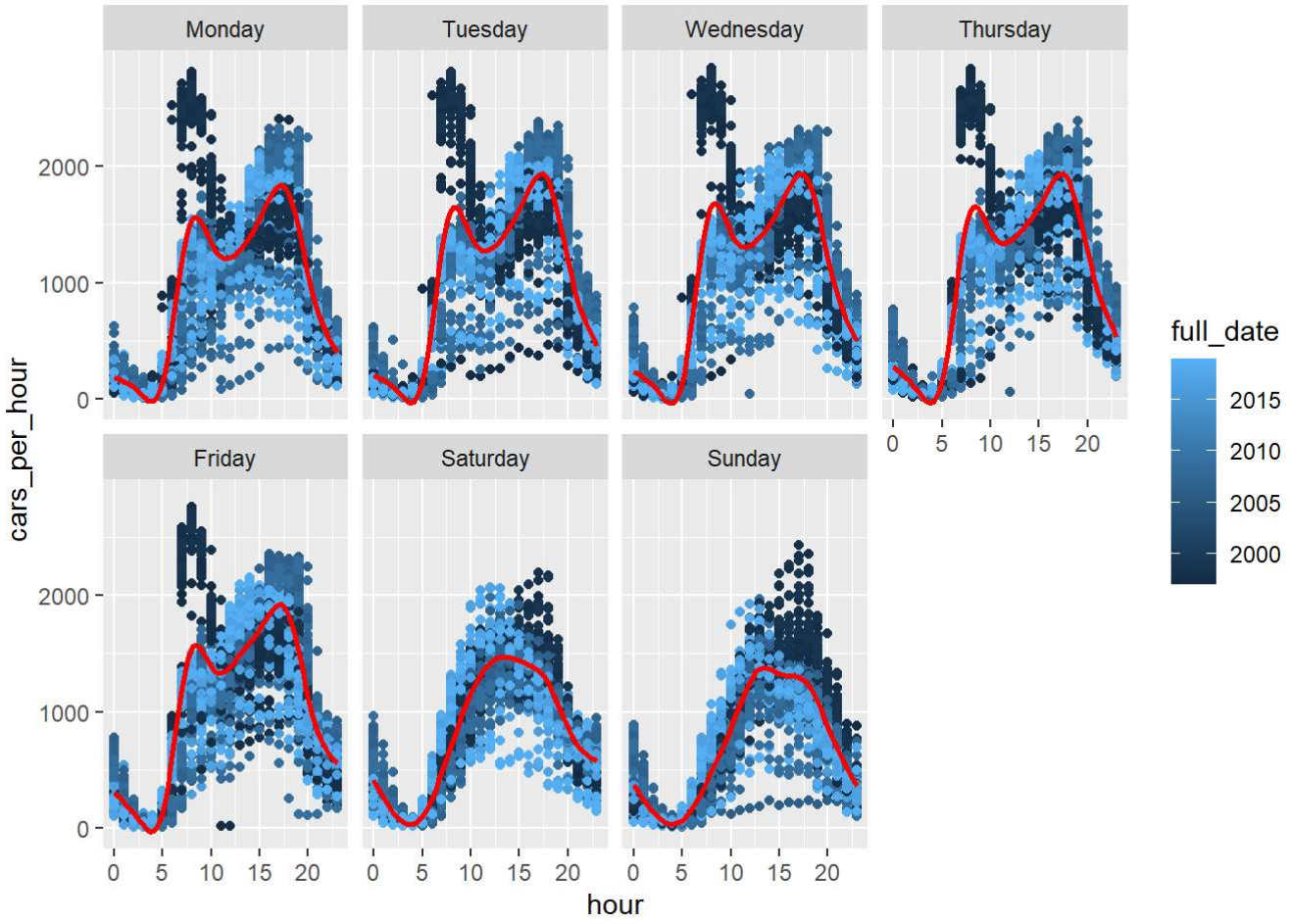
Weekends vs Weekdays

Concord Rotary

```
# Concord Rotary East Bound
elm_master_EB <- elm_master %>%
  filter(direction == "EB")

# Plotting the Data
elm_master_EB %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour, color = full_date)) +
  geom_point() +
  geom_smooth(se = FALSE, color = "RED") +
  facet_wrap(~day_of_week, nrow = 2)
```



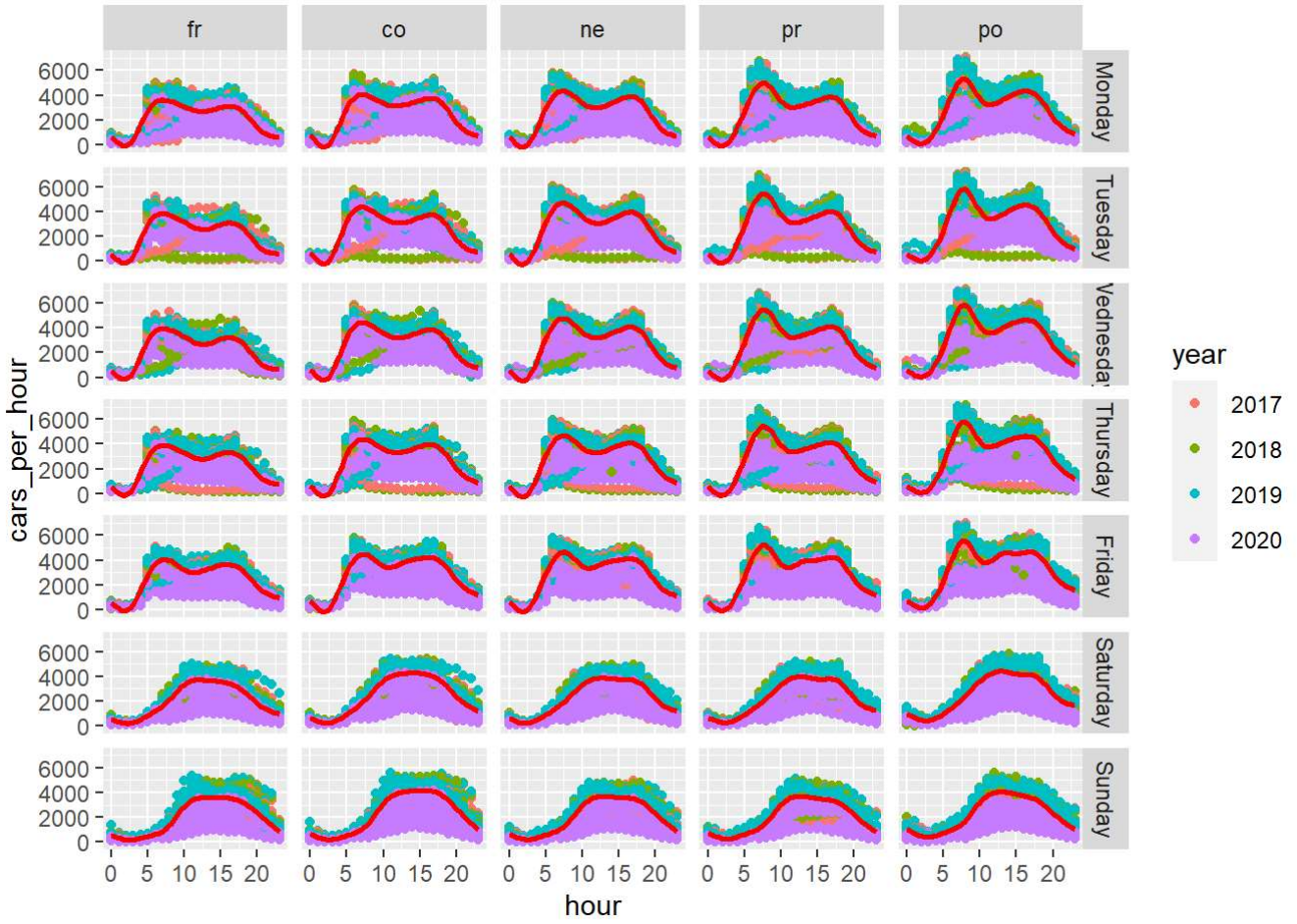

```
# Concord Rotary West Bound
elm_master_WB <- elm_master %>%
  filter(direction == "WB")

# Plotting the Data
elm_master_WB %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour, color = full_date)) +
  geom_point() +
  geom_smooth(se = FALSE, color = "RED") +
  facet_wrap(~day_of_week, nrow = 2)
```



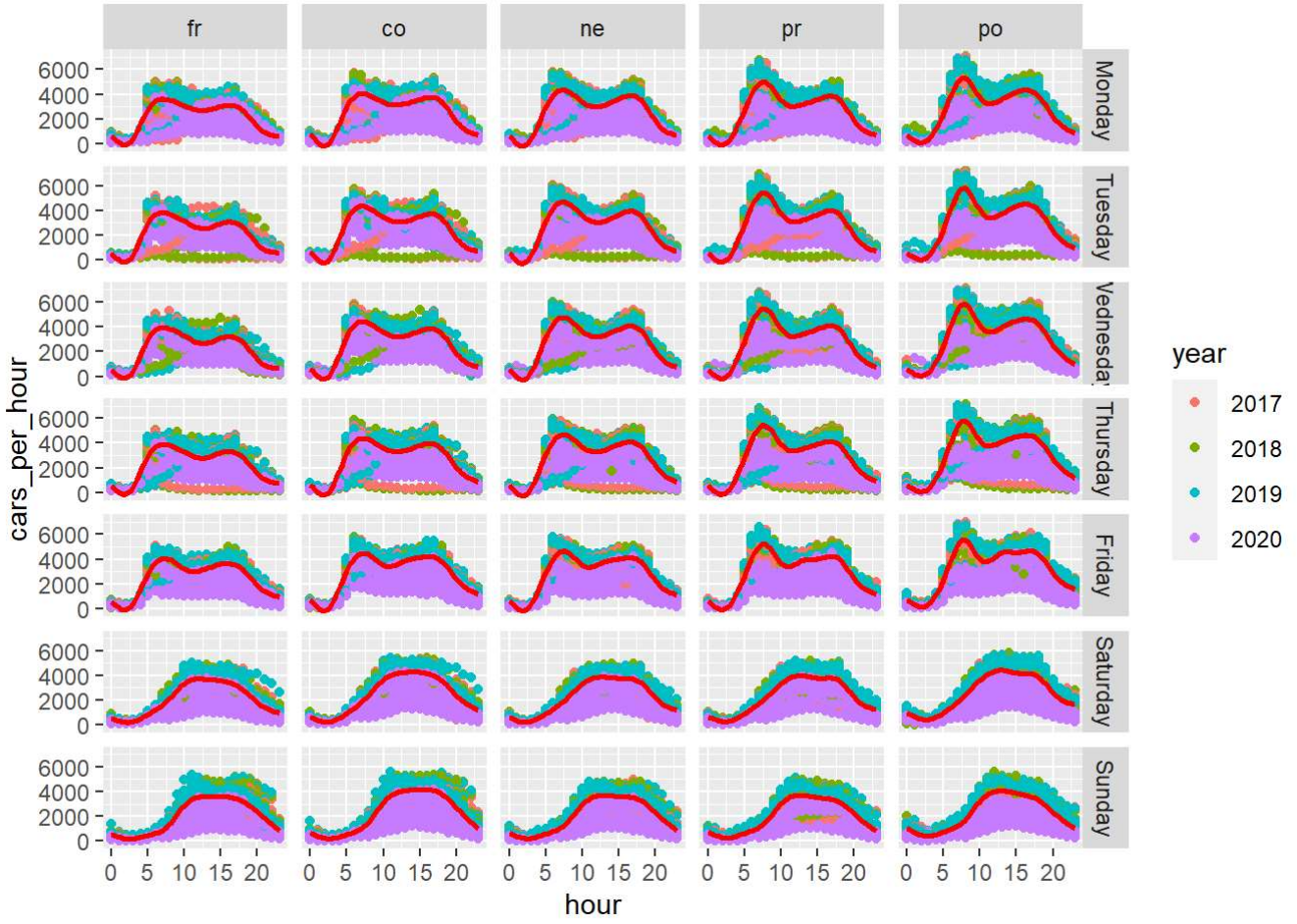
Looking at the resulting graphs at the Concord Rotary, it is clear that traffic patterns are different on the weekends than on the weekdays, regardless of year. There seems to be a single hump around noon time on the weekends, comparing to the two humps corresponding to rush hour on the weekdays. Interestingly enough, the peak weekend traffic looks to be similar in size as the valley in between the two humps for rush hour. Adding in a column for `year` and making it a factor will allow me to color code graphs without `ggplot` interpreting `full_date` as a continuous variable.

Mass Pike

```
# Mass Pike East Bound
pike_master_EB_with_weekends <- pike_master %>%
  drop_na() %>%
  filter(direction == "EB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po"))) %>%
  mutate(year = as.factor(year(full_date)))


# Plotting the Data
pike_master_EB_with_weekends %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))) %>%
  ggplot(aes(x = hour, y = cars_per_hour, color = year)) +
  geom_point() +
  geom_smooth(se = FALSE, color = "RED") +
  facet_grid(day_of_week~location)
```



```
# Mass Pike West Bound
pike_master_WB_with_weekends <- pike_master %>%
  drop_na() %>%
  filter(direction == "EB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po"))) %>%
  mutate(year = as.factor(year(full_date)))

# Plotting the Data
pike_master_EB_with_weekends %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))) %>%
  ggplot(aes(x = hour, y = cars_per_hour, color = year)) +
  geom_point() +
  geom_smooth(se = FALSE, color = "RED") +
  facet_grid(day_of_week~location)
```

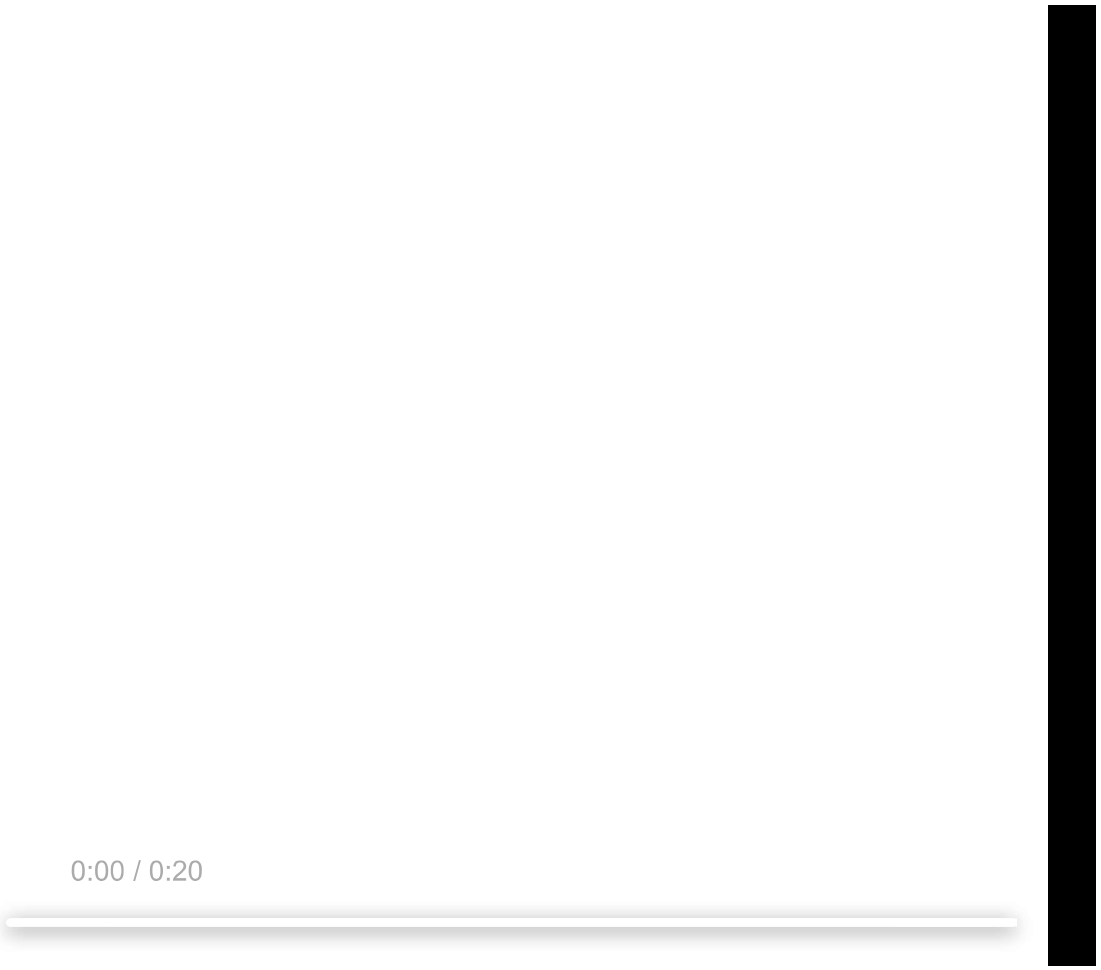


Again, looking at the resulting graphs at the several locations along the Mass Pike, it is clear that traffic patterns are different on the weekends than on the weekdays, regardless of year. There seems to be a single hump around noon time on the weekends, comparing to the two humps corresponding to rush hour on the weekdays. Interestingly enough, the peak weekend traffic looks to be similar in size as the valley in between the two humps for rush hour. Although there is no obvious pattern about which day is the worst, it is clear that in order to better understand commuters, I need to filter out the weekends.

gganimate

Since the data from before `2000` is likely a data entry error, I made animations without those years to see if the animations tell a different story.

Concord Rotary, East Bound, weekdays, post 2000:



Concord Rotary, West Bound, weekdays, post 2000:

Mass Pike Congestion Visualization

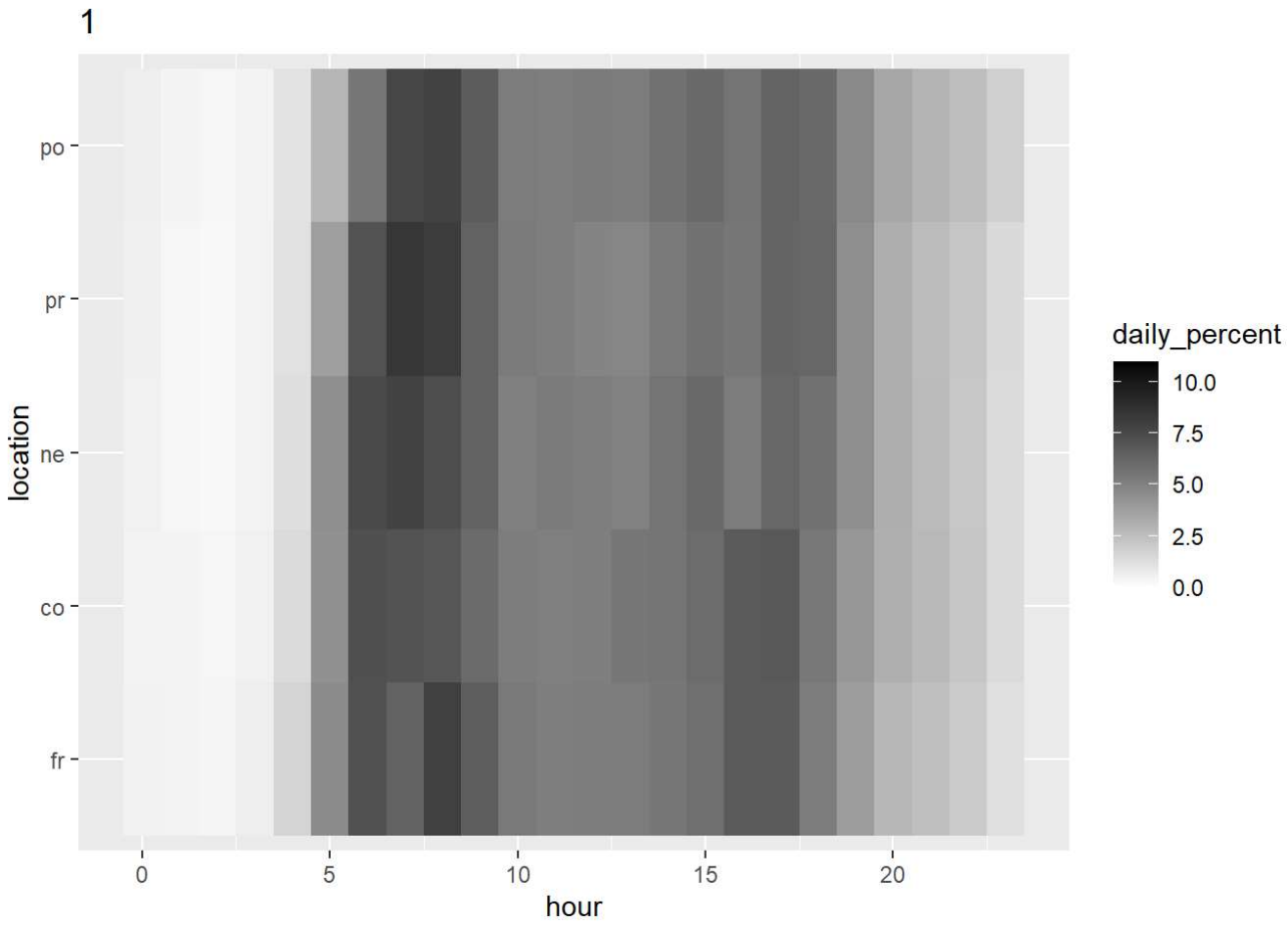
By Month

These are all of the monthly plots where `[[1]]` is equivalent to `Jaunuary` up to `[[12]]` being equivalent to `December`:

```
# East Bound
pike_master_EB <- pike_master %>%
  drop_na() %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday") %>%
  filter(direction == "EB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po"))) %>%
  mutate(year = as.factor(year(full_date)))
```

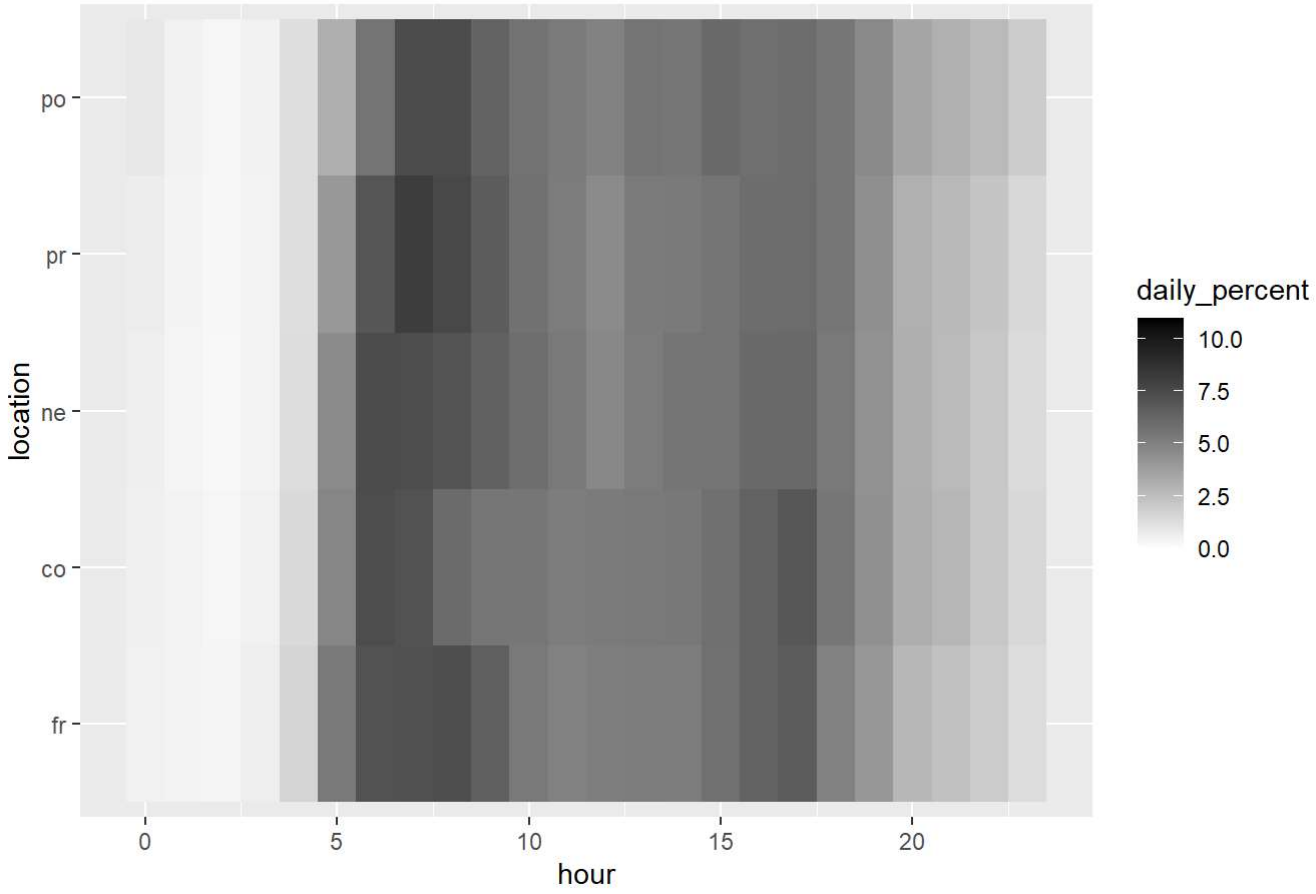
```
pike_master_EB %>%
  mutate(month = month(full_date)) %>%
  group_by(month) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
        geom_tile(aes(fill = daily_percent)) +
        scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
          labs(title = month(.$full_date)))
```

```
## [[1]]
```



