An Analysis of New York Uber Trips with R

Jeffrey Ohene

2022-08-23

Figure 1: Uber

This analysis will be focused on extracting insights from data collected in New York by rides company Uber. The analysis will be focused on central elements such as highest number of rides by month, week, weekday, hour among othermore in-depth findings like averages.

To begin with, we would need to load packages that will aid us in the analysis to be converted. For this analysis, we will use readr, tidyverse, dplyr, ggplot2, lubridate, scales and ggcharts.

Readr will help us import the datasets into R, tidyverse will be used to do some data cleaning, dplyr, to manipulate our dataset to answer a range of questions, ggplot2 to help us build visuals, lubridate will also help us deal with the dates in our dataset, scales wil help us scale our visuals in a more understandable way and ggcharts will be used to build simple and relatively aesthetically pleasing visuals.

```
library(readr)
library(tidyverse)
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify current timezone 'C':
## please set environment variable 'TZ'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v dplyr   1.0.9
## v tibble  3.1.7     v stringr 1.4.0
## v tidyr   1.2.0     v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Figure 2: New York Statue of Liberty

```
library(dplyr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(ggcharts)
```

We would then need to load the datasets for April, May, June, July, August and September into R. As we want to treat the datasets as a single dataset for more efficient analysis, we will then bind them together into a single dataframe for further analysis.

```
#Import datasets
apr <- read_csv("1uber-raw-data-apr14.csv")
```

```
## Rows: 564516 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```
may <- read_csv("2uber-raw-data-may14.csv")
```

```
## Rows: 652435 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```r
jun <- read_csv("3uber-raw-data-jun14.csv")
```

```
## Rows: 663844 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```r
jul <- read_csv("4uber-raw-data-jul14.csv")
```

```
## Rows: 796121 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```r
aug <- read_csv("5uber-raw-data-aug14.csv")
```

```
## Rows: 829275 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```r
sep <- read_csv("6uber-raw-data-sep14.csv")
```

```
## Rows: 1028136 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Date/Time, Base
## dbl (2): Lat, Lon
##
## 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.
```

```r
#Join datasets into a single dataframe
df <- rbind(
  apr,may,jun,jul,aug,sep
  )

df
```

```
## # A tibble: 4,534,327 x 4
##    `Date/Time`        Lat   Lon Base
##    <chr>            <dbl> <dbl> <chr>
##  1 4/1/2014 0:11:00  40.8 -74.0 B02512
##  2 4/1/2014 0:17:00  40.7 -74.0 B02512
##  3 4/1/2014 0:21:00  40.7 -74.0 B02512
##  4 4/1/2014 0:28:00  40.8 -74.0 B02512
##  5 4/1/2014 0:33:00  40.8 -74.0 B02512
##  6 4/1/2014 0:33:00  40.7 -74.0 B02512
##  7 4/1/2014 0:39:00  40.7 -74.0 B02512
##  8 4/1/2014 0:45:00  40.8 -74.0 B02512
##  9 4/1/2014 0:55:00  40.8 -74.0 B02512
## 10 4/1/2014 1:01:00  40.8 -74.0 B02512
## # ... with 4,534,317 more rows
```

```
#Dataset structure
str(df)
```

```
## spec_tbl_df [4,534,327 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Date/Time: chr [1:4534327] "4/1/2014 0:11:00" "4/1/2014 0:17:00" "4/1/2014 0:21:00" "4/1/2014 0:28
##  $ Lat      : num [1:4534327] 40.8 40.7 40.7 40.8 40.8 ...
##  $ Lon      : num [1:4534327] -74 -74 -74 -74 -74 ...
##  $ Base     : chr [1:4534327] "B02512" "B02512" "B02512" "B02512" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Date/Time` = col_character(),
##   ..   Lat = col_double(),
##   ..   Lon = col_double(),
##   ..   Base = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

A peak at our dataset structure shows our Date/Time column is in a character format. We would need to convert that to a proper date-time format before conducting any further action on our dataset. We will then use lubridate to create new columns for month, day, week day and hour. We will express the month and weekday columns as categorical variables since they have a finite amount of groups, that is, week days and months. The days of the month will be expressed in an integer format, that is to represent the 1-30/31 days in a month

```
#Convert Date/Time column to proper date time format
df$Date_Time<-as.POSIXct(
  df$`Date/Time`,format="%m/%d/%Y %H:%M:%S"
  )
df
```

```
## # A tibble: 4,534,327 x 5
##    `Date/Time`        Lat   Lon Base   Date_Time
##    <chr>            <dbl> <dbl> <chr>  <dttm>
##  1 4/1/2014 0:11:00  40.8 -74.0 B02512 2014-04-01 00:11:00
##  2 4/1/2014 0:17:00  40.7 -74.0 B02512 2014-04-01 00:17:00
##  3 4/1/2014 0:21:00  40.7 -74.0 B02512 2014-04-01 00:21:00
##  4 4/1/2014 0:28:00  40.8 -74.0 B02512 2014-04-01 00:28:00
##  5 4/1/2014 0:33:00  40.8 -74.0 B02512 2014-04-01 00:33:00
```

