Applied Data Science with R Capstone project

Sean Michael Gunawan

July 22 2022

Outline

- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Introduction

- The case assumed that the learner has just been hired by an AI – powered weather data analysis company as a data scientist.
- The data used in this case are Seoul Bike Sharing Demand Dataset, OpenWeather API Data, Global Bike Sharing Sytems Dataset, and World Cities Data.
- The end goal of this project is to combine our analysis results and create a dashboard displaying an interactive map and visualization of the current weather and the 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

Methodology

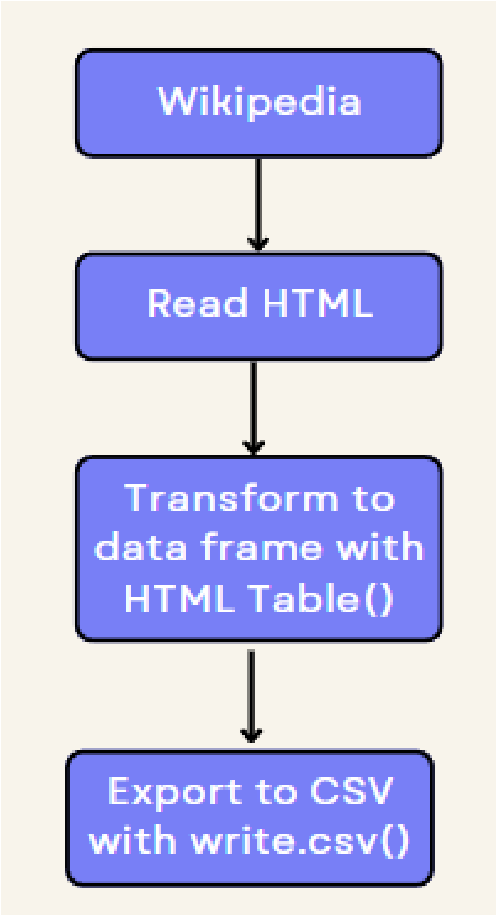
Data collection

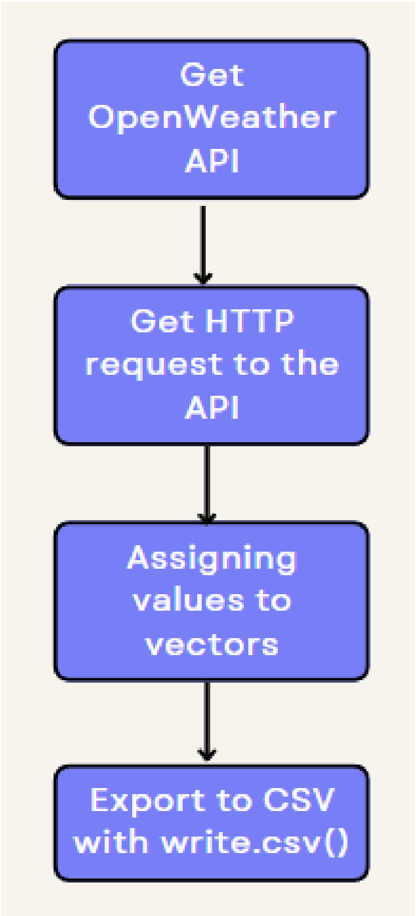
- The datasets were collected through 3 different ways:

a. HTTP Requests for OpenWeather,

b. Webscraping for Bike Sharing Systems (Wikipedia), and

c. File Downloads for Historical Bike Demand (IBM Cloud Storage).

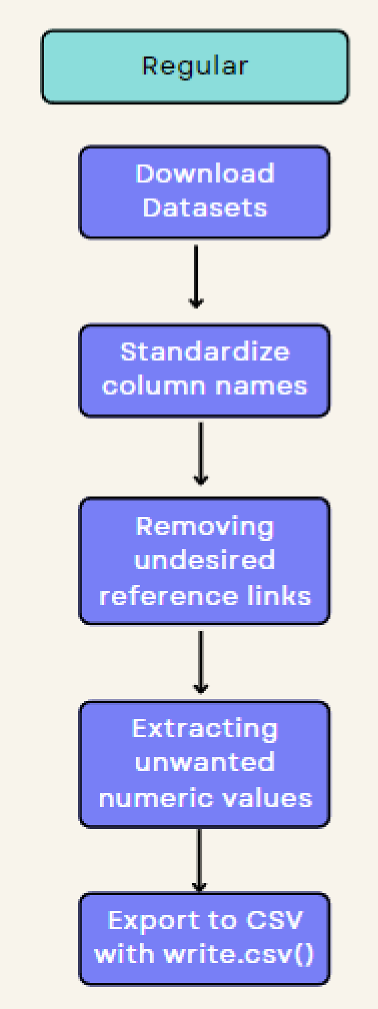


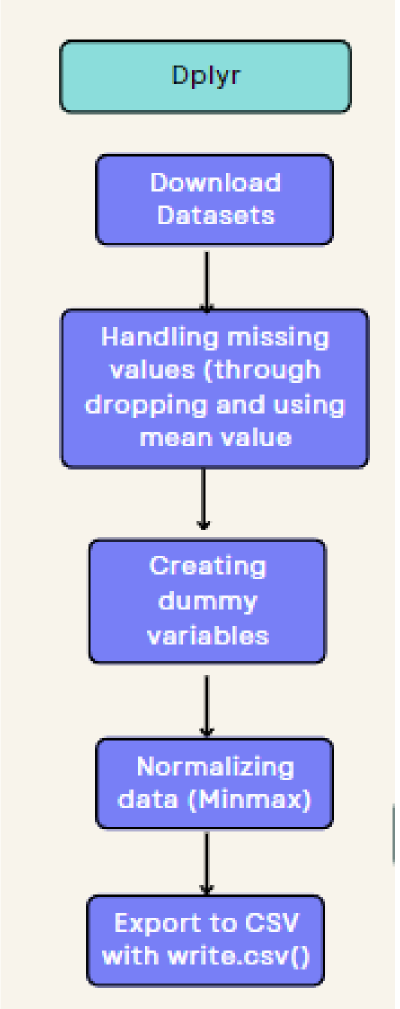


Data wrangling

- Next, the data that we collects go through data wrangling, in order to improve the data quality. In this case, we use 2 ways, with regular expressions and dplyr package.

- Add the screenshots of data wrangling code cell and output for regular expressions, missing values handling, generating indicator columns to the Appendix section for peer-review





EDA with SQL

- Next, we do EDA through SQL queries using RODBC. We determine many values, such as:

1. Total records in Seoul Bike Sharing Dataset,
2. Operational Hours of rented bike count,
3. Weather outlook for the next 3 hours,
4. Seasons that are included in the dataset,
5. First and last dates in the dataset,
6. Which date and hour has the highest rentals,
7. Average of hourly temperature and bike rentals per hour over each season,
8. Weather seasonality,
9. Total bike count and city info for Seoul, and
10. Comparable cities with total bike counts starting from 15,000 – 20,000.

EDA with data visualization

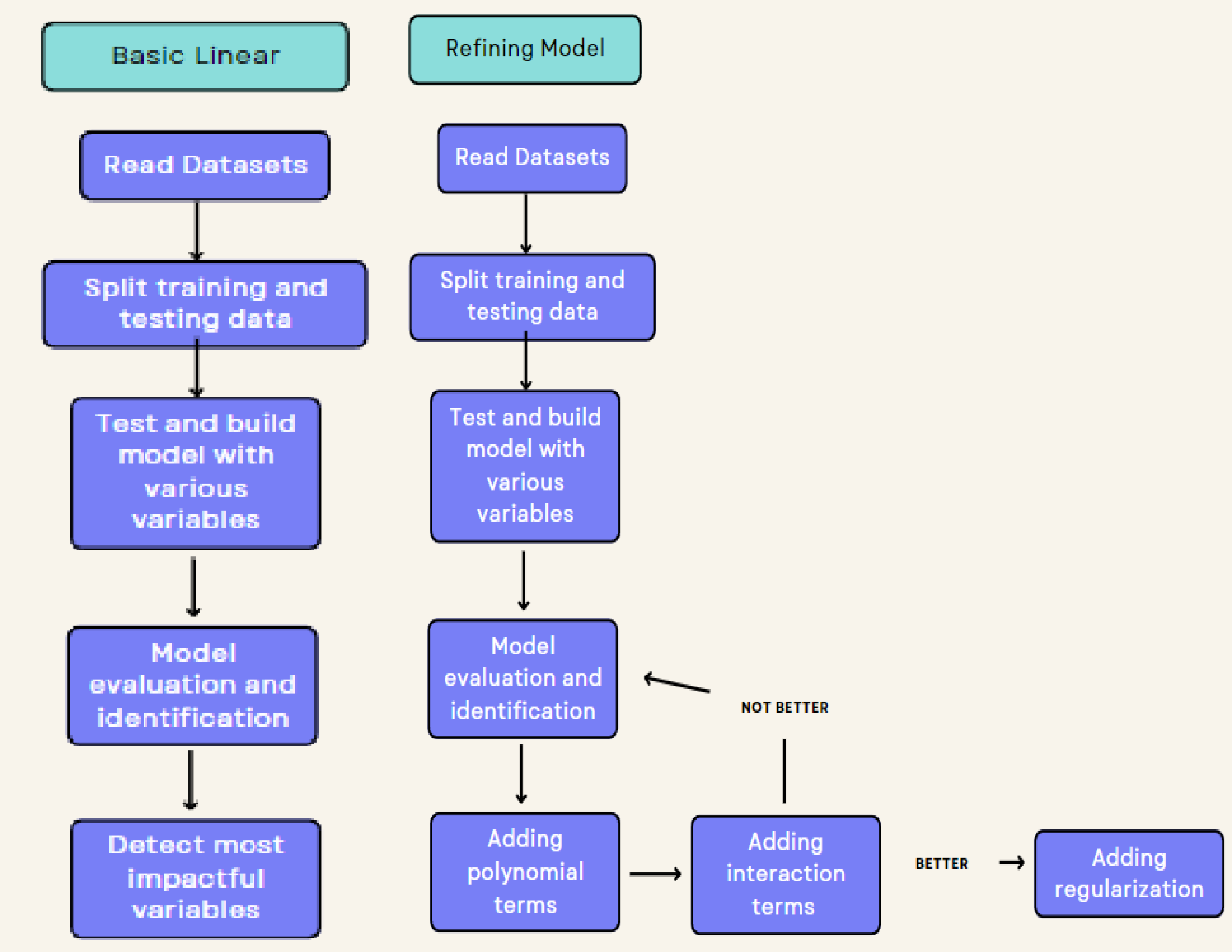
- Next, we begin visualizing the data through ggplot2. In here, we create:

1. Scatter plot of Rented Bike Count and Date,
2. Scatter plot of Rented Bike Count and Date with Hour as the colour,
3. Histogram overlaid with a kernel density curve
4. Scatter plot of Rented Bike Count and Temperature by Seasons, and
5. Boxplots of Rented Bike Count and Hour by Seasons.

- Each of the graph are used to search for correlations between the variables used (such as Rented Bike Count and Date).