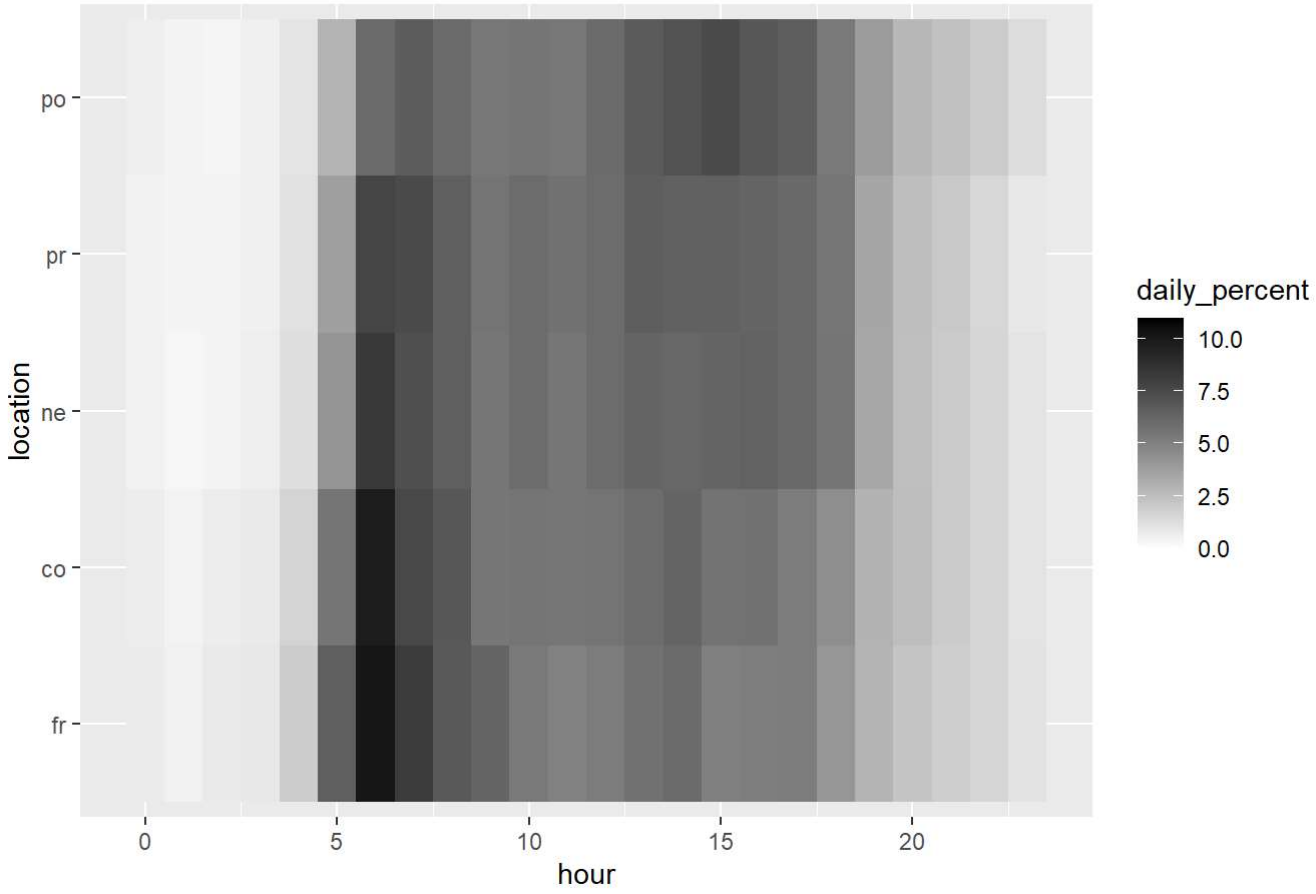

```
##
## [[2]]
```

2


po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

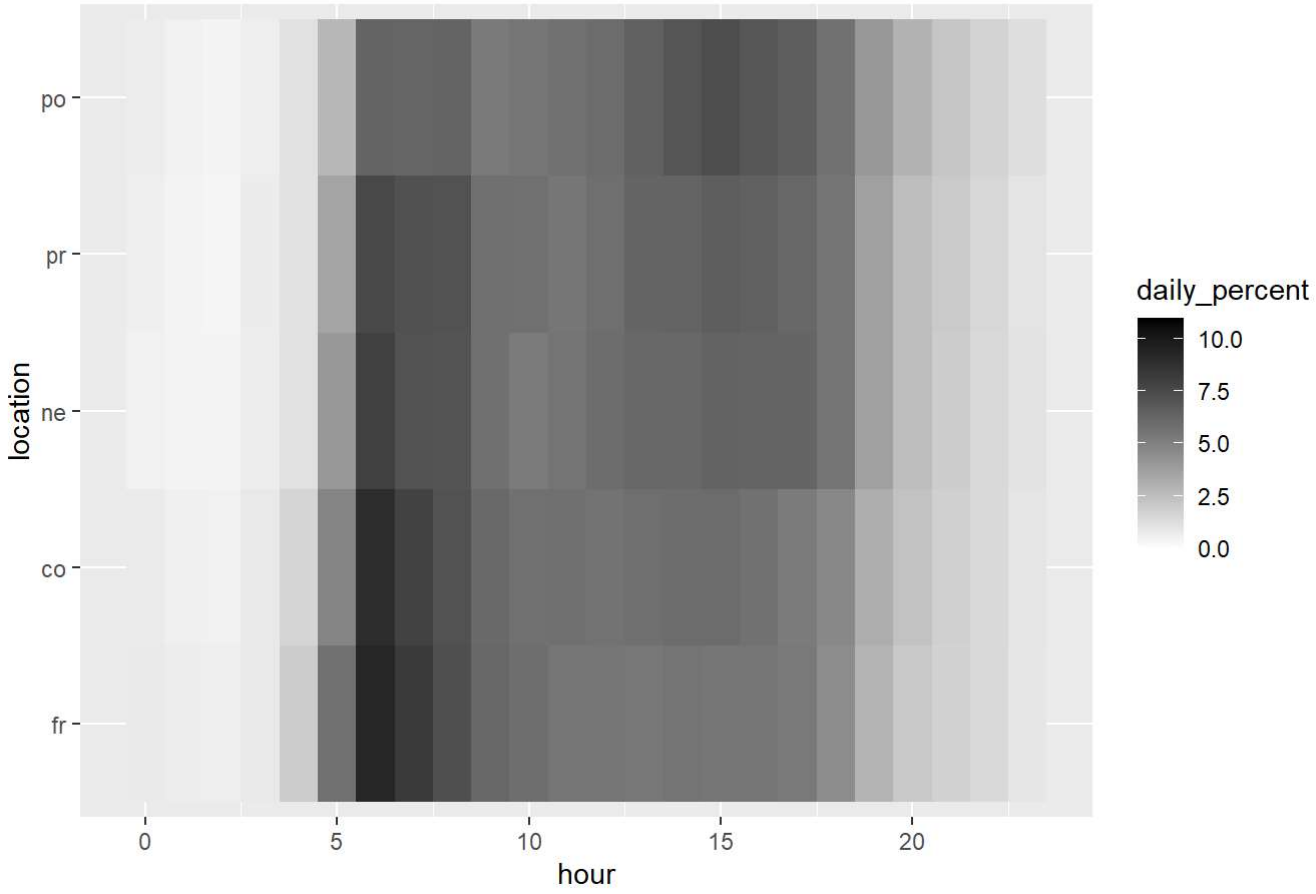
```
##
## [[3]]
```

3

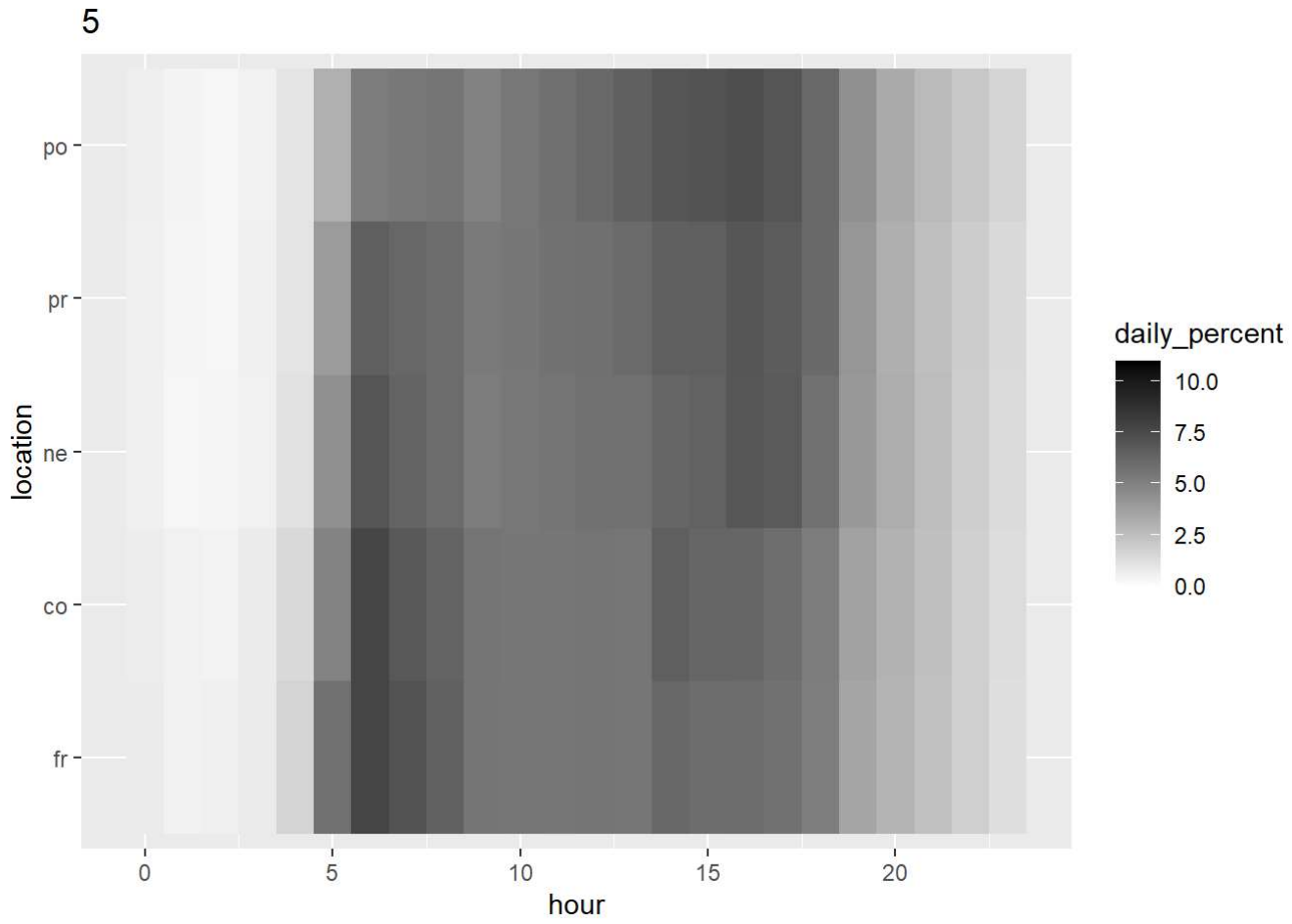

po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[4]]
```

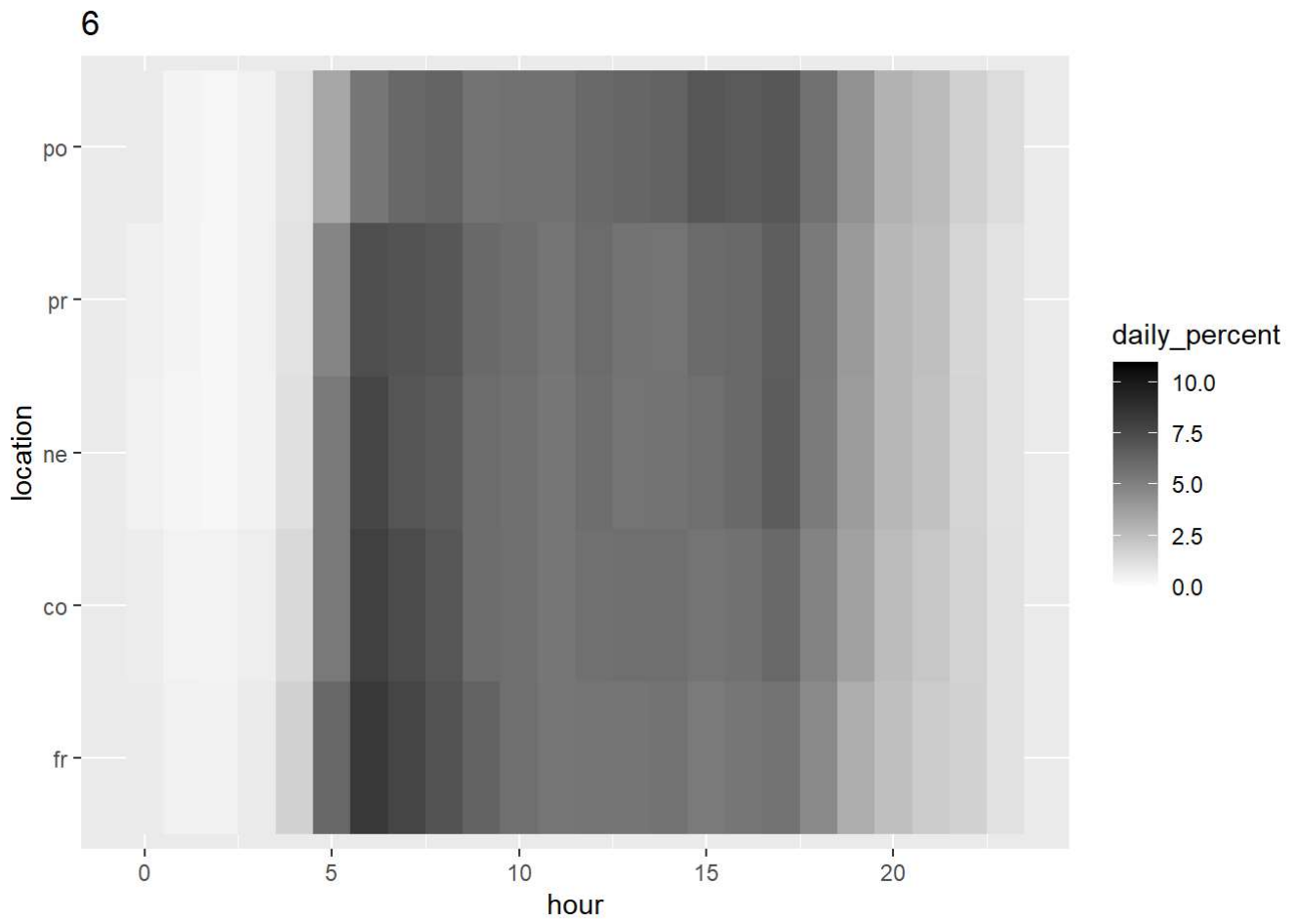
4


po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

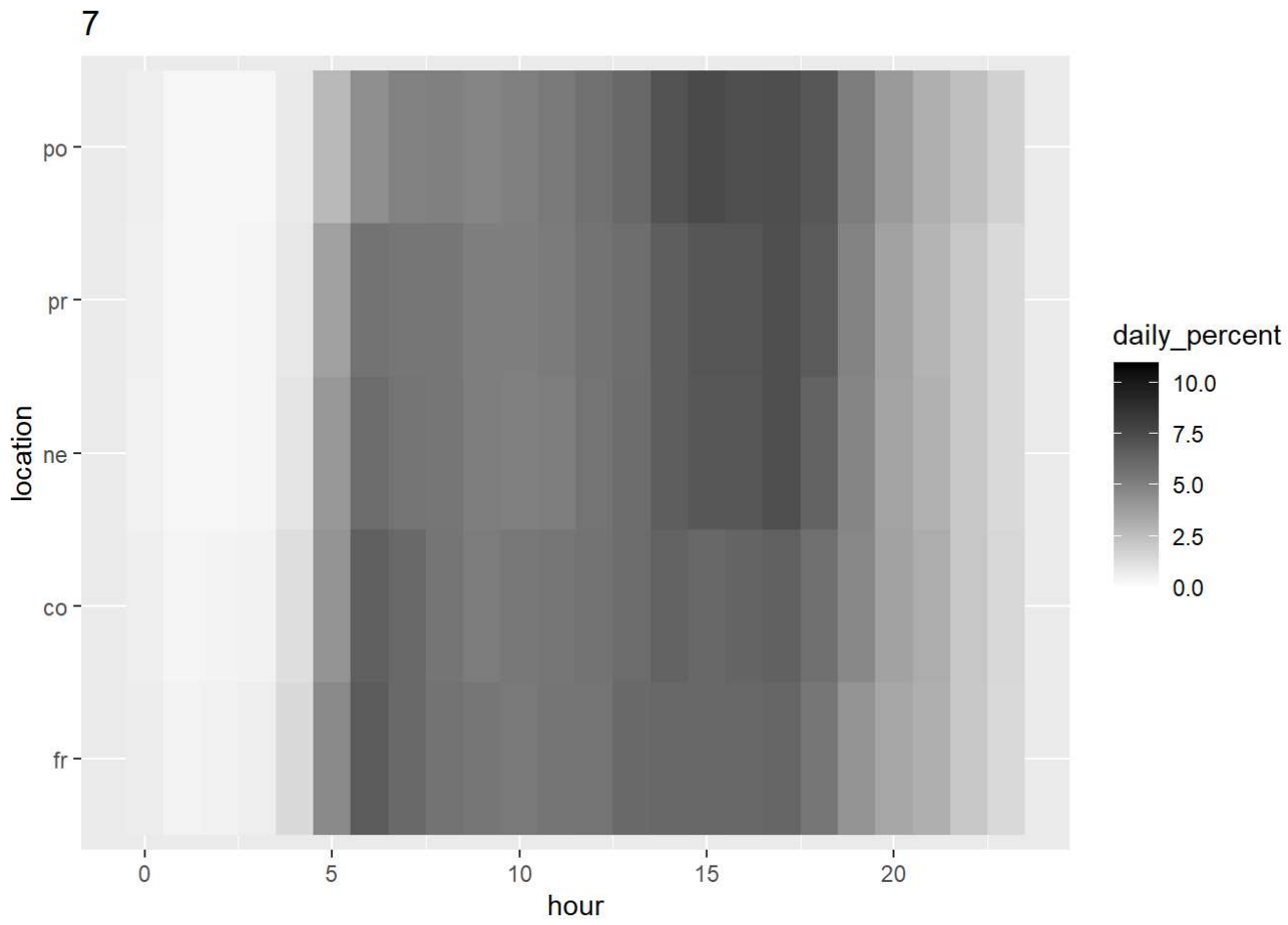
```
##
## [[5]]
```


5
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

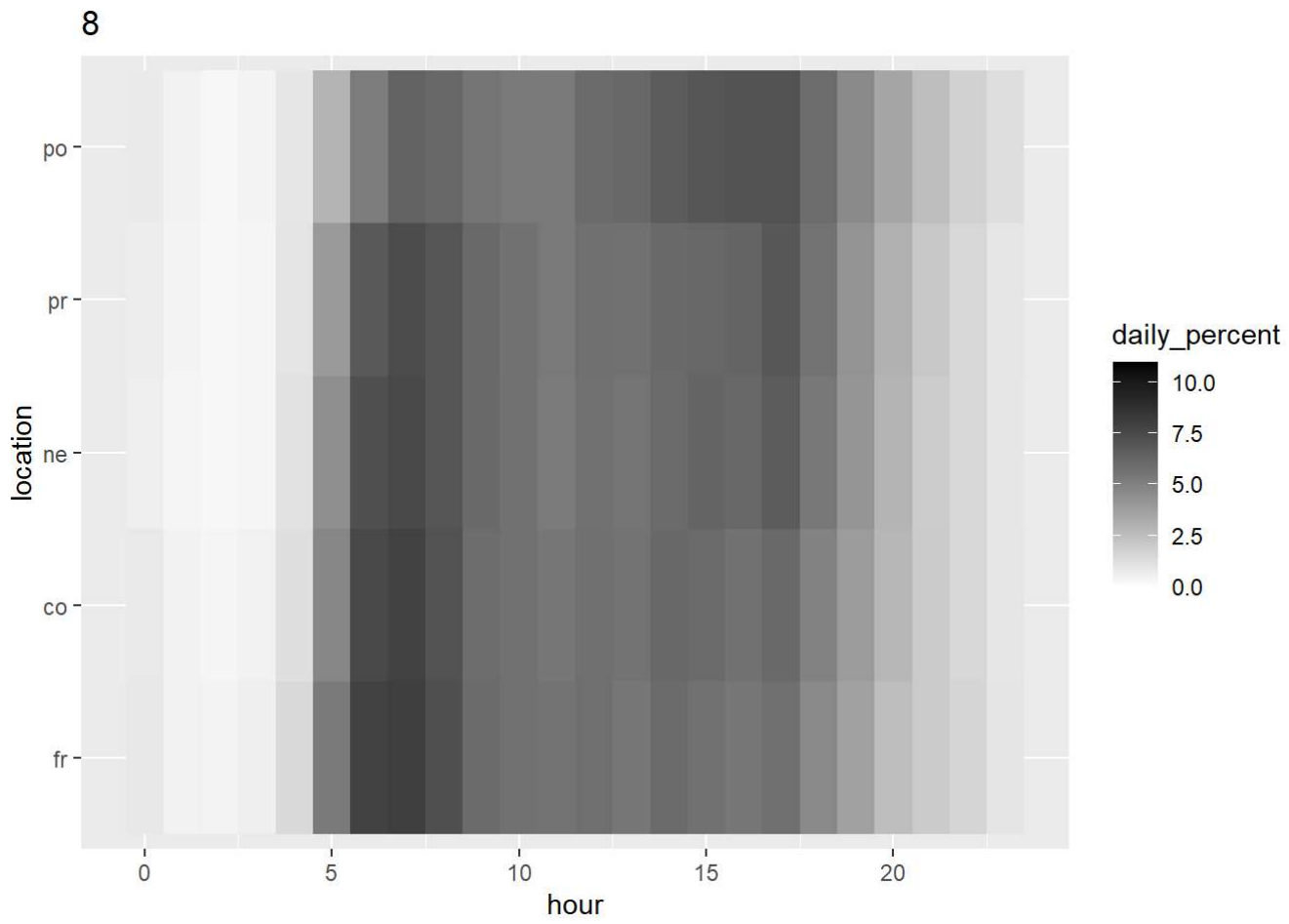
```
##
## [[6]]
```


6
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

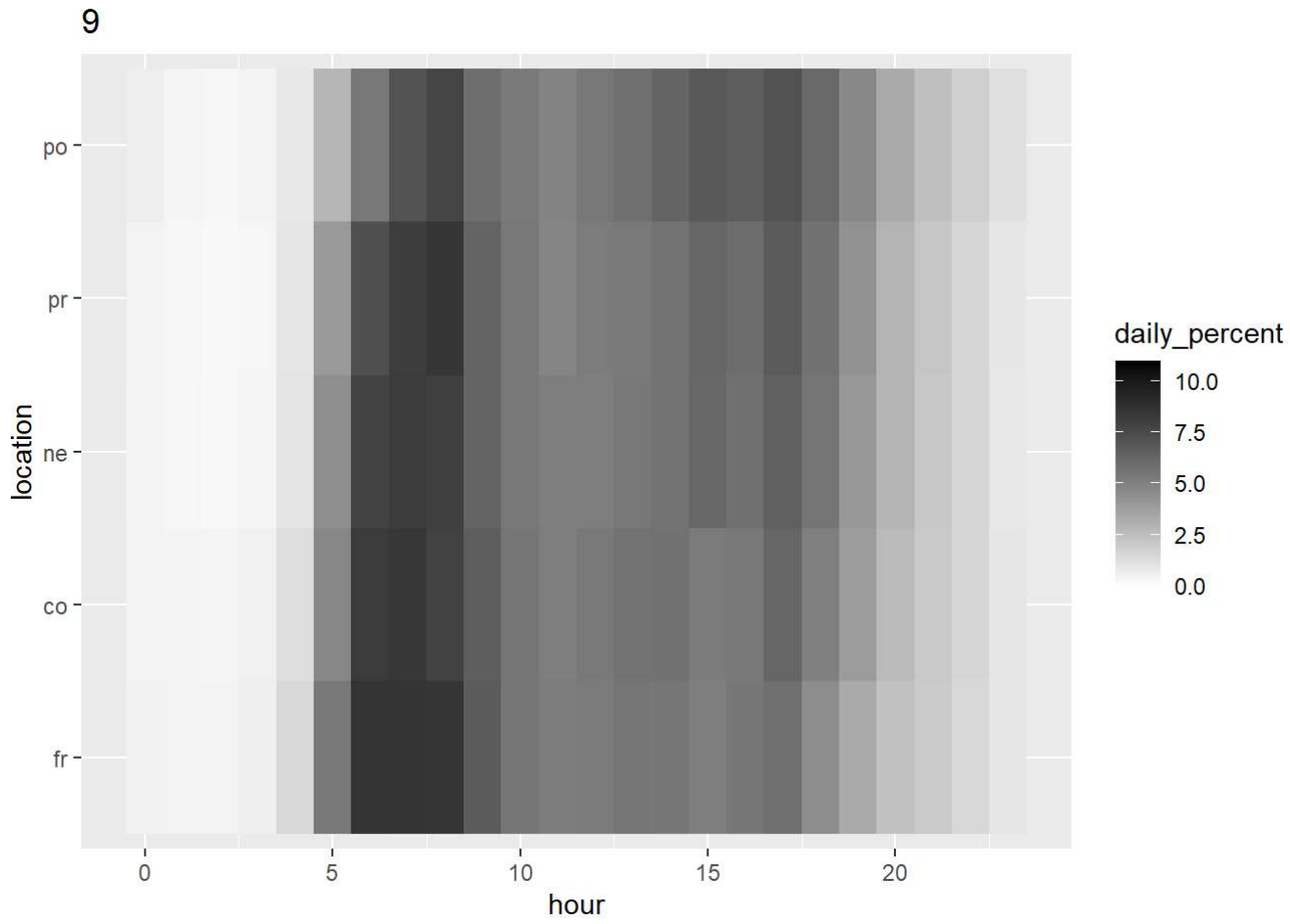
```
##
## [[7]]
```


7
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

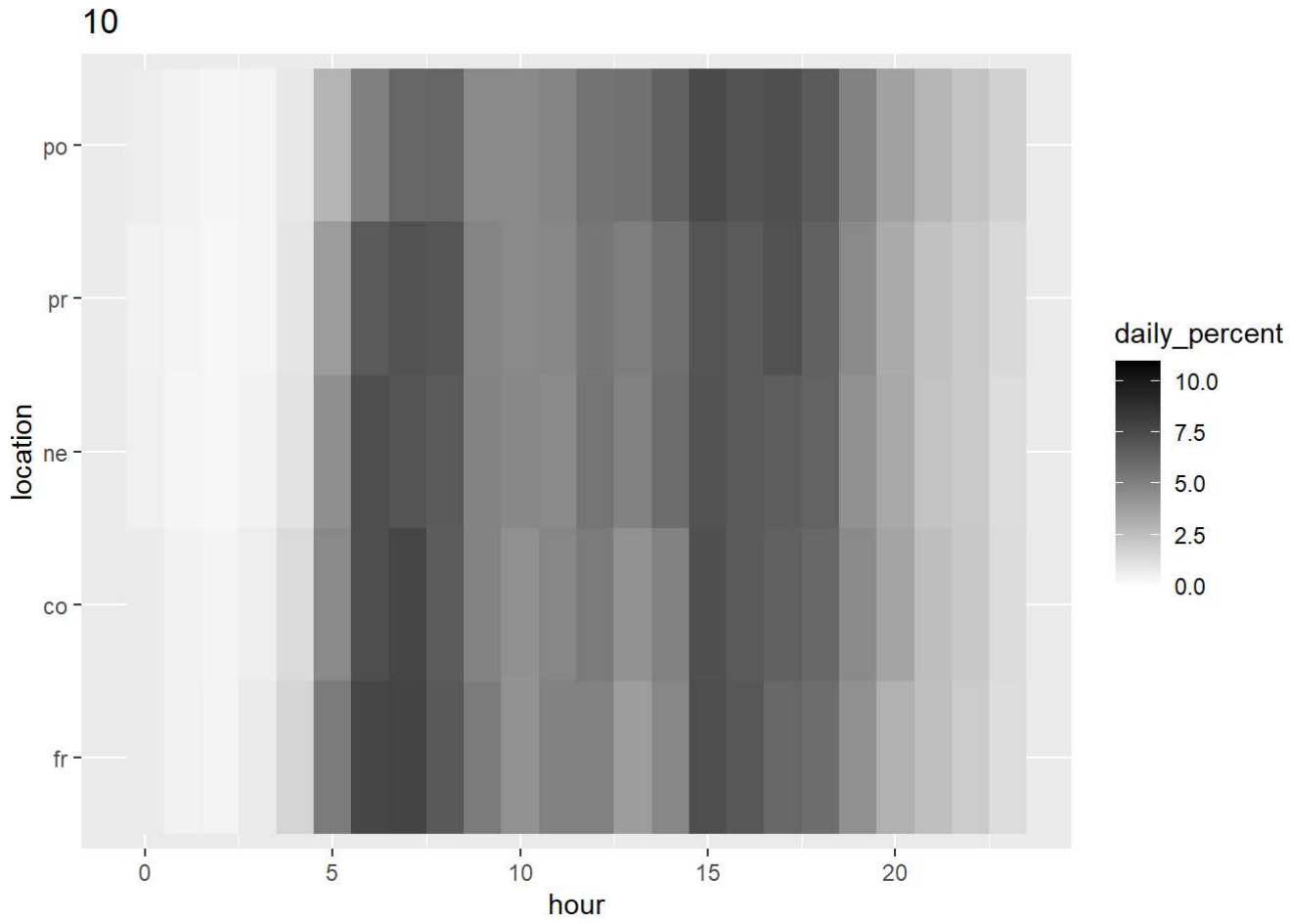
```
##
## [[8]]
```


8
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

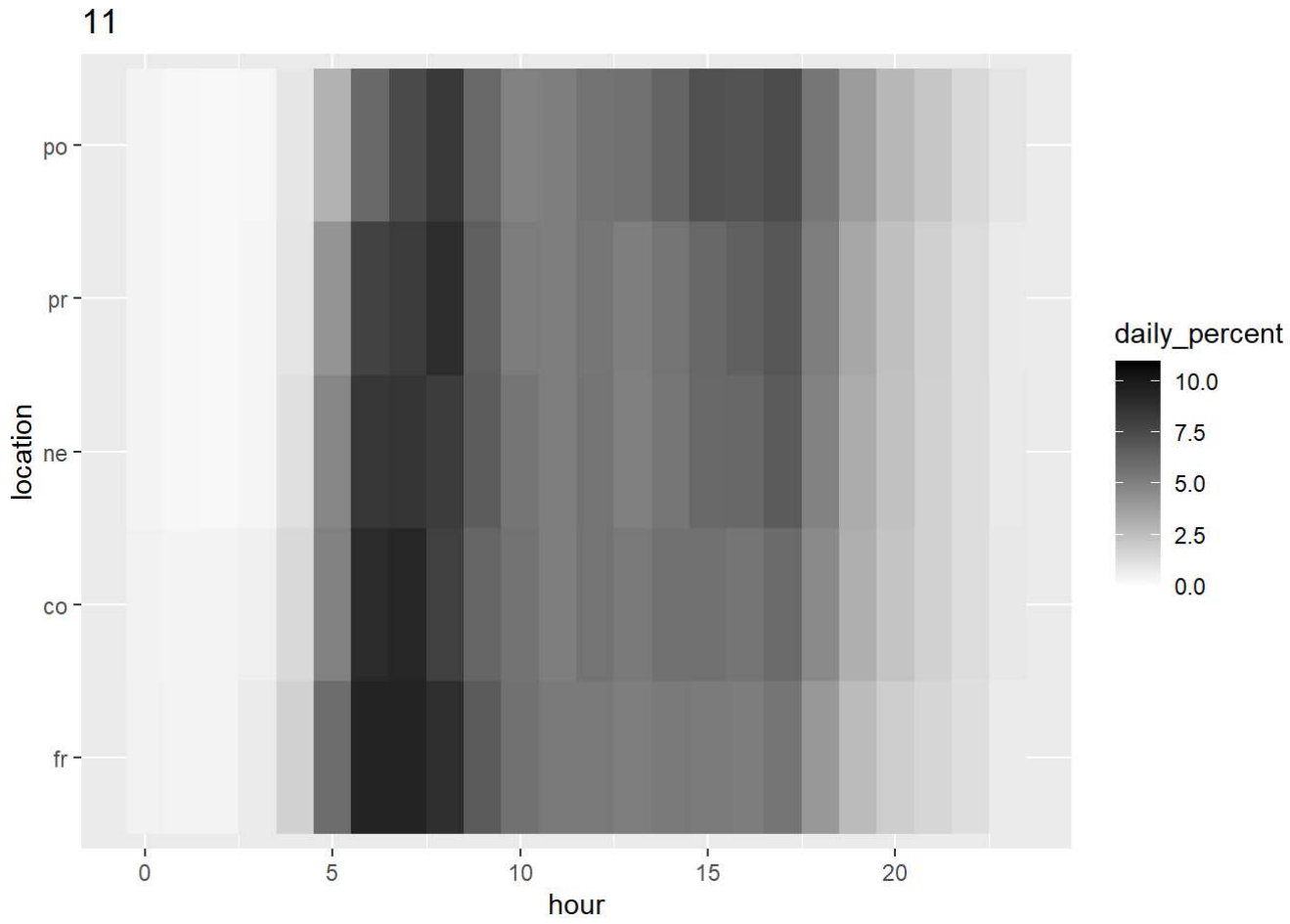
```
##
## [[9]]
```


9
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

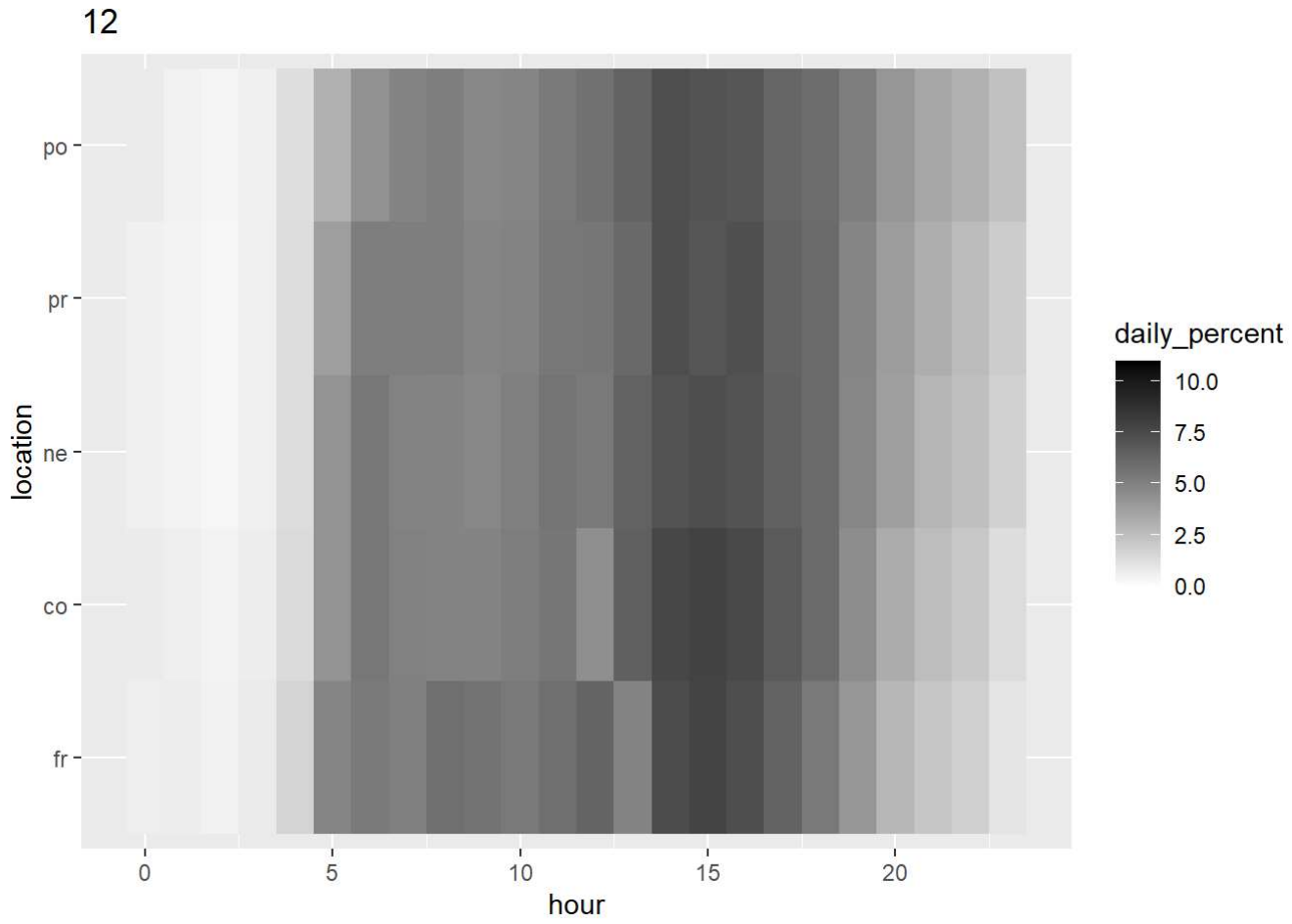
```
##
## [[10]]
```


10
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[11]]
```


