Uber analysis

AUTHOR
Eva Wanjiru

Uber Data Analysis through visualizations in R

Data storytelling is a crucial part of machine learning that allows businesses to comprehend the history of diverse processes. Companies can benefit from understanding complex data and gaining insights that will help them make decisions by using visualization. This project is more of a data visualization tutorial that shows you how to use the ggplot2 library to comprehend the data and cultivate an intuitive understanding of the travelers.

```
##load packages
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.2.3

Warning: package 'tibble' was built under R version 4.2.3

Warning: package 'tidyr' was built under R version 4.2.3

Warning: package 'readr' was built under R version 4.2.3

Warning: package 'purrr' was built under R version 4.2.3

Warning: package 'dplyr' was built under R version 4.2.3

Warning: package 'stringr' was built under R version 4.2.3

Warning: package 'forcats' was built under R version 4.2.3

Warning: package 'lubridate' was built under R version 4.2.3

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.2     ✓ readr     2.1.4
✓ forcats   1.0.0     ✓ stringr   1.5.0
✓ lubridate 1.9.2     ✓ tibble    3.2.1
✓ purrr     1.0.1     ✓ tidyr     1.3.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✕ dplyr::filter() masks stats::filter()
✕ dplyr::lag()    masks stats::lag()
```

```
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
library(lubridate)
library(ggthemes)
library(DT)
```

```
Warning: package 'DT' was built under R version 4.2.3
```

```
library(scales)
```

```

Attaching package: 'scales'

The following object is masked from 'package:purrr':

    discard

The following object is masked from 'package:readr':

    col_factor
```

```
##load the data
apr_data <- read.csv("uber-raw-data-apr14.csv")
may_data <- read.csv("uber-raw-data-may14.csv")
jun_data <- read.csv("uber-raw-data-jun14.csv")
jul_data <- read.csv("uber-raw-data-jul14.csv")
aug_data <- read.csv("uber-raw-data-aug14.csv")
sep_data <- read.csv("uber-raw-data-sep14.csv")
```

```
# Combine the data together
data <- rbind(apr_data, may_data, jun_data, jul_data, aug_data, sep_data)
cat("The dimensions of the data are:", dim(data))
```

```
The dimensions of the data are: 4534327 4
```

The dataset has 4534327 observations and 4 rows.

```
##first 6 rows
head(data)
```

```
         Date.Time     Lat      Lon   Base
1 4/1/2014 0:11:00 40.7690 -73.9549 B02512
2 4/1/2014 0:17:00 40.7267 -74.0345 B02512
3 4/1/2014 0:21:00 40.7316 -73.9873 B02512
4 4/1/2014 0:28:00 40.7588 -73.9776 B02512
5 4/1/2014 0:33:00 40.7594 -73.9722 B02512
6 4/1/2014 0:33:00 40.7383 -74.0403 B02512
```

```r
##structure of the data
str(data)
```

```
'data.frame':	4534327 obs. of  4 variables:
 $ Date.Time: chr  "4/1/2014 0:11:00" "4/1/2014 0:17:00" "4/1/2014 0:21:00" "4/1/2014 0:28:00"
...
 $ Lat      : num  40.8 40.7 40.7 40.8 40.8 ...
 $ Lon      : num  -74 -74 -74 -74 -74 ...
 $ Base     : chr  "B02512" "B02512" "B02512" "B02512" ...
```

Data cleaning

The datetime is formatted into a more readable format.

```r
##recode the variables into the right format (date column)
data$Date.Time <- as.POSIXct(data$Date.Time, format="%m/%d/%Y %H:%M:%S")
data$Time <- format(as.POSIXct(data$Date.Time, format = "%m/%d/%Y %H:%M:%S"), format="%H:%M:%S")
data$Date.Time <- ymd_hms(data$Date.Time)
```

```r
# Create individual columns for month day and year
data$day <- factor(day(data$Date.Time))
data$month <- factor(month(data$Date.Time, label=TRUE))
data$year <- factor(year(data$Date.Time))
data$dayofweek <- factor(wday(data$Date.Time, label=TRUE))
```

```r
# Add Time variables as well
data$second = factor(second(hms(data$Time)))
data$minute = factor(minute(hms(data$Time)))
data$hour = factor(hour(hms(data$Time)))

##copy the data
df<-data
```

Data visualization

```r
hourly_data <- df %>%
                    group_by(hour) %>%
                            dplyr::summarize(Total = n())

# Shows data in a searchable js table
datatable(hourly_data)
```

