Solution 1: Basic programming in R

Benjamin Smith

07/08/2025

Part 1: Matrix and vector operations.

1. Solve the following system:

$$\begin{bmatrix} a_1 & b_1 & & & 0 \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & a_{99} & b_{99} \\ 0 & & & c_{99} & a_{100} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{100} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{100} \end{bmatrix}$$

where

$$a_j = j, \quad b_j = 1/j, \quad c_j = 1, \quad d_j = \sin(j\pi/200)$$

and print $x_1, x_2, \ldots, x_5$.

Solution:

```
# Define A.
A <- matrix(rep(0, 100*100), nrow = 100, ncol = 100, byrow = TRUE)

for (i in c(1:100)) {
    A[i, i] <- i

    if (i + 1 < 101)  A[i, i + 1] <- 1/i
    if (i - 1 > 0)    A[i, i - 1] <- 1
}

# Define D.
d <- c(1:100)
d <- sin(d*pi/200)

# Solve Ax = d.
x <- solve(A, d)
x[1:5]
```

```
## [1] 0.005473329 0.010233988 0.010938907 0.012167224 0.012730871
```

Part 2: For loops.

1. Write a function that uses a `for` loop to calculate the following with a sequence of m, and generate a plot for m verses E_m. Avoid using a `for` loop, can you complete the same task?

$$E_m = 1 + \frac{1}{2} + \cdots + \frac{1}{2^m} - \log(2^m)$$

Solution:

```
# Using for loop.
E_m <- function(m) {
  res <- 1
  for (i in c(1:m)) {
    res <- res + 1/(2^i)
  }
  res <- res - log(2^m)

  return(res)
}

# Avoid using for loop.
E_m2 <- function(m) {
  res <- 1

  index <- c(1:m)
  denom <- 2^index
  res <- res + sum(1/denom)

  res <- res - log(2^m)

  return(res)
}

E_m(100)
```

```
## [1] -67.31472
```

```
E_m2(100)
```

```
## [1] -67.31472
```

Part 3: Analyze NYC flight delays.

Install the "nycflights13" package. The data comes from the US Bureau of Transportation Statistics. Using the data, complete the following tasks:

1. Find all flights that had an arrival delay of >4 hours, return the first 5 row. (Note: `arr_delay` is in mins)

2. Find all flight names that flew from JFK to IAH, i.e. return only unique values of "flight" variable after filtering. Hint: `unique()` would help.
3. Find how many flights were operated by UA.
4. Find how many unique flights were operated by UA.
5. Sort flights that have the most delayed flights. Show the first 5 row.
6. Generate a scatter plot with x-axis `dist` and y-axis `delay`, where each dot is a unique flights and destination, `dist` is the average distance of each destination `dest`, and `delay` is the average delay time `arr_delay`, with the size of `dot` equals to the count of delay records.

```
library(tidyverse)
library(nycflights13)
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Solution

1. Find all flights that had an arrival delay of >4 hours, i.e. return the first 5 row. (Note: `arr_delay` is in mins)

```
flights %>% filter(arr_delay > 240) %>% head(5)
```

```
## # A tibble: 5 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      848           1835       853     1001           1950
## 2  2013     1     1     1815           1325       290     2120           1542
## 3  2013     1     1     1842           1422       260     1958           1535
## 4  2013     1     1     2115           1700       255     2330           1920
## 5  2013     1     1     2205           1720       285       46           2040
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

2. Find all flight names that flew from JFK to IAH, i.e. return only unique values of "flight" variable after filtering. Hint: `unique()` would help.

```
df <- flights %>% filter(origin == "JFK" & dest == "IAH")
unique(df$flight)
```

```
## [1]  211 1901  523
```

3. Find how many flights were operated by UA.

```
nrow(filter(flights, carrier %in% c("UA")))
```

```
## [1] 58665
```

4. Find how many unique flights were operated by UA.

```
df <- filter(flights, carrier %in% c("UA"))
length(unique(df$flight))
```

```
## [1] 1285
```

