Group 18 - R Bootcamp

Noel Rinke, Yanik Baumann

February 18, 2022

Contents

1 Introduction

Noel Rinke and Yanik Baumann are students at the Lucerne University of Applied Sciences and Arts studying for a Master's degree in Applied Information and Data Science. Our assignment represents a complete data analysis where we use a wide variety of R functionalities. For this analysis, we use data from Citi Bike New York.

1.1 Background Information on New York City's Bike-Sharing Scheme

New York City's bike-sharing scheme *Citi Bike* began operating in 2013 [1]. With 19'000 bikes and more than 1'000 stations (as of 2022), Citi Bike is an integral part of New York City's transport network and the largest bike-sharing service in the United States [1]. The bike-sharing scheme consists of a fleet of specifically engineered bicycles that are connected to a network of stations which span with Manhattan, Brooklyn, Queens and the Bronx four of New York City's five boroughs - besides Jersey City and Hoboken.

The popularity and rise of bike sharing services stems from the benefits such services provide: While cars contribute to traffic congestion, greenhouse gas emissions and global warming, bicycles offer an environmentally friendlier, healthier, more economical and less space-consuming transportation alternative.

1.2 Aim of the Analysis

This analysis will show when the most trips are made, whether rain has an influence on the duration of the trips and where the Citibikes are used most.

1.3 Datasets used

In order to obtain the intended insights, the analysis required two different data sets: Citi Bike ride-sharing data and weather data.

The data with records on the bicycle rides was obtained from Citi Bike [2] which provides all data sets since the service began operating in 2013. To represent each season, four different data sets were downloaded for January, April, July, and December, and an equal sample size was drawn from each month in the data preparation section. In addition, it is important to note that the COVID-19 pandemic was thought to be a potential confounding factor, so we specifically chose to use data from 2019, which predated the pandemic.

The second data set used stems from the National Oceanic and Atmospheric Administration [3] which is part of the U.S. Department of Commerce. The data set includes weather data measured at the weather station with the number GHCND:USW00094728 which corresponds to New York City's Central Park. Important to note is that the weather data provided does not include measurements per hour, but per day.

2 Installation of Packages

```
require(knitr)         # Engine for dynamic report generation
require(dplyr)         # For data manipulation
require(tidyr)         # Provides various important functions that can be used for data cleaning
require(broom)         # For summarizing statistical model objects in tidy tibbles
require(lubridate)     # For working with date-times and time-spans
require(skimr)         # For summary statistic
require(ggplot2)       # tidyverse data visualization package
require(ggeasy)        # Series of aliases to commonly used but diff. to remember ggplot2 seq.
require(ggthemes)      # Extra themes, geoms, and scales for 'ggplot2'
require(scales)        # Internal scaling infrastructure to ggplot2 and its functions
require(maps)          # For computing the areas of regions in a projected map
require(ggmap)         # To visualize spatial data & models on top of static maps
require(RColorBrewer)  # Color palette
require(gridExtra)     # Provides a number of user-level functions to work with "grid" graphics
require(data.table)    # Widely used for fast aggregation of large datasets
require(NCmisc)        # Used to get list of used functions
```

3 Data Preparation

This chapter describes the import and the preparation of the data sets.

3.1 Data Import

The ride-sharing data provided by the company Lyft includes many millions of observations since 2013. We decided to reduce the size of the data used to a minimum in order to ensure high performance in the analysis. The months of January, April, July and October for the rides of the year 2019 as well as the weather data for the whole year 2019 are imported below.

```
citibike_jan19 <- read.csv(file = file.path('C:\\Users\\noelr\\switchdrive',
'\\SharedOstSchweiz\\R Bootcamp\\Data\\201901-citibike-tripdata.csv'))

citibike_apr19 <- read.csv(file = file.path('C:\\Users\\noelr\\switchdrive',
'\\SharedOstSchweiz\\R Bootcamp\\Data\\201904-citibike-tripdata.csv'))

citibike_jul19 <- read.csv(file = file.path('C:\\Users\\noelr\\switchdrive',
'\\SharedOstSchweiz\\R Bootcamp\\Data\\201907-citibike-tripdata.csv'))

citibike_oct19 <- read.csv(file = file.path('C:\\Users\\noelr\\switchdrive',
'\\SharedOstSchweiz\\R Bootcamp\\Data\\201910-citibike-tripdata.csv'))
```

```
nyc_weather_2019 <- read.csv(file = file.path('C:\\Users\\noelr\\switchdrive',
'\\SharedOstSchweiz\\R Bootcamp\\Data\\New York Weather 2019.csv'))
```

3.2 Taking Samples

As a further step to narrow down the amount of data to a minimum, we extract a sample of 4000 observations for the analysis with 1000 observations per selected month.

```
citibike_jan19_sample <- sample_n(citibike_jan19, size=1000)
citibike_apr19_sample <- sample_n(citibike_apr19, size=1000)
citibike_jul19_sample <- sample_n(citibike_jul19, size=1000)
citibike_oct19_sample <- sample_n(citibike_oct19, size=1000)
```

3.3 Merge of the Datasets and Creation of derived Variables

The following chunks of code show how the data sets were joined together, how additionally required variables were created and the data set cleaned by removing irrelevant variables.

Table 1: Overview of created and removed variables

Created additional variables	**Removed variables**
season, *DATE*, *meantemp*, *raingrouped*, *Snow_yesno*, *weekdays*, *daytime*, *tripdur_cat*	*TSUN*, *TAVG*, *AWND*, *STATION*, *NAME*

3.3.1 Creating the Variable Season

```
citibike_jan19_sample <- cbind(citibike_jan19_sample, season='winter')
citibike_apr19_sample <- cbind(citibike_apr19_sample, season='spring')
citibike_jul19_sample <- cbind(citibike_jul19_sample, season='summer')
citibike_oct19_sample <- cbind(citibike_oct19_sample, season='autumn')
```

3.3.2 Merging the Citibike Datasets

```
citibike_merged_2019 <- rbind(citibike_jan19_sample, citibike_apr19_sample,
                              citibike_jul19_sample, citibike_oct19_sample)
```

3.3.3 Adding a seperate Date-Column

```
citibike_merged_2019$DATE <- substr(citibike_merged_2019$starttime, 1, 10)
```

3.3.4 Cleaning the Weather Dataset

Variables that were considered irrelevant for the analysis were deleted.

```
nyc_weather_cleaned <- subset(nyc_weather_2019, select = -c(TSUN, TAVG, AWND, STATION, NAME))
```

3.3.5 Joining the Datasets

```
citi_cb <- left_join(citibike_merged_2019, nyc_weather_cleaned, by='DATE')
```

3.3.6 Calculate Average Temperature

```
citi_cb$meantemp <- (citi_cb$TMAX + citi_cb$TMIN) / 2
```

3.3.7 Grouping Rain Intensity

```
citi_cb$raingrouped <- ifelse(citi_cb$PRCP == 0, 'no rain',
                       ifelse(citi_cb$PRCP > 0   & citi_cb$PRCP <=2.0, 'weak',
                       ifelse(citi_cb$PRCP >2.0  & citi_cb$PRCP <=10.0, 'moderate',
                       ifelse(citi_cb$PRCP >10.0 & citi_cb$PRCP <=30.0, 'heavy', 'intense'))))

# To make sure that the categories are in the intended order
citi_cb$raingrouped <- factor(citi_cb$raingrouped, levels = c("no rain", "weak", "moderate",
 ↪  "heavy", "intense"))
```

3.3.8 Creating the Variable Snow Yes/No

```
citi_cb$Snow_yesno <- ifelse(citi_cb$SNOW > 0, 'yes', 'no')
```

3.3.9 Getting the weekdays

```
# Weekdays in english
Sys.setlocale("LC_TIME", "English")
```

```
## [1] "English_United States.1252"
```

```
# Define Weekday
citi_cb$weekday <- weekdays(as.Date(citi_cb$DATE))

# Setting correct factor order
citi_cb$weekday <- factor(citi_cb$weekday,levels = c("Monday","Tuesday","Wednesday",
                                                     "Thursday","Friday","Saturday","Sunday"))
```