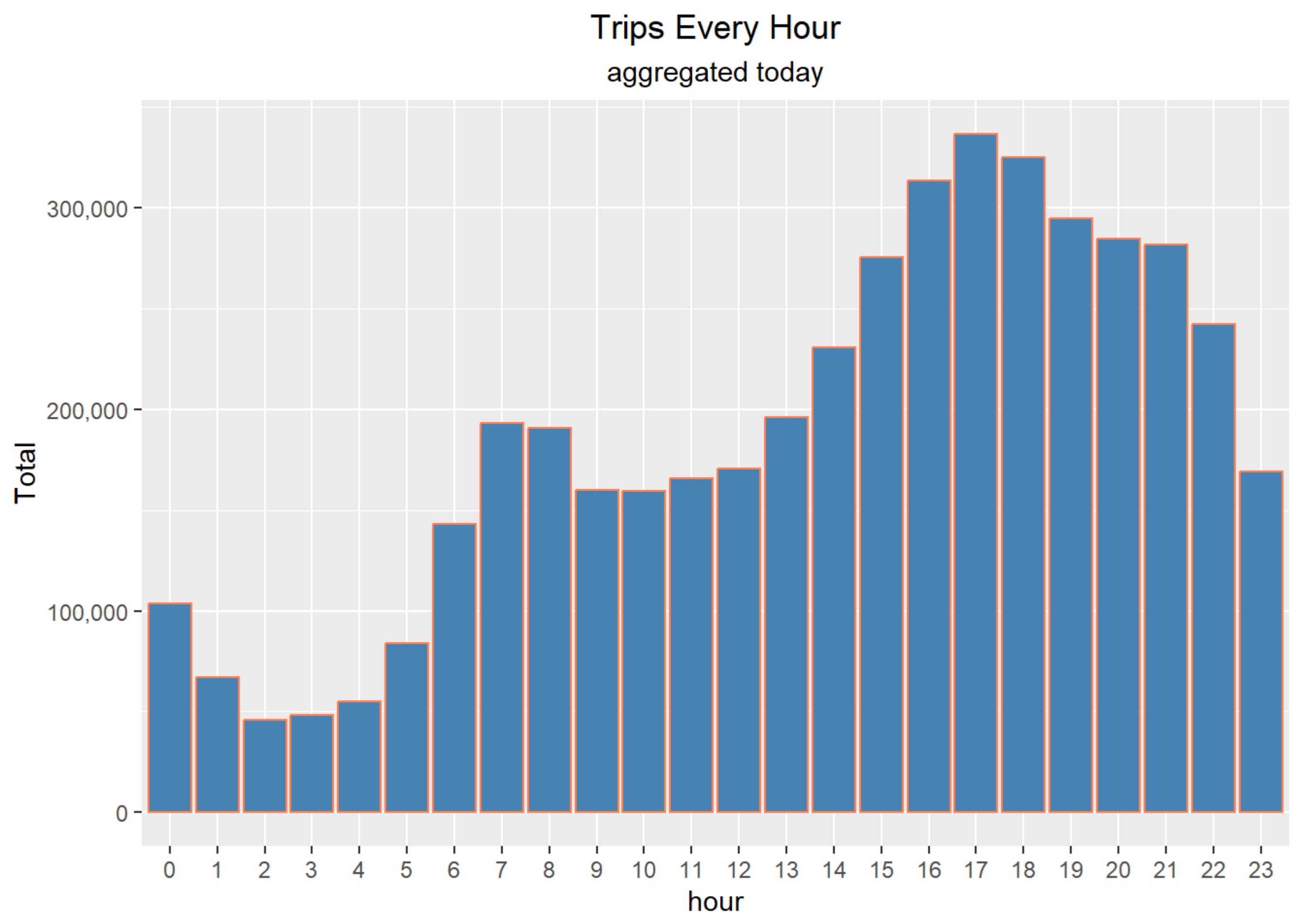
Show 10 entries   Search:

	hour	Total
1	0	103836
2	1	67227
3	2	45865
4	3	48287
5	4	55230
6	5	83939
7	6	143213
8	7	193094
9	8	190504
10	9	159967

Showing 1 to 10 of 24 entries

Previous 1 2 3 Next

```r
##plot the data by hour
hourly_data %>%
  ggplot(aes(x=hour,y=Total))+
geom_bar(stat="identity",
         fill="steelblue",
         color="coral") +
ggtitle("Trips Every Hour", subtitle = "aggregated today") +
theme(legend.position = "none",
      plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5)) +
scale_y_continuous(labels=comma)
```

