IBM Data Science Capstone Project

Quyen Nguyen

July, 2023

- Web scrape a Global Bike-Sharing Systems Wiki Page
- OpenWeather APIs Calls
    - Get the current weather data for a city
    - Get 5-day weather forecasts for a list of cities
- Data Wrangling with Regular Expressions
    - Standardize column names for all collected datasets
    - Clean up
- Data Wrangling with dplyr
    - Detect and handle missing values
    - Create indicator (dummy) variables for categorical variables
    - Normalize data using Min-Max normalization
- Exploratory Data Analysis
    - Standardize the data
    - Descriptive Statistics
    - Data Visualization
- Predictive Analysis
    - Basic Linear Regression Model
    - Improve the Linear model
- Conclusion

```
if (knitr::is_html_output()) "# [TOC]\n" else ""
```

```
## [1] "# [TOC]\n"
```

Web scrape a Global Bike-Sharing Systems Wiki Page

```
# Load neccessary packages
pacman::p_load(rvest, tidyverse, httr, stringr, magrittr)
```

The dataset we will be using for analysis is the bike sharing system worldwide, which can be accessed from https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems (https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems)

Since this HTML page at least contains three child table nodes under the root HTML node. So, we will need to use html_nodes(root_node, “table”) function to get all its child table nodes, then create a data frame for later analysis

```
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
data <- read_html(url)

#Get all the the child table nodes under the root HTML node
table_nodes <- html_nodes(data, "table")

#Convert data to data frame
bike_df <- data %>%
  html_element("table") %>%
  html_table()
print(df)
```

```
## function (x, df1, df2, ncp, log = FALSE)
## {
##     if (missing(ncp))
##         .Call(C_df, x, df1, df2, log)
##     else .Call(C_dnf, x, df1, df2, ncp, log)
## }
## <bytecode: 0x1103634c8>
## <environment: namespace:stats>
```

```
#Export to a CSV file
write.csv(bike_df,"/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-UniversityofNebraska-Lincoln/R/IBM Data Science with R/raw_bike_sharing_system.csv")
```

OpenWeather APIs Calls

Collecting real-time current and forecast weather data for cities using the OpenWeather API

Get the current weather data for a city

First, using R code to get the current weather data of Seoul and save it into a dataframe

Setting api key

```
#URL for current weather API
current_weather_url <- 'https://api.openweathermap.org/data/2.5/weather'

#List to hold URL parameter
current_query <- list(q="Seoul",appid=api_key,units="metric")

#Make a HTTP request to the current weather API
response <- GET(current_weather_url, query=current_query)
http_type(response)
```

```
## [1] "application/json"
```

```
#Read json http data
json_result <- content(response, as="parsed")
json_result
```

```
## $coord
## $coord$lon
## [1] 126.9778
##
## $coord$lat
## [1] 37.5683
##
##
## $weather
## $weather[[1]]
## $weather[[1]]$id
## [1] 804
##
## $weather[[1]]$main
## [1] "Clouds"
##
## $weather[[1]]$description
## [1] "overcast clouds"
##
## $weather[[1]]$icon
## [1] "04d"
##
##
##
## $base
## [1] "stations"
##
## $main
```

```
## $main$temp
## [1] 26.66
##
## $main$feels_like
## [1] 26.66
##
## $main$temp_min
## [1] 24.69
##
## $main$temp_max
## [1] 26.66
##
## $main$pressure
## [1] 1012
##
## $main$humidity
## [1] 93
##
## $main$sea_level
## [1] 1012
##
## $main$grnd_level
## [1] 1006
##
##
## $visibility
## [1] 10000
##
## $wind
## $wind$speed
## [1] 1.41
##
## $wind$deg
## [1] 214
##
## $wind$gust
## [1] 2.94
##
##
## $clouds
## $clouds$all
## [1] 100
##
##
## $dt
## [1] 1689556085
##
```

```
## $sys
## $sys$type
## [1] 1
##
## $sys$id
## [1] 5509
##
## $sys$country
## [1] "KR"
##
## $sys$sunrise
## [1] 1689539004
##
## $sys$sunset
## [1] 1689591152
##
##
## $timezone
## [1] 32400
##
## $id
## [1] 1835848
##
## $name
## [1] "Seoul"
##
## $cod
## [1] 200
```

```
# Create some empty vectors to hold data temporarily
city <- c()
weather <- c()
visibility <- c()
temp <- c()
temp_min <- c()
temp_max <- c()
pressure <- c()
humidity <- c()
wind_speed <- c()
wind_deg <- c()

#Assign values in json_result into different vector
city <- c(city, json_result$name)
weather <- c(weather, json_result$weather[[1]]$main)
visibility <- c(visibility, json_result$visibility)
temp <- c(temp, json_result$main$temp)
temp_min <-c(temp_min, json_result$main$temp_min)
temp_max <- c(temp_max, json_result$main$temp_max)
pressure <- c(pressure, json_result$main$pressure)
humidity <- c(humidity, json_result$main$humidity)
wind_speed <- c(wind_speed, json_result$wind$speed)
wind_deg <-c(wind_deg, json_result$wind$deg)

#Combine all vector into data frame
weather_df <- data.frame(city = city,
                                weather=weather,
                                 visibility=visibility,
                                 temp=temp,
                                 temp_min=temp_min,
                                 temp_max=temp_max,
                                 pressure=pressure,
                                 humidity=humidity,
                                 wind_speed=wind_speed,
                                 wind_deg=wind_deg)
print(weather_df)
```

```
##    city weather visibility  temp temp_min temp_max pressure humidity wind_speed
## 1 Seoul  Clouds      10000 26.66    24.69    26.66     1012       93       1.41
##   wind_deg
## 1      214
```

Get 5-day weather forecasts for a list of cities

```
# Get 5 -day weather forecast for a list of cities
weather_forecast_by_cities <- function(city_names) {
  df <- data.frame()
  for (city_name in city_names) {
    #forecast API URL
    forecast_url <-'https://api.openweathermap.org/data/2.5/weather'
    #create query parameter
    forecast_query <- list(q=city_name,appid=api_key, units="metric")
    #make HTTP GET call for the given city
    response <- GET(forecast_url, query=forecast_query)
    json_result <- content(response, as="parsed")
    results <- json_result$list
    #Loop the json result
    for(result in results) {
      city <- c(city, city_name)
    }
    # Add R lists into a data frame
    city <- c(city, json_result$name)
    weather <- c(weather, json_result$weather[[1]]$main)
    visibility <- c(visibility, json_result$visibility)
    temp <- c(temp, json_result$main$temp)
    temp_min <-c(temp_min, json_result$main$temp_min)
    temp_max <- c(temp_max, json_result$main$temp_max)
    pressure <- c(pressure, json_result$main$pressure)
    humidity <- c(humidity, json_result$main$humidity)
    wind_speed <- c(wind_speed, json_result$wind$speed)
    wind_deg <-c(wind_deg, json_result$wind$deg)

    #Combine all vector into data frame
    df <- data.frame(city = city,
                             weather=weather,
                              visibility=visibility,
                              temp=temp,
                              temp_min=temp_min,
                              temp_max=temp_max,
                              pressure=pressure,
                              humidity=humidity,
                              wind_speed=wind_speed,
                              wind_deg=wind_deg)
  }
  return(df)
}

cities <- c("Seoul", "Washington, D.C.", "Paris", "Suzhou")
cities_weather_df <- weather_forecast_by_cities(cities)
print(cities_weather_df)
```

```
##         city weather visibility  temp temp_min temp_max pressure humidity
## 1      Seoul  Clouds      10000 26.66    24.69    26.66     1012       93
## 2      Seoul  Clouds      10000 26.66    24.69    26.66     1012       93
## 3 Washington  Clouds      10000 33.46    27.76    35.32     1007       21
## 4      Paris   Clear      10000 15.69    11.34    16.75     1018       76
## 5     Suzhou  Clouds      10000 29.70    29.70    29.70     1008       75
##   wind_speed wind_deg
## 1       1.41      214
## 2       1.41      214
## 3       2.60      316
## 4       1.54      240
## 5       4.97      118
```

The data will be saved to `cities_weather_forecast.csv` for later use.

```
write.csv(cities_weather_df, "/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDri
ve-UniversityofNebraska-Lincoln/R/IBM Data Science with R/cities_weather_forecast.csv
", row.names = FALSE)
```

Data Wrangling with Regular Expressions

In this data collection process, I will collect some raw datasets from several different sources online. Then I will use regular expression along with the `stringr` package to clean-up the bike-sharing systems data.

List of datasets that will be used:

- `raw_bike_sharing_systems.csv` : A list of active bike-sharing systems across the world
- `raw_cities_weather_forecast.csv` : 5-day weather forecasts for a list of cities, from OpenWeather API
- `raw_worldcities.csv` : A list of major cities' info (such as name, latitude and longitude) across the world
- `raw_seoul_bike_sharing.csv` : Weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour, and date information, from Seoul bike-sharing systems

Download datasets

```
url1 <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevelo
perSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_worldcities.csv"
download.file(url1, destfile="/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDri
ve-UniversityofNebraska-Lincoln/R/IBM Data Science with R/raw_worldcities.csv")
download.file("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM
DeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_seoul_bike_sharing.cs
v",destfile = "raw_seoul_bike_sharing.csv")
```

Put the dataset downloaded into the datasets_list

```
dataset_list <- c('raw_bike_sharing_system.csv', 'raw_seoul_bike_sharing.csv', 'citie
s_weather_forecast.csv', 'raw_worldcities.csv')
```

Standardize column names for all collected datasets

```
#Convert iterate over the above datasets and convert their column names
for (dataset_name in dataset_list) {
  dataset <- read.csv(dataset_name)
  names(dataset) <- toupper(names(dataset))
  names(dataset) <- str_replace_all(names(dataset), " ", "_")
  write.csv(dataset, dataset_name, row.names = FALSE)
}
```

Clean up

Since the datasets are downloaded from the web, there are some values needed to cleaning up. For this project, we will focus on processing some relevant columns: COUNTRY, CITY, SYSTEM, BICYCLES

First load the datasets and take a look at it

```
bike_sharing_df <- read.csv("/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDriv
e-UniversityofNebraska-Lincoln/R/IBM Data Science with R/raw_bike_sharing_system.csv"
)
head(bike_sharing_df)
```

```
##   X   COUNTRY                   CITY                  NAME            SYSTEM
## 1 1   Albania              Tirana[5]              Ecovolis
## 2 2 Argentina    Buenos Aires[6][7]               Ecobici Serttel Brasil[8]
## 3 3 Argentina           Mendoza[10]             Metrobici
## 4 4 Argentina               Rosario  Mi Bici Tu Bici[11]
## 5 5 Argentina San Lorenzo, Santa Fe              Biciudad           Biciudad
## 6 6 Australia         Melbourne[12] Melbourne Bike Share         PBSC & 8D
##                       OPERATOR          LAUNCHED         DISCONTINUED STATIONS
## 1                                      March 2011                             8
## 2 Bike In Baires Consortium[9]               2010                           400
## 3                                            2014                             2
## 4                               2 December 2015                            47
## 5                              27 November 2016                             8
## 6                       Motivate        June 2010 30 November 2019[13]       53
##   BICYCLES DAILY.RIDERSHIP
## 1      200
## 2     4000           21917
## 3       40
## 4      480
## 5       80
## 6      676
```

Create a sub data frame of these four columns to process

```
sub_bike_sharing_df <- bike_sharing_df %>%
  select(COUNTRY, CITY, SYSTEM, BICYCLES)

#Check the type of data in those columns
sub_bike_sharing_df %>%
  summarize_all(class) %>%
  gather(variable, class)
```

```
##   variable     class
## 1  COUNTRY character
## 2     CITY character
## 3   SYSTEM character
## 4 BICYCLES character
```

Check why column BYCICLES is in character class

```
find_character <- function(strings) grepl("[^0-9]", strings) #Create Function

sub_bike_sharing_df %>%
  select(BICYCLES) %>% #Use the function to check BICYCLES column
  filter(find_character(BICYCLES)) %>%
  slice(0:10)
```

```
##                                BICYCLES
## 1                       1790 (2019)[21]
## 2                           4200 (2021)
## 3                              4115[25]
## 4  7270 (regular)  2395 (electric)[38]
## 5                               310[65]
## 6                               500[75]
## 7                                  [78]
## 8                               180[79]
## 9                               600[82]
## 10          initially 800 (later 2500)
```

Because there are some values associated with numeric and non-numeric value, BYCICLES was classified as character.