5. Sort flights that have the most delayed flights. Show the first 5 row.

```
flights %>% arrange(desc(dep_delay)) %>% head(5)
```

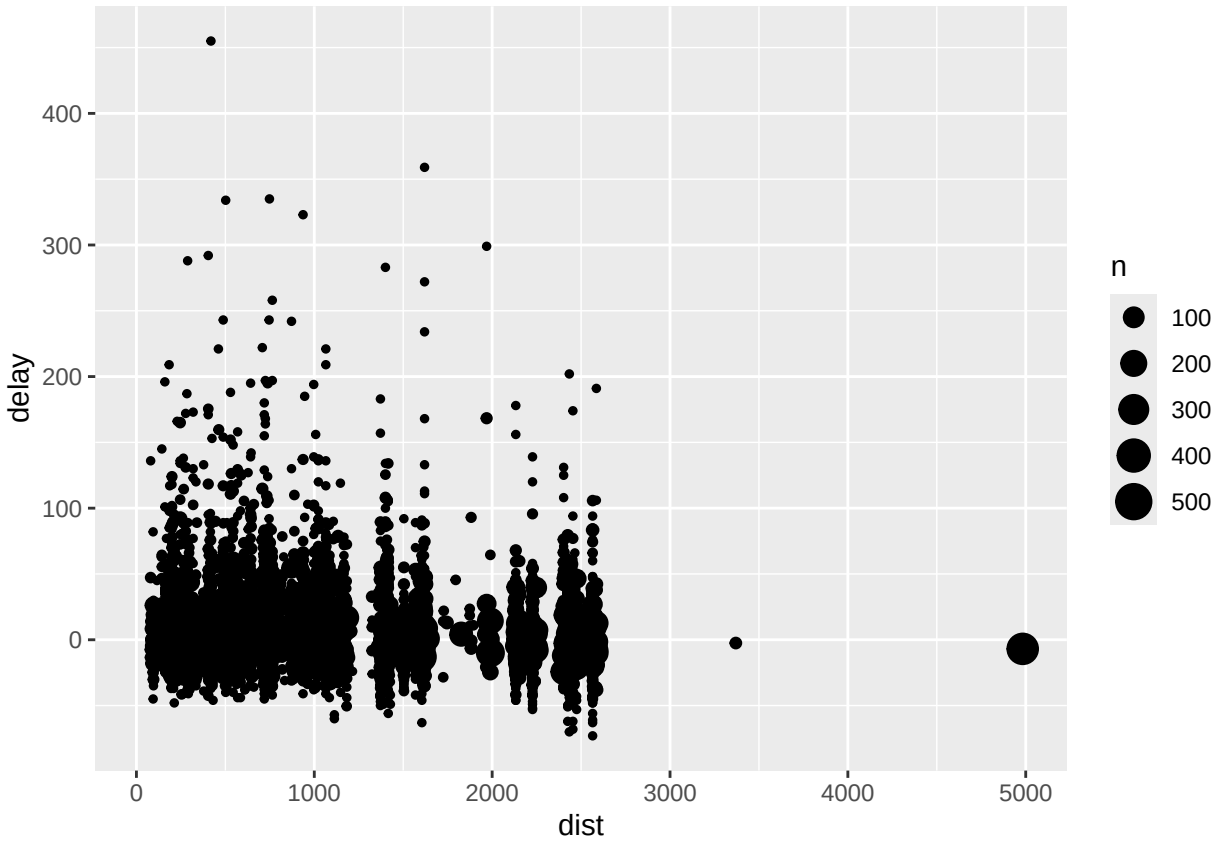
```
## # A tibble: 5 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     9      641            900      1301     1242           1530
## 2  2013     6    15     1432           1935      1137     1607           2120
## 3  2013     1    10     1121           1635      1126     1239           1810
## 4  2013     9    20     1139           1845      1014     1457           2210
## 5  2013     7    22      845           1600      1005     1044           1815
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

6. Generate a scatter plot with x-axis `dist` and y-axis `delay`, where each dot is a unique flights and destination, `dist` is the average distance of each destination `dest`, and `delay` is the average delay time `arr_delay`, with the size of `dot` equals to the count of delay records.