3.3.10 Getting Time and Daytimes

```
citi_cb$onlytime <- as.POSIXct(substr(citi_cb$starttime, 12, 19), format="%H:%M:%S")

# Only Hours and Minutes
citi_cb$time     <- strftime(citi_cb$onlytime,format="%H:%M")

# Only Hour
citi_cb$hour     <- as.numeric(substr(citi_cb$time,1,2))

# Set Daytime
citi_cb$daytime  <- ifelse(citi_cb$onlytime <= as.POSIXct("06:00:00",format="%H:%M:%S"),
 ↪  'night',
                    ifelse(citi_cb$onlytime > as.POSIXct("06:00:00",format="%H:%M:%S") &
                    citi_cb$onlytime <= as.POSIXct("12:00:00",format="%H:%M:%S"), 'morning',
                    ifelse(citi_cb$onlytime > as.POSIXct("12:00:00",format="%H:%M:%S") &
                    citi_cb$onlytime <= as.POSIXct("18:00:00",format="%H:%M:%S"), 'afternoon',
                    ifelse(citi_cb$onlytime > as.POSIXct("18:00:00",format="%H:%M:%S") &
                    citi_cb$onlytime <= as.POSIXct("24:00:00",format="%H:%M:%S"), 'evening',
 ↪  'night'))))

# Setting correct factor order
citi_cb$daytime <- factor(citi_cb$daytime,levels = c("morning","afternoon","evening","night"))
```

3.3.11 Creating Trip Duration Categories

```
citi_cb$tripdur_cat <- ifelse(citi_cb$tripduration <=300, '0 to 5 min',
                       ifelse(citi_cb$tripduration >300 &
                              citi_cb$tripduration <=900, '5 to 15 min',
                       ifelse(citi_cb$tripduration >900 &
                              citi_cb$tripduration <=1800, '15 to 30 min',
                       ifelse(citi_cb$tripduration >1800 &
                              citi_cb$tripduration <=3600, '30 to 60 min',
                       ifelse(citi_cb$tripduration >3600 &
                              citi_cb$tripduration <=14400, '1 to 4 h',
                       ifelse(citi_cb$tripduration >14400 &
                              citi_cb$tripduration <=43200, '4 to 12 h', 'over 12 h'))))))

# To make sure that the categories are in ascending order
citi_cb$tripdur_cat <- factor(citi_cb$tripdur_cat, levels = c("0 to 5 min", "5 to 15 min", "15
 ↪  to 30 min", "30 to 60 min", "1 to 4 h", "4 to 12 h", "over 12 h"))
```

3.4 Outlier Detection and Elimination

Outliers can have a large impact on the statistics derived from the dataset, which is why they need to be handled or at least acknowledged. The variable *tripduration* consists of the time (in seconds) a customer rents a bike from Citi Bike until the bike is returned to a station with an empty dock. Therefore, there can be both natural and non-natural outliers. While the non-natural outliers are due to measurement errors (e.g. system failures), natural outliers could be due to customers not returning the rented bike, customers not being able to return the rented bike until the next day, or similar. A boxplot is created to check the distribution of the variable:

```
options(scipen = 999) # Deactivate scientific notation

ggplot(citi_cb,aes(y=tripduration)) +
  geom_boxplot() +
  scale_x_discrete() +
```

```
  theme_light() +
  theme(plot.margin=unit(c(0,0,-0.01,0), "null")) + # Space between figure and caption
  labs(title = "Boxplot of Trip Duration",
       y = "Trip Duration (in seconds)") +
  ggeasy::easy_center_title() +
  coord_flip()
```

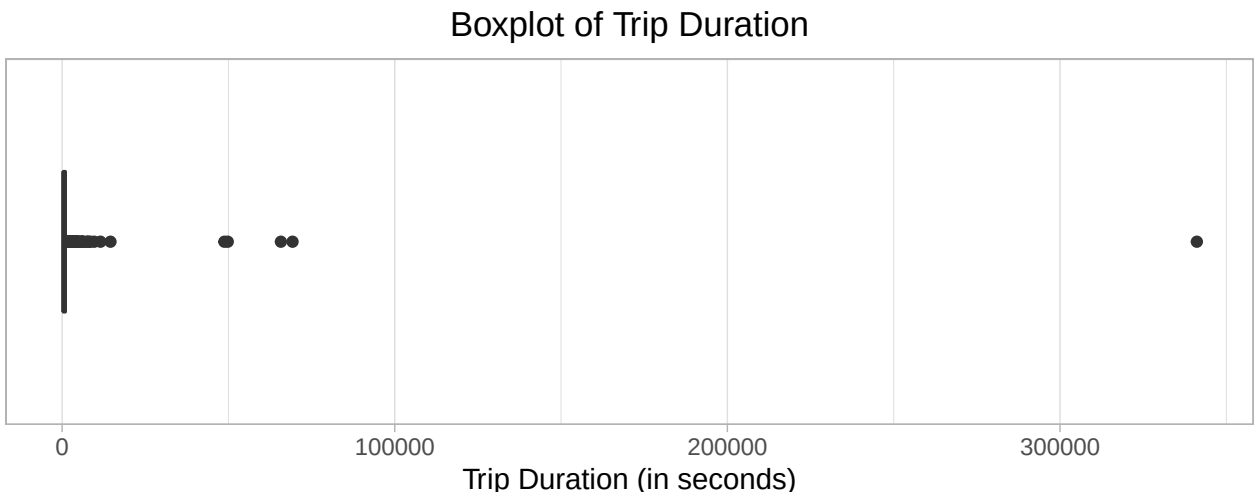

Figure 1: Boxplot to examine how the variable is distributed and if there are outliers.

The boxplot reveals that the variable *tripduration* contains outliers. There are for example values above 50'000 seconds which corresponds to approximately 15 hours. The fact that bicycles are rented for such long time durations can be considered as interesting information, although it needs to be noted that the pricing model is rather not designed for bike rentals for several hours, not to mention days. Citi Bike provides for instance an *annual membership* as well as a *day pass*. Both provide unlimited limited rides for the corresponding pass lengths. However, if the bike is not returned to a dock within 30 minutes, extra fees apply (extra $4 for each additional 15 minutes - as of 2022). Hence, the purpose of the service is to rent a bike rather multiple times per day but for not that much longer than 30 minutes and if needed by a customer for a longer distance for a few hours.

Before deciding on a threshold for removing outliers, we want to check how many bikes were rented for different time durations.

```
tripdur_cat_count <- citi_cb %>%
  group_by(tripdur_cat) %>%
  summarise(counts = n())

# Histogram trip duration categories
ggplot(tripdur_cat_count, aes(x=tripdur_cat, y=counts)) +
  geom_bar(fill = "#2980B9",
           stat = "identity") +
  geom_text(aes(label = counts), vjust = -0.3) + # Label above bar
  theme_light() +
  ggtitle("Histogram of Trip Duration Categories") +
  ggeasy::easy_center_title() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin=unit(c(0,0,-0.04,0), "null")) + # Space between figure and caption
  xlab('') + # Remove text on x-axis
  ylab("Number of rides")
```

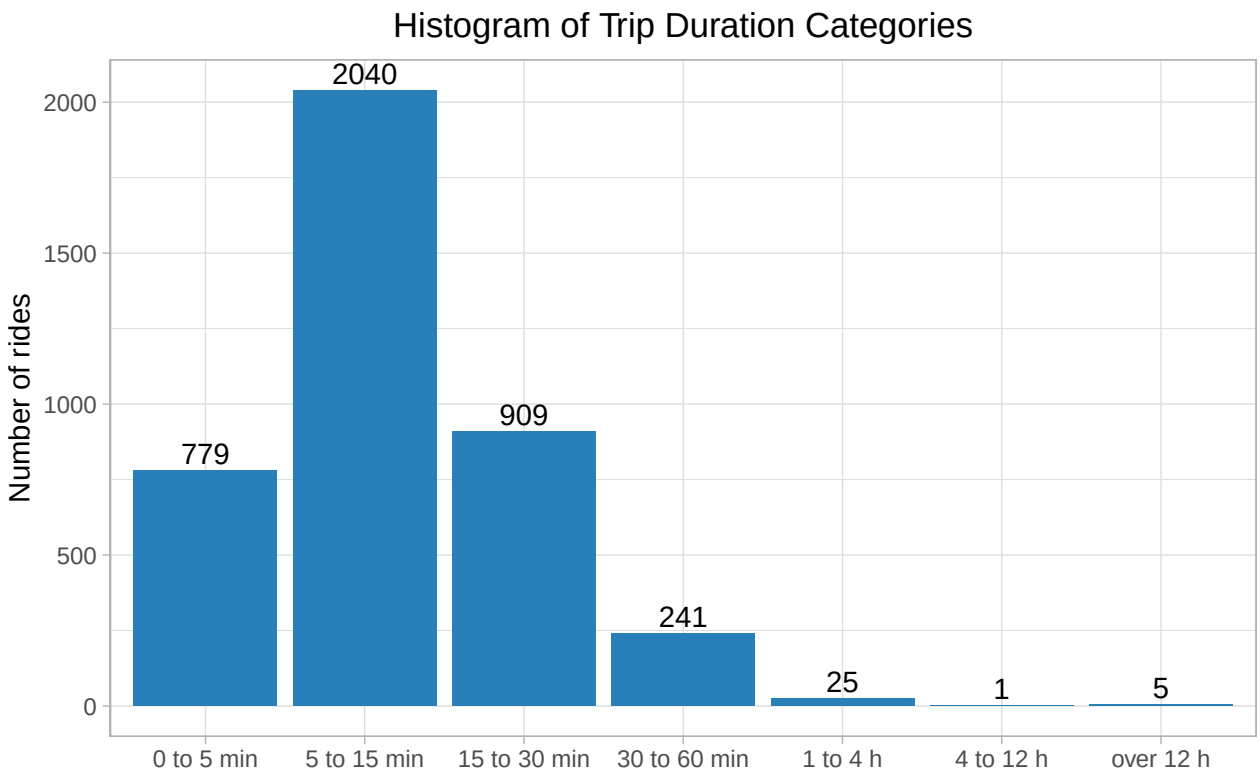

Figure 2: The histogram shows that the trip durations are mostly short-term.

