Applied Data Science with R Capstone project

OLUSANJO MICHAEL OYELAKIN

24/02/2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- 4 selected big cities over the worldWeather forecast for next 5 days while using Regression Model
- Created an app to access forecast + prediction
- Make data based decision according peaks
- The demand of bikes have factors that can influence their demand within a city. For example season, renting price, temperature, and hour of the day.

Introduction

- Problem: how to predict bike sharing demand in big cities?
- Requirement:
  - Use only few public available data sets
  - Apply regression model in programming language R
  - Predict number bikes rented each hour based on the weather
  - Use Data Collection and sources
  - Use Data exploration and analysis
  - Use Data Modeling
  - Create Dashboard
  - Derive Conclusion

Methodology

- Perform data collection
- Perform data wrangling
- Perform exploratory data analysis (EDA) using SQL and visualization
- Perform predictive analysis using regression models
  - How to build the baseline model
  - How to improve the baseline model
- Build a R Shiny dashboard app

Methodology

- Used Web Scraping from Wikipedia
- Used Openweather API
- Did data Wrangling
- Performed Data Exploration
- Performed Data Visualization
- Predicted hourly bikes rented using linear regression
- Then refined the process of baseline regression models

Data collection

- The data was available on Wikipedia, I used the R Programing language to scrape the table.

- Extracted the Bike Sharing System HTML table from a wikipage and converted it to a data frame.

- Used OpenWeatherAPI to get a 5-day weather forecast for a list of cities (Seoul, Washington D. C, Paris, Suzhou)

Data wrangling

- Data wrangling was performed using Regular Expressions
- Performed data wrangling using dplyr
- Standardize column names and processed the scraped data into its columns
- Removed all the undesired reference links using regular expressions
- Detected and handled missing values in the data using dplyr
- Normalized the data using dplyr

EDA with SQL

- Most bikes being rented during summer and the least during winter
- I established connection using RSQLite
- Seoul offers highest bike sharing per population compared to other big cities

EDA with data visualization

- Scatter plot was used to visualise correlation between Bike count and Temperature by Seasons
- Created histogram overlay where I deducted the following:
- We can see from the histogram that most of the time there are relatively few bikes rented. Indeed, the 'mode', or most frequent amount of bikes rented, is about 250.
  - Judging by the 'bumps' at about 700, 900, and 1900, and 3200 bikes, it looks like there may be other modes hiding within subgroups of the data.
  - Interestingly, judging from the tail of the distribution, on rare occasions there are many more bikes rented out than usual.
- Although the overall scale of bike rental counts changes with the seasons, key features remain very similar.
  For example, peak demand times are the same across all seasons, at 8 am and 6

Predictive analysis

- Split the data into training and testing data
- Did a model evaluation and identification of important variables
- Used linear regression model to perform variable correlations

Build a R Shiny dashboard

Plots

- Leaflet (world map and cities (incl. Barcelona))
- Temperature forecast
- Bikes rented prediction (cursor with additional info)
- Linear model bikes rented ~ humidity^5

Results

- Exploratory data analysis results
- Predictive analysis results
- A dashboard demo in screenshots

EDA with SQL

- Using SQL we found
- Maximum bike rental amount
- How the popularity of the rental is affected by the temperature
- The season ability of the bikes
- Found the total bike count in Seoul

Busiest bike rental times

- The busiest bike rental times are from the hour of 8 – 23 hour.

	DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WI
0	2017-01-12T00:00:00+08…	254	0	-5.2000	37	
1	2017-01-12T00:00:00+08…	204	1	-5.5000	38	
2	2017-01-12T00:00:00+08…	173	2	-6	39	
3	2017-01-12T00:00:00+08…	107	3	-6.2000	40	
4	2017-01-12T00:00:00+08…	78	4	-6	36	

Hourly popularity and temperature by seasons

	SEASONS	HOUR	AVG_bikes_rented	AVG_temp
1	Summer	18	2135.141	29.41868
2	Autumn	18	1983.333	16.03185
3	Summer	19	1889.250	28.29231
4	Summer	20	1801.924	27.06630
5	Summer	21	1754.065	26.27826
6	Spring	18	1689.311	15.97222
7	Summer	22	1567.870	25.69891
8	Autumn	17	1562.877	17.27778
9	Summer	17	1526.293	30.15444
10	Autumn	19	1515.568	15.06346

- The hour of 18 recorded the highest across the season

Rental Seasonality

- Rental Seasonality

	SEASONS	AVG_Rented	Max_Rented	MIN_Rented	STD_deviation
1	Summer	1034.0734	3556	9	690.0884
2	Autumn	924.1105	3298	2	617.3885
3	Spring	746.2542	3251	2	618.5247
4	Winter	225.5412	937	3	150.3374

- Summer has the most record of bike rented
- The least recorded bike rented is winter

Weather Seasonality

- Weather Seasonality

	SEASONS	AVG_Rented	AVG_TEMPERATURE	AVG_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT_TEMPERATURE
1	Summer	1034.0734	26.587711	64.98143	1.609420	1501.745	18.750136
2	Autumn	924.1105	13.821580	59.04491	1.492101	1558.174	5.150594
3	Spring	746.2542	13.021685	58.75833	1.857778	1240.912	4.091389
4	Winter	225.5412	-2.540463	49.74491	1.922685	1445.987	-12.416667

	AVG_SOLAR_RADIATION	AVG_RAINFALL	AVG_SNOWFALL
1	0.7612545	0.25348732	0.00000000
2	0.5227827	0.11765617	0.06350026
3	0.6803009	0.18694444	0.00000000
4	0.2981806	0.03282407	0.24750000

- The more variation in temperature the more impact on the number of bike rented.

Bike-sharing info in Seoul

- Find the total Bike count and city info for Seoul

	Bike Count	Population
Seoul	20000	1540234

- Seoul offers 20000 bikes for a population of 15 millions

Cities similar to Seoul

- Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system

	CITY	sum(s.BICYCLES)	LAT	LNG	POPULATION
1	Beijing	16000	39.9050	116.3914	19433000
2	Ningbo	15000	29.8750	121.5492	7639000
3	Shanghai	19165	31.1667	121.4667	22120000
4	Weifang	20000	36.7167	119.1000	9373000
5	Zhuzhou	20000	27.8407	113.1469	3855609

- SEOUL has biggest rental bike offer compared to similar big cities (37,500 vs 20,000 at most)
- While Shanghai with more population offers less bike rentals

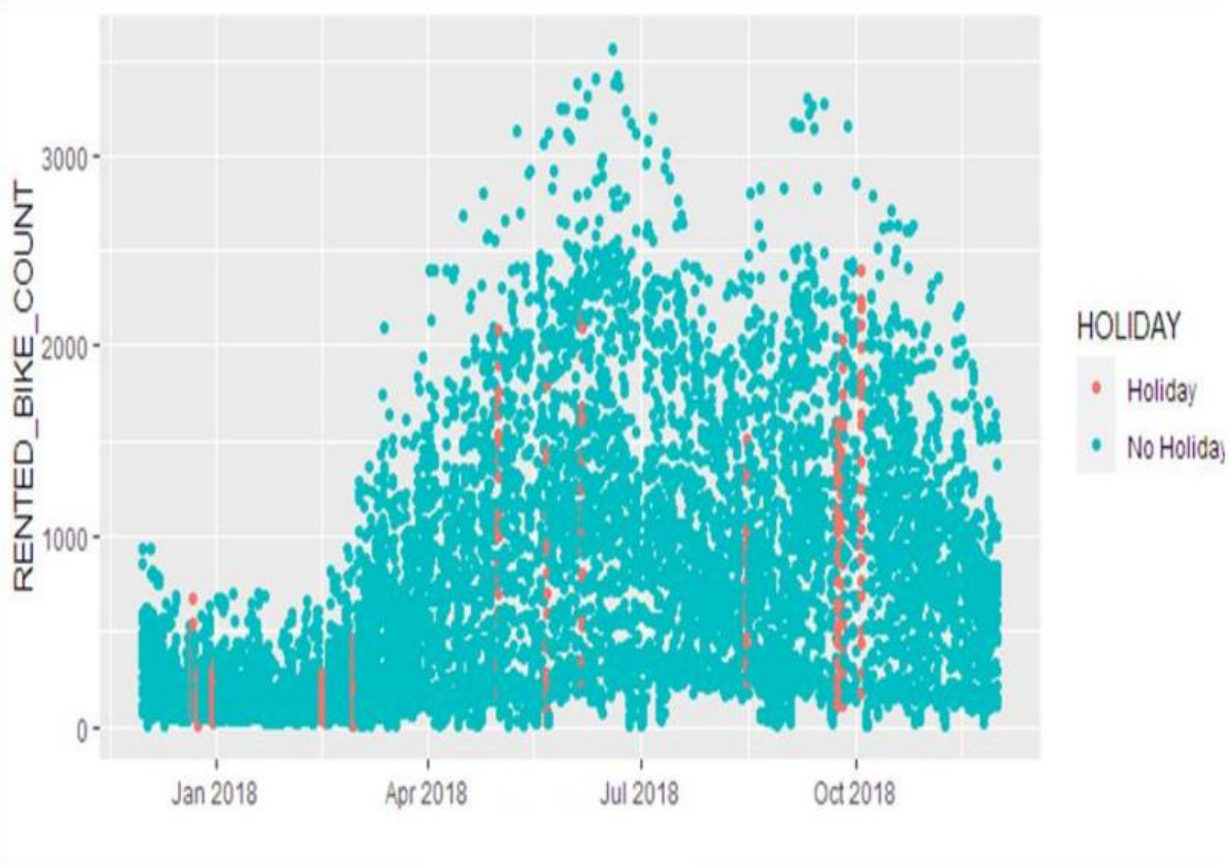
EDA with Visualization

- Charts that were plotted were
- Weather forecast
- Temperature
- The demand for bikes compared to the population of the country