Predictive analysis

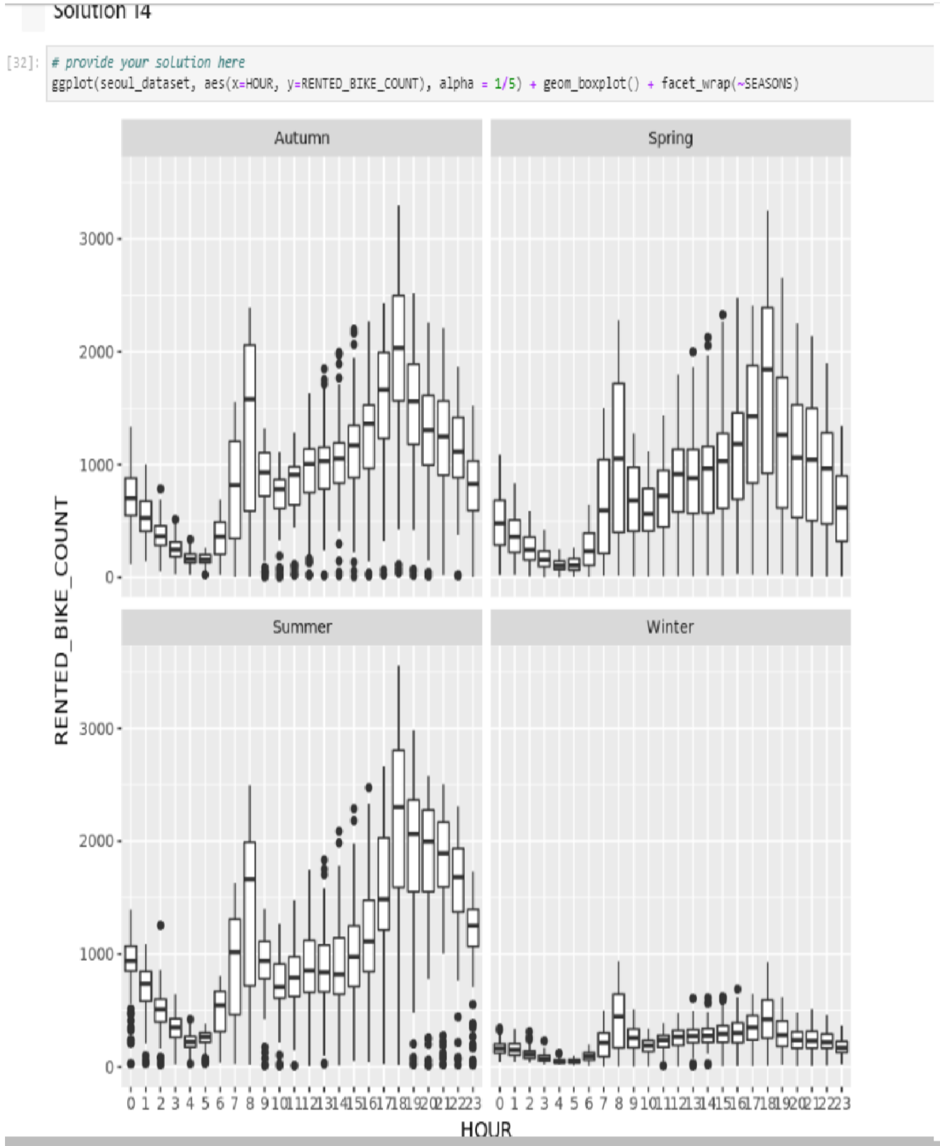
- Next, we create, compare, and find the best model to use. Here, we use linear regression to find out which variables has the most impact. After that, we add polynomial terms to handle non – linear data. Then, we add interaction terms to find correlations between response variables and predictor variables. At last, we add regularization to normalize the model.

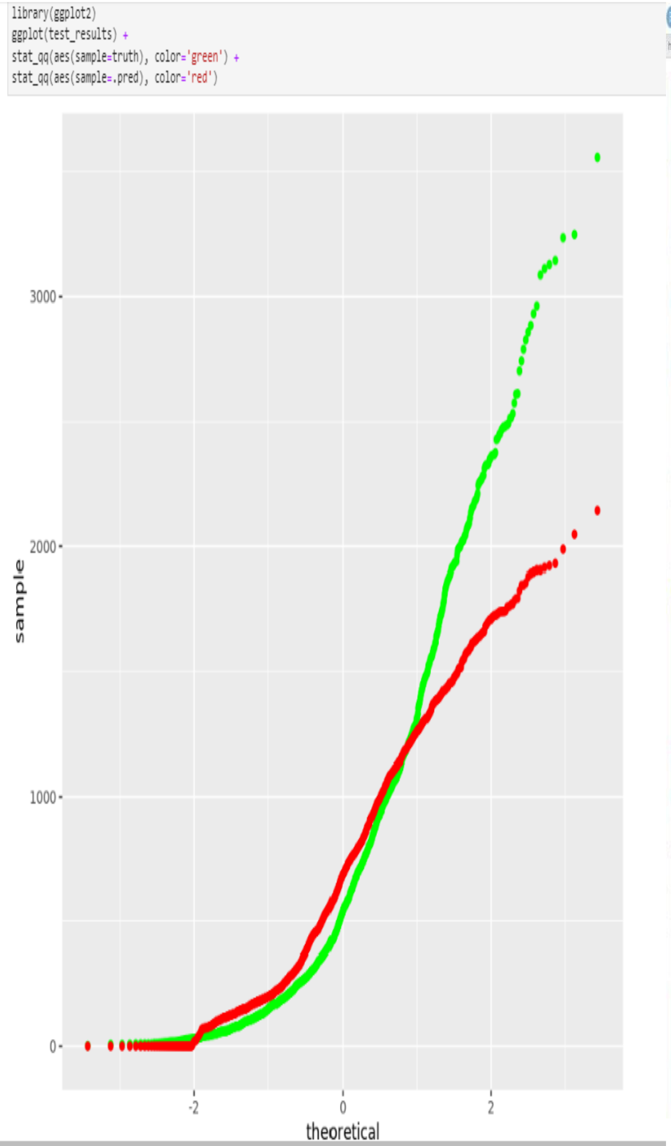


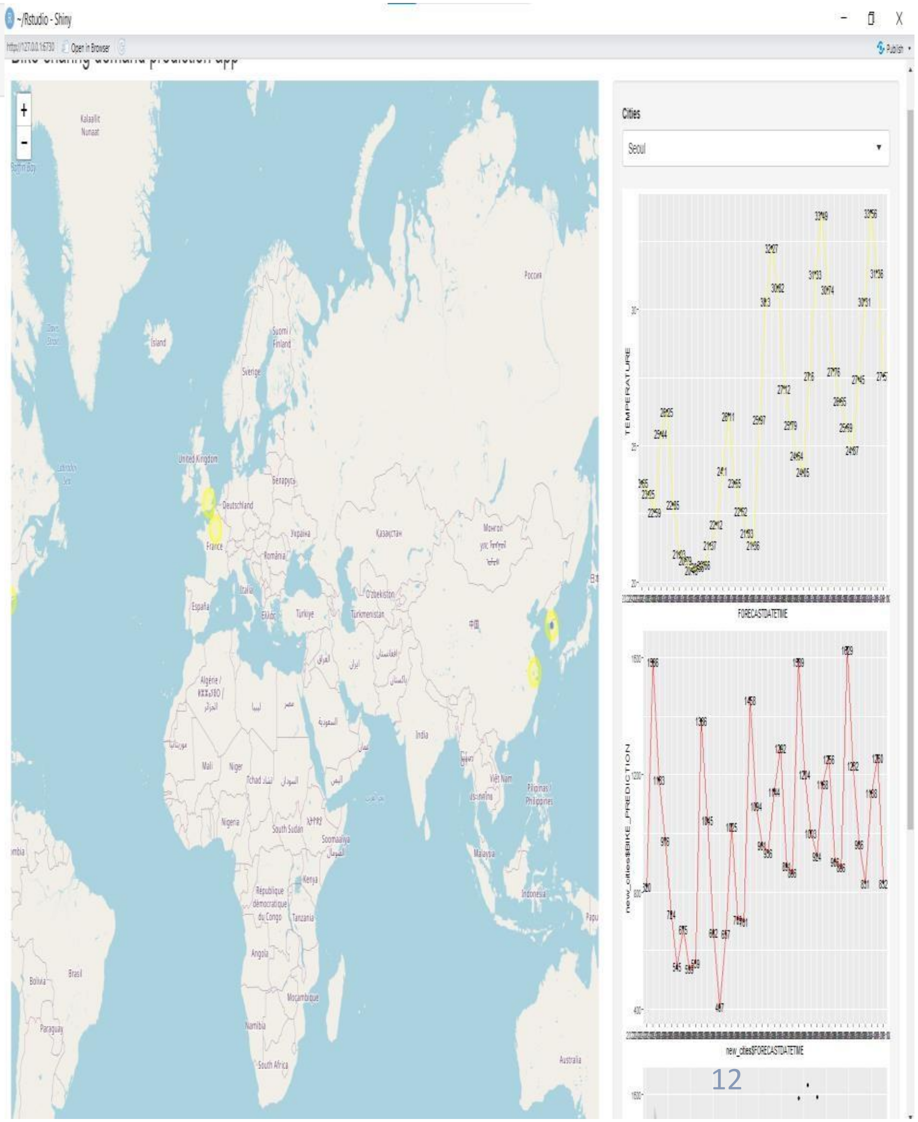
Build a R Shiny dashboard

- Next, we build a dashboard that points out 5 cities with labels such as weather. We also inserted graph visualization of FORECASTDATETIME and TEMPERATURE, FORECASTDATETIME and BIKE_PREDICTION, and HUMIDITY and BIKE_PREDICTION.

Results


Solution 14
[32]:
# provide your solution here
ggplot(seoul_dataset, aes(x=HOUR, y=RENTED_BIKE_COUNT), alpha = 1/5) + geom_boxplot() + facet_wrap(~SEASONS)
Autumn
Spring
Summer
Winter
RENTED_BIKE_COUNT
HOUR


library(ggplot2)
ggplot(test_results) +
stat_qq(aes(sample=truth), color='green') +
stat_qq(aes(sample=.pred), color='red')
sample
theoretical


~/Rstudio - Shiny
Cities
Seoul
TEMPERATURE
FORECASTDATETIME
new_cities$BIKE_PREDICTION
new_cities$FORECASTDATETIME

EDA with SQL

Busiest bike rental times

The busiest bike rental times could be search through selecting date and hour, then set the max rented bike count at the time.

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

[7]:
```
# provide your solution here
query6 <- paste("SELECT DATE, HOUR FROM SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT = (SELECT MAX(RENTED_BIKE_COUNT) FROM SEOUL_BIKE_SHARING)")
most_q <- sqlQuery(conn,query6)
most_q
```

A data.frame: 1 × 2

	DATE	HOUR
	<fct>	<int>
1	19/06/2018	18

Hourly popularity and temperature by seasons

Could be found through selecting hour, average temperature, average rented bike count, then grouping it by hour.