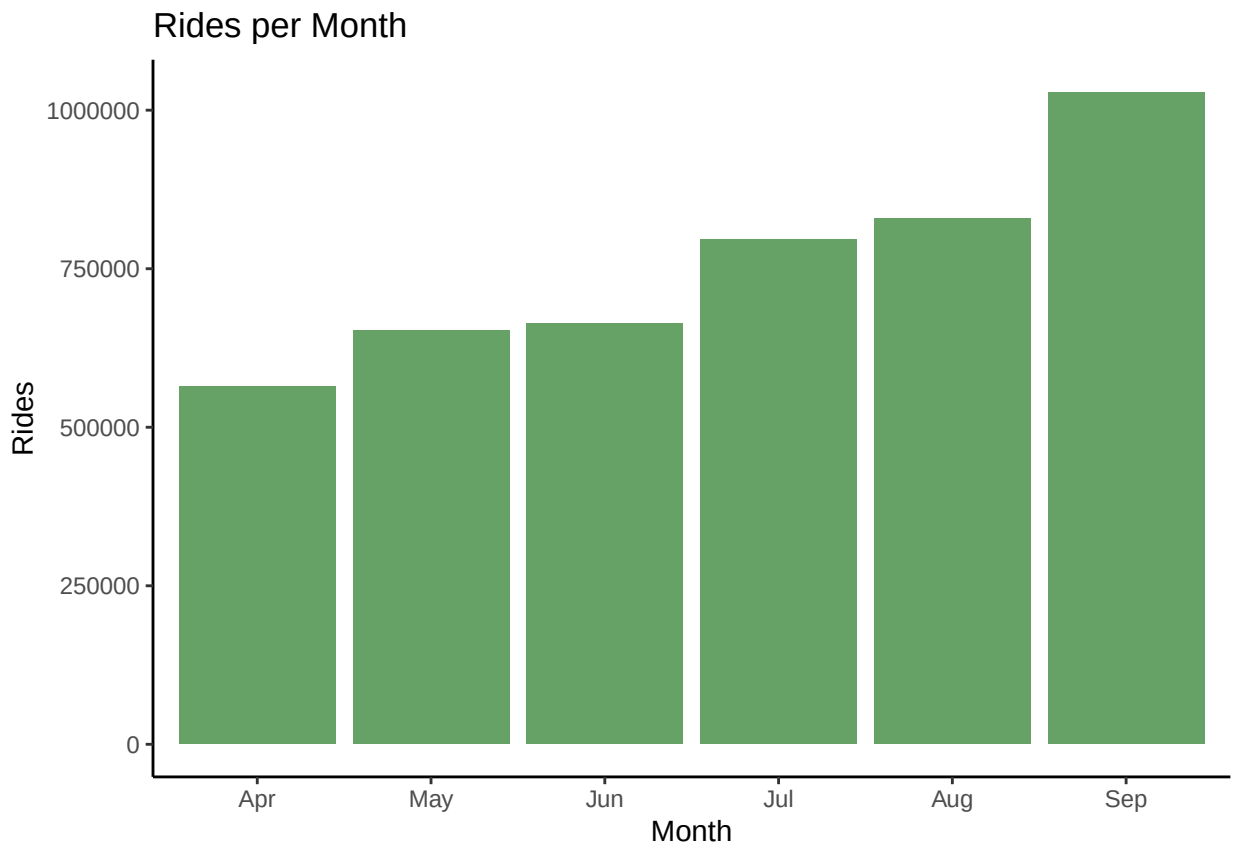
```
##  6 4/1/2014 0:33:00  40.7 -74.0 B02512 2014-04-01 00:33:00
##  7 4/1/2014 0:39:00  40.7 -74.0 B02512 2014-04-01 00:39:00
##  8 4/1/2014 0:45:00  40.8 -74.0 B02512 2014-04-01 00:45:00
##  9 4/1/2014 0:55:00  40.8 -74.0 B02512 2014-04-01 00:55:00
## 10 4/1/2014 1:01:00  40.8 -74.0 B02512 2014-04-01 01:01:00
## # ... with 4,534,317 more rows
```

```
#Add new columns for month, day, week day and hour
df$Month <- factor(
  month(df$Date_Time,label=TRUE)
  )
df$Day <- as.integer(
  format(df$Date_Time,"%d")
  )
df$Week_Day <- factor(wday(
  df$Date_Time,label=TRUE)
  )
df$Hour <- as.integer(
  format(df$Date_Time,"%H")
  )

df
```

```
## # A tibble: 4,534,327 x 9
##    `Date/Time`    Lat   Lon Base  Date_Time           Month   Day Week_Day  Hour
##    <chr>        <dbl> <dbl> <chr> <dttm>              <ord> <int> <ord>    <int>
##  1 4/1/2014 0:~  40.8 -74.0 B025~ 2014-04-01 00:11:00 Apr       1 Tue          0
##  2 4/1/2014 0:~  40.7 -74.0 B025~ 2014-04-01 00:17:00 Apr       1 Tue          0
##  3 4/1/2014 0:~  40.7 -74.0 B025~ 2014-04-01 00:21:00 Apr       1 Tue          0
##  4 4/1/2014 0:~  40.8 -74.0 B025~ 2014-04-01 00:28:00 Apr       1 Tue          0
##  5 4/1/2014 0:~  40.8 -74.0 B025~ 2014-04-01 00:33:00 Apr       1 Tue          0
##  6 4/1/2014 0:~  40.7 -74.0 B025~ 2014-04-01 00:33:00 Apr       1 Tue          0
##  7 4/1/2014 0:~  40.7 -74.0 B025~ 2014-04-01 00:39:00 Apr       1 Tue          0
##  8 4/1/2014 0:~  40.8 -74.0 B025~ 2014-04-01 00:45:00 Apr       1 Tue          0
##  9 4/1/2014 0:~  40.8 -74.0 B025~ 2014-04-01 00:55:00 Apr       1 Tue          0
## 10 4/1/2014 1:~  40.8 -74.0 B025~ 2014-04-01 01:01:00 Apr       1 Tue          1
## # ... with 4,534,317 more rows
```

Our first analysis, we would like to visualize the number of trips per month to see the months with the highest and lowest trips.