Bike rental vs. Date

Rented bikes during 2018 (holidays highlighted)

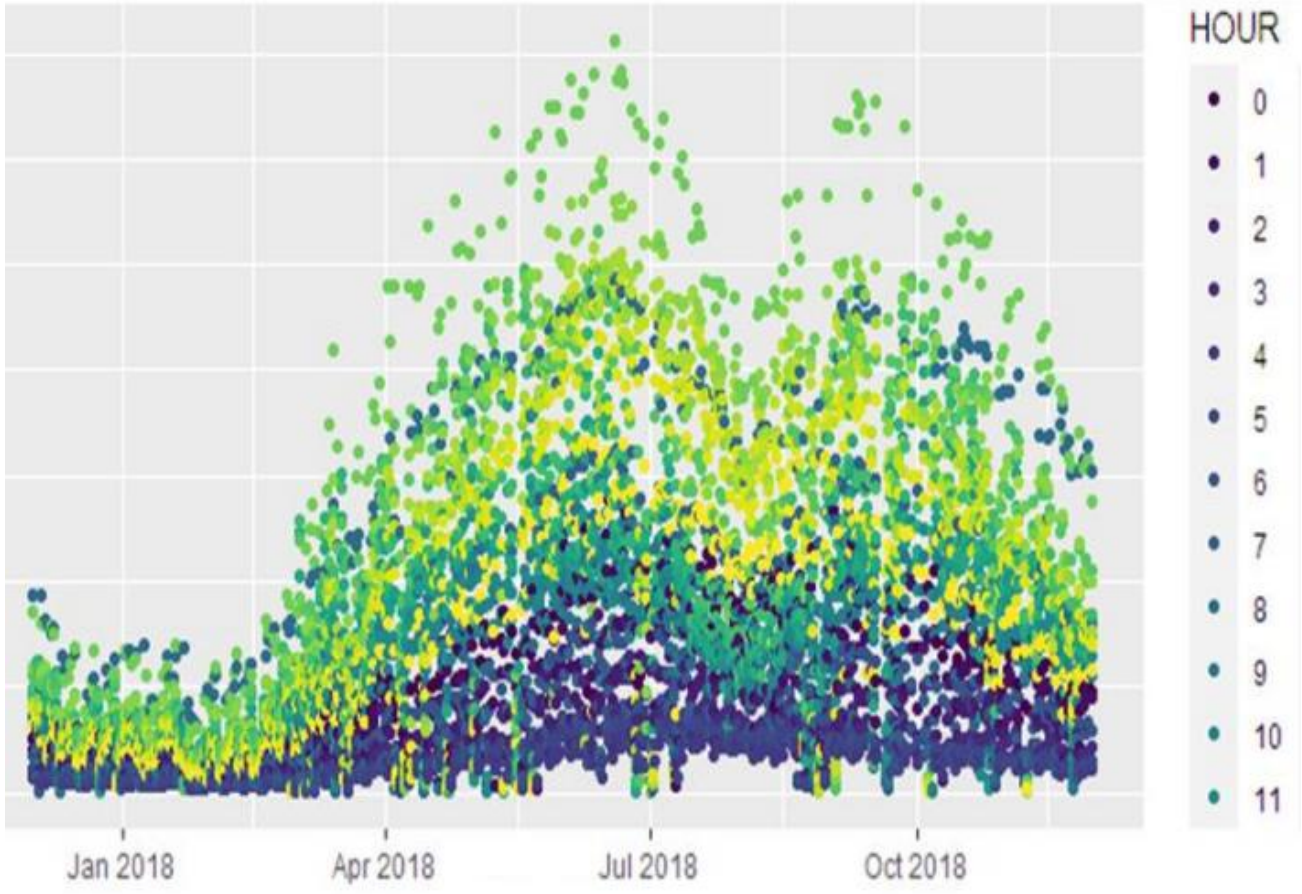
Fewest bike rented during winter, the most during at the peak of Summer



Bike rental vs. Datetime

RENTED_BIKE_COUNT time series (different hours highlighted)

Most rentings during 18th hour, Fewest during night hours(0h-7h)