```
flights %>%
  group_by(flight, dest) %>%
  summarise(delay = mean(arr_delay), dist = mean(distance), n = n()) %>%
  ggplot() +
  geom_point(aes(x = dist, y = delay, size = n))
```

```
## `summarise()` has grouped output by 'flight'. You can override using the
## `.groups` argument.
```

```
## Warning: Removed 2824 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



5. NYC bus delays

The data is from the kaggle dataset "Bus Breakdown and Delays NYC - When and why a bus was delayed? Bus delays 2015 to 2017". (https://www.kaggle.com/anthobau/busbreakdownanddelays).

The Bus Breakdown and Delay system collects information from school bus vendors operating out in the field in real time. Bus staff that encounter delays during the route are instructed to radio the dispatcher at the bus vendor's central office. The bus vendor staff are then instructed to log into the Bus Breakdown and Delay system to record the event and notify OPT. OPT customer service agents use this system to inform parents who call with questions regarding bus service. The Bus Breakdown and Delay system is publicly accessible and contains real time updates. All information in the system is entered by school bus vendor staff.

You can find data for years 2015 to 2017.

```
bus.delay <- read_csv("Bus_Breakdown_and_Delays.csv")
```

```
## Rows: 147972 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (19): School_Year, Run_Type, Bus_No, Route_Number, Reason, Schools_Servi...
## dbl  (2): Busbreakdown_ID, Number_Of_Students_On_The_Bus
##
## 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.
```

Alternatively, you can install the `RKaggle` package to load the dataset directly into R.

```
# Run this line to install RKaggle
# install.packages("RKaggle")
library(RKaggle)
bus.delay<-get_dataset("anthobau/busbreakdownanddelays")
```

```
head(bus.delay)
```

```
## # A tibble: 6 x 21
##   School_Year Busbreakdown_ID Run_Type          Bus_No Route_Number Reason
##   <chr>                 <dbl> <chr>             <chr>  <chr>        <chr>
## 1 2015-2016           1224901 Pre-K/EI          811    1            Other
## 2 2015-2016           1225098 Pre-K/EI          9302   1            Heavy Traff~
## 3 2015-2016           1215800 Pre-K/EI          358    2            Heavy Traff~
## 4 2015-2016           1215511 Pre-K/EI          331    2            Other
## 5 2015-2016           1215828 Pre-K/EI          332    2            Other
## 6 2015-2016           1225671 Special Ed AM Run 12568  P640         Heavy Traff~
## # i 15 more variables: Schools_Serviced <chr>, Occurred_On <chr>,
## #   Created_On <chr>, Boro <chr>, Bus_Company_Name <chr>,
## #   How_Long_Delayed <chr>, Number_Of_Students_On_The_Bus <dbl>,
## #   Has_Contractor_Notified_Schools <chr>,
## #   Has_Contractor_Notified_Parents <chr>, Have_You_Alerted_OPT <chr>,
## #   Informed_On <chr>, Incident_Number <chr>, Last_Updated_On <chr>,
## #   Breakdown_or_Running_Late <chr>, School_Age_or_PreK <chr>
```