Check if COUNTRY, CITY, SYSTEM have any reference link, such as Melbourne[12]

```
#Create a function to check if there is any reference link in the values
ref_pattern <- "\\[[A-z0-9]+\\]"
find_reference_pattern <- function(strings) grepl(ref_pattern, strings)
```

```
# Use the function to check if COUNTRY, CITY, SYSTEM have any reference link
sub_bike_sharing_df %>%
  select(COUNTRY) %>%
  filter(find_reference_pattern(COUNTRY)) %>%
  slice(1:10) #subset the df with first 11 rows (code will quickly find the match the
filter criteria without overwhelming)
```

```
## [1] COUNTRY
## <0 rows> (or 0-length row.names)
```

```
sub_bike_sharing_df %>%
  select(CITY) %>%
  filter(find_reference_pattern(CITY)) %>%
  slice(1:10)
```

```
##                       CITY
## 1                 Tirana[5]
## 2        Buenos Aires[6][7]
## 3                Mendoza[10]
## 4             Melbourne[12]
## 5             Melbourne[12]
## 6          Brisbane[14][15]
## 7         Lower Austria[16]
## 8  Different locations[19]
## 9              Brussels[24]
## 10                Namur[26]
```

```
sub_bike_sharing_df %>%
  select(SYSTEM) %>%
  filter(find_reference_pattern(SYSTEM)) %>%
  slice(1:10)
```

```
##                              SYSTEM
## 1                  Serttel Brasil[8]
## 2                       EasyBike[64]
## 3                         4 Gen.[72]
## 4               3 Gen. SmooveKey[135]
## 5 3 Gen. Smoove[162][163][164][160]
## 6                  3 Gen. Smoove[200]
## 7                  3 Gen. Smoove[202]
## 8                  3 Gen. Smoove[204]
```

COUNTRY column is clean, CITY and SYSTEM have some reference links need to be cleaned

```
#Create a function to remove reference links
remove_ref <- function(strings) {
  ref_pattern <- "\\[[A-z0-9]+\\]" # Define a pattern matching a reference link such
as [1]
  result <- stringr::str_replace_all(strings,ref_pattern,"")  # Replace all matched s
ubstrings with a white space
  result <-  trimws(result)
    return(result)
}
```

```
# Use the function to remove the reference links
sub_bike_sharing_df %<>% #use mutate and remove_ref fcn to remove ref in CITY and SYS
TEM
  mutate(SYSTEM=remove_ref(SYSTEM),
         CITY=remove_ref(CITY))

# Check whether all reference links are removed
sub_bike_sharing_df %>%
  select(COUNTRY, CITY, SYSTEM, BICYCLES) %>%
  filter(find_reference_pattern(COUNTRY) | find_reference_pattern(CITY) | find_refere
nce_pattern(CITY) |find_reference_pattern(BICYCLES) )
```

```
##           COUNTRY                                                 CITY
## 1         Belgium                                  Different locations
## 2         Belgium                                             Brussels
## 3          Canada                                             Montreal
## 4          Cyprus                           Limassol, Nicosia District
## 5         Czechia                                               Prague
## 6         Czechia                                             Prague 7
## 7         Czechia                                            Prostějov
## 8         Czechia                                              Ostrava
## 9         Denmark                                                Farsø
## 10        Finland                                              Kouvola
## 11        Finland                                               Kuopio
## 12        Finland                                                Lahti
## 13        Finland                                           Lappeenranta
## 14        Finland                                                 Pori
## 15        Finland                                              Raseborg
## 16        Finland                                             Riihimäki
## 17        Finland                                               Tampere
## 18        Finland                                                 Turku
## 19        Finland                                               Varkaus
## 20        Georgia                                                Batumi
## 21        Germany                                             Darmstadt
## 22         Greece                                                 Corfu
## 23        Hungary                                              Budapest
## 24        Hungary                                                  Győr
## 25        Hungary                                             Kaposvár
## 26          Italy                                                Milan
## 27      Lithuania                                               Kaunas
## 28    Netherlands Various Locations (especially railway stations)
## 29         Russia                                                Kazan
## 30       Slovakia                                           Bratislava
## 31       Slovakia                                           Bratislava
## 32       Slovakia                                               Košice
## 33       Slovakia                                 Moldava nad Bodvou
```

```
## 34       Slovakia                                        Žilina
## 35    South Korea                                      Changwon
## 36 United Kingdom                           Glasgow, Scotland
## 37 United Kingdom                 Greater Manchester, England
## 38 United Kingdom                         Edinburgh, Scotland
## 39 United Kingdom                          Liverpool, England
##                            SYSTEM                            BICYCLES
## 1                      Blue-bike                  1790 (2019)[21]
## 2                3 Gen. Cyclocity                        4115[25]
## 3                      PBSC & 8D 7270 (regular)  2395 (electric)[38]
## 4                  3 Gen. Smoove                         310[65]
## 5                                                        500[75]
## 6                      4 Gen. Ofo                           [78]
## 7                3 Gen. nextbike                         180[79]
## 8                3 Gen. nextbike                         600[82]
## 9                          2 Gen                           [89]
## 10               Donkey Republic  60(regular) 10(electric)(2022)[96]
## 11                      Freebike                350(2022) [97]
## 12                      Freebike               250 (2022) [98]
## 13               Donkey Republic                 120(2022)[99]
## 14                     Rolanbike               50 (2022) [103]
## 15               Donkey Republic               30 (2022) [104]
## 16               Donkey Republic              60 (2022)  [105]
## 17               CityBike Global               700 (2022)[106]
## 18                      Nextbike               700 (2022) [107]
## 19                          Juro                20 (2022) [110]
## 20               3 Gen. SmooveKey                       370[136]
## 21    3 & 4 Gen. Call a Bike flex                      350 [147]
## 22                 3 Gen. Smoove                        100[64]
## 23                        3 Gen.                       1526[180]
## 24                                                      180[182]
## 25                                32 (including 6 rollers)  [183]
## 26               3 Gen. Clear CC              5430 (1000 E)[195]
## 27                                                    150 E[207]
## 28 OV-Fiets/Nederlandse Spoorwegen                   21700 [210]
## 29               3 Gen. Cyclocity                        120[238]
## 30                                                      400[240]
## 31                                                       80[242]
## 32                                                      500[246]
## 33                                                       25[249]
## 34                      nextbike                        123[263]
## 35                                                     2348[272]
## 36               3 Gen. nextbike                         400[294]
## 37                                                     1500[296]
## 38                 Urban Sharing                         500[297]
## 39                                                     1000[298]
```

Extract the numeric value to clean the BICYCLES column

```
extract_num <- function(columns) {
  digitals_pattern <- "\\d+" #define a pattern matching digital substring
  str_extract(columns,digitals_pattern) %>%
  as.numeric()
}

sub_bike_sharing_df %<>% #use mutate and to apply function to BICYLCLES
  mutate(BICYCLES=extract_num(BICYCLES))
```

Write the clean dataset to CSV file

```
write.csv(sub_bike_sharing_df,"/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDr
ive-UniversityofNebraska-Lincoln/R/IBM Data Science with R/bike_sharing_system.csv")
```

Data Wrangling with dplyr

This part will focus on wrangling the Seoul bike-sharing demand historical dataset. This is the core dataset to build a predictive model later.

Detect and handle missing values

Standardize the column name for later use

```
dataset_list <- c('bike_sharing_system.csv','raw_seoul_bike_sharing.csv')
for (dataset_name in dataset_list) {
  dataset <- read.csv(dataset_name)
  names(dataset) <- toupper(names(dataset))
  names(dataset) <- str_replace_all(names(dataset), " ", "_")
  write.csv(dataset, dataset_name, row.names = FALSE)
}
```

```
# Load the dataset
bike_sharing_df <- read.csv("raw_seoul_bike_sharing.csv")
summary(bike_sharing_df)
```

```
##      DATE           RENTED_BIKE_COUNT      HOUR        TEMPERATURE
##  Length:8760        Min.   :   2.0    Min.   : 0.00   Min.   :-17.80
##  Class :character   1st Qu.: 214.0    1st Qu.: 5.75   1st Qu.:  3.40
##  Mode  :character   Median : 542.0    Median :11.50   Median : 13.70
##                     Mean   : 729.2    Mean   :11.50   Mean   : 12.87
##                     3rd Qu.:1084.0    3rd Qu.:17.25   3rd Qu.: 22.50
##                     Max.   :3556.0    Max.   :23.00   Max.   : 39.40
##                     NA's   :295                       NA's   :11
##     HUMIDITY       WIND_SPEED      VISIBILITY   DEW_POINT_TEMPERATURE
##  Min.   : 0.00   Min.   :0.000   Min.   :  27   Min.   :-30.600
##  1st Qu.:42.00   1st Qu.:0.900   1st Qu.: 940   1st Qu.: -4.700
##  Median :57.00   Median :1.500   Median :1698   Median :  5.100
##  Mean   :58.23   Mean   :1.725   Mean   :1437   Mean   :  4.074
##  3rd Qu.:74.00   3rd Qu.:2.300   3rd Qu.:2000   3rd Qu.: 14.800
##  Max.   :98.00   Max.   :7.400   Max.   :2000   Max.   : 27.200
##
##  SOLAR_RADIATION     RAINFALL          SNOWFALL         SEASONS
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.00000   Length:8760
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Class :character
##  Median :0.0100   Median : 0.0000   Median :0.00000   Mode  :character
##  Mean   :0.5691   Mean   : 0.1487   Mean   :0.07507
##  3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   :3.5200   Max.   :35.0000   Max.   :8.80000
##
##    HOLIDAY          FUNCTIONING_DAY
##  Length:8760        Length:8760
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
dim(bike_sharing_df) #show the dimension: number of rows, number of columns
```

```
## [1] 8760   14
```