With the condition that the outliers are to be examined more closely, a threshold value of 12 hours is provisionally defined. The values above this threshold are excluded from the analysis until a more detailed investigation is carried out.

```
citi_cb <- subset(citi_cb, tripduration < 43200)
```

4 Analysis

4.1 Data Inspection

The first and the last few entries of the data have been checked to make sure the data is valid and complete. Also it is helpful to get an overview of the different variables of the data set.

```
# Alternative to summary to get an overview of the data frame
str(citi_cb)
```

```
## 'data.frame':    3995 obs. of  31 variables:
##  $ tripduration           : int  371 768 132 863 2096 572 579 799 293 879 ...
##  $ starttime              : chr  "2019-01-29 18:24:45.1270" "2019-01-02 13:28:01.9870" "2019-01-17 15:5
##  $ stoptime               : chr  "2019-01-29 18:30:56.4220" "2019-01-02 13:40:50.0750" "2019-01-17 15:5
##  $ start.station.id       : chr  "3221" "520" "3131" "3170" ...
##  $ start.station.name     : chr  "47 Ave & 31 St" "W 52 St & 5 Ave" "E 68 St & 3 Ave" "W 84 St & Columb
##  $ start.station.latitude : num  40.7 40.8 40.8 40.8 40.7 ...
##  $ start.station.longitude: num  -73.9 -74 -74 -74 -74 ...
##  $ end.station.id         : chr  "3125" "3165" "3376" "3375" ...
##  $ end.station.name       : chr  "45 Rd & 11 St" "Central Park West & W 72 St" "E 65 St & 2 Ave" "3 Ave
##  $ end.station.latitude   : num  40.7 40.8 40.8 40.8 40.8 ...
##  $ end.station.longitude  : num  -73.9 -74 -74 -74 -74 ...
##  $ bikeid                 : int  26984 26908 18995 28892 17683 35655 16852 35280 29057 34815 ...
##  $ usertype               : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ birth.year             : int  1976 1969 2001 1975 1980 1985 1983 1983 1963 1991 ...
##  $ gender                 : int  1 1 1 2 1 1 1 1 1 1 ...
##  $ season                 : chr  "winter" "winter" "winter" "winter" ...
##  $ DATE                   : chr  "2019-01-29" "2019-01-02" "2019-01-17" "2019-01-10" ...
##  $ PRCP                   : num  5.8 0 0 0 0 0 0 0 0 0 ...
##  $ SNOW                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SNWD                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TMAX                   : num  6.1 4.4 0.6 1.1 3.9 3.9 -0.5 -1 0.6 -1 ...
##  $ TMIN                   : num  -3.8 1.7 -4.3 -2.1 -1 -1 -10.5 -6 -4.3 -6 ...
##  $ meantemp               : num  1.15 3.05 -1.85 -0.5 1.45 1.45 -5.5 -3.5 -1.85 -3.5 ...
##  $ raingrouped            : Factor w/ 5 levels "no rain","weak",..: 3 1 1 1 1 1 1 1 1 1 ...
##  $ Snow_yesno             : chr  "no" "no" "no" "no" ...
##  $ weekday                : Factor w/ 7 levels "Monday","Tuesday",..: 2 3 4 4 3 3 2 5 4 5 ...
##  $ onlytime               : POSIXct, format: "2022-02-18 18:24:45" "2022-02-18 13:28:01" ...
##  $ time                   : chr  "18:24" "13:28" "15:54" "08:29" ...
##  $ hour                   : num  18 13 15 8 6 13 12 17 16 11 ...
##  $ daytime                : Factor w/ 4 levels "morning","afternoon",..: 3 2 2 1 1 2 2 2 2 1 ...
##  $ tripdur_cat            : Factor w/ 7 levels "0 to 5 min","5 to 15 min",..: 2 2 1 2 4 2 2 2 1 2 ...
```

```
# Alternative to summary to get an overview of the data frame
skim(citi_cb)
```

Table 2: Data summary

Name	citi_cb
Number of rows	3995
Number of columns	31
Column type frequency:	
character	11
factor	4
numeric	15

POSIXct	1
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
starttime	0	1	24	24	0	3995	0
stoptime	0	1	24	24	0	3995	0
start.station.id	0	1	2	4	0	707	0
start.station.name	0	1	9	45	0	705	0
end.station.id	0	1	2	4	0	706	0
end.station.name	0	1	9	45	0	705	0
usertype	0	1	8	10	0	2	0
season	0	1	6	6	0	4	0
DATE	0	1	10	10	0	123	0
Snow_yesno	0	1	2	3	0	2	0
time	0	1	5	5	0	1088	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
raingrouped	0	1	FALSE	5	no : 2377, wea: 873, mod: 335, hea: 288
weekday	0	1	FALSE	7	Wed: 683, Tue: 674, Thu: 607, Mon: 591
daytime	0	1	FALSE	4	aft: 1538, mor: 1267, eve: 1075, nig: 115
tripdur_cat	0	1	FALSE	6	5 t: 2040, 15 : 909, 0 t: 779, 30 : 241

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p
tripduration	0	1	790.29	746.58	61.00	346.00	587.00	993.50	1455
start.station.latitude	0	1	40.74	0.03	40.66	40.72	40.74	40.76	40
start.station.longitude	0	1	-73.98	0.02	-74.02	-73.99	-73.98	-73.97	-73
end.station.latitude	0	1	40.74	0.03	40.66	40.72	40.74	40.76	40
end.station.longitude	0	1	-73.98	0.02	-74.03	-74.00	-73.99	-73.97	-73
bikeid	0	1	29549.28	7436.60	14536.00	25285.00	30954.00	34968.50	42041
birth.year	0	1	1979.85	12.39	1889.00	1970.00	1982.00	1990.00	200
gender	0	1	1.16	0.53	0.00	1.00	1.00	1.00	2
PRCP	0	1	3.14	8.18	0.00	0.00	0.00	1.00	4
SNOW	0	1	0.25	1.60	0.00	0.00	0.00	0.00	1
SNWD	0	1	0.28	2.87	0.00	0.00	0.00	0.00	3
TMAX	0	1	18.05	10.28	-9.90	11.10	18.30	26.70	3
TMIN	0	1	10.06	9.56	-16.60	3.30	10.60	17.20	2
meantemp	0	1	14.05	9.81	-12.70	6.65	14.70	22.80	3
hour	0	1	13.78	4.87	0.00	10.00	14.00	18.00	2

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
onlytime	0	1	2022-02-18 00:02:18	2022-02-18 23:59:07	2022-02-18 14:45:52	3867

4.2 Citi Bike Usage

```
## Histogramm Daytime
ggplot(data.frame(citi_cb$daytime), aes(x=citi_cb$daytime)) +
  geom_bar(fill='#2980B9') +
  theme(legend.position="none", axis.text.y=element_blank(), plot.title=element_text(hjust=.5))
   ↪  +
  theme_light() +
  ggtitle("Histogram with Number of Rides by Time of Day") +
  ggeasy::easy_center_title() +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Daytime") +
  ylab("Number of rides")
```

