APPLIED DATA SCIENCE WITH R CAPSTONE PROJECT

KWOBA FREDRICK

05/11/2023

OUTLINE

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

EXECUTIVE SUMMARY

- Project Objective
- Methodology
- Key Findings
- Implications
- Conclusion

EXECUTIVE SUMMARY

- **Project Objective**
  - Predicting bike sharing demand in urban center based on weather
- **Methodology**
  - Data Collection: Gathered comprehensive data on weather, demographics, and historical bike usage.
  - Engineered key variables that is weather and date/time predictors for robust predictive modeling.
  - Employed ensemble techniques for accurate demand forecasts, these included regularization, polynomial and interaction terms.

EXECUTIVE SUMMARY

- **Key Findings**
  - Bike sharing demand is high in summer then autumn, spring and finally winter
  - Bike sharing demand is high in the evening hours, the hour with the highest number of bike rented being ‘18’
- **Implications**
  - Informed Urban Mobility Planning: Insights support infrastructure development and resource allocation.
  - Enhanced User Experience: Anticipated demand aids in optimizing bike availability and service quality

EXECUTIVE SUMMARY

- **Conclusion**
  - The project demonstrates the potential for data-driven insights to revolutionize urban bike-sharing services, paving the way for more efficient and sustainable transportation solutions.

INTRODUCTION

- **Objective**: Analyze weather's impact on urban bike-sharing demand.
- **Approach**: Collect, process weather and demand data, conduct exploratory data analysis, build predictive models.
- **Outcome**: Real-time interactive dashboard showing weather and bike demand.

INTRODUCTION CONTINUED

- **Project Phases**
  - **Data Collection**: Gather weather and bike-sharing data from various sources.
  - **Exploratory Data Analysis (EDA)**: Uncover trends, patterns, and anomalies.
  - **Predictive Modeling**: Develop models to forecast bike-sharing demand.
- **Interactive Dashboard**
  - **Purpose**: Provide stakeholders with real-time insights.
  - **Features**: Interactive map, current weather, estimated bike demand.

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

DATA COLLECTION

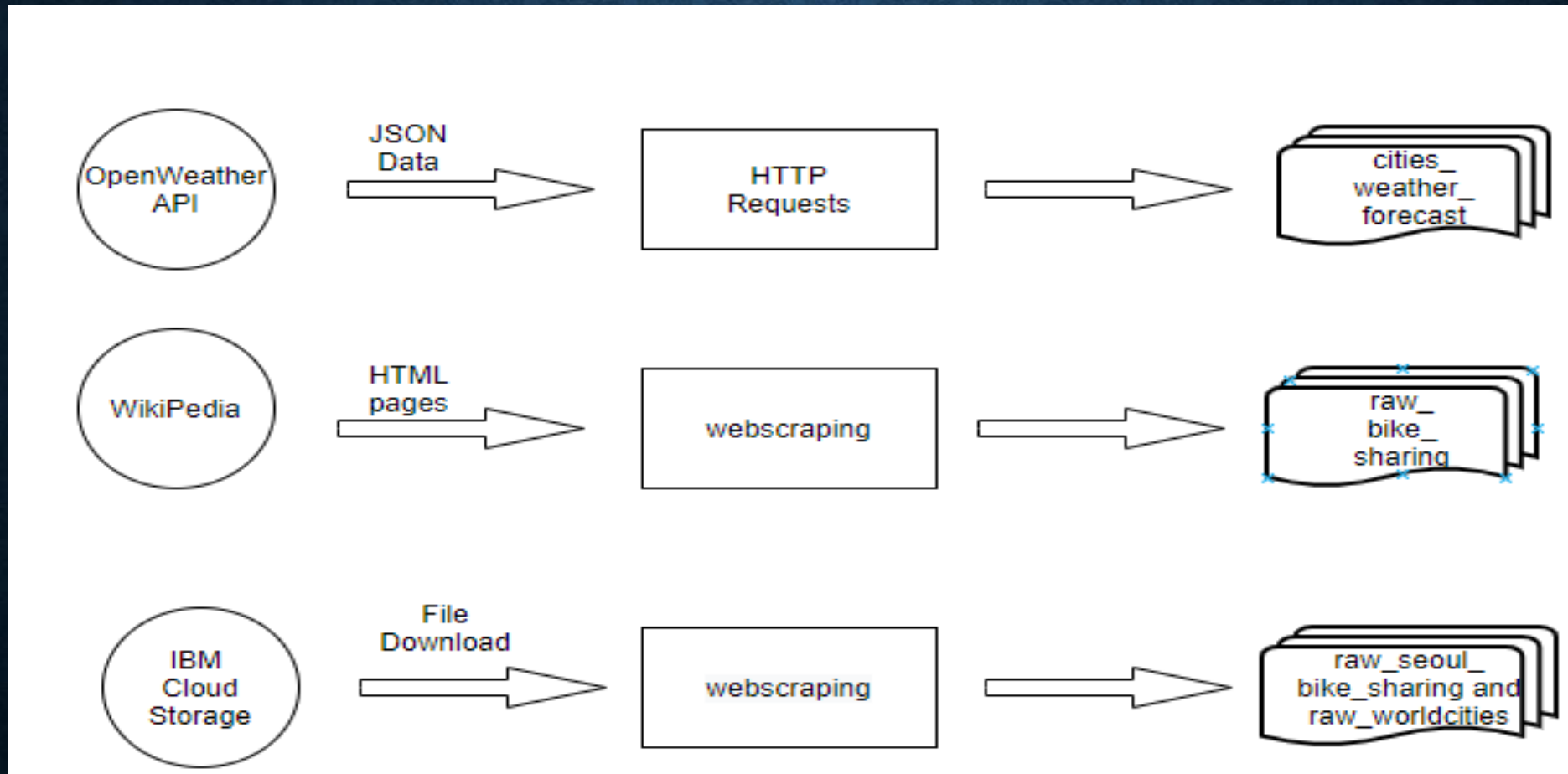
- I was able to collect three sets of data namely :
  - cities_weather_forecast.csv
  - raw_bike_sharing_systems.csv
  - raw_worldcities.csv
  - raw_seoul_bike_sharing.csv
- **cities_weather_forecast.csv** dataset was obtained from the OpenWeather API through HTTP requests

  The data you will be connecting to provides the weather forecast for every 3 hours over the next 5 days.
- **raw_bike_sharing_systems.csv** dataset was obtained by web scraping a Wikipedia table. It lists active bicycle-sharing systems around the world.
- **raw_seoul_bike_sharing.csv** and **raw_worldcities.csv** datasets were downloaded as csv files from cloud storage
- **World cities dataset** contains information such as name, latitude, and longitude, about major

  cities around the world.
- **raw_seoul_bike_sharing** contains weather information (Temperature, Humidity, Windspeed, Visibility,

  Dewpoint, Solar radiation, Snowfall, Rainfall), and the number of bikes rented per hour and date in Seoul .

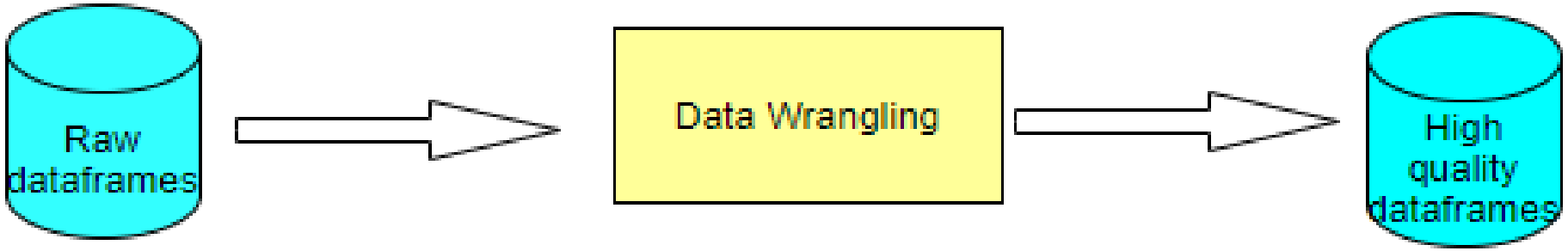
DATA COLLECTION PROCESS ILLUSTRATIONS OF DIFFERENT DATASETS.



DATA WRANGLING

- Describe how data sets were processed
  - Detecting and handling missing values, inconsistent data, and incorrect data formats
  - Creating indicator(dummy) variables for categorical variables
  - Normalizing the data

DATA WRANGLING PROCESS



Inconsistent
incorrect format
missing values

clean
desired format
no missing values

EDA WITH SQL

- Summarize performed SQL queries using bullet points

SQL queries performed were for the following intentions

❖Finding valuable statistics

❖Filter database based on different cities

❖Find patterns such as seasonality and similarity

EDA WITH DATA VISUALIZATION

- Summarize what charts were plotted using bullet points
  - **Histogram**

    This was to understand the distribution of data
  - **Scatter plots**

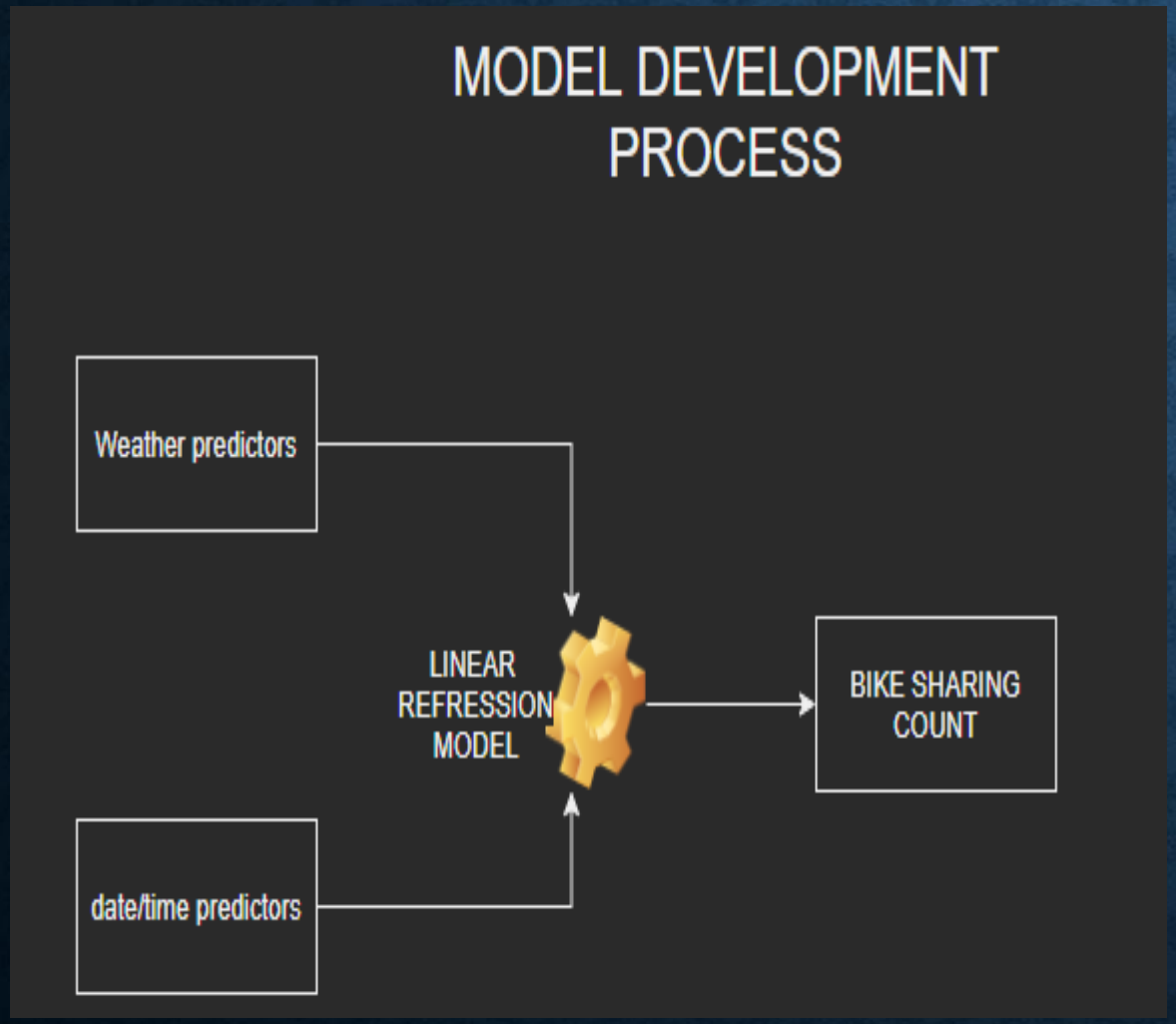
    For finding correlations between important features using scatter plots
  - **Boxplots**