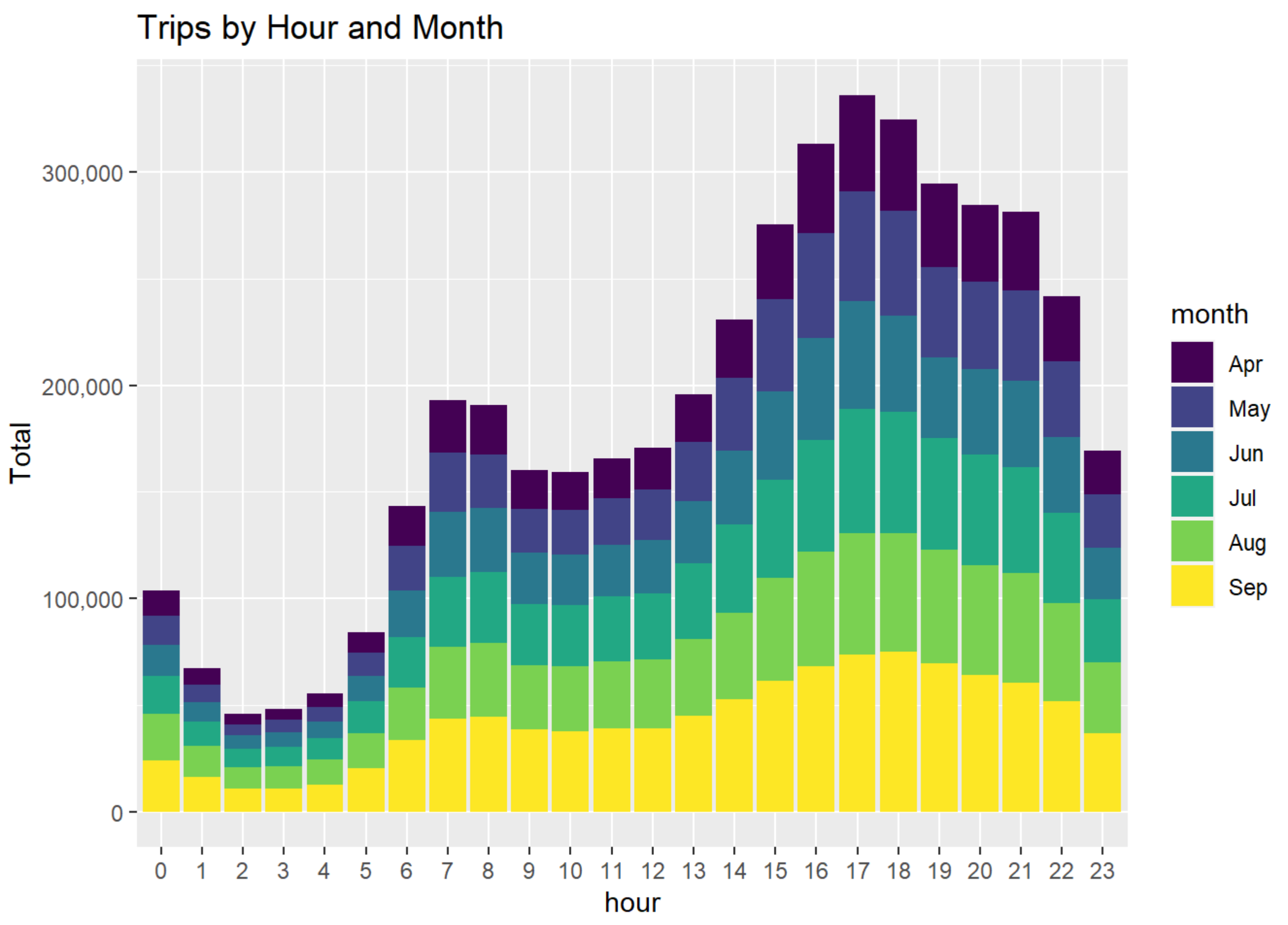


From the graph above, it an be seen that most trips are made between 1700hrs and 1800hrs.

```
# Aggregate the data by month and hour
month_hour_data <- df %>% group_by(month, hour) %>%  dplyr::summarize(Total = n())
```

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

```
ggplot(month_hour_data, aes(hour, Total, fill=month)) +
geom_bar(stat = "identity") +
ggtitle("Trips by Hour and Month") +
scale_y_continuous(labels = comma)
```