Solution 7

[8]:
```
# provide your solution here
query7 <- paste("SELECT HOUR, SEASONS, AVG(TEMPERATURE) AS HOURLY_TEMP, AVG(RENTED_BIKE_COUNT) AS HOURLY_RENTED FROM SEOUL_BIKE_SHARING
GROUP BY HOUR, SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC LIMIT 10")
hourly_q <- sqlQuery(conn,query7)
hourly_q
```

A data.frame: 10 × 4

	HOUR	SEASONS	HOURLY_TEMP	HOURLY_RENTED
	<int>	<fct>	<dbl>	<int>
1	18	Summer	29.38696	2135
2	18	Autumn	16.03086	1983
3	19	Summer	28.27283	1889
4	20	Summer	27.06630	1801
5	21	Summer	26.27826	1754
6	18	Spring	15.97222	1689
7	22	Summer	25.69891	1567
8	17	Autumn	17.27778	1562
9	17	Summer	30.07500	1526
10	19	Autumn	15.06049	1515

Rental Seasonality

Could be found by selecting hour, seasons and rented bike count. We apply min, max, and standard deviation to rented bike count to know the rental averages. Then, we group by seasons.

Solution 8

```
# provide your solution here
query8 <- paste("SELECT HOUR, AVG(RENTED_BIKE_COUNT) AS HOURLY_RENTED, MIN(RENTED_BIKE_COUNT) AS MIN, MAX(RENTED_BIKE_COUNT) AS MAX, STDDEV(RENTED_BIKE_COUNT) AS STD,
SEASONS FROM SEOUL_BIKE_SHARING
GROUP BY HOUR, SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC LIMIT 10")
rental_q <- sqlQuery(conn, query8)
rental_q
```

A data.frame: 10 × 6

	HOUR	HOURLY_RENTED	MIN	MAX	STD	SEASONS
	<int>	<int>	<int>	<int>	<dbl>	<fct>
1	18	2135	17	3556	884.0829	Summer
2	18	1983	40	3298	778.4414	Autumn
3	19	1889	18	2984	728.8799	Summer
4	20	1801	10	2579	662.2163	Summer
5	21	1754	17	2505	596.1374	Summer
6	18	1689	22	3251	898.8971	Spring
7	22	1567	16	2309	516.6434	Summer
8	17	1562	23	2432	554.3165	Autumn
9	17	1526	25	2664	608.7917	Summer
10	19	1515	19	2518	571.1497	Autumn

Weather Seasonality

Could be found by averaging weather variables, such as temperature, humidity, snowfall, rainfall, etc. Then, we group it by seasons.

Solution 9

```
query9 <- paste("SELECT AVG(RENTED_BIKE_COUNT) AS AVG_BIKE,
AVG(TEMPERATURE) AS AVG_TEMP,
AVG(HUMIDITY) AS AVG_HUMID,
AVG(WIND_SPEED) AS AVG_WIND,
AVG(VISIBILITY) AS AVG_VISIB,
AVG(DEW_POINT_TEMPERATURE) AS AVG_DEW,
AVG(SOLAR_RADIATION) AS AVG_SOLAR,
AVG(RAINFALL) AS AVG_RAINFALL,
AVG(SNOWFALL) AS AVG_SNOWFALL,
SEASONS FROM SEOUL_BIKE_SHARING GROUP BY SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC")
weather_seaq <- sqlQuery(conn, query9)
weather_seaq

# provide your solution here
```

A data.frame: 4 × 10

	AVG_BIKE	AVG_TEMP	AVG_HUMID	AVG_WIND	AVG_VISIB	AVG_DEW	AVG_SOLAR	AVG_RAINFALL	AVG_SNOWFALL	SEASONS
	<int>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	1034	26.587274	64	1.609420	1501	18.750136	0.7612545	0.25348732	0.00000000	Summer
2	924	13.821167	59	1.492101	1558	5.150594	0.5227827	0.11765617	0.06350026	Autumn
3	746	13.021389	58	1.857778	1240	4.091389	0.6803009	0.18694444	0.00000000	Spring
4	225	-2.540463	49	1.922685	1445	-12.416667	0.2981806	0.03282407	0.24750000	Winter

Bike-sharing info in Seoul

Could be found by using JOIN function and combining both BIKE_SHARING_SYSTEMS and WORLD_CITIES data, with CITY_ASCII and CITY as the key.

Solution 10

```
# provide your solution here
query10 <- paste("SELECT BS.BICYCLES, WC.CITY_ASCII, WC.COUNTRY, WC.LAT, WC.LNG, WC.POPULATION FROM BIKE_SHARING_SYSTEMS AS BS,
WORLD_CITIES AS WC WHERE WC.CITY_ASCII = BS.CITY AND WC.CITY = 'Seoul'")
total_bike <- sqlQuery(conn, query10)
total_bike
```

A data.frame: 1 × 6

	BICYCLES	CITY_ASCII	COUNTRY	LAT	LNG	POPULATION
	<int>	<fct>	<fct>	<dbl>	<dbl>	<int>
1	20000	Seoul	Korea, South	37.58	127	21794000

Cities similar to Seoul

Could be found by using JOIN, and stating some constraints, such as number of bicycles between 15000 and 20000.

Solution 11

```
# provide your solution here
query11 <- ("SELECT BS.BICYCLES, WC.CITY_ASCII, WC.COUNTRY, WC.LAT, WC.LNG, WC.POPULATION FROM BIKE_SHARING_SYSTEMS AS BS,
WORLD_CITIES AS WC WHERE WC.CITY_ASCII = BS.CITY AND BICYCLES BETWEEN 15000 AND 20000 ORDER BY BICYCLES DESC")
city_names <- sqlQuery(conn, query11)
city_names
```

A data.frame: 7 × 6

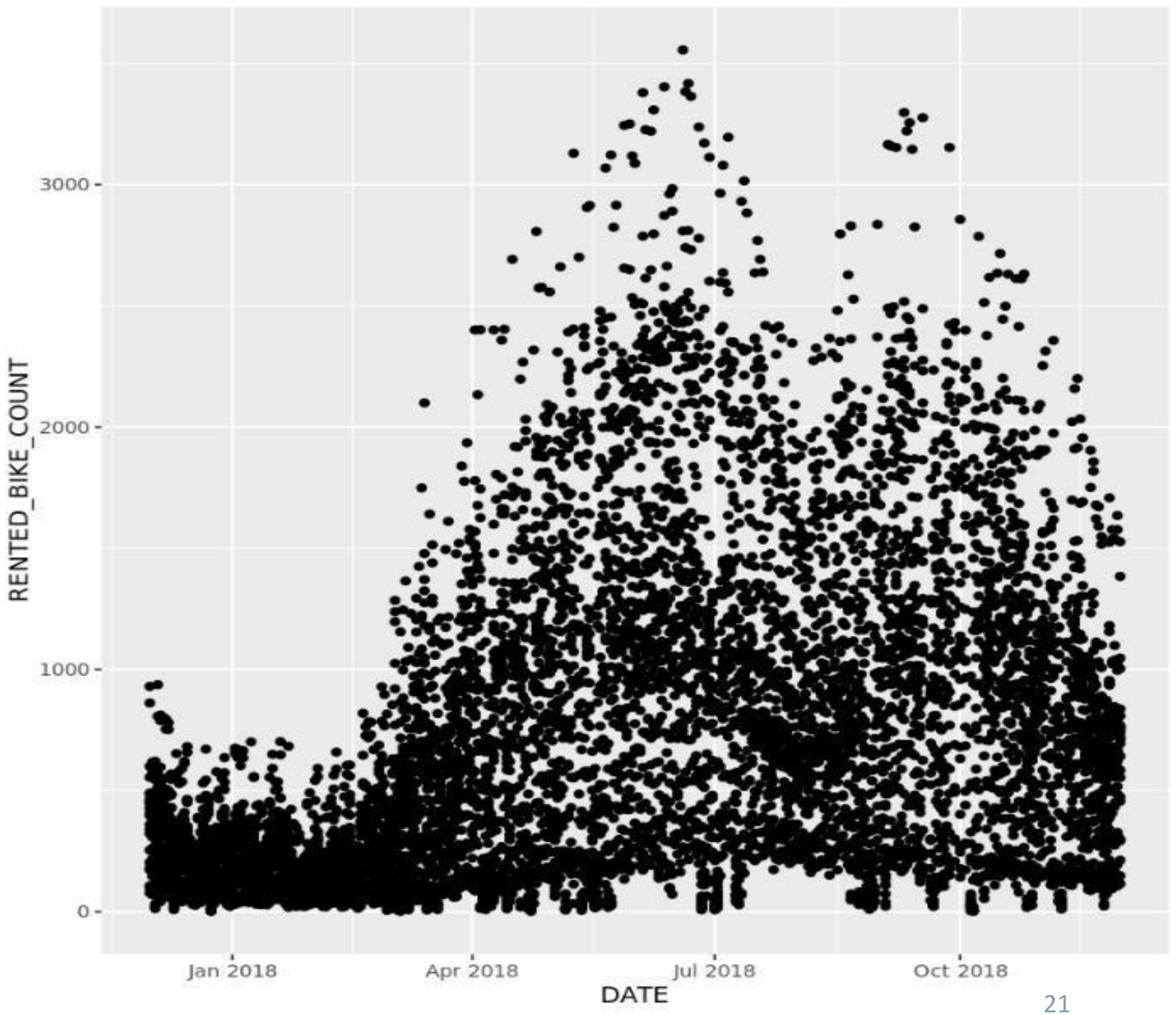
	BICYCLES	CITY_ASCII	COUNTRY	LAT	LNG	POPULATION
	<int>	<fct>	<fct>	<dbl>	<dbl>	<int>
1	20000	Seoul	Korea, South	37.58	127.00	21794000
2	20000	Weifang	China	36.71	119.10	9373000
3	20000	Xi'an	China	34.26	108.90	7135000
4	20000	Zhuzhou	China	27.84	113.14	3855609
5	19165	Shanghai	China	31.16	121.46	22120000
6	16000	Beijing	China	39.90	116.39	19433000
7	15000	Ningbo	China	29.87	121.54	7639000

EDA with Visualization

Bike rental vs. Date

The graph explains that there’s a fluctuation of the rented bike count amount, starting from April 2018 until October 2018, with some fluctuations.