11
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

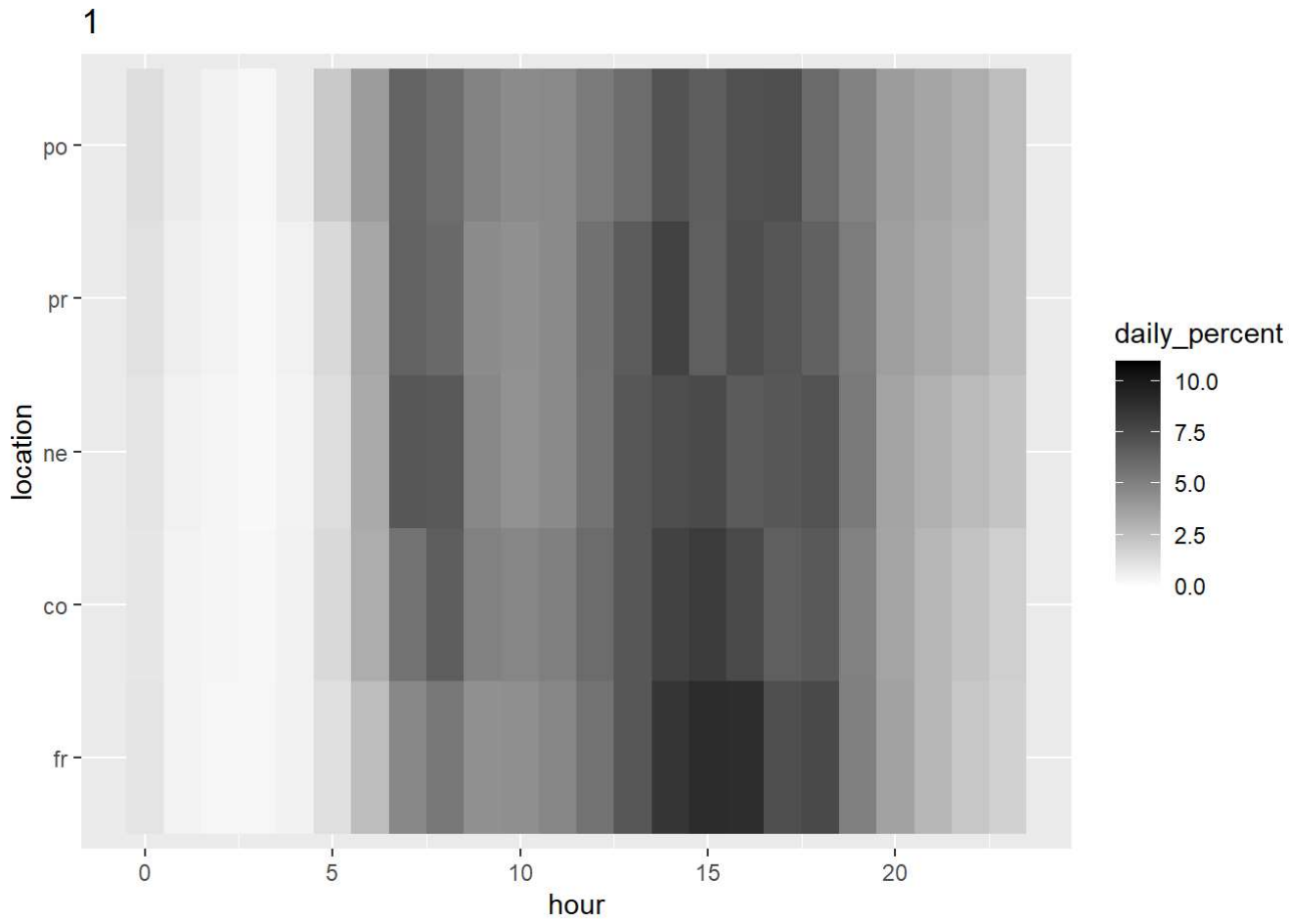
```
##
## [[12]]
```




```
# West Bound
pike_master_WB <- pike_master %>%
  drop_na() %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday") %>%
  filter(direction == "WB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po"))) %>%
  mutate(year = as.factor(year(full_date)))
```

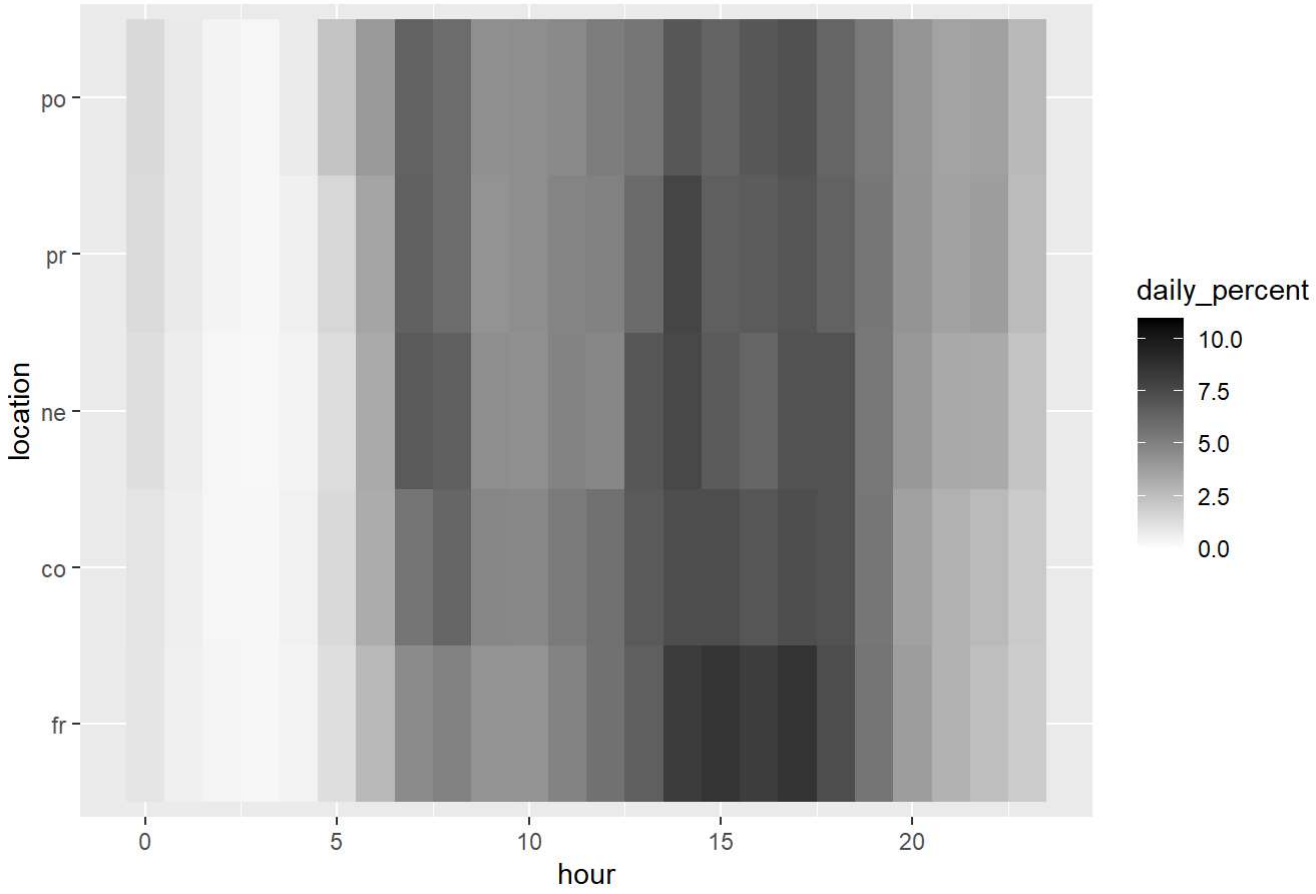
```
pike_master_WB %>%
  group_by(month(full_date)) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
              labs(title = month(.$full_date)))
```

```
## [[1]]
```

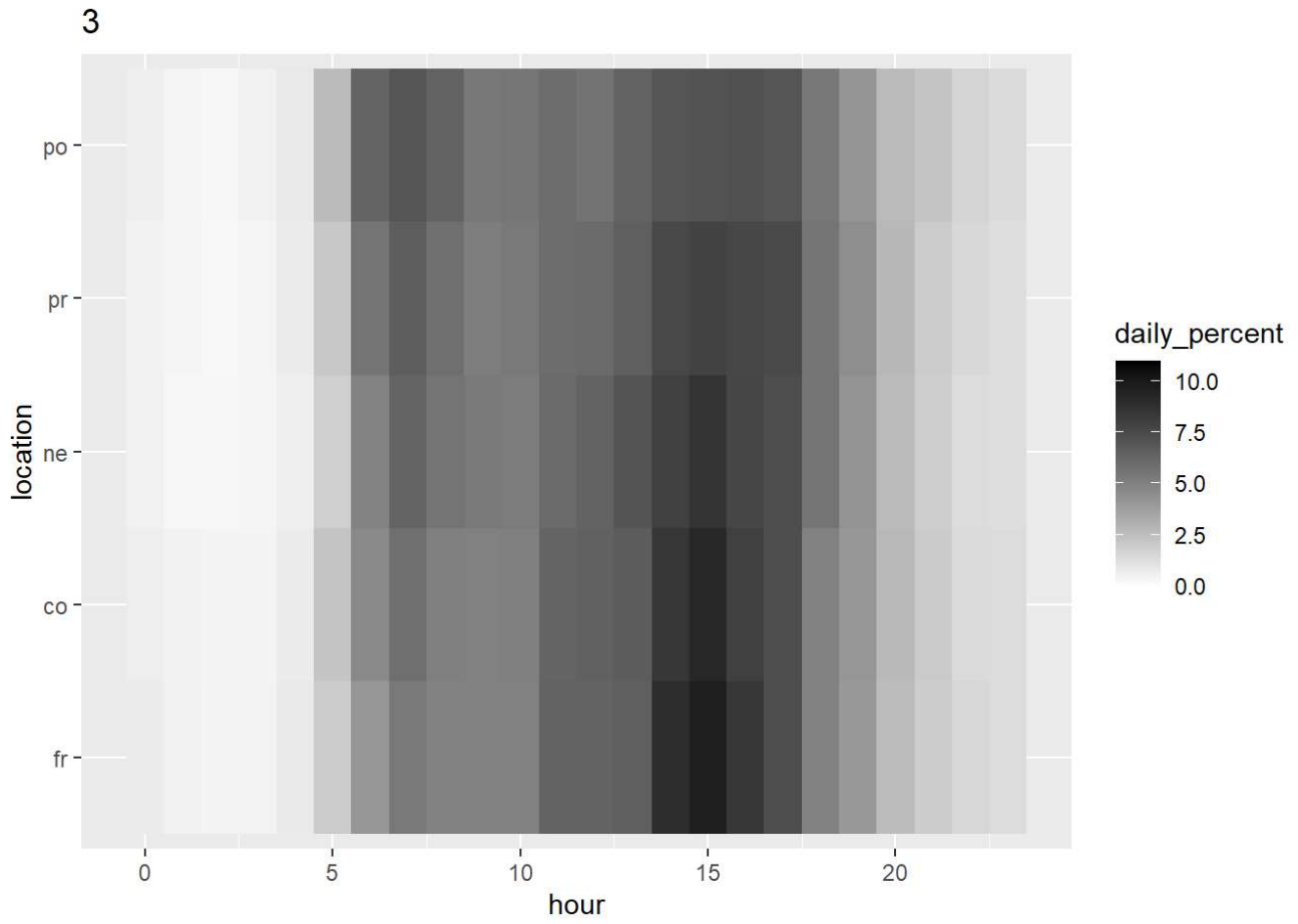

1
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[2]]
```

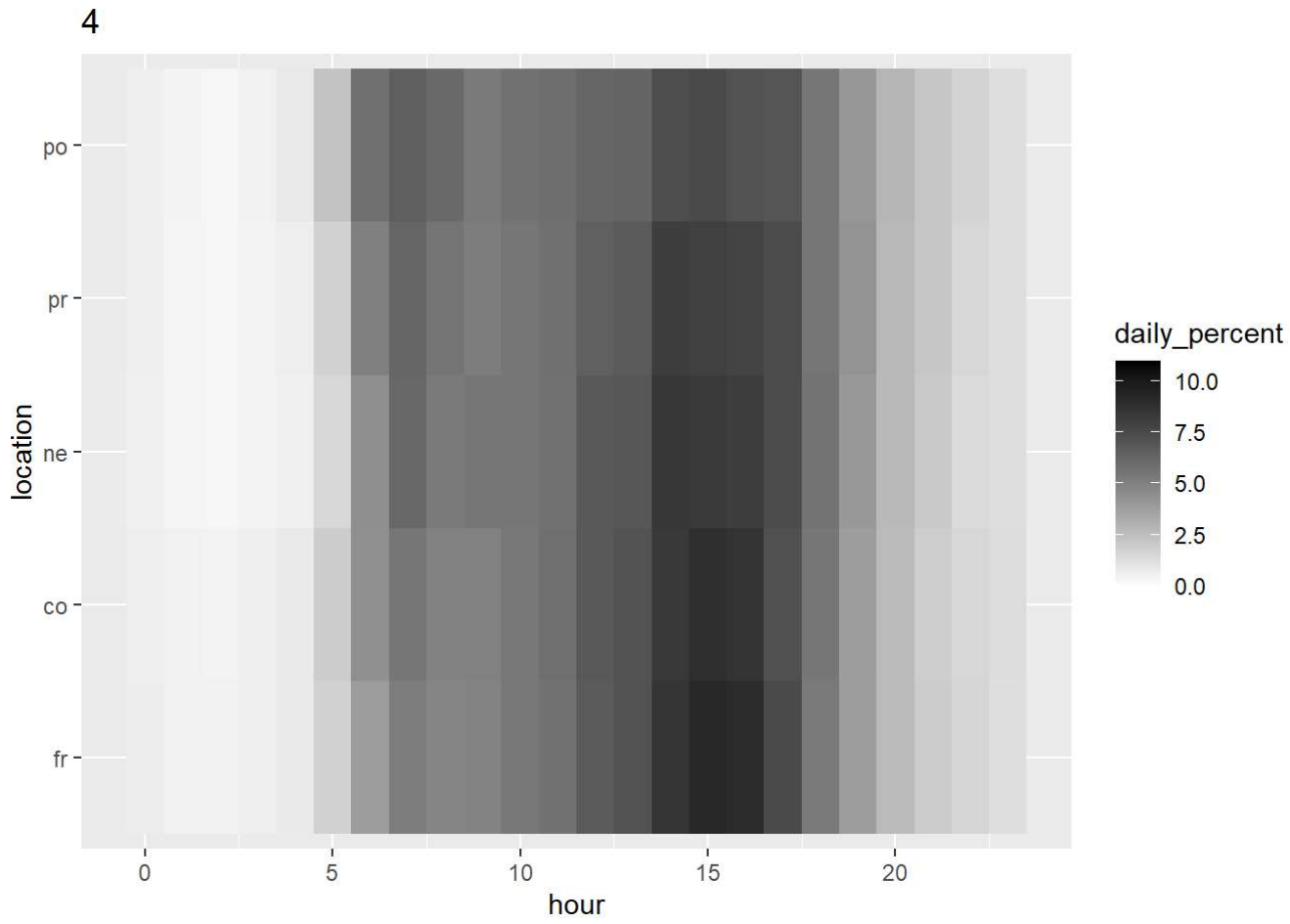
2


location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

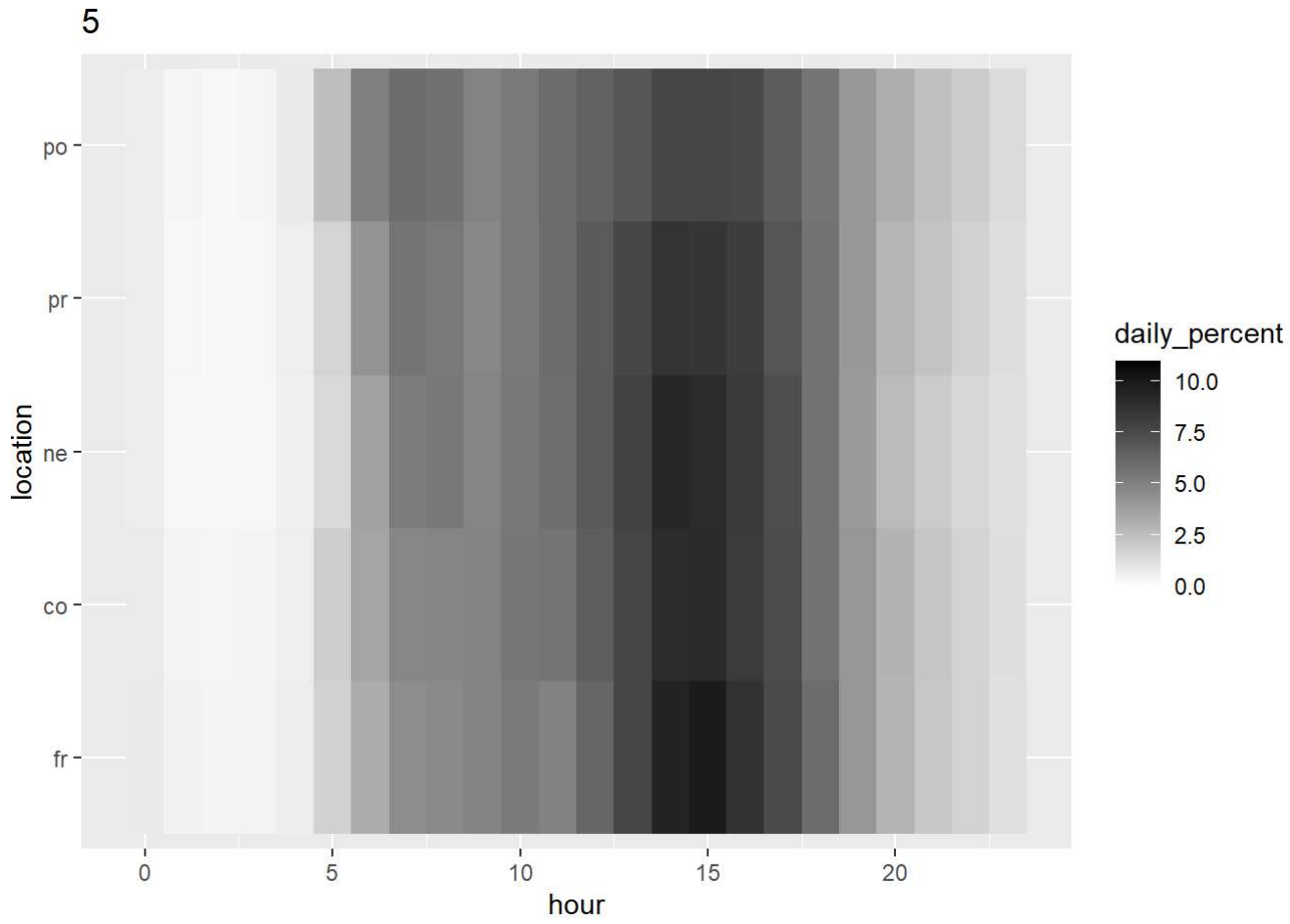
```
##
## [[3]]
```


3
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

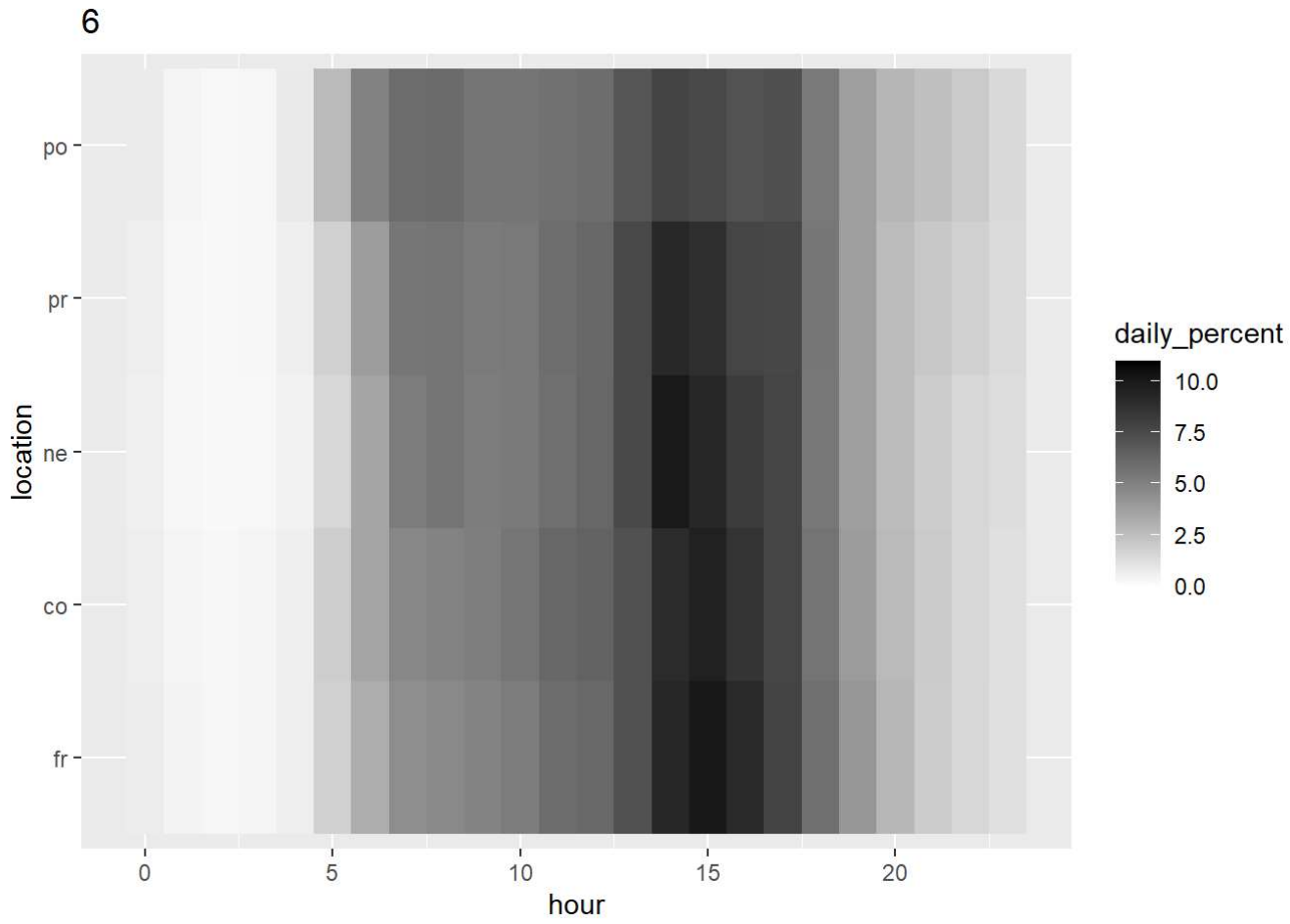
```
##
## [[4]]
```


4
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

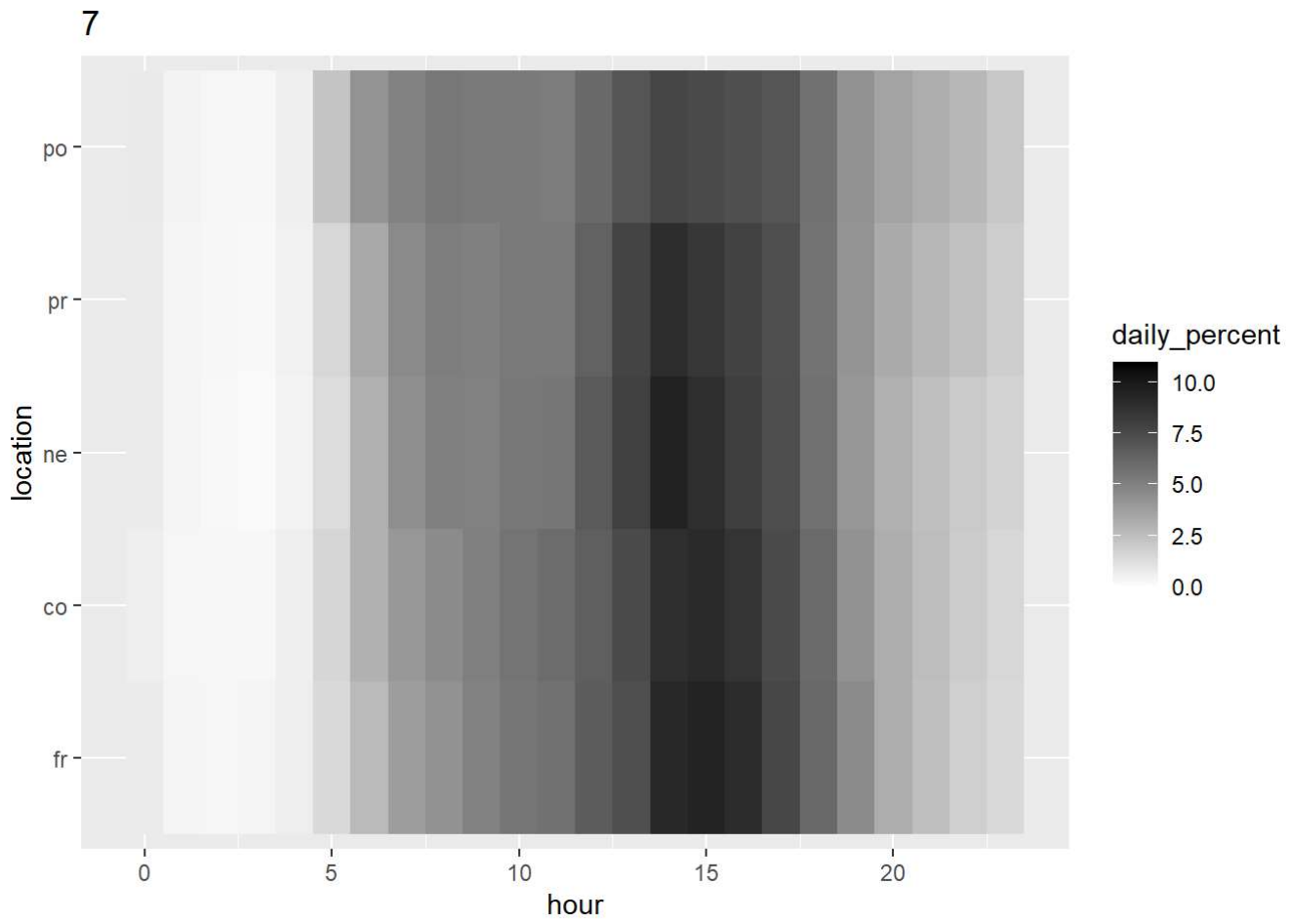
```
##
## [[5]]
```


5
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

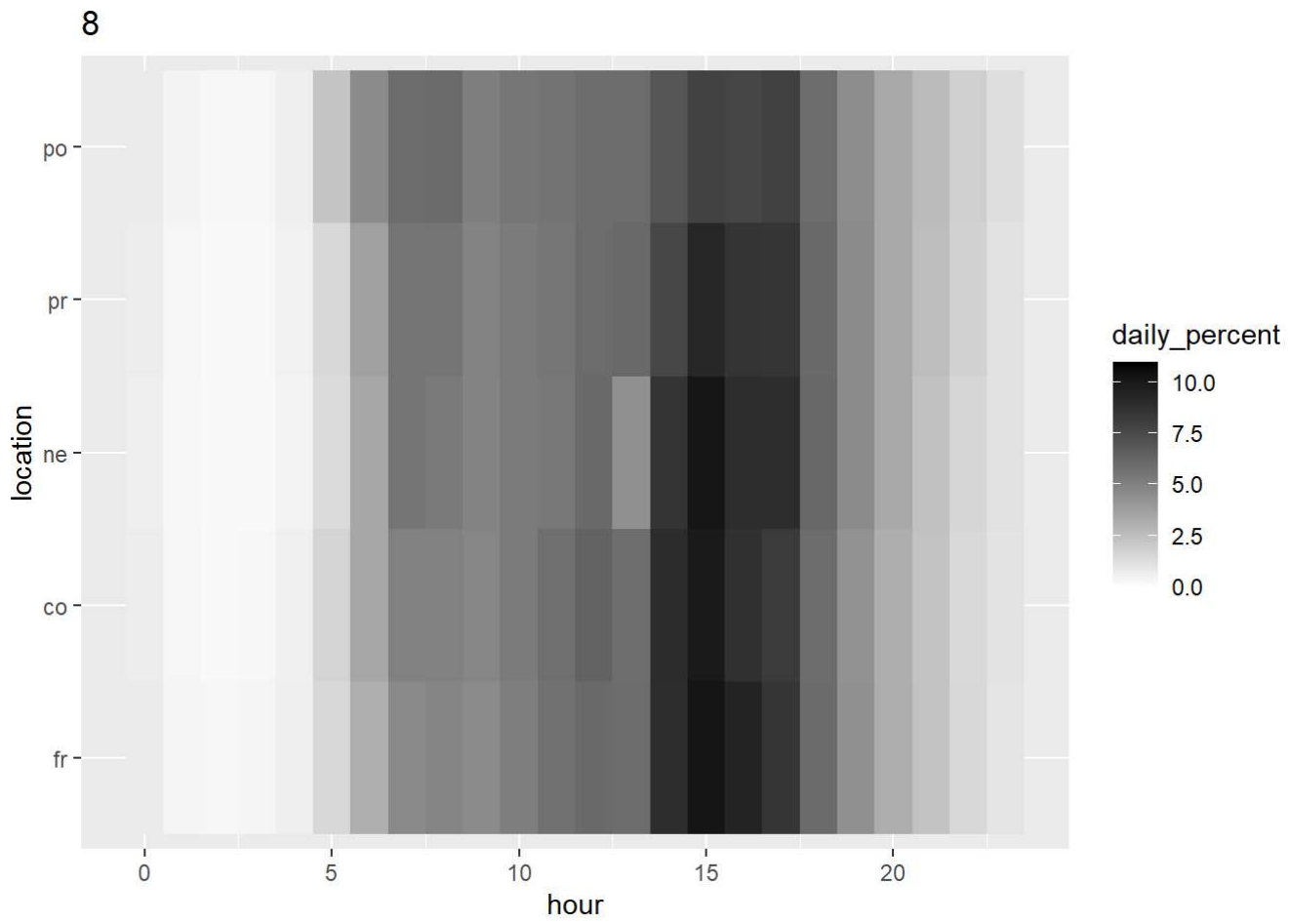
```
##
## [[6]]
```


6
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

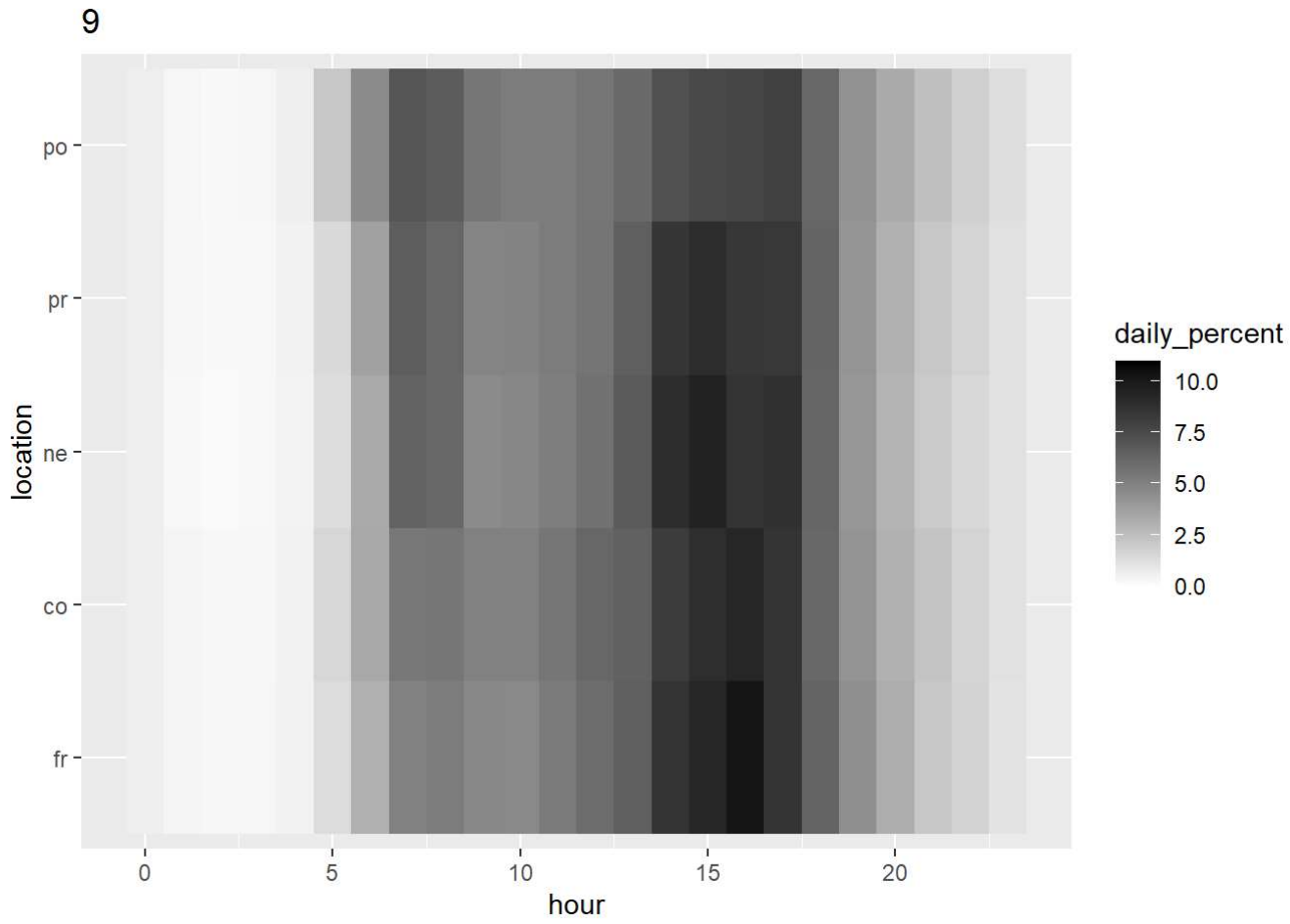
```
##
## [[7]]
```


7
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

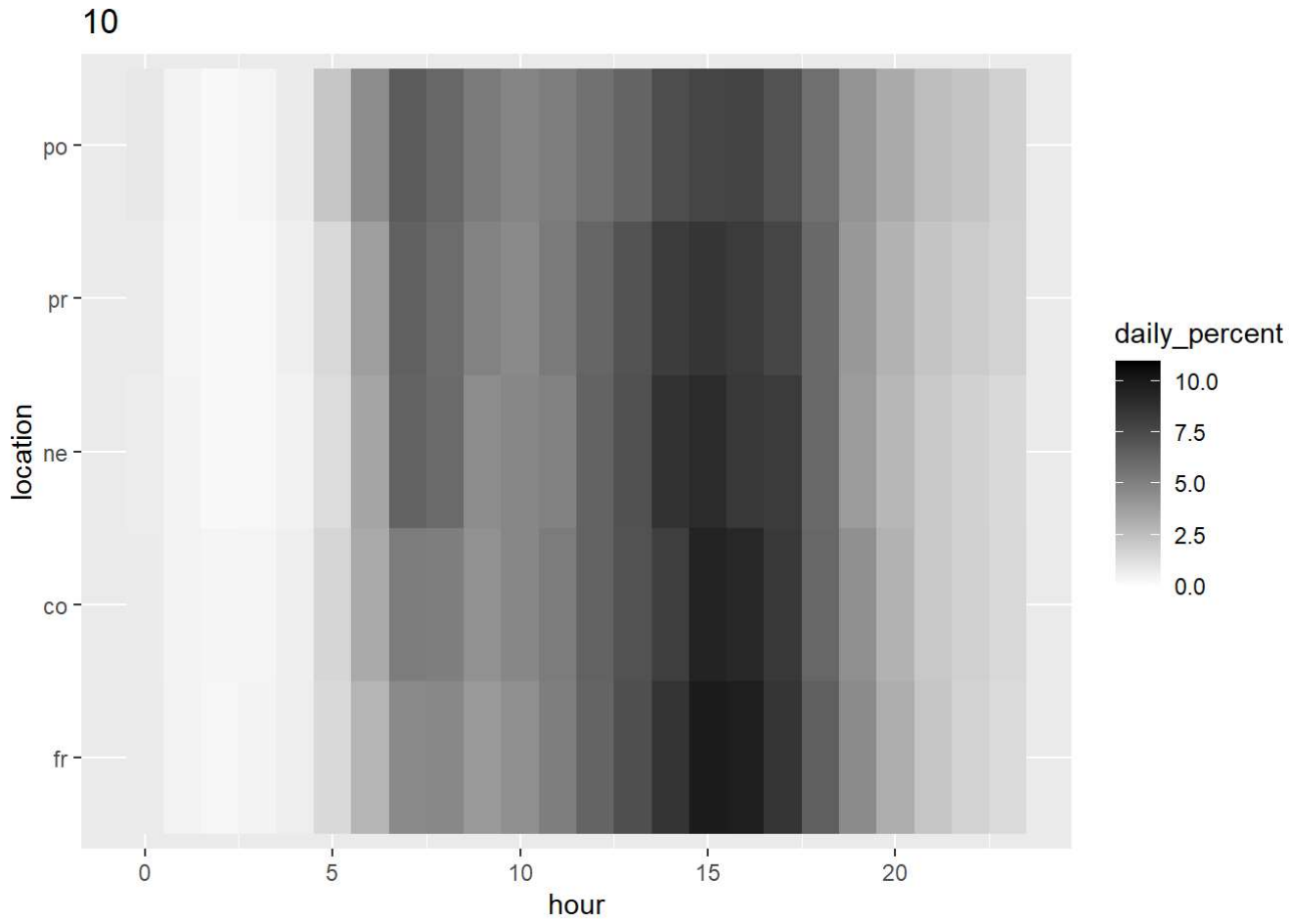
```
##
## [[8]]
```


8
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

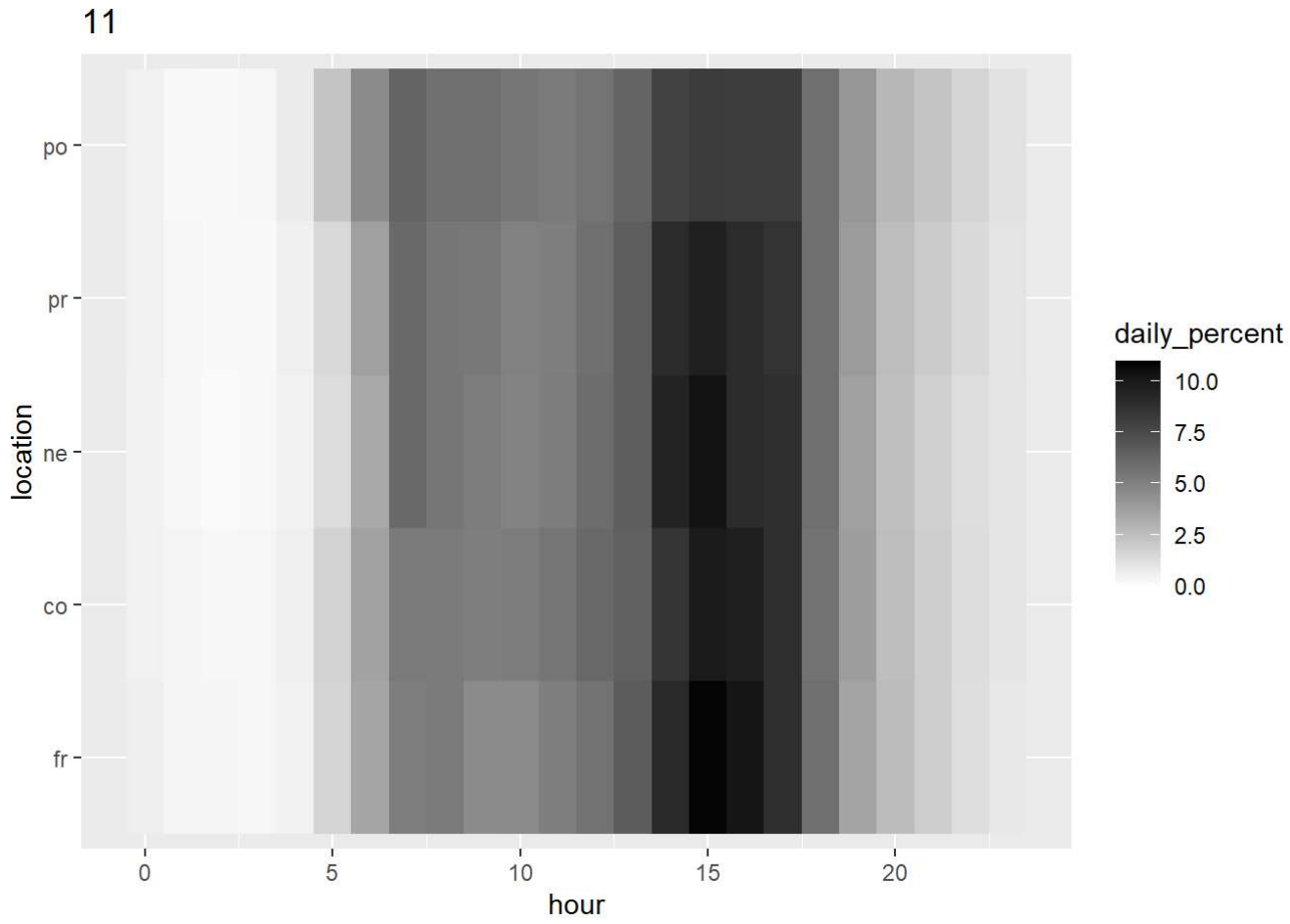
```
##
## [[9]]
```


9
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

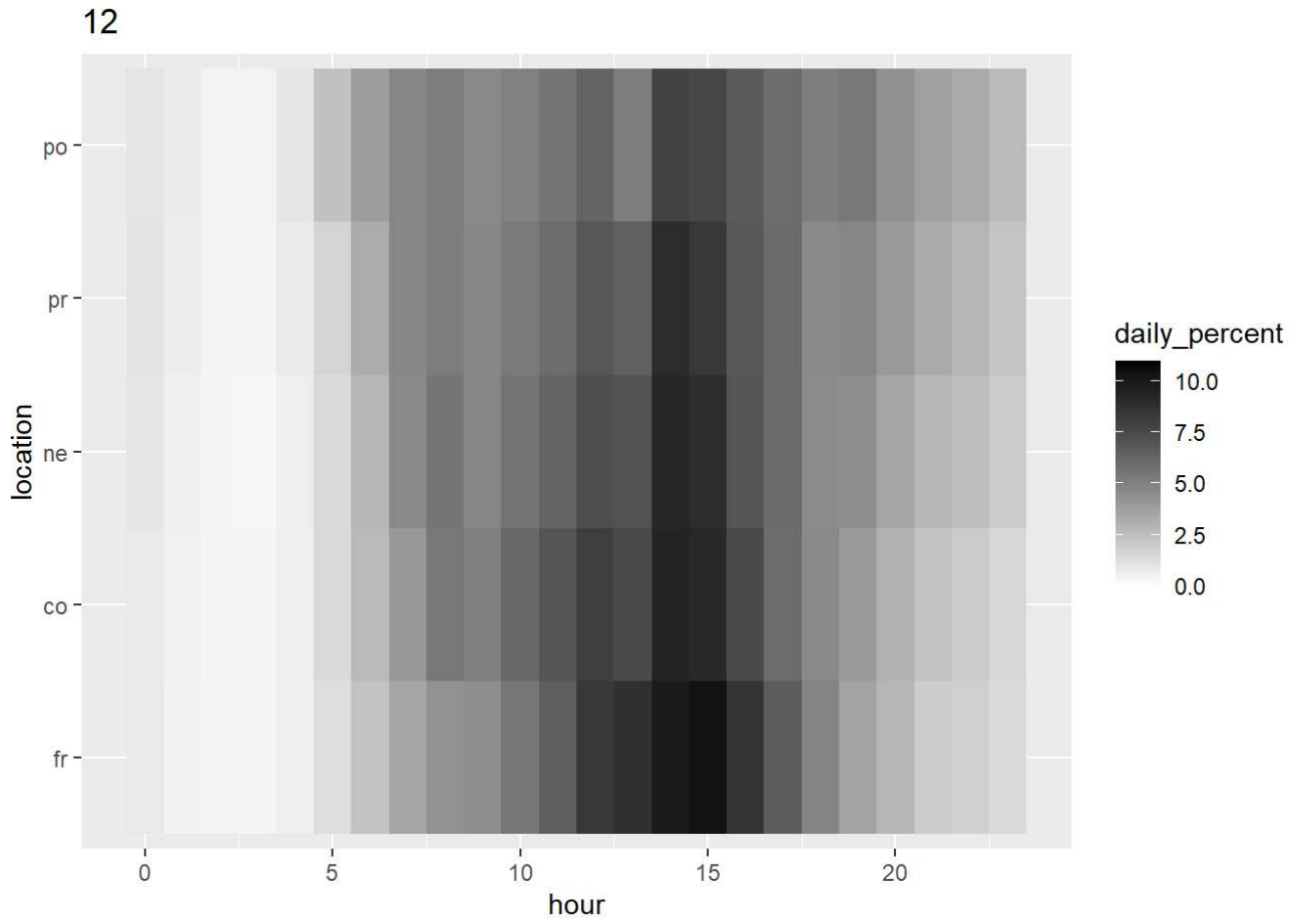
```
##
## [[10]]
```


10
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[11]]
```