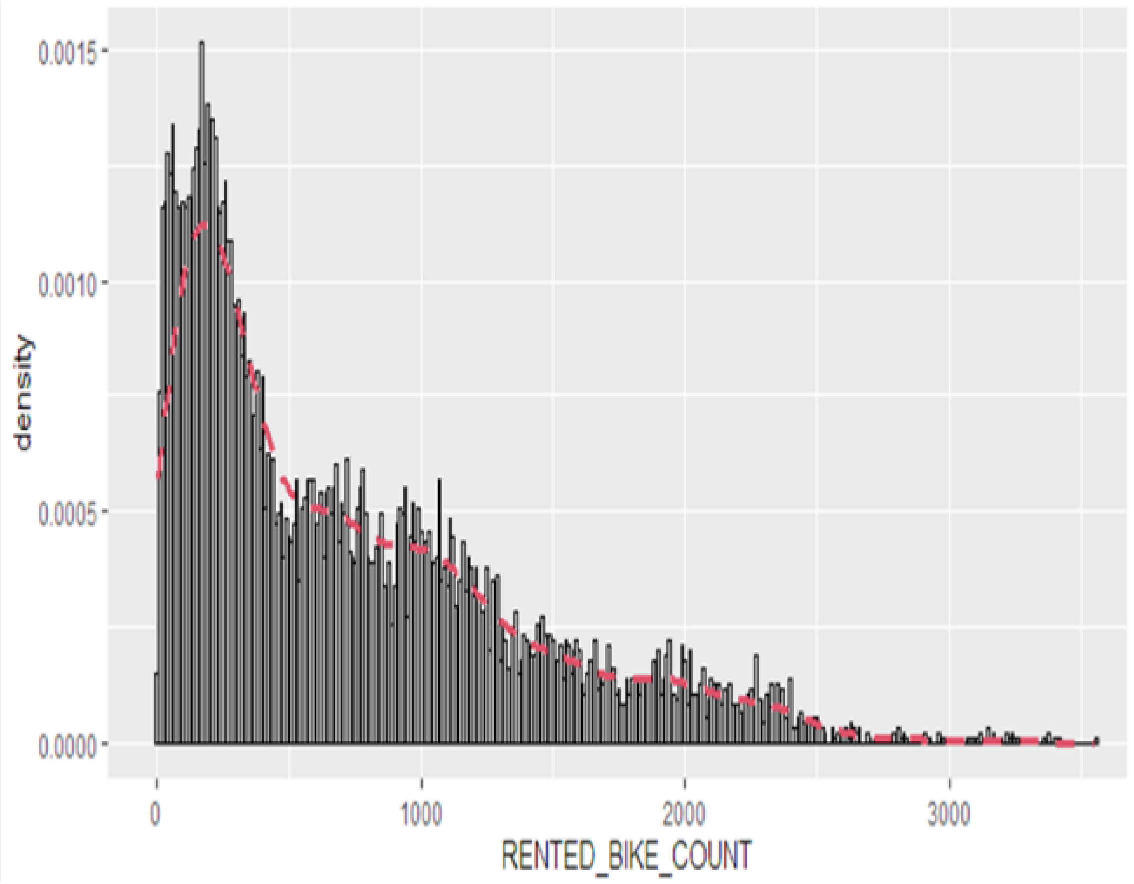


Bike rental histogram

histogram + density curve of rentings

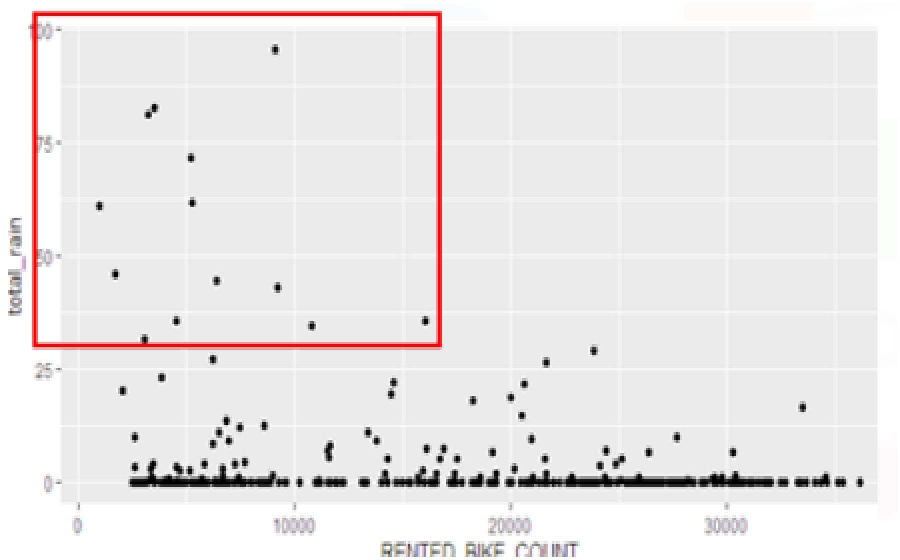
- Renting more than 2500 bikes

- Conclusion
- High peaks occur seldom, and basic demand exists

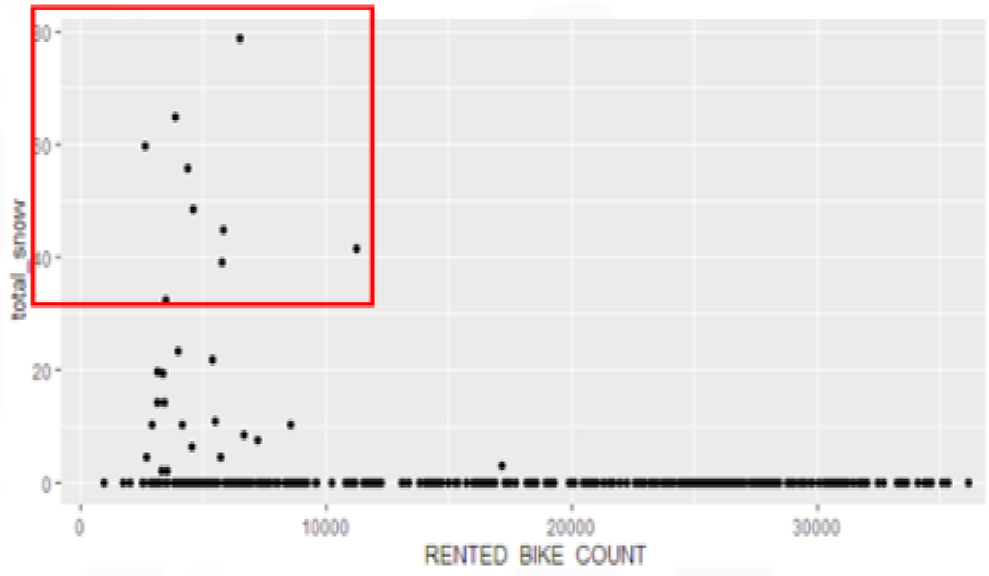


Daily total rainfall and snowfall

Few hours raining (rain level 0-100) when raining, then fewer bikes rented (pattern left upper corner)



snow similar to rain pattern (left upper corner)

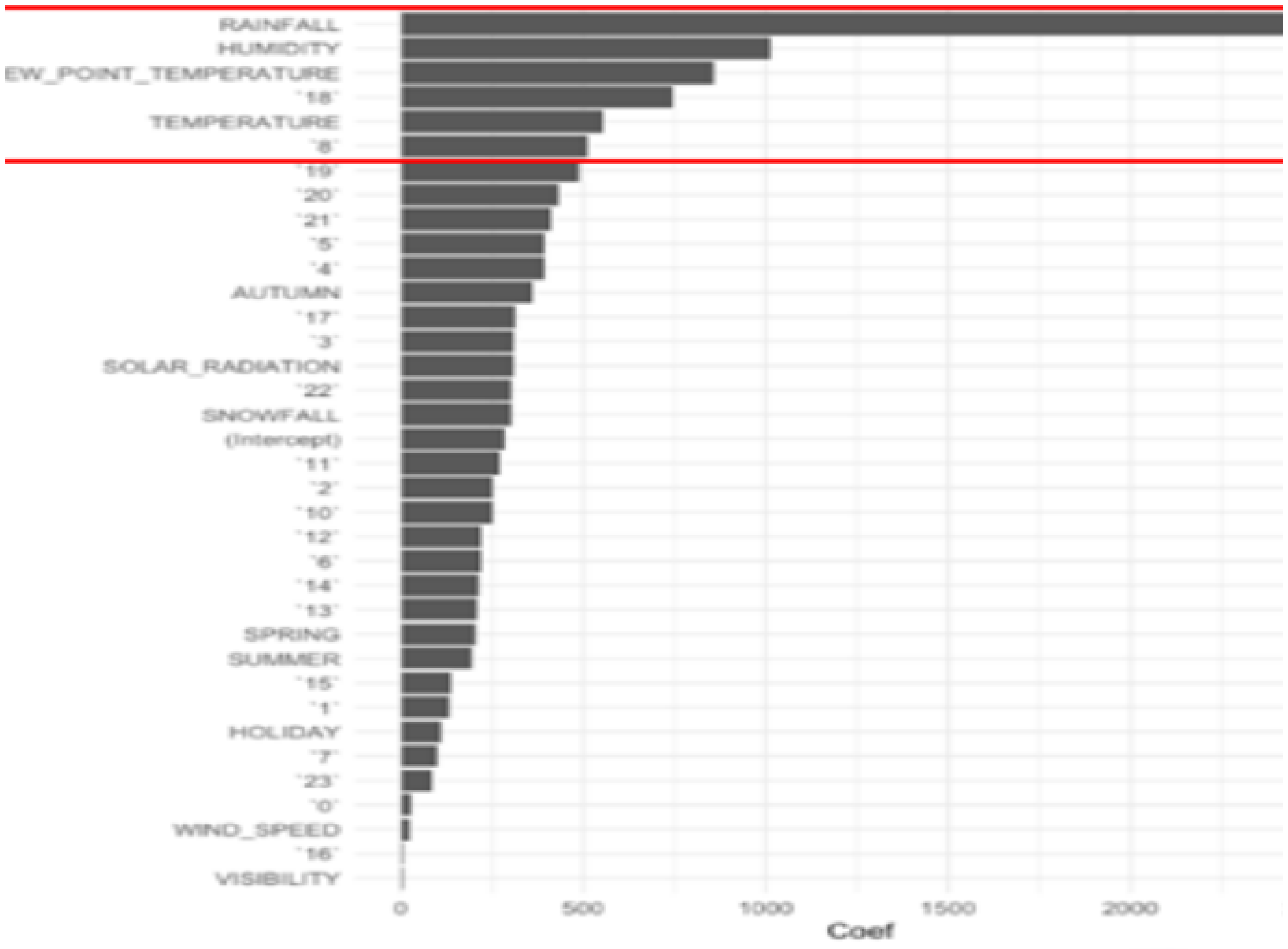


Predictive analysis

Ranked coefficients

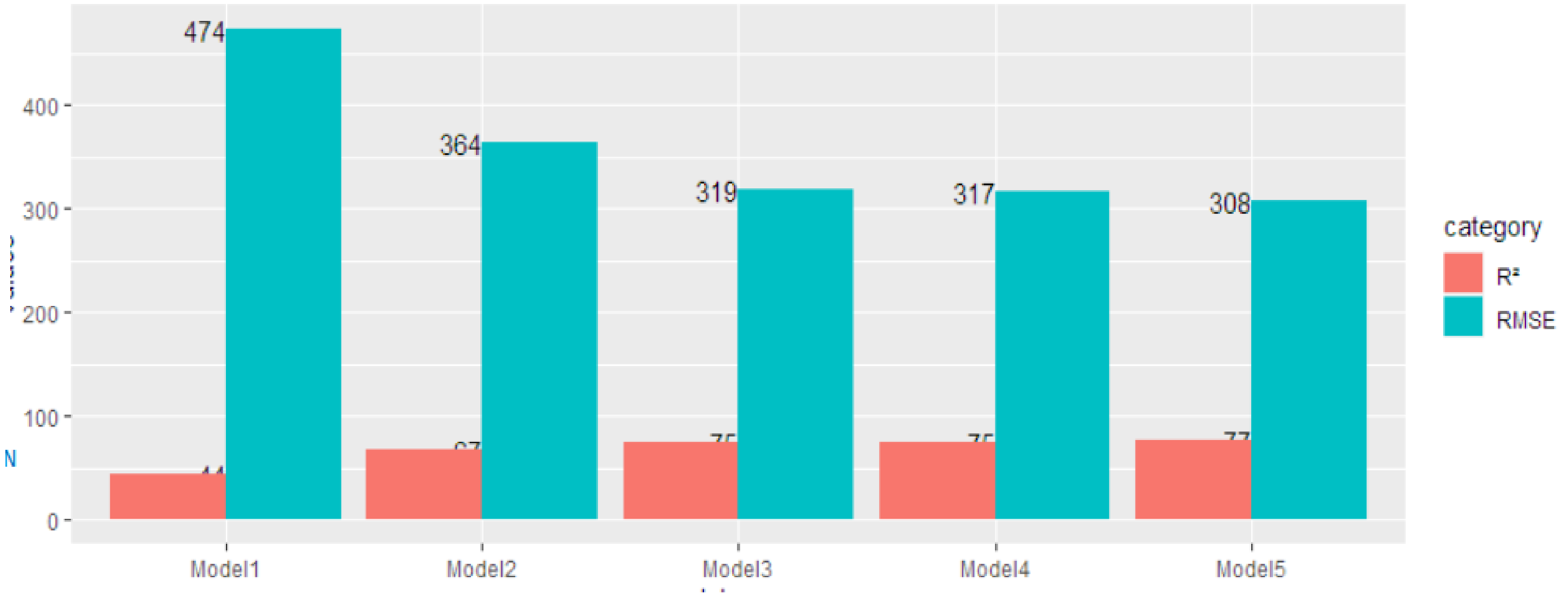
From the chart it shows that Rainfall has the most significant impact on the bike rent while humidity,