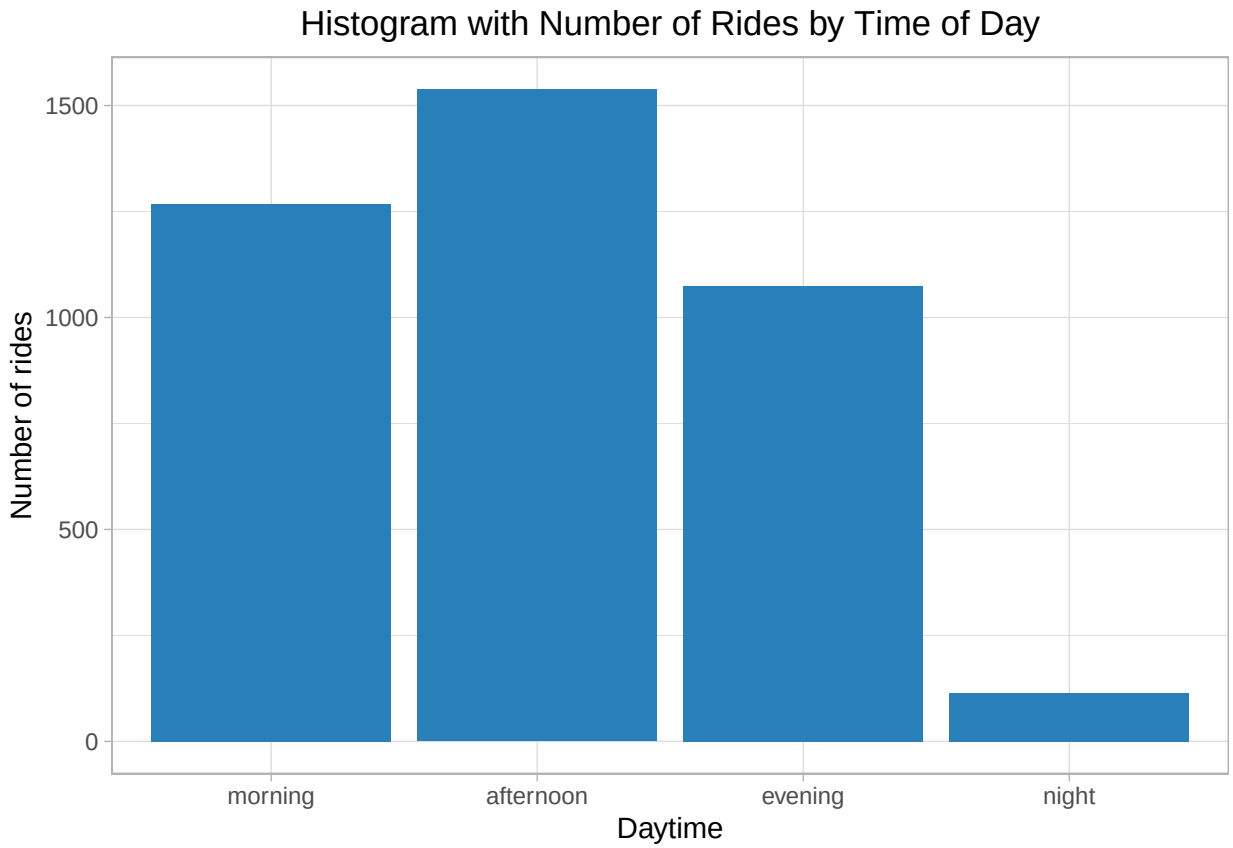

Figure 3: Most bikes were rented in the afternoon, while comparatively few bikes were rented at night.

4.2.1 Hourly Distribution of Rides

We are interested in finding out at what time of day most bikes are rented. The following graph shows the distribution for each day of the week:

```
citi_cb %>% ggplot(aes(x=hour,fill=factor(weekday))) +
  scale_fill_manual(values = c('#2980B9', '#2980B9', '#2980B9', '#2980B9', '#2980B9',
   ↪  '#2980B9', '#2980B9')) +
```

```
  geom_density(alpha=.2) +
  facet_wrap(~weekday,ncol=1) +
  theme(legend.position="none", axis.text.y=element_blank(),plot.title=element_text(hjust=.5))
   ↪  +
  ggtitle(expression(atop("Hourly Distribution of Rides")))
```

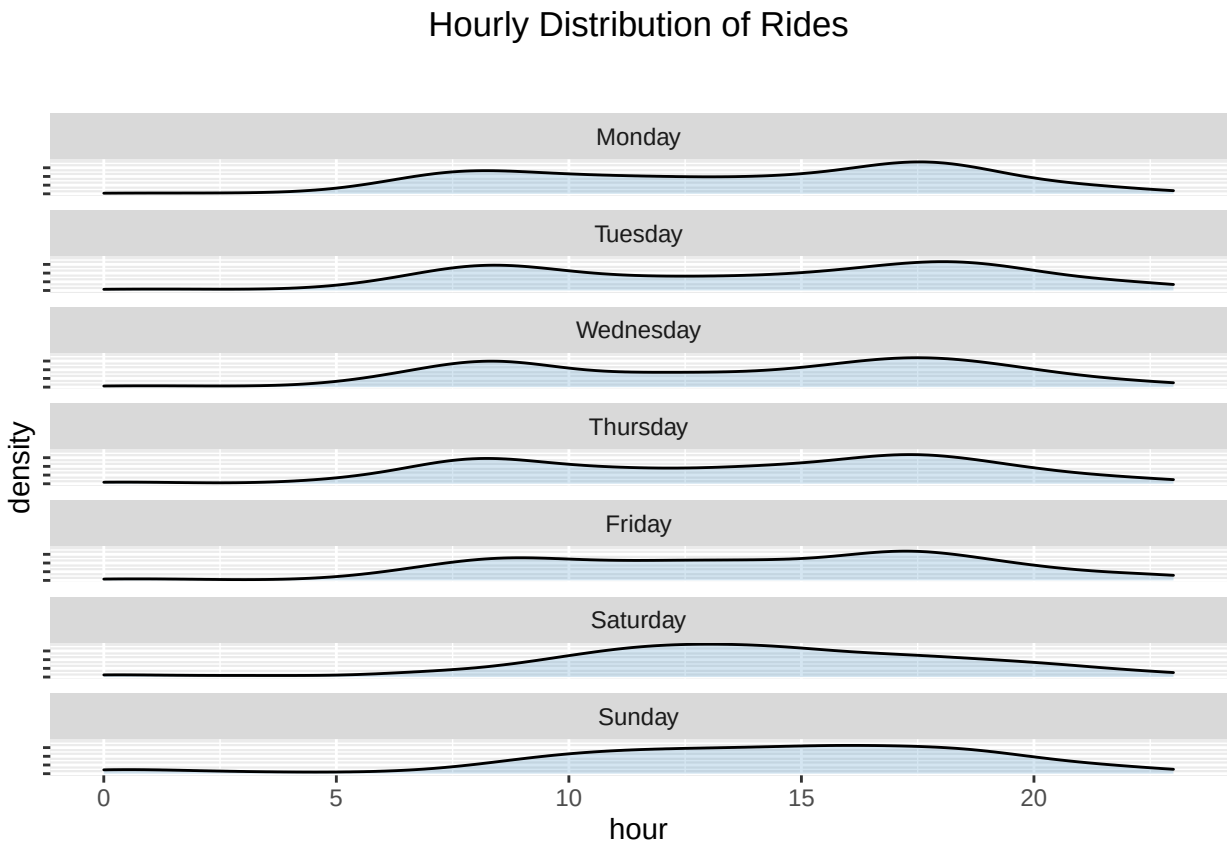

Figure 4: Density plot which shows the hourly distribution of bikes rented.

4.2.2 Most and least used Start and End Stations

An important factor for Lyft - the provider of Citi Bike - could be to track the most and least used stations. Stations that are used the most could provide the opportunity to increase the number of bikes and docking stations, and the stations where neither many customers start nor finish their ride could be relocated.

```
# Create a dataframe with a column which counts how often a start station was used
start.station.rank <- data.frame(citi_cb %>%
                                   group_by(start.station.name) %>%
                                   summarize(count.start=n()) %>%
                                   arrange(desc(count.start)))

# Create a dataframe with a column which counts how often an end station was used
end.station.rank <- data.frame(citi_cb %>%
                                 group_by(end.station.name) %>%
                                 summarize(count.end=n()) %>%
                                 arrange(desc(count.end)))

# Combine the start station and end station data frames, so that we have
# for each station the number of bikes rented and the number of bikes returned
```

```
station_usage <- left_join(x=start.station.rank, y=end.station.rank,
                           by=c("start.station.name"="end.station.name"))

# Check if there are start or end stations which were not used
na_check_start_bef <- sum(is.na(station_usage$count.start))
na_check_end_bef <- sum(is.na(station_usage$count.end))

print(paste0("There are ", na_check_start_bef, " start stations with NA values and ",
 ↪  na_check_end_bef, " end stations with NA values."))
```

```
## [1] "There are 0 start stations with NA values and 71 end stations with NA values."
```

```
# Replace all NAs with 0 (Stations which no one used should be shown as zero for the subsequent
 ↪  sum)
station_usage <- station_usage %>%
  mutate_all(~replace(., is.na(.), 0))

# Check if the NA values were successfully replaced
na_check_start_aft <- sum(is.na(station_usage$count.start))
na_check_end_aft <- sum(is.na(station_usage$count.end))

if (na_check_start_aft != 0){
  print(paste0("After replacement, there are ", na_check_start_aft, " start stations with NA
   ↪  values and ", na_check_end_aft, " end stations with NA values."))
} else{
  print("Successfully replaced NA values with the value 0.")}
```

```
## [1] "Successfully replaced NA values with the value 0."
```

```
# Create a new column which shows the combined number of bikes rented and returned
station_usage$station.count.comb <- rowSums(station_usage[, c("count.start", "count.end")])

# Order the dataframe by the combined number (from largest to smallest) to get a ranking
station_usage <- station_usage[order(-station_usage$station.count.comb),]

# Rename columns for improved readability
setnames(station_usage, old = c('start.station.name','count.start', 'count.end',
 ↪  'station.count.comb'), new = c('Station Name','Bikes rented', 'Bikes returned', 'Total'))

# Check the 10 most and least used stations
kable(head(station_usage, 10), caption="Most used stations")
```