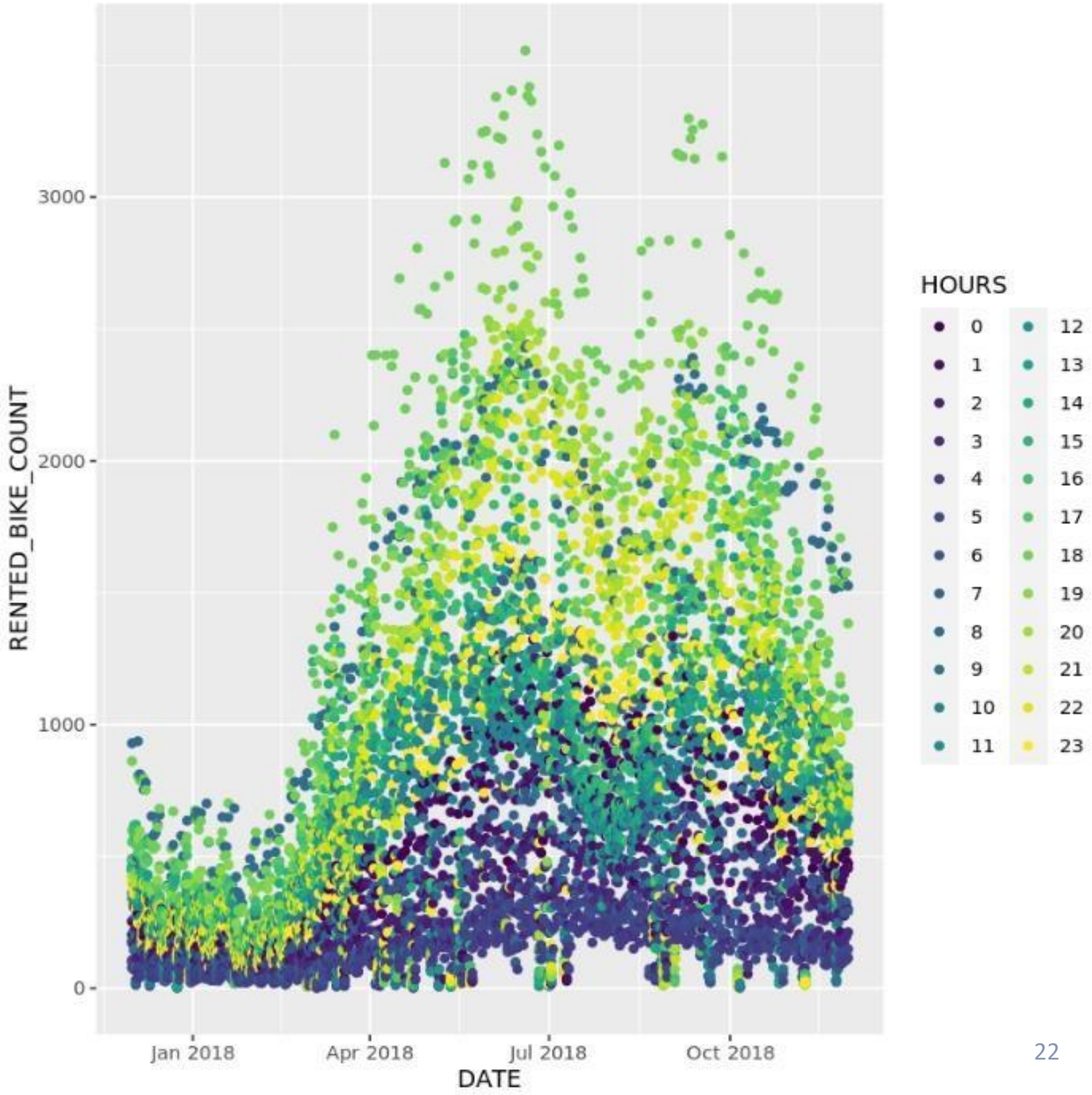
```
# provide your solution here
qplot(DATE, RENTED_BIKE_COUNT, data = seoul_dataset, alpha = I(50))
```



Bike rental vs. Datetime

As we dive deeper, by adding hour as the color in the graph, we see here that the usage of rented bike count happens at around 17 – 19 and 6 – 7 (the colour green and dark blue filling most of the graph).

```
ggplot(seoul_dataset, aes(x = DATE, y = RENTED_BIKE_COUNT, color = HOUR), alpha = I(50)) + geom_point() + labs(color = "HOURS")
```



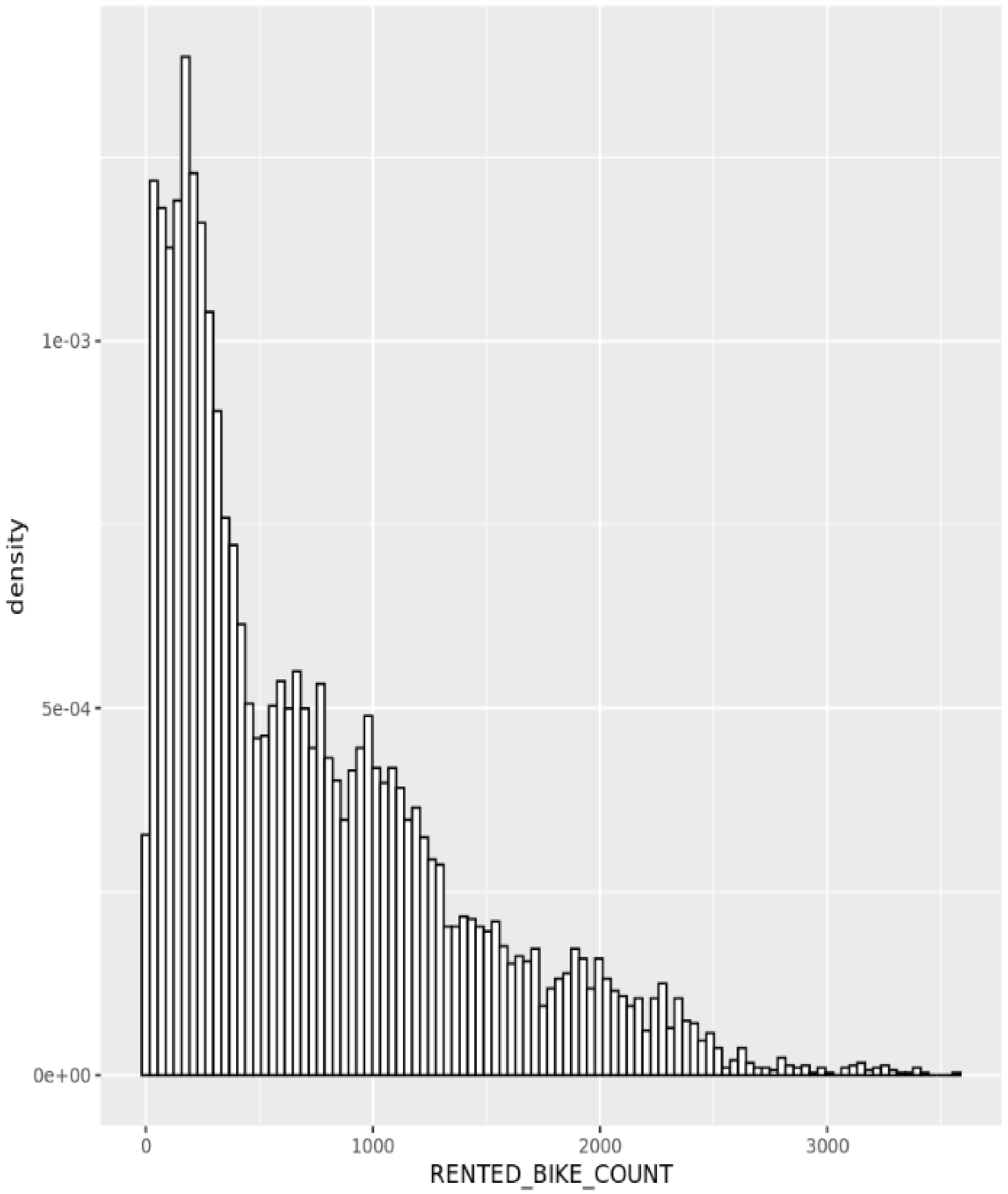
Bike rental histogram

We can see from the histogram that most of the time there are relatively few bikes rented. Indeed, the 'mode', or most frequent number 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.

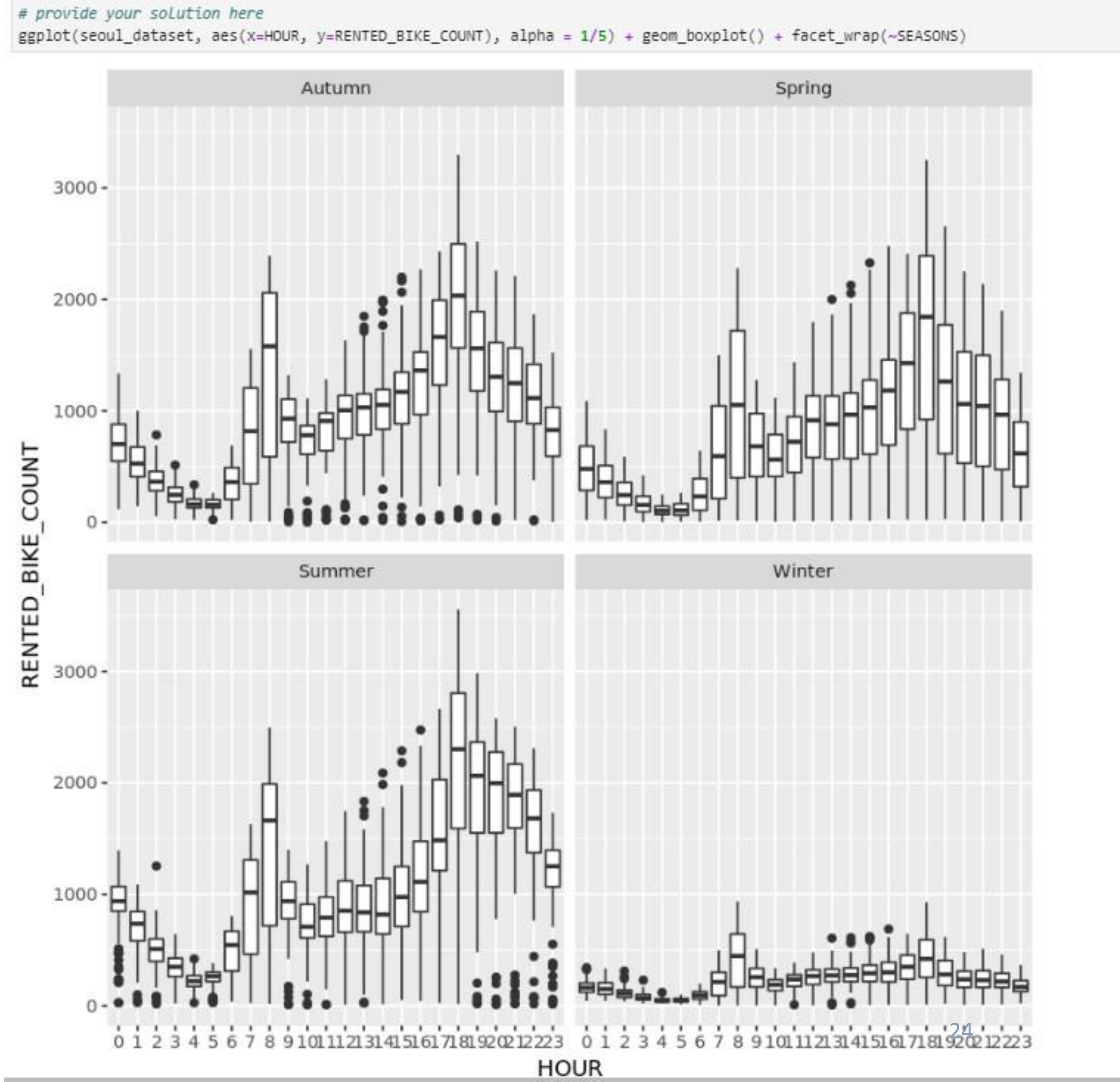
Interestingly, judging from the tail of the distribution, on rare occasions there are many more bikes rented out than usual.

```r
# provide your solution here
ggplot(seoul_dataset, aes(x = RENTED_BIKE_COUNT)) + geom_histogram(binwidth = 35, color = "black", alpha = I(50), fill = I("white"), aes(y = ..density..))
```



Daily total rainfall and snowfall

As we can see in the graph, 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 pm.