11
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[12]]
```



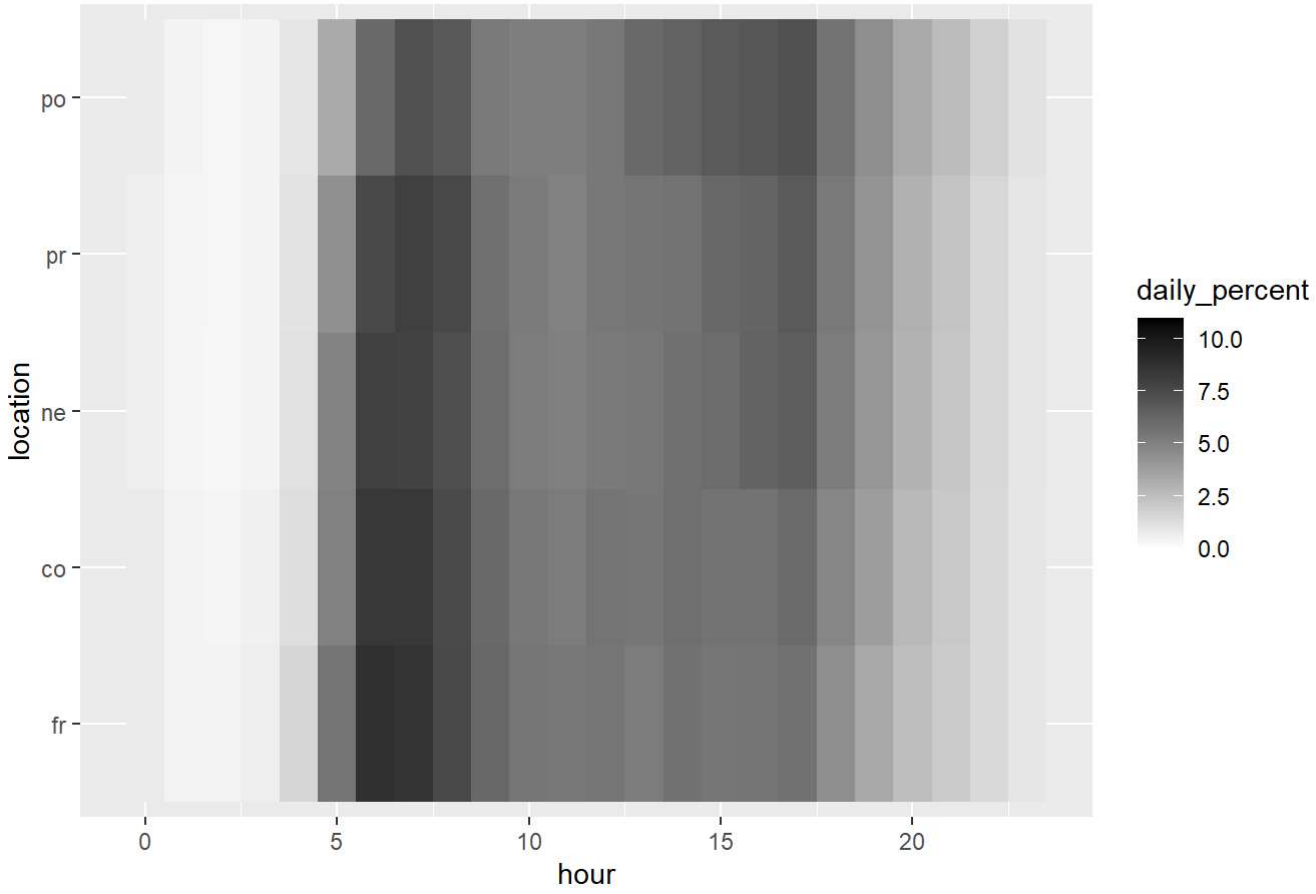
By Weekday

These are all of the plots per day of the week where `[[1]]` is equivalent to `Monday` up to `[[5]]` being equivalent to `Friday` .

```
# East Bound
pike_master_EB %>%
  group_by(day_of_week) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
              labs(title = (day(.$full_date - 1))))
```

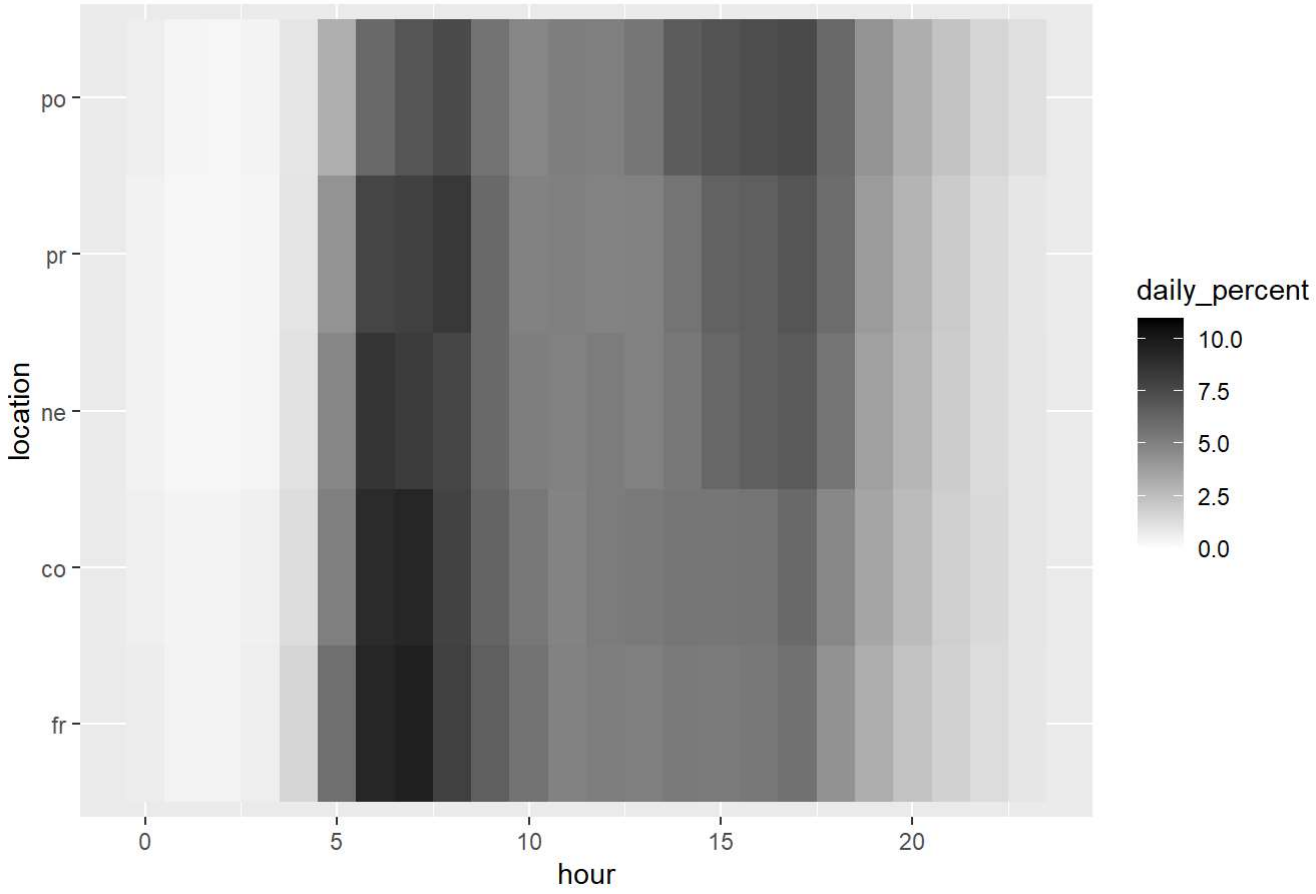
```
## [[1]]
```

1

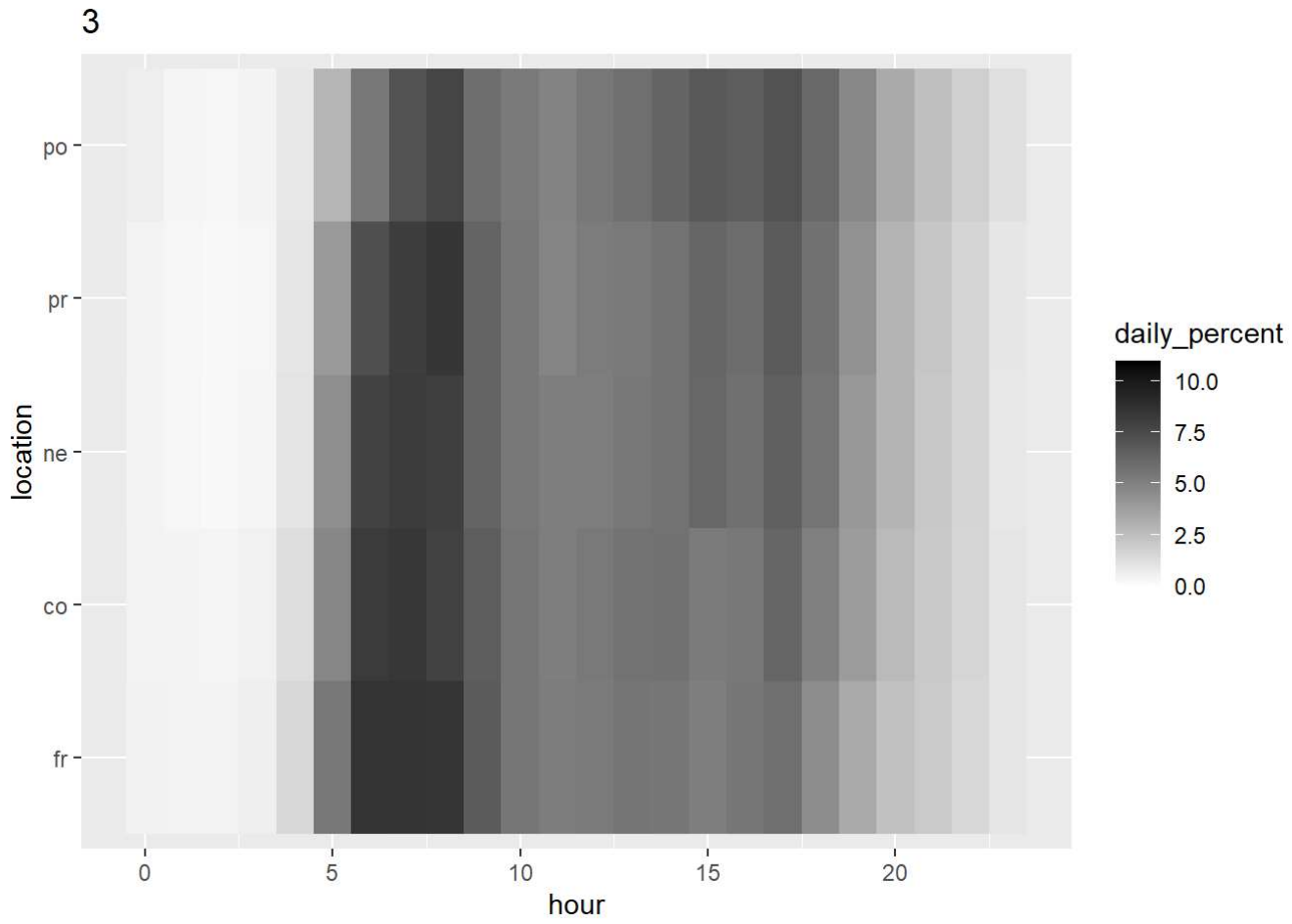

po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[2]]
```

2

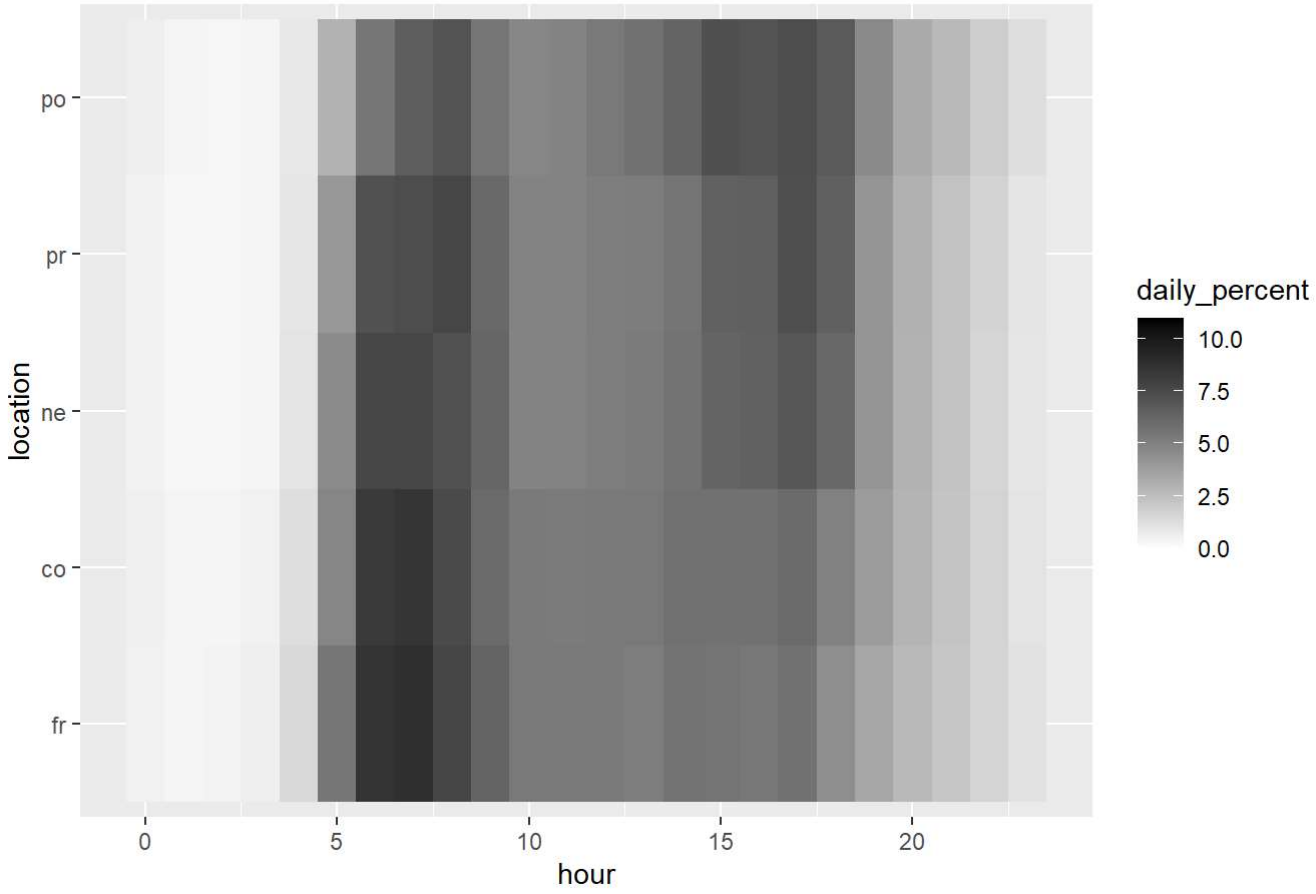

po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[3]]
```

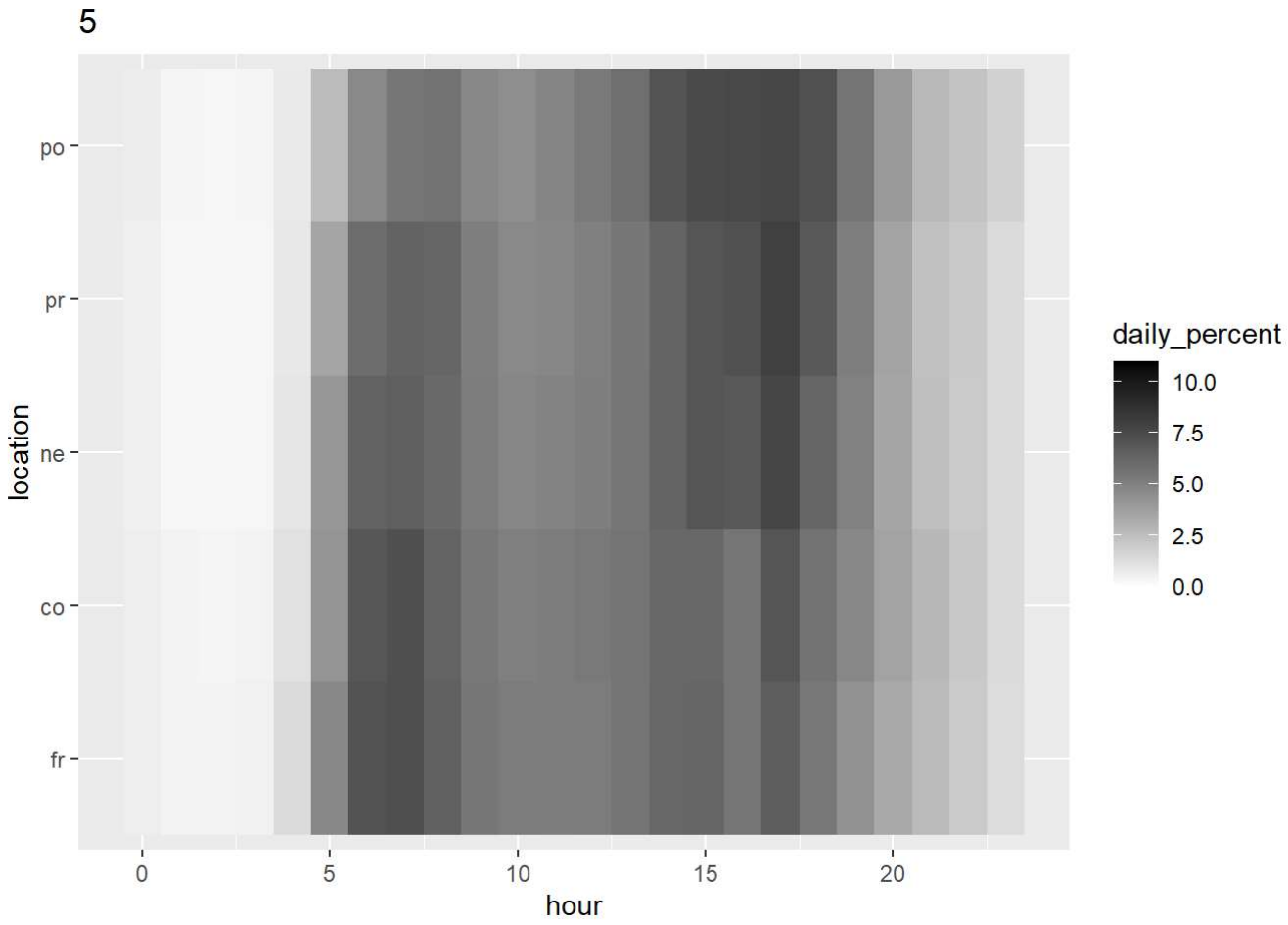

3
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[4]]
```

