March 2024

UBER DATA ANALYSIS REPORT

PREPARED BY

Mohammad Abu Huzaifa

Computer Science (Data Analytics)

Asia Pacific University of Technology & Innovation

CONTACT INFO

+601160646911

Mohammad Abu Huzaifa

Mohammad Abu Huzaifa

ABSTRACT

This documentation provides an exhaustive examination of Uber ride data spanning from April to September 2014, with a specific emphasis on trends and patterns pertaining to ride demand and frequency. By importing, exploring, and visualising data, crucial insights were obtained, such as evening ride peak hours, monthly fluctuations in ride volumes, and ride patterns specific to each base. Uber is advised to optimise its resource allocation during periods of high demand, to concentrate promotional efforts on off-peak days and months, and to enhance its base management strategies. Reliance on 2014 data and a focus on New York City are constraints that indicate the necessity for further investigations employing more recent and diverse datasets in order to augment the applicability and significance of the findings.

Table of Contents

ABSTRACT **2**

1.0 INTRODUCTION **4**

1.1 Data Description 4
1.2 Assumptions 4
1.3 Objectives 4
1.3.1 Methodology *5*
1.3.2 Outcome *5*

2.0 METHODS & TOOLS **5**

3.0 DATA IMPORT **6**

3.1 Importing the Essential Package 6
3.2 Set Working Directory 10
3.3 Creating vector of colours 10
3.4 Read data into their designated variables 11

4.0 DATA EXPLORATION **12**

4.1 Checking Data Type 12
4.2 Print First Six Rows 12
4.3 Print Last Five Rows 13
4.4 Print the names of an object 13
4.5 Print the Dimension of the Dataset 14
4.6 Print the structure of an object 14
4.7 Print summary statistics for the dataset 14

5.0 DATA VISUALIZATION **15**

5.1 Trips by every hours 15
5.2 Trips by Hour & Month 17
5.3 Trips by Every day 19
5.4 Trips by Day and Month 21
5.5 Trips by Month 23
5.6 Trips by Hour and Day 25
5.7 Trips by DayofWeek and Month 27
5.8 Trips by Bases 29
5.9 Trips by Bases and Month 30
5.10 Trips by Bases and Day of Week 31
5.11 Heat Map by Hour and Day 32
5.12 Heat Map by Month and Day 34
5.13 Heat Map by Month and day of week 36
5.14 Heat map by Month and Bases 37
5.15 Heat map by Bases and Day of Week 39
5.16 NYC Map based of Uber Rides During 2014 (APR-SEP) 40
5.17 NYC Map Based on Uber Rides During 2014 (APR-SEP) by Base 41

6.0 CONCLUSION **43**

6.1 Interpret the Results 43
6.2 Recommendation 43
6.3 Limitations and Future Direction 43

DATASET LINK **44**

REFERENCE **44**

1.0 Introduction

With reference to the Uber data analysis dataset, this documentation offers a comprehensive overview of the procedures used for data import, exploration, and visualization for Uber rides during the month of April to September in 2014 in New York City. The foundation for additional in-depth analyses and insights is laid by users' increased comprehension of the dataset's structure, contents, and possible analytical opportunities through these processes.

1.1 Data Description

The Uber data analysis dataset presents comprehensive details on customer rides or journeys taken during the April, May, June, July, August, and September months of 2014 in New York City. This dataset is divided into six CSV files, with each file containing data specific to a particular month. The dataset provides a comprehensive and in-depth summary of Uber ride trends throughout this period, with a whopping 17,128,659,371 rows and 4 columns in a single file.

The large size of the information makes it possible to thoroughly examine and analyze a number of Uber ride-related factors, such as frequency, length, pickup and drop-off locations, and maybe useful patterns or trends. Through the utilization of this dataset, analysts and researchers may acquire significant understanding regarding consumer behavior, demand trends, and the general dynamics of Uber's operations during the given timeframe.

1.2 Assumptions

It may be inferred that over the designated six-month period in 2014, there were specific patterns and trends in the number of Uber trips. These visualisations probably provide information on the busiest times of day for Uber trips as well as the popularity of the service in relation to various months, days of the week, and places (shown by bases). The heat maps may also show regions with a larger volume of Uber rides, which is helpful information for figuring out how popular Uber services are throughout New York City at that particular moment.

1.3 Objectives

The aim of this documentation is to present a thorough overview of the data analysis project for Uber. The project's goal was to examine Uber ride data for the months of April, May, June,

July, August, and September of 2014in New York City. I want to learn more about the patterns and trends of Uber trips throughout these months through this study.

1.3.1 Methodology

The project was divided into three main phases: data import, data exploration, and data visualization.

- I used several project-specific required packages, established the working directory, made a vector of colors, and read the data into the appropriate variables for the data import stage.
- I checked and analyzed several variables during the data exploration phase, such as data types, printing the first six rows, the last five rows, object names, dataset dimensions, object structures, and creating summary statistics for the dataset.
- I used the ggplot2 and geom_point tools to visualise the data during the data visualisation stage. To visualise the data in sixteen various ways, this involved making bar plots, graph, heat maps, and maps.

1.3.2 Outcome

The project's final product was a thorough analysis of the Uber ride data for the designated half-year of 2014 in New York City. I learned about the patterns and trends in Uber trips over this period through data exploration and visualisation, which may be helpful for future research or decision-making procedures.

2.0 Methods & Tools

The dataset will explored and analysed using an integrated development environment for the R programming language called RStudio .For this project, we used RStudio to analyse the Uber dataset. For analysing, we used different types of packages such as ‘ggplot2’, ‘dplyr’, ‘ggthemes’, ‘lubridate’, ‘tidyr’, ‘DT’, ‘scales’ etc. And then we use lot of function to more visualise the data such as ‘geom_bar’, ‘geom_point’, ‘geom_box’, ‘plot’, ‘aes’, ‘print’, ‘summary’ etc.

3.0 Data Import

Data import is the process of bringing data into a software environment for analysis, manipulation, and visualization. In R, data import typically involves reading data from external sources such as files (CSV, Excel, text file), databases, or web APIs into R objects like data frames or matrices. Data import is an essential step in the data analysis process, as it allows you to work with real-world data and perform various analyses to gain insights and make informed decisions.

3.1 Importing the Essential Package

In the first step of our R project, we will import the essential packages that we will use in this Uber Data Analysis project. Some of the important packages installed and libraries of R that we will use are –

```
library(ggplot2)
install.packages("ggthemes")
library(ggthemes)
install.packages("lubridate")
base::date()
base::intersect()
base::setdiff()
base::union()
library(lubridate)
library(dplyr)
library(tidyr)
install.packages("DT")
library(DT)
library(scales)
```

- **ggplot2**

This is the backbone of this project. ggplot2 is the most popular data visualization library that is most widely used for creating aesthetic visualization plots. Some of the examples or reasons we use ggplot2 and scenarios where it's beneficial include:

Customization: ggplot2 allows for highly customizable plots, making it easy to modify it easy to create complex visualizations.

Layering: We can add multiple layers to a plot, including points, lines bars, and text, making it easy to create complex visualizations.

Faceting: ggplot2 supports faceting, allowing us to create multiple plots based on different subsets of the data, which can be helpful for exploring patterns in the data.

Themes: ggplot2 includes themes that control the overall appearance of the plot, making it easy to create publication-quality graphics.

Compatibility: ggplot2 works well with other tidyverse packages making it easy to integrate with our data manipulation workflow.

- ## ggthemes

This is more of an add-on to our main ggplot2 library. With this, we can create better create extra themes and scales with the mainstream ggplot2 package. Some reasons we use ggthemes and scenarios where it's beneficial include:

Customization: Provides a variety of themes that can be applied to ggplot2 plots, allowing us to customize the appearance of our plots to match the style of our analysis or presentation.

Consistency: By using themes from ggthemes, we can ensure that our plots have a consistent appearance throughout our analysis project, which can improve readability and interpretation.

Publication-quality graphics: The themes provided by ggthemes are designed to produce high-quality, publication-ready graphics, making it easier to create professional-looking plots for reports or publications.

Ease of use: ggthemes is easy to use and integrates seamlessly with ggplot2, allowing us to apply themes to our plots with just a few lines of code.

- ## lubridate

Our dataset involves various time-frames. In order to understand our data in separate time categories, we will make use of the lubridate package. Some reasons we use lubridate and scenarios where it's beneficial include:

Parsing Dates: Provides functions to parse dates from a variety of formats, making it easier to work with date data imported from different sources.

Extracting Components: lubridate allows us to easily extract components of dates and times, such as the year, month, day, hour, minute, and second.

Arithmetic Operations: lubridate supports arithmetic operations on dates and times, allowing us to add or subtract time intervals from dates, calculate differences between dates, and more.

Handling Time Zones: lubridate provides functions to work with time zones, making it easier to handle data with different time zones.

Formatting Dates: lubridate allows us to format dates and times in a variety of ways for display or further analysis.

- **dplyr**

We use the dplyr package in R for data manipulation tasks in our analysis project. Some common functions from dplyr that are frequently used include:

filter(): Used to filter rows based on certain conditions.

select(): Used to select specific columns.

mutate(): Used to create new columns or modify existing ones.

summarize(): Used to summarize data, such as calculating mean, median, etc.

arrange(): Used to reorder rows based on one or more variables.

- **tidyr**

This package will help you to tidy your data. The basic principle of tidyr is to tidy columns where each variable is present in a column, each observation is represented by a row and each value depicts a cell. Some reasons we use tidyr and scenarios where it's beneficial include:

Rephasing Data: tidyr functions like pivot_longer() and pivot_wider() help reshape data from wide to long format and vice versa, which can be useful for certain types of analysis and visualization.

Handling Missing Data: tidyr provides functions like drop_na() and fill() to handle missing values in our data, ensuring that our analysis is not affected by missing data.

Separating and Combining Columns: tidyr functions like separate() and unite() allow us to split and combine columns, which can be useful for working with messy or poorly formatted data.

Working with nested Data: tidyr provides functions like nest() and unnest() for working with nested data structures, which can be common in certain types of data sets.

- **DT**

In R, the DT package provides an interface to the JavaScript library DataTables, which allows for the creation of interactive data tables. We use the DT function in out analysis project when we want to display data in a tabular format with additional features such as sorting, searching,

pagination, and the ability to customize the appearance of the table. Some reasons we use the DT function and scenarios where it's beneficial include:

Interactive Data Exploration: DT tables allow users to interactively explore the data by sorting, searching, and filtering the table, which can be helpful for understanding the data's structure and contents.

Data Presentation: DT tables provide a visually appealing way to present data in reports or dashboards, making it easier for stakeholders to understand the data.

Customization: DT tables can be customized with various options such as custom stylling, row highlighting, and the addition of buttons or dropdowns, allowing for a more tailored and interactive user experience.

Integration with Shiny: If we are building a Shiny web application for our analysis project, the DT package integrates seamlessly with Shiny, allowing us to create interactive data tables that update dynamically based on user input.

- ## Scales

In R, the scales package provides function for scaling and formatting plots. We use the scales package in our analysis project for several reasons:

Scaling: The scales package provides functions to scale data, such as "rescale()" for linearly scaling data to a specified range or "log_trans()" for logarithmic scaling.

Formatting: The scales package provides functions to format data for better readability in plots, such as comma() for formatting numbers with commas or "percent()" for formatting numbers as percentages.

Labels: The scales package provides functions to customize axis labels, such as "label_dollar()" for formatting currency labels or "label_date()" for formatting date labels.

Breaks: The scales package provides functions to customize axis breaks, such as "breaks_comma()" for formatting breaks with commas or "breaks_width()" for setting the width between breaks.

```
> library(ggplot2)
> library(ggthemes)
> library(lubridate)
> library(dplyr)
> library(tidyr)
> library(DT)
> library(scales)
```

3.2 Set Working Directory

In R, the working directory is the default location where R looks for files and where it saves files by default. The working directory is important because it determines where R will read and write files unless a specific path is provided. The "setwd()" function in R is used to set the working directory. For example, our dataset folder name is "Uber dataset" and it's located on "/Users/nafizsp/Documents/Data Analysis" this path line, so our command would like this:

```
setwd('/Users/nafizsp/Documents/Data Analysis/Uber-dataset')
```

3.3 Creating vector of colours

In R, you can create a vector of colours using the "c()" function, which concatenates elements into a vector. In this step of data analysis project, we will create a vector of our colours that will be included in our plotting functions. You can also select your own colours. Here's the command that we use in our dataset to create vector colours.

```
colors <- c("#CC1011", "#665555", "#05a399", "#cfcaca",
            "#f5e840", "#0683c9", "#e075b0",
            "#FF4500", "#008080", "#800080", "#000080",
            "#FFD700", "#808080", "#00FF00", "#008000",
            "#00FFFF", "#0000FF", "#FF00FF", "#800000",
            "#FFFF00", "#00FF00", "#FFA07A", "#40E0D0",
            "#7B68EE", "#FA8072")
```

3.4 Read data into their designated variables

After seeing the working directory, any files you read using relative paths (read.csv("file.csv")) will be read from the specified directory, and any files you save using relative paths (saveRDS(data, "data.rds")) will be saved to the specified directory. In our dataset, there are six csv file, that's why we use 6 different variable to read the dataset with "read.csv("file.csv")" function. All the read command are shown below picture:

```
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")
```

It is important to note that setting the working directory is not always necessary or compulsory, especially if use absolute paths ("read.csv("/Users/nafizsp/Documents/Data Analysis/Uber-dataset/file.csv")) or if you are working in an integrated development environment (IDE) that manages the working directory for you. However, in scripts or projects where you need to manage the working directory explicitly, the "setwd()" function is useful.