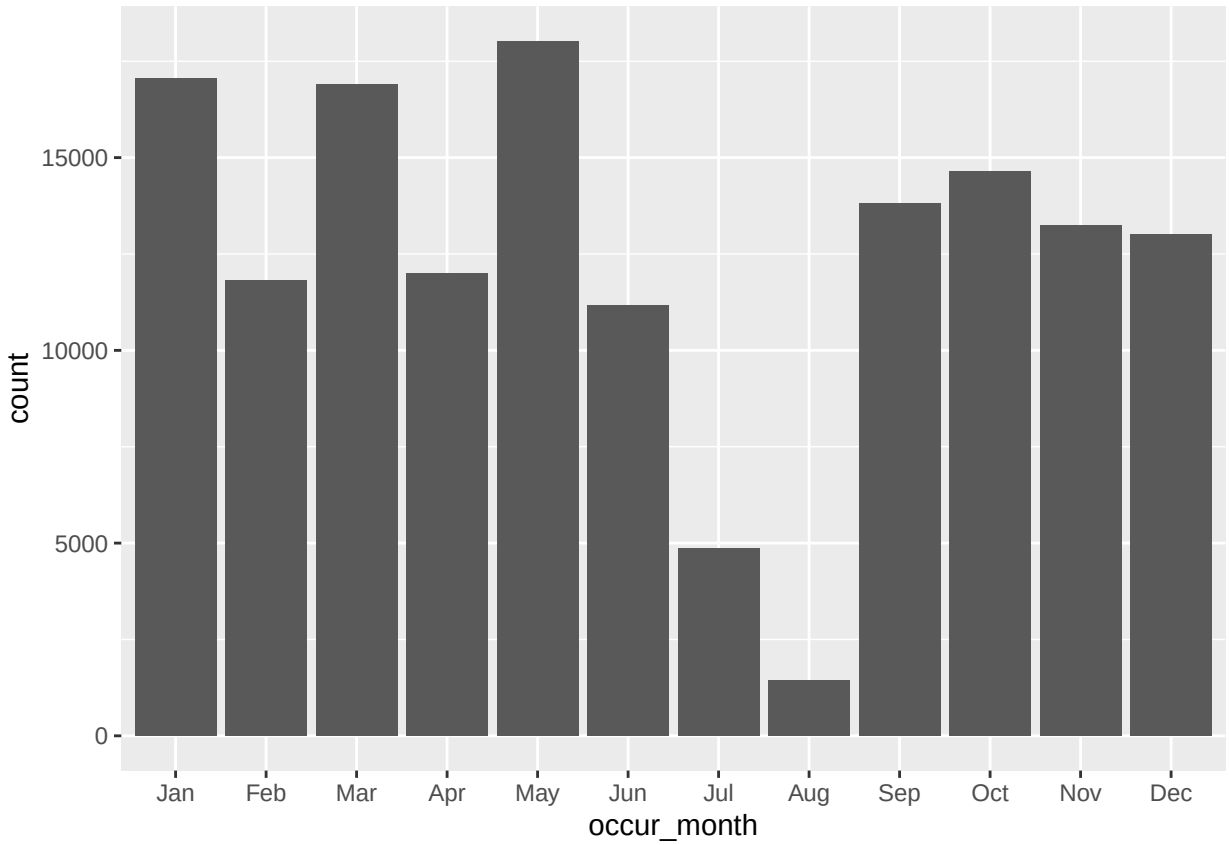
The following code creates new features

```
library(lubridate)

bus.delay$occur_date <- as.POSIXct(bus.delay$Occurred_On, format= "%m/%d/%Y %H:%M:%S %p")

# Day of the week
bus.delay$occur_weekday <- wday(bus.delay$occur_date, label = T)
bus.delay$occur_month <- month(bus.delay$occur_date, label = T)
bus.delay$occur_year <- year(bus.delay$occur_date)
#head(bus.delay)
```

1. Group by `occur_month`, and generate a bar plot that showing the number of delays by month.

```
library(tidyverse)
```

```
#library(ggplot2)
# tidyverse
bus.delay %>%
  group_by(occur_month) %>%
  summarise(count = n()) %>%
  ggplot(aes(x = occur_month, y = count)) + #axis
  geom_bar(stat = 'identity') # stacked
```



The following code generates a new variable called "duration" in mins.