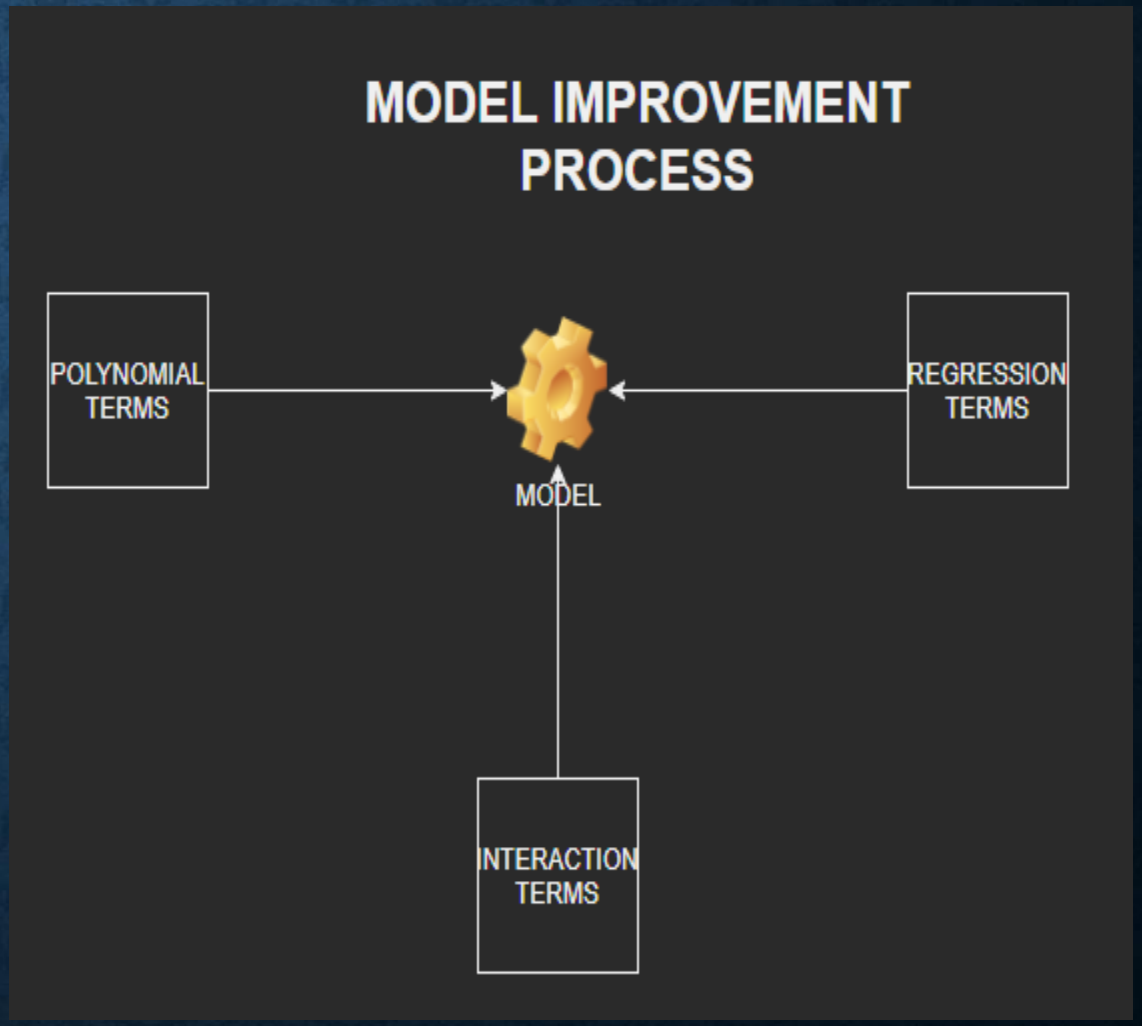
    For spotting outliers and irregular behavior in different features of data

PREDICTIVE ANALYSIS

- Summarize how you built, evaluated, improved and found the best performing model
  - ✓ I built five linear regression models by using both the Date/time variables as predictors and the bike sharing count as the response variable
  - ✓ I was able to evaluate these models based on their R-squared and RMSE values
  - ✓ These models were improved by use of polynomial and interaction terms and to reduce the complexity and curb overfitting, we implemented different regularization techniques in these models

MODEL DEVELOPMENT PROCESS





BUILD A R SHINY DASHBOARD

- Summarize what plots and interactions you built into the dashboard
  - ❑ A static temperature trend line
  - ❑An interactive bike-sharing demand prediction trend line
  - ❑A static humidity and bike sharing demand prediction correlation plot

RESULTS

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

EDA WITH SQL

The following queries were created in this section

- ✓ **Busiest bike rental times**
- ✓ **Hourly popularity and temperature by seasons**
- ✓ **Rental Seasonality**
- ✓ **Weather Seasonality**
- ✓ **Bike-sharing info in Seoul**
- ✓ **Cities similar to Seoul**

BUSIEST BIKE RENTAL TIMES

	DATE	HOUR	RENTED_BIKE_COUNT
1	19/06/2018	18	3556
2	21/06/2018	18	3418

19/06/2018 at 18th hour had the most bike rentals of up to 35556 bikes

HOURLY POPULARITY AND TEMPERATURE BY SEASONS

	rental_hour	SEASONS	avg_temperature	avg_bike_count
1	18	Summer	29.38791	2135.141
2	18	Autumn	16.03185	1983.333
3	19	Summer	28.27378	1889.250
4	20	Summer	27.06630	1801.924
5	21	Summer	26.27826	1754.065
6	18	Spring	15.97222	1689.311
7	22	Summer	25.69891	1567.870
8	17	Autumn	17.27778	1562.877
9	17	Summer	30.07691	1526.293
10	19	Autumn	15.06346	1515.568

- It's observed that the 18th hour in summer has the highest average bike count and average temperature values

RENTAL SEASONALITY

- In the proceeding slides showing rental seasonality, it is observed that summer has the highest values of average bike count

RENTAL SEASONALITY FOR AUTUMN AND SPRING

	season	rental_hour	avg_bike_count	min_bike_count	max_bike_count	std_dev_bike_count
1	Autumn	0	709.43750	119	1336	219.14298
2	Autumn	1	552.50000	144	1001	191.54216
3	Autumn	2	377.47500	55	785	144.90134
4	Autumn	3	256.55000	28	514	102.53108
5	Autumn	4	169.02500	24	338	58.63957
6	Autumn	5	163.41250	24	264	53.88174
7	Autumn	6	359.48750	23	691	180.27049
8	Autumn	7	788.87654	5	1556	457.96861
9	Autumn	8	1345.03704	6	2391	758.35956
10	Autumn	9	848.43210	5	1322	334.52653
11	Autumn	10	715.27160	2	1113	252.56839
12	Autumn	11	802.95062	20	1284	302.57238
13	Autumn	12	934.64198	17	1634	347.69066
14	Autumn	13	1002.66667	18	1849	369.30997
15	Autumn	14	1058.82716	17	1995	403.85277
16	Autumn	15	1156.70370	8	2200	455.21810
17	Autumn	16	1293.20988	14	2270	498.10638
18	Autumn	17	1562.87654	23	2432	554.31649
19	Autumn	18	1983.33333	40	3298	778.44135
20	Autumn	19	1515.56790	19	2518	571.14972
21	Autumn	20	1321.70370	12	2259	508.99136
22	Autumn	21	1253.81481	21	2212	490.92858
23	Autumn	22	1130.20988	14	1868	411.96934
24	Autumn	23	828.72840	4	1523	292.97832
25	Spring	0	481.08889	22	1089	253.38673
26	Spring	1	363.94444	23	837	206.04322
27	Spring	2	252.96667	9	590	145.32266
28	Spring	3	168.47778	2	423	101.20444
29	Spring	4	108.22222	2	252	59.64577
30	Spring	5	116.17778	6	265	64.93117
31	Spring	6	257.23333	5	646	171.97416
32	Spring	7	615.28889	15	1502	433.10666
33	Spring	8	1036.37778	16	2281	700.59774
34	Spring	9	670.38889	9	1277	341.92932
35	Spring	10	570.88889	6	1119	271.26811
36	Spring	11	681.46667	10	1438	318.21688
37	Spring	12	836.37778	10	1798	398.16257
38	Spring	13	886.08889	11	2000	426.05291
39	Spring	14	925.66667	17	2128	456.82396
40	Spring	15	990.97778	14	2329	534.98910
41	Spring	16	1101.45556	31	2479	579.07509
42	Spring	17	1307.97778	24	2410	666.95940
43	Spring	18	1689.31111	22	3251	898.89711
44	Spring	19	1247.71111	29	2656	738.11564
45	Spring	20	1060.51111	12	2252	646.16839
46	Spring	21	1021.12222	7	2140	601.26919
47	Spring	22	897.42222	8	1900	516.68221
48	Spring	23	622.95556	9	1343	358.76182

RENTAL SEASONALITY FOR SUMMER AND WINTER

49	Summer	0	899.06522	26	1394	285.31199
50	Summer	1	698.77174	28	1088	239.68091
51	Summer	2	505.75000	17	1254	194.63827
52	Summer	3	342.67391	22	644	127.53355
53	Summer	4	223.81522	27	421	79.24027
54	Summer	5	245.93478	28	383	79.82056
55	Summer	6	485.83696	37	807	209.10359
56	Summer	7	902.78261	25	1629	466.21836
57	Summer	8	1418.59783	20	2495	712.95259
58	Summer	9	911.00000	9	1401	304.09660
59	Summer	10	723.50000	9	1269	242.33674
60	Summer	11	786.85870	11	1478	270.05521
61	Summer	12	875.96739	9	1747	342.11203
62	Summer	13	890.19565	24	1834	367.56328
63	Summer	14	894.67391	14	2088	408.23329
64	Summer	15	1009.71739	51	2288	433.48370
65	Summer	16	1174.17391	41	2474	503.91596
66	Summer	17	1526.29348	25	2664	608.79167
67	Summer	18	2135.14130	17	3556	884.08294
68	Summer	19	1889.25000	18	2984	728.87994
69	Summer	20	1801.92391	10	2579	662.21626
70	Summer	21	1754.06522	17	2505	596.13742
71	Summer	22	1567.86957	16	2309	516.64344
72	Summer	23	1153.90217	15	1732	377.80666