Handle missing value in RENTED_BIKE_COUNT and TEMPERATURE column

```
bike_sharing_df <- drop_na(bike_sharing_df, RENTED_BIKE_COUNT) #drop the NA value bec
ause this is a dependent variable, only 3% of the dataset
```

```
na_rows <- bike_sharing_df[is.na(bike_sharing_df$TEMPERATURE), ]
print(na_rows) #-> all the NA value is in the summer
```

```
##            DATE RENTED_BIKE_COUNT HOUR TEMPERATURE HUMIDITY WIND_SPEED
## 4483 07/06/2018              3221   18          NA       57        2.7
## 4599 12/06/2018              1246   14          NA       45        2.2
## 4626 13/06/2018              2664   17          NA       57        3.3
## 4722 17/06/2018              2330   17          NA       58        3.3
## 4796 20/06/2018              2741   19          NA       61        2.7
## 5030 30/06/2018              1144   13          NA       87        1.7
## 5147 05/07/2018               827   10          NA       75        1.1
## 5290 11/07/2018               634    9          NA       96        0.6
## 5311 12/07/2018               593    6          NA       93        1.1
## 5525 21/07/2018               347    4          NA       77        1.2
## 6288 21/08/2018              1277   23          NA       75        0.1
##      VISIBILITY DEW_POINT_TEMPERATURE SOLAR_RADIATION RAINFALL SNOWFALL SEASONS
## 4483       1217                  16.4            0.96      0.0        0  Summer
## 4599       1961                  12.7            1.39      0.0        0  Summer
## 4626        919                  16.4            0.87      0.0        0  Summer
## 4722        865                  16.7            0.66      0.0        0  Summer
## 4796       1236                  17.5            0.60      0.0        0  Summer
## 5030        390                  23.2            0.71      3.5        0  Summer
## 5147       1028                  20.8            1.22      0.0        0  Summer
## 5290        450                  24.9            0.41      0.0        0  Summer
## 5311        852                  24.3            0.01      0.0        0  Summer
## 5525       1203                  21.2            0.00      0.0        0  Summer
## 6288       1892                  20.8            0.00      0.0        0  Summer
##         HOLIDAY FUNCTIONING_DAY
## 4483 No Holiday             Yes
## 4599 No Holiday             Yes
## 4626 No Holiday             Yes
## 4722 No Holiday             Yes
## 4796 No Holiday             Yes
## 5030 No Holiday             Yes
## 5147 No Holiday             Yes
## 5290 No Holiday             Yes
## 5311 No Holiday             Yes
## 5525 No Holiday             Yes
## 6288 No Holiday             Yes
```

Since all of the NA values in TEMPERATURE is in the summer and TEMPERATURE is an independent variables, they can't be dropped but should be replaced with the average temperature in summer.

```
#calculate the average temperature in summer
summer_temp <- bike_sharing_df[bike_sharing_df$SEASONS == "Summer", ]
summer_avg_temp <- mean(summer_temp$TEMPERATURE, na.rm=TRUE)
print(summer_avg_temp)
```

```
## [1] 26.58771
```

```
# replace NA with average temperature in summer
bike_sharing_df["TEMPERATURE"][is.na(bike_sharing_df["TEMPERATURE"])] <- summer_avg_t
emp
```

Save the clean dataset

```
write.csv(bike_sharing_df,"/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-
UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing.csv")
```

Create indicator (dummy) variables for categorical variables

```
library(fastDummies) #package to create dummy variables
```

```
bike_sharing_df <- read.csv("/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDriv
e-UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing.csv")
```

In the bike-sharing demand dataset, SEASONS, HOLIDAY, FUNCTIONING_DAY are categorical variable. HOUR is considered categorical variable because it levels range from 0 to 23

```
bike_sharing_df %>%
  mutate(HOUR=as.character(HOUR)) %>%  #convert HOUR to character because it's from 0
to 23
head(10)
```

```
##     X       DATE RENTED_BIKE_COUNT HOUR TEMPERATURE HUMIDITY WIND_SPEED
## 1   1 01/12/2017              254    0        -5.2       37        2.2
## 2   2 01/12/2017              204    1        -5.5       38        0.8
## 3   3 01/12/2017              173    2        -6.0       39        1.0
## 4   4 01/12/2017              107    3        -6.2       40        0.9
## 5   5 01/12/2017               78    4        -6.0       36        2.3
## 6   6 01/12/2017              100    5        -6.4       37        1.5
## 7   7 01/12/2017              181    6        -6.6       35        1.3
## 8   8 01/12/2017              460    7        -7.4       38        0.9
## 9   9 01/12/2017              930    8        -7.6       37        1.1
## 10 10 01/12/2017              490    9        -6.5       27        0.5
##    VISIBILITY DEW_POINT_TEMPERATURE SOLAR_RADIATION RAINFALL SNOWFALL SEASONS
## 1        2000                -17.6            0.00        0        0  Winter
## 2        2000                -17.6            0.00        0        0  Winter
## 3        2000                -17.7            0.00        0        0  Winter
## 4        2000                -17.6            0.00        0        0  Winter
## 5        2000                -18.6            0.00        0        0  Winter
## 6        2000                -18.7            0.00        0        0  Winter
## 7        2000                -19.5            0.00        0        0  Winter
## 8        2000                -19.3            0.00        0        0  Winter
## 9        2000                -19.8            0.01        0        0  Winter
## 10       1928                -22.4            0.23        0        0  Winter
##       HOLIDAY FUNCTIONING_DAY
## 1  No Holiday             Yes
## 2  No Holiday             Yes
## 3  No Holiday             Yes
## 4  No Holiday             Yes
## 5  No Holiday             Yes
## 6  No Holiday             Yes
## 7  No Holiday             Yes
## 8  No Holiday             Yes
## 9  No Holiday             Yes
## 10 No Holiday             Yes
```

For later usage to build the model, SEASONS, HOLIDAY and HOUE should be converted into indicator columns.

```
# load the package
pacman::p_load(fastDummies)
```

```
bike_sharing_df <- dummy_cols(bike_sharing_df, select_columns = "HOUR")
bike_sharing_df <- dummy_cols(bike_sharing_df, select_columns = "HOLIDAY")
bike_sharing_df <- dummy_cols(bike_sharing_df, select_columns = "SEASONS")

#Change the colnames for shorterning
colnames(bike_sharing_df)[c(40,41,42,43,44,45)] <- c("HOLIDAY", "NO HOLIDAY", "AUTUMN
", "SPRING", "SUMMER","WINTER")
```

Save the dataset

```
write.csv(bike_sharing_df,"/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-
UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing_converted.c
sv")
```

Normalize data using Min-Max normalization

```
#Create a function for min-max normalization
minmax_norm <- function(x){
  (x-min(x))/(max(x)-min(x))}
```

```
bike_sharing_df <- read.csv("/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDriv
e-UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing_converted
.csv")

#Apply min-max normalization function to numerical columns in df
bike_sharing_df %<>%
  mutate(TEMPERATURE = minmax_norm(TEMPERATURE),
         HUMIDITY = minmax_norm(HUMIDITY),
         WIND_SPEED = minmax_norm(WIND_SPEED),
         VISIBILITY = minmax_norm(VISIBILITY),
         DEW_POINT_TEMPERATURE = minmax_norm(DEW_POINT_TEMPERATURE),
         SOLAR_RADIATION = minmax_norm(SOLAR_RADIATION),
         RAINFALL = minmax_norm(RAINFALL),
         SNOWFALL = minmax_norm(SNOWFALL))
head(bike_sharing_df)
```

```
##   X.1 X       DATE RENTED_BIKE_COUNT HOUR TEMPERATURE  HUMIDITY WIND_SPEED
## 1   1 1 01/12/2017              254    0   0.2202797 0.3775510  0.2972973
## 2   2 2 01/12/2017              204    1   0.2150350 0.3877551  0.1081081
## 3   3 3 01/12/2017              173    2   0.2062937 0.3979592  0.1351351
## 4   4 4 01/12/2017              107    3   0.2027972 0.4081633  0.1216216
## 5   5 5 01/12/2017               78    4   0.2062937 0.3673469  0.3108108
## 6   6 6 01/12/2017              100    5   0.1993007 0.3775510  0.2027027
##   VISIBILITY DEW_POINT_TEMPERATURE SOLAR_RADIATION RAINFALL SNOWFALL SEASONS
## 1          1             0.2249135               0        0        0  Winter
## 2          1             0.2249135               0        0        0  Winter
## 3          1             0.2231834               0        0        0  Winter
## 4          1             0.2249135               0        0        0  Winter
## 5          1             0.2076125               0        0        0  Winter
## 6          1             0.2058824               0        0        0  Winter
##      HOLIDAY FUNCTIONING_DAY HOUR_0 HOUR_1 HOUR_2 HOUR_3 HOUR_4 HOUR_5 HOUR_6
## 1 No Holiday             Yes      1      0      0      0      0      0      0
## 2 No Holiday             Yes      0      1      0      0      0      0      0
## 3 No Holiday             Yes      0      0      1      0      0      0      0
## 4 No Holiday             Yes      0      0      0      1      0      0      0
## 5 No Holiday             Yes      0      0      0      0      1      0      0
## 6 No Holiday             Yes      0      0      0      0      0      1      0
##   HOUR_7 HOUR_8 HOUR_9 HOUR_10 HOUR_11 HOUR_12 HOUR_13 HOUR_14 HOUR_15 HOUR_16
## 1      0      0      0       0       0       0       0       0       0       0
## 2      0      0      0       0       0       0       0       0       0       0
## 3      0      0      0       0       0       0       0       0       0       0
## 4      0      0      0       0       0       0       0       0       0       0
## 5      0      0      0       0       0       0       0       0       0       0
## 6      0      0      0       0       0       0       0       0       0       0
##   HOUR_17 HOUR_18 HOUR_19 HOUR_20 HOUR_21 HOUR_22 HOUR_23 HOLIDAY.1 NO.HOLIDAY
## 1       0       0       0       0       0       0       0         0          1
## 2       0       0       0       0       0       0       0         0          1
## 3       0       0       0       0       0       0       0         0          1
## 4       0       0       0       0       0       0       0         0          1
## 5       0       0       0       0       0       0       0         0          1
## 6       0       0       0       0       0       0       0         0          1
##   AUTUMN SPRING SUMMER WINTER
## 1      0      0      0      1
## 2      0      0      0      1
## 3      0      0      0      1
## 4      0      0      0      1
## 5      0      0      0      1
## 6      0      0      0      1
```

Save the dataset

```
#Save as seoul_bike_sharing_converted_normalized.csv
write.csv(bike_sharing_df,"/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing_converted_normalized.csv")
```

Standardize the column names again for the new datasets

```
dataset_list <- c('seoul_bike_sharing.csv', 'seoul_bike_sharing_converted.csv', 'seoul_bike_sharing_converted_normalized.csv')

for (dataset_name in dataset_list) {
  dataset <- read.csv(dataset_name)
  names(dataset) <- toupper(names(dataset))
  names(dataset) <- str_replace_all(names(dataset), " ", "_")
  write.csv(dataset, dataset_name, row.names = FALSE)
}
```

Exploratory Data Analysis

This part is to perform Exploratory Data Analysis using tidyverse and ggplot2 R packages, with the objective is to explore and generate some insights from the analysis.