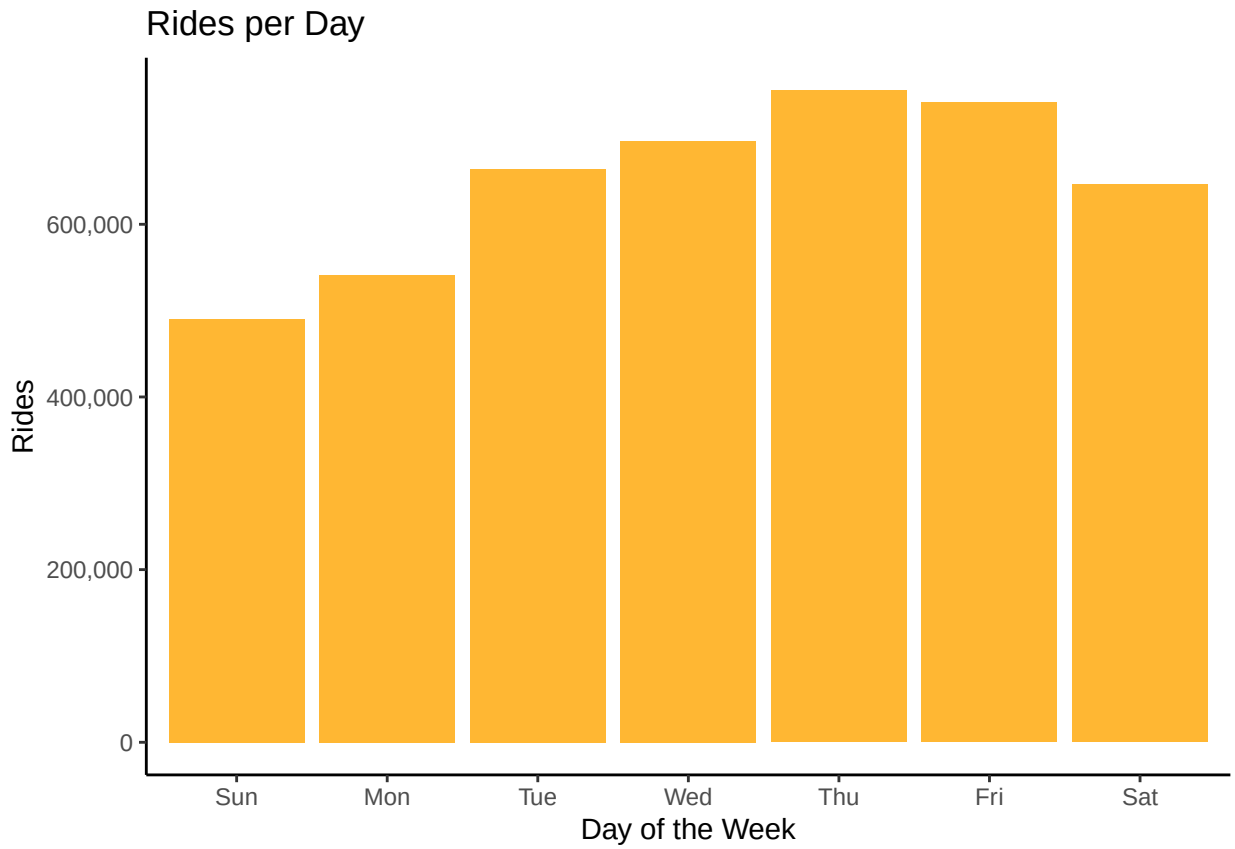
```
#Trips by Month
trips_month <- df%>%
  group_by(Month)%>%
  summarize(tot_mrides=n()) %>%
  ggplot(aes(Month,tot_mrides))+
  geom_col(fill="darkgreen",alpha=0.6)+
  labs(
    x="Month",
    y="Rides",
  title="Rides per Month")+
  theme_classic()

trips_month
```



As we can see, the total number of rides gradually increased as the year went by. April had the lowest number of rides and in September, the highest rides total was recorded surpassing 1 million rides.

Now that we know September had the highest number of total rides, wouldn't we want to see what day of the week people booked a lot of rides or the fewest number of rides? Do people book more trips on say Friday than any day of the week? Or is it Monday? Enough suspense! Let's find out!