Try to tell a story why some variables are important while some are not for predicting bike-sharing demand



Model evaluation

Model1= initial model “weather“ (RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL)



Models from 1 to 5 improving: RMSE down, while R^2 up

Find the best performing model

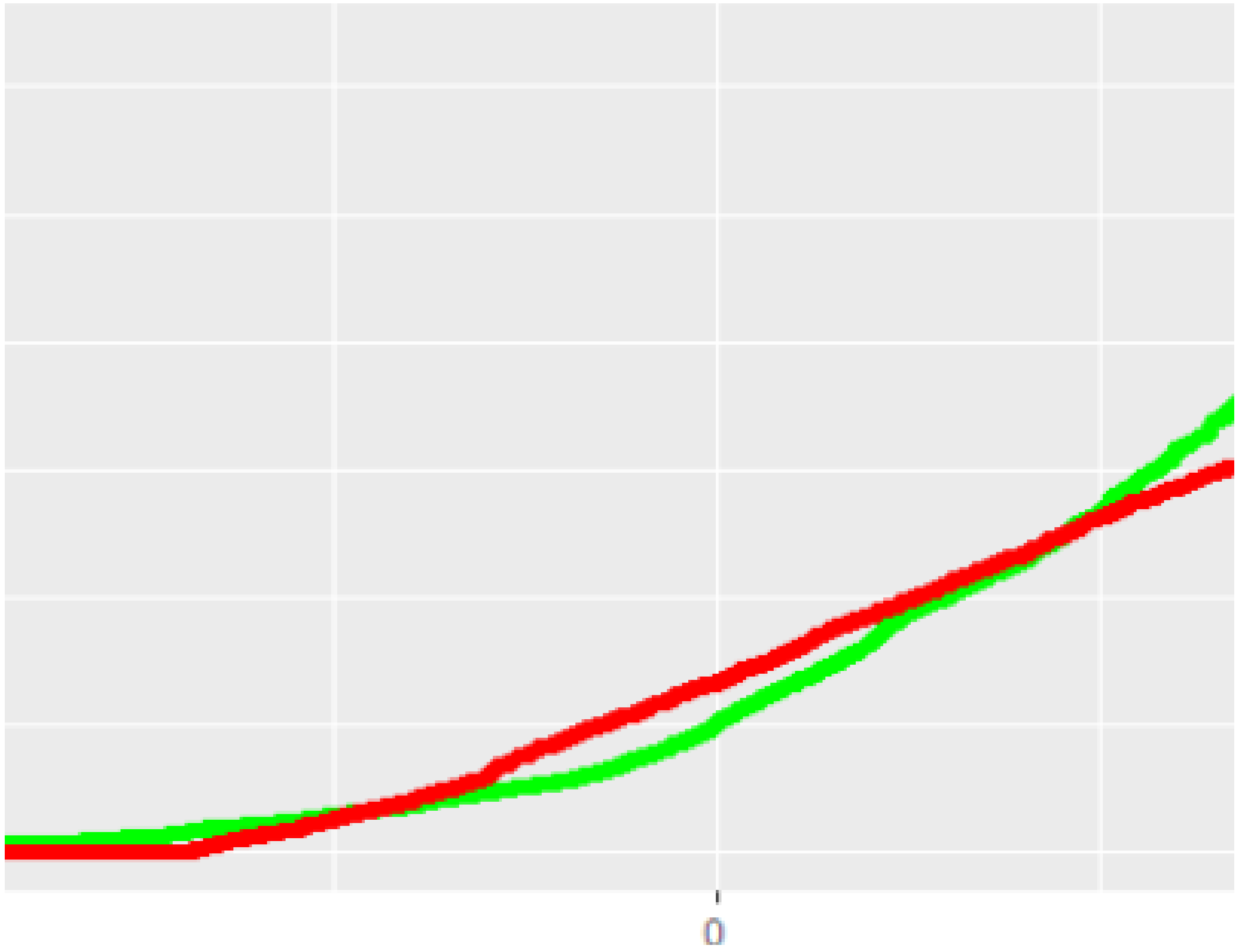
- • Model5 with RMSE 308, R^2 76.5% :

	R²	RMSE	model type
Model1	0.4388245	474.1718	"Basic M"
Model2	0.6697276	363.7369	"All variables"
Model3	0.7476018	318.715	"Ridge"
Model4	0.7497703	316.5961	"ElasticNet"
Model5	0.7651839	308.3724	"Best Modell Lasso"

Q-Q plot of the best model

Test results (predicted) vs truths;

- “truths” values like a curve (tail with 0s at start).
- Conclusion: model is similar to the reality. This means that the prediction can be used since it is not expected to be 100% accurate