Table 7: Most used stations

	Station Name	Bikes rented	Bikes returned	Total
1	Pershing Square North	33	30	63
4	E 17 St & Broadway	25	24	49
7	Broadway & E 22 St	24	24	48
2	Broadway & W 60 St	29	18	47
5	W 21 St & 6 Ave	25	22	47
29	W 20 St & 11 Ave	17	23	40
6	W 38 St & 8 Ave	25	14	39
10	Christopher St & Greenwich St	22	17	39
11	8 Ave & W 31 St	21	18	39
8	West St & Chambers St	24	14	38

```
kable(tail(station_usage, 10), caption="Least used stations")
```

Table 8: Least used stations

	Station Name	Bikes rented	Bikes returned	Total
667	Madison Ave & E 120 St	1	0	1
677	Nostrand Ave & Myrtle Ave	1	0	1
680	Powers St & Olive St	1	0	1
683	Putnam Ave & Wyckoff Ave	1	0	1
687	Stockholm St & Wilson Ave	1	0	1
691	W 106 St & Amsterdam Ave	1	0	1
701	Waterbury St & Stagg St	1	0	1
702	White St & Moore St	1	0	1
703	Willoughby Ave & Wyckoff Ave	1	0	1
704	Withers St & Kingsland Ave	1	0	1

```
# Create variables with the most and least used stations
top10 <- head(station_usage, 10)
least10 <- tail(station_usage, 10)

# Here we add the coordinates to our dataframe
top_10_coord <- left_join(x=top10, y=citi_cb, by=c("Station Name"="start.station.name"))

top10_coord <- top_10_coord %>%
  select('Station Name','start.station.latitude', 'start.station.longitude', 'Bikes rented',
   ↪  'Bikes returned', 'Total')

top10_coord <- distinct(top10_coord)

# Here we add the coordinates to our dataframe
least10_coord <- left_join(x=least10, y=citi_cb, by=c("Station Name"="start.station.name"))

least10_coord <- least10_coord %>%
  select('Station Name','start.station.latitude', 'start.station.longitude', 'Bikes rented',
   ↪  'Bikes returned', 'Total')

least10_coord <- distinct(least10_coord)

# Set longitudes and latitudes to define map tile for New York City
bbox <- c(left = -74.1, bottom = 40.65, right = -73.875, top = 40.825)
top10_map <- ggmap(get_stamenmap(bbox, zoom = 12, maptype = "toner-lite")) +
                geom_point(data=top10_coord,
                aes(x=start.station.longitude,
                    y=start.station.latitude),
                color = 'red', size = 2, alpha = 0.5) +
                theme_update(plot.title = element_text(hjust = 0.5)) +
                ggtitle("10 most used Stations") +
                ggeasy::easy_center_title() +
                xlab('') +                          # Remove text on x-axis
                ylab('') +                          # Remove text on y-axis
                theme(axis.line = element_blank(),    # Remove axis line
                      axis.text = element_blank(),    # Remove axis text
                      axis.ticks = element_blank(),   # Remove axis ticks
                      plot.margin=unit(
                        c(-0.1,0.001,-0.3,0), "null"))# Margins around graph
```

```
least10_map <- ggmap(get_stamenmap(bbox, zoom = 12, maptype = "toner-lite")) +
                geom_point(data=least10_coord,
                aes(x=start.station.longitude,
                    y=start.station.latitude),
                color = 'red', size = 2, alpha = 0.5) +
                theme_update(plot.title = element_text(hjust = 0.5)) +
                ggtitle("10 least used Stations") +
                xlab('') +                              # Remove text on x-axis
                ylab('') +                              # Remove text on y-axis
                theme(axis.line = element_blank(),      # Remove axis line
                      axis.text = element_blank(),      # Remove axis text
                      axis.ticks = element_blank(),     # Remove axis ticks
                      plot.margin=unit(
                        c(-0.1,0,-0.3,0.001), "null"))# Margins around graph

grid.arrange(top10_map, least10_map, ncol=2)          # Show the two maps side by side
```

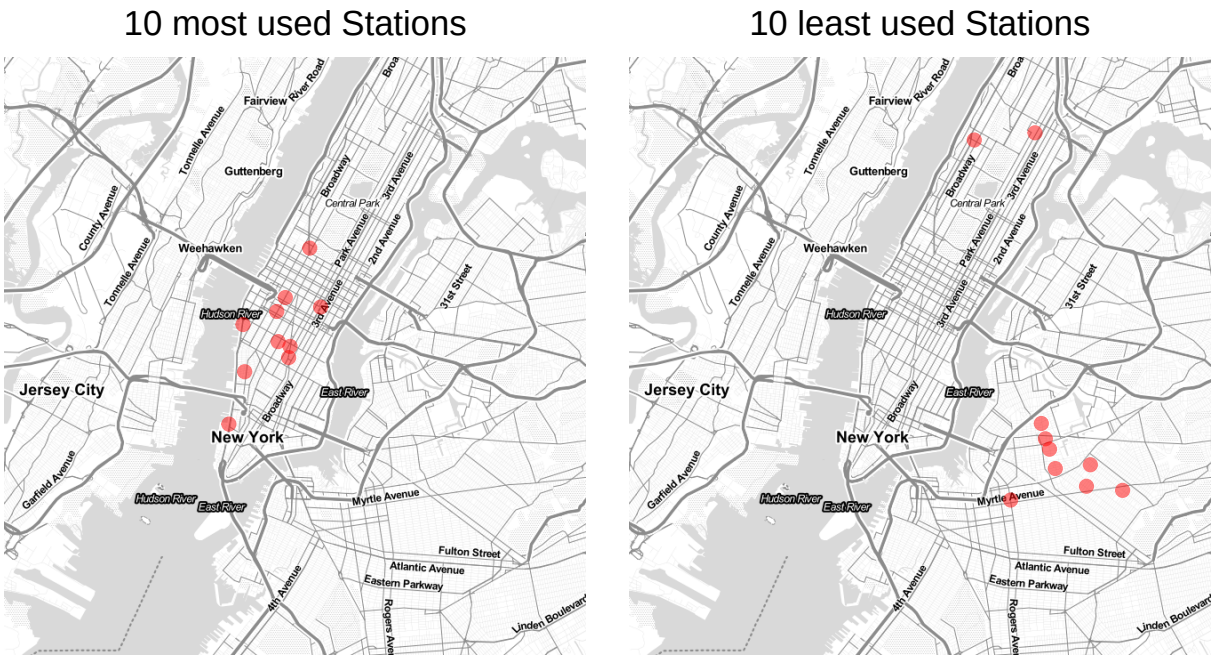

Figure 5: Maps of New York which show with red dots the most and least used stations.

The two maps show that the stations in Manhattan close to the Broadway are used the most. The stations in Brooklyn, on the other hand, are utilised very rarely. It is particularly important to mention here that this is based on a small sample. The analysis should be repeated with a larger number of data and over several periods of time before deciding on increasing the number of docks or relocating bike stations.