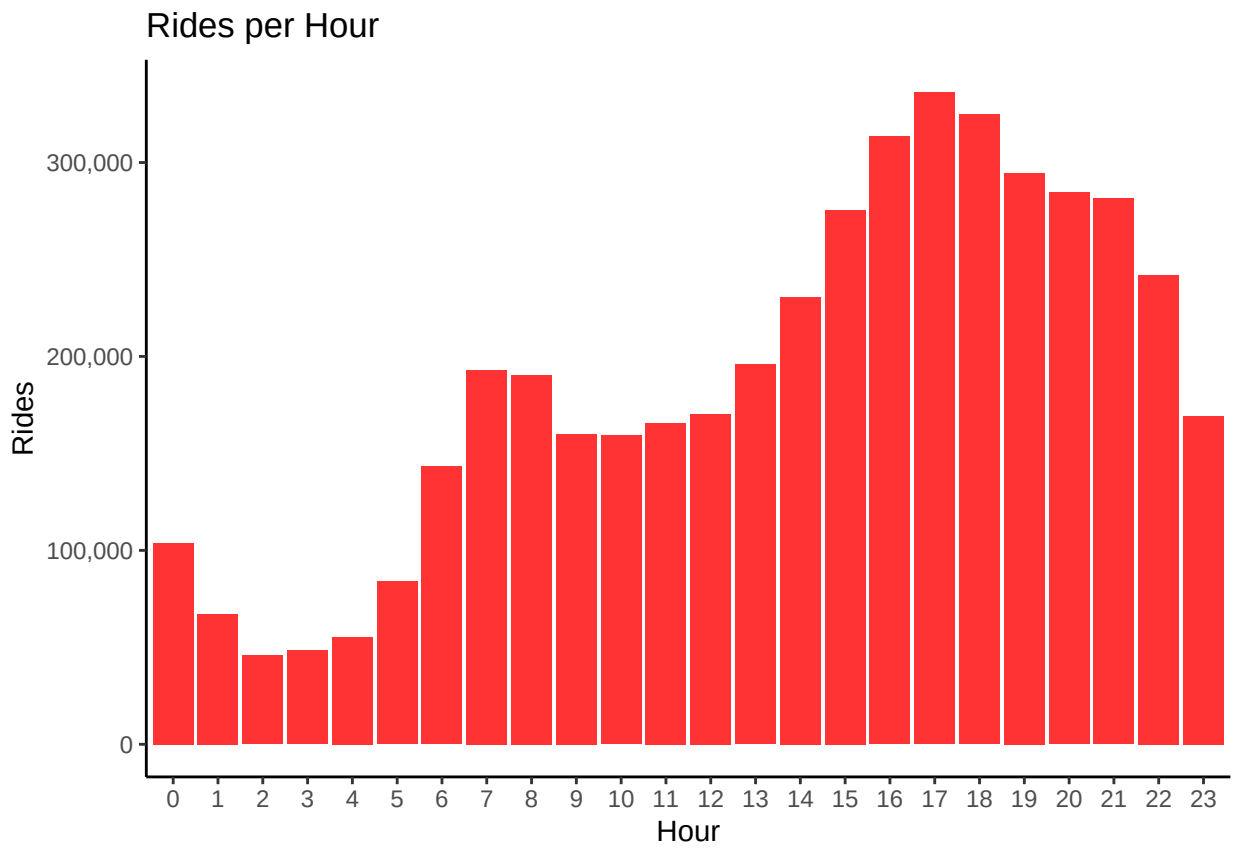
```
#Trips by Weekday
trips_wday <- df%>%
  group_by(Week_Day)%>%
  summarize(tot_wrides=n()) %>%
  ggplot(aes(Week_Day,tot_wrides))+
  geom_col(fill="orange",alpha=0.8)+
  labs(
    x="Day of the Week",
    y="Rides",
    title="Rides per Day")+
  scale_y_continuous(labels=comma)+
  theme_classic()

trips_wday
```



Interestingly, the week day with the highest number of rides was Thursday. Friday comes in close second followed by Wednesday. An interesting pattern the visual above shows us is that, the total number of rides per day peak from Wednesday to Friday, then steeply falls off on Saturday then recovers slowly to peak again on Thursday.

Okay now time for some more deeper insights. Does our data have anything else to tell us? We have found the month and day of the week with the highest number of trips but what about the hour of the day in which most rides happen in? Will that reveal something else?

```
#Trips by hour
trips_hour <- df%>%
  group_by(Hour)%>%
  summarize(tot_hrides=n()) %>%
  ggplot(aes(factor(Hour),tot_hrides))+
  geom_col(fill="red",alpha=0.8)+
  labs(
    x="Hour",
    y="Rides",
    title="Rides per Hour")+
  scale_y_continuous(labels=comma)+
  theme_classic()

trips_hour
```