73	winter	0	165.17778	42	342	63.81163
74	winter	1	159.05556	43	337	65.84247
75	winter	2	117.78889	38	312	54.94533
76	winter	3	77.81111	21	230	37.68256
77	winter	4	50.47778	20	120	19.05561
78	winter	5	51.22222	13	100	18.98759
79	winter	6	92.82222	9	200	45.48372
80	winter	7	209.56667	3	498	125.47121
81	winter	8	422.20000	4	937	253.13778
82	winter	9	254.60000	3	509	111.70276
83	winter	10	186.40000	4	339	67.98150
84	winter	11	228.32222	9	388	78.23964
85	winter	12	263.63333	4	479	88.68163
86	winter	13	275.25556	7	606	98.32616
87	winter	14	284.28889	17	611	105.02394
88	winter	15	298.64444	7	621	117.54473
89	winter	16	308.57778	9	688	126.50569
90	winter	17	342.51111	8	646	131.99514
91	winter	18	438.30000	11	929	216.66413
92	winter	19	304.03333	19	619	140.32070
93	winter	20	243.23333	8	483	96.22694
94	winter	21	240.50000	10	512	98.08061
95	winter	22	225.13333	13	461	91.87687
96	winter	23	173.43333	23	366	67.49684

WEATHER SEASONALITY

	season	avg_temperature	avg_humidity	avg_wind_speed	avg_visibility	avg_dew_point_temperature	avg_solar_radiation	avg_rainfall	avg_snowfall	avg_bike_count
1	Summer	26.587711	64.98143	1.609420	1501.745	18.750136	0.7612545	0.25348732	0.00000000	1034.0734
2	Autumn	13.821580	59.04491	1.492101	1558.174	5.150594	0.5227827	0.11765617	0.06350026	924.1105
3	Spring	13.021685	58.75833	1.857778	1240.912	4.091389	0.6803009	0.18694444	0.00000000	746.2542
4	Winter	-2.540463	49.74491	1.922685	1445.987	-12.416667	0.2981806	0.03282407	0.24750000	225.5412

- Again, Summer has the highest average bike count value amidst all the weather variables consideration, as shown in the screenshot above

BIKE-SHARING INFO IN SEOUL

	CITY	COUNTRY	LAT	LNG	POPULATION	total_bikes_in_seoul
1	Seoul	Korea, South	37.5833	127	21794000	20000

- Seoul, a city in South Korea has a total of 20000 bikes, with a population of 21794000 people as outlined in the above screenshot

CITIES SIMILAR TO SEOUL

	CITY	COUNTRY	LAT	LNG	POPULATION	BICYCLES
1	Beijing	China	39.9050	116.3914	19433000	16000
2	Ningbo	China	29.8750	121.5492	7639000	15000
3	Shanghai	China	31.1667	121.4667	22120000	19165
4	Weifang	China	36.7167	119.1000	9373000	20000
5	Zhuzhou	China	27.8407	113.1469	3855609	20000
6	Seoul	Korea, South	37.5833	127.0000	21794000	20000

- Cities like Beijing, Ningbo, Shangai, Weifang, Zhuzhou can be compared to Seoul in terms of bicycles and population as shown in the screenshot

EDA WITH VISUALIZATION

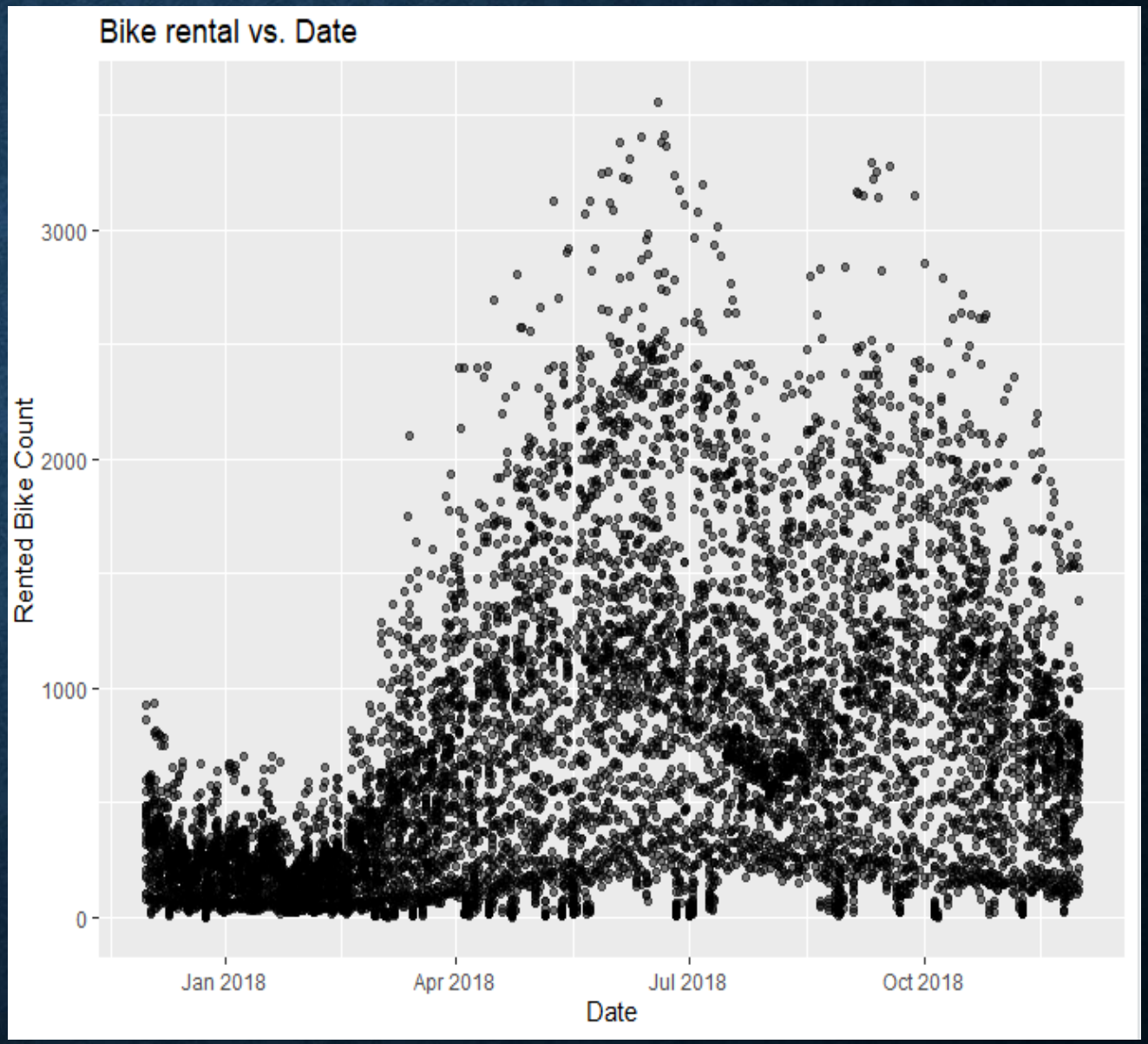
In this section, the following visuals were created

- ✓ A scatter plot of RENTED_BIKE_COUNT vs. DATE
- ✓ A scatter plot of RENTED_BIKE_COUNT vs. DATETIME
- ✓ Bike rental histogram with a kernel density curve
- ✓ Boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS

BIKE RENTAL VS. DATE

In the earlier months of the year, the rental bike count is below 1000's.

Immediately before April, the rental bike count shoots 3500, but it goes below that quantity for the rest of the year.

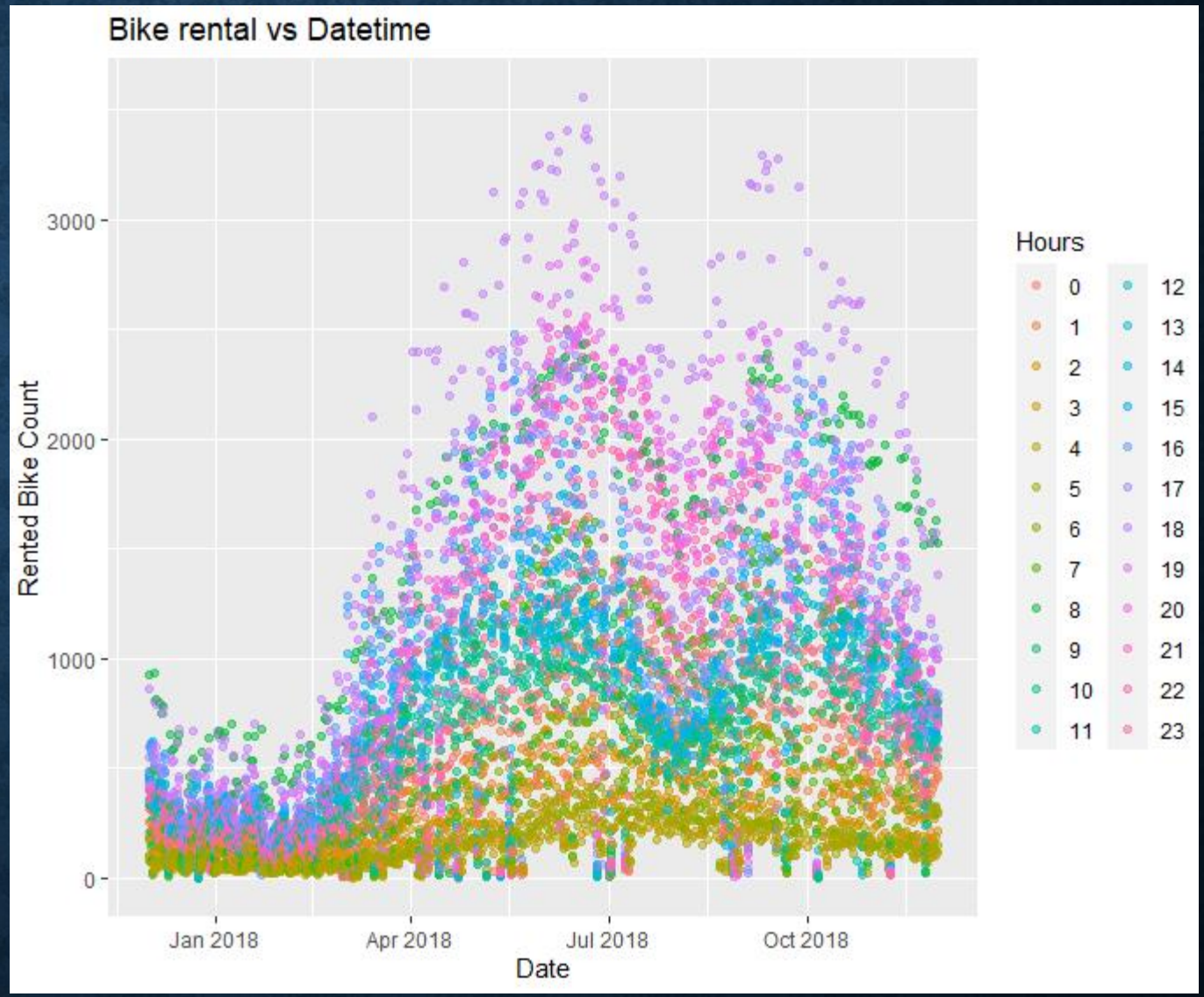


BIKE RENTAL VS. DATETIME

In the earlier months of the year, the rental bike count is below 1000's.