Standardize the data

```
seoul_bike_sharing <- read.csv("/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing.csv")
str(seoul_bike_sharing)
```

```
## 'data.frame':    8465 obs. of  15 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ DATE                 : chr  "01/12/2017" "01/12/2017" "01/12/2017" "01/12/2017"
...
##  $ RENTED_BIKE_COUNT    : int  254 204 173 107 78 100 181 460 930 490 ...
##  $ HOUR                 : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ TEMPERATURE          : num  -5.2 -5.5 -6 -6.2 -6 -6.4 -6.6 -7.4 -7.6 -6.5 ...
##  $ HUMIDITY             : int  37 38 39 40 36 37 35 38 37 27 ...
##  $ WIND_SPEED           : num  2.2 0.8 1 0.9 2.3 1.5 1.3 0.9 1.1 0.5 ...
##  $ VISIBILITY           : int  2000 2000 2000 2000 2000 2000 2000 2000 2000 1928 .
..
##  $ DEW_POINT_TEMPERATURE: num  -17.6 -17.6 -17.7 -17.6 -18.6 -18.7 -19.5 -19.3 -19
.8 -22.4 ...
##  $ SOLAR_RADIATION      : num  0 0 0 0 0 0 0 0 0.01 0.23 ...
##  $ RAINFALL             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SNOWFALL             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SEASONS              : chr  "Winter" "Winter" "Winter" "Winter" ...
##  $ HOLIDAY              : chr  "No Holiday" "No Holiday" "No Holiday" "No Holiday"
...
##  $ FUNCTIONING_DAY      : chr  "Yes" "Yes" "Yes" "Yes" ...
```

```
 #make sure there's no missing values
```

```
seoul_bike_sharing$DATE = as.Date(seoul_bike_sharing$DATE,format="%d/%m/%Y") #recast
date as a date format
seoul_bike_sharing$HOUR <- factor(seoul_bike_sharing$HOUR, levels = 0:23, ordered = T
RUE) #cast the HOUR as categorical variables
seoul_bike_sharing$SEASONS <- factor(seoul_bike_sharing$SEASONS, levels=c("Winter", "
Spring", "Summer","Autumn"))
class(seoul_bike_sharing$HOUR)
```

```
## [1] "ordered" "factor"
```

```
class(seoul_bike_sharing$DATE)
```

```
## [1] "Date"
```

```
class(seoul_bike_sharing$SEASONS)
```

```
## [1] "factor"
```

```
sum(is.na(seoul_bike_sharing))
```

```
## [1] 0
```

Descriptive Statistics

```
summary(seoul_bike_sharing)
```

```
##        X             DATE            RENTED_BIKE_COUNT      HOUR
##  Min.   :   1   Min.   :2017-12-01   Min.   :   2.0    7      : 353
##  1st Qu.:2117   1st Qu.:2018-02-27   1st Qu.: 214.0    8      : 353
##  Median :4233   Median :2018-05-28   Median : 542.0    9      : 353
##  Mean   :4233   Mean   :2018-05-28   Mean   : 729.2    10     : 353
##  3rd Qu.:6349   3rd Qu.:2018-08-24   3rd Qu.:1084.0    11     : 353
##  Max.   :8465   Max.   :2018-11-30   Max.   :3556.0    12     : 353
##                                                       (Other):6347
##   TEMPERATURE        HUMIDITY       WIND_SPEED      VISIBILITY
##  Min.   :-17.80   Min.   : 0.00   Min.   :0.000   Min.   :  27
##  1st Qu.:  3.00   1st Qu.:42.00   1st Qu.:0.900   1st Qu.: 935
##  Median : 13.50   Median :57.00   Median :1.500   Median :1690
##  Mean   : 12.77   Mean   :58.15   Mean   :1.726   Mean   :1434
##  3rd Qu.: 22.70   3rd Qu.:74.00   3rd Qu.:2.300   3rd Qu.:2000
##  Max.   : 39.40   Max.   :98.00   Max.   :7.400   Max.   :2000
##
##  DEW_POINT_TEMPERATURE SOLAR_RADIATION     RAINFALL          SNOWFALL
##  Min.   :-30.600       Min.   :0.0000   Min.   : 0.0000   Min.   :0.00000
##  1st Qu.: -5.100       1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000
##  Median :  4.700       Median :0.0100   Median : 0.0000   Median :0.00000
##  Mean   :  3.945       Mean   :0.5679   Mean   : 0.1491   Mean   :0.07769
##  3rd Qu.: 15.200       3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   : 27.200       Max.   :3.5200   Max.   :35.0000   Max.   :8.80000
##
##    SEASONS       HOLIDAY          FUNCTIONING_DAY
##  Winter:2160   Length:8465        Length:8465
##  Spring:2160   Class :character   Class :character
##  Summer:2208   Mode  :character   Mode  :character
##  Autumn:1937
##
##
##
```

```
#calculate how many holiday there are
holiday_count <- table(seoul_bike_sharing$HOLIDAY)
num_holiday <- holiday_count['Holiday']
num_holiday
```

```
## Holiday
##     408
```

```
#% if records fall on a holiday
num_holiday/(num_holiday +holiday_count['No Holiday'])
```

```
##    Holiday
## 0.04819846
```

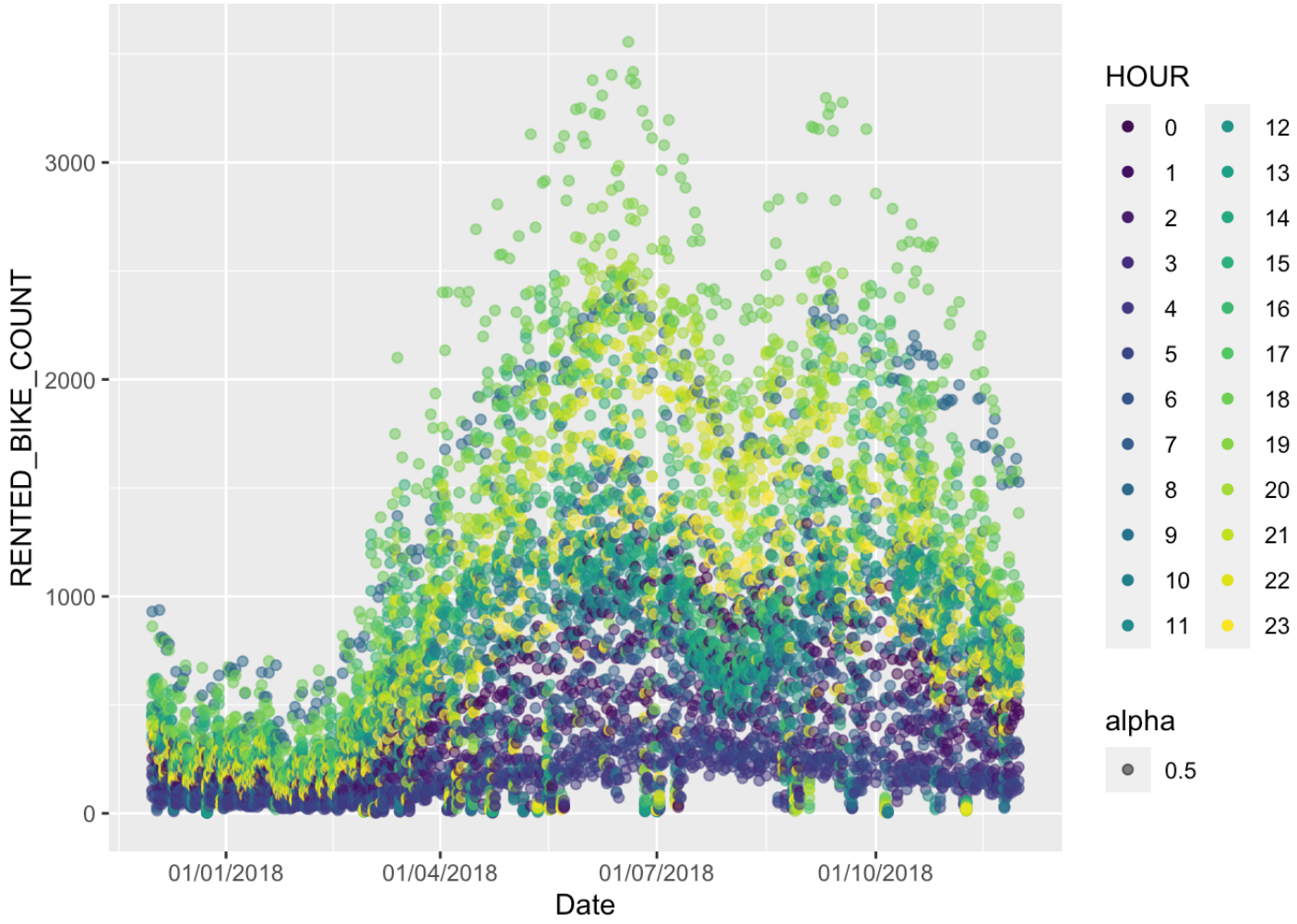
```
#calculate the number of rainfall and snowfall by seasons
seasonal_total <- seoul_bike_sharing %>%
  group_by(SEASONS) %>%
  summarize(total_rainfall=sum(RAINFALL), total_snowfall=sum(SNOWFALL))
seasonal_total
```

```
## # A tibble: 4 × 3
##   SEASONS total_rainfall total_snowfall
##   <fct>            <dbl>          <dbl>
## 1 Winter            70.9           535.
## 2 Spring           404.              0
## 3 Summer           560.              0
## 4 Autumn           228.            123
```

Data Visualization

Scatter plot of RENTED_BIKE_COUNT and DATE (plot1.png)

```
seoul_bike_sharing %>%
  mutate(DATE = as.Date(DATE,format="%d/%m/%Y"))%>%
  ggplot(aes(x = as.Date(DATE), y = RENTED_BIKE_COUNT, color = HOUR, alpha=0.5)) +
  geom_point() +
  scale_x_date(date_labels = "%d/%m/%Y") +
  labs(x= "Date")
```