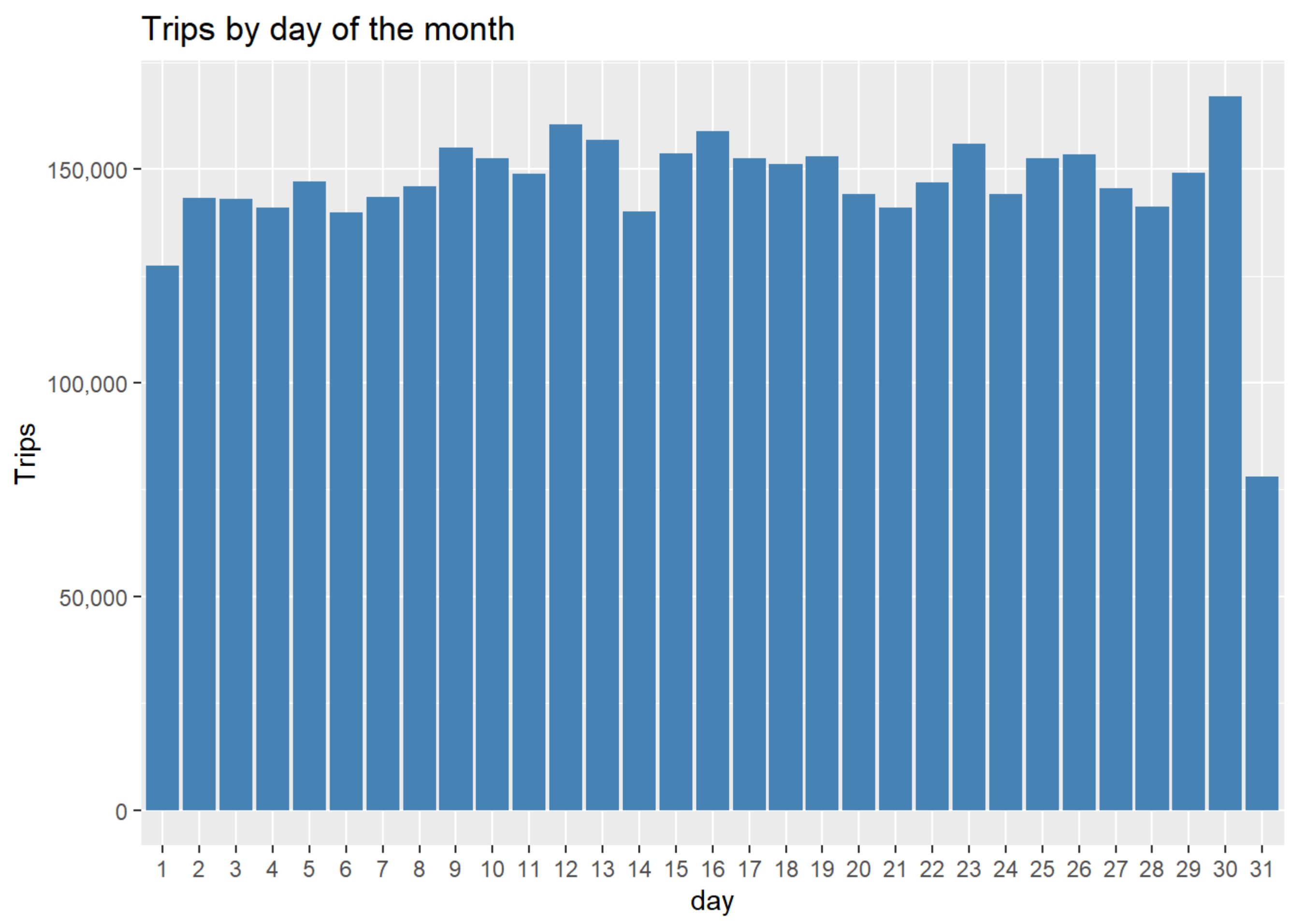


Plotting trips during every day of the month

```
# Aggregate data by day of the month
day_data <- df %>% group_by(day) %>% dplyr::summarize(Trips = n())
day_data
```

```
# A tibble: 31 × 2
   day    Trips
   <fct>  <int>
 1 1     127430
 2 2     143201
 3 3     142983
 4 4     140923
 5 5     147054
 6 6     139886
 7 7     143503
 8 8     145984
 9 9     155135
10 10    152500
# ℹ 21 more rows
```

```
# Plot the data for the day
ggplot(day_data, aes(day, Trips)) +
geom_bar(stat = "identity", fill = "steelblue") +
ggtitle("Trips by day of the month") +
theme(legend.position = "none") +
scale_y_continuous(labels = comma)
```



Most trips are recorded on the 30th day of the month.

```
##select the color vector
colors = c("#CC1011", "#665555", "#05a399", "#cfcaca", "#f5e840", "#0683c9", "#e075b0")
```

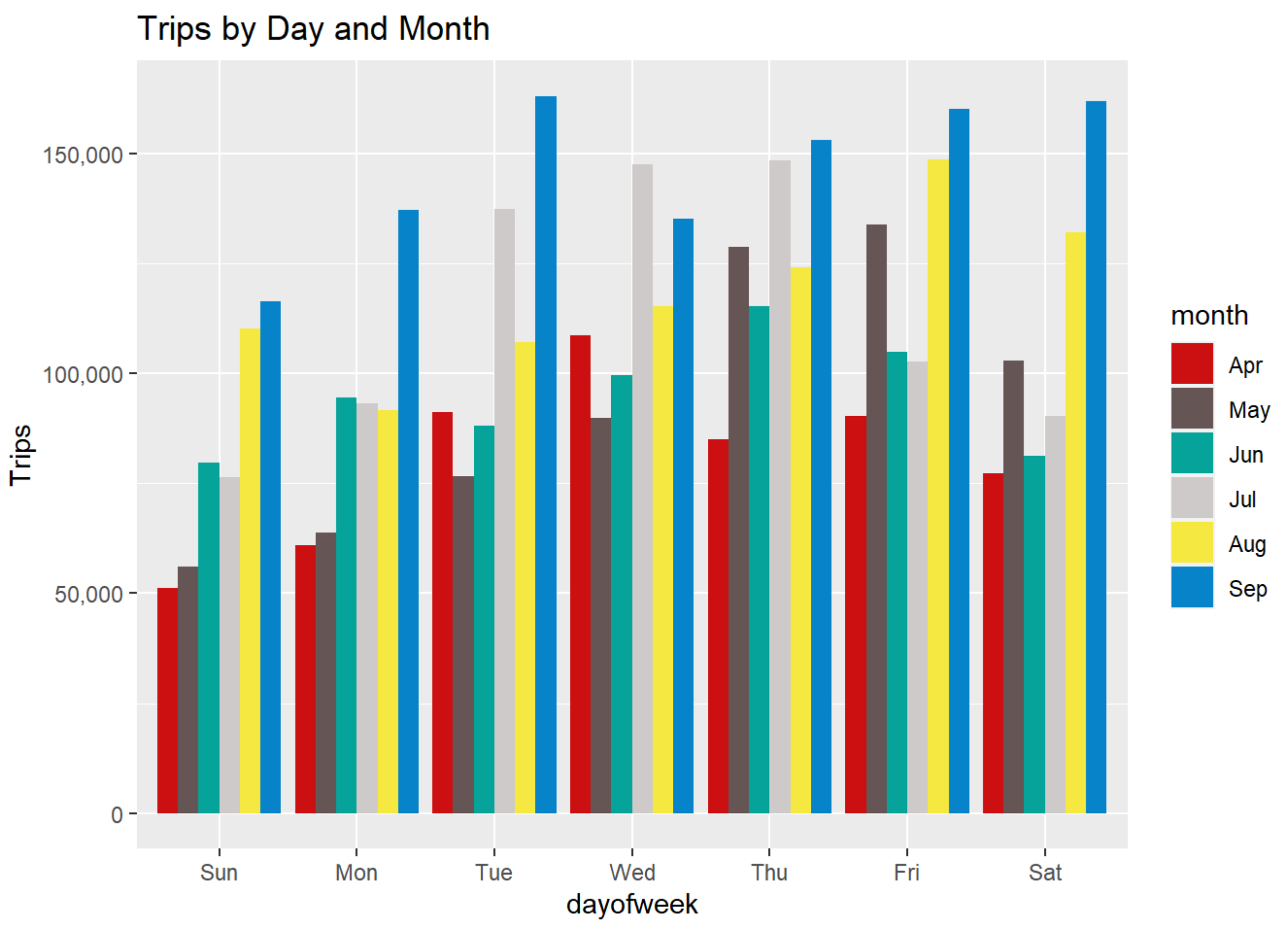
```
# Collect data by day of the week and month
day_month_data <- df %>% group_by(dayofweek, month) %>% dplyr::summarize(Trips = n())
```