After we have read all the csv files, we will combine all of this data into a single data frame called "data_2014". Then, in the next step, we will perform the appropriate formatting of Date.Time column. Then, we will proceed to create factors of time objects like day, month, year etc. Then we will proceed to create factors of time objects like second, hour, minute etc.

```
data_2014 <- rbind(apr_data,may_data, jun_data, jul_data, aug_data, sep_data)
data_2014$Date.Time <- as.POSIXct(data_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S")
data_2014$Time <- format(as.POSIXct(data_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S"),
                         format="%H:%M:%S")
data_2014$Date.Time <- ymd_hms(data_2014$Date.Time)

data_2014$day <- factor(day(data_2014$Date.Time))
data_2014$month <- factor(month(data_2014$Date.Time, label = TRUE))
data_2014$year <- factor(year(data_2014$Date.Time))
data_2014$dayofweek <- factor(wday(data_2014$Date.Time, label = TRUE))

data_2014$hour <- factor(hour(hms(data_2014$Time)))
data_2014$minute <- factor(minute(hms(data_2014$Time)))
data_2014$second <- factor(second(hms(data_2014$Time)))
```

4.0 Data Exploration

In this part, we will explore the dataset so that we can easily analyse the data. Data exploration is the initial phase of data analysis where you explore, summarize, and visualize the main characteristics of a dataset. The goal of data exploration is to understand the data, discover patterns, spot anomalies, and formulate hypotheses for further analysis. Data exploration typically involves the following steps such as, data collection, data cleaning, data summarization, data visualization, pattern discovery, outlier detection, and hypothesis formulation. For our dataset we will summarize the data for this step is:

```
class(data_2014)
head(data_2014)
tail(data_2014, 5)
names(data_2014)
names(apr_data)
dim(data_2014)
str(data_2014)
summary(data_2014)
```

4.1 Checking Data Type

In R, the class() function is used to determine the class(data type) of an object. It returns a character vector indicating the class or classes of the object. For example, our dataset variable is “data_2014” and when I use the class function it will print the class of the dataset.

```
> class(data_2014)
[1] "data.frame"
```

4.2 Print First Six Rows

In R, the head() function is used to view the first few rows of a dataframe or a matrix. It is helpful for quickly inspecting the structure and contents of a dataset. By default, head() displays the first 6 rows of the dataset. You can specify the number of rows to display by providing the desired number as an argument to the function. For example, in our dataset we use default head function to print the first 6 rows of the dataset.

```
> head(data_2014)
           Date.Time     Lat      Lon   Base     Time day month year dayofweek hour minute second
1 2014-04-01 00:11:00 40.7690 -73.9549 B02512 00:11:00   1   Apr 2014       Tue    0     11      0
2 2014-04-01 00:17:00 40.7267 -74.0345 B02512 00:17:00   1   Apr 2014       Tue    0     17      0
3 2014-04-01 00:21:00 40.7316 -73.9873 B02512 00:21:00   1   Apr 2014       Tue    0     21      0
4 2014-04-01 00:28:00 40.7588 -73.9776 B02512 00:28:00   1   Apr 2014       Tue    0     28      0
5 2014-04-01 00:33:00 40.7594 -73.9722 B02512 00:33:00   1   Apr 2014       Tue    0     33      0
6 2014-04-01 00:33:00 40.7383 -74.0403 B02512 00:33:00   1   Apr 2014       Tue    0     33      0
```

4.3 Print Last Five Rows

In R, the tail() function is used to view the last few rows of a data frame or a vector. It is particularly useful when you want to quickly inspect the end of a dataset to see the last observations. The tail() function takes two arguments: the object you want to view and the number of rows you want to display. For example, in our dataset we use 5 so that it will print the last 5 rows of the dataset.

```
> tail(data_2014, 5)
                Date.Time     Lat      Lon   Base     Time day month year dayofweek hour minute second
4534323 2014-09-30 22:57:00 40.7668 -73.9845 B02764 22:57:00  30   Sep 2014       Tue   22     57      0
4534324 2014-09-30 22:57:00 40.6911 -74.1773 B02764 22:57:00  30   Sep 2014       Tue   22     57      0
4534325 2014-09-30 22:58:00 40.8519 -73.9319 B02764 22:58:00  30   Sep 2014       Tue   22     58      0
4534326 2014-09-30 22:58:00 40.7081 -74.0066 B02764 22:58:00  30   Sep 2014       Tue   22     58      0
4534327 2014-09-30 22:58:00 40.7140 -73.9496 B02764 22:58:00  30   Sep 2014       Tue   22     58      0
```

4.4 Print the names of an object

In R, the names() function is used to get or set the names of an object, such as a vector, list, or data frame. It returns a character vector of the names of the elements in the object. If the object has no names, NULL is returned. In our dataset, when we want to see whole merged dataset that is the “data_2014” it will print like this:

```
> names(data_2014)
 [1] "Date.Time" "Lat"       "Lon"       "Base"      "Time"      "day"
"month"     "year"      "dayofweek" "hour"      "minute"    "second"
```

And when we want to see only “apr_data” it will print like this:

```
> names(apr_data)
[1] "Date.Time" "Lat"       "Lon"       "Base"
```

4.5 Print the Dimension of the Dataset

In R, the dim() function is used to get or set the dimensions (number of rows and columns) of an object, such as a matrix or data frame. When used with a matrix or data frame, the dim() function returns a numeric vector with two elements: the number of rows followed by the number of columns.

```
> dim(data_2014)
[1] 4534327      12
```

4.6 Print the structure of an object

In R, the str() function is used to compactly display the internal structure of an R object. It gives a concise summary of the structure of the object, showing the type and length of each component, along with the first few values. The str() function is particularly useful for understanding the structure of complex objects like data frames, lists, and arrays. It helps you quickly see the variables (columns) in a data frame, their types, and the first few values, which can be helpful for data exploration and debugging.

```
> str(data_2014)
'data.frame':   4534327 obs. of  12 variables:
 $ Date.Time: POSIXct, format: "2014-04-01 00:11:00" "2014-04-01 00:17:00" ...
 $ 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" ...
 $ Time     : chr  "00:11:00" "00:17:00" "00:21:00" "00:28:00" ...
 $ day      : Factor w/ 31 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ month    : Ord.factor w/ 6 levels "Apr"<"May"<"Jun"<..: 1 1 1 1 1 1 1 1 1 1 ...
 $ year     : Factor w/ 1 level "2014": 1 1 1 1 1 1 1 1 1 1 ...
 $ dayofweek: Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 3 3 3 3 3 3 3 3 3 3 ...
 $ hour     : Factor w/ 24 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 2 ...
 $ minute   : Factor w/ 60 levels "0","1","2","3",..: 12 18 22 29 34 34 40 46 56 2 ...
 $ second   : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
```

4.7 Print summary statistics for the dataset

In R, the summary() function is used to generate summary statistics for a dataset or a specific variable. The summary() function provides a concise summary of the data, including measures

such as the minimum, 1st quartile, median, mean, 3rd quartile, and maximum for numeric variables, as well as the frequency of each unique value for categorical variables.

```
> summary(data_2014)
   Date.Time                       Lat             Lon             Base              Time
 Min.   :2014-04-01 00:01:00.00   Min.   :39.66   Min.   :-74.93   Length:4534327    Length:4534327
 1st Qu.:2014-05-28 15:16:00.00   1st Qu.:40.72   1st Qu.:-74.00   Class :character   Class :character
 Median :2014-07-17 14:43:00.00   Median :40.74   Median :-73.98   Mode  :character   Mode  :character
 Mean   :2014-07-11 18:49:41.19   Mean   :40.74   Mean   :-73.97
 3rd Qu.:2014-08-27 21:55:00.00   3rd Qu.:40.76   3rd Qu.:-73.97
 Max.   :2014-09-30 22:59:00.00   Max.   :42.12   Max.   :-72.07
 NA's   :2211
      day            month           year           dayofweek          hour            minute
 30     : 167101   Apr : 564264   2014:4532116   Thu     : 754940   17     : 336190   10     :  77757
 12     : 160532   May : 652124   NA's:   2211   Fri     : 740778   18     : 324679   14     :  77161
 16     : 158867   Jun : 663545                  Wed     : 696296   16     : 313400   15     :  77124
 13     : 156767   Jul : 795732                  Tue     : 663651   19     : 294513   13     :  76957
 23     : 155958   Aug : 828805                  Sat     : 645532   20     : 284604   12     :  76849
 (Other):3732891   Sep :1027646                  (Other):1030919   21     : 281460   8      :  76719
 NA's   :   2211   NA's:   2211                  NA's    :   2211   (Other):2699481   (Other):4071760
 second
 0:4534327
```

5.0 Data Visualization

Data visualization is the graphical representation of data to communicate information clearly and efficiently. It involves the use of charts, graphs and maps to visually explore and present patterns, trends, and relationships in the data. Data visualization helps to make complex data more understandable and enables users to identify insights and patterns that may not be apparent from the raw data. Common tools for data visualization in R include ggplot2, plotly, and ggvis.

5.1 Trips by every hours

```
hour_data <- data_2014 %>%
  group_by(hour) %>%
  dplyr::summarize(Total = n())
datatable(hour_data)

ggplot(hour_data, aes(hour, Total)) +
  geom_bar( stat = "identity", fill = "steelblue", color = "red") +
  ggtitle("Trips Every Hour") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma)
```

We will use the ggplot2 function to plot the number of trips that the passengers had made in a day. We will also use dplyr to aggregate our data. In the resulting visualizations, we can understand how the number of passengers fares throughout the day. We observe that the number of trips are higher in the evening around 5:00 and 6:00 PM.

Show 10 entries

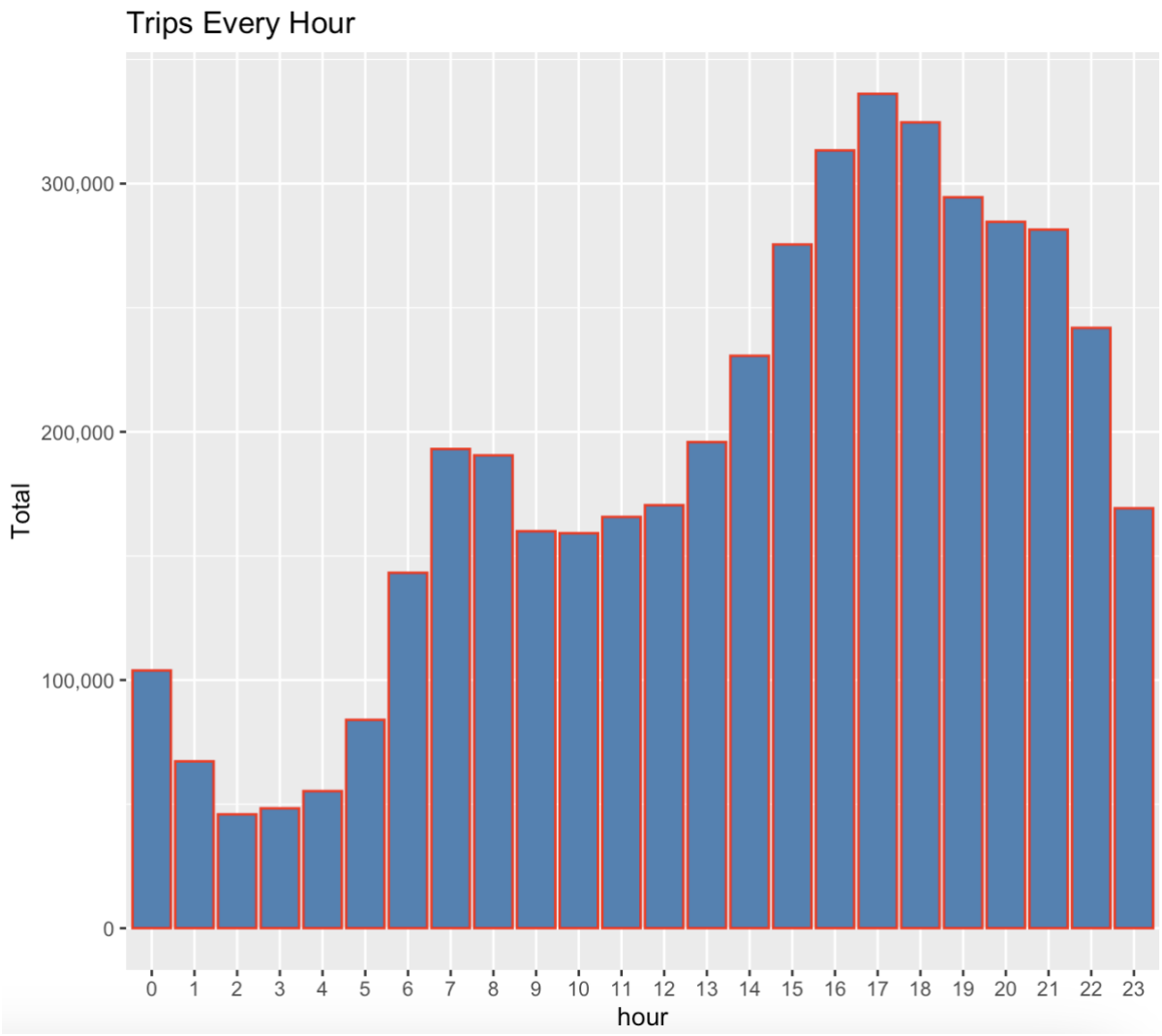
Search:

	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

This is the output of the data table which was made by dplyr function to aggregate the data with hourly basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.2 Trips by Hour & Month

```
month_hour <- data_2014 %>%
  group_by(month, hour) %>%
  dplyr::summarize(Total = n())
datatable(month_hour)

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

In the next step of data visualization, we will use the ggplot2 function to plot the number of trips that the passengers had made by hour and month. We will also use dplyr to aggregate our

data. In the resulting visualizations, we can understand how the number of passengers fares throughout the hour and month. We observe that the number of trips are higher in the evening around 5:00 and 6:00 PM. But, the highest number of trips that the passengers had made around 5:00 PM.

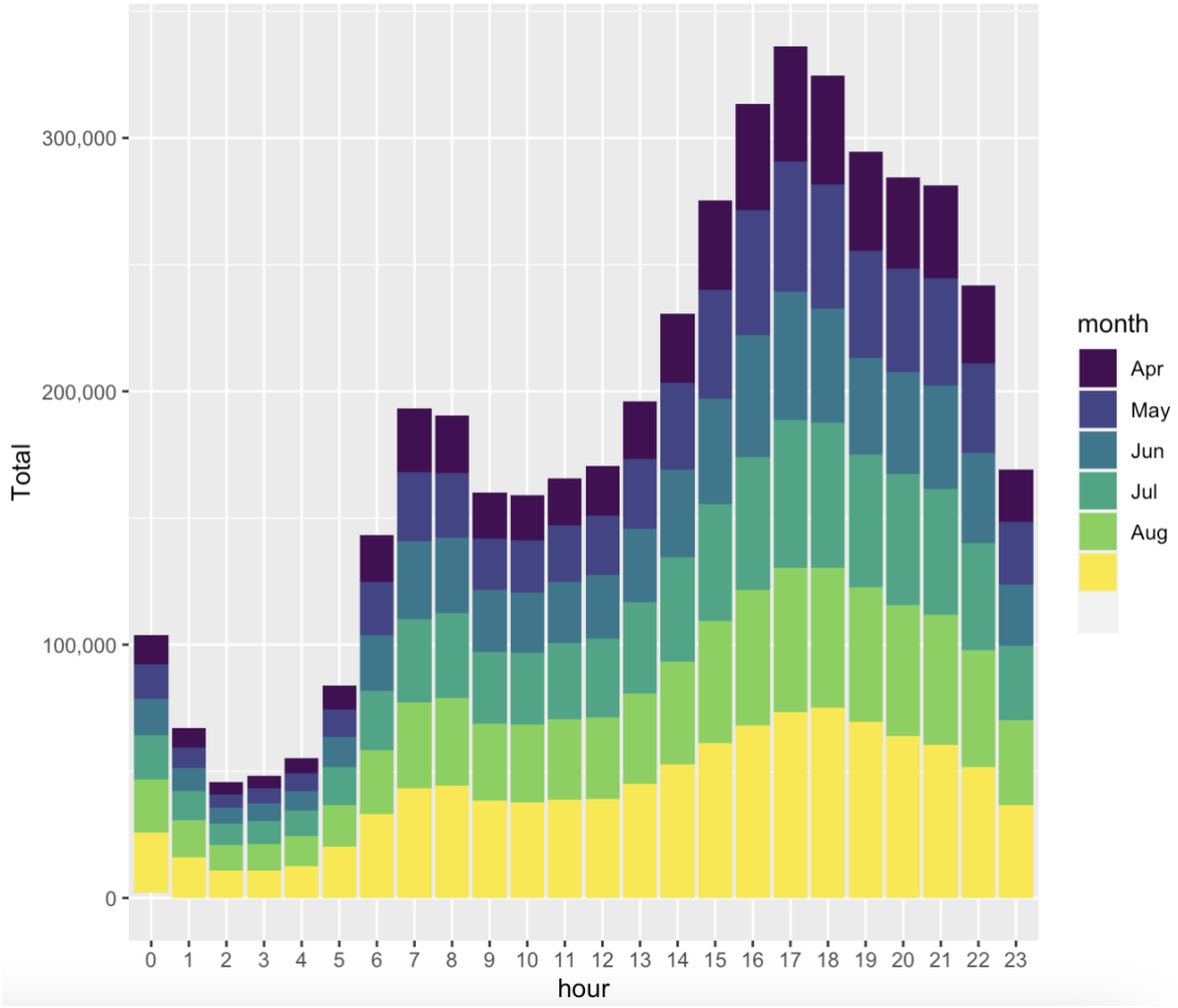
Show 10 entries Search:

	month	hour	Total
1	Apr	0	11658
2	Apr	1	7769
3	Apr	2	4935
4	Apr	3	5040
5	Apr	4	6095
6	Apr	5	9476
7	Apr	6	18498
8	Apr	7	24924
9	Apr	8	22843
10	Apr	9	17939

Showing 1 to 10 of 145 entries

Previous 1 2 3 4 5 … 15 Next

This is the output of the data table which was made by dplyr function to aggregate the data with hourly and monthly basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.3 Trips by Every day

```
day_group <- data_2014 %>%
  group_by(day) %>%
  dplyr::summarize(Total = n())
datatable(day_group)

ggplot(day_group, aes(day, Total)) +
  geom_bar( stat = "identity", fill = "steelblue") +
  ggtitle("Trips Every Day") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma)
```

In the next step of our data analysis project, we will use ggplot2 function to plot the number of trips that the passenger had made by every day of the month. We will aggregate the data by using dplyr function. We observe from the resulting visualization that 30th of the month had the highest trips in the year which is mostly contributed by the month of April.

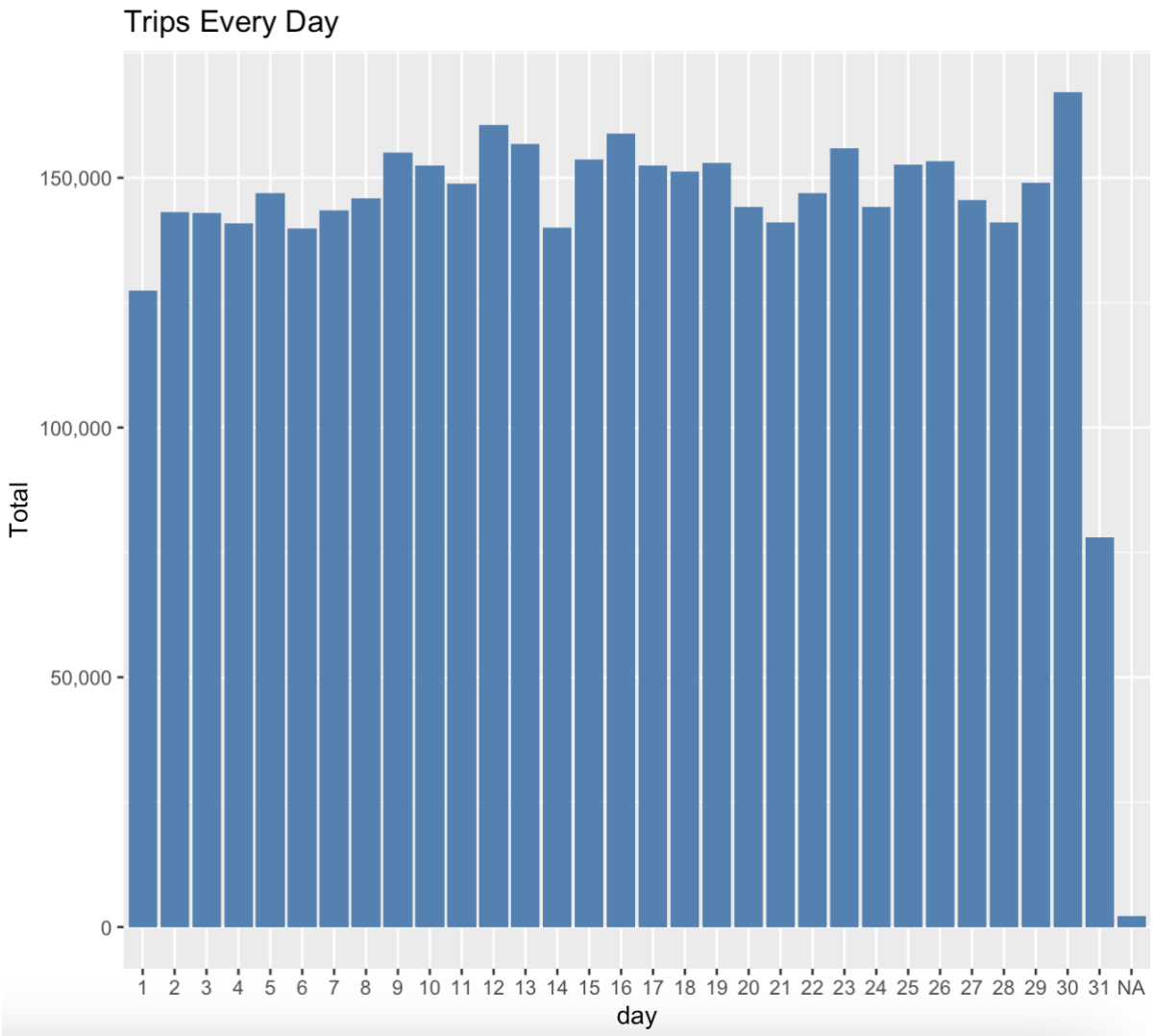
Show 10 entries Search:

	day	Total
1	1	127360
2	2	143146
3	3	142914
4	4	140869
5	5	146970
6	6	139800
7	7	143422
8	8	145932
9	9	155077
10	10	152439

Showing 1 to 10 of 32 entries

Previous 1 2 3 4 Next

This is the output of the data table which was made by dplyr function to aggregate the data with every day basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.4 Trips by Day and Month

```
day_month_group <- data_2014 %>%
  group_by(month, day) %>%
  dplyr::summarize(Total = n())
datatable(day_month_group)

ggplot(day_month_group, aes(day, Total, fill = month)) +
  geom_bar( stat = "identity") +
  ggtitle("Trips by Day and Month") +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)
```

In the next step of our data analysis project, we will use ggplot2 function to plot the number of trips that the passenger had made by day and month. We will aggregate the data by using dplyr

function. In the resulting of data visualization, we observe that the number of trips are higher in the month of September and the day are 5th and 13th.

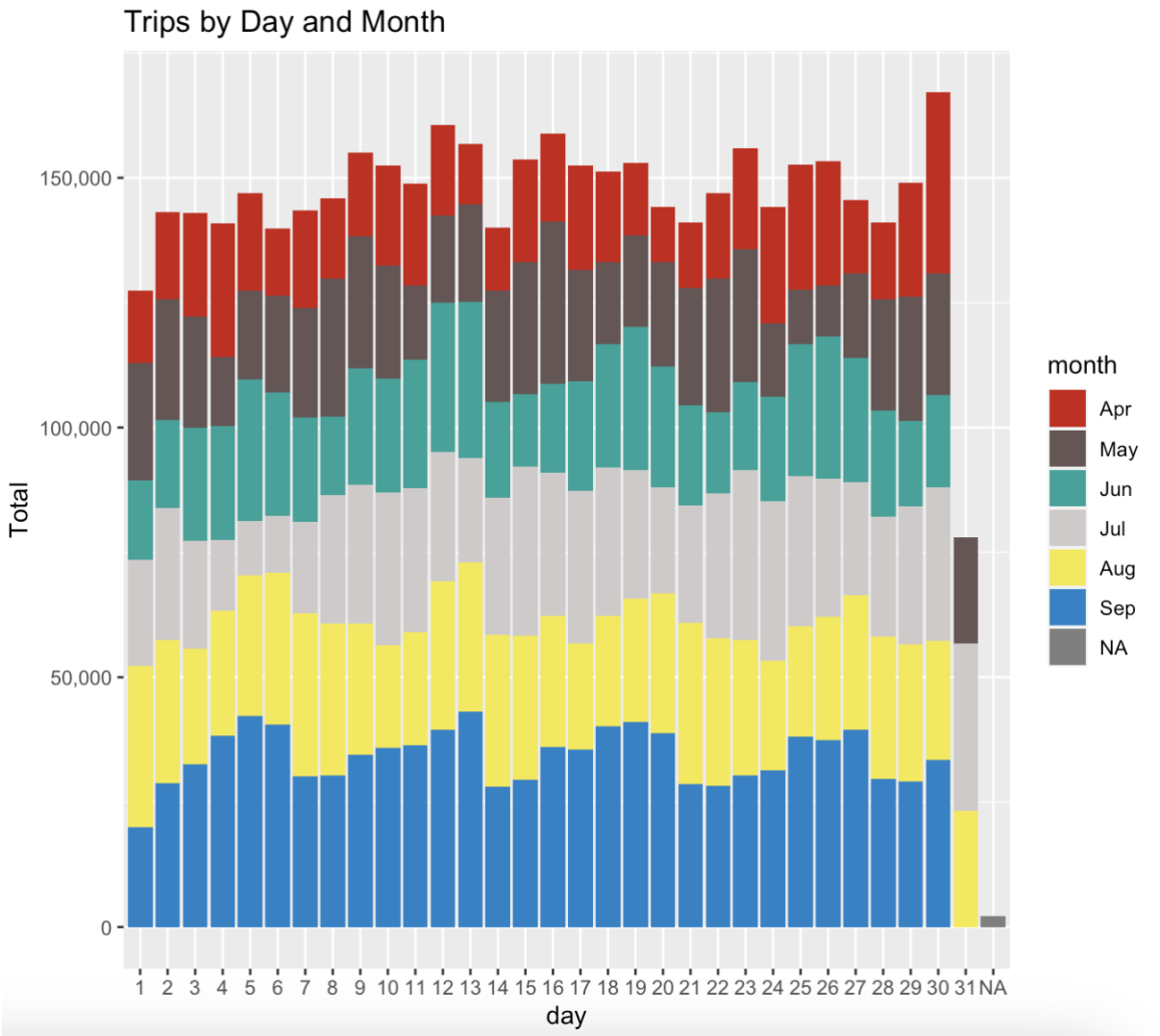
Show 10 entries Search:

	month	day	Total
1	Apr	1	14543
2	Apr	2	17469
3	Apr	3	20697
4	Apr	4	26707
5	Apr	5	19506
6	Apr	6	13415
7	Apr	7	19547
8	Apr	8	16188
9	Apr	9	16837
10	Apr	10	20040

Showing 1 to 10 of 184 entries

Previous 1 2 3 4 5 … 19 Next

This is the output of the data table which was made by dplyr function to aggregate the data with day and month basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.5 Trips by Month