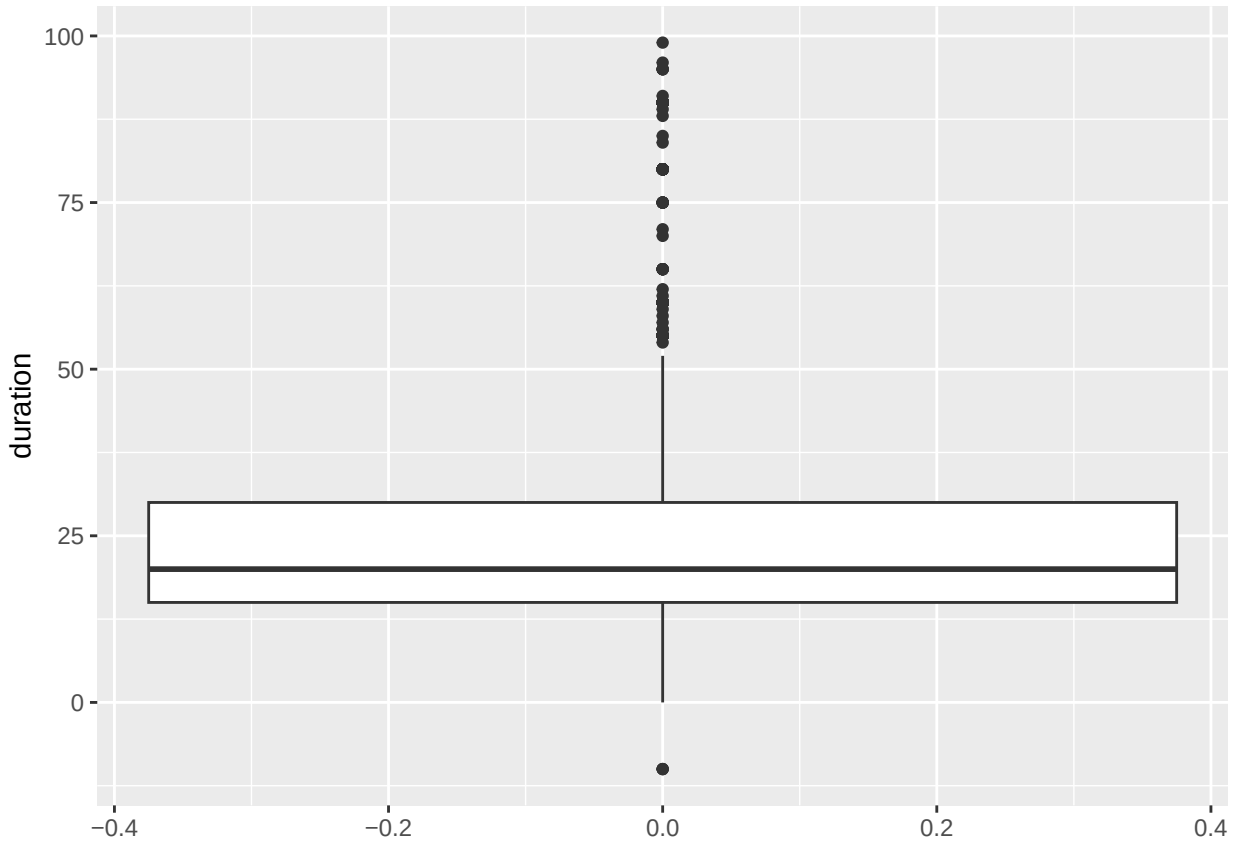
```
bus.delay$duration <- as.numeric(gsub("([0-9]{1,2}).*$", "\\1", bus.delay$How_Long_Delayed))
```

```
## Warning: NAs introduced by coercion
```