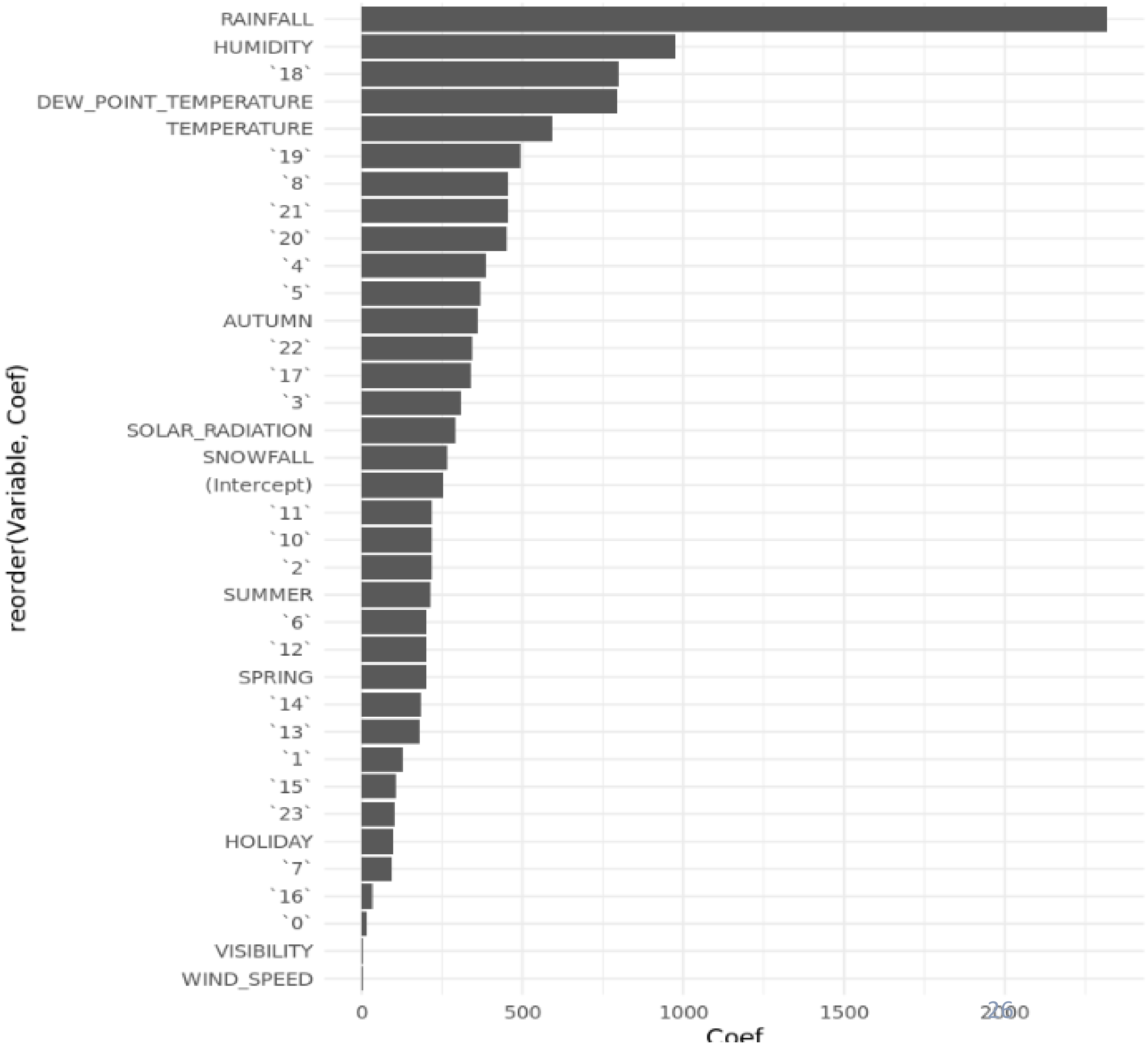


Predictive analysis

Ranked coefficients

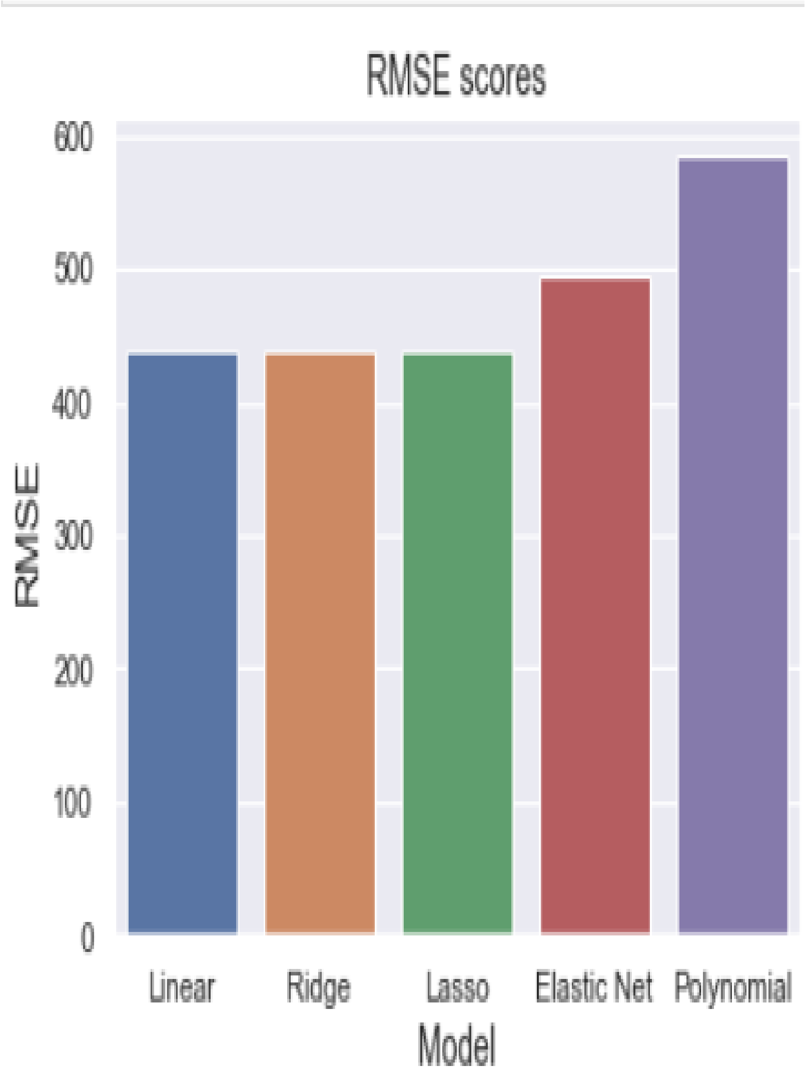
As we ranked these coefficients, we could see the most impactful variable is rainfall. Some of the other variables, such as wind speed and visibility, doesn’t impact the rented bike count at all.

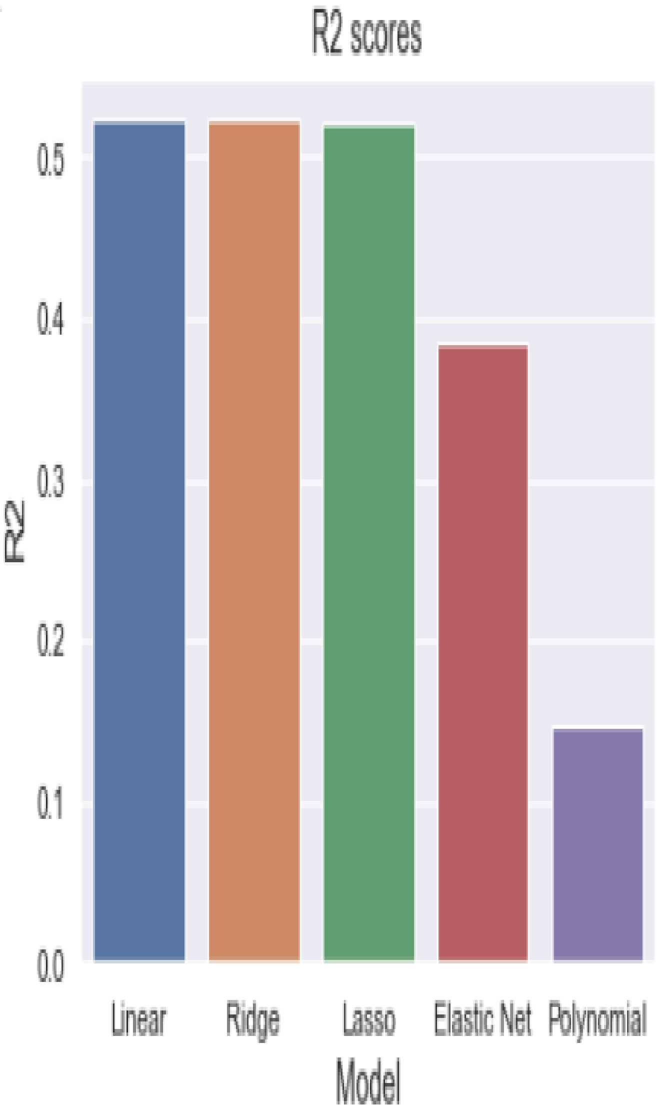
```
ggplot(data=coefs_sorted, aes(x= reorder(Variable,Coef),Coef)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal()
```



Model evaluation

As we can see from the bar, Polynomial has the highest RMSE scores, but it has the lowest R2 score. So, in order to valuate the best model, we can use some combination of poly and other model.