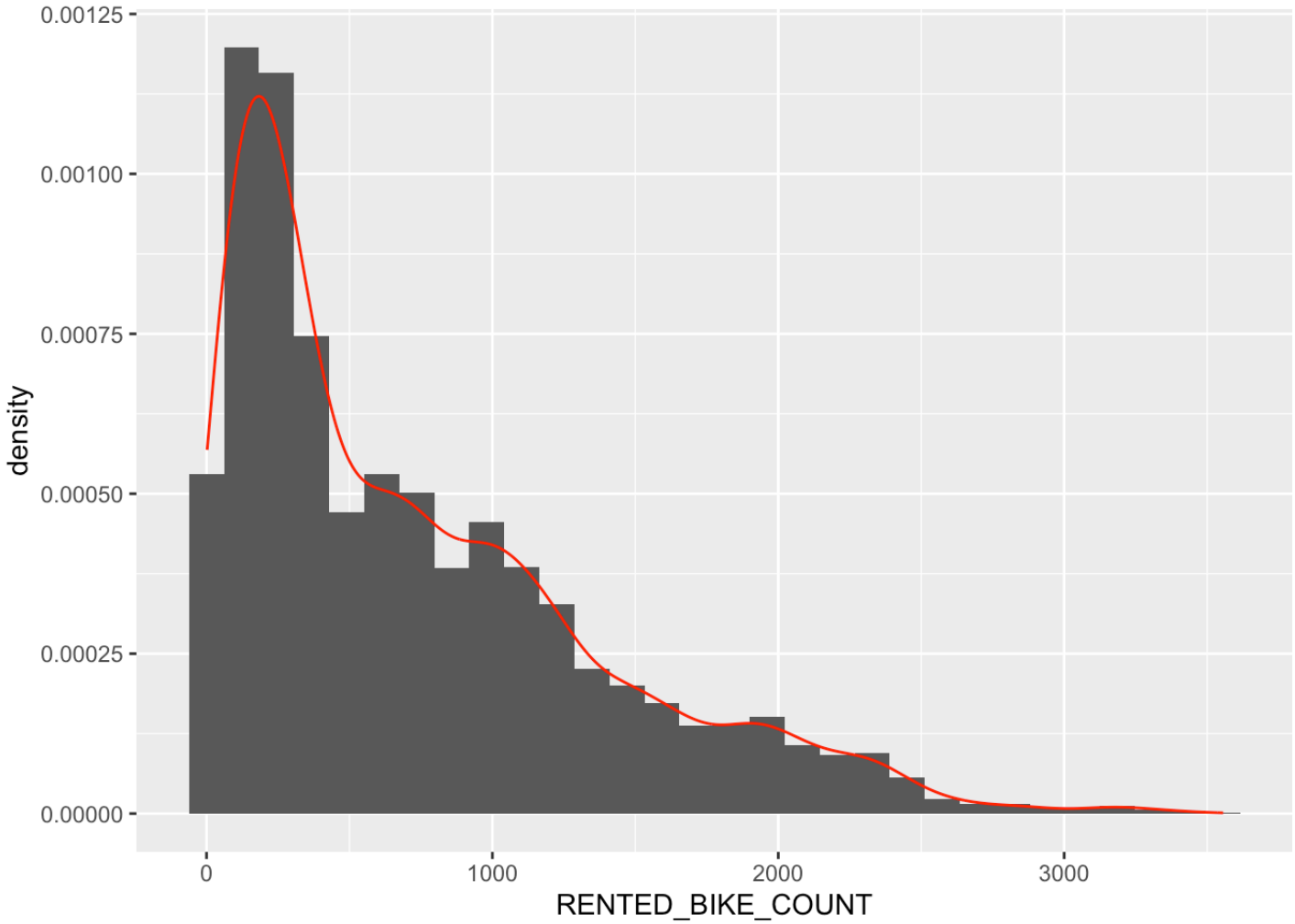


Histogram overlaid with a kernel density curve (histogram.png)

```
ggplot(seoul_bike_sharing, aes(RENTED_BIKE_COUNT)) +
  geom_histogram(aes(y=..density..))+
  geom_density(col="red")
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can see from the histogram that most of the time there are relatively few bikes rented, mode is about 250.

Predictive Analysis

Basic Linear Regression Model

```
#Load the packages:
pacman::p_load(rlang, tidymodels, stringr, broom, ggplot2)

#Load the dataset
bike_sharing_df <- read.csv("/Users/ngocquyenquyennguyen/Library/CloudStorage/OneDrive-UniversityofNebraska-Lincoln/R/IBM Data Science with R/seoul_bike_sharing_converted_normalized.csv")
```

Since DATE and FUNCTIONING_DAY is unnecessary, they are dropped

```
bike_sharing_df <- bike_sharing_df %>%
  select(-DATE, -FUNCTIONING_DAY, -X.2, -X.1, -X, -HOUR, -SEASONS, -HOLIDAY)
```

```
colnames(bike_sharing_df)[c(34,35)] <- c("HOLIDAY", "NO_HOLIDAY")
```

Split the training and testing data with 75% of the original dataset

```
set.seed(1234)
data_split <- initial_split(bike_sharing_df, prop = 3/4) #set the training dataset wi
th 75% of the original dataset
bike_train <- training(data_split)
bike_test <- testing(data_split)
```

Build some simple linear models

```
# Task: Build a linear regression model using weather variables only
lm_model_weather <- lm(RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + VISI
BILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL,
                       data=bike_train)
summary(lm_model_weather)
```

```
##
## Call:
## lm(formula = RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED +
##     VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL +
##     SNOWFALL, data = bike_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1348.46  -294.03   -57.28   208.59  2329.78
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)              156.71      58.07   2.699  0.00698 **
## TEMPERATURE             2399.74     261.66   9.171  < 2e-16 ***
## HUMIDITY                -918.38     126.79  -7.243  4.9e-13 ***
## WIND_SPEED               404.47      48.16   8.399  < 2e-16 ***
## VISIBILITY                12.56      24.86   0.505  0.61351
## DEW_POINT_TEMPERATURE   -316.92     278.83  -1.137  0.25575
## SOLAR_RADIATION         -444.85      34.69 -12.824  < 2e-16 ***
## RAINFALL               -1764.01     182.65  -9.658  < 2e-16 ***
## SNOWFALL                 317.78     131.58   2.415  0.01576 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 487.3 on 6339 degrees of freedom
## Multiple R-squared:  0.4303, Adjusted R-squared:  0.4296
## F-statistic: 598.5 on 8 and 6339 DF,  p-value: < 2.2e-16
```

```
#Task: Build a linear regression model using all variables
lm_model_all <- lm(RENTED_BIKE_COUNT ~ .,
                       data=bike_train)
summary(lm_model_all)
```

```
##
## Call:
## lm(formula = RENTED_BIKE_COUNT ~ ., data = bike_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1401.45  -218.96    -7.31   199.53  1780.67
##
## Coefficients: (3 not defined because of singularities)
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              316.008     52.341   6.037 1.65e-09 ***
## TEMPERATURE              782.658    212.129   3.690 0.000227 ***
```

```
## HUMIDITY                -886.730     99.492  -8.913  < 2e-16 ***
## WIND_SPEED                31.913     40.275   0.792 0.428169
## VISIBILITY                21.872     20.262   1.079 0.280439
## DEW_POINT_TEMPERATURE    598.387    221.369   2.703 0.006888 **
## SOLAR_RADIATION          276.882     41.466   6.677 2.64e-11 ***
## RAINFALL               -2064.638    143.276 -14.410  < 2e-16 ***
## SNOWFALL                 260.973    103.498   2.522 0.011709 *
## HOUR_0                  -133.107     33.323  -3.994 6.56e-05 ***
## HOUR_1                  -220.655     32.838  -6.719 1.98e-11 ***
## HOUR_2                  -341.020     32.910 -10.362  < 2e-16 ***
## HOUR_3                  -423.680     33.498 -12.648  < 2e-16 ***
## HOUR_4                  -490.101     33.297 -14.719  < 2e-16 ***
## HOUR_5                  -466.528     32.826 -14.212  < 2e-16 ***
## HOUR_6                  -307.927     32.990  -9.334  < 2e-16 ***
## HOUR_7                     2.949     33.207   0.089 0.929246
## HOUR_8                   347.169     32.967  10.531  < 2e-16 ***
## HOUR_9                  -103.808     33.853  -3.066 0.002175 **
## HOUR_10                 -341.327     35.106  -9.723  < 2e-16 ***
## HOUR_11                 -351.192     36.879  -9.523  < 2e-16 ***
## HOUR_12                 -312.150     37.820  -8.253  < 2e-16 ***
## HOUR_13                 -295.163     38.411  -7.684 1.77e-14 ***
## HOUR_14                 -296.250     37.268  -7.949 2.21e-15 ***
## HOUR_15                 -213.542     36.764  -5.808 6.61e-09 ***
## HOUR_16                  -80.680     35.369  -2.281 0.022575 *
## HOUR_17                  201.739     34.547   5.839 5.50e-09 ***
## HOUR_18                  690.995     33.487  20.634  < 2e-16 ***
## HOUR_19                  419.180     33.099  12.664  < 2e-16 ***
## HOUR_20                  328.187     32.827   9.997  < 2e-16 ***
## HOUR_21                  342.772     32.918  10.413  < 2e-16 ***
## HOUR_22                  238.833     32.723   7.299 3.26e-13 ***
## HOUR_23                       NA         NA      NA       NA
## HOLIDAY                 -124.424     22.948  -5.422 6.11e-08 ***
## NO_HOLIDAY                    NA         NA      NA       NA
## AUTUMN                   358.999     20.290  17.694  < 2e-16 ***
## SPRING                   191.365     19.362   9.884  < 2e-16 ***
## SUMMER                   198.142     29.187   6.789 1.24e-11 ***
## WINTER                        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 377.9 on 6312 degrees of freedom
## Multiple R-squared:  0.6589, Adjusted R-squared:  0.657
## F-statistic: 348.4 on 35 and 6312 DF,  p-value: < 2.2e-16
```

We use R-square (rsq) and Root Mean Square to evaluate and identify the most important varibales

```
# Use model to make prediction
lm_model_weather_pred <- predict(lm_model_weather, newdata = bike_test)
test_results_weather <- data.frame(PREDICTION=lm_model_weather_pred, TRUTH = bike_tes
t$RENTED_BIKE_COUNT)

lm_model_all_pred <- predict(lm_model_all, newdata = bike_test)
```

```
## Warning in predict.lm(lm_model_all, newdata = bike_test): prediction from a
## rank-deficient fit may be misleading
```