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

```
day_month_data
```

```
# A tibble: 42 × 3
# Groups:   dayofweek [7]
   dayofweek month  Trips
   <ord>     <ord>  <int>
 1 Sun       Apr    51251
 2 Sun       May     56168
 3 Sun       Jun     79656
 4 Sun       Jul     76327
 5 Sun       Aug    110246
 6 Sun       Sep    116532
 7 Mon       Apr     60861
 8 Mon       May     63846
 9 Mon       Jun     94655
10 Mon       Jul     93189
# ℹ 32 more rows
```

```
# Plot
ggplot(day_month_data, aes(dayofweek, Trips, fill = month)) +
geom_bar(stat = "identity", aes(fill = month), position = "dodge") +
ggtitle("Trips by Day and Month") +
scale_y_continuous(labels = comma) +
scale_fill_manual(values = colors)
```



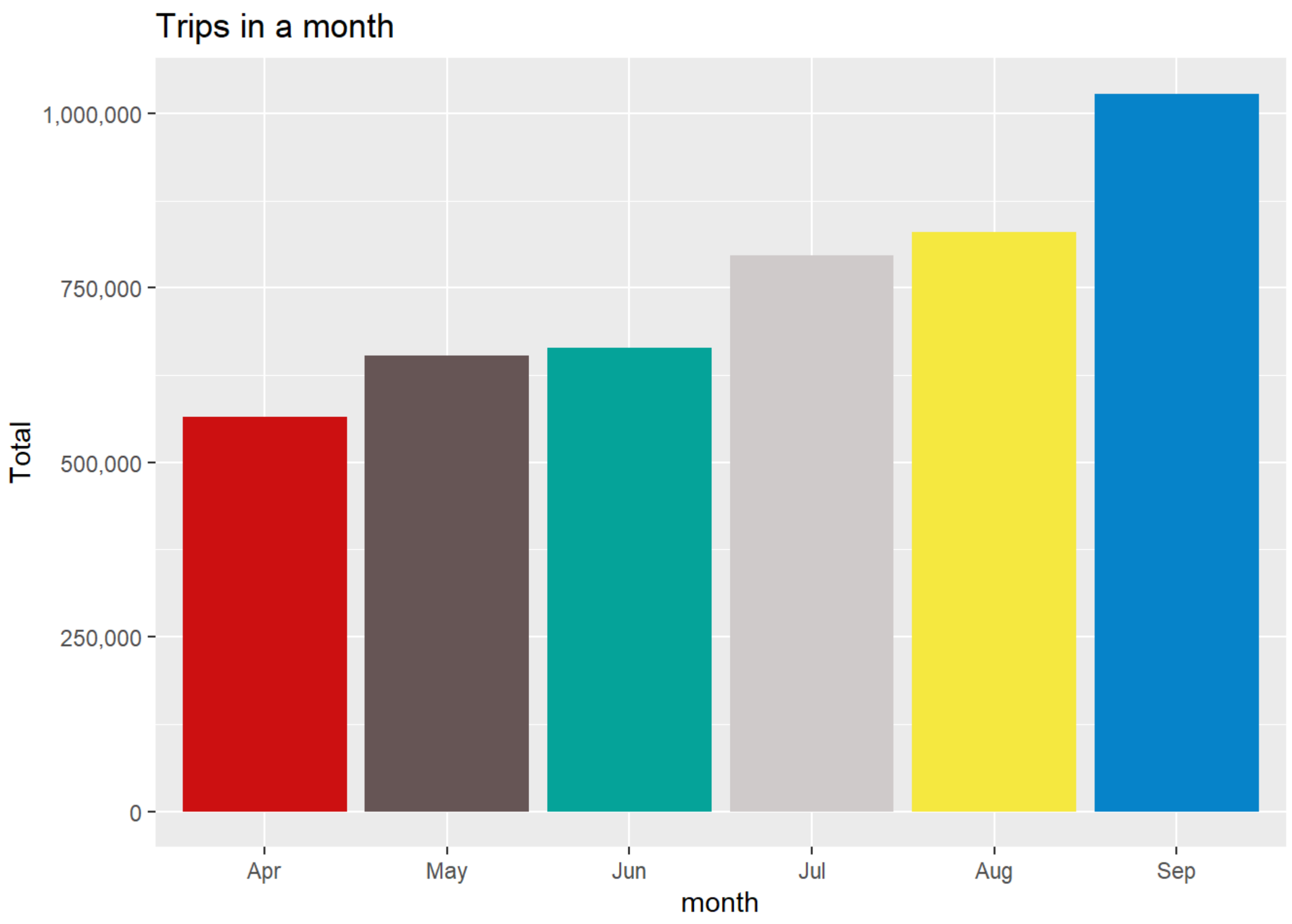
Number of trips during months in a year

```
month_data <- df %>% group_by(month) %>% dplyr::summarize(Total = n())
```

```
month_data
```

```
# A tibble: 6 × 2
  month   Total
  <ord>   <int>
1 Apr    564516
2 May    652435
3 Jun    663844
4 Jul    796121
5 Aug    829275
6 Sep   1028136
```

```
ggplot(month_data, aes(month, Total, fill = month)) +
geom_bar(stat = "Identity") +
ggtitle("Trips in a month") +
theme(legend.position = "none") +
scale_y_continuous(labels = comma) +
scale_fill_manual(values = colors)
```