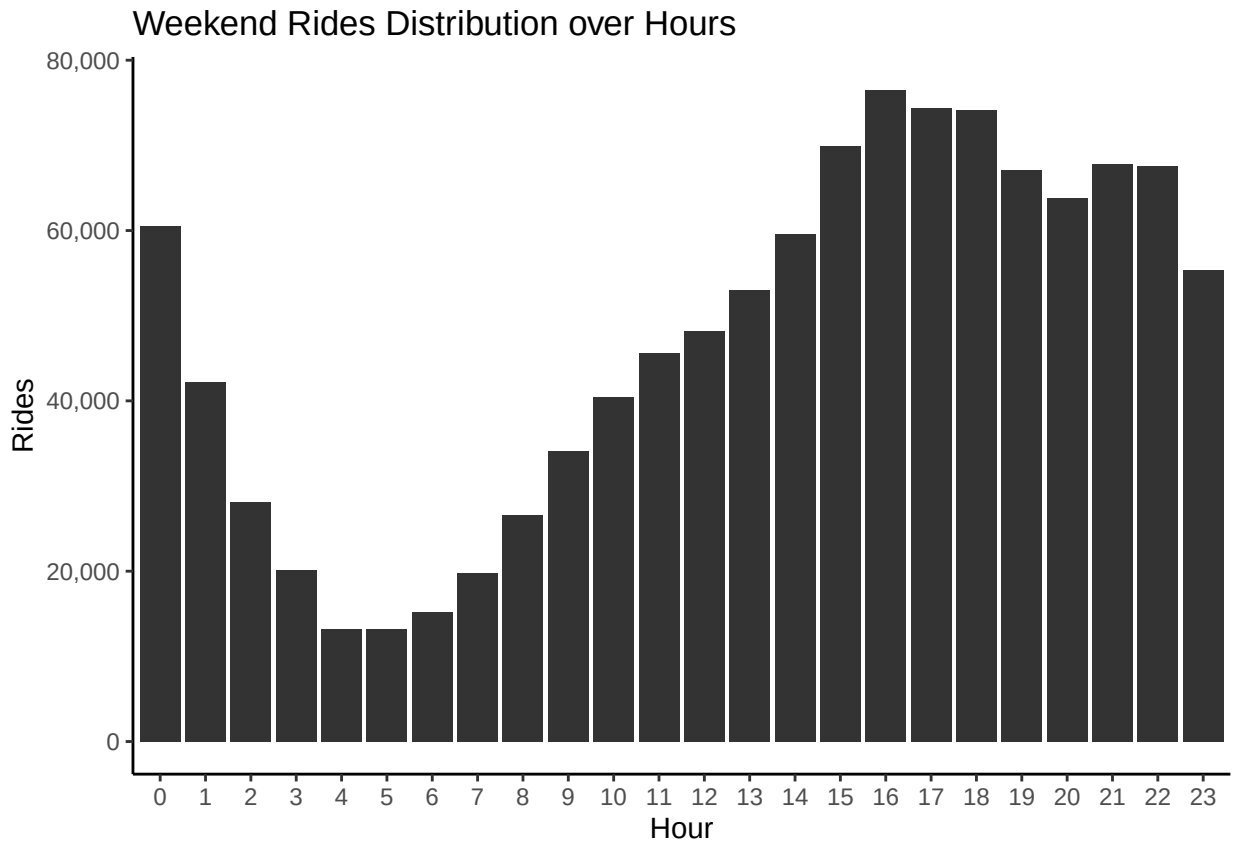


From our above visual, we definitely have some even more interesting insights we have uncovered. To begin with if we group our data into two parts; 7:00 - 9:00 and 16:00 - 18:00, we can presume that, more New Yorkers use Uber in their evening commute from work than to work. 17:00 recorded more trips than any other hour and we can see a steady decline in trips from that hour to 02:00, the hour of the day with the lowest number of rides, 45,865. This shows us that in New York, Uber is heavily used at every single hour in any given day.

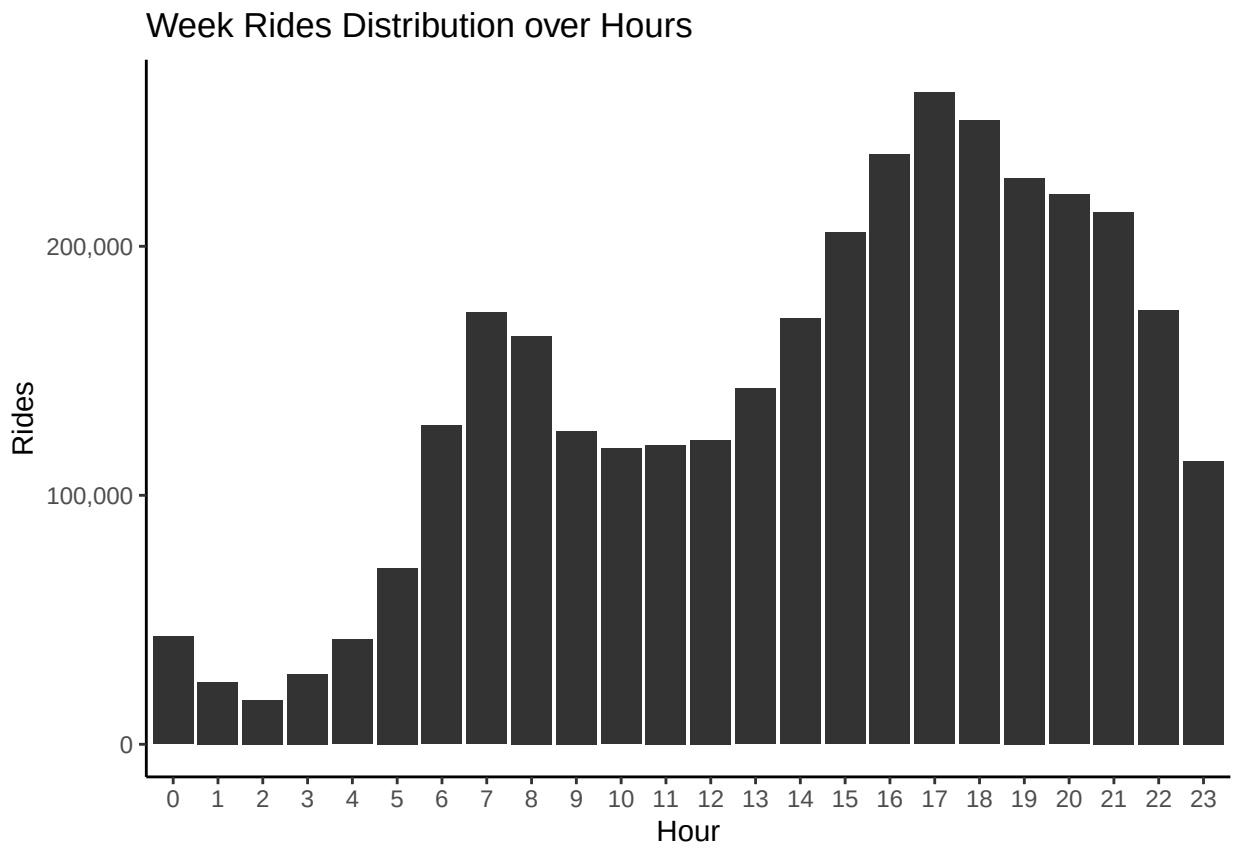
We will now want to check the pattern of weekend trips against working day trips to check for any pattern variation.

```
#Working Day Trips
wkd_trips <- df %>%
  select(Week_Day,Hour)%>%
  filter(Week_Day %in% c("Sat","Sun"))%>%
  group_by(Hour)%>%
  summarize(wkd_hour=n())%>%
  ggplot(aes(factor(Hour),wkd_hour))+
  geom_col(fill="black",alpha=0.8)+
  labs(
    x="Hour",
    y="Rides",
    title="Weekend Rides Distribution over Hours")+
  scale_y_continuous(labels=comma)+
  theme_classic()


#Weekend Trips
```

```
wkd_trips
```