4

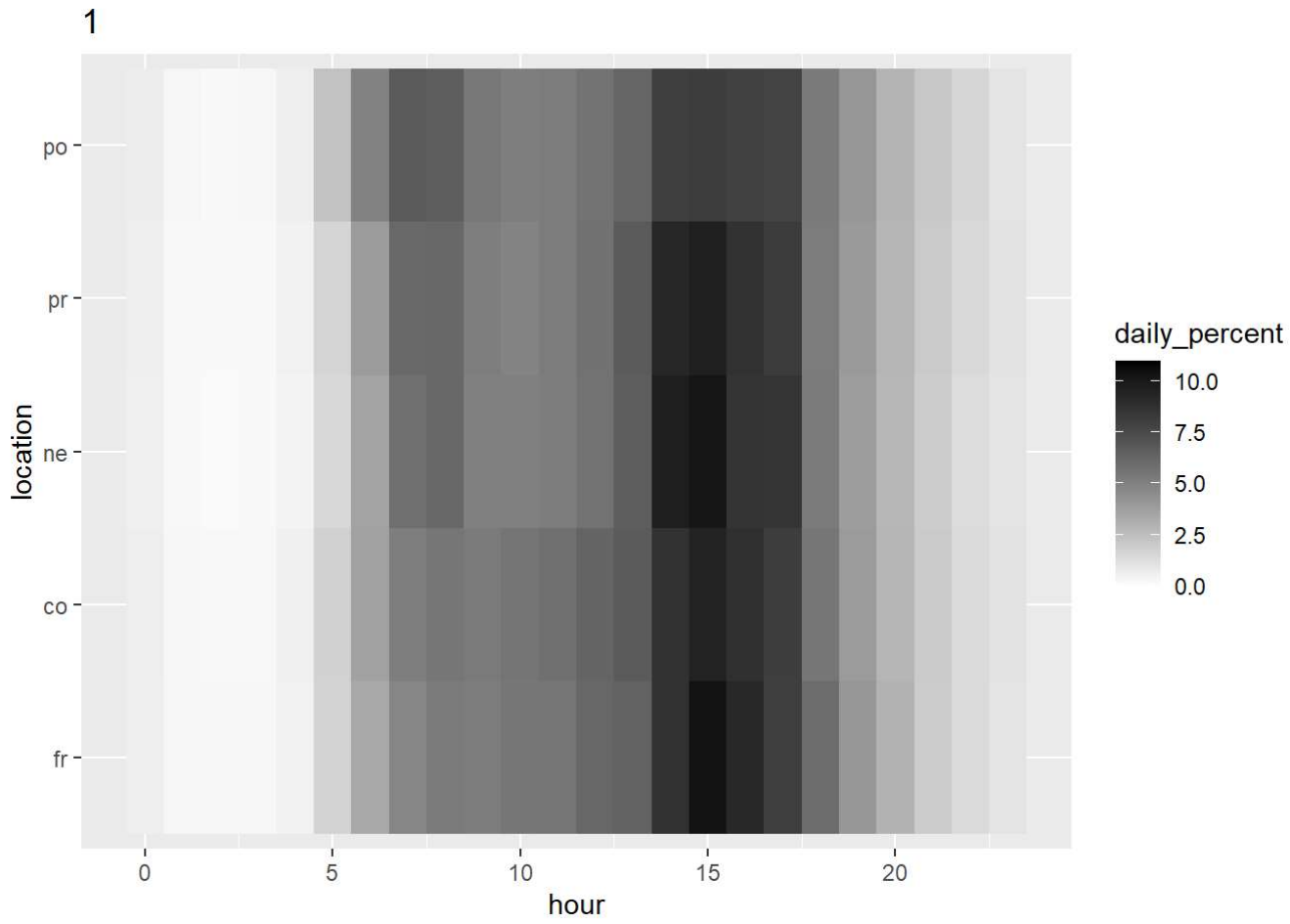

po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[5]]
```



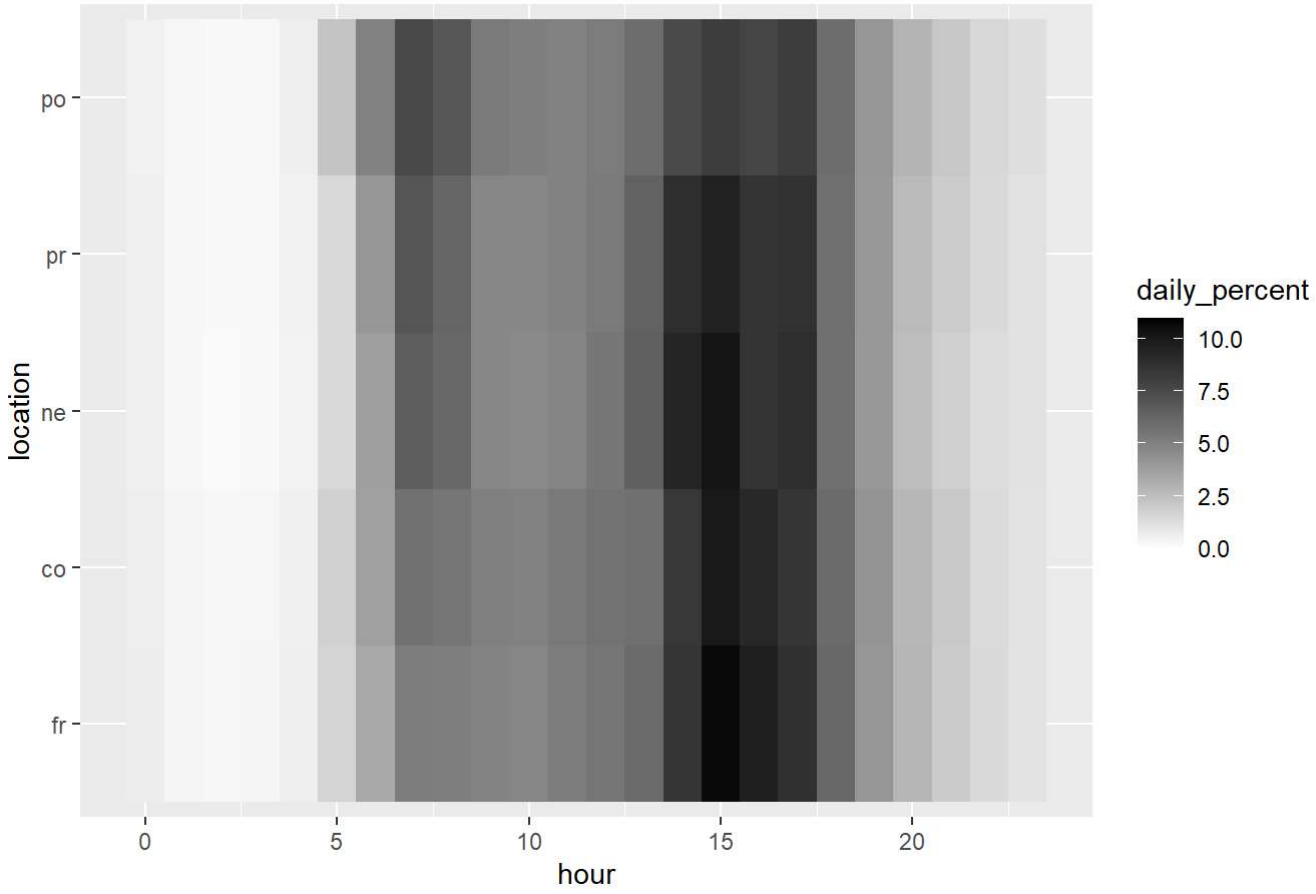

```
# West Bound
pike_master_WB %>%
  group_by(day_of_week) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
              labs(title = (day(.$full_date - 1))))
```

```
## [[1]]
```

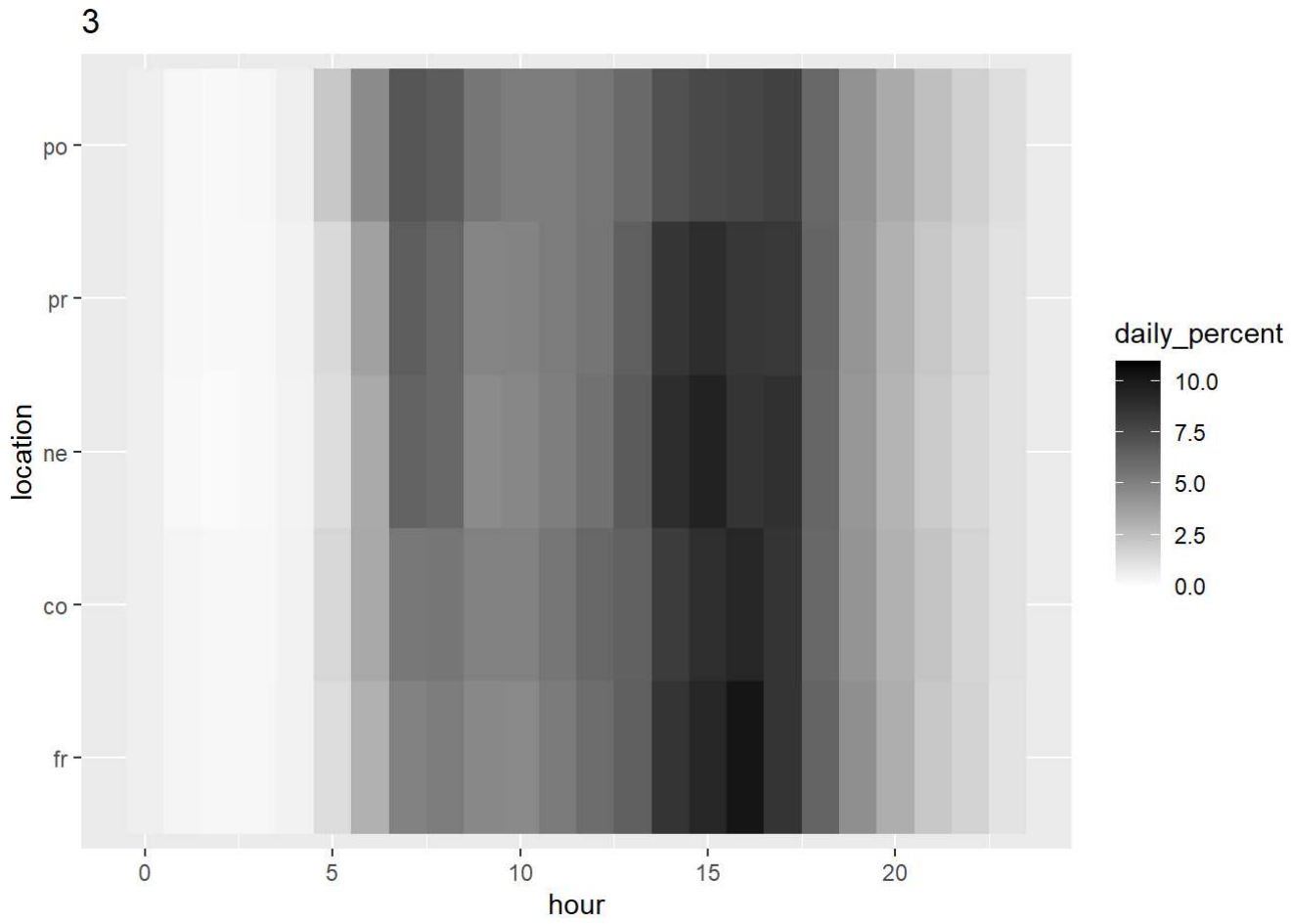

1
location
po
pr
ne
co
fr
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[2]]
```

2

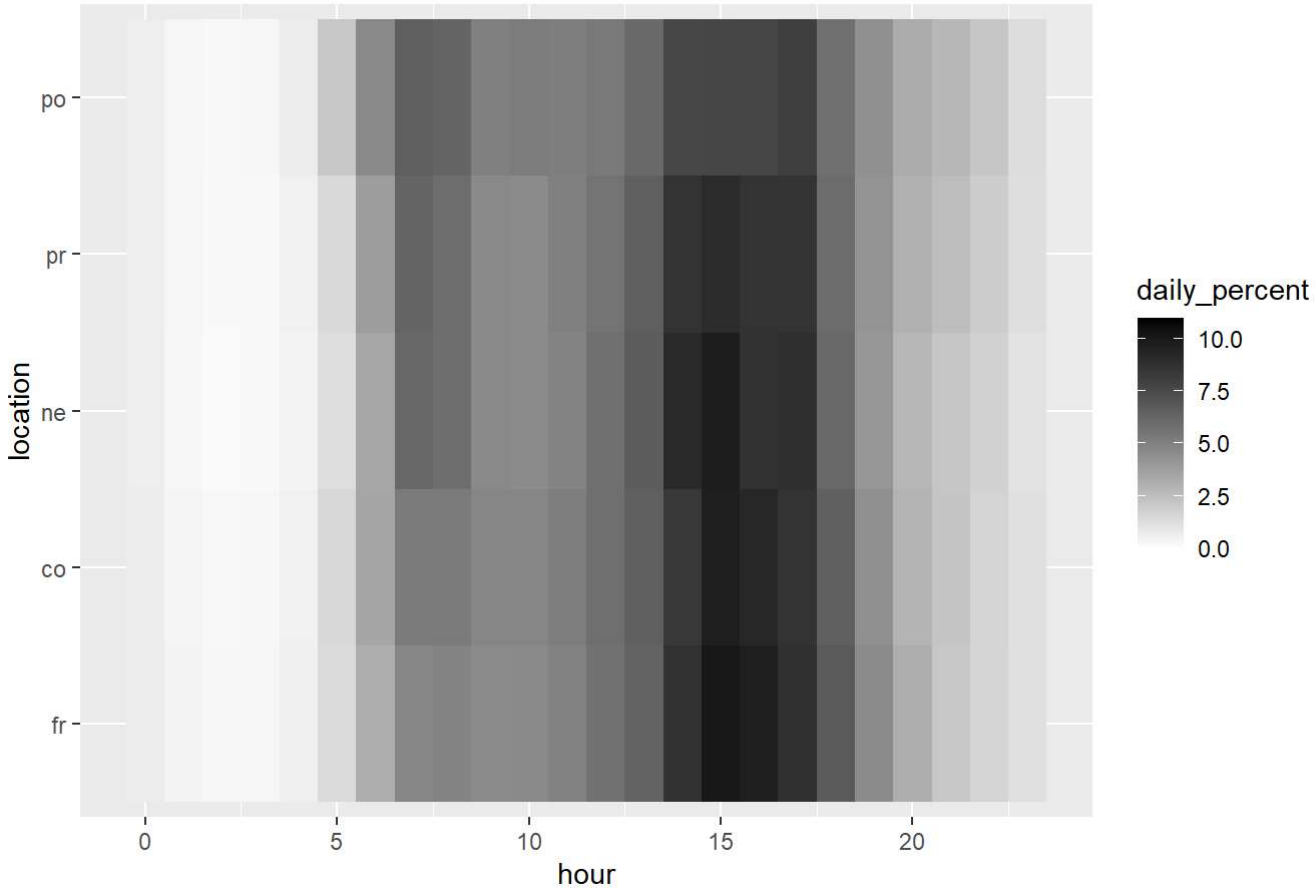

po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[3]]
```

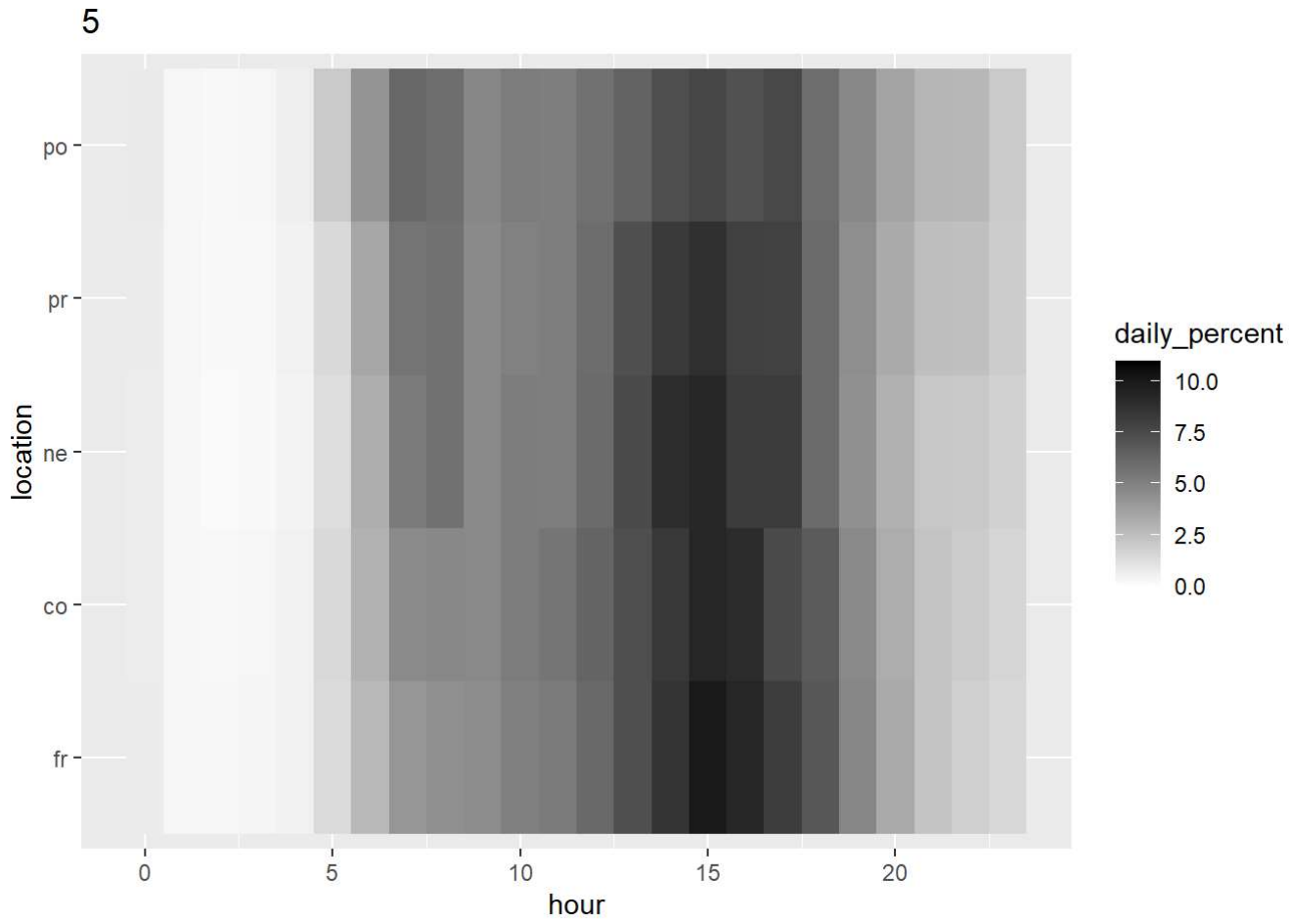

3
po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[4]]
```

4


po
pr
ne
co
fr
location
0
5
10
15
20
hour
daily_percent
10.0
7.5
5.0
2.5
0.0

```
##
## [[5]]
```



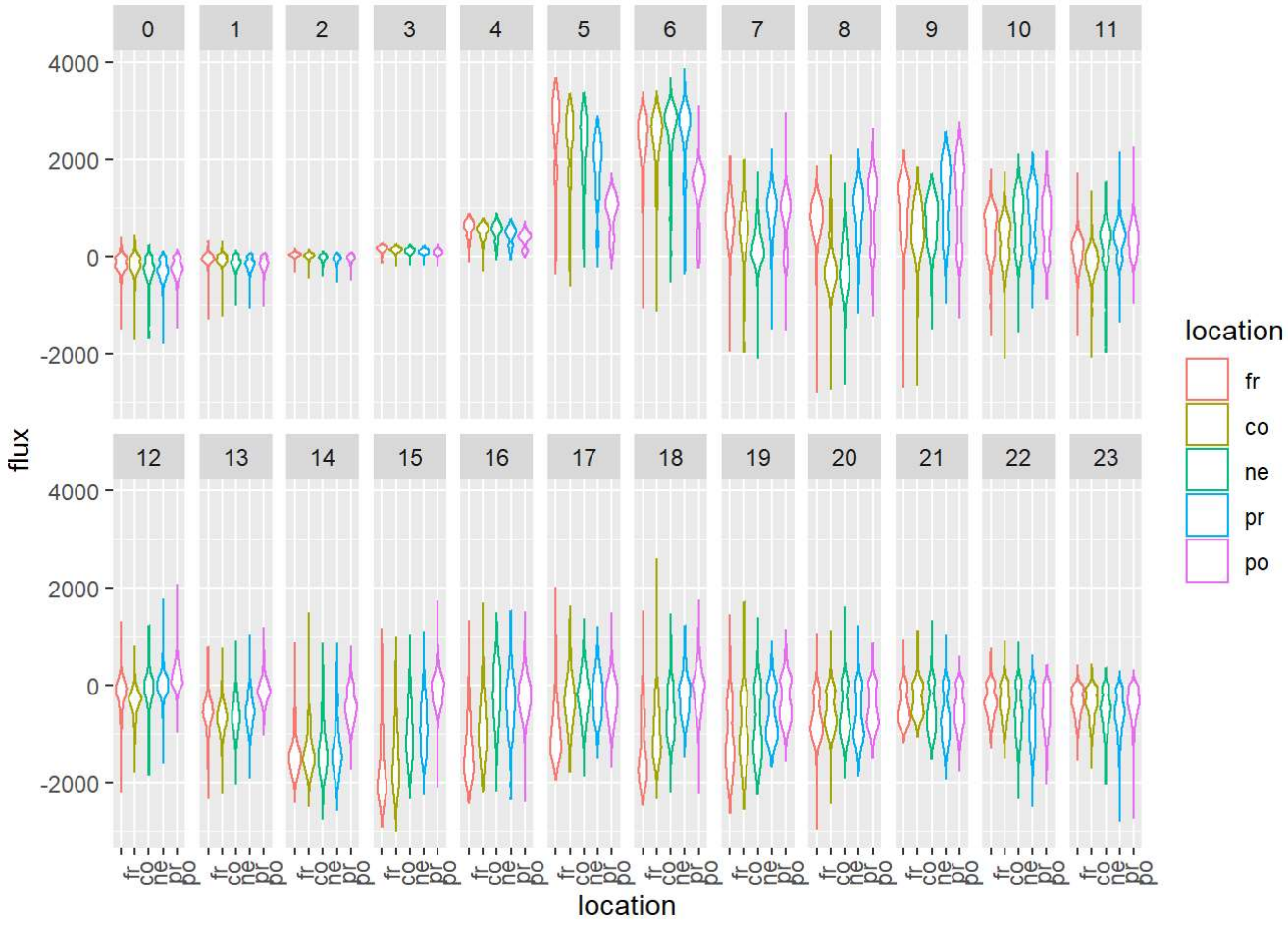
Flux per Hour

Borrowing code used from the Final Project portion for `Flux`

```
pike_master_EB_flux_hour <- pike_master_EB %>%
  mutate(full_date = as.character(full_date)) %>%
  select(day, hour, cars_per_hour, day_of_week, direction, location, full_date)
pike_master_WB_flux_hour <- pike_master_WB %>%
  mutate(full_date = as.character(full_date)) %>%
  select(full_date, day, hour, cars_per_hour, day_of_week, direction, location)


pike_master_flux_per_hour <-
  left_join(pike_master_EB_flux_hour, pike_master_WB_flux_hour, by = c("full_date", "location",
"hour")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(year = year(full_date)) %>%
  mutate(cars_per_hour_EB = cars_per_hour.x) %>%
  mutate(cars_per_hour_WB = cars_per_hour.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (cars_per_hour_EB - cars_per_hour_WB)) %>%
  select(full_date, day, hour, month, year, cars_per_hour_EB, cars_per_hour_WB, day_of_week, loc
ation, flux)
```

I will use violin plots to represent the data.

```
ggplot(pike_master_flux_per_hour, aes(x = location, y = flux, color = location)) +
  geom_violin() +
  facet_wrap(~hour, nrow = 2) +
  theme(axis.text.x = element_text(angle = 90))
```



It looks like Rush “Hour” starts at some point in between 5am and 6am on a typical weekday. It makes sense that Flux is positive in the morning because when more cars traveling East minus less cars traveling West leads to a positive value. The afternoon and evening having negative Flux makes sense because there are lest cars traveling East minus more cars traveling West leads Flux to be negative. Also, it looks like people living further away from Boston tend to leave earlier than those living closer. These graphs are too small to see, so I decide to split them into `AM` and `PM` hours, as what is in the Final Project portion.

Before and After Electronic Tolls on the Mass Pike

Building Datasets for `2017` and `2019`

```
# Setting up the Next Code Chunk
pike_master_EB_11_hour <- pike_master_EB %>%
  filter(hour == 11) %>%
  select(day, hour, Total, day_of_week, direction, location, full_date)
pike_master_WB_11_hour <- pike_master_WB %>%
  filter(hour == 11) %>%
  select(full_date, day, hour, Total, day_of_week, direction, location)


pike_master_EB_11_hour_location <- pike_master_EB_11_hour %>%
  group_by(location)
pike_master_WB_11_hour_location <- pike_master_WB_11_hour %>%
  group_by(location)


pike_master_flux_day <-
  left_join(pike_master_EB_11_hour_location, pike_master_WB_11_hour_location, by = c("full_dat
e", "location")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(hour = hour.x) %>%
  mutate(year = year(full_date)) %>%
  mutate(Total_EB = Total.x) %>%
  mutate(Total_WB = Total.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (Total_EB - Total_WB)) %>%
  mutate(full_date = as.character(full_date)) %>%
  select(full_date, day, hour, month, year, Total_EB, Total_WB, day_of_week, location, flux)
```

Same reasoning as the *Flux per Hour* section.

```
# 2017

# Part 1
pike_master_flux_day_pre_tolls <- pike_master_flux_day %>%
  filter(full_date <= ymd(20181028)) %>%
  mutate(pre_year_total = sum(Total_EB + Total_WB)) %>%
  count(pre_year_total)

# Part 2
pre_tibble <- tibble(
  location = c("fr", "co", "ne", "pr", "po"),
  time = "pre"
) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 3
pike_master_flux_day_pre_tolls <-
  left_join(pike_master_flux_day_pre_tolls, pre_tibble, by = "location")
```

The goal is to get the number of observations and record whether those observations were made before (`pre`) or after (`post`) the implementation of electronic tolls on the Mass Pike. **Part 1:** The `filter()` step includes all days before official end date of making toll booths electronic `(10/28/2018)` . For the `mutate()` step, I am not worried about direction for now. I want to see if there is any initial overall changes. `count()` allows me to know how many observations there were. Since I am totaling both East Bound and West Bound, I would expect n to be less than 730 (365 x 2). **Part 2: I** want to assign a time period, `time` , that corresponds to that these observations were made before (`pre`) electronic tolls. For some reason, when I do not use `fct_relevel` here, the locations are sorted alphabetically instead of based on location. This manually overrides the default. **Part 3:** This was all a round about way to add a column full of `pre` to the `pike_master_flux_day_pre_tolls` tibble. I repeat the process below for `2019` :

```
# Part 1:
pike_master_flux_day_post_tolls <- pike_master_flux_day %>%
  filter(full_date > ymd(20181028)) %>%
  mutate(post_year_total = sum(Total_EB + Total_WB)) %>%
  count(post_year_total)

# Part 2:
post_tibble <- tibble(
  location = c("fr", "co", "ne", "pr", "po"),
  time = "post"
) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 3:
pike_master_flux_day_post_tolls <-
  left_join(pike_master_flux_day_post_tolls, post_tibble, by = "location")
```

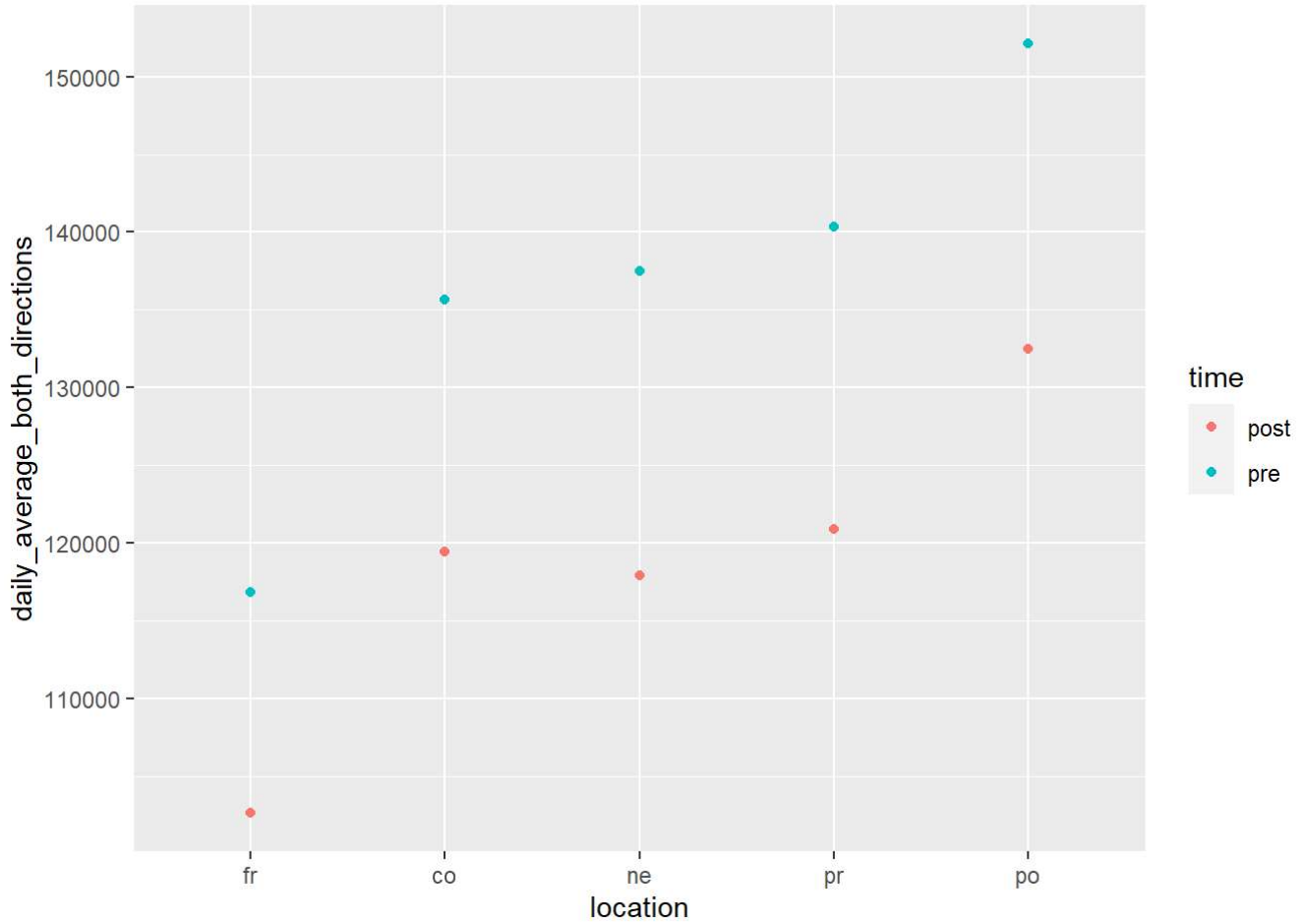
Combining `pre` and `post` Datasets in One Tibble

```
# Part 1
pike_master_flux_toll <- bind_rows(pike_master_flux_day_pre_tolls, pike_master_flux_day_post_tol
ls)

# Part 2
pike_master_flux_toll[is.na(pike_master_flux_toll)] <- 0

# Part 3
pike_master_flux_toll <- pike_master_flux_toll %>%
  mutate(year_total = pre_year_total + post_year_total) %>%
  mutate(daily_average_both_directions = year_total / n) %>%
  select(location, n, year_total, time, daily_average_both_directions)
```

Part 1: Since both `pike_master_flux_day_pre_tolls` and `pike_master_flux_day_post_tolls` both have the same column names, this `bind_rows()` will essentially stack the two tibbles on top of each other, resulting in an easy to use, tidy dataset. **Part 2:** From the previous part, observations made before tolls do not have observations made afterwards. This leaves a bunch of `NA` values. Converting the `NA` values to `0` and then adding that to the total, while keeping `time` as `pre` or `post` will allow me to plot things nice and neat. **Part 3:** Adding `0` (used to be `NA`) to a value will not change the value. This is an attempt to normalize things. There are roughly 70 or so more observations after the tolls, meaning comparing raw numbers at this point is not going to be as helpful as an average.

Plotting Results

```
ggplot(pike_master_flux_toll, aes(x = location, y = daily_average_both_directions, color = tim
e)) +
  geom_point()
```



I am pretty skeptical about these results. Remembering that 2020 has been a dumpster fire of a year because of the COVID pandemic. People have more responsible and have been staying home when they can. Therefore, this extreme year and its extreme values are dragging down the average as some satanic reverse curve. I decided to compare a complete year before (`2017`) and a complete year after (`2019`) implementation in the Final Project.

Looking at the Post Allston Interchange Location in More Detail

Looking at the `po` location, I am having a hard time believing that there are really close to 160000 cars traveling in both directions per day! I'm going to dot my is and cross my ts and make sure this is right.