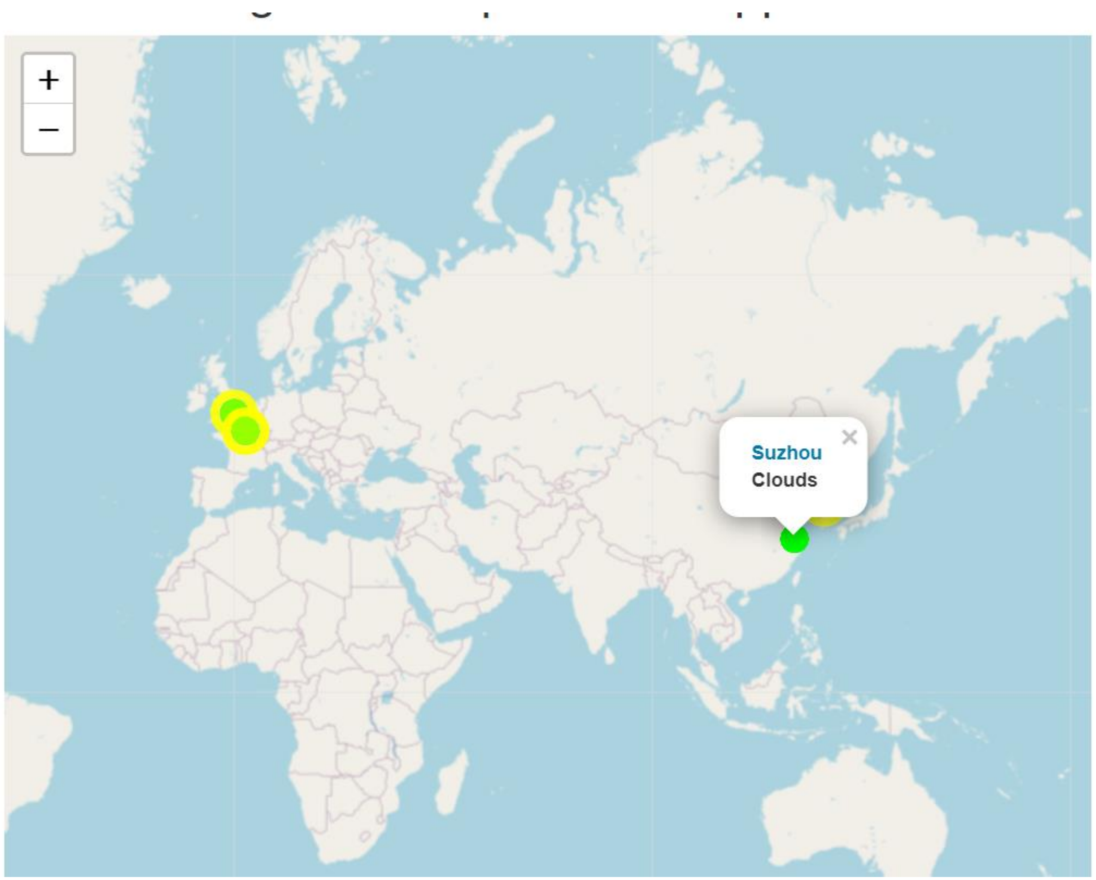
Dashboard

@ https://posit.cloud/content/7667416

*Desktop browser recommended

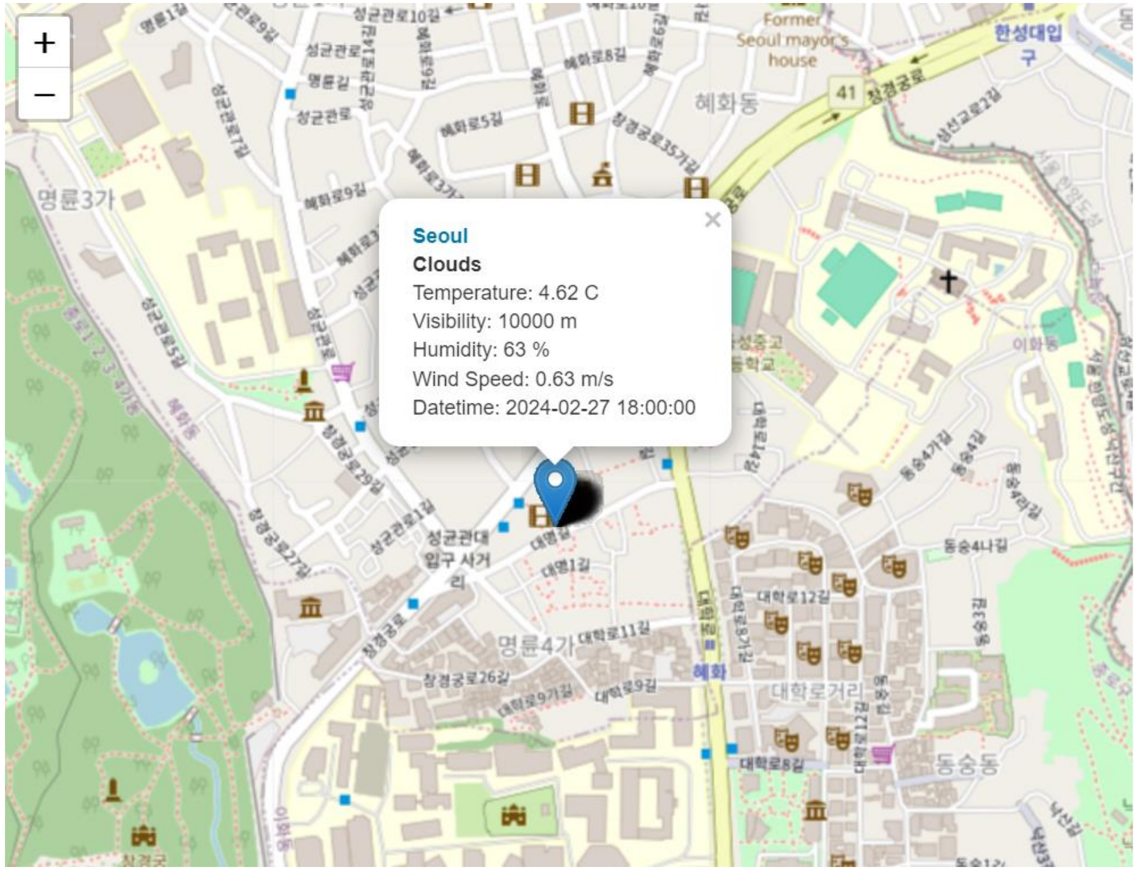
Worldmap: forecast bike renting(cities)

Map showing Suzhou with the least demand bike sharing (green) and other cities with high demand (yellow + green)



Seoul City Selected

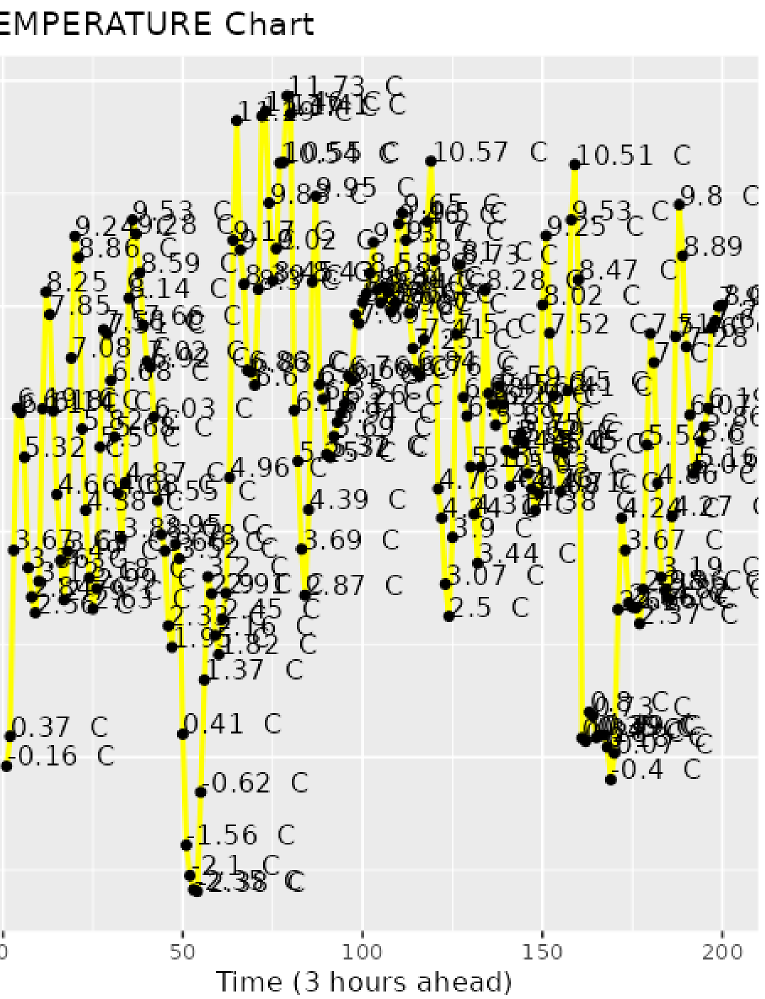
- Seoul City selected during low temperature