Immediately before April, the rental bike count shoots 3500, but it goes below that quantity for the rest of the year.

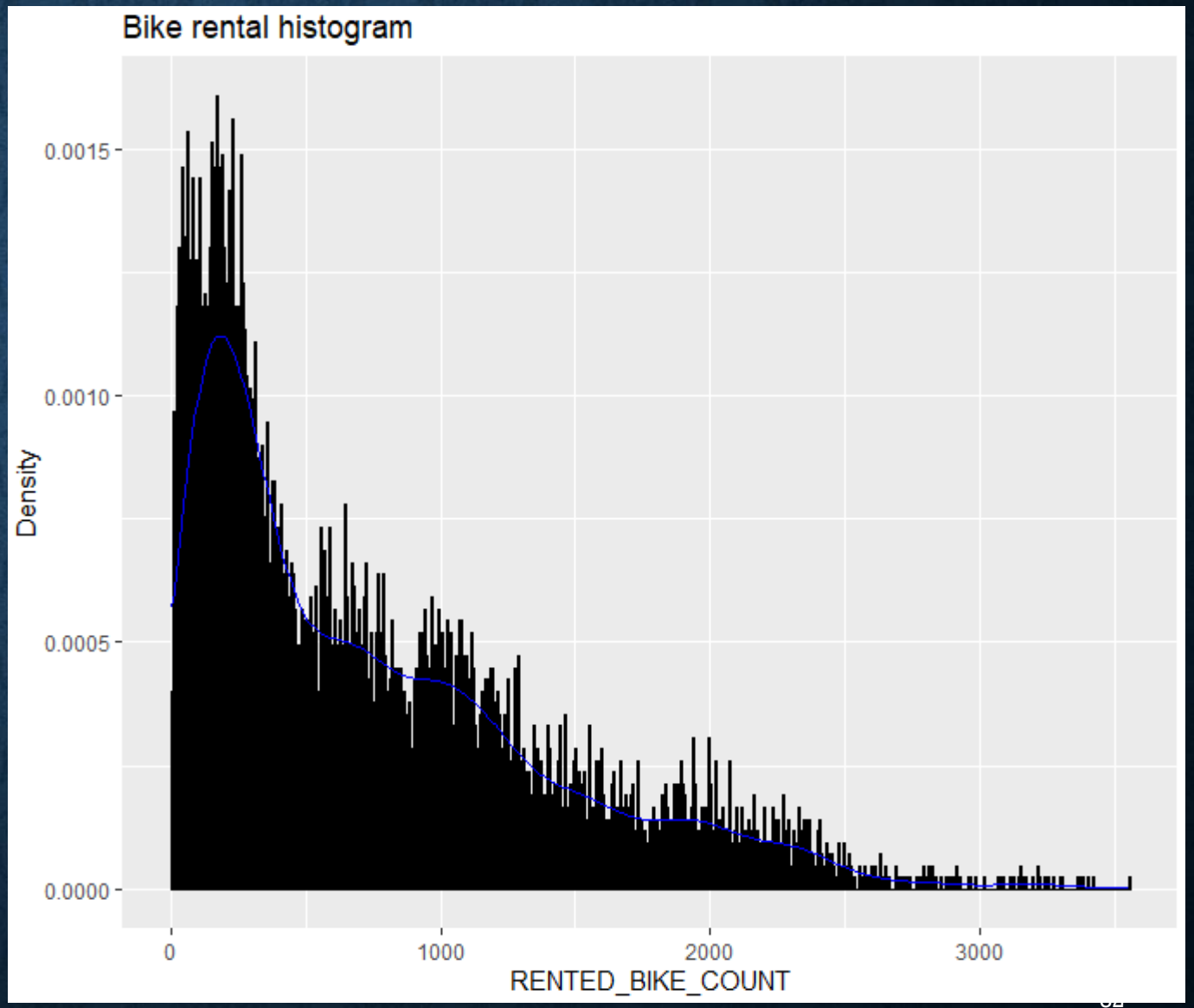
But this time, there is hour of the day incorporated within, but we notify that the morning and afternoon hours dominate



BIKE RENTAL HISTOGRAM

From the screenshot, there is a very high density of Rented bike count below 500

The density goes on decreasing from 1000's, to 2000's and 3000's having the least density of bike counts rented

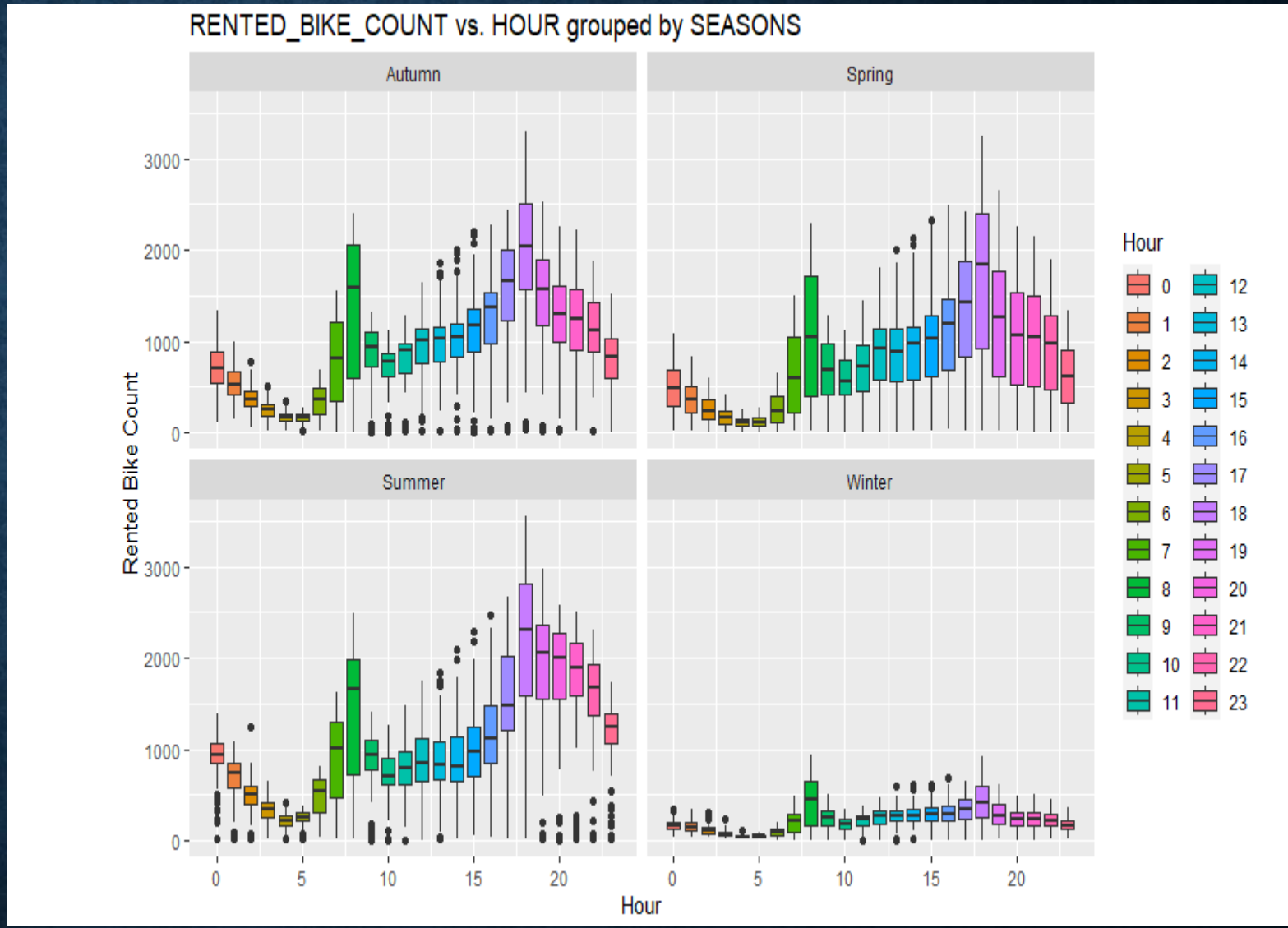


RENTED_BIKE_COUNT VS. HOUR GROUPED BY SEASONS

Generally, summer has the highest number of rented bike count, followed by Autumn, spring and lastly the winter season

Winter and Autumn have a number of outliers more than other seasons below the lower extreme values

And as we can observe, all seasons have a few outliers above the upper extreme values



PREDICTIVE ANALYSIS

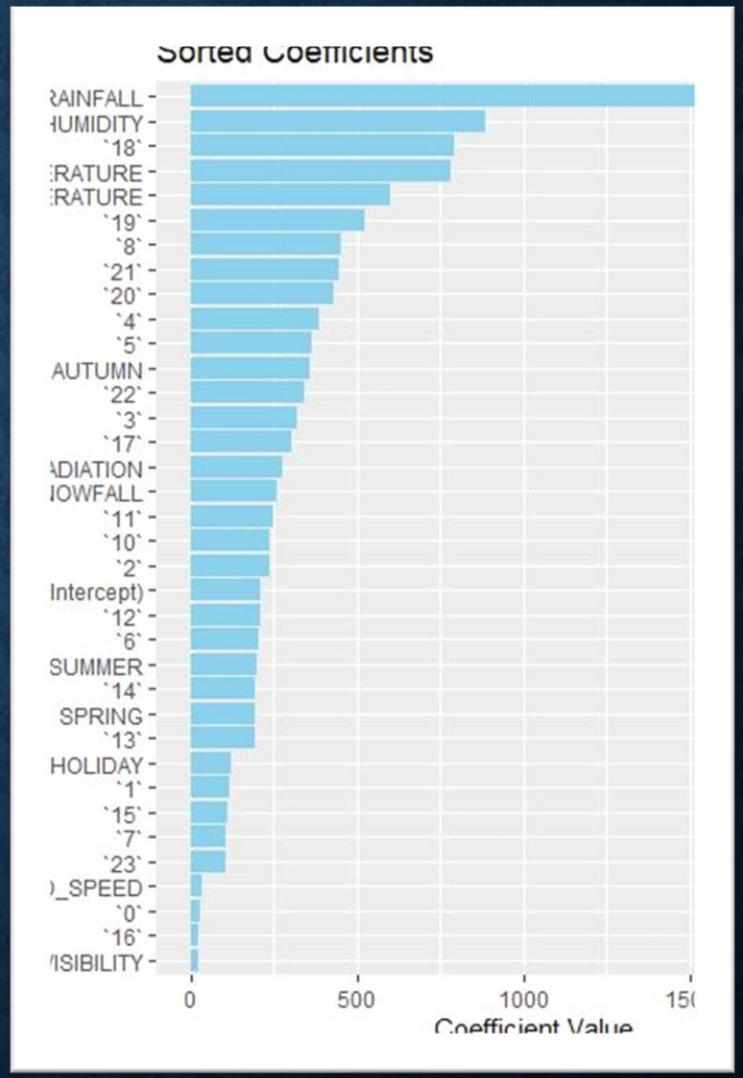
The following activities were carried out in this section:

- ✓ Ranked coefficients
- ✓ Model evaluation
- ✓ Best performing model
- ✓ Q-Q plot of the best model

RANKED COEFFICIENTS

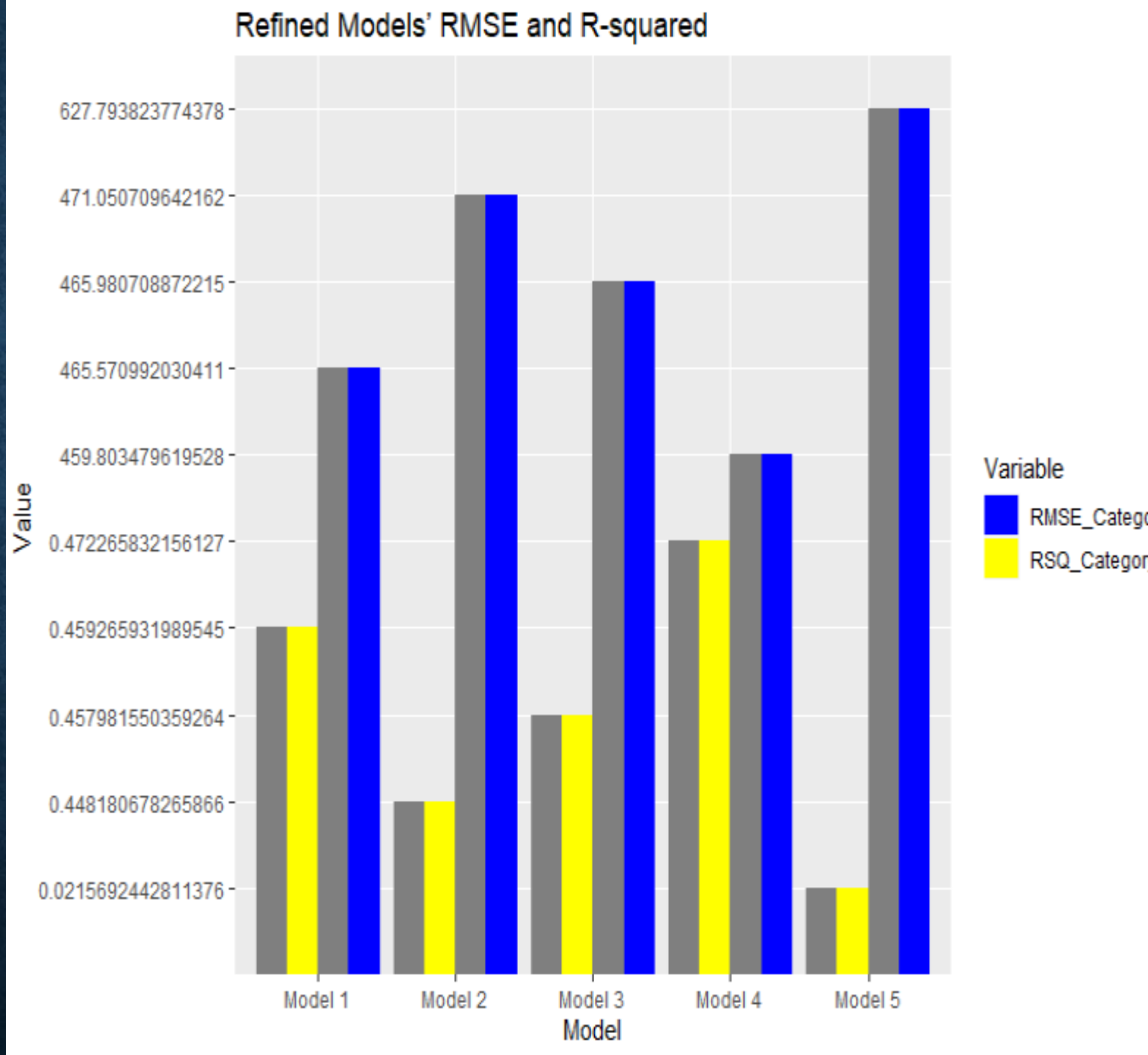
Why some variables are important while some are not for predicting bike-sharing demand ?

- **Multicollinearity:** When two or more independent variables are highly correlated, it can be difficult for the model to separate their individual effects. In such cases, one of the correlated variables may be dropped to reduce multicollinearity.
- **Overfitting:** Including too many variables in a model can lead to overfitting.



MODEL EVALUATION

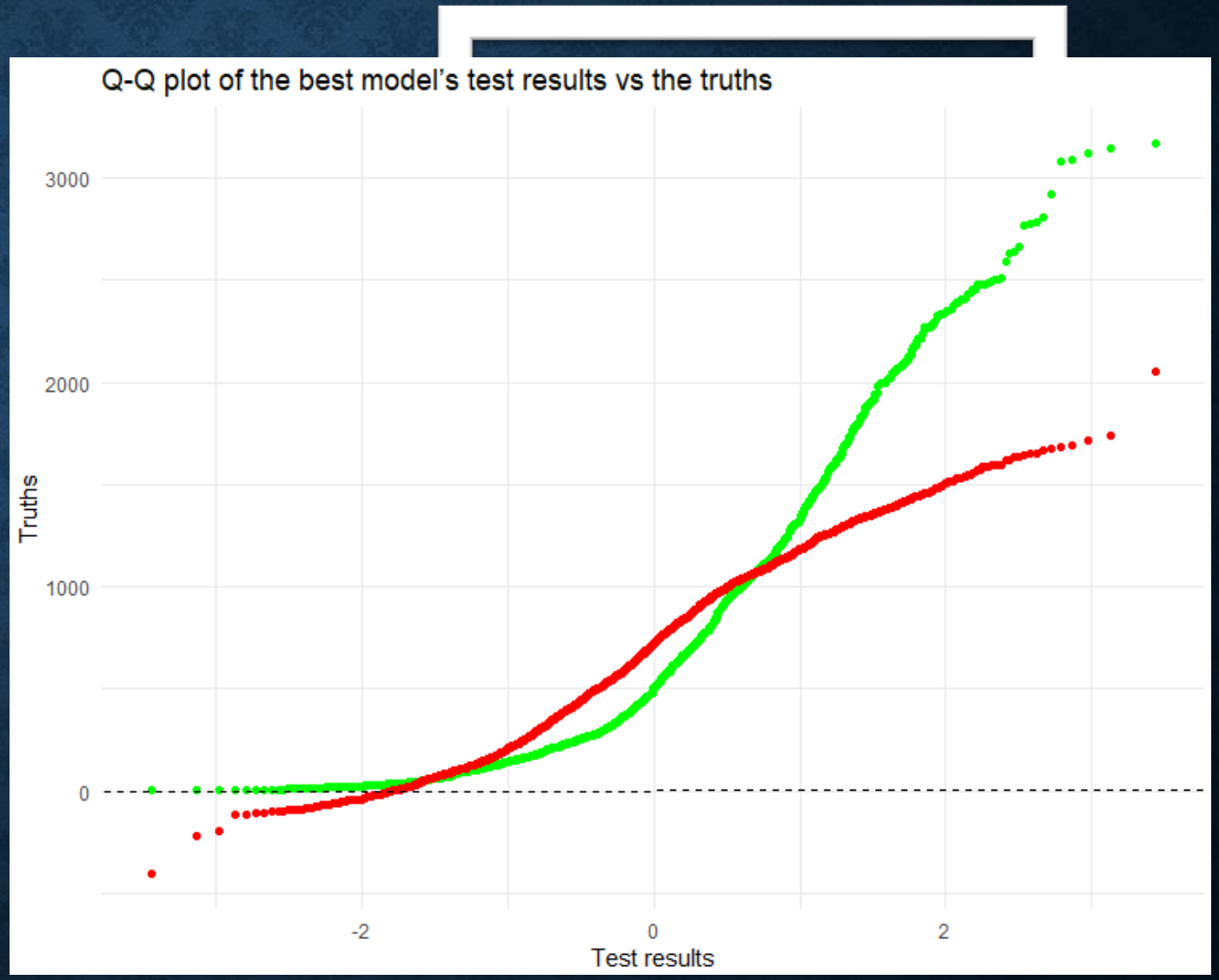
Visual of the refined models' RMSE and R-squared using grouped bar chart



FIND THE BEST PERFORMING MODEL

- interaction_term4 <- RAINFALL* HUMIDITY
- RENTED_BIKE_COUNT ~ TEMPERATURE_poly + interaction_term4 * DEW_POINT_TEMPERATURE+ SOLAR_RADIATION *AUTUMN +SNOWFALL

Q-Q PLOT OF THE BEST MODEL

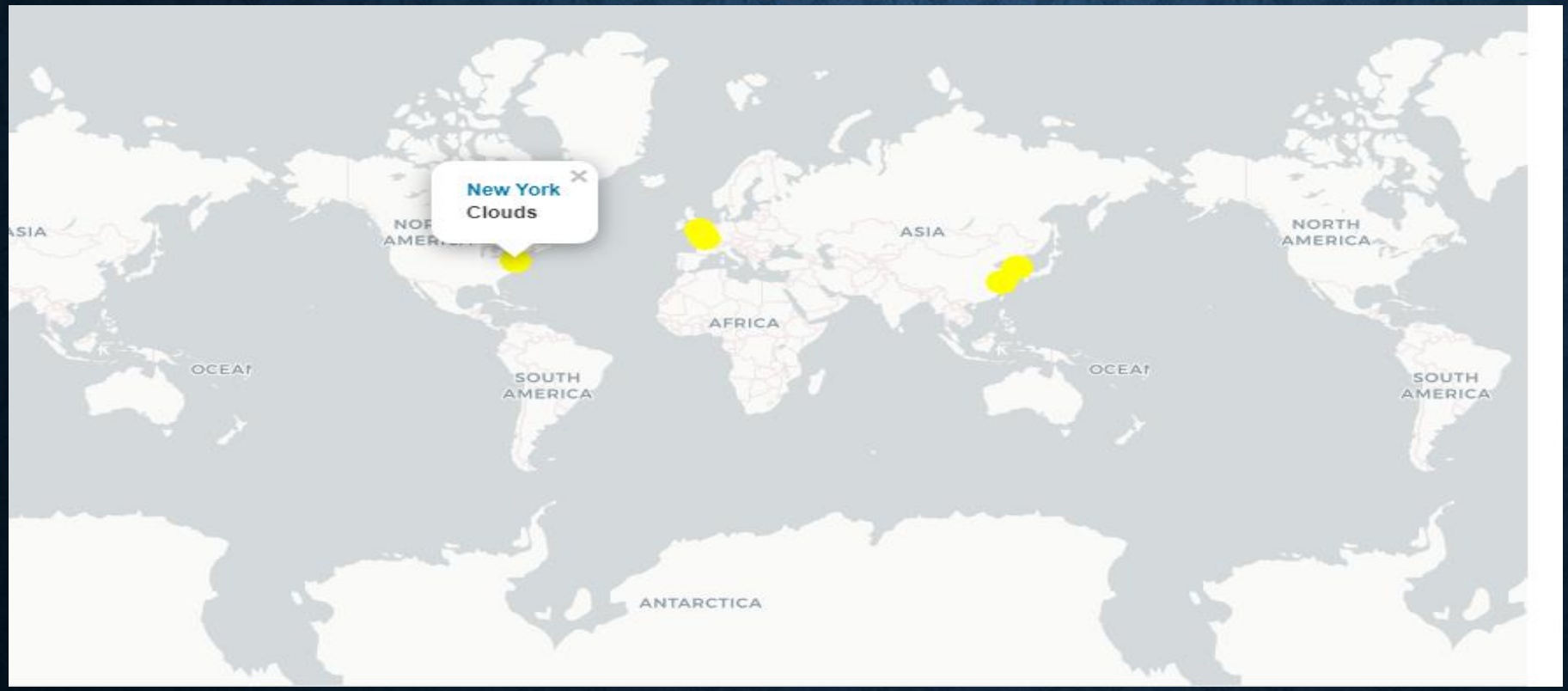


DASHBOARD

This section contains the following

- ✓ Bike-sharing demand prediction app
- ✓ Bike demand prediction in Paris
- ✓ Bike demand prediction in Suzhou