2. Plot the boxplot of duration.

```
bus.delay %>%
  ggplot(aes(y = duration)) +
  geom_boxplot()
```

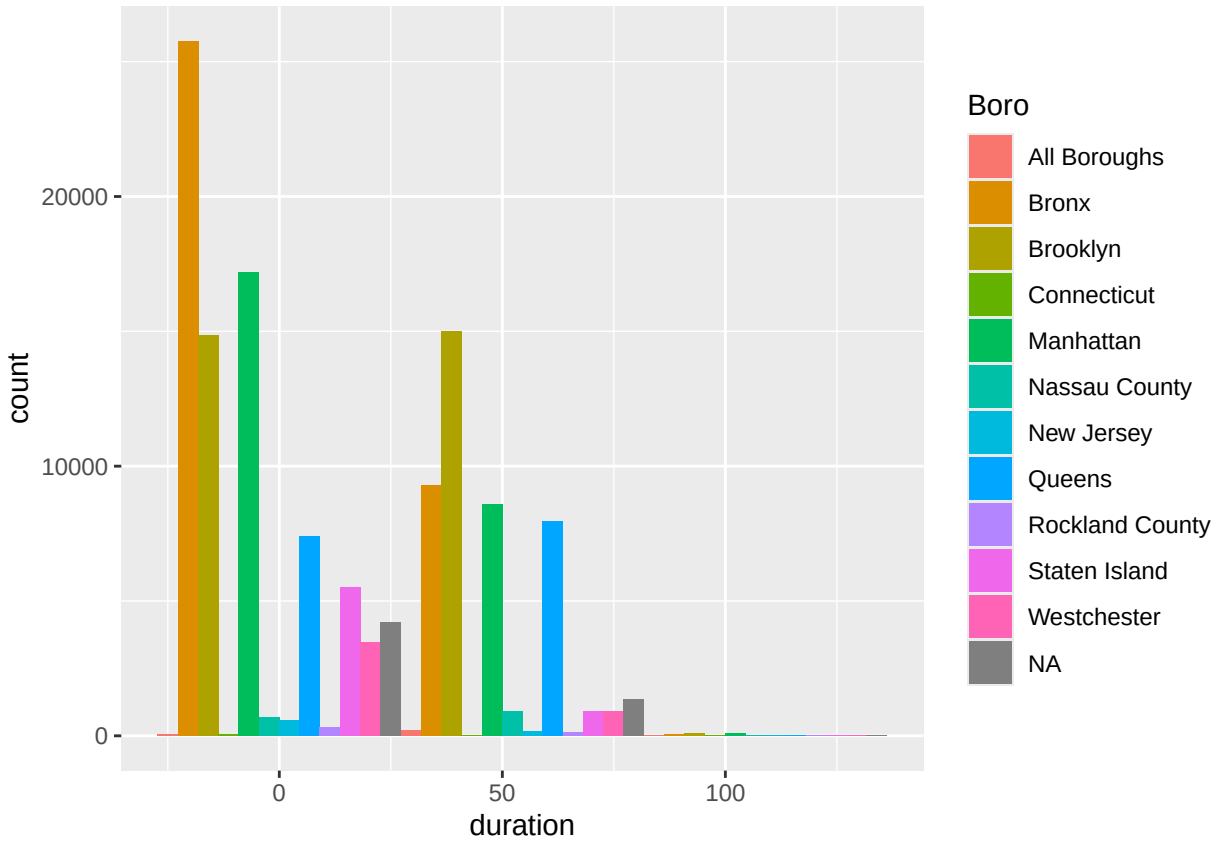
```
## Warning: Removed 22163 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



3. Plot a histogram of durations and group by "Boro".