```
month_group <- data_2014 %>%
  group_by(month) %>%
  dplyr::summarize(Total = n())
datatable(month_group)

ggplot(month_group , aes(month, Total, fill = month)) +
  geom_bar( stat = "identity") +
  ggtitle("Trips by Month") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)
```

In the next part of our data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by Month. We will aggregate the data by using dplyr function. In the resulting of data visualization, we observe that the number of trips are higher in the month of September and the number of trips are lower in the month of April.

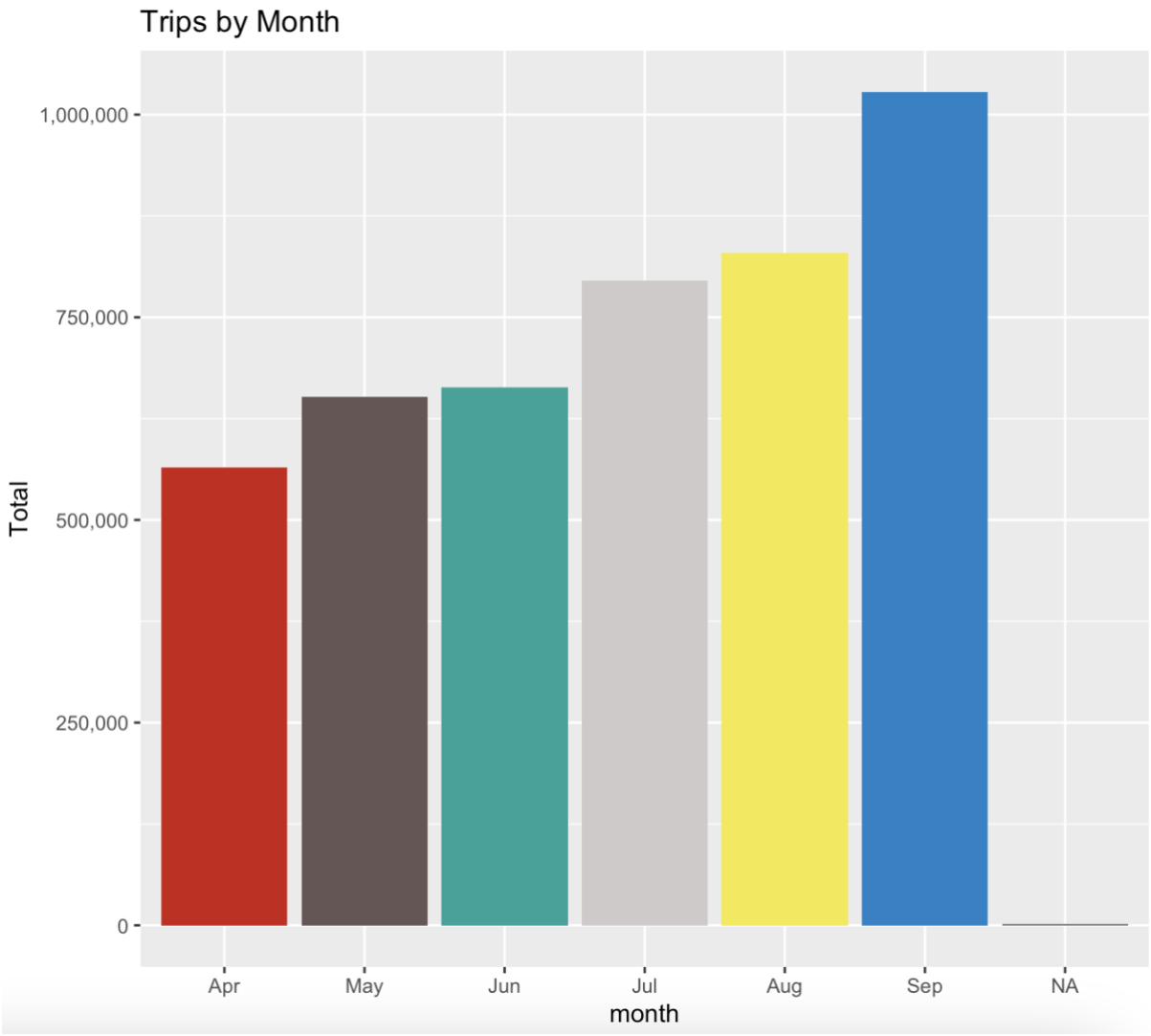
Show 10 entries Search:

	month	Total
1	Apr	564264
2	May	652124
3	Jun	663545
4	Jul	795732
5	Aug	828805
6	Sep	1027646
7		2211

Showing 1 to 7 of 7 entries

Previous 1 Next

This is the output of the data table which was made by dplyr function to aggregate the data with month basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.6 Trips by Hour and Day

```
day_and_hour <- data_2014 %>%
  group_by(day, hour) %>%
  dplyr::summarize(Total = n())
datatable(day_and_hour)
```

```
ggplot(day_and_hour, aes(day, Total, fill = hour)) +
  geom_bar( stat = "identity") +
  ggtitle("Trips by Hour and Day") +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)
```

In the next part of our data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by Hour and Day. We will aggregate the data by using

dplyr function. In the resulting of data visualization, we observe that the number of trips are higher in the day of 30th and the time in around evening, and the number of trips are lower in the day of 31st and the time around midnight.

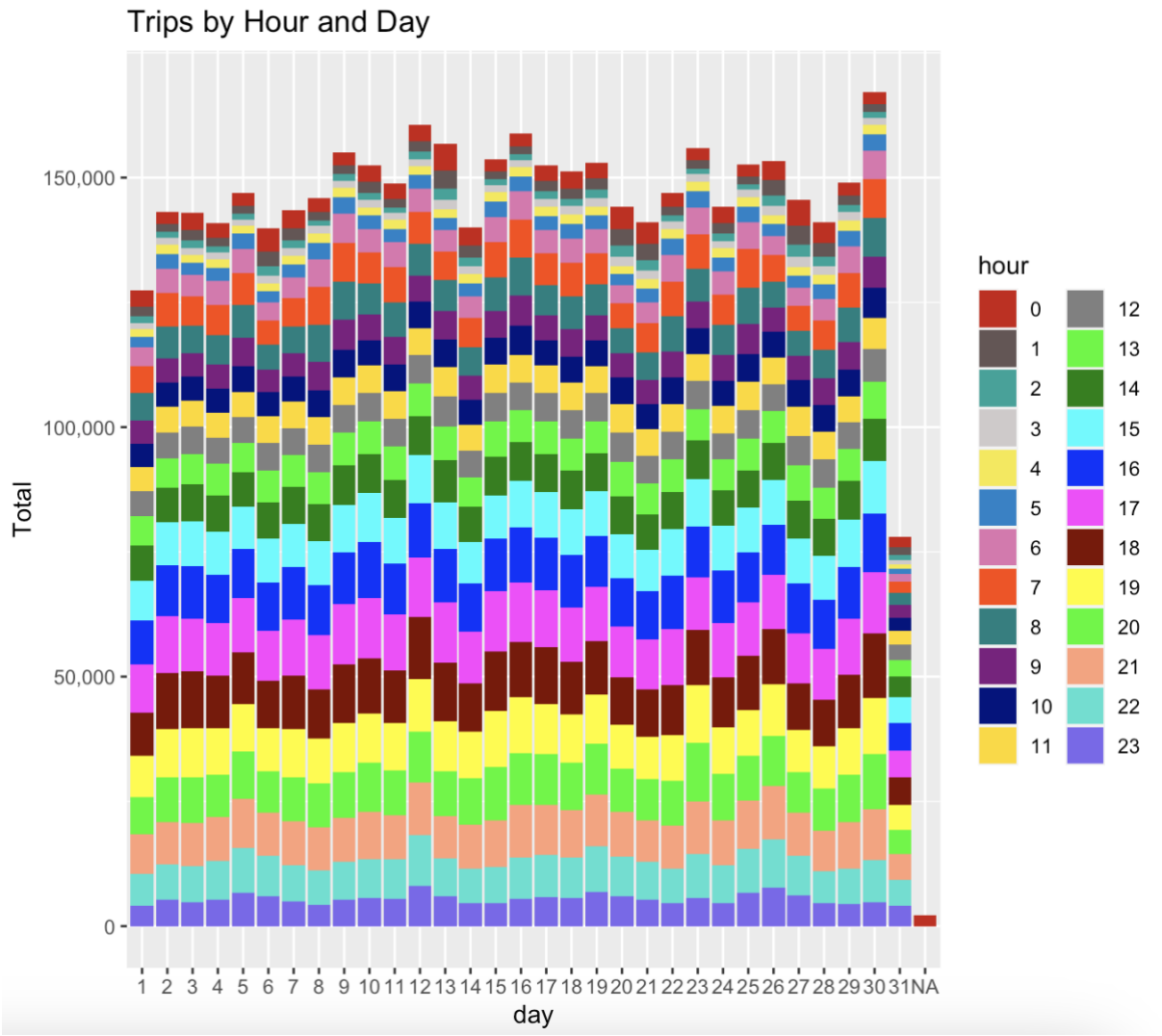
Show 10 entries Search:

	day	hour	Total
1	1	0	3177
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 745 entries

Previous 1 2 3 4 5 … 75 Next

This is the output of the data table which was made by dplyr function to aggregate the data with hour and day basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.7 Trips by DayofWeek and Month

```
month_weekday <- data_2014 %>%
  group_by(month, dayofweek) %>%
  dplyr::summarize(Total = n())
datatable(month_weekday)

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

In the next step of our Uber data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by DayofWeek and Month. We will aggregate the data by using dplyr function. In the visualizing result of the data, we understand that the number of trips are higher in the day of week is Tuesday, September.