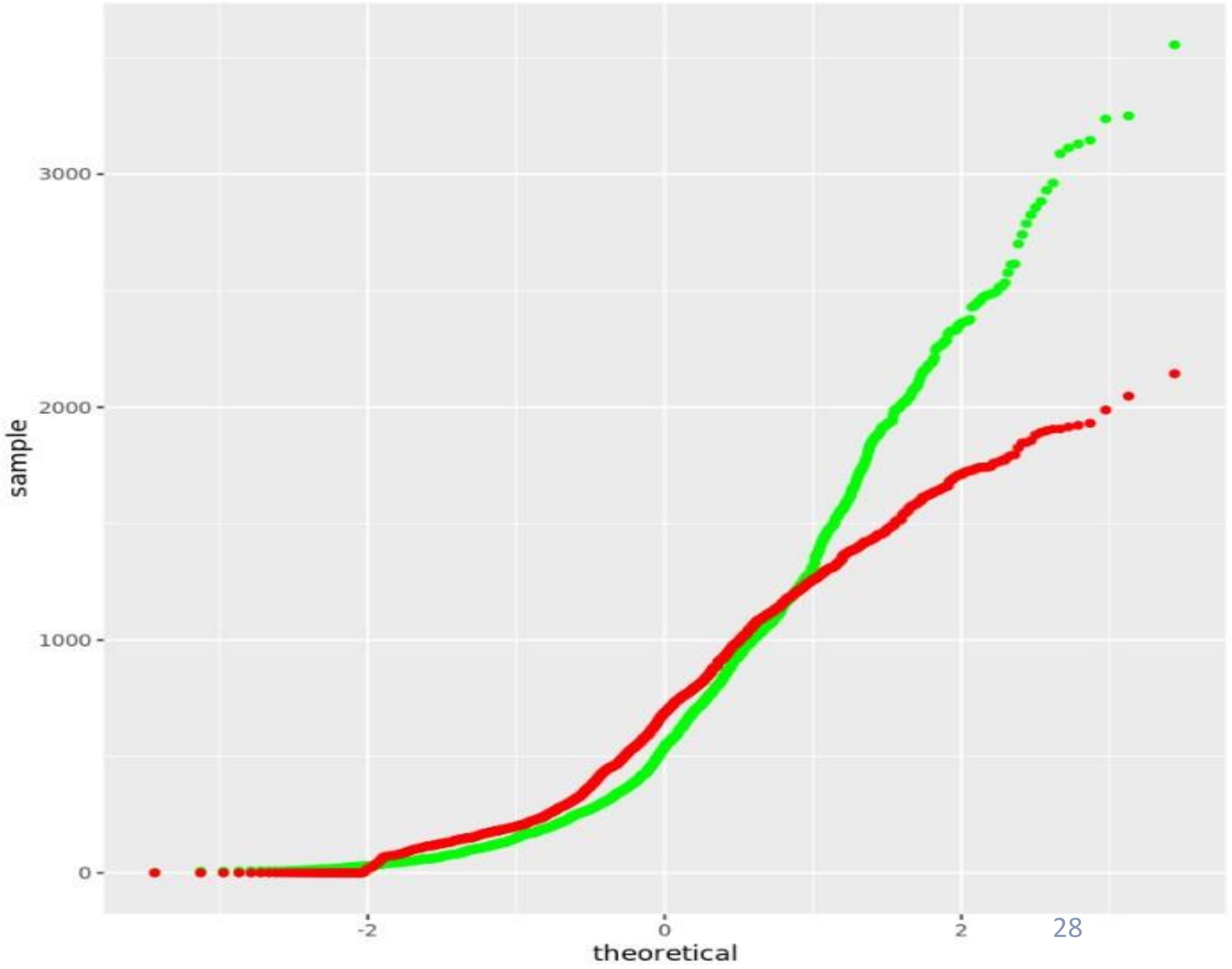


Find the best performing model

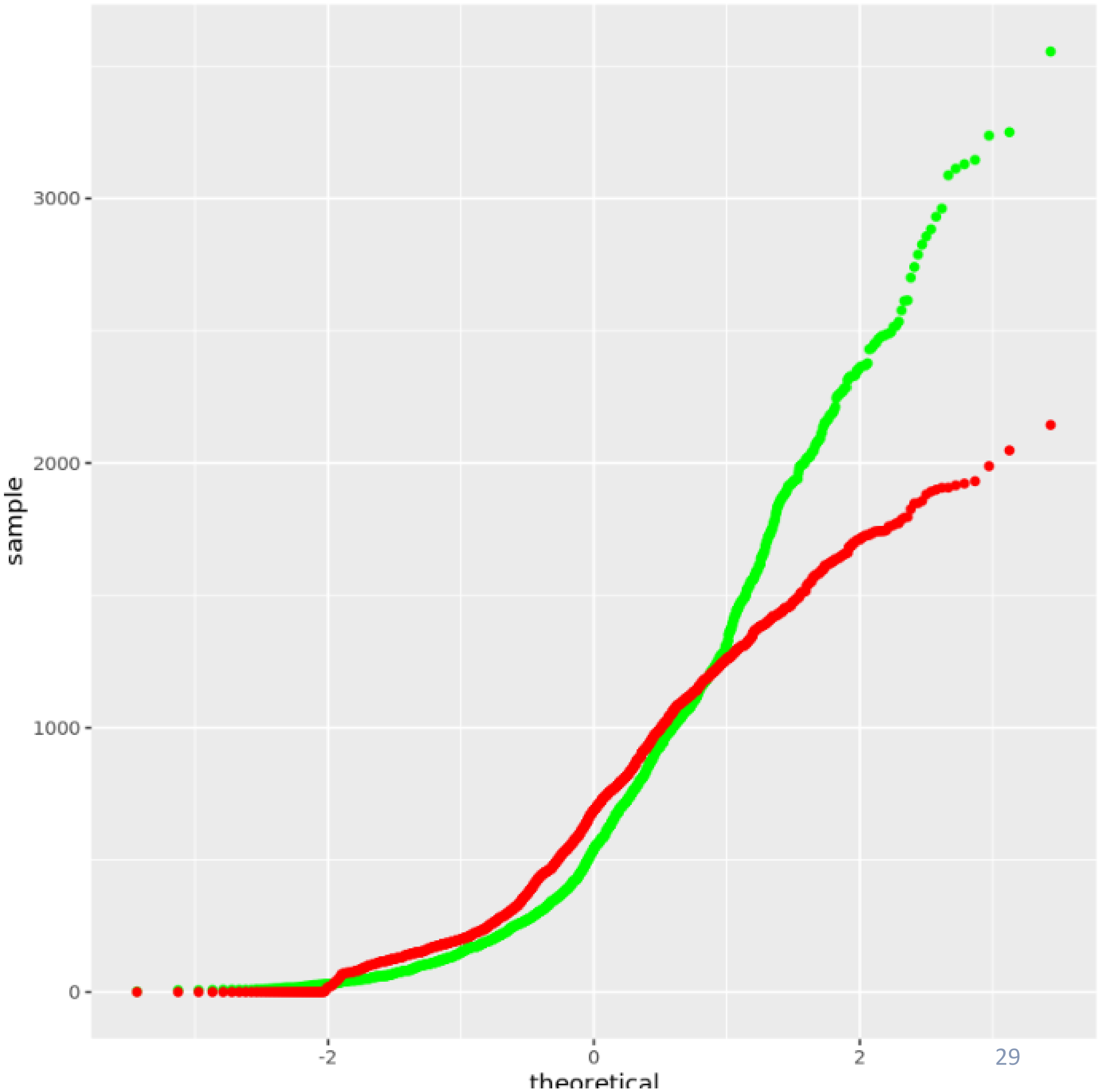
Model:

fit(RENTED_BIKE_COUNT ~ poly(TEMPERATURE,6) + poly(RAINFALL*HUMIDITY,8) + poly(HUMIDITY*TEMPERATURE, 6) + poly(DEW_POINT_TEMPERATURE*TEMPERATURE,6) + AUTUMN + SUMMER + SPRING + HOLIDAY + poly(SOLAR_RADIATION,2) + poly(SNOWFALL*TEMPERATURE, 2), data = train_data)

```
library(ggplot2)
ggplot(test_results) +
stat_qq(aes(sample=truth), color='green') +
stat_qq(aes(sample=.pred), color='red')
```