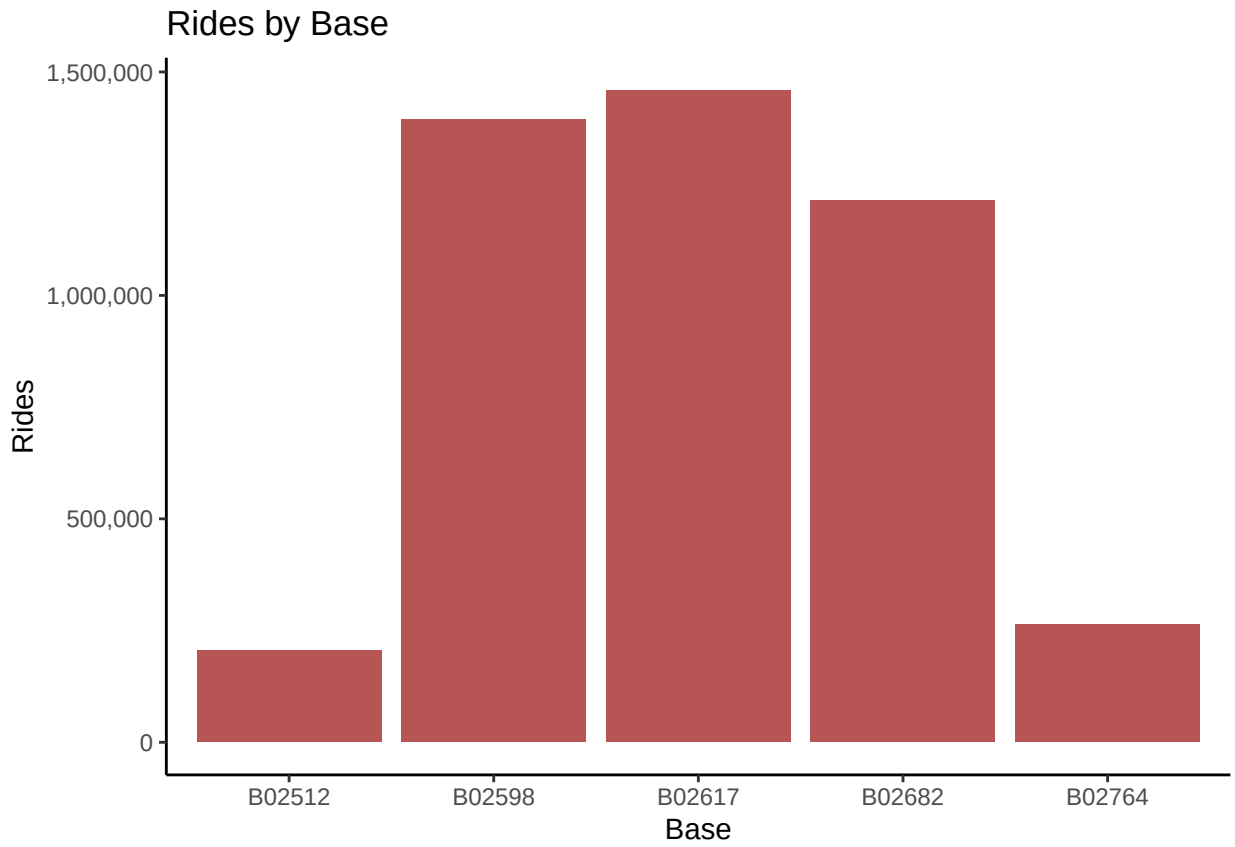



```
wday_trips <- df %>%
  select(Week_Day,Hour)%>%
  filter(!Week_Day %in% c("Sat","Sun"))%>%
  group_by(Hour)%>%
  summarize(wk_hour=n())%>%
  ggplot(aes(factor(Hour),wk_hour))+
  geom_col(fill="black",alpha=0.8)+
  labs(
    x="Hour",
    y="Rides",
    title="Week Rides Distribution over Hours")+
  scale_y_continuous(labels=comma)+
  theme_classic()

wday_trips
```