BIKE-SHARING DEMAND PREDICTION APP

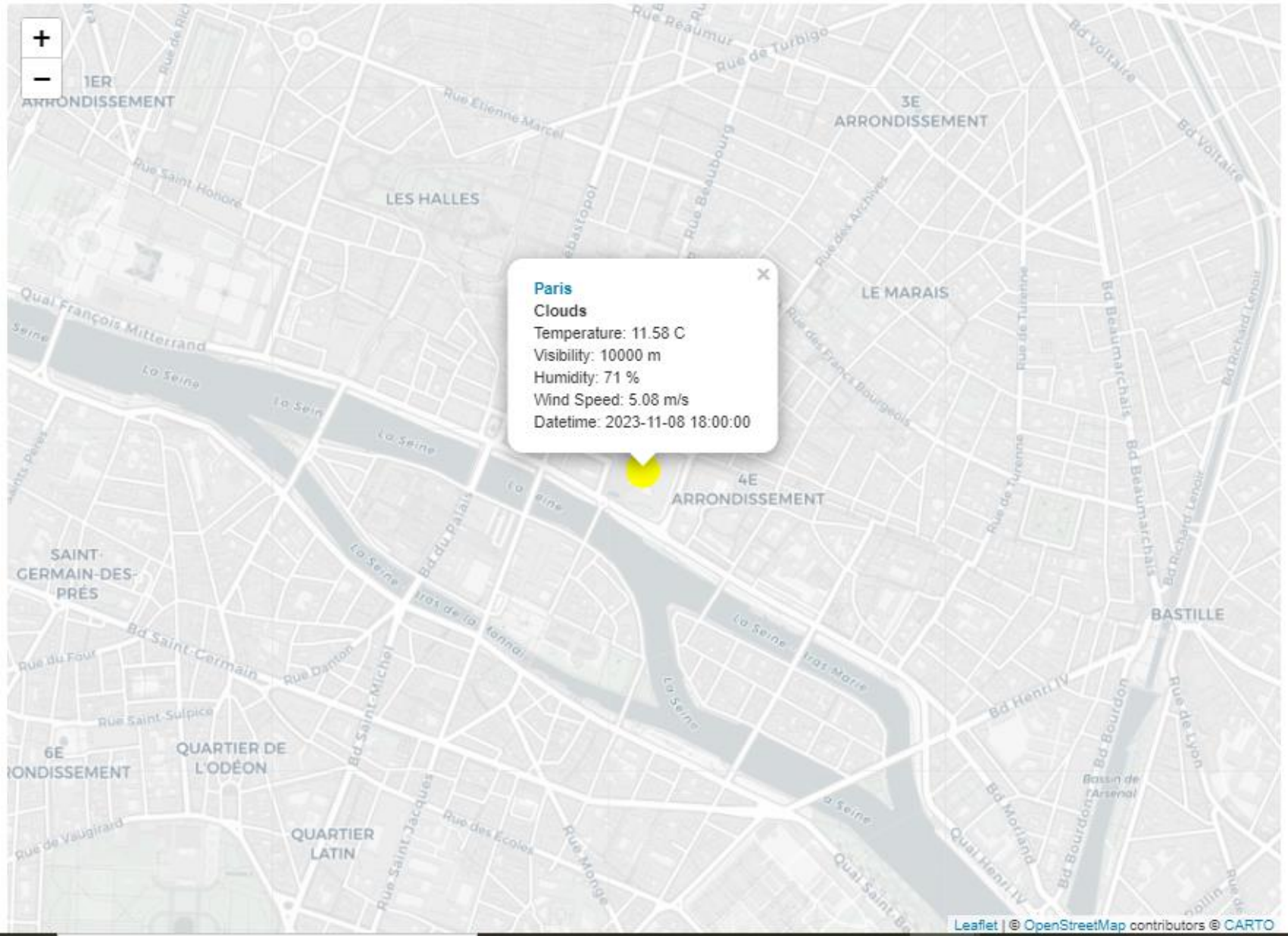


A map to show which cities may have very large bike-sharing demand in the next 5 days

THE IMPORTANT ELEMENTS ON THE SCREENSHOT

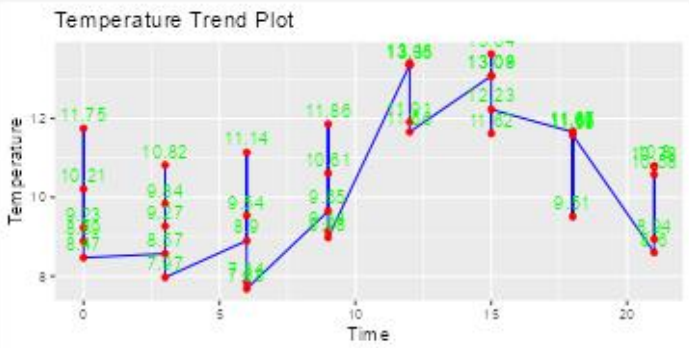
- A map shows which cities may have very large bike-sharing demand in the next 5 days
- Circle markers with different sizes and colors, and popup labels on each city's location indicate the number of rented bikes in different cities

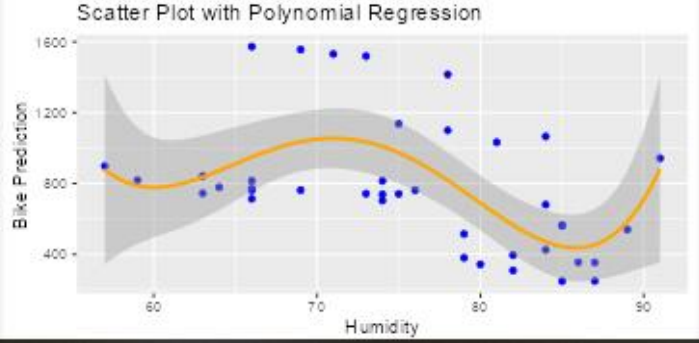
BIKE DEMAND PREDICTION IN PARIS



select a city

Paris



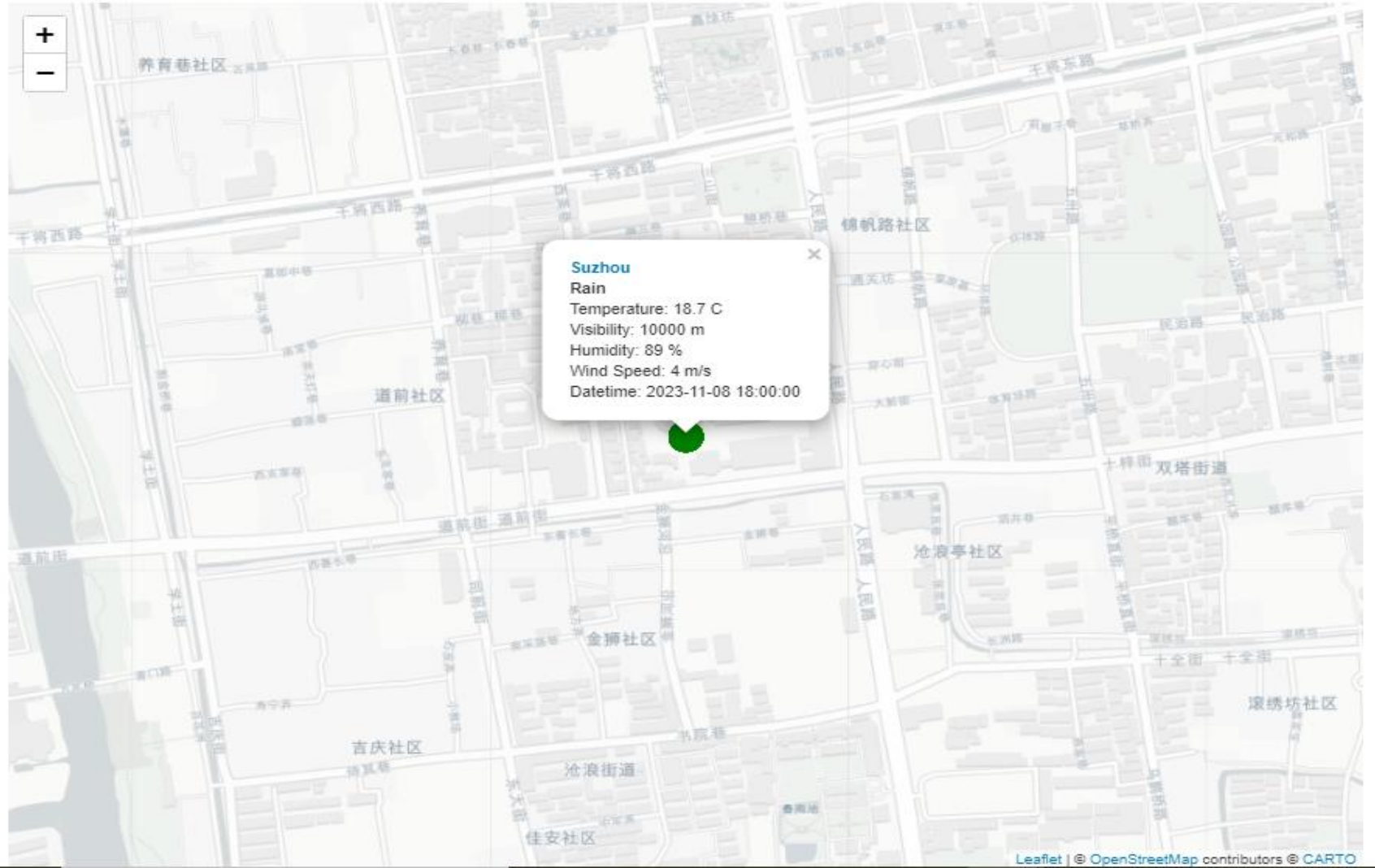


THE IMPORTANT ELEMENTS ON THE SCREENSHOT

- In Paris, it's cloudy so the bike sharing count is medium
- The days start off with low temperature but increase with after morning hours as shown on the temperature trend line curve
- The other thing , there is a quadratic relationship between Bike prediction and humidity within this period, as observed from the scatter plot