Preparing to Find the Number of Days Above 75000 Trips in Both

Directions per Day

```
# Part 1
threshold_75k_2017_WB <- pike_master %>%
  filter(location == "po") %>%
  filter(hour == 11) %>%
  filter(year(full_date) == 2017) %>%
  filter(direction == "WB") %>%
  mutate(days_over_75k = (Total > 75000)) %>%
  drop_na()

# Part 2
as.integer(as.logical(threshold_75k_2017_WB$days_over_75k)) %>%
  sum()
```

```
## [1] 105
```

Part 1: The first `filter()` step is ooking at the `po` location only. The second `filter()` step is obtaining one Total number of cars per day. The third `filter()` step is looking at 2017 first. The final `filter()` step is looking at one direction for now (West Bound). The `mutate()` step selects days where traffic exceeded 75000 cars and returns `TRUE` or `FALSE`. Then I remove `NA` values from my total to avoid them messing things up later. **Part 2:** Converting TRUE to 1 and FALSE to 0. The `sum()` is looking at the number of days above my arbitrary threshold, `75000`. There were 105 days with the number of trips West Bound exceeded 75000 in 2017.

Looking at Each Year and Direction

This part repeats the steps from above in brute force hard code because I spent over an hour attempting to write functions, if statements, for loops, and I have to call it a day if it is only going to make it in the Supplemental Materials section where it is likely to be overlooked.

```
# 2017, East Bound
threshold_75k_2017_EB <- pike_master %>%
  filter(location == "po") %>%
  filter(hour == 11) %>%
  filter(year(full_date) == 2017) %>%
  filter(direction == "EB") %>%
  mutate(days_over_75k = (Total > 75000)) %>%
  drop_na()

as.integer(as.logical(threshold_75k_2017_EB$days_over_75k)) %>%
  sum()
```

```
## [1] 185
```

```
# 2019, West Bound
threshold_75k_2019_WB <- pike_master %>%
  filter(location == "po") %>%
  filter(hour == 11) %>%
  filter(year(full_date) == 2019) %>%
  filter(direction == "WB") %>%
  mutate(days_over_75k = (Total > 75000)) %>%
  drop_na()

as.integer(as.logical(threshold_75k_2019_WB$days_over_75k)) %>%
  sum()
```

```
## [1] 219
```

```
# 2019, East Bound
threshold_75k_2019_EB <- pike_master %>%
  filter(location == "po") %>%
  filter(hour == 11) %>%
  filter(year(full_date) == 2019) %>%
  filter(direction == "EB") %>%
  mutate(days_over_75k = (Total > 75000)) %>%
  drop_na()

as.integer(as.logical(threshold_75k_2019_EB$days_over_75k)) %>%
  sum()
```

```
## [1] 237
```

Summary: There were 185 days with the number of trips East Bound exceeded `75000` in `2017`. There were 219 days with the number of trips West Bound exceeded 75000 in `2019`. There were 237 days with the number of trips East Bound exceeded `75000` in `2019`. There was almost a twofold increase between `2017` and `2019`. The fact that the average is about `150000` is not as surprising and hard to believe as I initially thought.

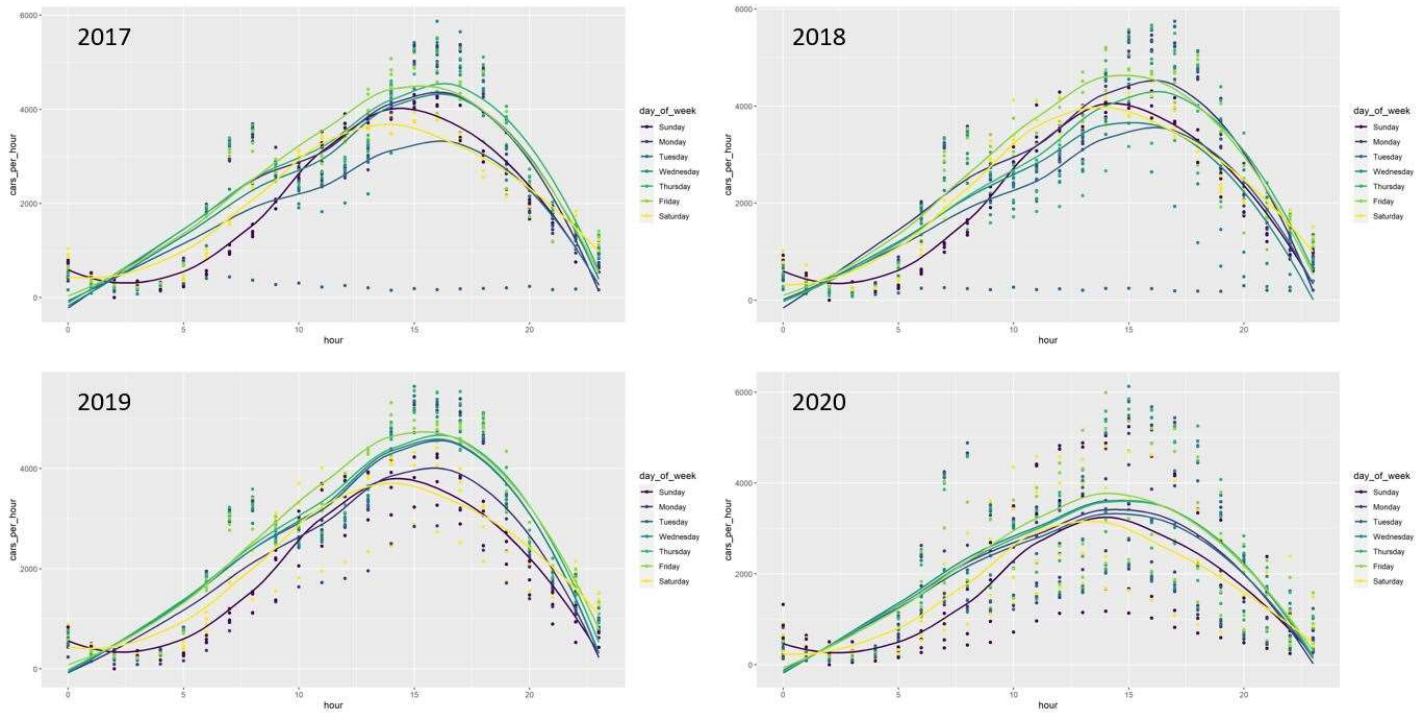
Coronavirus Pandemic (from saved images)

```
for (i in 1:length(files)){
  file_list[[i]] %>%
    ggplot() +
    aes(x = hour, y = cars_per_hour, color = day_of_week) +
    geom_point() +
    geom_smooth(se = FALSE)
  ggsave(filename = paste(location[i], years[i], months[i], ".png", sep = " "))
}
```

This function makes a plot for each file downloaded and saves it as an image.

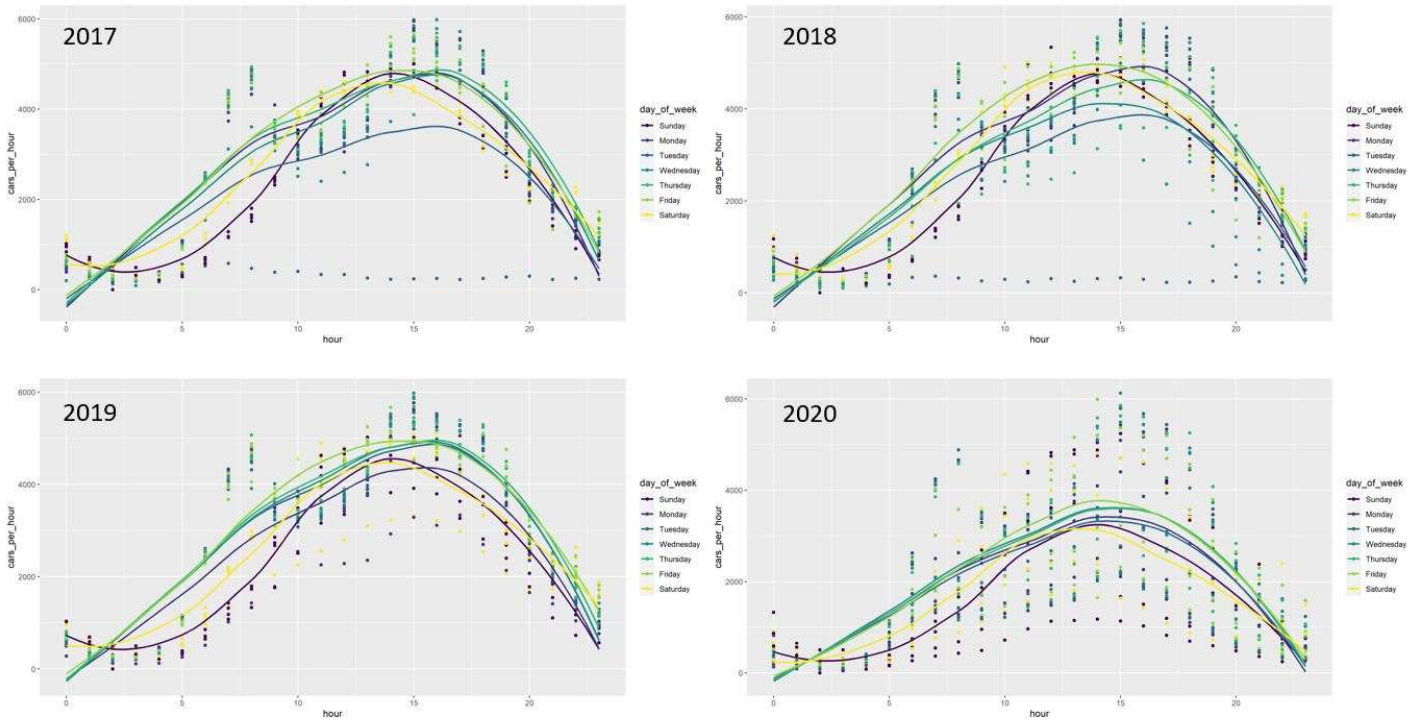
Framingham

March



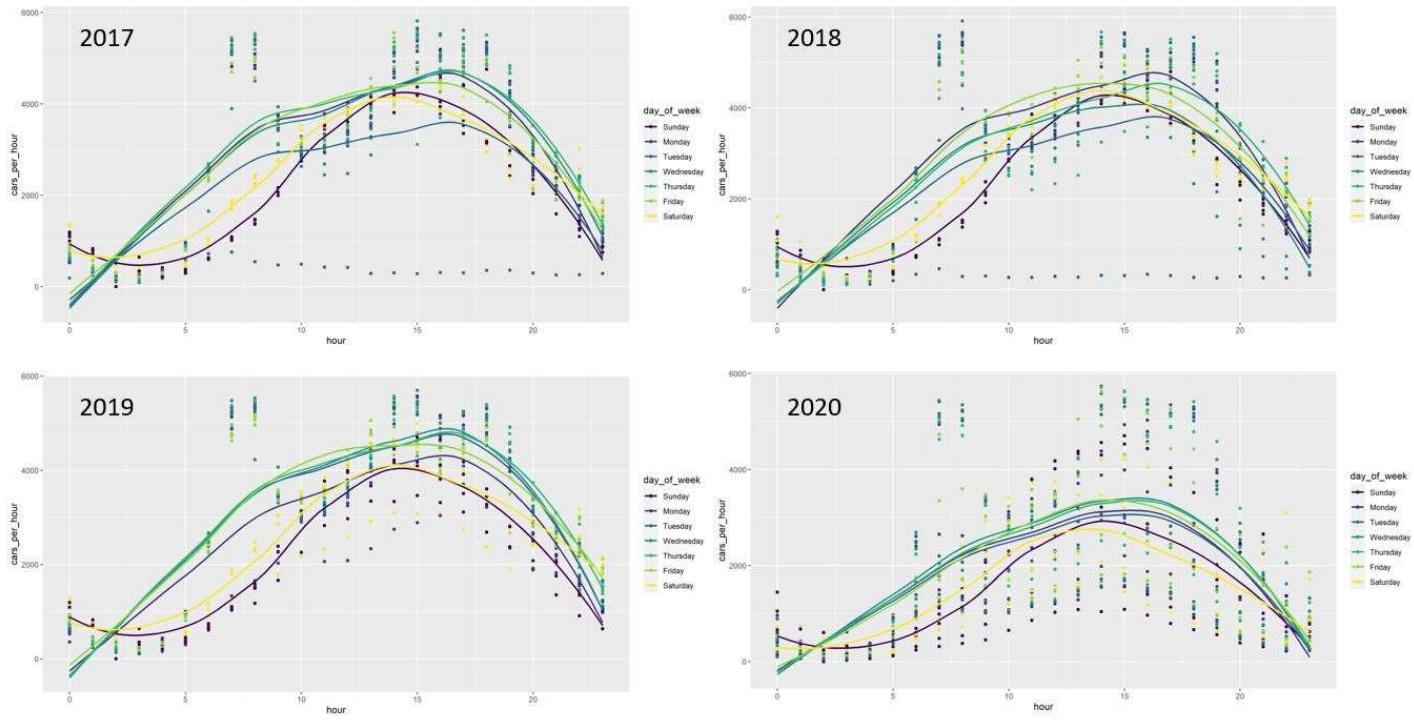
Cochituate

March



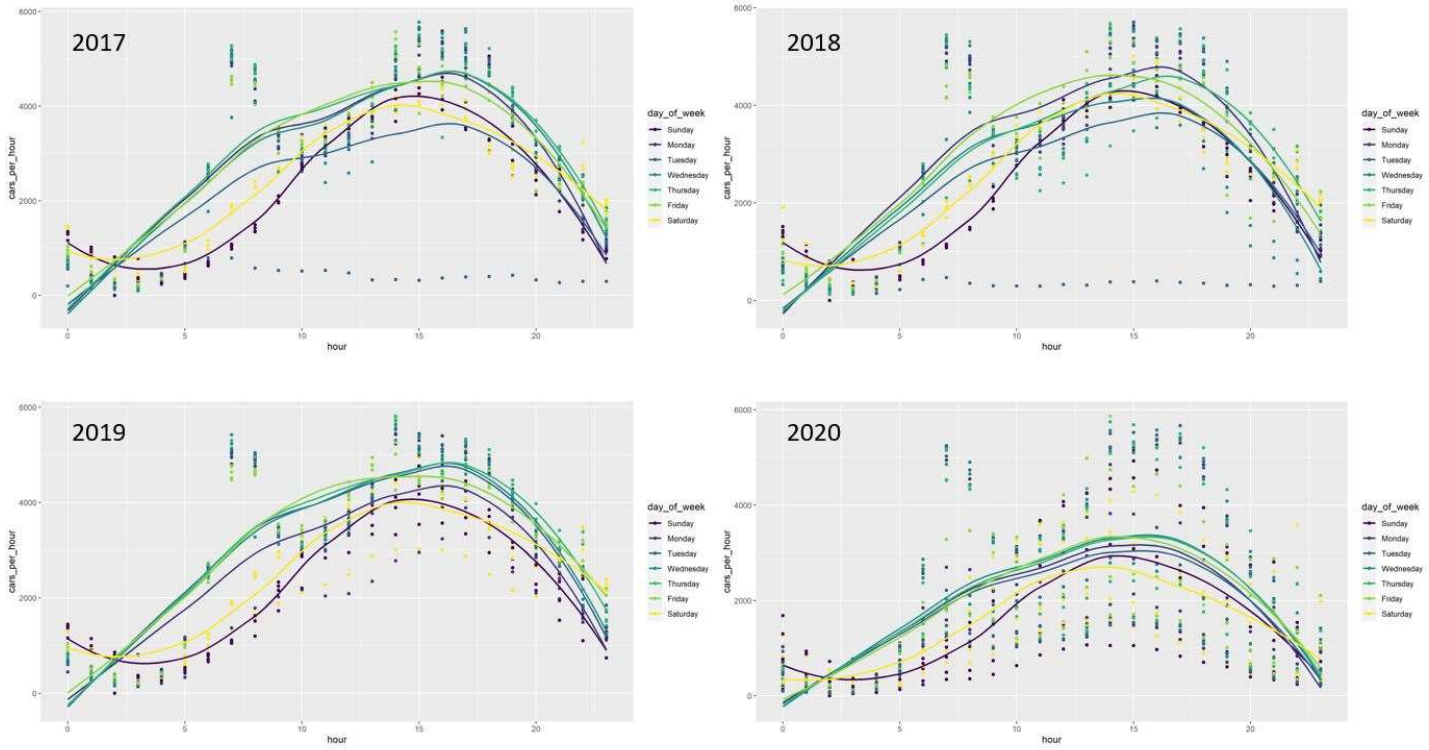
Newton

March



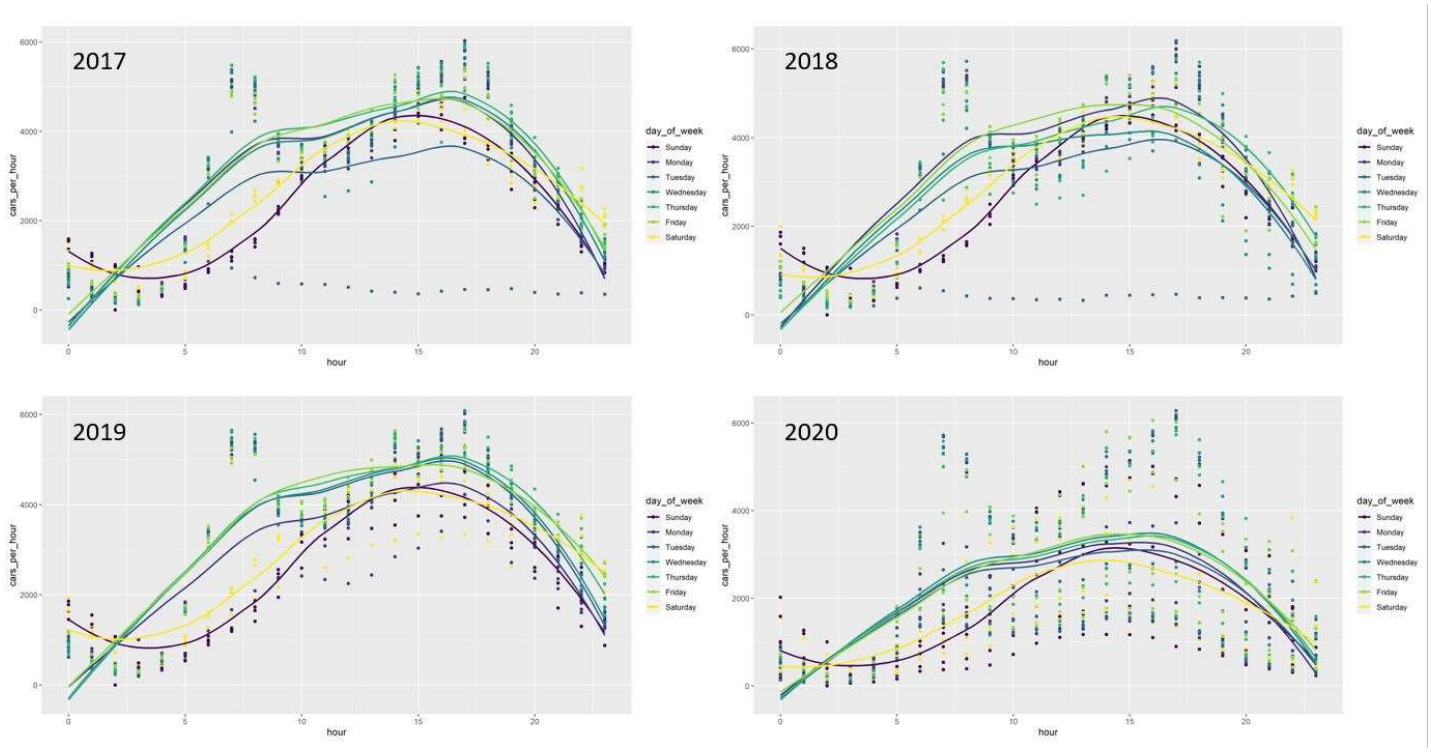
Pre Allston Interchange

March



Post Allston Interchange

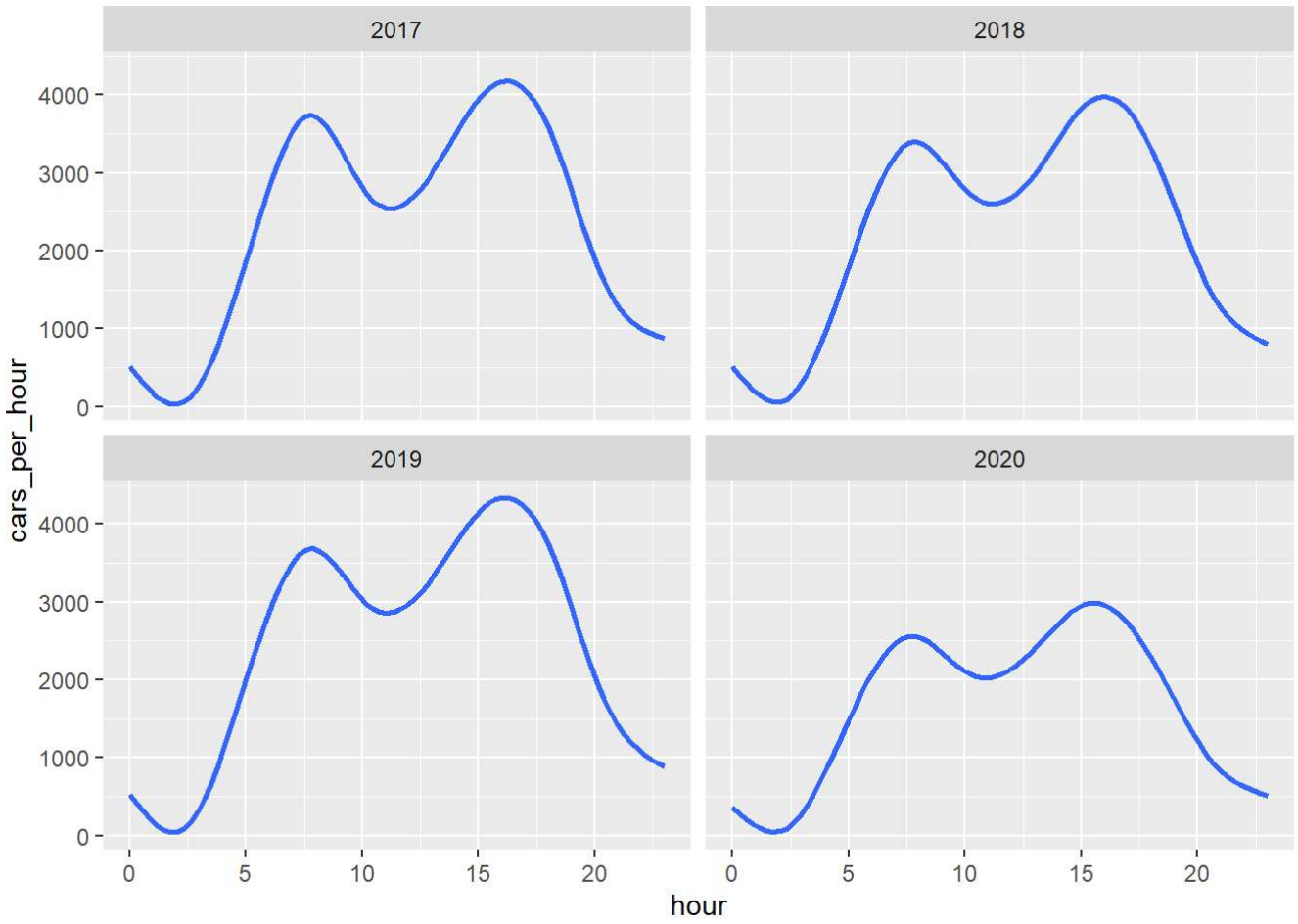
March



Coronavirus Pandemic Visualized Through `ggplot`

Framingham in March

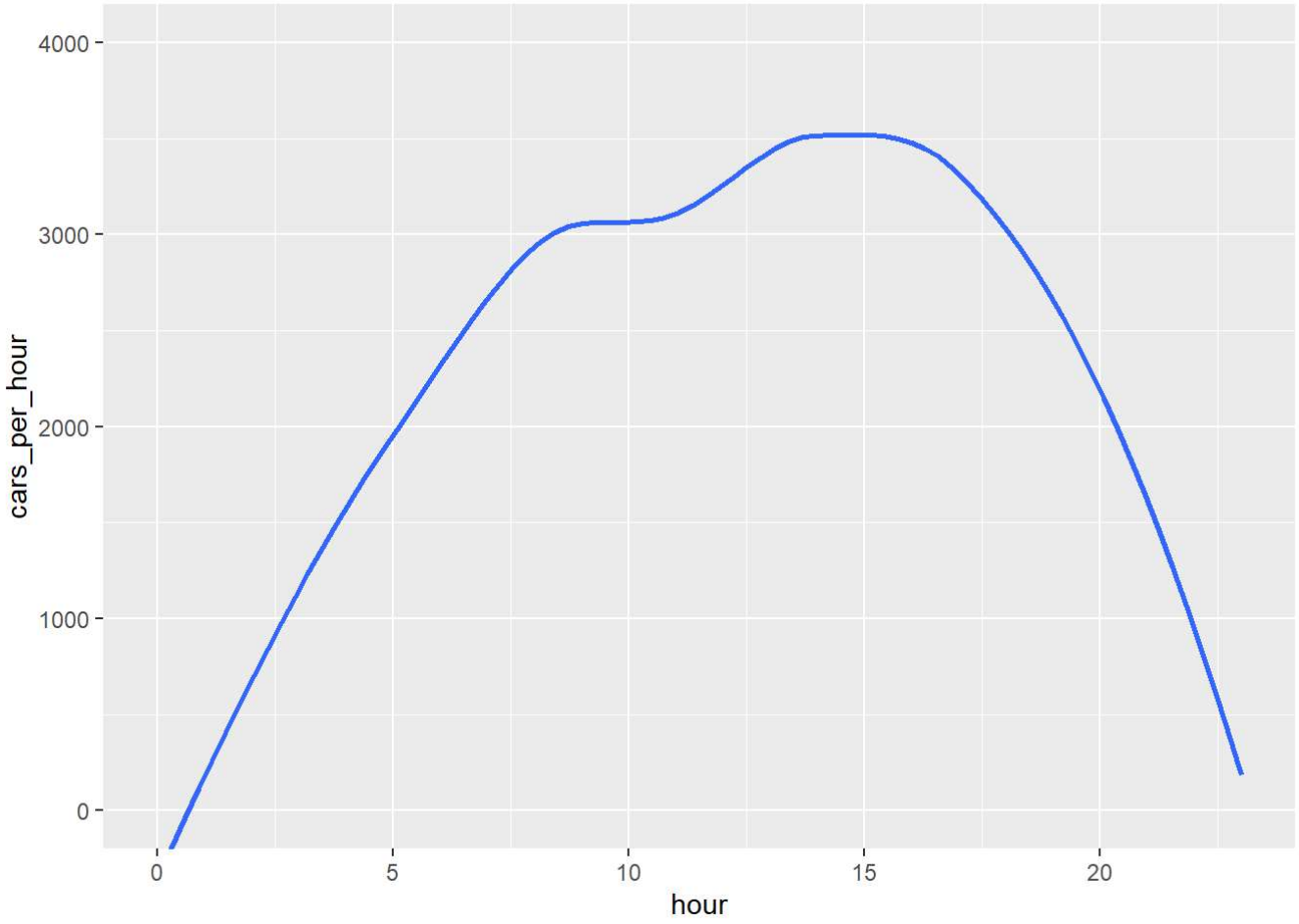
```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "fr",
         month(full_date) == 3
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



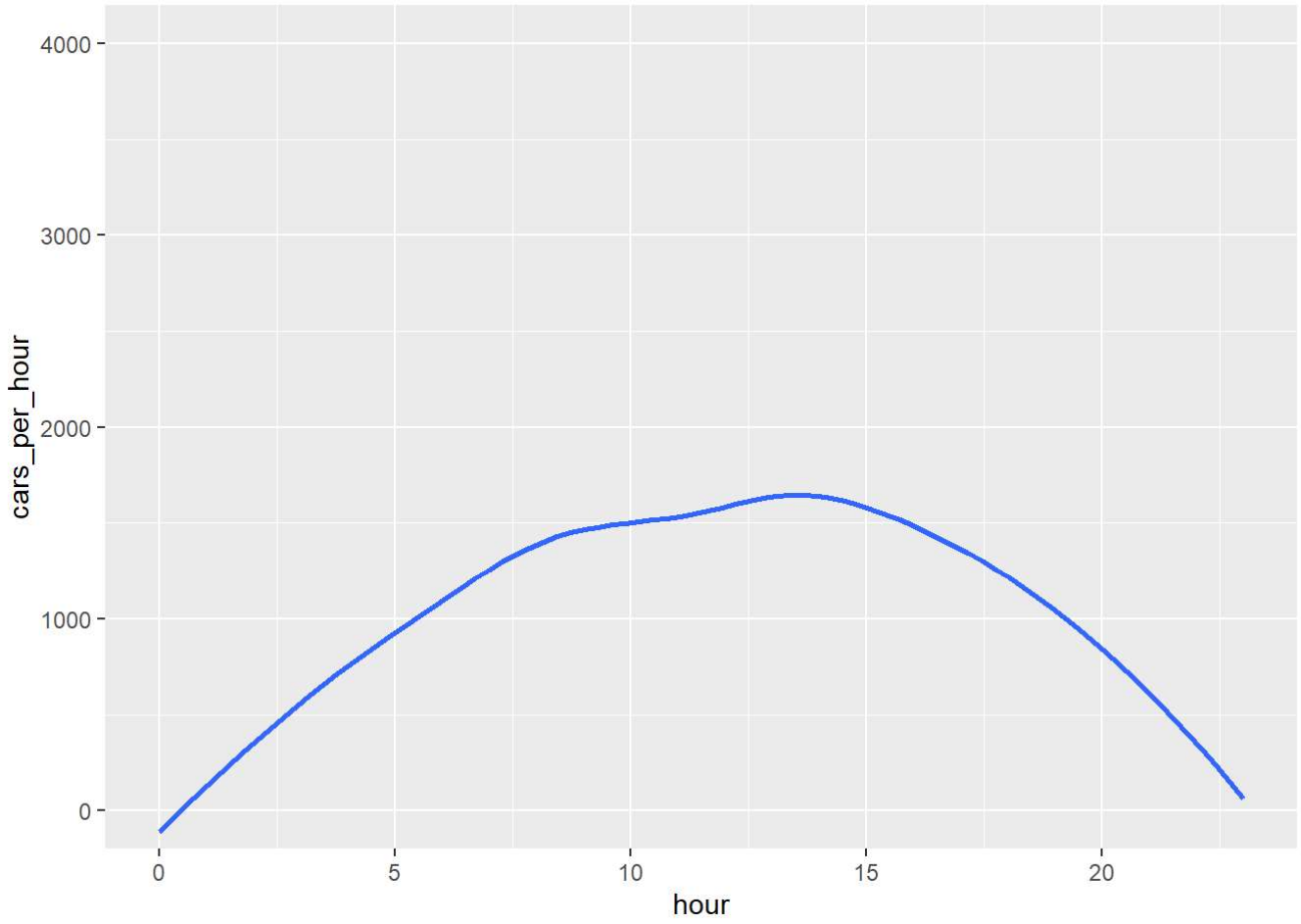
There is a fairly wide spread in the month of March. This might be due to the COVID-19 State of Emergency (https://www.mass.gov/info-details/covid-19-state-of-emergency) issued in the middle of the month on March 18th, encouraging people to work from home.

Before and After State of Emergency Issued

```
# Before
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "fr",
         full_date > ymd(20200301) & full_date < ymd(20200318)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```