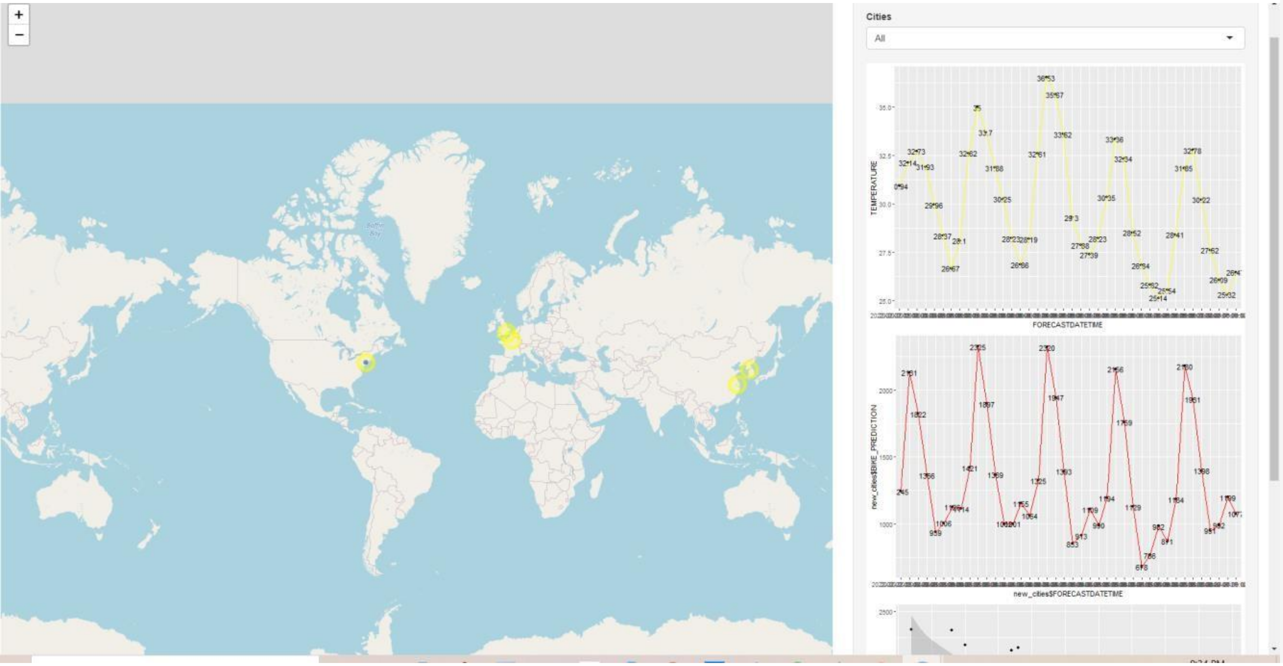
Q-Q plot of the best model

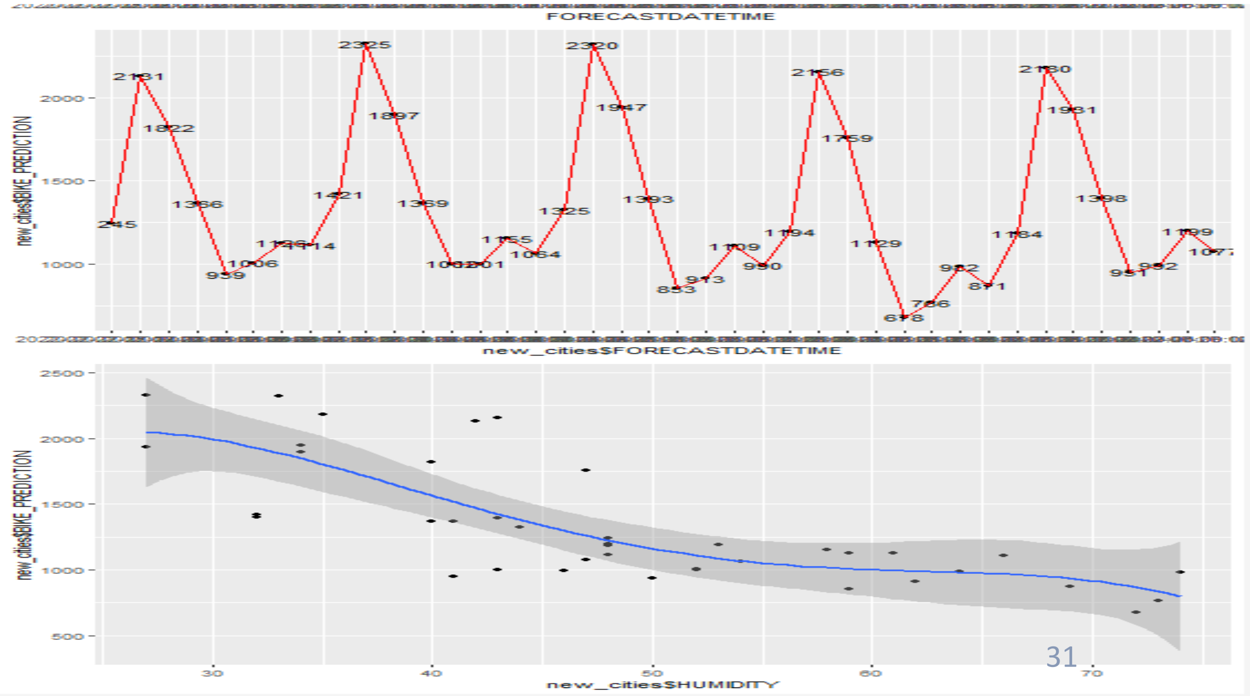


Dashboard

Dashboard All World

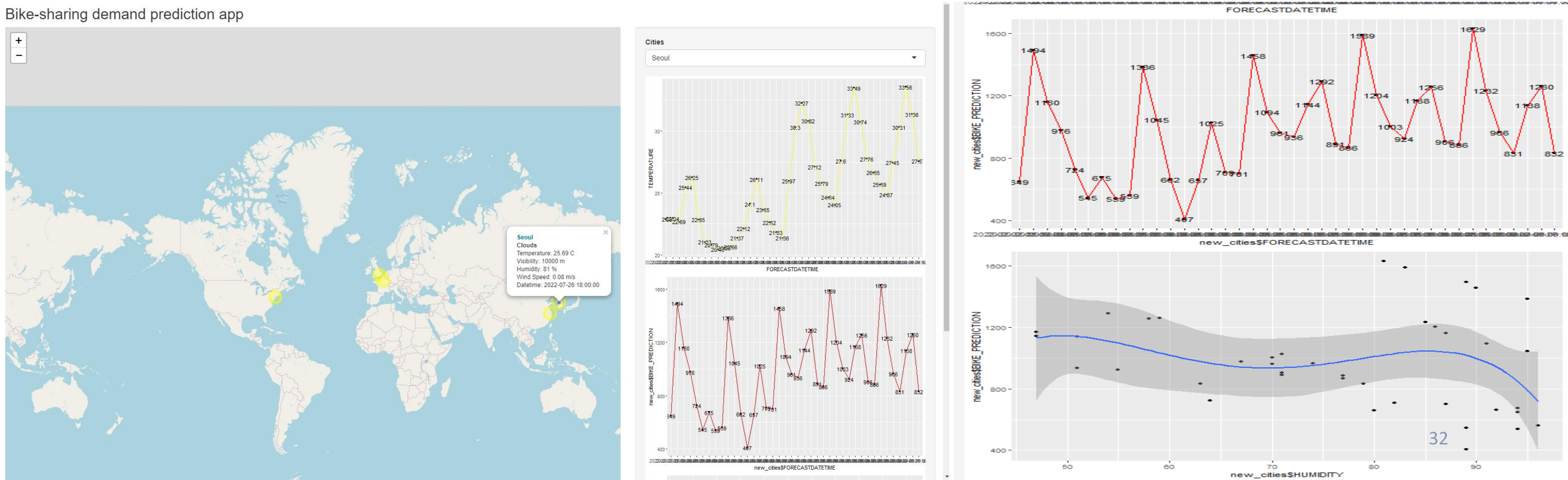
Below are the max bike prediction for All Around the world. The graph consists of FORECASTDATETIME, BIKE_PREDICTION, HUMIDITY, and TEMPERATURE.





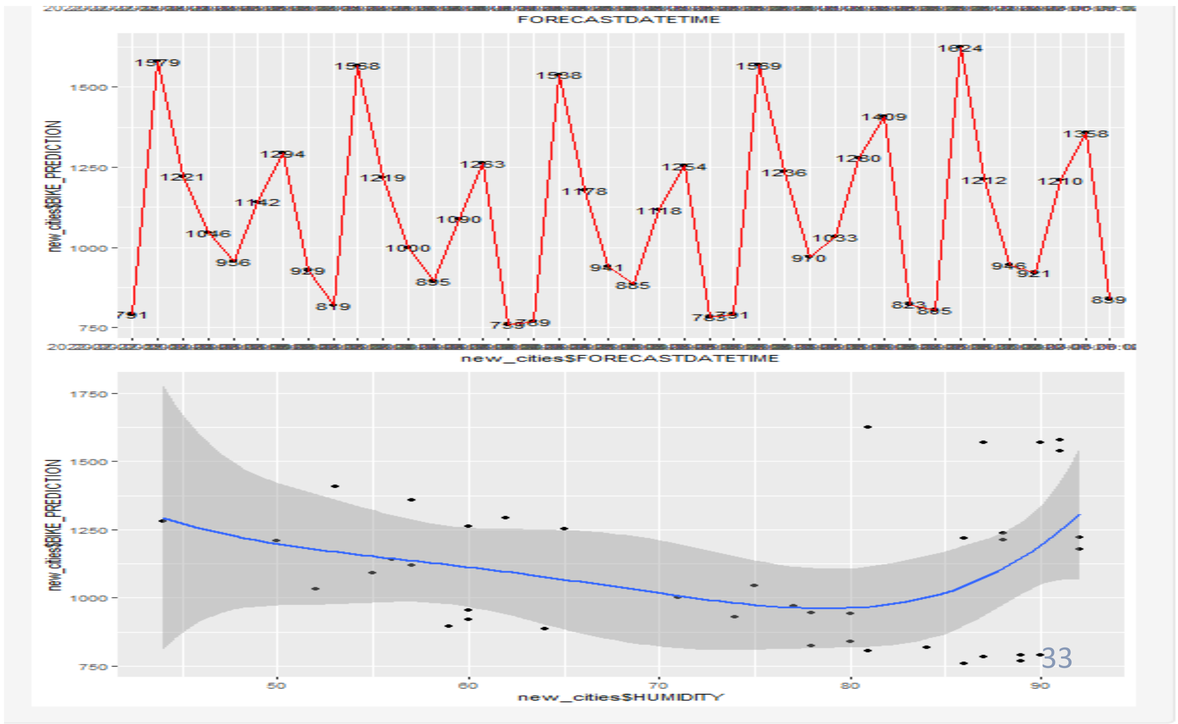
Dashboard Seoul

- Below is the dashboard for Seoul. Here, we can see the label popups with detailed weather, and the graph consists of the same element in All World, but a different results for Seoul data.



Dashboard Suzhou

- Below is the dashboard for Suzhou. Here, we can see the label popups with detailed weather, and the graph consists of the same element in All World, but a different results for Suzhou data.



CONCLUSION

- Temperature in a day could fluctuate and affect the Bike prediction
- Different day affected by the temperature and weather further influence the Bike Prediction
- Humidity also affect the Bike Prediction

APPENDIX

- Data Collection

Next, you need to convert this HTML table into a data frame using the `html_table()` function. You may choc

```
[13]: # Convert the bike-sharing system table into a dataframe
table_nodes <- html_nodes(root_node, "table")
table_nodes
bike_share <- html_table(table_nodes, fill = TRUE)
data_bike_share = bike_share[[2]]
```

```
{xml_nodeset (5)}
[1] <table class="box-Copy_edit plainlinks metadata ambox ambox-style ambox-C ...
[2] <table class="wikitable sortable" style="text-align:left"><tbody>\n<tr>\n ...
[3] <table class="nowraplinks mw-collapsible autocollapse navbox-inner" style ...
[4] <table class="nowraplinks navbox-subgroup" style="border-spacing:0"><tbod ...
[5] <table class="nowraplinks navbox-subgroup" style="border-spacing:0"><tbod ...
```

Summarize the bike sharing system data frame

```
[14]: # Summarize the dataframe
summary(data_bike_share)
```

```
   Country              City                 Name                System
 Length:520          Length:520          Length:520          Length:520
 Class :character    Class :character    Class :character    Class :character
 Mode  :character    Mode  :character    Mode  :character    Mode  :character
   Operator            Launched          Discontinued          Stations
 Length:520          Length:520          Length:520          Length:520
 Class :character    Class :character    Class :character    Class :character
 Mode  :character    Mode  :character    Mode  :character    Mode  :character
   Bicycles          Daily ridership
 Length:520          Length:520
 Class :character    Class :character
 Mode  :character    Mode  :character
```

Export the data frame as a csv file called `raw_bike_sharing_systems.csv`

```
[15]: # Export the dataframe into a csv file
write.csv(data_bike_share, "raw_bike_sharing_systems.csv")
```

For more details about webscraping with `rvest`, please refer to the previous webscraping notebook here:

```
# Get forecast data for a given city list
get_weather_forecaset_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/forecast'
        # Create query parameters
        forecast_query <- list(q = city_name, appid = "8e9f83353198340479e43d48cef10a2f", units="metric")
        # Make HTTP GET call for the given city
        response_forecast <- GET(forecast_url, query=forecast_query)

        # Note that the 5-day forecast JSON result is a list of lists. You can print the reponse to check the results
        #results <- json_list$list
        json_result2 <- content(response_forecast, as="parsed")
        result <- json_result$list
        # Loop the json result
        for(result in results) {
            city <- c(city, json_result2$city)
            weather <- c(weather, json_result2$weather[[1]]$main)
            visibility <- c(visibility, json_result2$visibility)
            temp <- temp, json_result$main$temp)
            temp_min <- c(temp_min, json_result2$main$temp_min)
            temp_max <- c(temp_max, json_result2$main$temp_max)
            pressure <- c(pressure, json_result2$main$pressure)
            humidity <- c(humidity, json_result2$main$humidity)
            wind_speed <- c(wind_speed, json_result2$wind$speed)
            wind_deg <- c(wind_deg, json_result$wind$deg)
            forecast_datetime <- c(forecast_datetime, jsonresult2$dt_text)
            library(zoo)
            months_forecast <- as.numeric(format(as.Date(forecast_datetime), "%m"))
            indx <- setNames(rep(c("Winter","Spring", "Summer", "Fall"), each=3), c(12, 1:11))
            season <- unname(indx[as.character(months_forecast)])
        }

        # Add the R lists into a 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,
                                 forecast_datetime = forecast_datetime, season=season)
    }

    # Return a data frame
    return(df)
}
```

APPENDIX

- Data Wrangling

TASK: Extract the numeric value using regular expressions

TODO: Write a custom function using `stringr::str_extract` to extract the first digital substring match and convert it into numeric type For e

```
[70]: # Extract the first number
extract_num <- function(columns){
    # Define a digital pattern
    digitals_pattern <- "[0-9]+"
    as.numeric(str_extract(columns, digitals_pattern))
    # Find the first match using str_extract
    # Convert the result to numeric using the as.numeric() function
}
```

TODO: Use the `dplyr::mutate()` function to apply `extract_num` on the `BICYCLES` column

```
[71]: # Use the mutate() function on the BICYCLES column
result <- sub_bike_sharing_df %>% mutate(BICYCLES=extract_num(BICYCLES))
```

TODO: Use the summary function to check the descriptive statistics of the numeric `BICYCLES` column

```
[79]: summary(result$BICYCLES)
final_result <- sub_bike_sharing_df %>% mutate(BICYCLES=extract_num(BICYCLES)) %>% mutate(CITY=remove_ref(CITY), SYSTEM=remove_ref2(SYSTEM))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      5     100     343    2012    1400   78000      76
```

TODO: Write the cleaned bike-sharing systems dataset into a csv file called `bike_sharing_systems.csv`

```
[80]: # Write dataset to `bike_sharing_systems.csv`
write.csv(final_result, "bike_sharing_systems.csv", row.names=FALSE)
```