Hmm... interesting. The visuals above show us some difference in ride patterns for weekends. First of, the number of rides on weekends is lower than that of weekdays and the number of rides per hour for weekends peak earlier than that for working days. The pattern of working days show a steady but sure decrease in rides from 17:00 to 02:00 but for weekends, the opposite is true. There is a relatively high and sustained rate of rides from 16:00 through to 00:00. There is then a steady decrease from 0:00 to 05:00 then a consistent increase in rides till 16:00 again, completely different to weekdays where rides are peak at 17:00 and fall off till about 02:00 and increase again till 07:00 and drops till 10:00 then records a steady increase again till 17:00, the peak hour for rides in New York on working days.

```
#Per Base

base_tot <- df%>%
  select(Month,Base)%>%
  group_by(Base)%>%
  summarize(b_total=n())%>%
  ggplot(aes(Base,b_total))+
  geom_col(fill="brown",alpha=0.8)+
  labs(
    x="Base",
    y="Rides",
    title="Rides by Base")+
  scale_y_continuous(labels=comma)+
  theme_classic()

base_tot
```

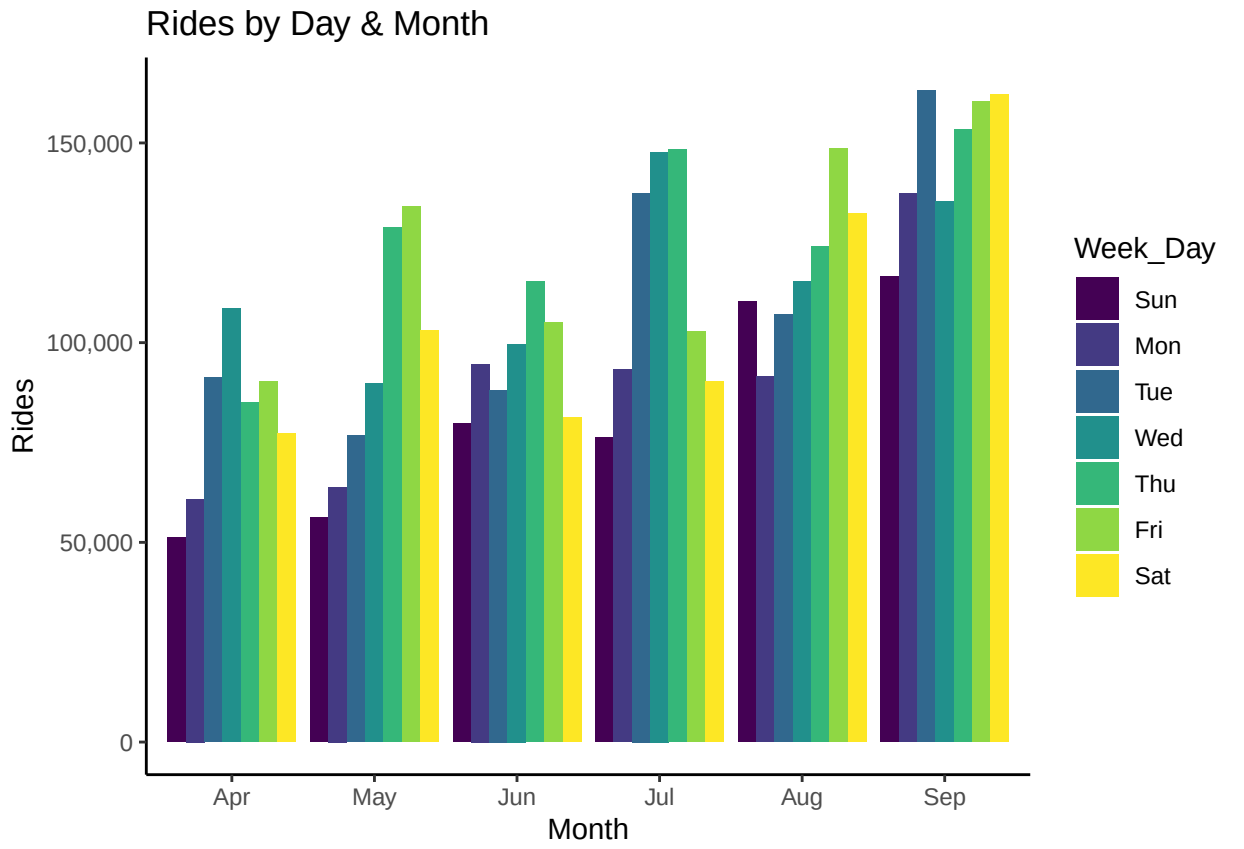


Base B02617 recorded the highest number of total rides with 1.45 million rides taking place from there. Base B02512 in comparison had the lowest number of rides with only 205,673 rides taking place in the base. Base B02617 had a total seven times larger than Base B02512.

```
#Rides by Day & Month
m_d_rides <- df%>%
  select(Month,Week_Day)%>%
  group_by(Month,Week_Day) %>%
  summarize(total=n())%>%
  ggplot(aes(Month,total,fill=Week_Day))+
  geom_bar(stat="identity", position="dodge")+
  labs(
    x="Month",
    y="Rides",
    title="Rides by Day & Month")+
  scale_y_continuous(labels=comma)+
  theme_classic()
```