4.2.3 Identifiying the Popularity of different Areas in New York City

The map shows by density how strongly the concentration of started bike rentals is distributed across the city. The denser the colour, the higher the concentration of started rides.

```
# Set longitudes and latitudes to define map tile for New York City
bbox <- c(left = -74.1, bottom = 40.65, right = -73.875, top = 40.825)
```

```
# Create density map for start stations
start_station_dens <- ggmap(get_stamenmap(bbox, zoom = 12, maptype = "toner-lite")) +
  stat_density_2d(
    data = citi_cb,
    aes(x=start.station.longitude, y=start.station.latitude, fill= ..level..),
    alpha = .15, bins = 18, geom = "polygon") +
  scale_fill_gradientn(colors = brewer.pal(7, "YlGnBu")) + # library RColorbrewer
  ggtitle(label = "Start Station Density") +
  theme(plot.title = element_text(hjust = 0.5),        # Center the plot title
        plot.margin=unit(c(0,0,-0.04,0), "null")) +    # Space between figure and caption
  xlab("") +
  ylab("")

# Create density map for end stations
end_station_dens <- ggmap(get_stamenmap(bbox, zoom = 12, maptype = "toner-lite")) +
  stat_density_2d(data = citi_cb,
                  aes(x=end.station.longitude, y=end.station.latitude, fill= ..level..),
                  alpha = .15, bins = 18, geom = "polygon") +
  scale_fill_gradientn(colors = brewer.pal(7, "YlGnBu")) + # library RColorbrewer
  ggtitle(label = "End Station Density") +
  theme(plot.title = element_text(hjust = 0.5),        # Center the plot title
        plot.margin=unit(c(0,0,-0.04,0), "null")) +    # Space between figure and caption
  xlab("") +
  ylab("")
```

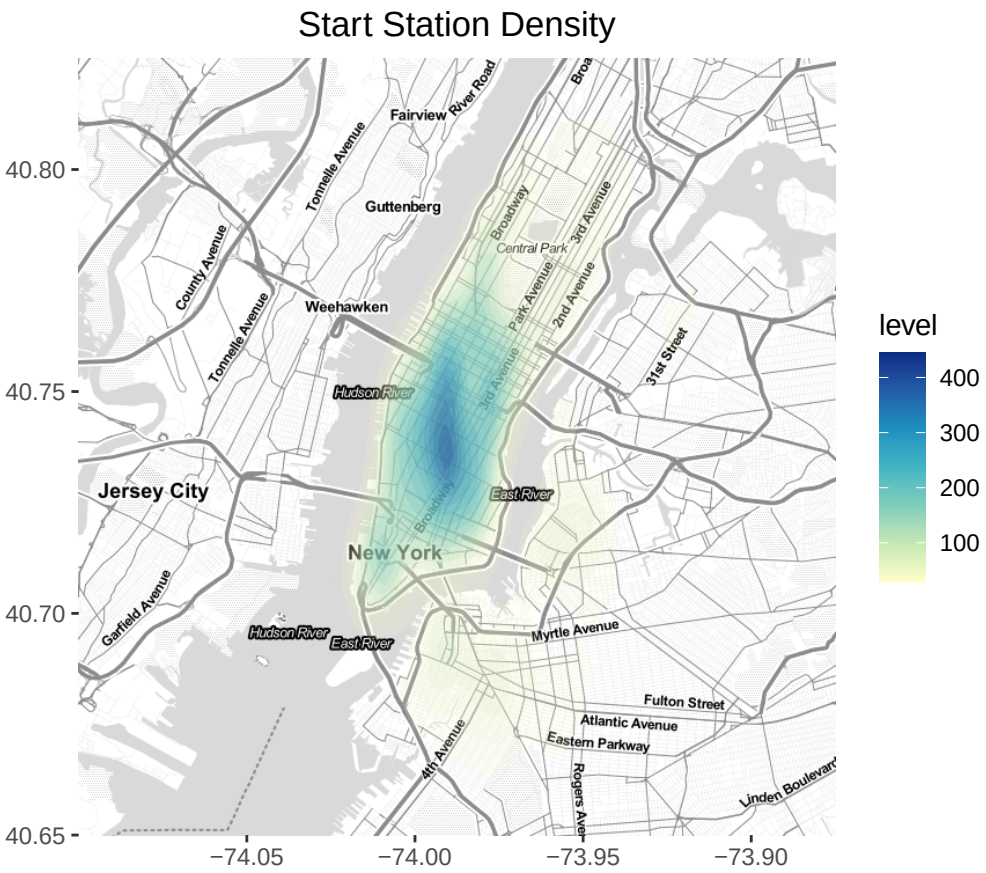

Figure 6: Map of New York which shows where the most bikes were rented.

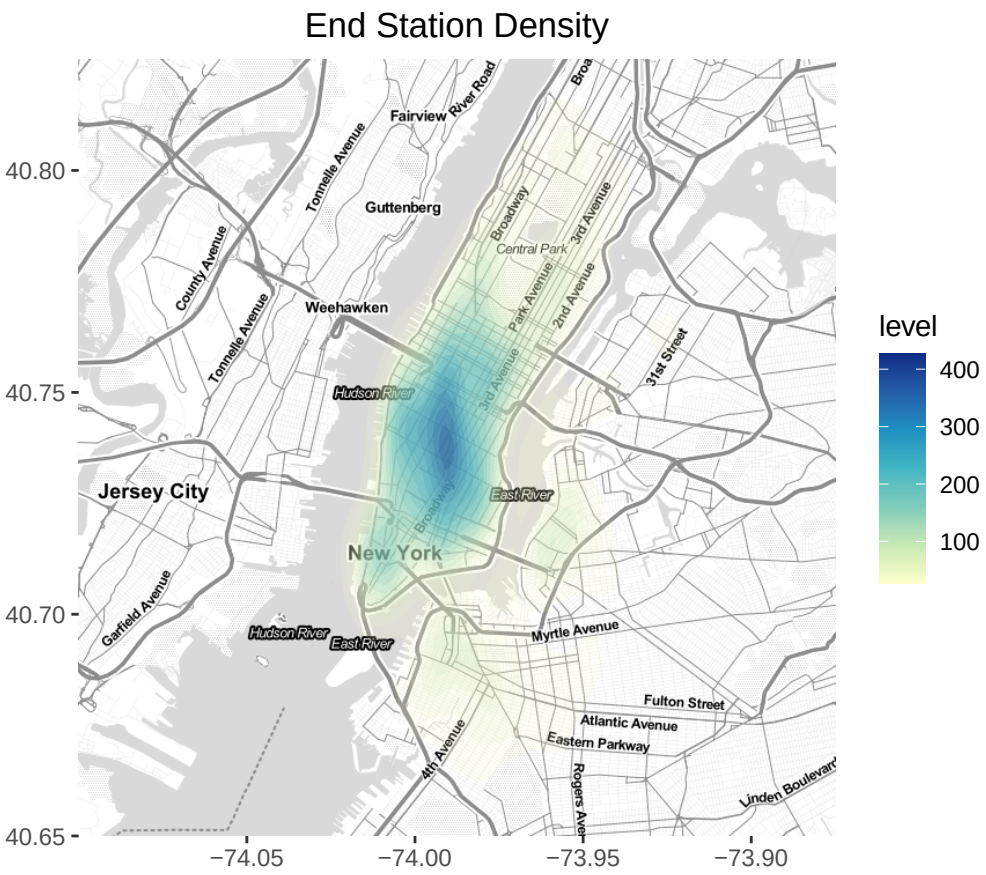

Figure 7: Map of New York which shows where the most bikes were returned.

4.3 Influence of Meteorological Factors

Furthermore, it is of interest how strongly the bike-sharing service is dependent on meteorological factors such as rain and temperatures, respectively how strongly the number of trips is influenced by these factors.