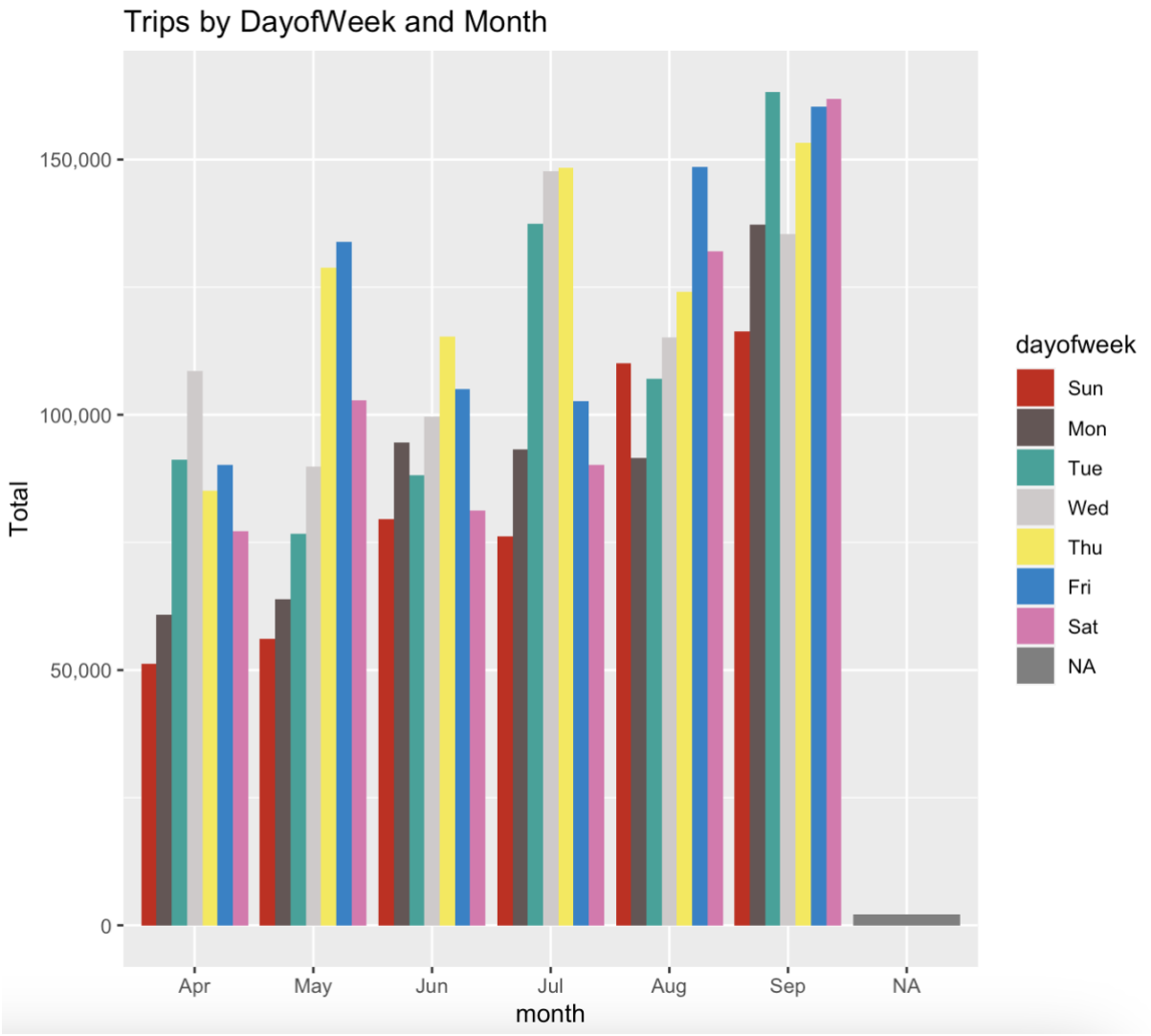
Show 10 entries Search:

	month	dayofweek	Total
1	Apr	Sun	51162
2	Apr	Mon	60848
3	Apr	Tue	91172
4	Apr	Wed	108604
5	Apr	Thu	85051
6	Apr	Fri	90272
7	Apr	Sat	77155
8	May	Sun	56084
9	May	Mon	63826
10	May	Tue	76647

Showing 1 to 10 of 43 entries

Previous 1 2 3 4 5 Next

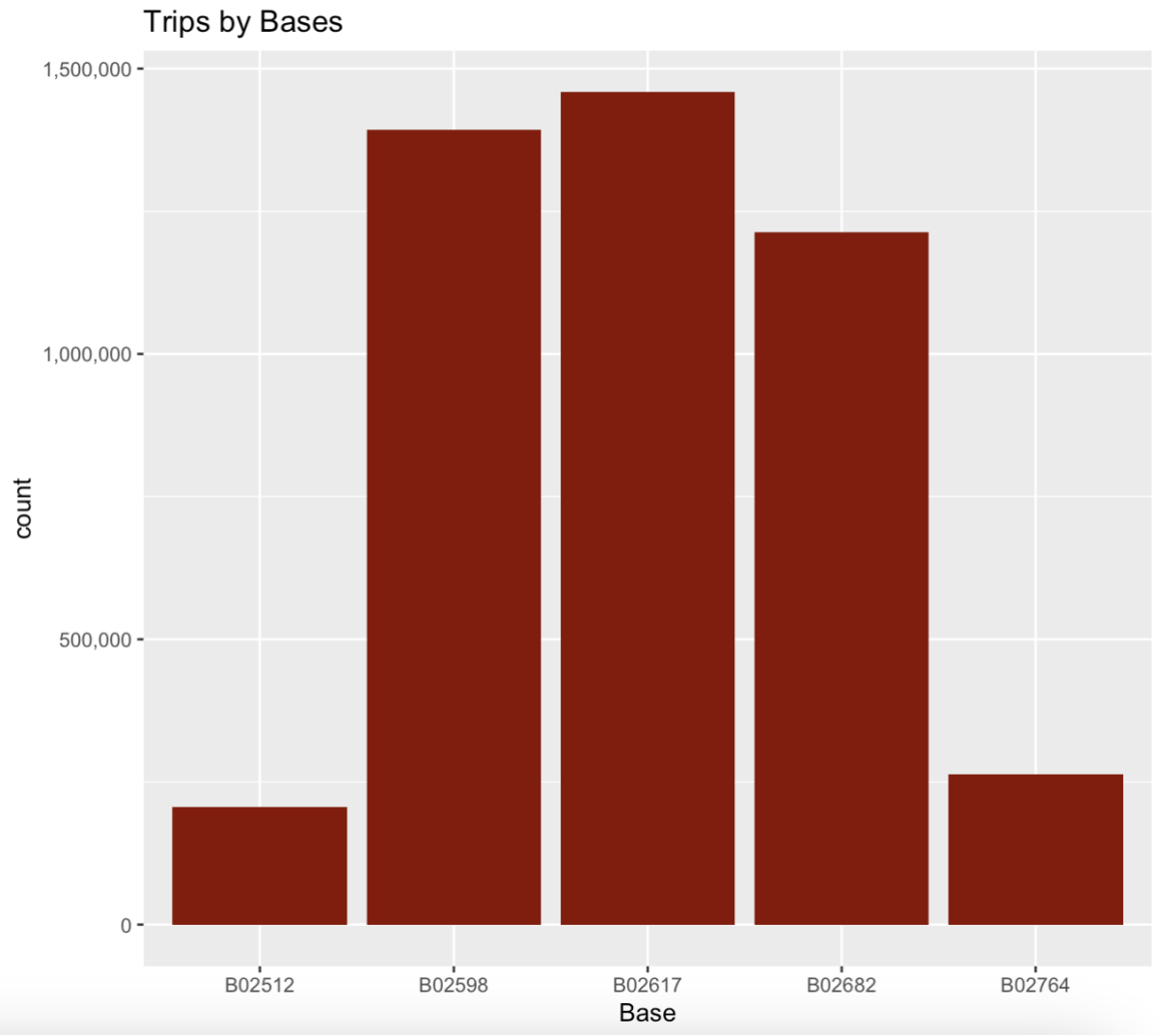
This is the output of the data table which was made by dplyr function to aggregate the data with dayofweek and month basis. Then we will plotting this table in to a bar chart to more visualize our dataset.



5.8 Trips by Bases

```
ggplot(data_2014, aes(Base)) +
  geom_bar(fill = "darkred") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases")
```

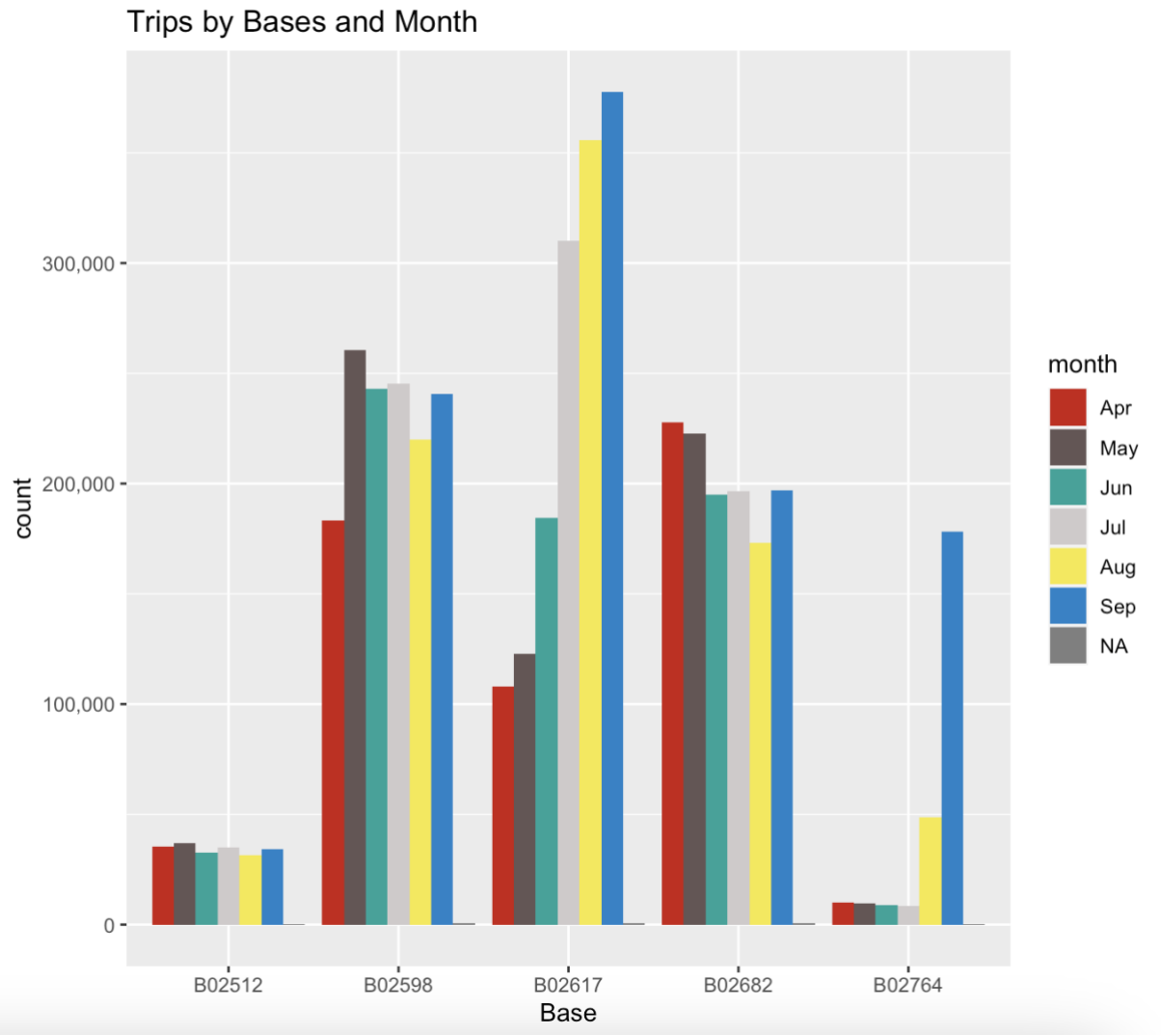
In the next step of our Uber data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by Bases. There are five bases in all out of which, we observe that B02617 had the highest number of trips and B02512 had the lowest number of trips. Furthermore, this base had the highest number of trips in the month B02617. Thursday observed highest number of trips in the three bases - B02598, B02617, B02682.



5.9 Trips by Bases and Month

```
ggplot(data_2014, aes(Base, fill = month)) +
  geom_bar(position = "dodge") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases and Month") +
  scale_fill_manual(values = colors)
```