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

```
m_d_rides
```

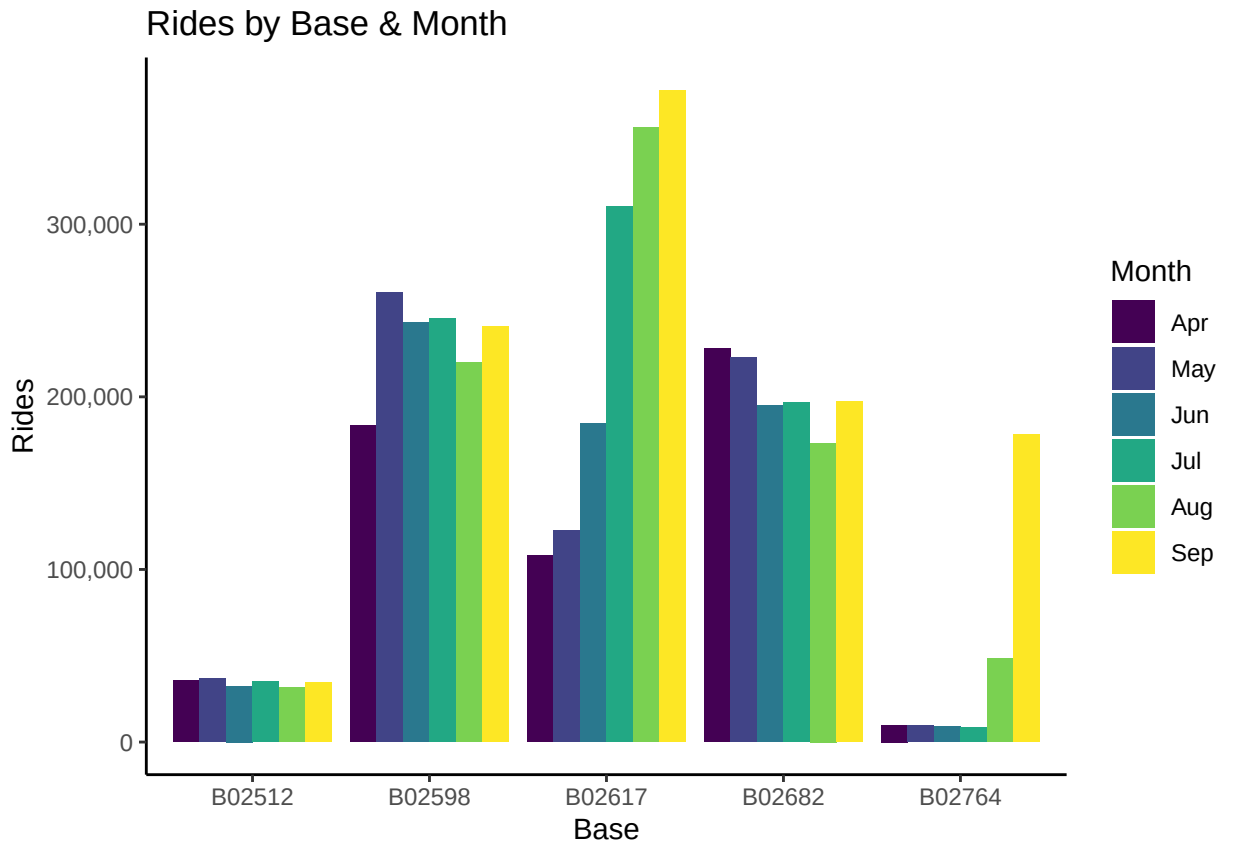


As displayed here, September has the highest number of rides for each weekday out of the other months. It is also interesting to note that, July, despite having a comparatively lower number of total rides than August, ranks next best for total number of rides on Tuesdays, Wednesdays and Thursdays. April ranks lowest for all days while May has the third highest trips on a Friday.

```
#Rides by Base
rides_base <- df%>%
  select(Base,Month)%>%
  group_by(Base,Month) %>%
  summarize(total=n())%>%
  ggplot(aes(Base,total,fill=Month)) +
  geom_bar(stat="identity",position="dodge") +
  labs(
    x="Base",
    y="Rides",
    title="Rides by Base & Month")+
  scale_y_continuous(labels=comma)+
  theme_classic()
```

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

```
rides_base
```



Base B02617, the base with the highest number of total rides was first in total rides for the months July, August and September, with 310,160, 355,803 & 377,695 total rides respectively. There was a gradual increase in the number of rides from April to September. Base B02598 also recorded the highest number of rides for the months May and June with 260,549 and 242,975 respectively. Base B02682 recorded the highest number of rides for April then dropped off in May then in June. Base B02764 recorded the lowest ride total for April, May and June but experienced a surge in rides in September to 178,333, three times its next highest total rides.