4.3.1 Influence of Temperature

```
# How strong is the effect of the average daily temperature on the trip duration?
trip_dur_filter <- citi_cb %>%
  filter(tripduration < 8000)

ggplot(trip_dur_filter, aes(x=meantemp, y=tripduration)) +
  geom_point(color='#2980B9', size=1.5) +
  geom_smooth(method=lm, color='darkred', linetype="dashed") +
  theme_light() +
  ggtitle("Effect of Average daily Temperature on Trip Duration") +
  ggeasy::easy_center_title() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin=unit(c(0,0,-0.04,0), "null")) +
  xlab("Average daily temperature (in Celsius)") +
  ylab("Trip duration (in seconds)")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

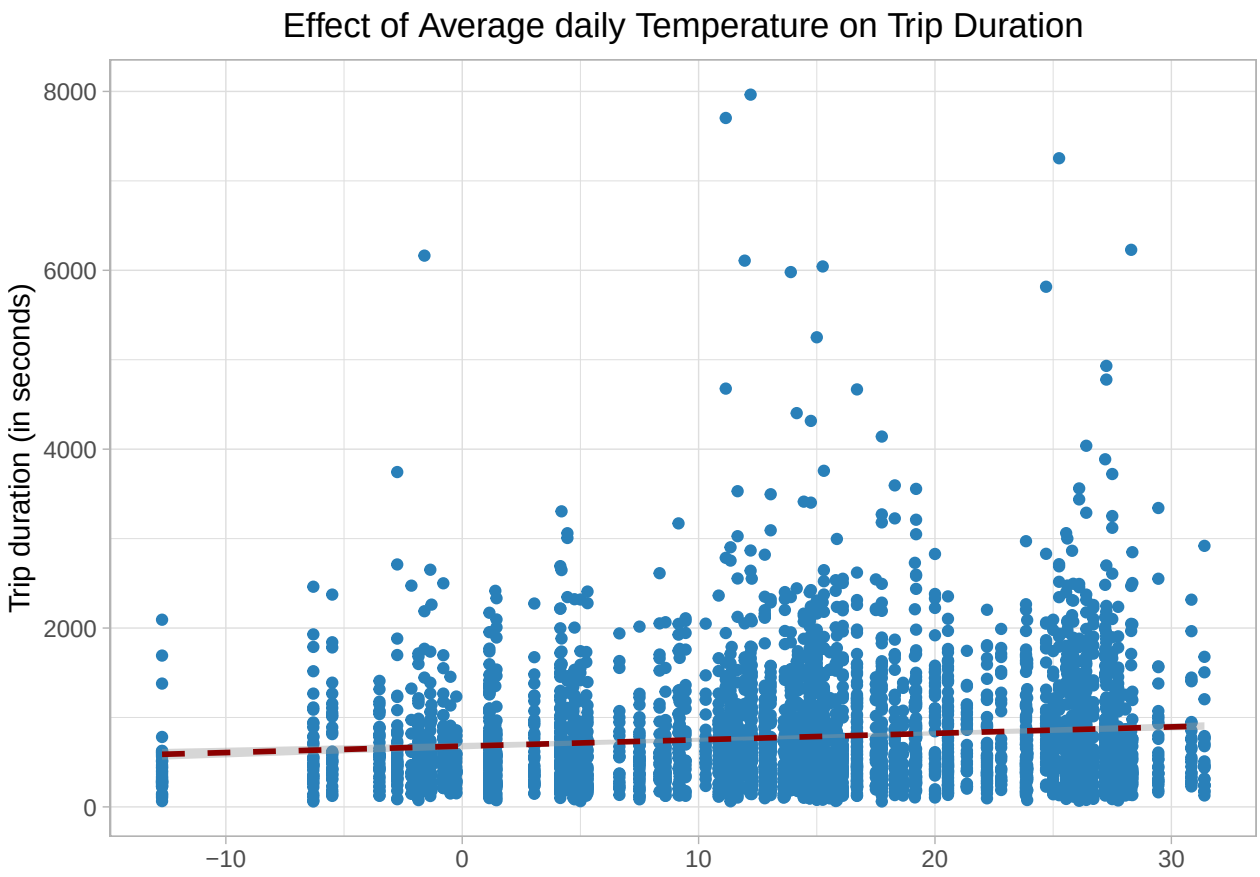

Figure 8: Scatter plot to examine the influence of the average daily temperature on the trip durations.

```
# Spearman because there are two quantitative variables
cor_tripdur_meantemp <- round(cor(x=citi_cb$meantemp, y=citi_cb$tripduration,
 ↪  method="spearman"),2)
```

[1] "Interpretation: For every degree in Celsius, the time a customer rents a bike changes by 0.1 seconds. Since we are using samples and not the whole population, the result needs to be interpreted with caution. To investigate the result further, we use inferential statistics in the form of an ANOVA test."

4.3.2 Influence of Precipitation

```
# How strong is the effect of precipitation on the number of trips?
# Calculation for each category of precipitation how often it rained
rain_cat_freq <- citi_cb %>%
  group_by(raingrouped) %>%
  summarise(counts = n())

# Plot of the result
rain_hist <- ggplot(rain_cat_freq, aes(x = raingrouped, y = counts)) +
  geom_bar(fill = "#2980B9",stat = "identity") +
  geom_text(aes(label = counts), vjust = -0.3) +   # Label above bar
  theme_light() +
  ggtitle("Histogram with Rain Categories") +
  ggeasy::easy_center_title() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin=unit(c(0,0,-0.04,0), "null")) +
```

```
  xlab("Rain Category") +
  ylab("Number of Rides")
```

With the histogram we can see how much and how often it rained during the randomly sampled trips. For the categorization of precipitation, the scale of Meteo Schweiz [4] is used:

Table 9: Precipitation scale

Term	**24h sum in mm**	**Explanation**
Weak	0.1-2	Some rain, some snowfall, isolated snowflakes, weak rain, weak snowfall, isolated rain showers, isolated snow showers, drizzle, (snow drizzle).
Moderate	2-10	Some precipitation, downpours, some snowfall, some rain showers, some snow showers some showers, some thunderstorms.
Heavy	10-30	Heavy precipitation, heavy rainfall, heavy snowfall, heavy rain showers, heavy snow showers, thunderstorms without (intensity term).
Very heavy	30-50	Partly heavy rain, heavy rainfall, heavy snowfall, heavy thunderstorms, heavy thunderstorms.
Intense	>50	Heavy rain, heavy precipitation, heavy snowfall, heavy thunderstorms, heavy thunderstorms.

```
rain_hist
```

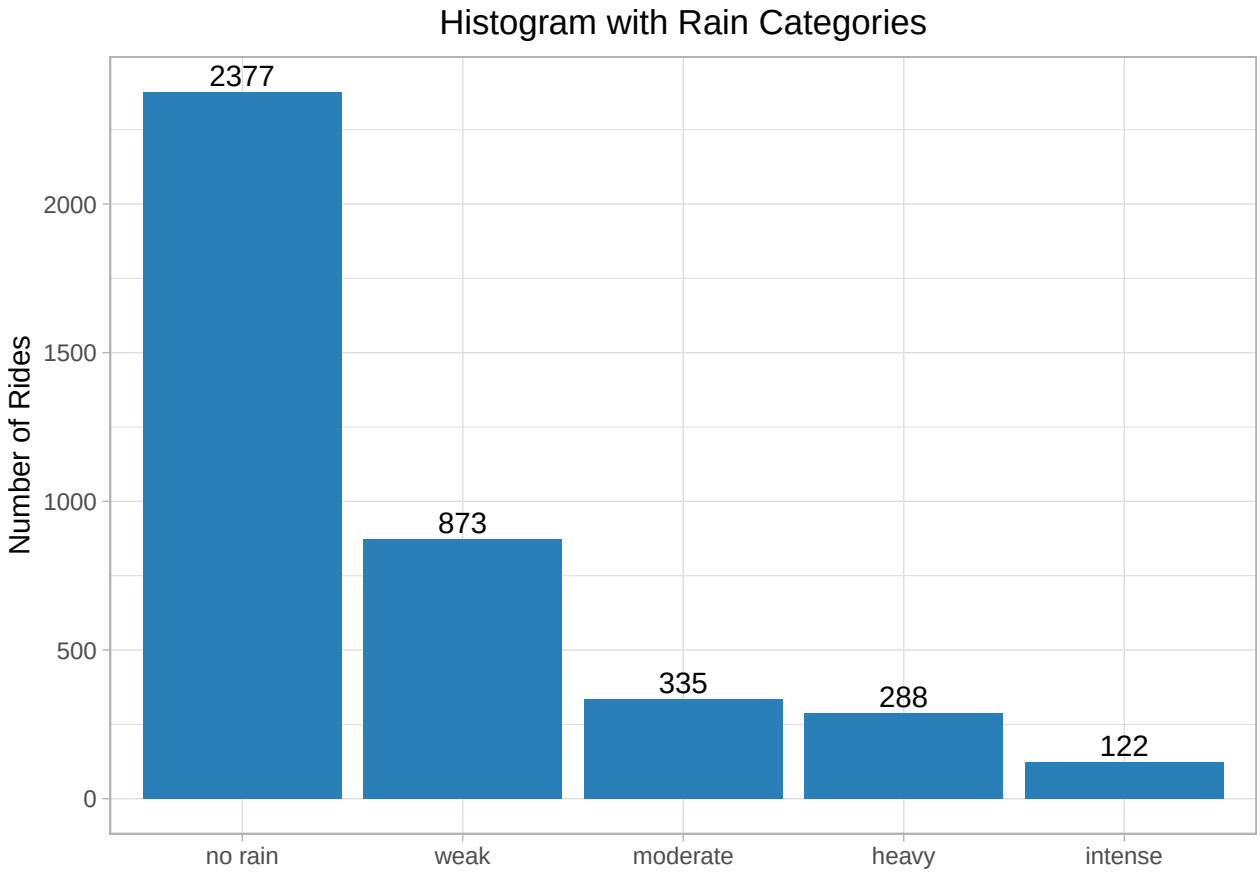

Figure 9: Histogram which shows the distribution by rain category.