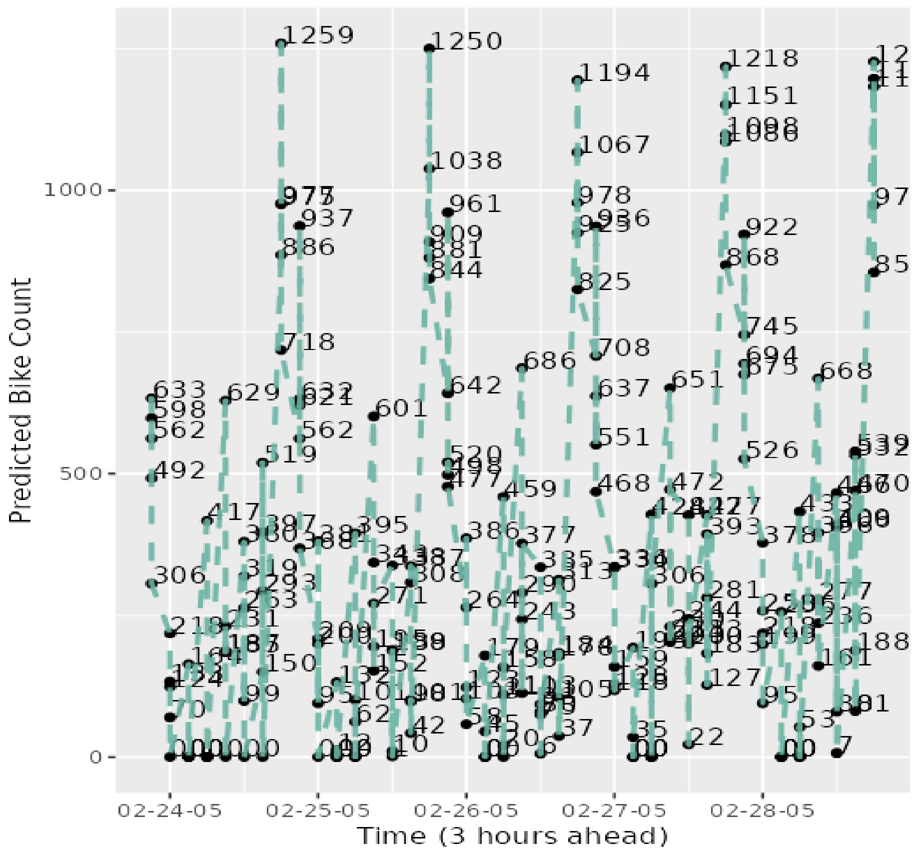


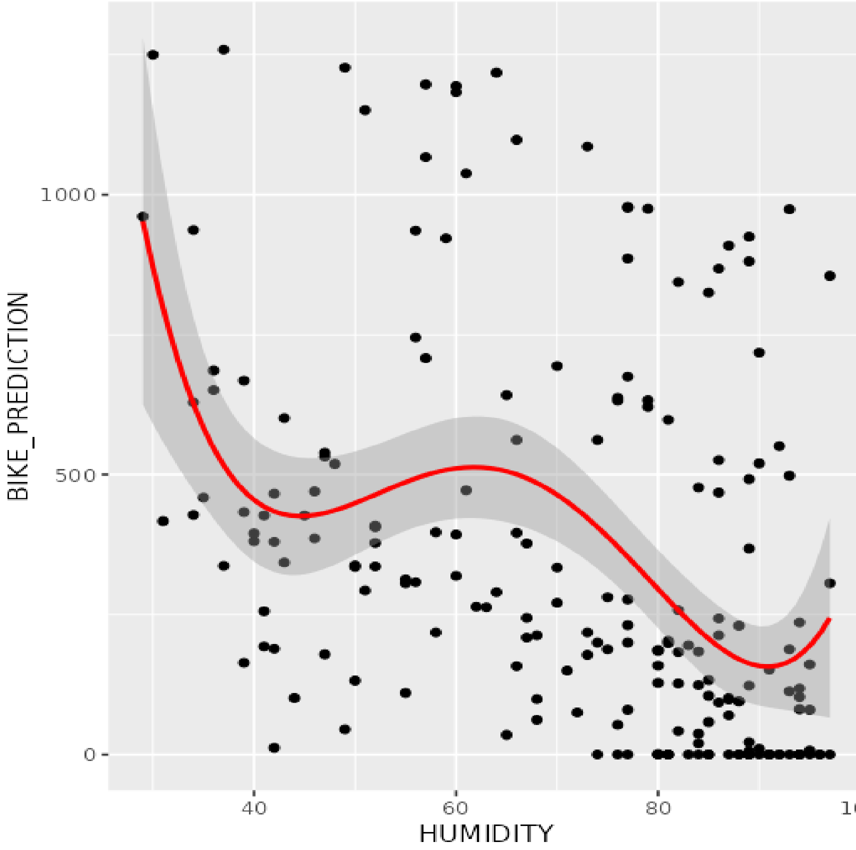
CONCLUSION

- Demand for bikes are influenced by cities, availability of renting bikes, seasons, temperature, hour of the day and holidays
- Linear regression model is recommend to predict the demand of bikes
- Feasible to combine files, API, web scraping

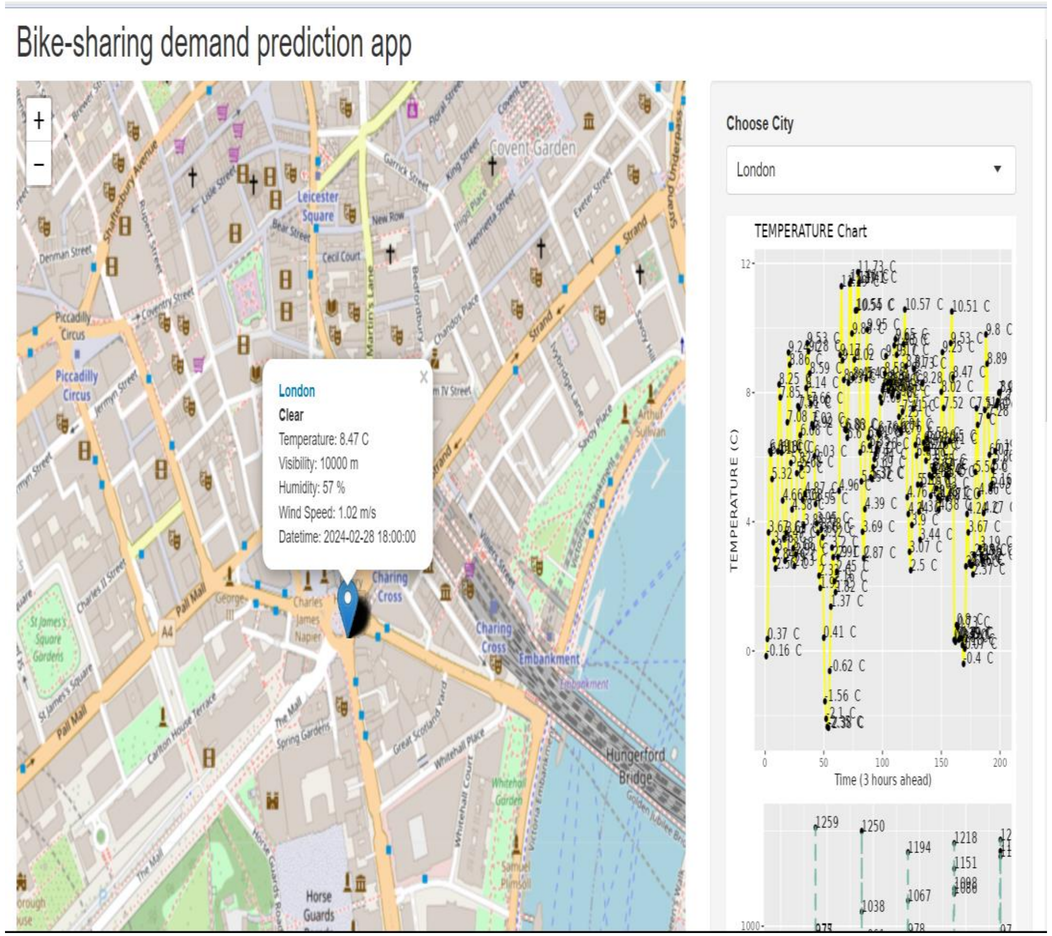
APPENDIX


EMPERATURE Chart
Time (3 hours ahead)


Predicted Bike Count
Time (3 hours ahead)


BIKE_PREDICTION
HUMIDITY

APPENDIX




```sql
SELECT RENTED_BIKE_COUNT, HOUR FROM seoul_bike_sharing WHERE HOUR > '0'
```