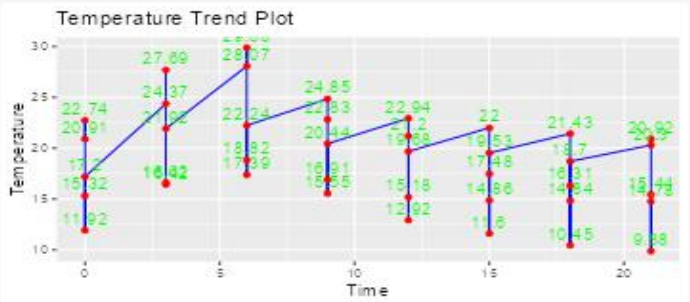
BIKE DEMAND PREDICTION IN SUZHOU

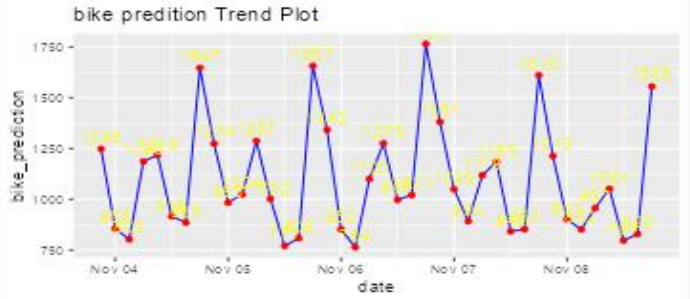
Bike-sharing demand prediction app

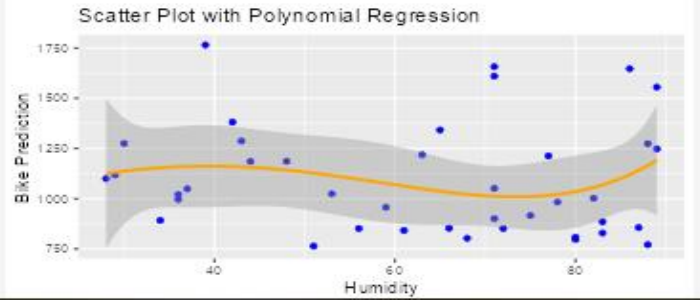


select a city

Suzhou







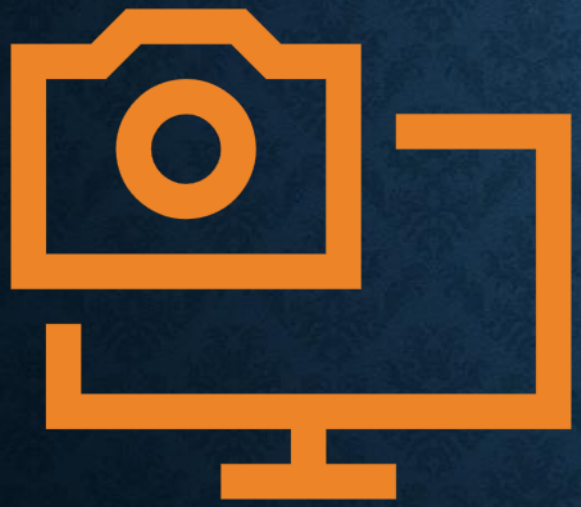
THE IMPORTANT ELEMENTS ON THE SCREENSHOT

- In Suzhou, it's raining, so the bike count level is small
- The period starts off with high temperatures but decrease after morning hours as shown on the temperature trend line curve
- The other thing, there is almost no correlation between Bike prediction and humidity within this period, as observed from the scatter plot

CONCLUSION

- Summer season has the least number of bike count rentals
- Temperature has a positive correlation with the bike count rentals
- Seoul of South Korea has the highest usage of bikes
- The hour of 18 has the highest number of bikes rented

APPENDIX



- This section contains relevant assets such R-code snippets, SQL queries and Notebook outputs

CODE FOR OPENWEATHERAPI

```
# Load required library
library(httr)
library(tidyr)

# Define API endpoint and parameters

get_weather_forecaset_by_cities <- function(city_names){
url <- 'https://api.openweathermap.org/data/2.5/forecast'
api_key <- '786492856e74afacdd517464a9bae978'  # Replace with your actual API key

# Create an empty list to store the responses
responses <- list()

# Loop through the list of cities
for (city in city_names) {
  # Make the API request
  response <- GET(url, query = list(q = city, appid = api_key))

  # Check if the request was successful (status code 200)
  if (http_type(response) == "application/json") {
    # Parse the JSON response
    weather_data <- content(response, "parsed")
    responses[[city]] <- weather_data
  } else {
    print(paste('Error for city:', city))
  }
}
```

CODE FOR OPENWEATHER API

```
# Now, responses will contain weather data for each city

# Assuming 'responses' contains the weather data

# Create an empty list to store all data
all_data <- list()

# Loop through the responses
for (city in city_names) {
  weather_data <- responses[[city]]

  # Extract the specific information you're interested in (e.g., temperature, humidity)
  extracted_data <- data.frame(
    city = rep(city, length(weather_data$list)),
    date_time = sapply(weather_data$list, function(x) x$dt_txt),
    temperature = sapply(weather_data$list, function(x) x$main$temp),
    humidity = sapply(weather_data$list, function(x) x$main$humidity),
    temp_min= sapply(weather_data$list, function(x) x$main$temp_min),
    temp_max= sapply(weather_data$list, function(x) x$main$temp_max),
    pressure= sapply(weather_data$list, function(x) x$main$pressure),
    wind_speed= sapply(weather_data$list, function(x) x$wind$speed),
    wind_deg= sapply(weather_data$list, function(x) x$wind$deg)
    # Add more variables as needed
  )
  # Append the extracted data to the list
  all_data <- append(all_data, list(extracted_data))
}
# Combine the list of extracted data into a single dataframe
weather_df <- do.call(rbind, all_data)
# Print the dataframe
return(weather_df)
}
```

OUTPUT FOR OPENWEATHER API

```
> cities <- c("Seoul", "Washington, D.C.", "Paris", "Suzhou")
> get_weather_forecaset_by_cities(cities)
```

	city	date_time	temperature	humidity	temp_min	temp_max	pressure	wind_speed	wind_deg
1	Seoul	2023-11-04 21:00:00	289.66	77	288.68	289.66	1021	2.08	74
2	Seoul	2023-11-05 00:00:00	289.33	79	288.66	289.33	1021	2.85	86
3	Seoul	2023-11-05 03:00:00	289.16	82	288.91	289.16	1020	2.35	89
4	Seoul	2023-11-05 06:00:00	289.65	84	289.65	289.65	1017	2.48	85
5	Seoul	2023-11-05 09:00:00	290.61	82	290.61	290.61	1015	3.35	113
6	Seoul	2023-11-05 12:00:00	292.16	76	292.16	292.16	1011	5.67	141
7	Seoul	2023-11-05 15:00:00	291.33	88	291.33	291.33	1006	7.14	148
8	Seoul	2023-11-05 18:00:00	292.49	92	292.49	292.49	1001	7.16	199
9	Seoul	2023-11-05 21:00:00	289.86	86	289.86	289.86	1003	7.68	256
10	Seoul	2023-11-06 00:00:00	288.22	68	288.22	288.22	1008	6.42	269
11	Seoul	2023-11-06 03:00:00	289.51	53	289.51	289.51	1008	8.74	256
12	Seoul	2023-11-06 06:00:00	288.05	41	288.05	288.05	1008	8.90	250
13	Seoul	2023-11-06 09:00:00	285.21	42	285.21	285.21	1010	8.23	257
14	Seoul	2023-11-06 12:00:00	282.94	56	282.94	282.94	1012	9.90	257
15	Seoul	2023-11-06 15:00:00	281.00	70	281.00	281.00	1014	9.11	286
16	Seoul	2023-11-06 18:00:00	278.81	63	278.81	278.81	1017	7.88	297
17	Seoul	2023-11-06 21:00:00	277.37	60	277.37	277.37	1020	6.12	300
18	Seoul	2023-11-07 00:00:00	277.51	56	277.51	277.51	1023	4.90	311
19	Seoul	2023-11-07 03:00:00	279.93	46	279.93	279.93	1023	3.96	307
20	Seoul	2023-11-07 06:00:00	282.26	41	282.26	282.26	1022	3.83	302
21	Seoul	2023-11-07 09:00:00	281.69	42	281.69	281.69	1024	2.72	311
22	Seoul	2023-11-07 12:00:00	280.94	41	280.94	280.94	1025	1.05	325
23	Seoul	2023-11-07 15:00:00	280.58	39	280.58	280.58	1024	0.58	324
24	Seoul	2023-11-07 18:00:00	280.09	41	280.09	280.09	1025	1.34	20
25	Seoul	2023-11-07 21:00:00	279.45	46	279.45	279.45	1024	1.43	76
26	Seoul	2023-11-08 00:00:00	280.83	39	280.83	280.83	1025	1.49	85
27	Seoul	2023-11-08 03:00:00	285.83	24	285.83	285.83	1023	1.81	75
28	Seoul	2023-11-08 06:00:00	288.19	21	288.19	288.19	1022	1.95	105
29	Seoul	2023-11-08 09:00:00	287.59	24	287.59	287.59	1022	1.02	81

WEB SCRAPING SOME OF THE REQUIRED DATASETS

```
> # Download several datasets
>
> # 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 = "D:\\data analysis\\data Viz with r\\r-capstone project\\data used\\raw_worldcities.csv")
trying URL 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_worldcities.csv'
Content type 'text/csv' length 2601845 bytes (2.5 MB)
downloaded 2.5 MB


>
> # 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 = "D:\\data analysis\\data Viz with r\\r-capstone project\\data used\\raw_seoul_bike_sharing.csv")
trying URL 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/raw_seoul_bike_sharing.csv'
Content type 'text/csv' length 595658 bytes (581 KB)
downloaded 581 KB
```

DATA WRANGLING CODE AND OUTPUT FOR REGULAR EXPRESSIONS

```
> library(stringr)
> library(dplyr)
> # Custom function to remove reference links
> remove_ref <- function(strings) {
+   # Define a pattern matching a reference link
+   ref_pattern <- "\\[\\d+\\]"  # This pattern matches references like [1], [2], etc.
+
+   # Replace all matched substrings with an empty character using str_replace_all()
+   result <- str_replace_all(strings, pattern = ref_pattern, replacement = "")
+
+   return(result)
+ }
> #TODO: Use the dplyr::mutate() function to apply the remove_ref function to the CITY and SYSTEM columns
> sub_bike_sharing_df1=sub_bike_sharing_df %>% mutate(CITY=remove_ref(CITY), SYSTEM=remove_ref(SYSTEM) )
> head(sub_bike_sharing_df1)
# A tibble: 6 × 4
  COUNTRY   CITY                    SYSTEM          BICYCLES
  <chr>     <chr>                   <chr>              <dbl>
1 Albania   Tirana                  NA                   200
2 Argentina Buenos Aires            Serttel Brasil      4000
3 Argentina Mendoza                 NA                    40
4 Argentina Rosario                 NA                   480
5 Argentina San Lorenzo, Santa Fe   Biciudad              80
6 Australia Melbourne               PBSC & 8D            676
```