```
trip_dur_filter <- citi_cb %>%
  filter(tripduration < 8000)

ggplot(trip_dur_filter, aes(x=PRCP, y=tripduration)) +
  geom_point(color='#2980B9', size=1.5) +
  geom_smooth(method=lm, color='darkred', linetype="dashed") +
  theme_light() +
  ggtitle("Effect of Precipitation on Trip Duration") +
  ggeasy::easy_center_title() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin=unit(c(0,0,-0.04,0), "null")) +
  xlab("Precipitation (in mm)") +
  ylab("Trip duration (in seconds)")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

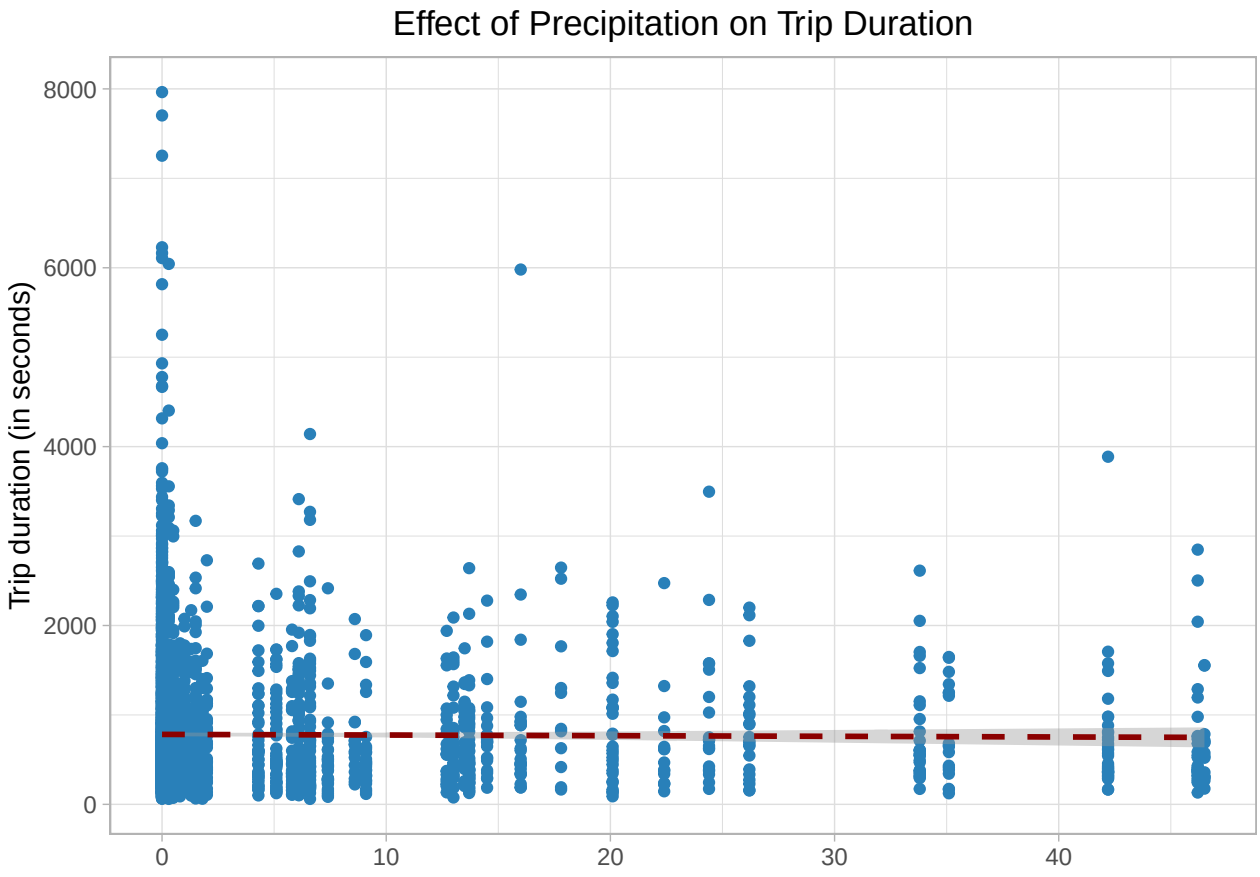

Figure 10: Scatter plot to examine the influence of precipitation on the trip durations.

```
cor_tripduration_precipitation <- round(cor(x=citi_cb$PRCP, y=citi_cb$tripduration,
                                            method="spearman"),2)
```

[1] "Interpretation: For every mm of precipitation, the time a customer rents a bike changes by -0.01 seconds. Since we are using samples and not the whole population, the result needs to be interpreted with caution. To investigate the result further, we use inferential statistics in the form of an ANOVA test."

4.3.3 Influence of Temperature on Trip Duration (in seconds)

Using the analysis of variance, we would like to see whether the factors of average temperature and rain have an influence on the length of a trip.

```
model <- aov(tripduration ~ meantemp + PRCP, data = citi_cb)
summary(model)
```

```
##               Df    Sum Sq  Mean Sq F value   Pr(>F)
## meantemp       1 2.422e+07 24223715  43.938 3.84e-11 ***
## PRCP           1 1.066e+06  1066392   1.934    0.164
## Residuals   3992 2.201e+09   551320
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5 Results

The data analysis carried out has produced many valuable insights. However, it should be mentioned again at this point that the analysis is based on a sample and not on the entirety of the data. For decisions, the entirety or at least a sample containing records of all months since the start of operations in 2013 would be necessary. The data analysis carried out provides the basis for evaluation and could easily be extended with more datasets and the rides and the sample size increased. The findings obtained with the selected sample are as follows:

Most rides are done in the afternoon. During the week, peaks are measured between 08:00 - 09:00 and between 17:00 - 18:00. On weekends, the busiest time is between 11:00 - 16:00. Citibike New York is especially popular in the Manhattan borough. Rain has a strong influence on the number of rides per day. When there is no rain, the Citibikes are used much more often. The average temperature has a highly significant influence on the duration of the trips. Also the rain has a weak influence on the duration of the trips.

List of Figures

List of Tables

6 Sources

[1] Lyft, Inc., “Bike rental NYC: Explore new york like never before – on two wheels! Renting a bike is the best way to see the city.” 2022.Available: https://ride.citibikenyc.com/how-it-works/bike-rental-nyc

[2] Citi Bike NYC, “Citi bike system data | citi bike NYC.” 16.02.2022.Available: https://ride.citibikenyc.com/system-data

[3] National Oceanic and Atmospheric Administration (NOAA), “Climate data online (CDO).” 16.02.2022.Available: https://www.ncdc.noaa.gov/cdo-web/search;jsessionid=F402F60568CE5FC512B4627FEAC3F67C

[4] Bundesamt für Meteorologie und Klimatologie MeteoSchweiz, “Niederschlag - MeteoSchweiz.” n.d.Available: https://www.meteoschweiz.admin.ch/home/wetter/wetterbegriffe/niederschlag.html

7 Appendix: Functions used

```
# knitr::purl("Group18_Submission.Rmd")
# Group18_Submission.R gets created
list.functions.in.file("Group18_Submission.R")
```

```
## $`c("package:dplyr", "package:stats")`
## [1] "filter"
##
## $`character(0)`
## [1] "atop"
##
## $`package:base`
##  [1] "as.Date"       "as.numeric"    "as.POSIXct"    "c"
##  [5] "cbind"         "data.frame"    "expression"    "factor"
##  [9] "file.path"     "ifelse"        "is.na"         "library"
## [13] "options"       "order"         "paste0"        "print"
## [17] "rbind"         "replace"       "require"       "round"
## [21] "rowSums"       "strftime"      "subset"        "substr"
## [25] "sum"           "summary"       "Sys.setlocale" "weekdays"
##
## $`package:data.table`
## [1] "set"      "setnames"
##
## $`package:dplyr`
##  [1] "arrange"    "desc"       "distinct"   "group_by"   "left_join"
##  [6] "mutate_all" "n"          "sample_n"   "select"     "summarise"
## [11] "summarize"
##
## $`package:ggeasy`
## [1] "easy_center_title"
##
## $`package:ggmap`
## [1] "get_stamenmap" "ggmap"
##
## $`package:ggplot2`
##  [1] "aes"                  "coord_flip"           "element_blank"
##  [4] "element_text"         "facet_wrap"           "geom_bar"
##  [7] "geom_boxplot"         "geom_density"         "geom_point"
## [10] "geom_smooth"          "geom_text"            "ggplot"
## [13] "ggtitle"              "labs"                 "scale_fill_gradientn"
## [16] "scale_fill_manual"    "scale_x_discrete"     "stat_density_2d"
## [19] "theme"                "theme_light"          "theme_update"
## [22] "unit"                 "xlab"                 "ylab"
##
## $`package:gridExtra`
## [1] "grid.arrange"
##
## $`package:knitr`
## [1] "kable"
##
## $`package:RColorBrewer`
## [1] "brewer.pal"
##
## $`package:skimr`
## [1] "skim"
```

```
##
## $`package:stats`
## [1] "aov" "cor"
##
## $`package:utils`
## [1] "head"     "read.csv" "str"      "tail"
```