```
bus.delay %>%
  ggplot() +
  geom_histogram(aes(x = duration,
                     fill = Boro),
                 position = 'dodge',
                 bins = 3)
```

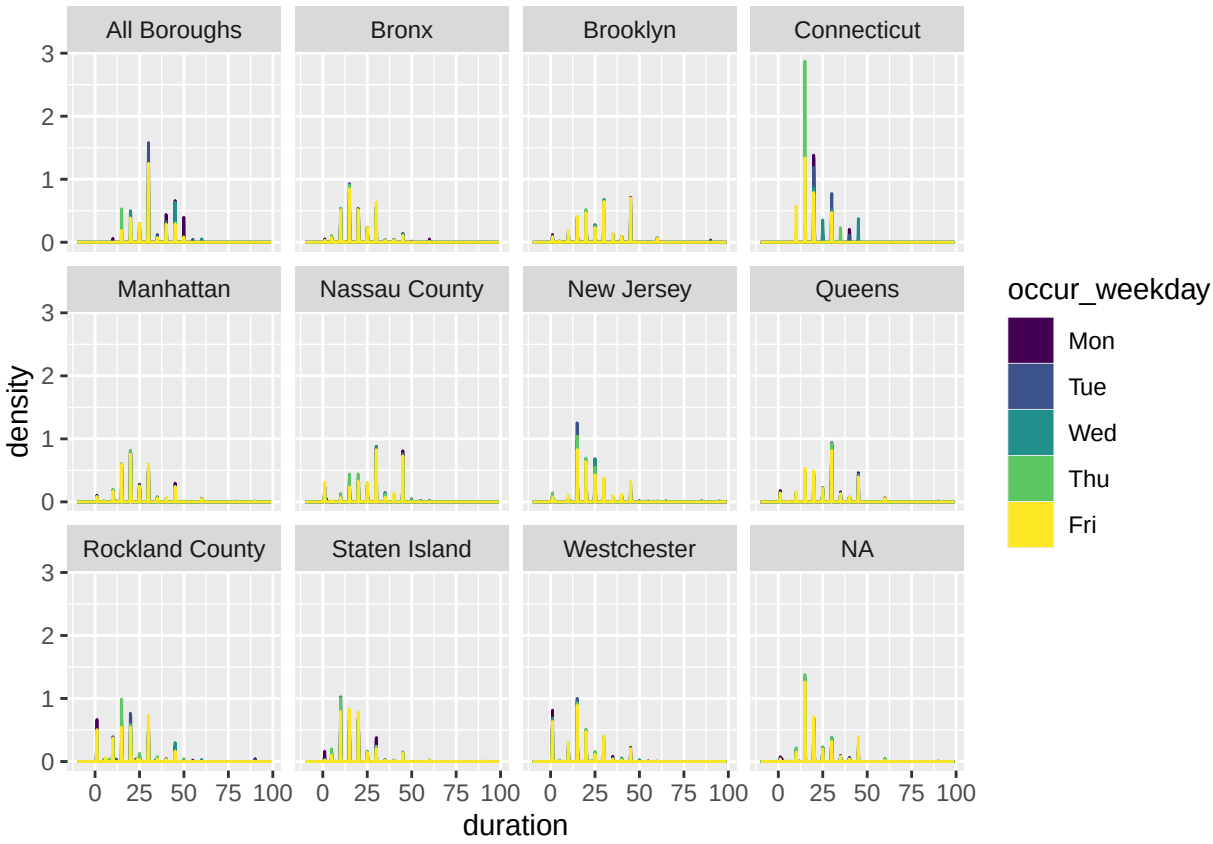
```
## Warning: Removed 22163 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



4. Plot the density of duration, group by "occur_weekday" and use facet to stratify on" "Boro".

```
bus.delay %>%
  ggplot() +
  geom_density(aes(x = duration,
                   color = occur_weekday,
                   fill = occur_weekday),
               bw = 0.08) +
  facet_wrap(~Boro, ncol = 4)
```

```
## Warning: Removed 22163 rows containing non-finite outside the scale range
## (`stat_density()`).
```



5. Print top five Boro by mean delay.

```
bus.delay %>%
  group_by(Boro) %>%
  summarise(mean_delay = mean(duration,
                              na.rm = TRUE)) %>%
  arrange(desc(mean_delay)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   Boro          mean_delay
##   <chr>              <dbl>
## 1 All Boroughs        31.7
## 2 Brooklyn            27.8
## 3 Nassau County       27.8
## 4 Queens              26.5
## 5 Manhattan           24.1
```