MISSING VALUES HANDLING

```
> bike_sharing_df %>% filter(is.na(TEMPERATURE))
# A tibble: 0 x 14
# i 14 variables: DATE <chr>, RENTED_BIKE_COUNT <dbl>, HOUR <dbl>, TEMPERATURE <dbl>, HUMIDITY <dbl>, WIND_SPEED <dbl>, VISIBILITY <dbl>,
#   DEW_POINT_TEMPERATURE <dbl>, SOLAR_RADIATION <dbl>, RAINFALL <dbl>, SNOWFALL <dbl>, SEASONS <chr>, HOLIDAY <chr>, FUNCTIONING_DAY <chr>
>
> #TODO: Impute missing values for the TEMPERATURE column using its mean value.
> # Calculate the summer average temperature
> temp_avg= mean(bike_sharing_df$TEMPERATURE,na.rm = TRUE)
>
> # Impute missing values for TEMPERATURE column with summer average temperature
> bike_sharing_df= bike_sharing_df %>% mutate(TEMPERATURE= replace_na(TEMPERATURE , temp_avg))
>
> # Save the dataset as `seoul_bike_sharing.csv`
> print(bike_sharing_df)
# A tibble: 8,465 x 14
   DATE       RENTED_BIKE_COUNT  HOUR TEMPERATURE HUMIDITY WIND_SPEED VISIBILITY DEW_POINT_TEMPERATURE SOLAR_RADIATION RAINFALL SNOWFALL SEASONS HOLIDAY
   <chr>                  <dbl> <dbl>       <dbl>    <dbl>      <dbl>      <dbl>                 <dbl>           <dbl>    <dbl>    <dbl> <chr>   <chr>
 1 01/12/2017               254     0        -5.2       37        2.2       2000                 -17.6               0        0        0 Winter  No Holiday
 2 01/12/2017               204     1        -5.5       38        0.8       2000                 -17.6               0        0        0 Winter  No Holiday
 3 01/12/2017               173     2        -6         39        1         2000                 -17.7               0        0        0 Winter  No Holiday
 4 01/12/2017               107     3        -6.2       40        0.9       2000                 -17.6               0        0        0 Winter  No Holiday
 5 01/12/2017                78     4        -6         36        2.3       2000                 -18.6               0        0        0 Winter  No Holiday
 6 01/12/2017               100     5        -6.4       37        1.5       2000                 -18.7               0        0        0 Winter  No Holiday
 7 01/12/2017               181     6        -6.6       35        1.3       2000                 -19.5               0        0        0 Winter  No Holiday
 8 01/12/2017               460     7        -7.4       38        0.9       2000                 -19.3               0        0        0 Winter  No Holiday
 9 01/12/2017               930     8        -7.6       37        1.1       2000                 -19.8               0.01     0        0 Winter  No Holiday
10 01/12/2017               490     9        -6.5       27        0.5       1928                 -22.4               0.23     0        0 Winter  No Holiday
# i 8,455 more rows
# i 1 more variable: FUNCTIONING_DAY <chr>
# i Use `print(n = ...)` to see more rows
```

GENERATING INDICATOR COLUMNS

```
> bike_sharing_df<- bike_sharing_df %>%
+   mutate(dummy = 1)%>%
+   spread(key = SEASONS, value = dummy, fill = 0)
> # Step 2: Spread HOLIDAY
> bike_sharing_df<- bike_sharing_df %>%
+   mutate(dummy = 1)%>%spread(key = HOLIDAY, value = dummy, fill = 0)
> # Step 3: Spread FUNCTIONING_DAY
> bike_sharing_df<- bike_sharing_df %>%
+   mutate(dummy = 1)%>%spread(key = FUNCTIONING_DAY, value = dummy, fill = 0)
> # Step 4: Spread HOUR
> bike_sharing_df<- bike_sharing_df %>%
+   mutate(dummy = 1)%>%spread(key = HOUR, value = dummy, fill = 0)
> # Remove the 'dummy' column
> #bike_sharing_df <- select(bike_sharing_df, -dummy)
> print(bike_sharing_df)
# A tibble: 8,465 x 41
   DATE       RENTED_BIKE_COUNT TEMPERATURE HUMIDITY WIND_SPEED VISIBILITY DEW_POINT_TEMPERATURE SOLAR_RADIATION RAINFALL SNOWFALL Autumn Spring Summer Winter
   <chr>                  <dbl>       <dbl>    <dbl>      <dbl>      <dbl>                 <dbl>           <dbl>    <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1 01/12/2017               254        -5.2       37        2.2       2000                 -17.6               0        0        0      0      0      0      1
 2 01/12/2017               204        -5.5       38        0.8       2000                 -17.6               0        0        0      0      0      0      1
 3 01/12/2017               173        -6         39        1         2000                 -17.7               0        0        0      0      0      0      1
 4 01/12/2017               107        -6.2       40        0.9       2000                 -17.6               0        0        0      0      0      0      1
 5 01/12/2017                78        -6         36        2.3       2000                 -18.6               0        0        0      0      0      0      1
 6 01/12/2017               100        -6.4       37        1.5       2000                 -18.7               0        0        0      0      0      0      1
 7 01/12/2017               181        -6.6       35        1.3       2000                 -19.5               0        0        0      0      0      0      1
 8 01/12/2017               460        -7.4       38        0.9       2000                 -19.3               0        0        0      0      0      0      1
 9 01/12/2017               930        -7.6       37        1.1       2000                 -19.8               0.01     0        0      0      0      0      1
10 01/12/2017               490        -6.5       27        0.5       1928                 -22.4               0.23     0        0      0      0      0      1
# i 8,455 more rows
# i 27 more variables: Holiday <dbl>, `No Holiday` <dbl>, Yes <dbl>, `0` <dbl>, `1` <dbl>, `10` <dbl>, `11` <dbl>, `12` <dbl>, `13` <dbl>, `14` <dbl>,
#   `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>, `2` <dbl>, `20` <dbl>, `21` <dbl>, `22` <dbl>, `23` <dbl>, `3` <dbl>, `4` <dbl>, `5` <dbl>,
#   `6` <dbl>, `7` <dbl>, `8` <dbl>, `9` <dbl>
# i Use `print(n = ...)` to see more rows
```

SQL-QUERIES: BUSIEST BIKE RENTAL TIMES

```
dbGetQuery(conn, 'SELECT DATE, HOUR, RENTED_BIKE_COUNT
FROM SEOUL_BIKE_SHARING
WHERE (DATE, HOUR) IN (
    SELECT DATE, HOUR
    FROM (
        SELECT DATE, HOUR, MAX(RENTED_BIKE_COUNT) AS total_rentals
        FROM SEOUL_BIKE_SHARING
        GROUP BY DATE
        ORDER BY total_rentals DESC
        LIMIT 2
    ) top_date
)
  ')
```

HOURLY POPULARITY AND TEMPERATURE BY SEASONS

```
dbGetQuery(conn, "SELECT
      HOUR as rental_hour,SEASONS,
     AVG(br.TEMPERATURE) as avg_temperature,
     AVG(br.RENTED_BIKE_COUNT) as avg_bike_count
FROM
     SEOUL_BIKE_SHARING br


GROUP BY br.SEASONS,rental_hour

ORDER BY
     avg_bike_count DESC
LIMIT 10
")
```

RENTAL SEASONALITY

```
dbGetQuery(conn, "SELECT
    br.SEASONS as season,
    br.HOUR as rental_hour,
    AVG(br.RENTED_BIKE_COUNT) as avg_bike_count,
    MIN(br.RENTED_BIKE_COUNT) as min_bike_count,
    MAX(br.RENTED_BIKE_COUNT) as max_bike_count,
    SQRT(AVG(br.RENTED_BIKE_COUNT * br.RENTED_BIKE_COUNT) - AVG(br.RENTED_BIKE_COUNT) * AVG(br.RENTED_BIKE_COUNT)) as std_dev_bike_count
FROM
    SEOUL_BIKE_SHARING br
GROUP BY
    br.SEASONS, br.HOUR
ORDER BY
    season, rental_hour
")
```

WEATHER SEASONALITY

```
dbGetQuery(conn, "SELECT
    br.SEASONS as season,
    AVG(br.TEMPERATURE) as avg_temperature,
    AVG(br.HUMIDITY) as avg_humidity,
    AVG(br.WIND_SPEED) as avg_wind_speed,
    AVG(br.VISIBILITY) as avg_visibility,
    AVG(br.DEW_POINT_TEMPERATURE) as avg_dew_point_temperature,
    AVG(br.SOLAR_RADIATION) as avg_solar_radiation,
    AVG(br.RAINFALL) as avg_rainfall,
    AVG(br.SNOWFALL) as avg_snowfall,
    AVG(br.rented_bike_count) as avg_bike_count
FROM
    SEOUL_BIKE_SHARING br

GROUP BY SEASONS
ORDER BY
    avg_bike_count DESC;
")
```

BIKE-SHARING INFO IN SEOUL

```
dbGetQuery(conn, "SELECT
     wc.CITY,
     wc.COUNTRY,
     wc.LAT,
     wc.LNG,
     wc.POPULATION,
     SUM(bss.BICYCLES) as total_bikes_in_seoul
FROM
     WORLD_CITIES wc,
     BIKE_SHARING_SYSTEMS bss
WHERE
     wc.CITY = 'Seoul' AND
     wc.CITY = bss.CITY
GROUP BY
     wc.CITY, wc.COUNTRY, wc.LAT, wc.LNG, wc.POPULATION;
")
```

CITIES SIMILAR TO SEOUL

```
dbGetQuery(conn, "SELECT
     wc.CITY,
     wc.COUNTRY,
     wc.LAT,
     wc.LNG,
     wc.POPULATION,
     bss.BICYCLES
FROM
     WORLD_CITIES wc
JOIN
     BIKE_SHARING_SYSTEMS bss ON wc.CITY = bss.CITY
WHERE
     bss.BICYCLES BETWEEN 15000 AND 20000
")
```

GGPLOT CODE SNIPPETS:BIKE RENTAL VS. DATE

```
# Create the scatter plot
scatter_plot <- ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT)) +
  geom_point(alpha = 0.5) +  # Adjust opacity with alpha parameter
  labs(title="Bike rental vs. Date",x = "Date", y = "Rented Bike Count")  # Add labels for x and y axes


# Print the scatter plot
print(scatter_plot)
```

BIKE RENTAL VS. DATETIME

```
# Create the time series plot with color encoding by HOURS
time_series_plot <- ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT, color = as.factor(HOUR))) +
  geom_point(alpha = 0.5) +  # Adjust opacity with alpha parameter
  labs( title ="Bike rental vs Datetime" ,x = "Date", y = "Rented Bike Count") +  # Add labels for x and y axes
  scale_color_discrete(name = "Hours")  # Add legend for hours


# Print the time series plot
print(time_series_plot)
```

BIKE RENTAL HISTOGRAM

```
histogram_density_plot <- ggplot(seoul_bike_sharing, aes(x = RENTED_BIKE_COUNT, y = ..density..)) +
  geom_histogram(binwidth = 5, fill = "white", color = "black") +
  geom_density(color = "blue", alpha = 0.3) +
  labs(title="Bike rental histogram",x = "RENTED_BIKE_COUNT", y = "Density")

# Print the histogram with kernel density plot
print(histogram_density_plot)
```

BOXPLOTS OF RENTED_BIKE_COUNT VS. HOUR GROUPED BY SEASONS.

```
# Create the boxplots
boxplot_display <- ggplot(seoul_bike_sharing, aes(x = HOUR, y = RENTED_BIKE_COUNT, fill = as.factor(HOUR))) +
  geom_boxplot() +
  facet_wrap(~SEASONS, ncol = 2) +
  labs(title="RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS",x = "Hour", y = "Rented Bike Count") +
  scale_fill_discrete(name = "Hour")

# Print the boxplot display
print(boxplot_display)
```