Most trips are made in September.

Heat map visualizations

Heatmap by hour and day

```
day_hour_data <- df %>% group_by(day, hour) %>% dplyr::summarize(Total = n())
```

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

```
datatable(day_hour_data)
```

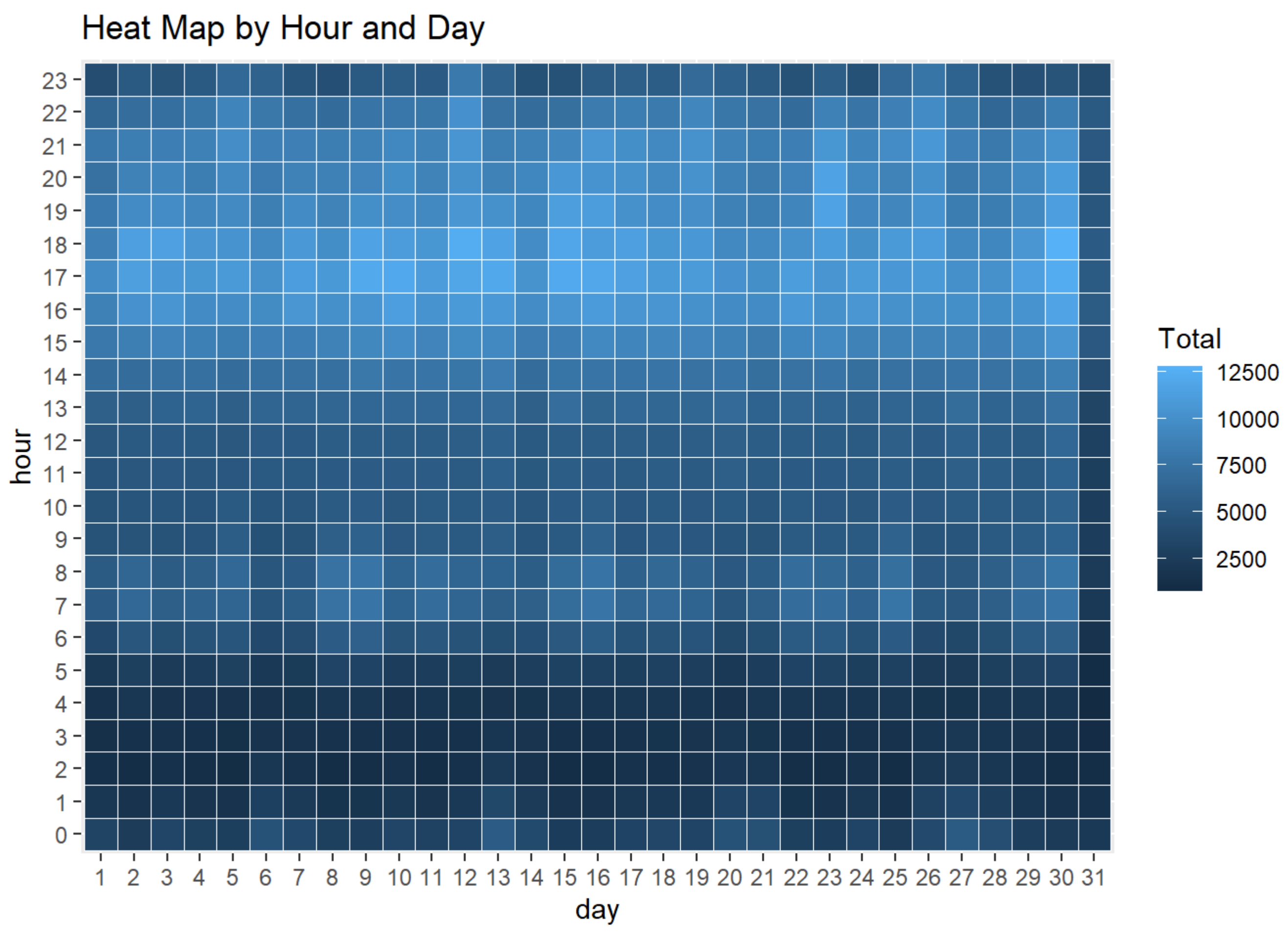
Show 10 entries Search:

	day	hour	Total
1	1	0	3247
2	1	1	1982
3	1	2	1284
4	1	3	1331
5	1	4	1458
6	1	5	2171
7	1	6	3717
8	1	7	5470
9	1	8	5376
10	1	9	4688