History | Results

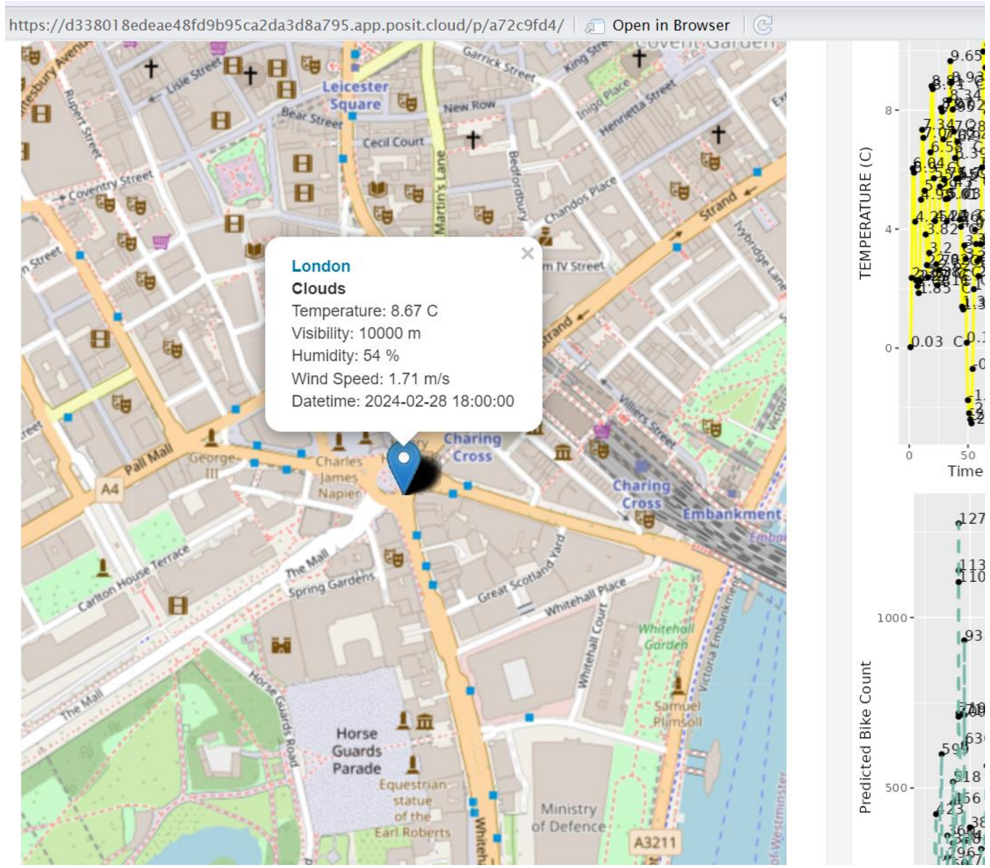
Result set 1 | Details

Filter table   Total:8113

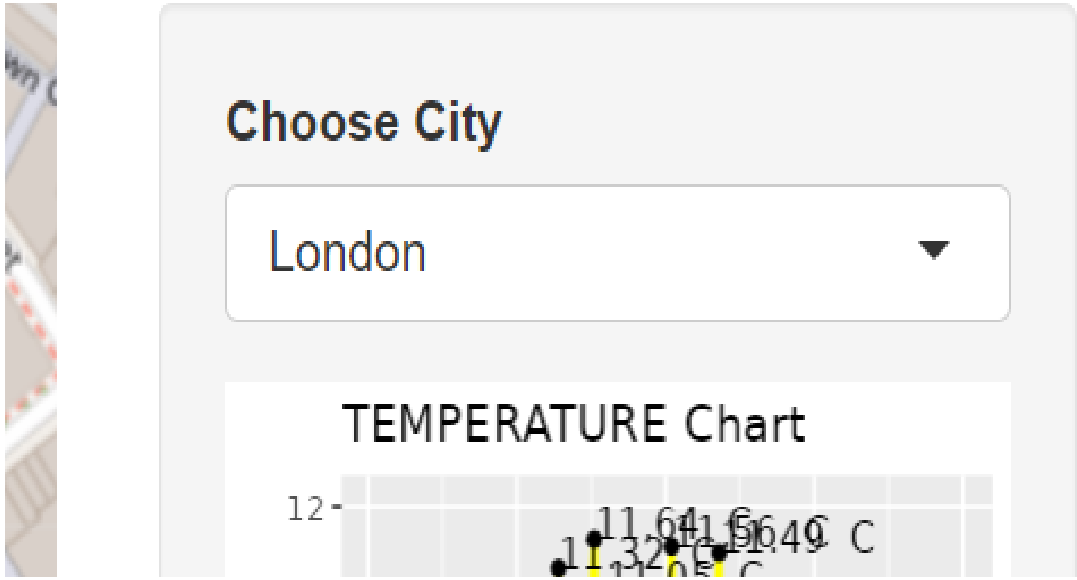
RENTED_BIKE_COUNT	HOUR
204	1
173	2
107	3
78	4

Items per page: 50   1–50 of 8113 items   1   1 of 163 pages

APPENDIX



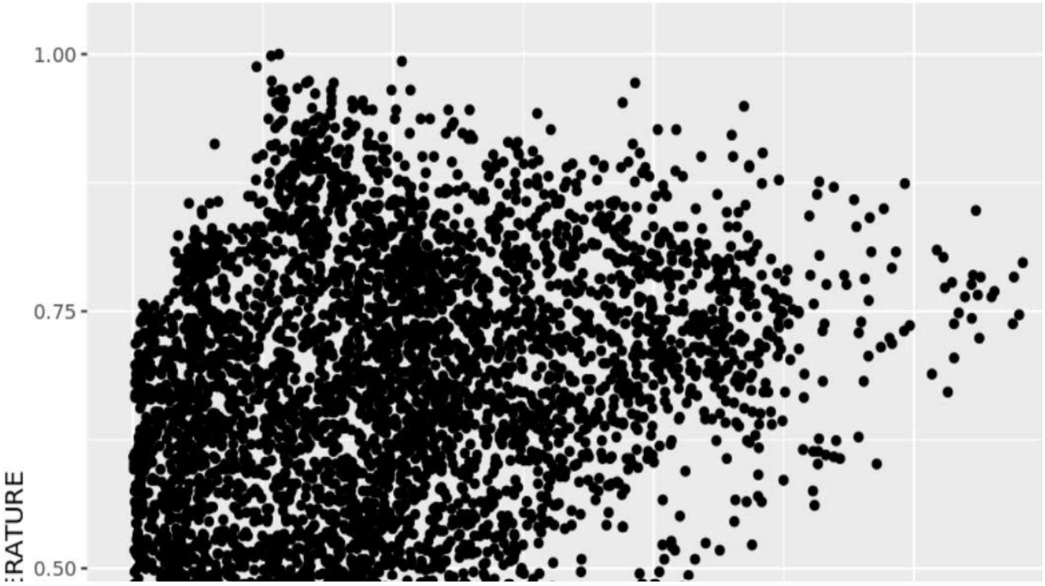
https://d338018edeae48fd9b95ca2da3d8a795.app.posit.cloud/p/a72c9fd4/
Open in Browser
London
Clouds
Temperature: 8.67 C
Visibility: 10000 m
Humidity: 54 %
Wind Speed: 1.71 m/s
Datetime: 2024-02-28 18:00:00
TEMPERATURE (C)
Time
Predicted Bike Count

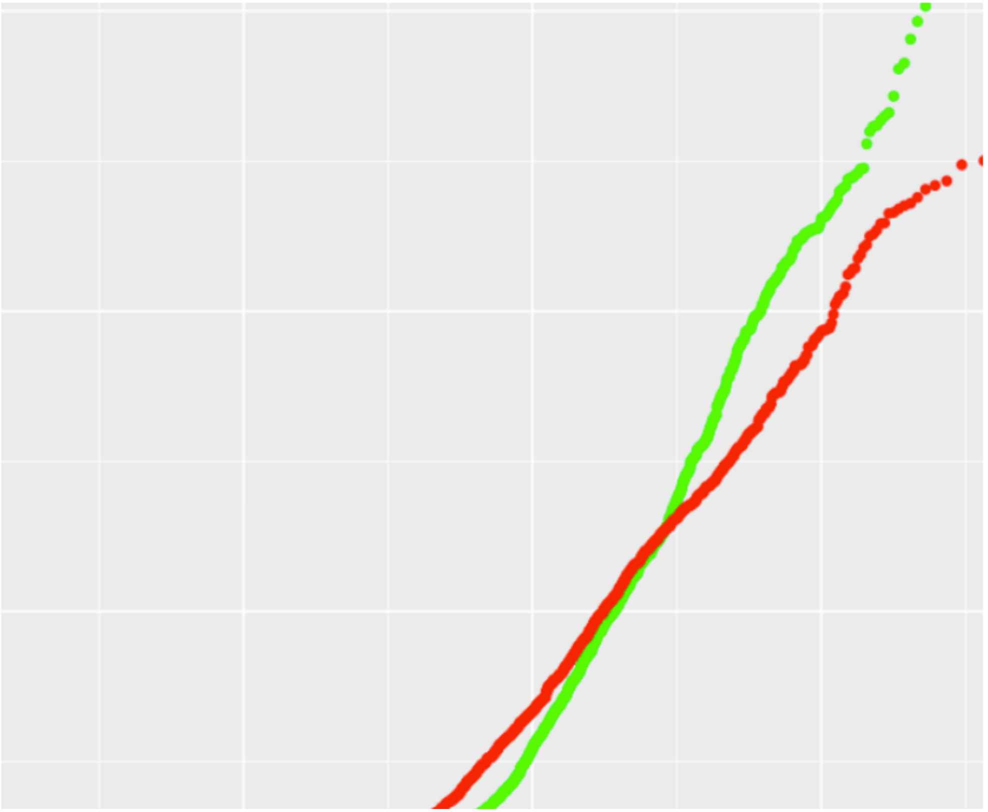


Choose City
London
TEMPERATURE Chart
12

APPENDIX

```
[6]: ggplot(data = train_data, aes(RENTED_BIKE_COUNT, TEMPERATURE)) +
         geom_point()
```





C

```sql
SELECT RENTED_BIKE_COUNT, HOUR FROM seoul_bike_sharing WHERE HOUR > '0'
```

History | **Results**

Result set 1 | Details

Filter table — Total:8113

RENTED_BIKE_COUNT	HOUR
204	1
173	2
107	3
78	4

Items per page: 50 — 1–50 of 8113 items — 1 — 1 of 163 pages

Week 5 > Peer Review: Submit your Work and Review your Peers — Previous — Next

Web Scrapping

```python
# Call the get_wiki_covid19_page function and print the response
get_wiki_covid19_page()
```

```
Response [https://en.wikipedia.org/w/index.php?Title=Template%3ACOVID-19_testing_by_country]
  Date: 2023-11-30 07:13
  Status: 200
  Content-Type: text/html; charset=UTF-8
  Size: 100 kB
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-fea...
<head>
<meta charset="UTF-8">
<title>Wikipedia, the free encyclopedia</title>
<script>(function(){var className="client-js vector-feature-language-in-heade...
"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","...
```

APPENDIX

DATE	HOUR	max (RENTED_BIKE_COUNT)
19/06/2018	18	3556

	DATE	HOUR	RENTED_BIKE_COUNT
1	19/06/2018	18	3556
2	21/06/2018	18	3418
3	12/06/2018	18	3404
4	20/06/2018	18	3384
5	04/06/2018	18	3380
6	22/06/2018	18	3365
7	08/06/2018	18	3309
8	10/09/2018	18	3298
9	17/09/2018	18	3277
10	12/09/2018	18	3256

```
ui <- fluidPage(
  titlePanel("Trends in Demographics and Income"),
  fluidRow(
    column(width = 12,
           wellPanel(
             selectInput("country", "filter by country",
                         choices = c("United-States", "Canada", "Mexico", "Germany", "Phillipines")
                         )
                        )
           )
  ),
  fluidRow(
    column(width = 4,
           radioButtons(inputId = "continous_variables",
                        choices = c("age", "hours_per_week")),
           radioButtons(inputId = "graph_type",
                        choices = c("histogram", "boxplot")
                        )
```