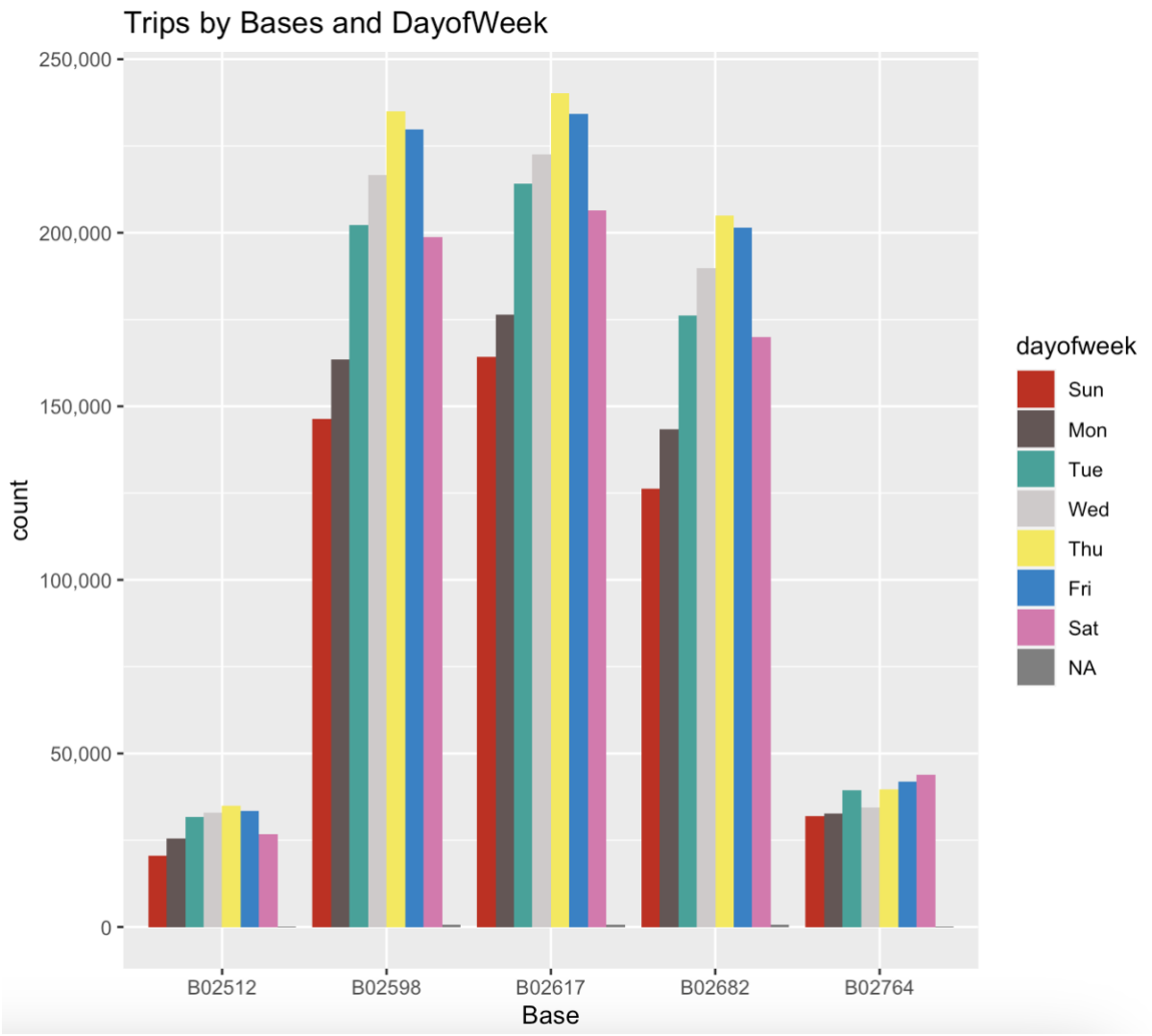
In the next step of our Uber data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by Bases and Month. There are five bases in all out of which, we observe that B02617 had the highest number of trips in the month of September and B02764 had the lowest number of trips in the month of July. Furthermore, this base had the highest number of trips in the month B02617.



5.10 Trips by Bases and Day of Week

```
ggplot(data_2014, aes(Base, fill = dayofweek)) +
  geom_bar(position = "dodge") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases and DayofWeek") +
  scale_fill_manual(values = colors)
```

In the next step of our Uber data analysis project, we will use ggplot2 function to plotting the number of trips that the passenger had made by Bases and DayofWeek. There are five bases in all out of which, we observe that B02617 had the highest number of trips in the day of Thursday and B02512 had the lowest number of trips in the day of Sunday. Furthermore, B02598, B02617, B02682 are almost same percentage of overall trips in a month.



5.11 Heat Map by Hour and Day

```
day_and_hour <- data_2014 %>%
  group_by(day, hour) %>%
  dplyr::summarize(Total = n())
datatable(day_and_hour)

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

In the next step of our data analysis project, we will create heatmap by using ggplot2 function. Heatmap is actually a 2-dimensional data visualization technique that represents the magnitude

of individual values within a dataset as a colour. The variation in colour may be by hue or intensity. First, we will aggregate our data by day and hour basis. Then we will visualize the data by using heatmap.

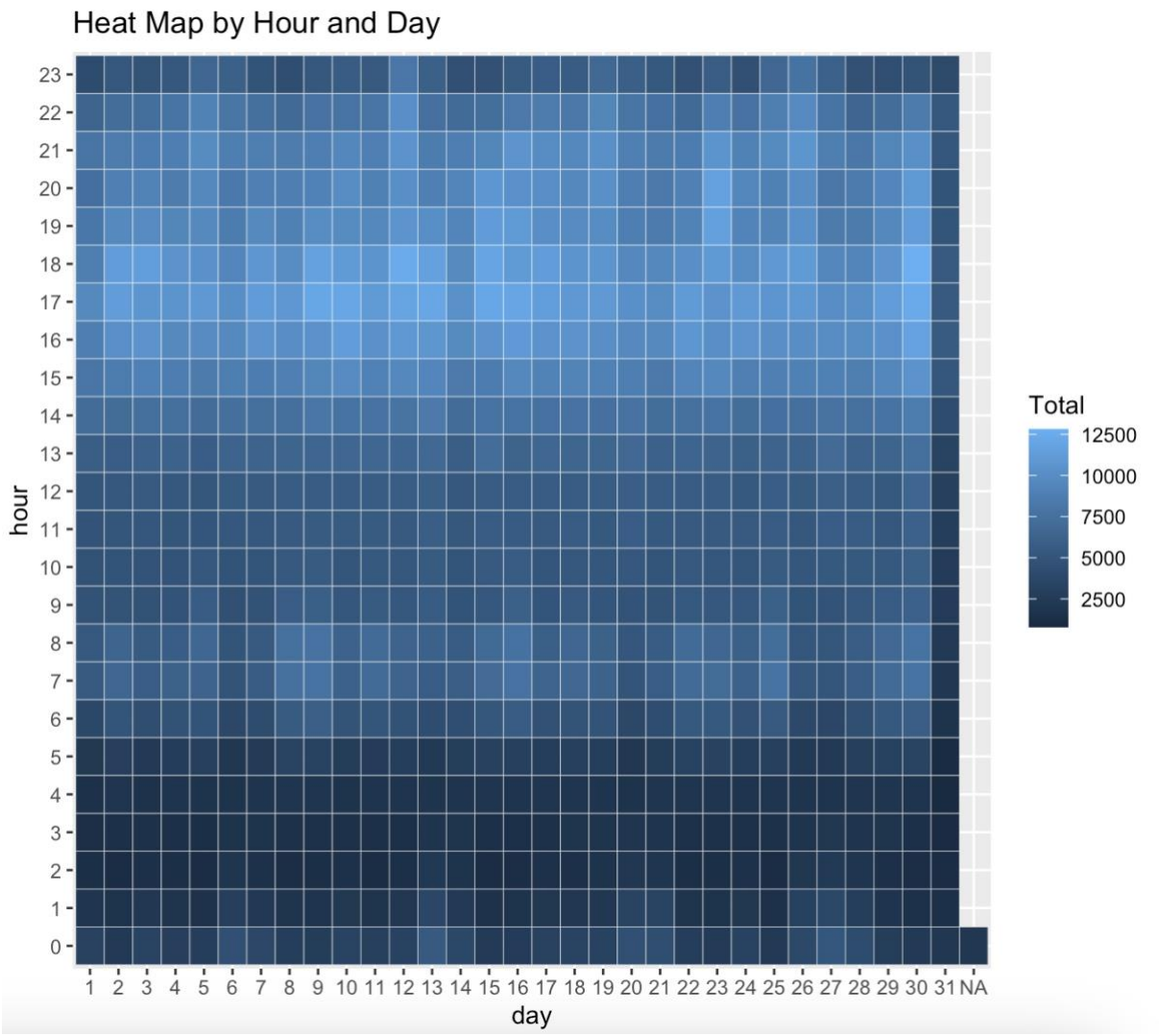
Show 10 entries Search:

	day	hour	Total
1	1	0	3177
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 745 entries

Previous 1 2 3 4 5 … 75 Next

This is the output of the data table which was made by dplyr function to aggregate the data with hour and day basis. Then we will plotting this table in to a Heatmap to more visualize our dataset.

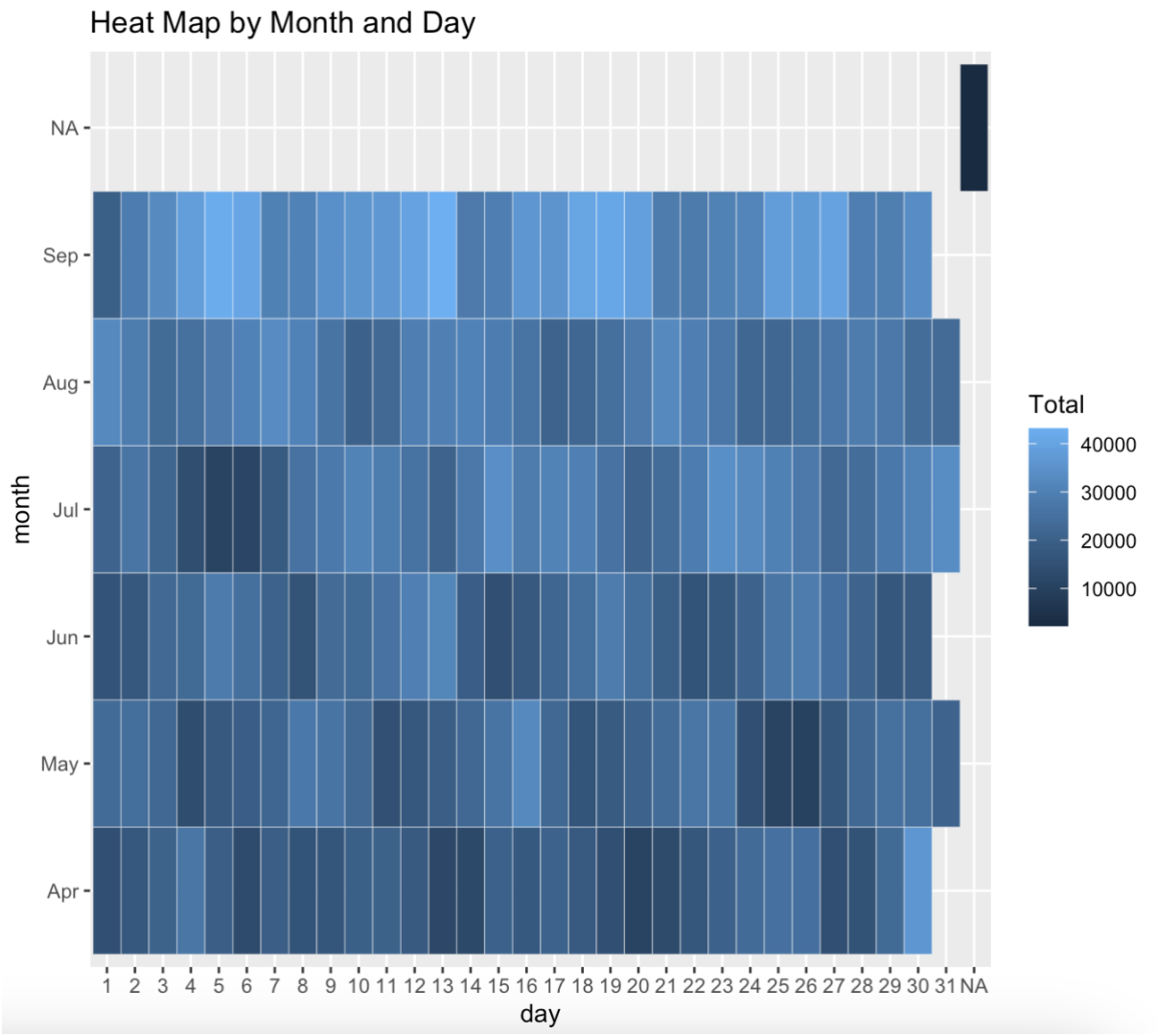


It is heatmap showing the number of Uber trips that took place in a particular city, likely in the United States, over a one-month period. Each cell shows the total number of trips that began at a specific hour and day of the week. For example, the cell at a row 14, column 5 (hour 5, day Saturday) shows that there were 12,500 trips that began at that time. The heat map shows that there is a clear pattern to when Uber trips are most frequent. There are more trips during the day, especially on weekdays, than at or on weekends, there is a peak in ridership at around 5pm on weekdays, and ridership is lowest between 3am and 6am on most days.

5.12 Heat Map by Month and Day

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

In the next step of our data analysis project, we will create heatmap by using ggplot2 function. We will aggregate our data by Month and Day basis. Then we will visualize the data by using heatmap.



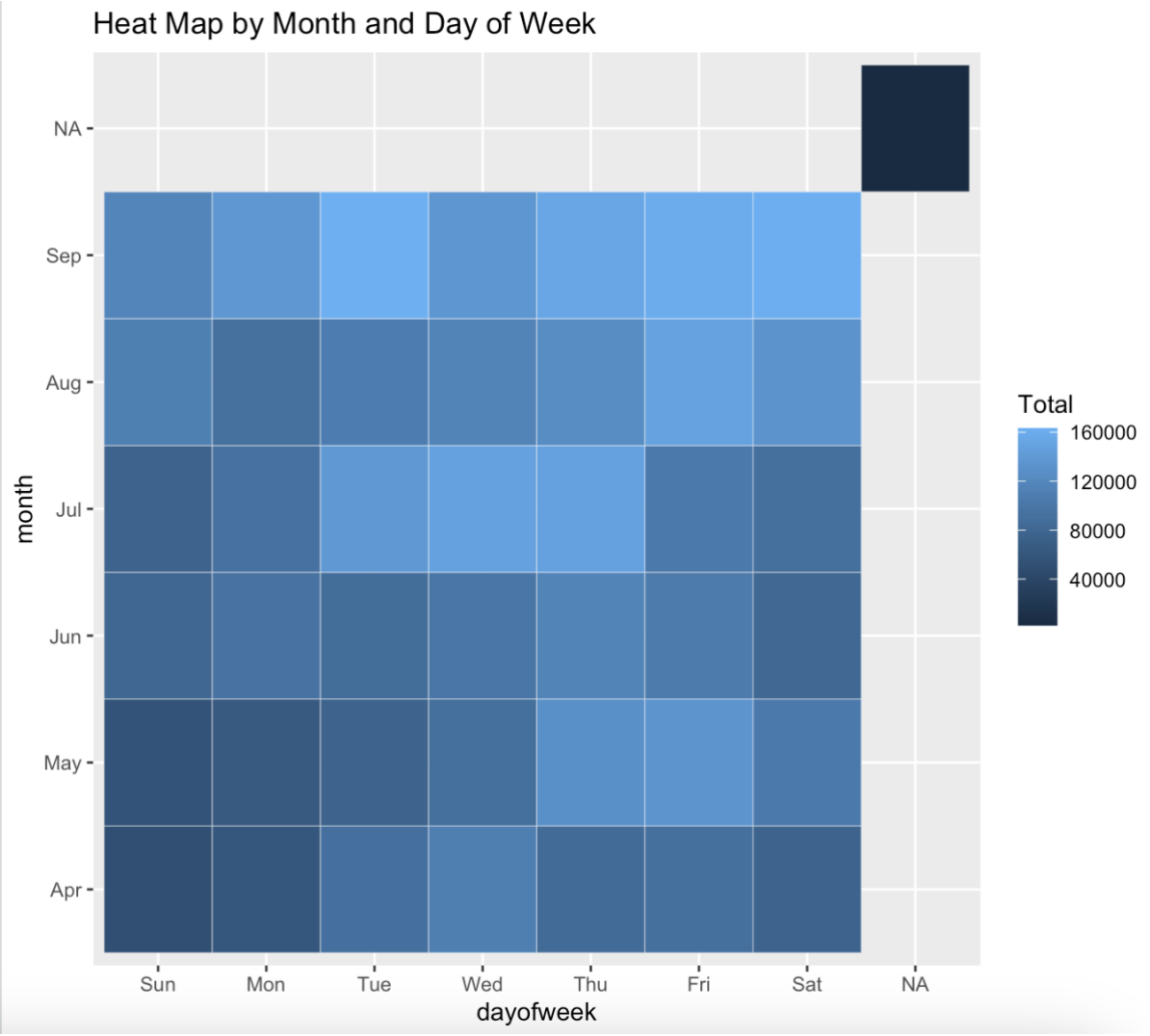
It is actually a time series graph that shows the total number of Uber trips taken each day for the month of May. The x-axis of the graph represents the day of the month, while the y-axis represents the total number of trips. The graph shows that there were more Uber trips taken on weekends (days 5, 6, 12, 13, 19, 20, 26, 27, and 30) than on weekdays. There is also a peak in the number of trips taken on Memorial Day, which is on day 28 of the month. It is important

to note that the data used to create this graph is likely a small sample of all Uber trips taken in the United states during the month of May.

5.13 Heat Map by Month and day of week