```
test_results_all <- data.frame(PREDICTION = lm_model_all_pred, TRUTH = bike_test$RENT
ED_BIKE_COUNT)

summary(lm_model_weather)$r.squared #0.4303
```

```
## [1] 0.4302915
```

```
summary(lm_model_all)$r.squared #0.6589
```

```
## [1] 0.6589113
```

```
rmse_weather <- sqrt(mean((test_results_weather$TRUTH-test_results_weather$PREDICTION
)^2))
rmse_all <- sqrt(mean((test_results_all$TRUTH-test_results_all$PREDICTION)^2))

print(rmse_weather) #474.6247
```

```
## [1] 474.6247
```

```
print(rmse_all) #361.9543
```

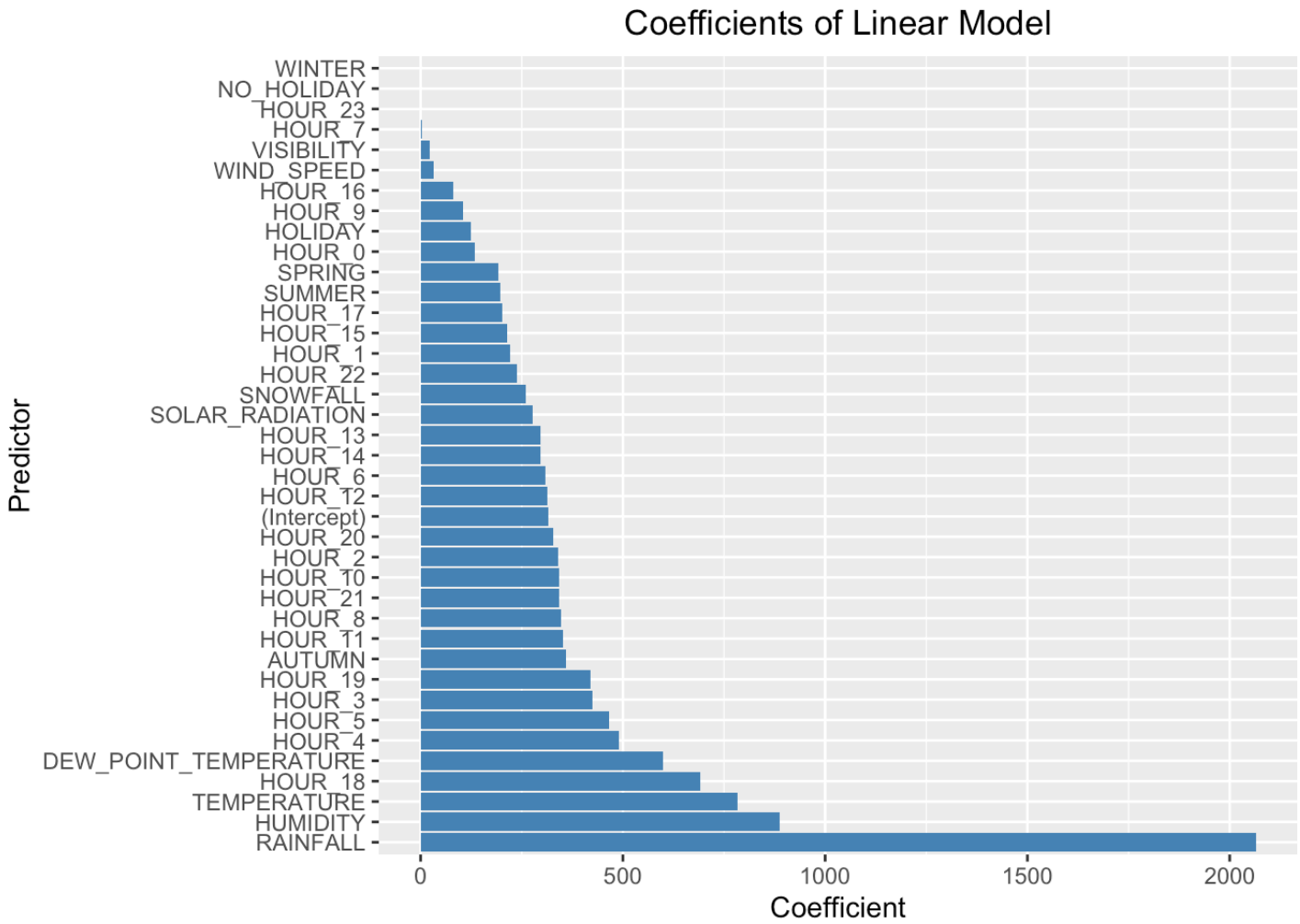
```
## [1] 364.4235
```

Plotting the coeficients in a bar chart

```
# create a data frame of coefficients
coef_df <- tidy(lm_model_all)
```

```
# plot the coefficients in a bar chart (coef plot.png)
ggplot(coef_df, aes(x = reorder(term, desc(abs(estimate))), y = abs(estimate))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  xlab("Predictor") +
  ylab("Coefficient") +
  ggtitle("Coefficients of Linear Model") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
```



The prediction from model using all variables may be misleading because there is colinearity in the predictor variables. This issue can be solved using glmnet model, polynomials and interaction terms

Improve the Linear model

```
#load the packages:
pacman::p_load(tidymodels, tidyverse, stringr)
```

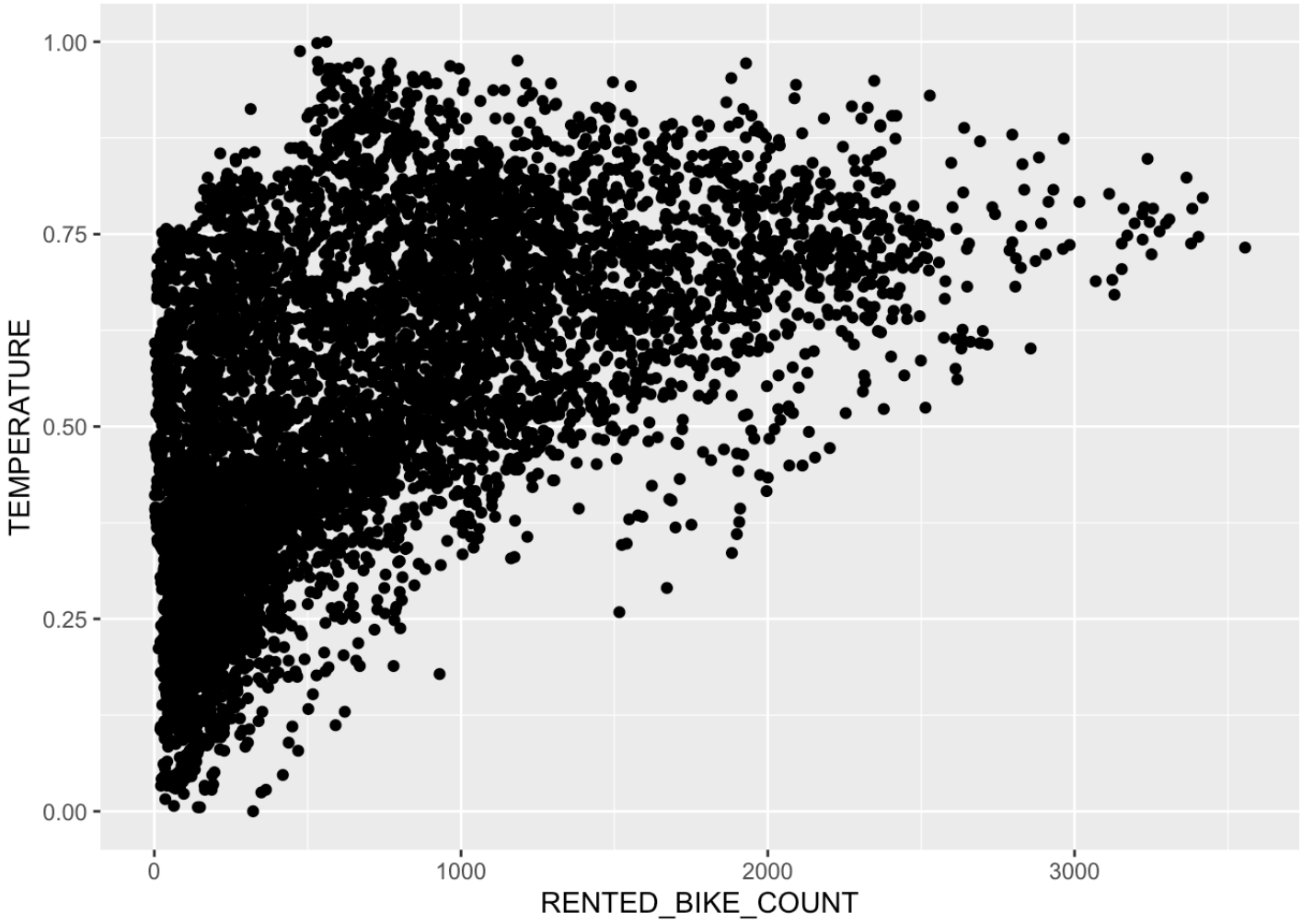
```
#define a linear regression model specification.
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
lm_spec
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

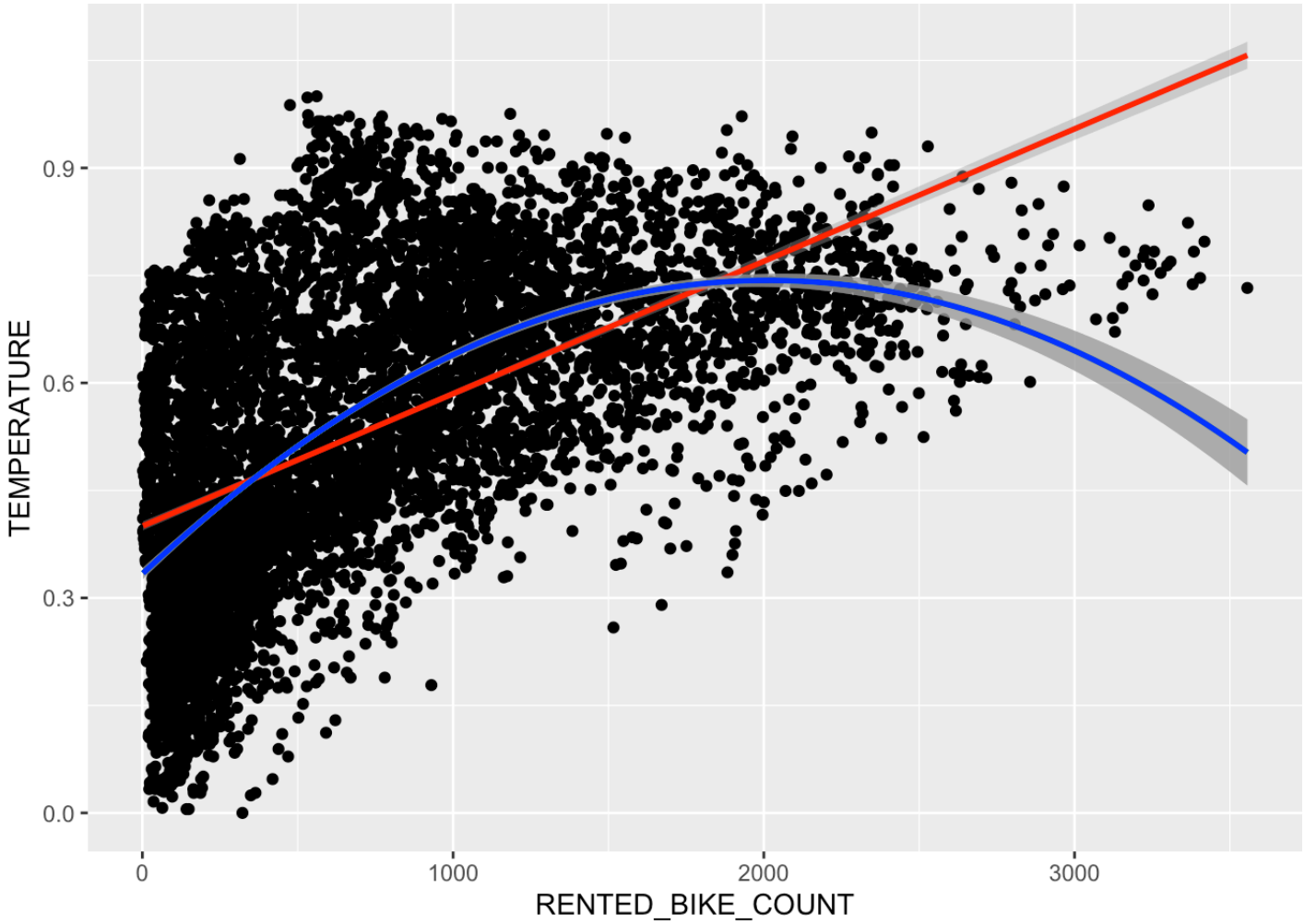
```
#split training and test data
set.seed(1234)
data.split <- initial_split(bike_sharing_df, prop=4/5)
bike_train <- training(data.split)
bike_test <- testing(data.split)
```