APPENDIX: DATA COLLECTION WEB SCRAPING

```
#Web Scraping:
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"

root_node <- read_html(url)
table_nodes <- html_nodes(root_node, "table")
table_node <- html_node(root_node, "table")

length_table <- length(table_nodes)
for (i in 1:length_table) {   print (table_nodes[[i]]) }  #seen 1st table is relevant

table_node <- table_nodes[[1]]  #1st table
df <- as.data.frame(html_table(table_node))
summary(df)
write.csv(df, file="C:/Users/M/Desktop/Studieren 2013/R complete/IBM R Capstone/wiki_bicycle.csv", row.names=FALSE)
#end web scraping
```

```
> summary(df)
   Country              City               Name              System            Operator           Launched          Discontinued
 Length:564         Length:564         Length:564         Length:564         Length:564         Length:564         Length:564
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Stations           Bicycles         Daily ridership
 Length:564         Length:564         Length:564
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character
```

APPENDIX: DATA COLLECTION CSV FILE

```r
# Download some general city information such as name and locations
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_worldcities.csv"
# download the file
download.file(url, destfile = "C:/Users/M/Desktop/Studieren 2013/R complete/IBM R Capstone/raw_worldcities.csv")


# Download a specific hourly Seoul bike sharing demand dataset
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_seoul_bike_sharing.csv"
# download the file
download.file(url, destfile = "C:/Users/M/Desktop/Studieren 2013/R complete/IBM R Capstone/raw_seoul_bike_sharing.csv")
```

D collection results: 4 files handled in next steps

```r
(dataset_list <- c('wiki_bicycle.csv',
  'raw_seoul_bike_sharing.csv', 'cities_weather_forecast.csv',
  'raw_worldcities.csv'))
```

APPENDIX: DATA WRANGLING

1. Standardize Uppercases, no white spaces
2. Regular Expressions to trim results
3. **Handle missing values (NA)**
4. Generate indicator columns (seasons,holiday)
5. Normalization (weather parameters)

```
#RENTED_BIKE_COUNT only has about 3% missing values (295 / 8760)
# Drop rows with `RENTED_BIKE_COUNT` column == NA
bike_sharing_df <- bike_sharing_df %>% drop_na(RENTED_BIKE_COUNT)
```

```
#missing values for TEMPERATURE are found in SEASONS == Summer, so
#reasonable to impute those missing values with the summer average temperature.
AVG_summer_temp <-bike_sharing_df %>%
  filter(SEASONS=="Summer") %>%
  group_by(SEASONS) %>%
  summarize(mean= mean(TEMPERATURE, na.rm=TRUE)) #AVG_summer_temp$mean

bike_sharing_df<-bike_sharing_df %>%
mutate(TEMPERATURE = ifelse(is.na(TEMPERATURE),AVG_summer_temp$mean,TEMPERATURE))
```

APPENDIX: DATA WRANGLING

1. Standardize Uppercases, no white spaces

2. Regular Expressions to trim results
3. Handle missing values (NA)
4. Generate indicator columns (seasons,holiday)
5. Normalization (weather parameters)

```
for (dataset_name in dataset_list){
  # Read dataset
  dataset <- read_csv(dataset_name)
  # Standardized its columns:

  # Convert all column names to uppercase
  names(dataset)<-toupper(names(dataset))

  # Replace any white space separators by underscores, using the str_replace_all function
  names(dataset)<-str_replace_all(names(dataset)," ","_")

    # Save the dataset
  write.csv(dataset, dataset_name, row.names=FALSE)
}
```

APPENDIX: DATA WRANGLING

1. Standardize Uppercases, no white spaces

2. Regular Expressions to trim results

3. Handle missing values (NA)
4. Generate indicator columns (seasons,holiday)
5. Normalization (weather parameters)

```
# remove reference link
remove_ref <- function(strings) {
  #ref_pattern <- "Define a pattern matching a reference link such as [1]"
    ref_pattern <- "\\[[0-9]+\\]"

  # Replace all matched substrings with a white space using str_replace_all()
    #result<-str_replace_all(strings,ref_pattern," ")  #official default
    result<-str_replace_all(strings,ref_pattern," ")   #my preference

  # Trim the result if you want
        result<-str_trim(result, side= c("right"))

    return(result)
}
```

```
# Extract the first number
extract_num <- function(columns){
  # Define a digital pattern
  digitals_pattern <-"[0-9]+"  #"Define a pattern matching a digital substring"

  # Find the first match using str_extract
  first_match<- str_extract(columns,digitals_pattern)

  # Convert the result to numeric using the as.numeric() function
  result <- as.numeric(first_match)
  return (result)
  }
```

```
sub_bike_sharing_df<-sub_bike_sharing_df %>%
  #select(SYSTEM) %>%
  mutate(SYSTEM=remove_ref(SYSTEM),CITY=remove_ref(CITY))
```

```
sub_bike_sharing_df<-sub_bike_sharing_df %>%
  mutate(BICYCLES=extract_num(BICYCLES))
```

APPENDIX: DATA WRANGLING WITH SQL

- dbGetQuery(conn, 'SELECT COUNT(DATE) FROM SEOUL_BIKE_SHARING_table')
- dbGetQuery(conn, 'SELECT COUNT(HOUR) FROM SEOUL_BIKE_SHARING_table WHERE RENTED_BIKE_COUNT <>0 ‘)
- dbGetQuery(conn, 'SELECT * FROM CITIES_WEATHER_FORECAST_table limit 1 ')
- dbGetQuery(conn, 'SELECT distinct(SEASONS) FROM SEOUL_BIKE_SHARING_table‘)
- dbGetQuery(conn, 'SELECT (Date) FROM SEOUL_BIKE_SHARING_table limit 1')

dbGetQuery(conn, 'SELECT DISTINCT(Date) FROM SEOUL_BIKE_SHARING_table WHERE Date=(SELECT MIN(Date) FROM SEOUL_BIKE_SHARING_table) OR Date=(SELECT MAX(Date) FROM SEOUL_BIKE_SHARING_table) ')

- dbGetQuery(conn, 'SELECT Date,HOUR,max (RENTED_BIKE_COUNT)FROM SEOUL_BIKE_SHARING_table ')
- dbGetQuery(conn, 'SELECT SEASONS, HOUR,AVG (RENTED_BIKE_COUNT)as AVG_bikes_rented, AVG(TEMPERATURE)as AVG_tempFROM SEOUL_BIKE_SHARING_table group by SEASONS, HOUR order by AVG_bikes_rented desc LIMIT 10')
- dbGetQuery(conn, 'SELECT SEASONS, AVG (RENTED_BIKE_COUNT) as AVG_Rented, MAX (RENTED_BIKE_COUNT)Max_Rented, MIN (RENTED_BIKE_COUNT) as MIN_Rented ,SQRT(AVG(RENTED_BIKE_COUNT*RENTED_BIKE_COUNT) - AVG(RENTED_BIKE_COUNT)*AVG(RENTED_BIKE_COUNT)) as STD_deviationFROM SEOUL_BIKE_SHARING_table group by SEASONS order by AVG_Rented DESC ')
- dbGetQuery(conn, 'SELECT SEASONS, AVG (RENTED_BIKE_COUNT) as AVG_Rented,AVG (TEMPERATURE) as AVG_TEMPERATURE, AVG (HUMIDITY) as AVG_HUMIDITY, AVG (WIND_SPEED) as AVG_WIND_SPEED, AVG (VISIBILITY) as AVG_VISIBILITY, AVG (DEW_POINT_TEMPERATURE) as AVG_DEW_POINT_TEMPERATURE, AVG (SOLAR_RADIATION) as AVG_SOLAR_RADIATION, AVG (RAINFALL) as AVG_RAINFALL, AVG (SNOWFALL) as AVG_SNOWFALL FROM SEOUL_BIKE_SHARING_table group by SEASONS order by AVG_Rented desc ')
- dbGetQuery(conn, 'SELECT S.BICYCLES, S.CITY, S.COUNTRY, W.LAT, W.LNG, W.POPULATIONFROM BIKE_SHARING_SYSTEMS_table S, WORLD_CITIES_table W WHERE S.CITY="Seoul" AND S.CITY=W.CITY ')
- dbGetQuery(conn, 'SELECT S.CITY, sum(s.BICYCLES),W.LAT, W.LNG, W.POPULATION FROM BIKE_SHARING_SYSTEMS_table S, WORLD_CITIES_table W WHERE S.CITY=W.CITY group by S.CITY Having S.BICYCLES between 15000 AND 20000 ‘)