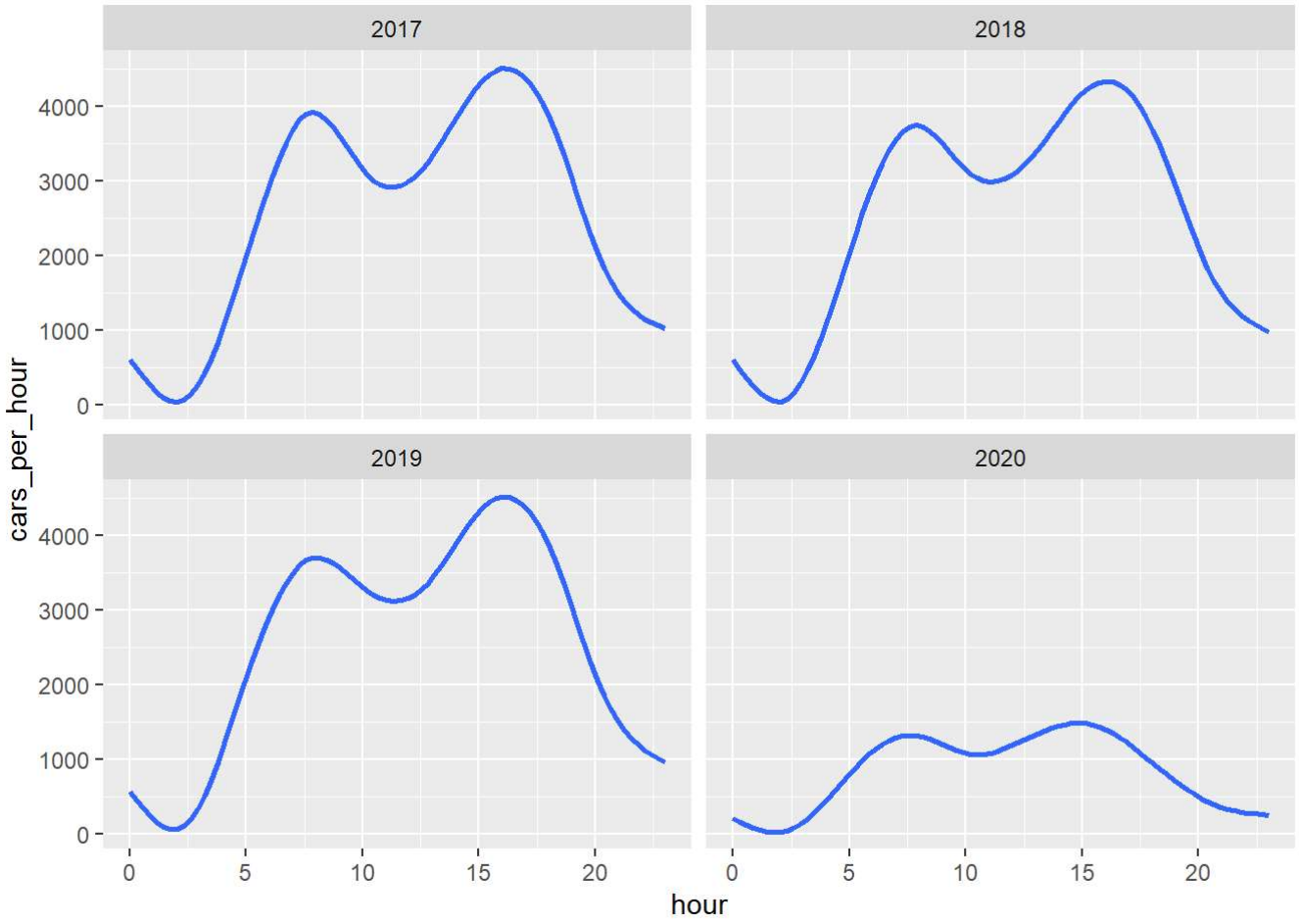



```
# After
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "fr",
         full_date >= ymd(20200318) & full_date <= ymd(20200331)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```



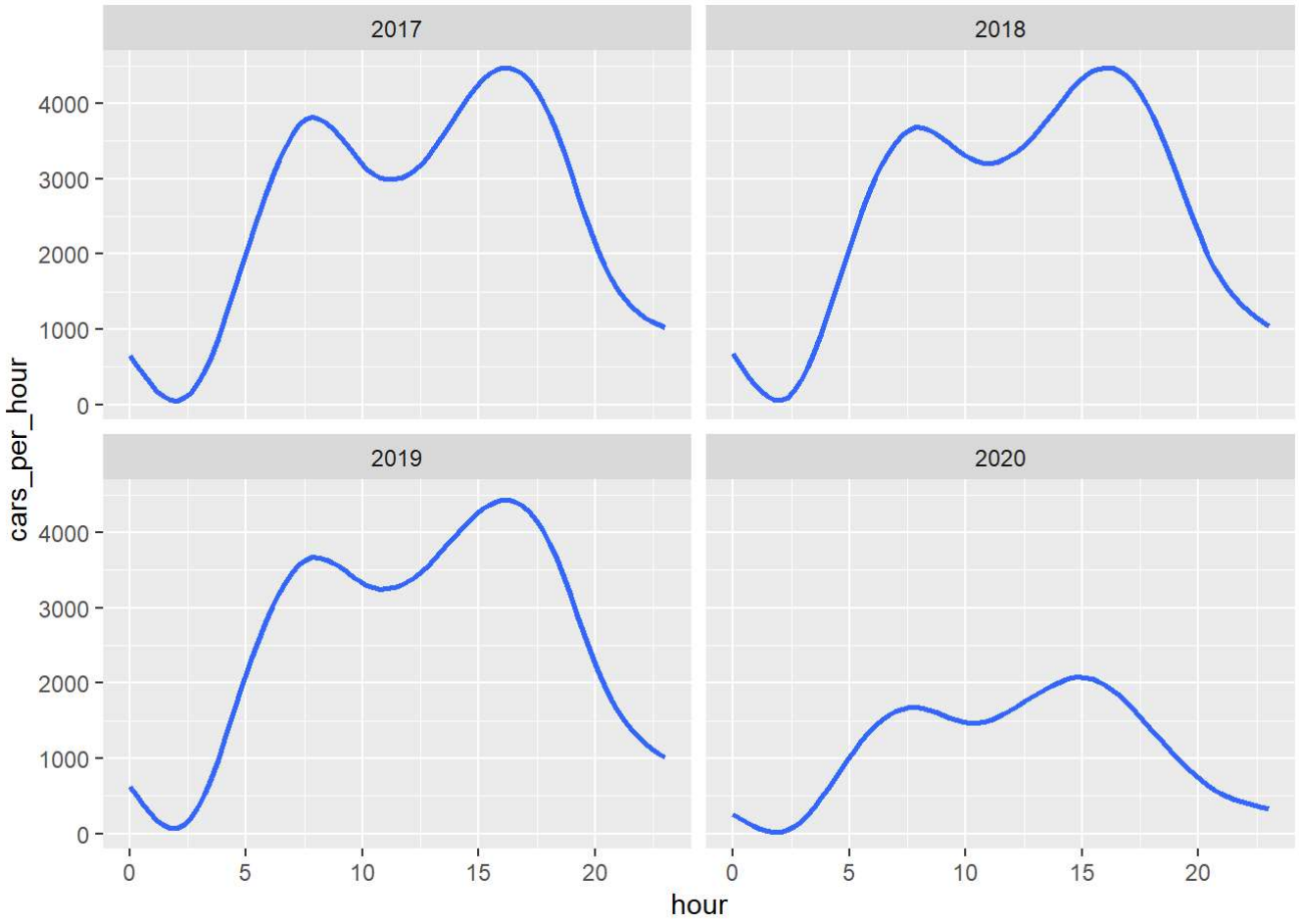
Framingham in April

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "fr",
         month(full_date) == 4
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



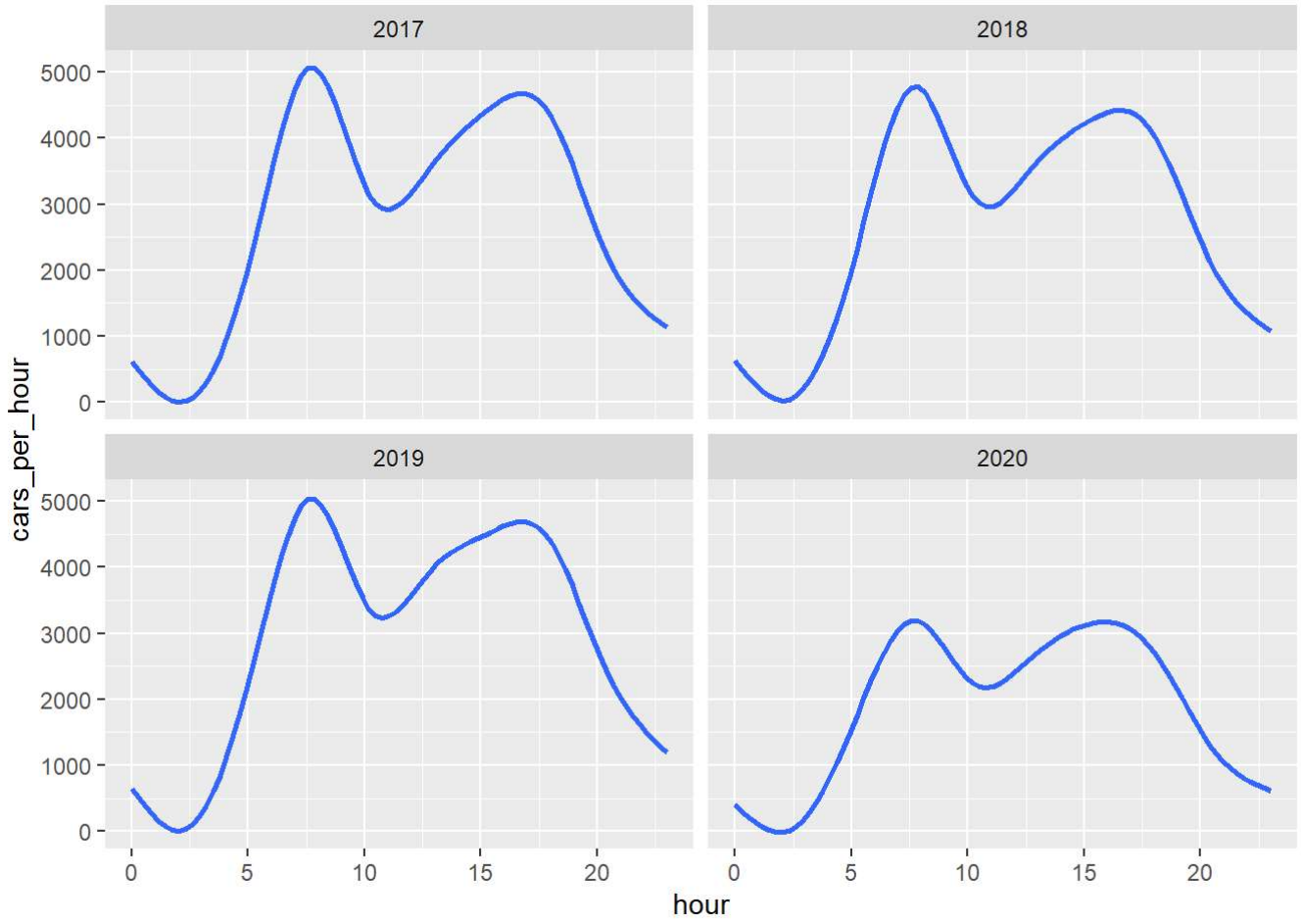
Framingham in May

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
       location == "fr",
       month(full_date) == 5
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



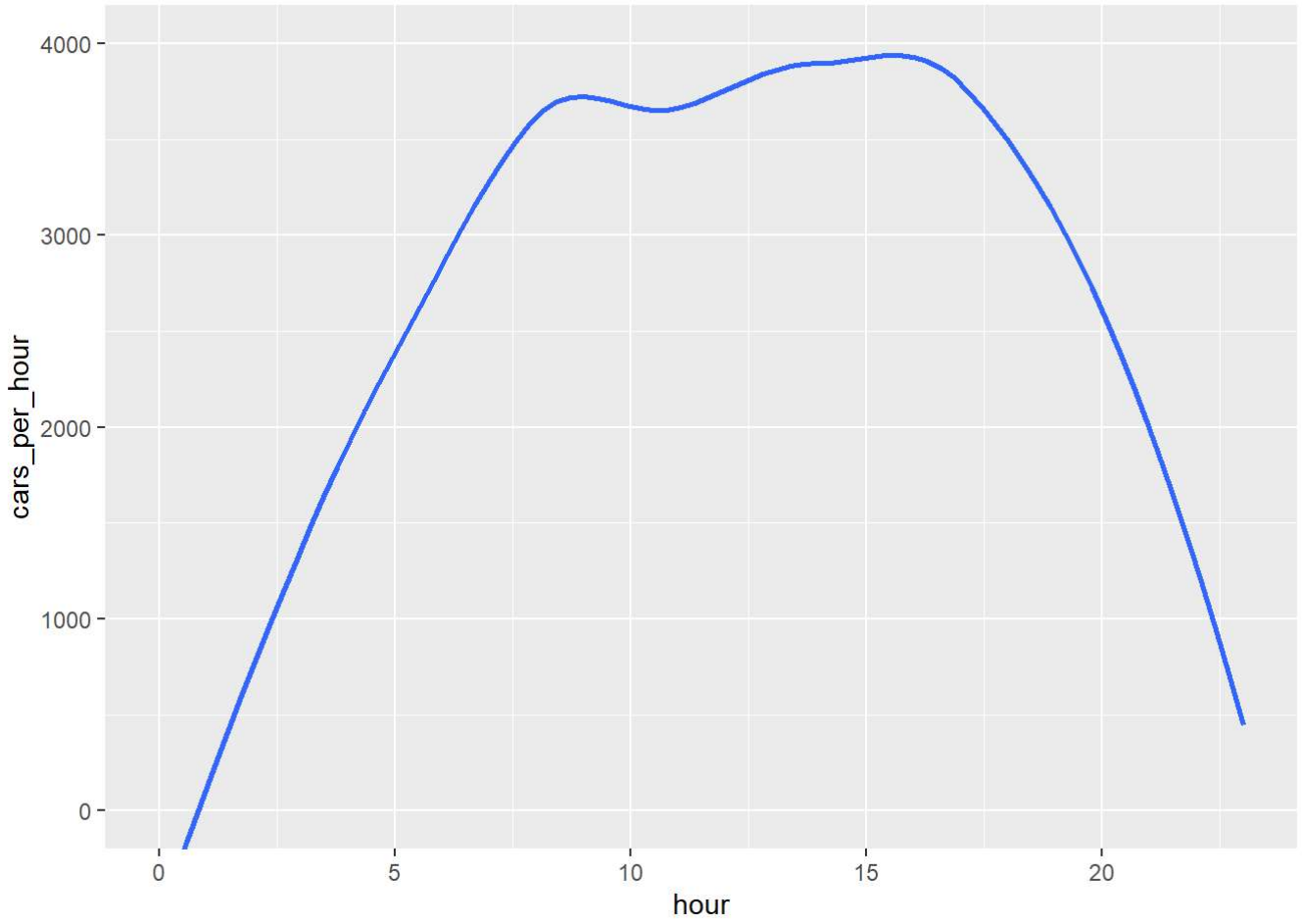
Newton in March

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "ne",
         month(full_date) == 3
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```

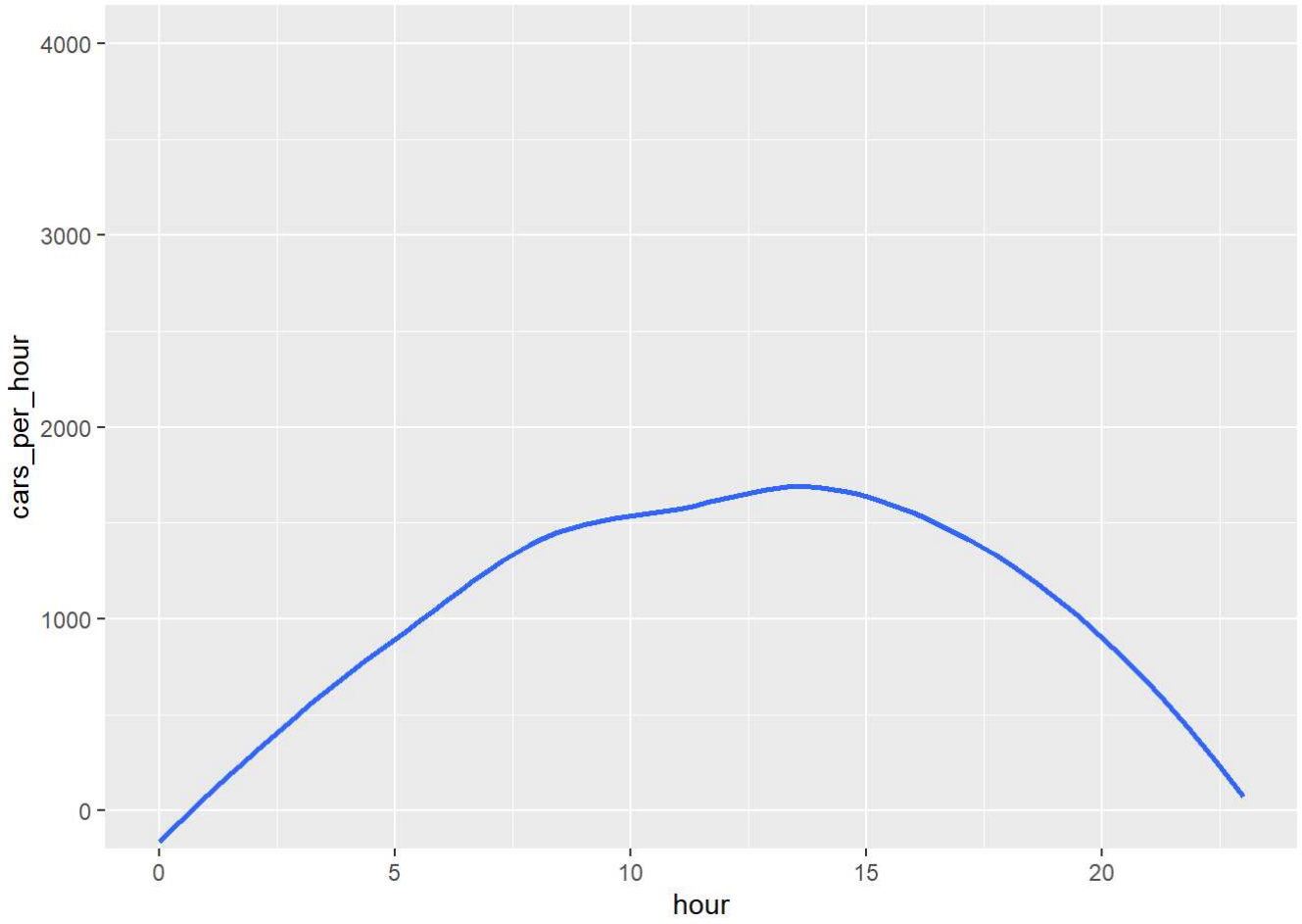


Before and After State of Emergency Issued

```
# Before
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "ne",
         full_date > ymd(20200301) & full_date < ymd(20200318)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```

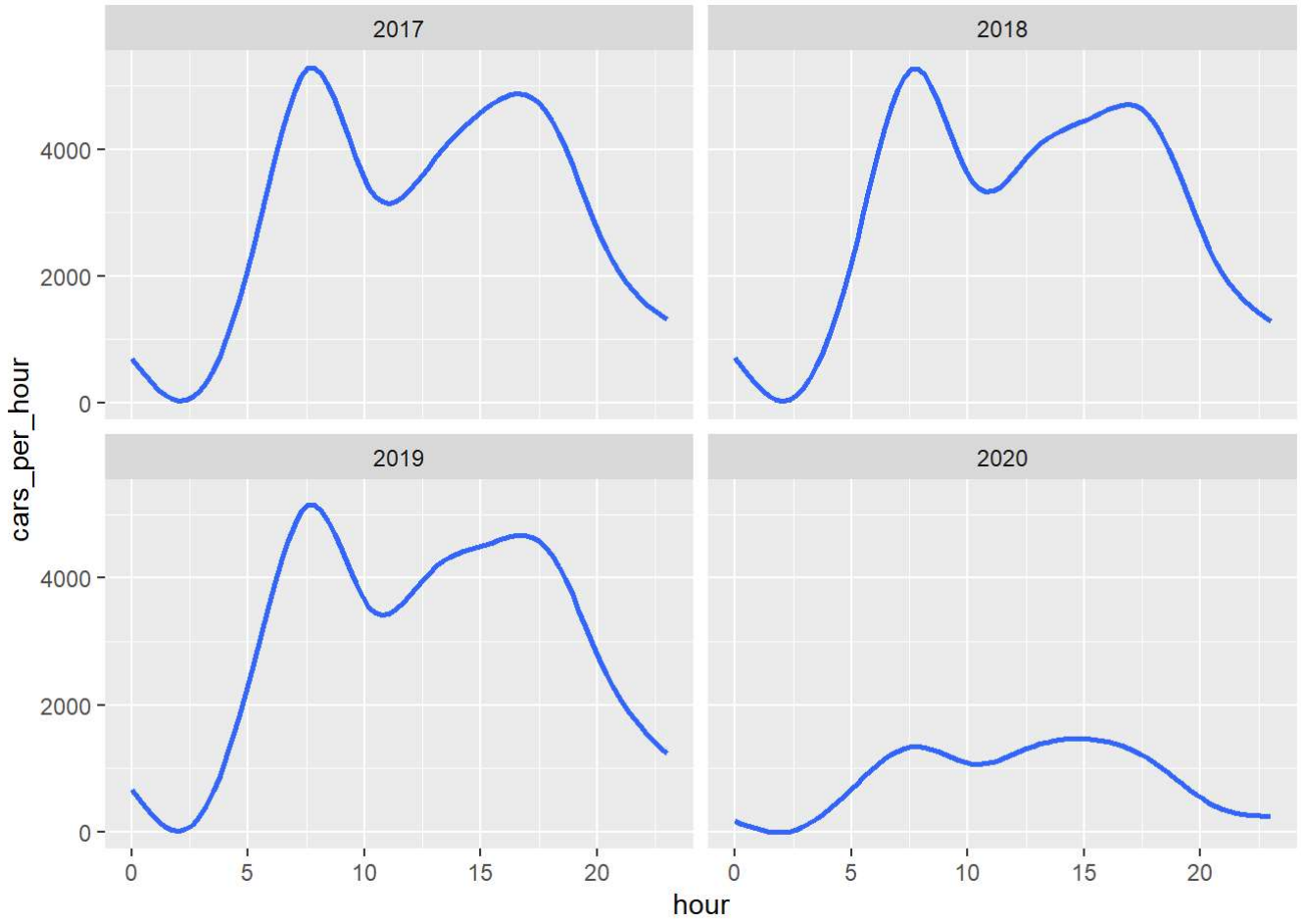



```
# After
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "ne",
         full_date >= ymd(20200318) & full_date <= ymd(20200331)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```



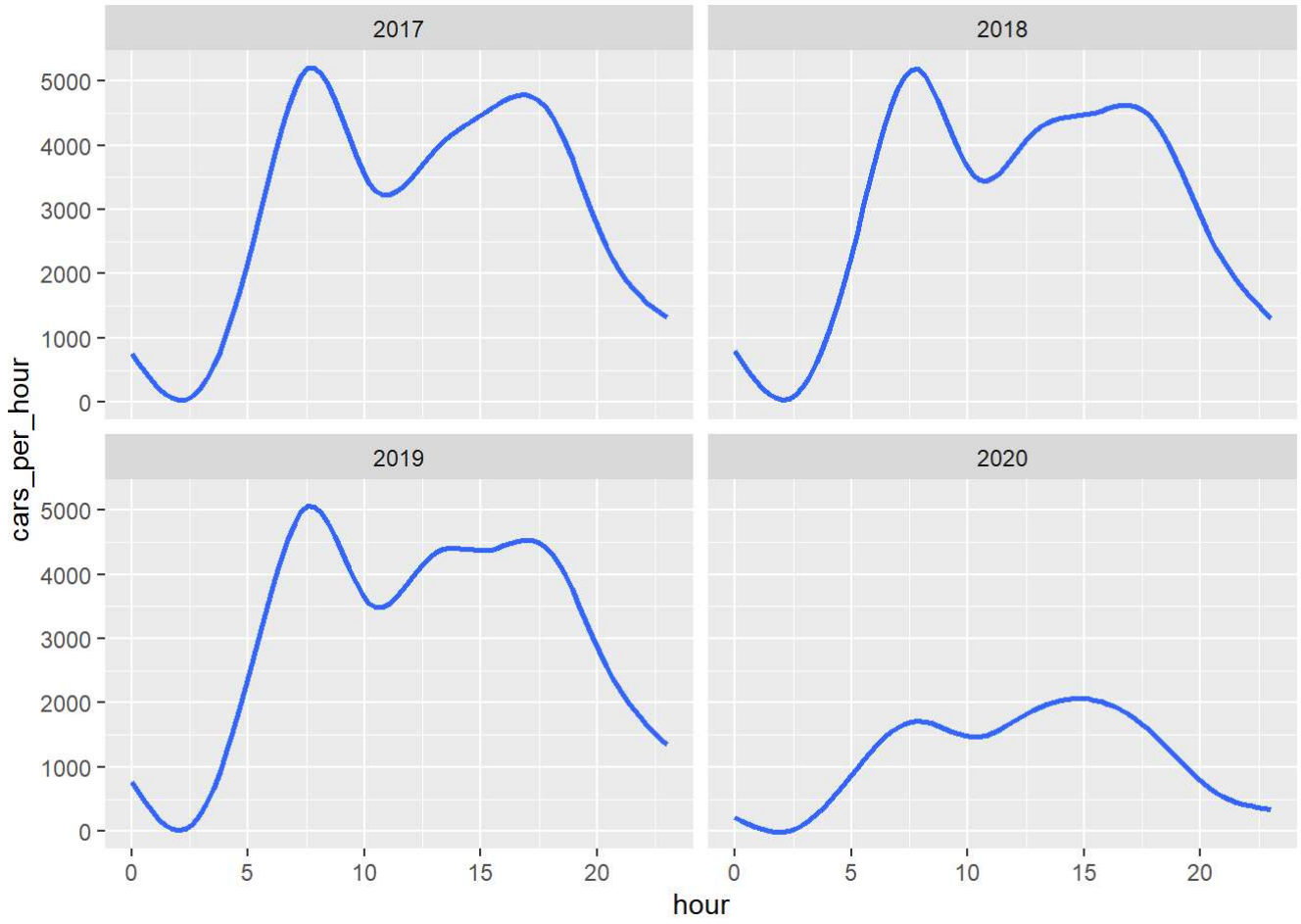
Newton in April

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "ne",
         month(full_date) == 4
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



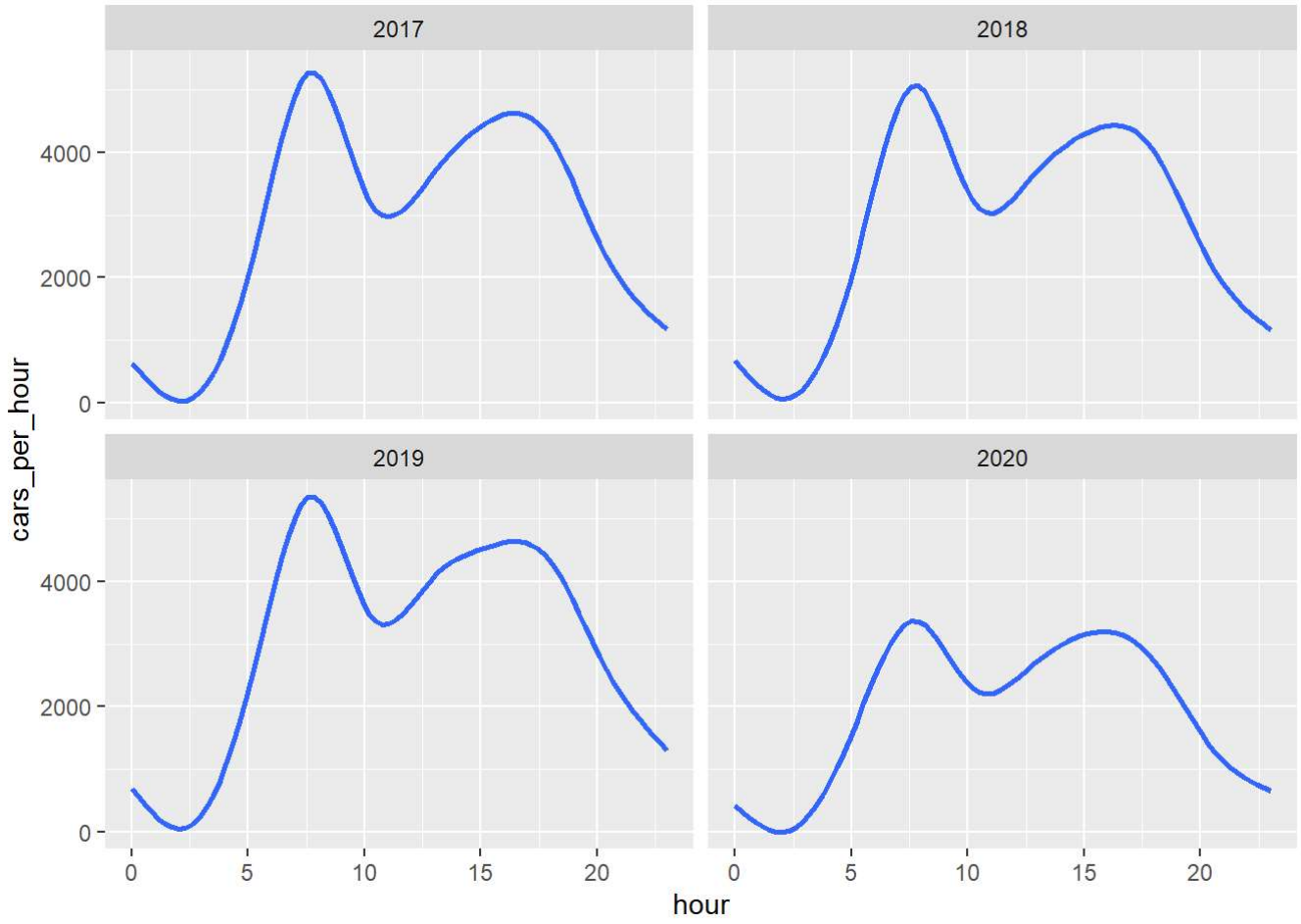
Newton in May

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
       location == "ne",
       month(full_date) == 5
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



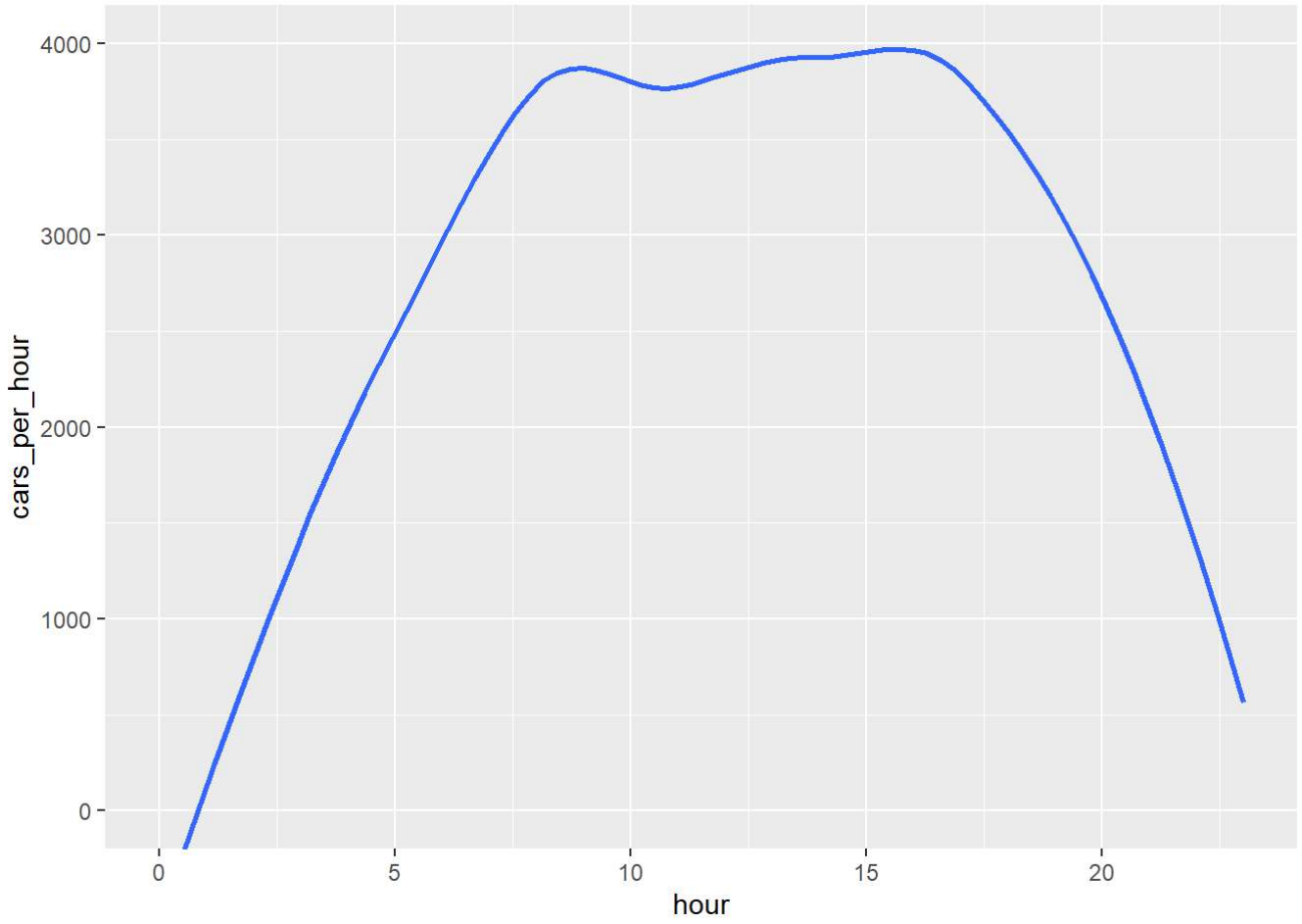
Pre Allston Interchange in March

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "pr",
         month(full_date) == 3
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```

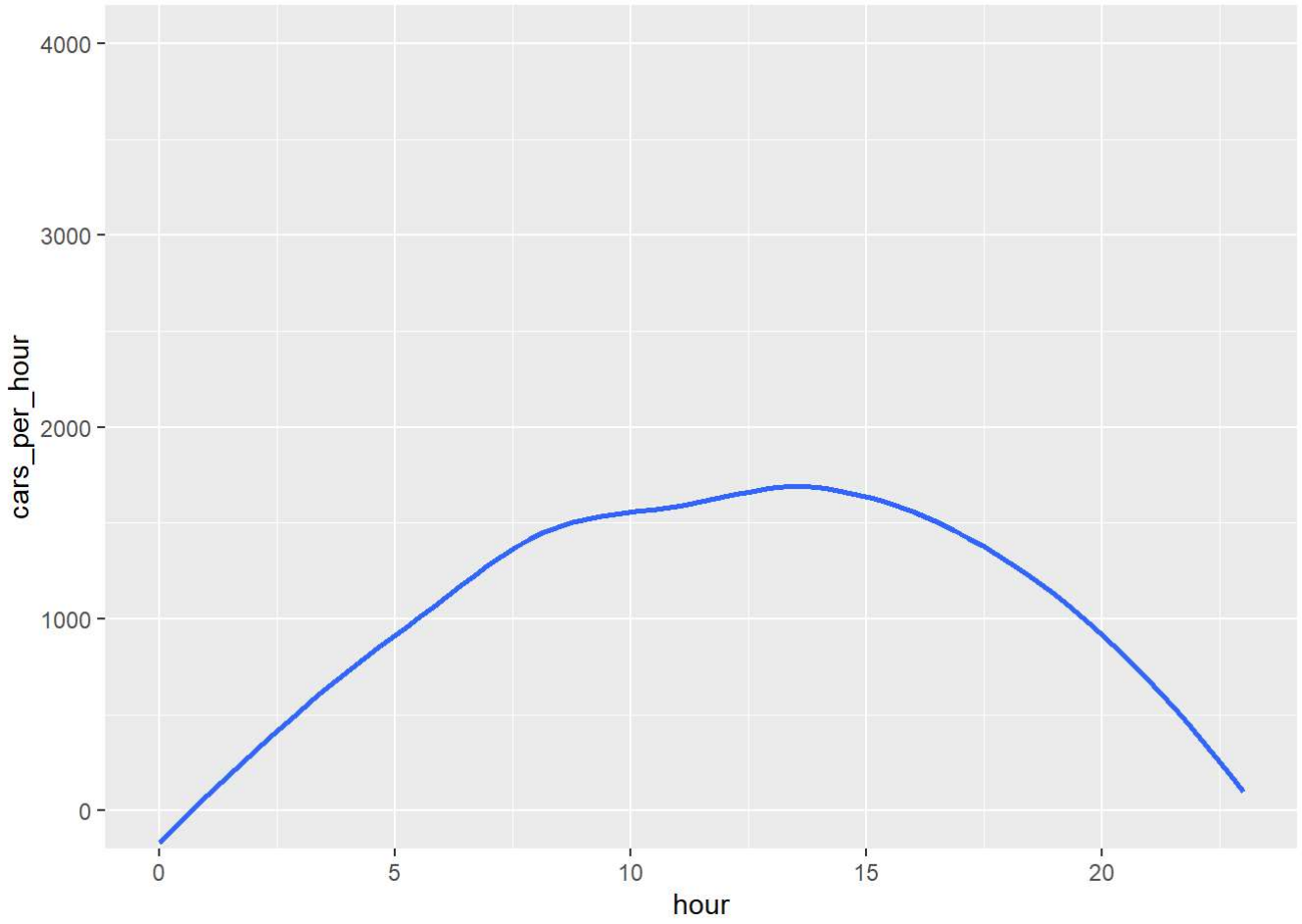


Before and After State of Emergency Issued