Showing 1 to 10 of 744 entries Previous 1 2 3 4 5 … 75 Next

```
# Plot a heatmap

ggplot(day_hour_data, aes(day, hour, fill = Total)) +
geom_tile(color = "white") +
ggtitle("Heat Map by Hour and Day")
```



Heatmap by day and month

```
# Collect data by month and day
month_day_data <- df %>% group_by(month, day) %>% dplyr::summarize(Trips = n())
```

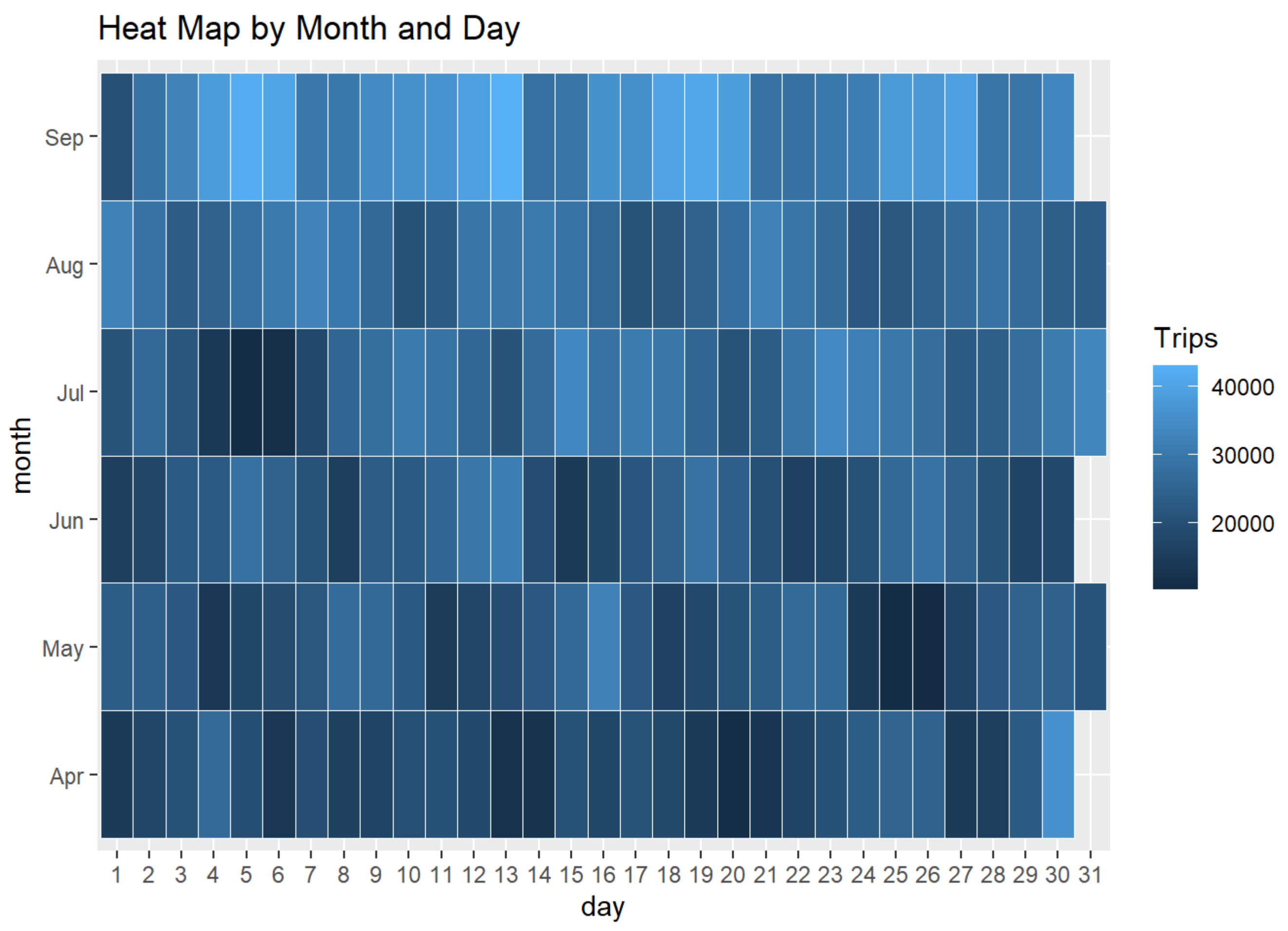
```
`summarise()` has grouped output by 'month'. You can override using the
`.groups` argument.
```

```
month_day_data
```

```
# A tibble: 183 × 3
# Groups:   month [6]
   month day   Trips
   <ord> <fct> <int>
 1 Apr   1     14546
 2 Apr   2     17474
 3 Apr   3     20701
 4 Apr   4     26714
 5 Apr   5     19521
 6 Apr   6     13445
 7 Apr   7     19550
 8 Apr   8     16188
 9 Apr    9     16843
10 Apr   10     20041
# ℹ 173 more rows
```

```
# Plot a heatmap

ggplot(month_day_data, aes(day, month, fill = Trips)) +
geom_tile(color = "white") +
ggtitle("Heat Map by Month and Day")
```