```
ggplot(month_weekday, aes(dayofweek, month, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Day of Week")
```

In the next step of our data analysis project, we will create heatmap by using ggplot2 function. We will aggregate our data by Month and Day of Week basis. Then we will visualize the data by using heatmap.



In this case, the month and day of week are categorical and the value (represented by colour) is quantitative. Specifically, this heat map shows what appears to be total counts, likely

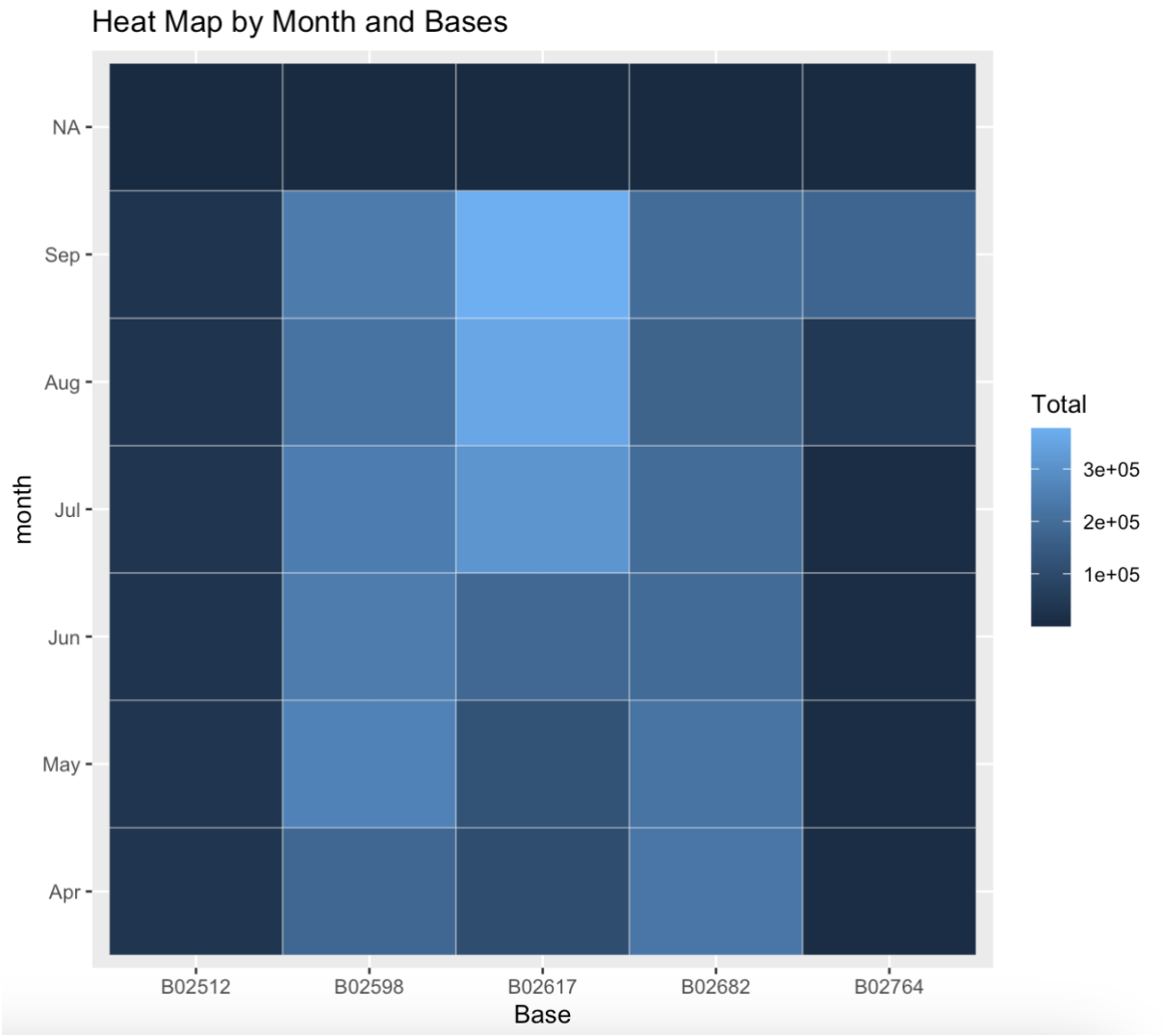
customer uses Uber, for each combination of month and day of week. For example, the blue square at Sep-Fri, Sat, and Tue indicates that there were most highest trips taken by the customers on Friday, Saturday, and Tuesday in September, while the dark blue square at Apr-Sun indicates that there were few trips taken by customer on Sunday in April. Overall, the heat map suggests that there is a seasonal pattern to the data, with more visits in some months than others. There also appears to be a weekly pattern, with some days of the week having more visits than others.

5.14 Heat map by Month and Bases

```
month_base <-  data_2014 %>%
  group_by(Base, month) %>%
  dplyr::summarize(Total = n())

ggplot(month_base, aes(Base, month, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Bases")
```

In the next step of our data analysis project, we will create heatmap by using ggplot2 function. We will aggregate our data by Month and Base basis. Then we will visualize the data by using heatmap.



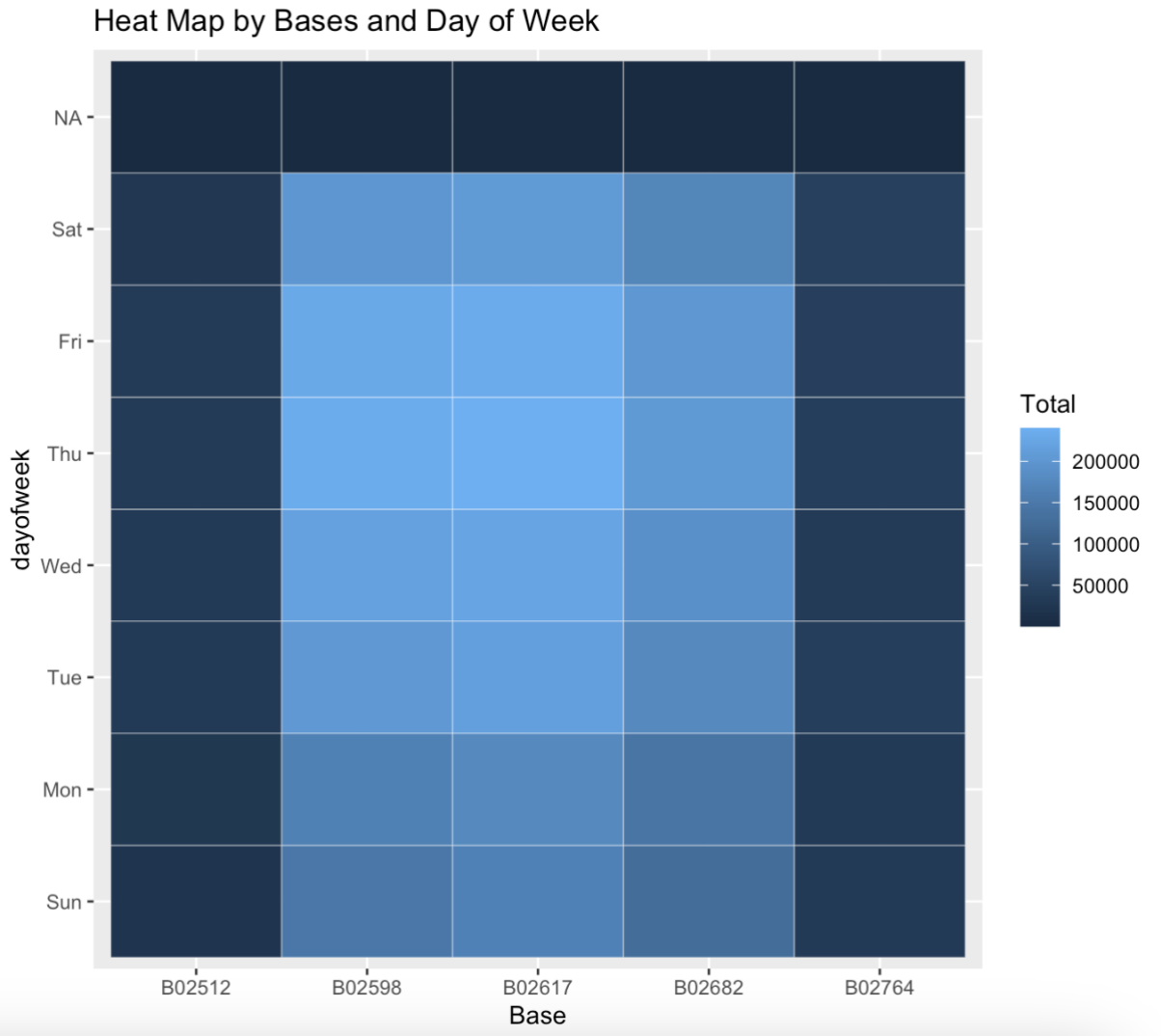
The picture shows that it appears to show the number of Uber trips that took place at different Uber bases in each month. The month are listed down the left-hand side of the graph, from April to September. Across the top of the graph are the base IDs, B02512, B02598, B02617, B02682, and B02764. The numbers in the body of the graph show the number of trips. For instance, in July there were 200,000 trips (2e_05) from base B02617. The deepest colour on the heat map appears to be dark red, and this corresponds to the highest number of trips in a month from a particular base. Lighter shades of red correspond to fewer trips. It's important to note that the legend for the heat map isn't shown, so it's difficult to say exactly how many trips each colour represents. Overall, the heat map suggests that there is a lot of vacation in the number of trips that take place at each base from month to month. For instance, in July there were many more trips from base B02617 than from any other base. However, in September there appear to be far fewer trips from that same base.

5.15 Heat map by Bases and Day of Week