```
# Before
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "pr",
         full_date > ymd(20200301) & full_date < ymd(20200318)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```

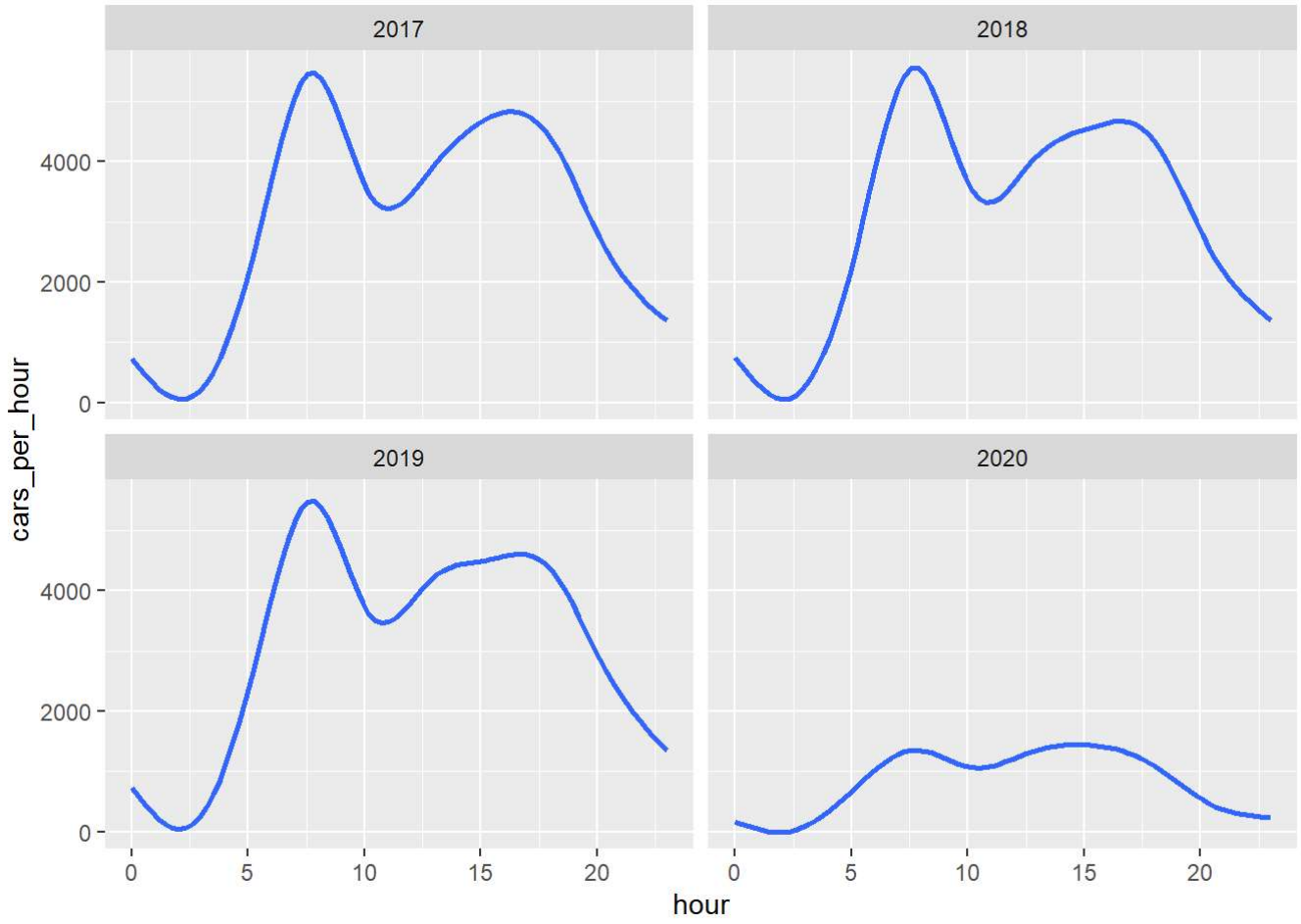



```
# After
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "pr",
         full_date >= ymd(20200318) & full_date <= ymd(20200331)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```



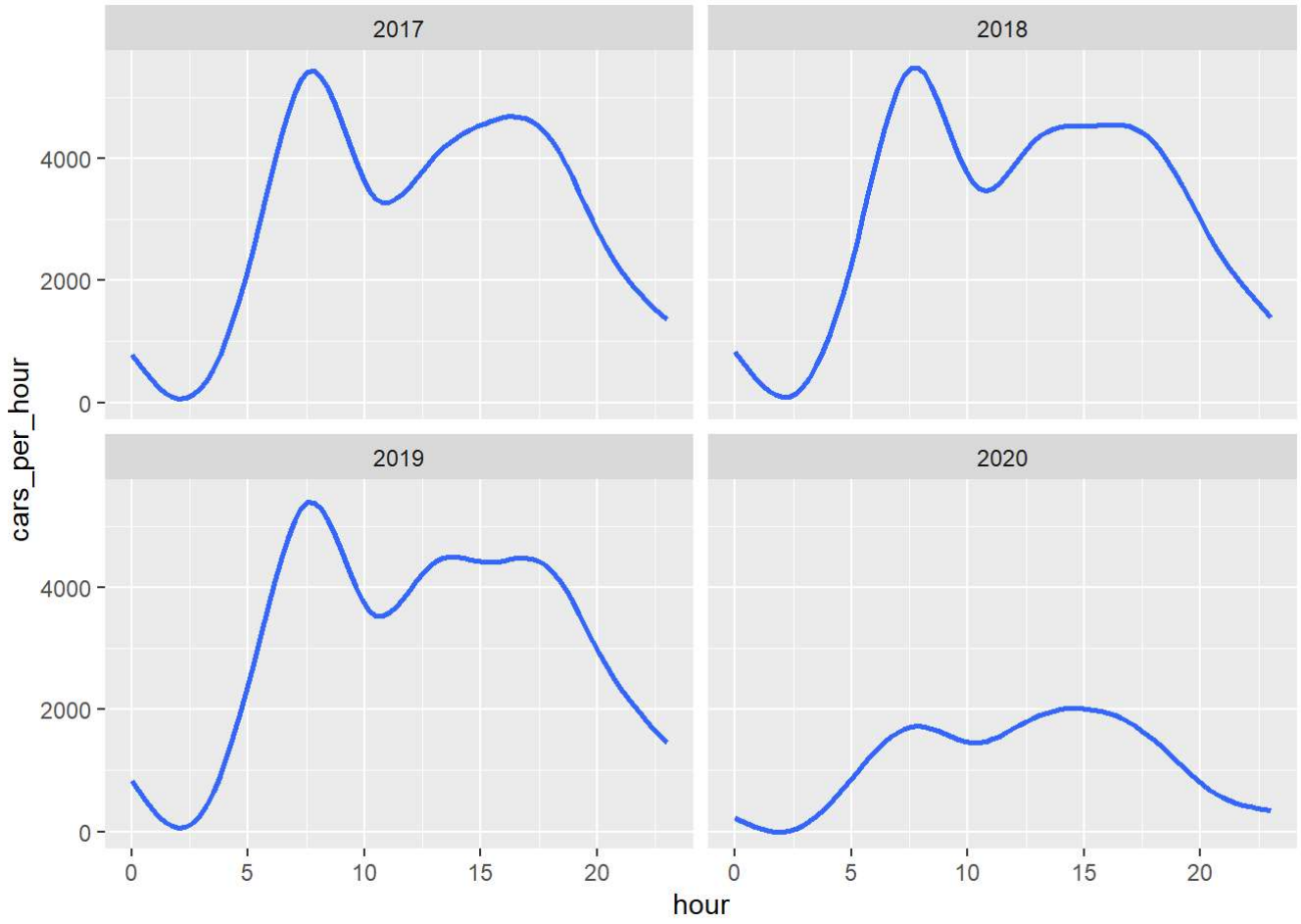
Pre Allston Interchange in April

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "pr",
         month(full_date) == 4
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



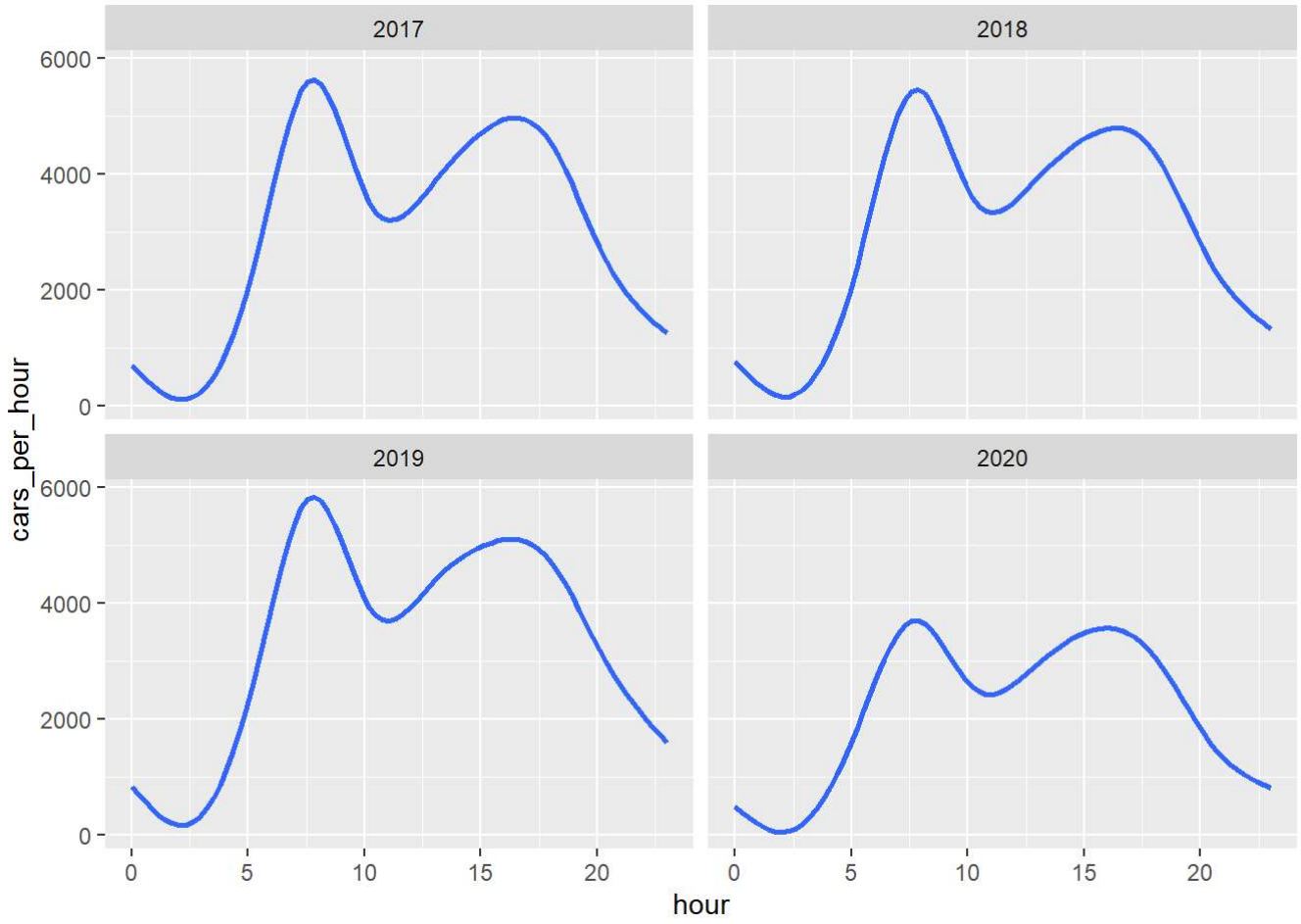
Pre Allston Interchange in May

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
       location == "pr",
       month(full_date) == 5
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



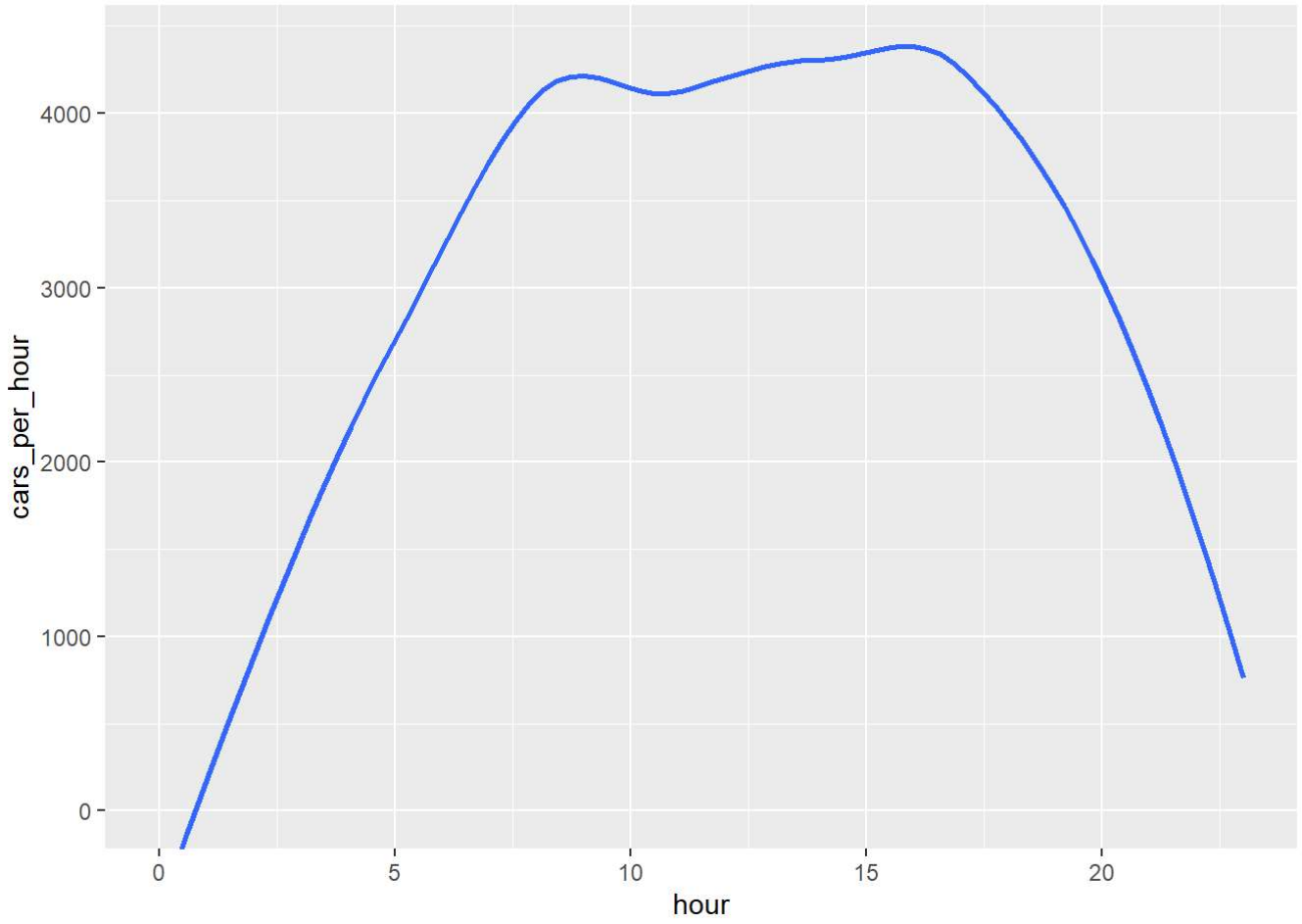
Post Allston Interchange in March

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "po",
         month(full_date) == 3
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```

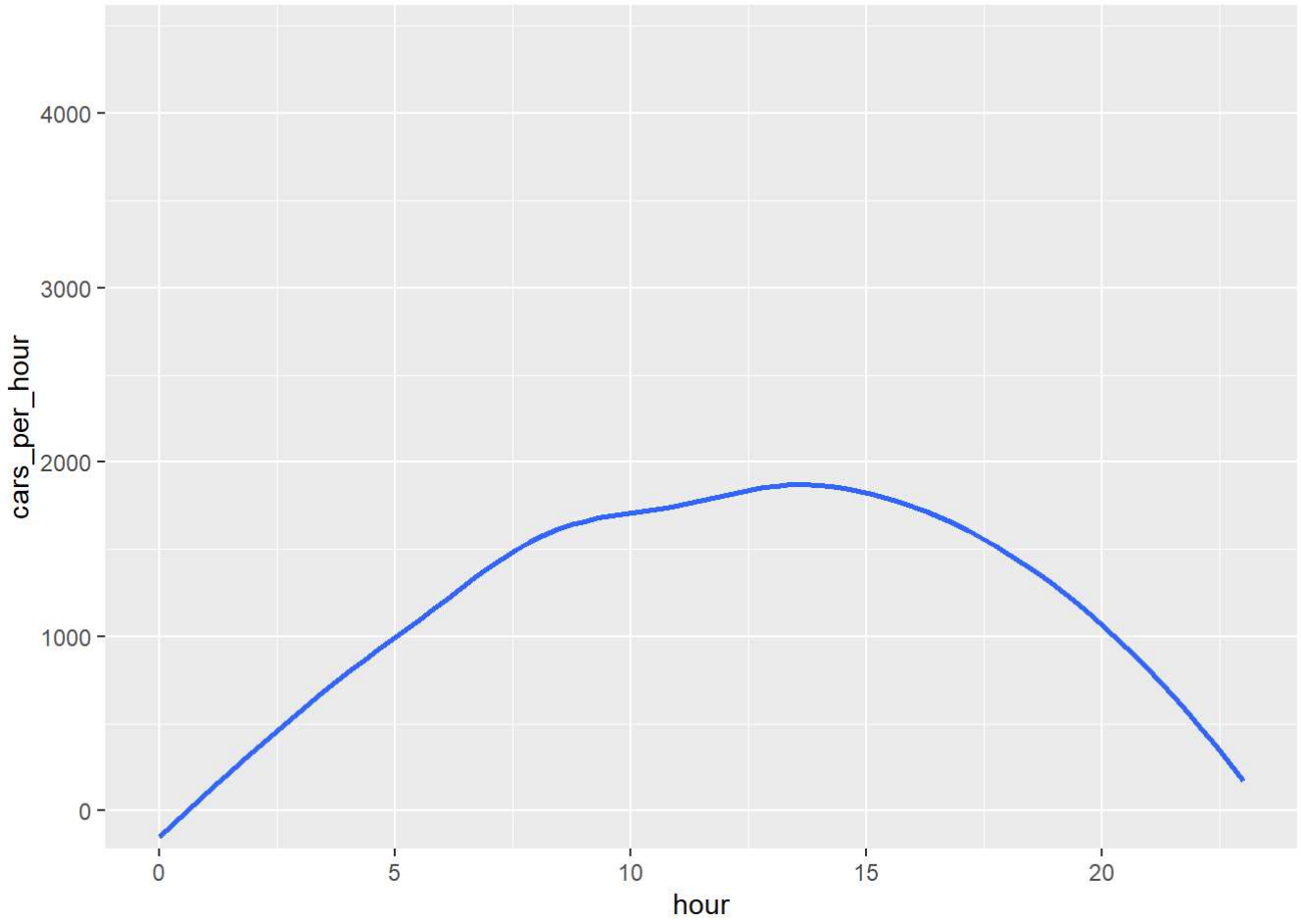


Before and After State of Emergency Issued

```
# Before
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "po",
         full_date > ymd(20200301) & full_date < ymd(20200318)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4400))
```

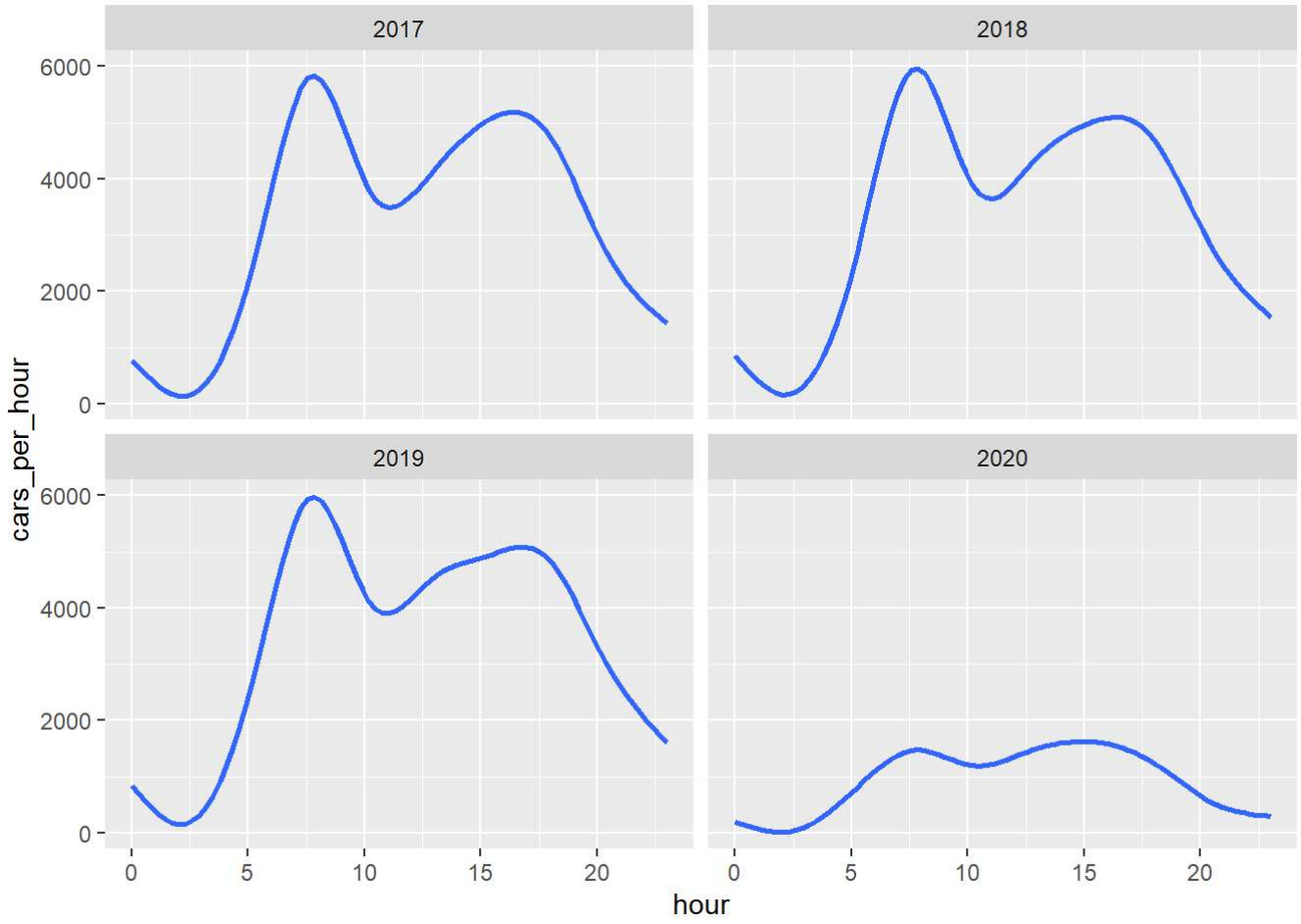



```
# After
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "po",
         full_date >= ymd(20200318) & full_date <= ymd(20200331)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4400))
```



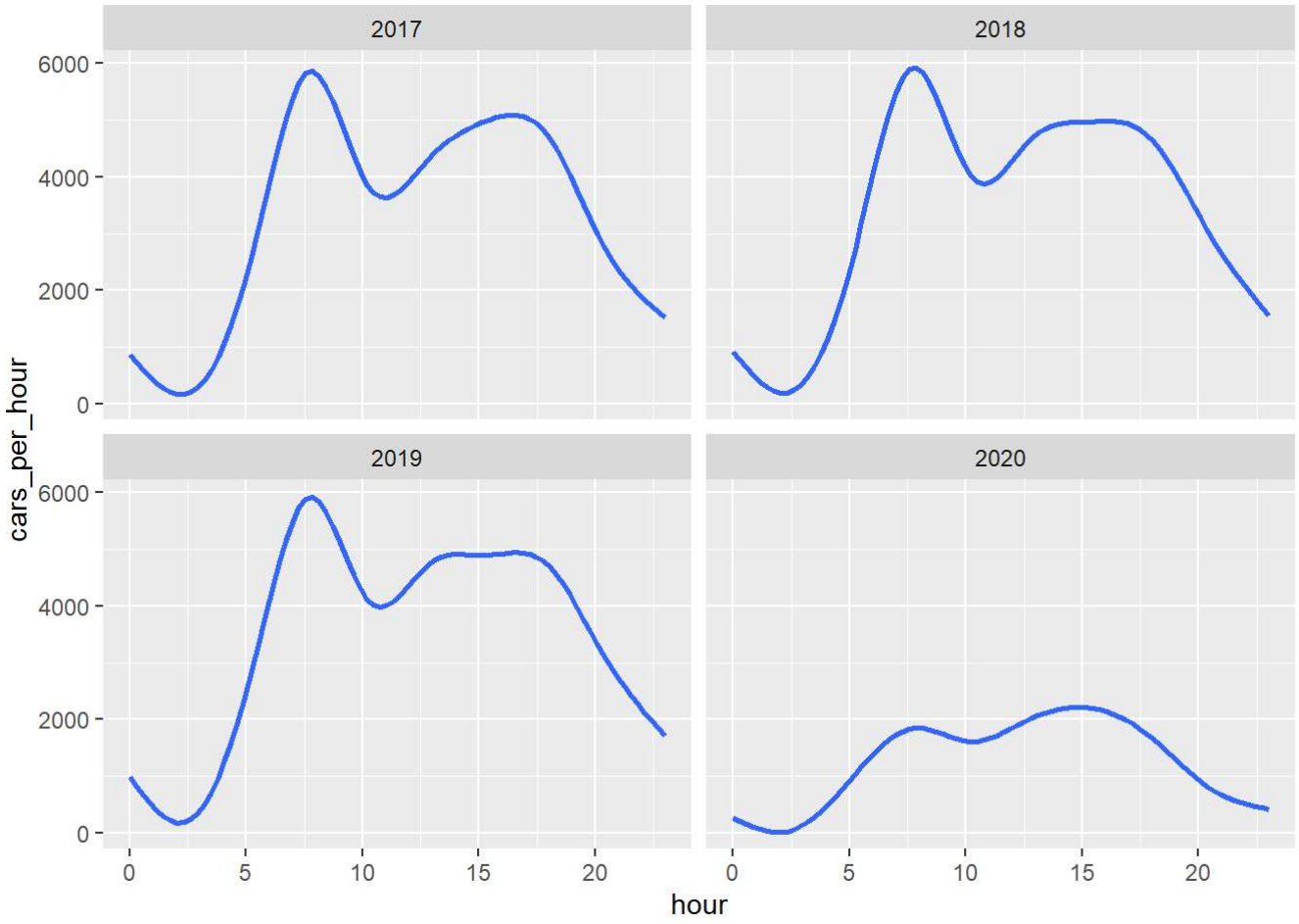
Post Allston Interchange in April

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "po",
         month(full_date) == 4
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



Post Allston Interchange in May

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
       location == "po",
       month(full_date) == 5
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```


2017
2018
2019
2020
cars_per_hour
hour
0
2000
4000
6000
0
5
10
15
20