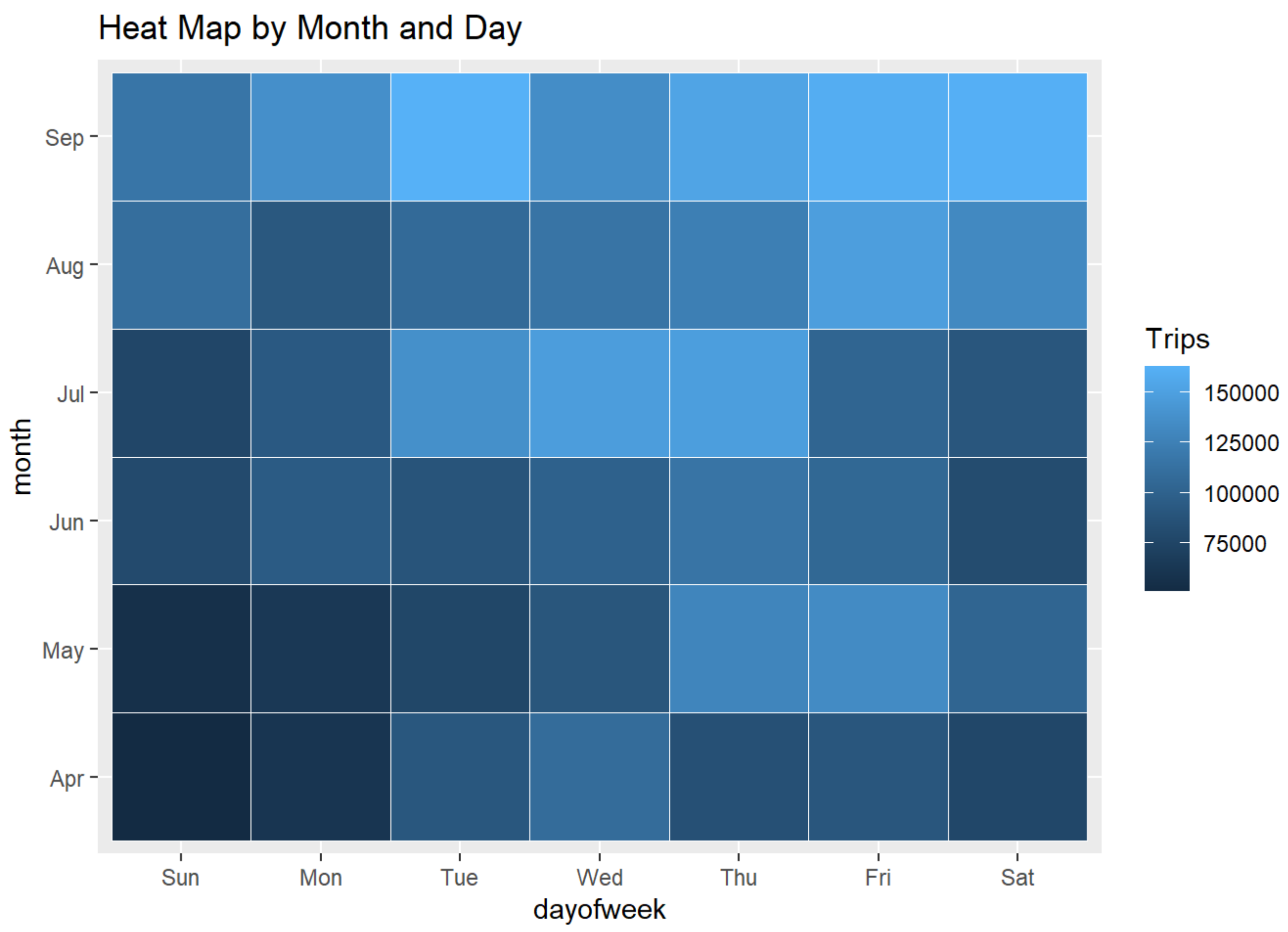


```
##plot an interactive heat map
#df <- normalize(month_day_data)
#heatmaply(month_day_data)
```

Heatmap by day of the week and month

```
# Plot a heatmap by day of the week and month

ggplot(day_month_data, aes(dayofweek, month, fill = Trips)) +
geom_tile(color = "white") +
ggtitle("Heat Map by Month and Day")
```



Map visualizations

```
# Set Map Constants
min_lat <- 40
max_lat <- 40.91
min_long <- -74.15
max_long <- -73.7004
```

```
ggplot(df, aes(x=Lon, y=Lat)) +
  geom_point(size=1, color = "coral") +
     scale_x_continuous(limits=c(min_long, max_long)) +
      scale_y_continuous(limits=c(min_lat, max_lat)) +
        theme_map() +
           ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)")
```

```
Warning: Removed 70180 rows containing missing values (`geom_point()`).
```

NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)