First, adding polynomial terms

```
#(poly1.png)
ggplot(data=bike_train, aes(RENTED_BIKE_COUNT, TEMPERATURE)) +
  geom_point() #nonlinearity -> polynomial regression
```




```
# (poly2.png)
ggplot(data=bike_train, aes(RENTED_BIKE_COUNT, TEMPERATURE)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y~x, color="red") +
  geom_smooth(method="lm", formula = y~poly(x,2), color="yellow") +
  geom_smooth(method="lm", formula = y~poly(x,2), color="green") +
  geom_smooth(method="lm", formula = y~poly(x,2), color="blue")
```




```
# Fit a linear model with higher order polynomial on some important variables
lm_poly <- lm(RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) +
                poly(HUMIDITY, 4)+
                poly(RAINFALL,2), data = bike_train)
summary(lm_poly$fit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -713.1   350.2   758.6   734.9  1136.4  1457.4
```

```
lm_poly_pred <- predict(lm_poly, newdata = bike_test) #predict
test_results_poly = data.frame(PREDICTION = lm_poly_pred, TRUTH = bike_test$RENTED_BI
KE_COUNT) #create df for test results

#convert all negative prediction to 0 (RENTED_BIKE_COUNT can't be negative)
test_results_poly <- test_results_poly %>%
  mutate(PREDICTION = ifelse(PREDICTION <0, 0, PREDICTION))

#calculate R_squared and RMSE (better than lm_weather but worse than lm_all)
summary(lm_poly)$r.squared #0.4861
```

```
## [1] 0.4861033
```

```
rmse_poly <- sqrt(mean ( (test_results_poly$TRUTH - test_results_poly$PREDICTION)^2)
)
rmse_poly #451.7091
```

```
## [1] 451.7091
```

The effect of predictor variable TEMPERATURE on RENTED_BIKE_COUNT may also depend on other variables such as HUMIDITY, RAINFALL, or both (they interact) and the effect of SEASON on RENTED_BIKE_COUNT may also depend on HOLIDAY, HOUR, or both.

```
#Task: Add Interaction Terms
lm_poly_interaction <- lm(RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY, 4
)+poly(RAINFALL,2)+
                          RAINFALL*HUMIDITY + TEMPERATURE*HUMIDITY,
                          data = bike_train)
summary(lm_poly_interaction)
```

```
##
## Call:
## lm(formula = RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY,
##     4) + poly(RAINFALL, 2) + RAINFALL * HUMIDITY + TEMPERATURE *
##     HUMIDITY, data = bike_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1315.33  -254.96   -65.41   171.27  2220.84
##
## Coefficients: (3 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1503.3       61.3  24.522  < 2e-16 ***
## poly(TEMPERATURE, 6)1  58400.4     1574.8  37.085  < 2e-16 ***
## poly(TEMPERATURE, 6)2  -5101.4      480.5 -10.618  < 2e-16 ***
## poly(TEMPERATURE, 6)3 -12619.0      486.6 -25.930  < 2e-16 ***
## poly(TEMPERATURE, 6)4  -4205.2      460.9  -9.124  < 2e-16 ***
## poly(TEMPERATURE, 6)5   -710.4      456.6  -1.556 0.119806
## poly(TEMPERATURE, 6)6    388.2      458.2   0.847 0.396934
## poly(HUMIDITY, 4)1      8108.6     1538.5   5.270 1.40e-07 ***
## poly(HUMIDITY, 4)2     -7946.3      497.4 -15.977  < 2e-16 ***
## poly(HUMIDITY, 4)3       367.7      483.1   0.761 0.446703
## poly(HUMIDITY, 4)4     -2632.2      477.2  -5.516 3.60e-08 ***
## poly(RAINFALL, 2)1    -87183.4    22252.0  -3.918 9.02e-05 ***
## poly(RAINFALL, 2)2      1059.9      532.5   1.990 0.046592 *
## RAINFALL                    NA         NA      NA       NA
## HUMIDITY                    NA         NA      NA       NA
## TEMPERATURE                 NA         NA      NA       NA
## RAINFALL:HUMIDITY      30974.2     8112.5   3.818 0.000136 ***
## HUMIDITY:TEMPERATURE   -2773.9      160.2 -17.314  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 452.3 on 6757 degrees of freedom
## Multiple R-squared:  0.5086, Adjusted R-squared:  0.5076
## F-statistic: 499.5 on 14 and 6757 DF,  p-value: < 2.2e-16
```

```
lm_poly_interaction_pred <- predict(lm_poly_interaction, newdata = bike_test)
```

```
## Warning in predict.lm(lm_poly_interaction, newdata = bike_test): prediction
## from a rank-deficient fit may be misleading
```

```
test_results_poly_interaction <- data.frame(PREDICTION = lm_poly_interaction_pred, TR
UTH=bike_test$RENTED_BIKE_COUNT)

#model performance (improved model)
summary(lm_poly_interaction)$r.squared #0.5086
```

```
## [1] 0.5085865
```

```
rmse_poly_interaction <- rmse(test_results_poly_interaction, TRUTH, PREDICTION )
rmse_poly_interaction #442
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        442.
```

Adding regularization to overcome the issue of complicated, difficult and overfitting. We can use a more advanced and generalized glmnet engine. It provides a generalized linear model with Lasso, Ridge, and Elastic Net regularizations.

```
pacman::p_load(glmnet, yardstick)
```

Creating the model prediction function and model evaluation function

```
#prediction function
model_prediction <- function(lm_model, test_data) {
  results <- lm_model %>%
    predict(new_data=test_data) %>%
    mutate(TRUTH=test_data$RENTED_BIKE_COUNT)
  results[results<0] <-0
  return(results)
}

#model evaluation function
model_evaluation <- function(results) {
  rmse = rmse(results, truth=TRUTH, estimate=.pred)
  rsq = rsq(results, truth=TRUTH, estimate=.pred)
  print(rmse)
  print(rsq)
}
```

```
#Use grid to define the best penalty (lambda)
penalty_value <- 10^seq(-4,4, by = 0.5) #penalty values ranging from 10^-4 to 10^4
x = as.matrix(bike_train[,-1]) #define a matrix for CV
y= bike_train$RENTED_BIKE_COUNT
```

We can use cross-validation to define the lambda with 10-fold validation

```
cv_ridge <- cv.glmnet(x,y, alpha = 0, lambda = penalty_value, nfolds = 10)
cv_lasso <- cv.glmnet(x,y, alpha = 1, lambda = penalty_value, nfolds = 10)
cv_elasticnet <- cv.glmnet(x,y, alpha = 0.5, lambda = penalty_value, nfolds = 10)
```

```
#glmnet spec (using CV above, best optimal is 0.3 and 0.5)
glmnet_spec <- linear_reg(penalty = 0.3, mixture=0.5) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
```

The suggested performance requirements for the best model includes: The RMSE should be less than 330 (rougly 10% of the max value in test dataset) R-squared should be greater than 0.72

```
#Fit the model (best model)
glmnet_best <- glmnet_spec %>%
  fit(RENTED_BIKE_COUNT ~ RAINFALL*HUMIDITY*TEMPERATURE + SPRING*SUMMER*HOLIDAY*HOUR_
18* HOUR_19* HOUR_8* HOUR_21* HOUR_20* HOUR_4 +
        poly(RAINFALL, 8) + poly(HUMIDITY, 5) +  poly(TEMPERATURE, 5) + poly(DEW_POIN
T_TEMPERATURE, 5) + poly(SOLAR_RADIATION, 5) + poly(SNOWFALL,5) +
        SPRING + SUMMER  + HOLIDAY + WIND_SPEED + VISIBILITY +
        HOUR_18+ HOUR_4 + HOUR_5 + HOUR_3 + HOUR_19 + HOUR_11 + HOUR_8 + HOUR_21 + HO
UR_10 + HOUR_2 + HOUR_20,
      data = bike_train)

glmnet_best_pred <- model_prediction(glmnet_best, bike_test)
model_evaluation(glmnet_best_pred) #rsq = 0.783, rmse = 296
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        315.
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.753
```

```
glmnet_best_rsq = rsq(glmnet_best_pred, truth = TRUTH, estimate = .pred)
glmnet_best_rmse = rmse(glmnet_best_pred, truth = TRUTH, estimate = .pred)
```

```
# Fit the model (with top 10 coeficients)
glmnet_top10 <- glmnet_spec %>%
  fit(RENTED_BIKE_COUNT ~ RAINFALL*HUMIDITY*TEMPERATURE + SPRING*SUMMER + SUMMER +
        poly(RAINFALL, 6) + poly(HUMIDITY, 5) +  poly(TEMPERATURE, 5) + poly(DEW_POIN
T_TEMPERATURE,5) +
        HOUR_18 + HOUR_4 + HOUR_5 +HOUR_3,
      data=bike_train
        )
glmnet_top10_pred <- model_prediction(glmnet_top10, bike_test)
model_evaluation(glmnet_top10_pred) #rsq = 0.640, rmse = 381 (not good)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        385.
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.631
```

```
glmnet_top10_rsq = rsq(glmnet_top10_pred, truth = TRUTH, estimate = .pred)
glmnet_top10_rmse = rmse(glmnet_top10_pred, truth = TRUTH, estimate = .pred)
```

```
# Fit Ridge Regression
glmnet_ridge <- glmnet(x,y, alpha=0)
glmnet_ridge_pred <- predict(glmnet_ridge, s=cv_ridge$lambda.min,
                             newx = as.matrix(bike_test[,-1]))

ridge_rmse = sqrt(mean( (bike_test[,1] - glmnet_ridge_pred)^2))
ridge_rmse #365.06
```

```
## [1] 365.0562
```

```
ridge_mse = mean( (bike_test[,1] - glmnet_ridge_pred)^2)
ridge_rsq = 1 - ridge_mse / var(bike_test[,1])
ridge_rsq #0.667
```

```
## [1] 0.6674863
```

```
# Fit Lasso
glmnet_lasso <- glmnet(x,y,alpha=1)
glm_lasso_pred <- predict(glmnet_lasso, s=cv_lasso$lambda.min,
                          newx=as.matrix(bike_test[,-1]))

lasso_rmse = sqrt(mean( (bike_test[,1] - glm_lasso_pred)^2))
lasso_rmse #364.0492
```

```
## [1] 364.0492
```

```
lasso_mse = mean( (bike_test[,1] - glm_lasso_pred)^2)
lasso_rsq = 1 - lasso_mse/var(bike_test[,1])
lasso_rsq #0.6693
```

```
## [1] 0.6693181
```

```
# Fit Elastic Net
glmnet_elasticnet <- glmnet(x,y,alpha=0.7)
glm_elasticnet_pred <- predict(glmnet_elasticnet, s=cv_elasticnet$lambda.min,
                          newx=as.matrix(bike_test[,-1]))

elasticnet_rmse = sqrt(mean( (bike_test[,1] - glm_elasticnet_pred)^2))
elasticnet_rmse #364.0468
```

```
## [1] 364.2295
```

```
elasticnet_mse = mean( (bike_test[,1] - glm_elasticnet_pred)^2)
elasticnet_rsq = 1 - elasticnet_mse/var(bike_test[,1])
elasticnet_rsq #0.6693
```