```
day0fweek_bases <-  data_2014 %>%
  group_by(Base, dayofweek) %>%
  dplyr::summarize(Total = n())

ggplot(day0fweek_bases, aes(Base, dayofweek, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Bases and Day of Week")
```

In the next step of our data analysis project, we will create heatmap by using ggplot2 function. We will aggregate our data by dayofweek and Base basis. Then we will visualize the data by using heatmap.



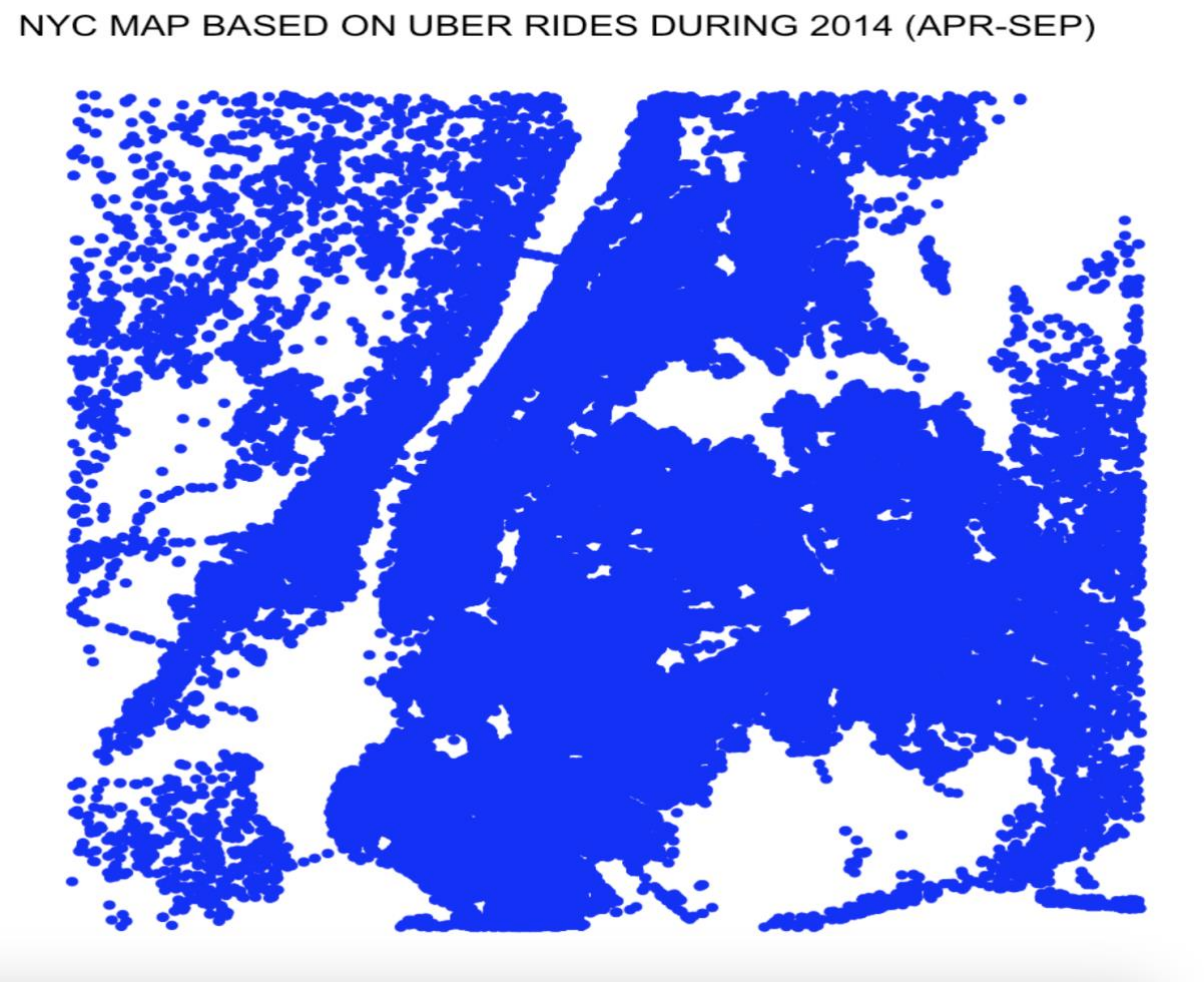
The graph shows that the number of Uber trips that started at each base on a given day of the week. The days of the week are listed across the bottom of the graph, from Sunday on the left to Saturday on the right. The bases are listed up the left side of the graph, with a code like B02512. The numbers in the body of the graph show the number of trips that base on that day of the week. The colour coding uses a blue colour for fewer trips and a yellow colour for more trips. For example, the cell that is in the third row down and the fourth column across shows that there were 150,000 trips that started at base B02617 on Wednesday. The totals on the right hand side of the chart show the total number of trips that started at each base for the entire week. For example, the bottom cell shows that there were a total of 200,000 trips that started at base B02764 for the entire week.

5.16 NYC Map based of Uber Rides During 2014 (APR-SEP)

```
min_lat <- 40.5774
max_lat <- 40.9176
min_long <- -74.15
max_long <- -73.7004

data_2014_filtered <- data_2014[complete.cases(data_2014$Lon, data_2014$Lat), ]
ggplot(data_2014, aes(x=Lon, y=Lat)) +
  geom_point(size=1, color = "blue") +
  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)")
```

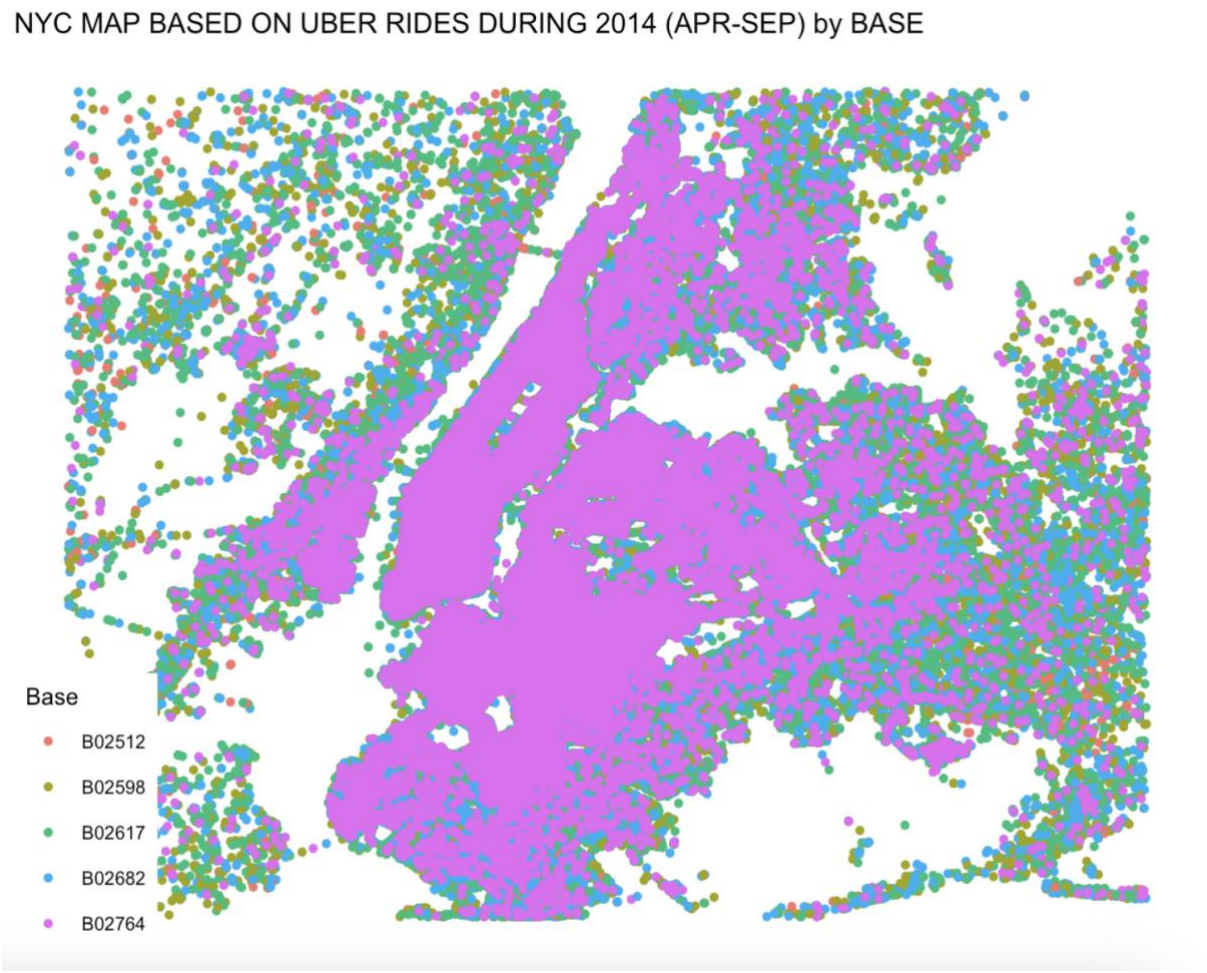
In the final section, we will visualize the rides in New York city by creating a geo-plot that will help us to visualize the rides during 2014 (Apr - Sep) and by the bases in the same period. This image shows that the map of New York City with dots representing the number of Uber rides that took place in certain areas during April to September of 2014. Each dot represents a specific location, and the size of the dot corresponds to the number of rides that started or ended at that location. This map likely shows that ride hailing was more popular in Manhattan and Brooklyn compared to the other boroughs, with some areas in Queens and the Bronx having a moderate amount of Uber usage. There is very little to no Uber usage in Staten Island according to the data visualized in this map.



5.17 NYC Map Based on Uber Rides During 2014 (APR-SEP) by Base

```
ggplot(data_2014, aes(x=Lon, y=Lat, color = Base)) +
  geom_point(size=1) +
  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) by BASE")
```

In the next step of our data analysis project or the last step of our Uber data analysis project, we will create a map by using the ggplot2, geom_point function. The map is created based on the Uber rides during 2014 (Apr - Sep) by Base. Below is the picture of the map.



This map shows that the visualization of Uber rides in New York City during a six-month period (Apr - Sep) in 2014 by Base. It doesn't show the number of rides itself, but the density of Uber rides in different parts of the city. Areas with the most Uber ride activity are coloured as dark pink. These areas are likely in Manhattan, around Central park, and Brooklyn. There is less Uber ride activity in areas coloured light yellow and ash. There areas include Staten Island, the Bronx, and parts of Queens. Overall, the graph suggests that Uber rides were more popular in central areas of New York City during this time Period. There are a few possible explanations for this picture. For example, these areas may have had better public transportation options, or they may have been more densely populated.

6.0 Conclusion

6.1 Interpret the Results

Uber ride data from April to September 2014 was analyzed, and the results show numerous important trends and patterns. Peak riding hours are always in the evening, between 5:00 and 6:00 PM, indicating a typical period of strong demand. The month with the most rides occurs in September, while the least rides occur in April. Every month on the 30th—especially in April—stands out as a day with noticeably higher ride counts, suggesting possible monthly trends. Furthermore, it seems that September's riding activity is at its peak on Tuesdays. All analysis show that Base B02617 leads in ride counts, with Thursdays being especially active for all bases. These tendencies are further supported by the heatmap analysis, which indicate greater ride frequencies on weekends and peak ridership around 5:00 PM on weekdays.

6.2 Recommendation

Uber may put these findings to use by using focused tactics. Since these are the hours of greatest traffic, assigning more drivers to work during these hours might improve customer satisfaction by cutting down on wait times. To increase demand, marketing campaigns can concentrate on advertising trips on days and months that have traditionally had lower ridership, such Sundays and April. Optimising resources based on daily and monthly riding patterns might be beneficial to base management. Offering specials and discounts during off-peak times may also help stabilise demand swings and boost ridership.

6.3 Limitations and Future Direction

Noting that this analysis is predicated on 2014 data may preclude it from reflecting current travel trends. Further research may benefit from incorporating more recent data in order to enhance the pertinence of the findings. A further limitation is that the analysis is restricted to New York City and its applicability to other cities or regions may be limited. The findings' validity may also be affected by the quality of the data; therefore, future research should strive to obtain comprehensive, high-quality datasets. Weather conditions, events, and economic indicators are a few additional variables that may impact ride demand. By investigating these

additional factors, the industry can gain a more comprehensive comprehension of ride-hailing patterns and enhance its decision-making processes.

Dataset Link

https://drive.google.com/drive/folders/1-OQJcNCoOfjR3zsiihbALLVwG4LGtruY?usp=drive_link

Reference

Awan, A. A. (2023, January 23). How to Import Data Into R: A Tutorial. https://www.datacamp.com/tutorial/r-data-import-tutorial

Kosourova, E. (2023, March 6). The Easy Way to Install a Package in R (with 8 Code Examples). Dataquest. https://www.dataquest.io/blog/install-package-r/

Bhalla, D. (n.d.). Data Exploration with R. Retrieved from https://www.listendata.com/2014/06/data-exploration-using-r.html

R Tutorial. (n.d.). Retrieved from https://www.w3schools.com/r/default.asp

Schweinberger, M. (n.d.). Introduction to Data Visualization in R. Retrieved from https://ladal.edu.au/introviz.html#:~:text=When%20creating%20a%20visualization%20with,x%2D%20and%20y%2Daxes.&text=In%20a%20next%20step%2C%20we,that%20we%20want%20to%20display.

M. (2020, April 25). Heatmap in R: Static and Interactive Visualization - Datanovia. Retrieved from https://www.datanovia.com/en/lessons/heatmap-in-r-static-and-interactive-visualization/#:~:text=A%20basic%20heatmap%20can%20be,parameters%20such%20as%20cell%20size.