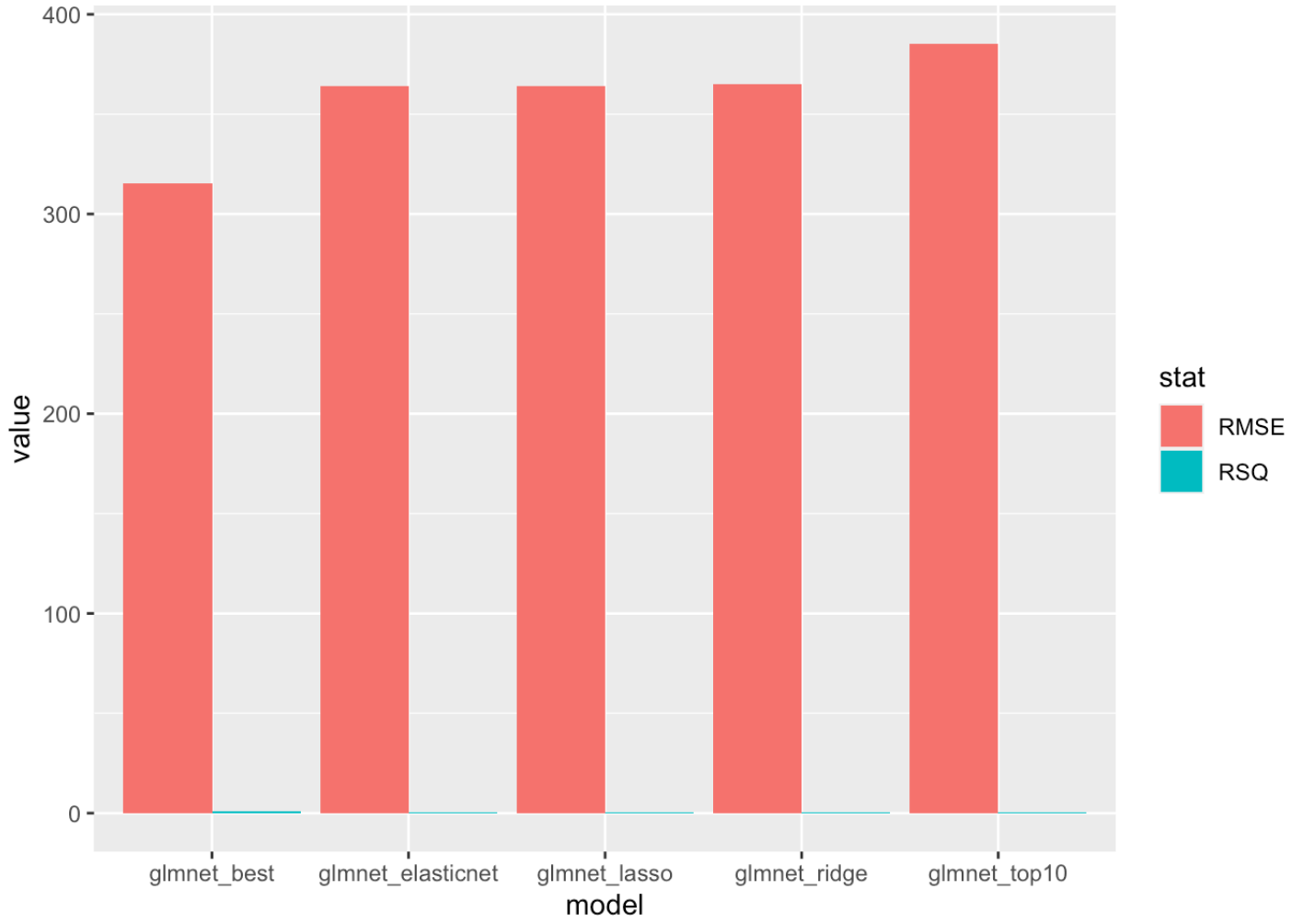
```
## [1] 0.6689906
```

To compare the performance of models built previously, creating a group bar chart for rsq and rmse

```
#Create data frame for group bar chart
model <- c(rep("glmnet_best",2), rep("glmnet_top10",2),
           rep("glmnet_ridge",2), rep("glmnet_lasso",2), rep("glmnet_elasticnet",2))
stat <- rep(c("RSQ", "RMSE"),5)
value <- c(glmnet_best_rsq$.estimate, glmnet_best_rmse$.estimate,
           glmnet_top10_rsq$.estimate, glmnet_top10_rmse$.estimate,
           ridge_rsq, ridge_rmse,
           lasso_rsq,lasso_rmse,
           elasticnet_rsq, elasticnet_rmse)
model_df <-  data.frame(model, stat, value)
print(model_df)
```

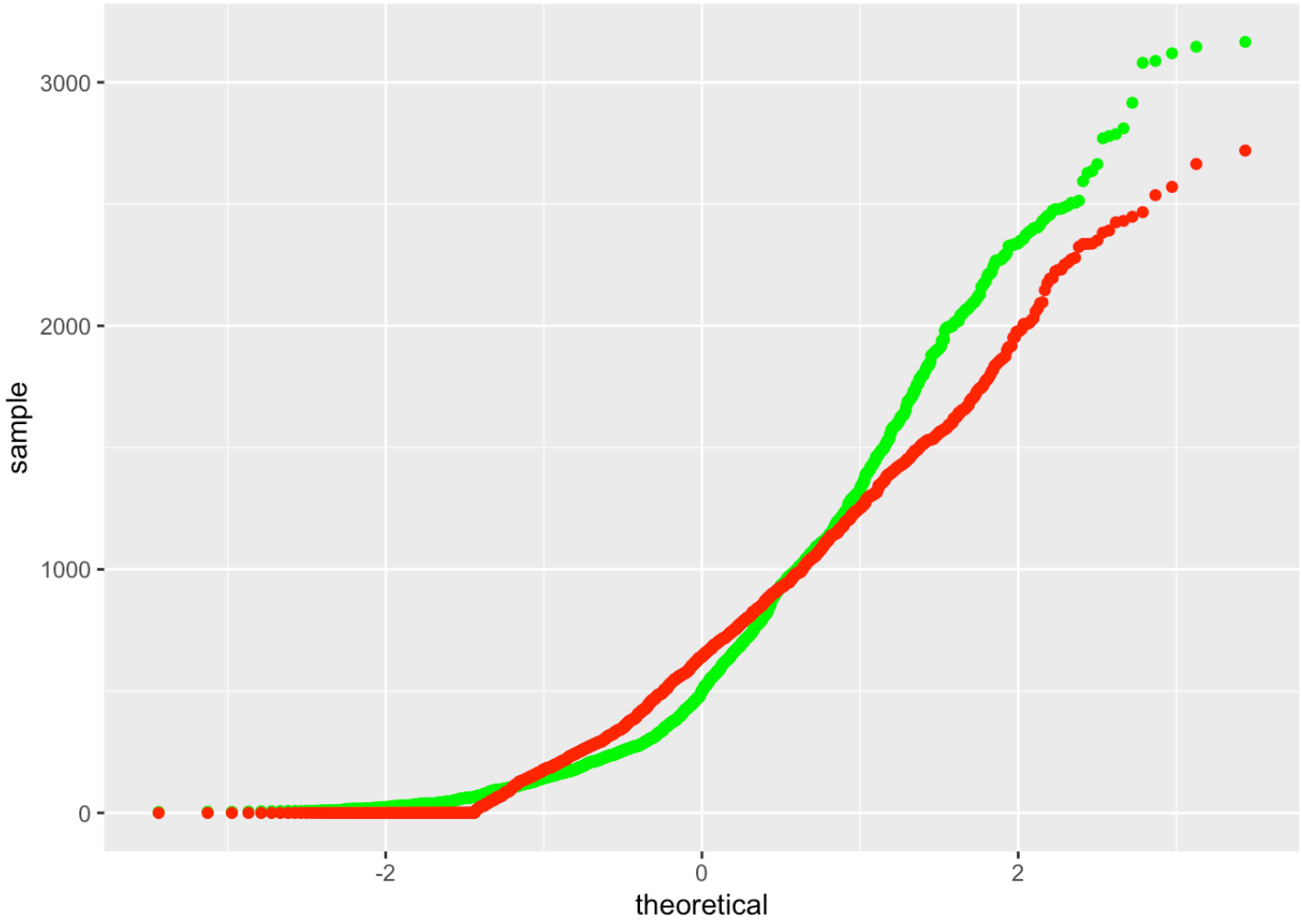
```
##                model stat       value
## 1        glmnet_best  RSQ   0.7534620
## 2        glmnet_best RMSE 315.1054426
## 3       glmnet_top10  RSQ   0.6313216
## 4       glmnet_top10 RMSE 385.3212657
## 5       glmnet_ridge  RSQ   0.6674863
## 6       glmnet_ridge RMSE 365.0561548
## 7       glmnet_lasso  RSQ   0.6693181
## 8       glmnet_lasso RMSE 364.0492378
## 9  glmnet_elasticnet  RSQ   0.6689906
## 10 glmnet_elasticnet RMSE 364.2294665
```

```
# Create group bar chart for rsq and rmse (model evaluation.png)
model_df %>%
  ggplot(aes(fill=stat, x=model, y=value)) +
  geom_bar(position="dodge", stat="identity")
```



For the best model glmnet_best, creating a Q-Q chart by plotting the distribution difference between the predictions generated by your best model and the true values on the test dataset.

```
# Create a Q-Q chart for best model: glmnet_best (Q-Q chart.png)
glmnet_best_pred %>%
  ggplot() +
  stat_qq(aes(sample=TRUTH), color='green') +
  stat_qq(aes(sample=.pred), color='red')
```



Conclusion

In conclusion, the model using top 10 coefficients does not have good performance. While Ridge Regression, Lasso and Elastic Net Regularization perform better than the models using polynomials and interaction terms, it is still not the best model to use.

The number of bike rented depends on multiple variables, including weather, seasons and hours. The best statistical learning model to use for prediction is linear regression with penalty = 0.3, mixture=0.5.