Standardize the column names again for the new datasets

Since you have added many new indicator variables, you need to standardize their column names again by using the following code:

```
[39]: # Dataset list
dataset_list <- c('seoul_bike_sharing.csv', 'seoul_bike_sharing_converted.csv', 'seoul_bike_sharing_converted_normalized.csv')

for (dataset_name in dataset_list){
    # Read dataset
    dataset <- read_csv(dataset_name)
    # Standardized its columns:
    # Convert all columns names to uppercase
    names(dataset) <- toupper(names(dataset))
    # Replace any white space separators by underscore, using str_replace_all function
    names(dataset) <- str_replace_all(names(dataset), " ", "_")
    # Save the dataset back
    write.csv(dataset, dataset_name, row.names=FALSE)
}
```

```
Warning message:
"Missing column names filled in: 'X1' [1]"Parsed with column specification:
cols(
  X1 = col_double(),
  DATE = col_character(),
  RENTED_BIKE_COUNT = col_double(),
  HOUR = col_double(),
  TEMPERATURE = col_double(),
  HUMIDITY = col_double(),
  WIND_SPEED = col_double(),
  VISIBILITY = col_double(),
  DEW_POINT_TEMPERATURE = col_double(),
  SOLAR_RADIATION = col_double(),
  RAINFALL = col_double(),
  SNOWFALL = col_double(),
  SEASONS = col_character(),
  HOLIDAY = col_character(),
  FUNCTIONING_DAY = col_character()
)
Parsed with column specification:
cols(
  .default = col_double(),
  DATE = col_character()
)
See spec(...) for full column specifications.
Parsed with column specification:
cols(
  .default = col_double(),
  DATE = col_character(),
  RENTED_BIKE_COUNT = col_logical(),
  TEMPERATURE = col_logical()
)
See spec(...) for full column specifications.
```

APPENDIX

- EDA with SQL

Solution 1

```
[2]: query <- paste("SELECT COUNT(*) AS COUNT FROM SEOUL_BIKE_SHARING")
seoul_bike_count <- sqlQuery(conn, query)
seoul_bike_count
```

A data.frame: 1 × 1

	COUNT
	<int>
1	8465

Task 2 - Operational Hours

Determine how many hours had non-zero rented bike count.

Solution 2

```
[3]: # provide your solution here
query2 <- paste("SELECT SUM(HOUR) FROM SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT<>0")
total_hrs <- sqlQuery(conn, query2)
total_hrs
```

A data.frame: 1 × 1

	1
	<int>
1	97407

Task 3 - Weather Outlook

Query the the weather forecast for Seoul over the next 3 hours.

Recall that the records in the CITIES_WEATHER_FORECAST dataset are 3 hours apart, so we just need the first record from the query.

Solution 3

```
[4]: # provide your solution here
query3 <- paste("SELECT * FROM SEOUL_BIKE_SHARING LIMIT 1")
weather_Q <- sqlQuery(conn,query3)
weather_Q
```

A data.frame: 1 × 14

	DATE	RENTED BIKE COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND SPEED	VISIBILITY	DEW POINT TEMPERATURE	SOLAR RADIATION	RAINFALL	SNOWFALL	SEASONS	HOLIDAY	FUNCTIONING DAY
	<fct>	<int>	<int>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<fct>	<fct>
1	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	No Holiday	Yes

Task 4 - Seasons

Find which seasons are included in the seoul bike sharing dataset.

Solution 4

```
[5]: # provide your solution here
query4 <- paste("SELECT DISTINCT SEASONS FROM SEOUL_BIKE_SHARING")
season_q <- sqlQuery(conn,query4)
season_q
```

A data.frame: 4 × 1

	SEASONS
	<fct>
1	Autumn
2	Spring
3	Summer
4	Winter

Task 5 - Date Range

Find the first and last dates in the Seoul Bike Sharing dataset.

Solution 5

```
[6]: # provide your solution here
query5 <- paste("SELECT MIN(DATE), MAX(DATE) FROM SEOUL_BIKE_SHARING")
date_q <- sqlQuery(conn,query5)
date_q
```

A data.frame: 1 × 2

	1	2
	<fct>	<fct>
1	01/01/2018	31/12/2017

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

```
[7]: # provide your solution here
query6 <- paste("SELECT DATE, HOUR FROM SEOUL_BIKE_SHARING WHERE RENTED_BIKE_COUNT = (SELECT MAX(RENTED_BIKE_COUNT) FROM SEOUL_BIKE_SHARING)")
most_q <- sqlQuery(conn,query6)
most_q
```

A data.frame: 1 × 2

	DATE	HOUR
	<fct>	<int>
1	19/06/2018	18

APPENDIX

- EDA with SQL

Solution 7

```
# provide your solution here
query7 <- paste("SELECT HOUR, SEASONS, AVG(TEMPERATURE) AS HOURLY_TEMP, AVG(RENTED_BIKE_COUNT) AS HOURLY_RENTED FROM SEOUL_BIKE_SHARING
GROUP BY HOUR, SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC LIMIT 10")
hourly_q <- sqlQuery(conn,query7)
hourly_q
```

A data.frame: 10 × 4

	HOUR	SEASONS	HOURLY TEMP	HOURLY RENTED
	<int>	<fct>	<dbl>	<int>
1	18	Summer	29.38696	2135
2	18	Autumn	16.03086	1983
3	19	Summer	28.27283	1889
4	20	Summer	27.06630	1801
5	21	Summer	26.27826	1754
6	18	Spring	15.97222	1689
7	22	Summer	25.69891	1567
8	17	Autumn	17.27778	1562
9	17	Summer	30.07500	1526
10	19	Autumn	15.06049	1515

Task 8 - Rental Seasonality

Find the average hourly bike count during each season.

Also include the minimum, maximum, and standard deviation of the hourly bike count for each season.

Solution 8

```
# provide your solution here
query8 <- paste("SELECT HOUR, AVG(RENTED_BIKE_COUNT) AS HOURLY_RENTED, MIN(RENTED_BIKE_COUNT) AS MIN, MAX(RENTED_BIKE_COUNT) AS MAX, STDDEV(RENTED_BIKE_COUNT) AS STD, SEASONS FROM SEOUL_BIKE_SHARING
GROUP BY HOUR, SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC LIMIT 10")
rental_q <- sqlQuery(conn, query8)
rental_q
```

A data.frame: 10 × 6

	HOUR	HOURLY RENTED	MIN	MAX	STD	SEASONS
	<int>	<int>	<int>	<int>	<dbl>	<fct>
1	18	2135	17	3556	884.0829	Summer
2	18	1983	40	3298	778.4414	Autumn
3	19	1889	18	2984	728.8799	Summer
4	20	1801	10	2579	662.2163	Summer
5	21	1754	17	2505	596.1374	Summer
6	18	1689	22	3251	898.8971	Spring
7	22	1567	16	2309	516.6434	Summer
8	17	1562	23	2432	554.3165	Autumn
9	17	1526	25	2664	608.7917	Summer
10	19	1515	19	2518	571.1497	Autumn

Solution 9

```
query9 <- paste("SELECT AVG(RENTED_BIKE_COUNT) AS AVG_BIKE,
AVG(TEMPERATURE) AS AVG_TEMP,
AVG(HUMIDITY) AS AVG_HUMID,
AVG(WIND_SPEED) AS AVG_WIND,
AVG(VISIBILITY) AS AVG_VISIB,
AVG(DEW_POINT_TEMPERATURE) AS AVG_DEW,
AVG(SOLAR_RADIATION) AS AVG_SOLAR,
AVG(RAINFALL) AS AVG_RAINFALL,
AVG(SNOWFALL) AS AVG_SNOWFALL,
SEASONS FROM SEOUL_BIKE_SHARING GROUP BY SEASONS ORDER BY AVG(RENTED_BIKE_COUNT) DESC")
weather_seaq <- sqlQuery(conn, query9)
weather_seaq

# provide your solution here
```

A data.frame: 4 × 10

	AVG BIKE	AVG TEMP	AVG HUMID	AVG WIND	AVG VISIB	AVG DEW	AVG SOLAR	AVG RAINFALL	AVG SNOWFALL	SEASONS
	<int>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1	1034	26.587274	64	1.609420	1501	18.750136	0.7612545	0.25348732	0.00000000	Summer
2	924	13.821167	59	1.492101	1558	5.150594	0.5227827	0.11765617	0.06350026	Autumn
3	746	13.021389	58	1.857778	1240	4.091389	0.6803009	0.18694444	0.00000000	Spring
4	225	-2.540463	49	1.922685	1445	-12.416667	0.2981806	0.03282407	0.24750000	Winter

Task 10 - Total Bike Count and City Info for Seoul

Use an implicit join across the WORLD_CITIES and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes avaialble in Seoul, plus the following city information about Seoul: CITY, COUNTRY, LAT, LON, POPULAT

Notice that in this case, the CITY column will work for the WORLD_CITIES table, but in general you would have to use the CITY_ASCII column.

Solution 10

```
# provide your solution here
query10 <- paste("SELECT BS.BICYCLES, WC.CITY_ASCII, WC.COUNTRY, WC.LAT, WC.LNG, WC.POPULATION FROM BIKE_SHARING_SYSTEMS AS BS, WORLD_CITIES AS WC WHERE WC.CITY_ASCII = BS.CITY AND WC.CITY = 'Seoul'")
total_bike <- sqlQuery(conn, query10)
total_bike
```

A data.frame: 1 × 6

	BICYCLES	CITY ASCII	COUNTRY	LAT	LNG	POPULATION
	<int>	<fct>	<fct>	<dbl>	<dbl>	<int>
1	20000	Seoul	Korea, South	37.58	127	21794000

Task 11 - Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system

Find all cities with total bike counts between 15000 and 20000. Return the city and country names, plus the coordinates (LAT, LNG), population, and number of bicycles for each city.

Later we will ask you to visualize these similar cities on leaflet, with some weather data.

Solution 11

```
# provide your solution here
query11 <- ("SELECT BS.BICYCLES, WC.CITY_ASCII, WC.COUNTRY, WC.LAT, WC.LNG, WC.POPULATION FROM BIKE_SHARING_SYSTEMS AS BS, WORLD_CITIES AS WC WHERE WC.CITY_ASCII = BS.CITY AND BICYCLES BETWEEN 15000 AND 20000 [illegible] BY BICYCLES DESC")
city_names <- sqlQuery(conn, query11)
city_names
```

APPENDIX

- EDA with Data Visualization

Solution 1

[1]:
```
# provide your solution here
library(tidyverse)
seoul_dataset <- read.csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets/seoul_bike_sharing.csv",
                          colClasses=c(DATE="character"))
```

```
— Attaching packages ————————————————————————————————————— tidyverse 1.3.0 —
✓ ggplot2 3.3.0     ✓ purrr   0.3.4
✓ tibble  3.0.1     ✓ dplyr   0.8.5
✓ tidyr   1.0.2     ✓ stringr 1.4.0
✓ readr   1.3.1     ✓ forcats 0.5.0
— Conflicts ————————————————————————————————————— tidyverse_conflicts() —
✗ dplyr::filter() masks stats::filter()
✗ dplyr::lag()    masks stats::lag()
```

Task 2 - Recast `DATE` as a date

Use the format of the data, namely "%d/%m/%Y".

Solution 2

[2]:
```
# provide your solution here
seoul_dataset$DATE <- as.Date(seoul_dataset$DATE, format = "%d/%m/%Y")
class(seoul_dataset$DATE)
```

'Date'

Task 3 - Cast `HOURS` as a categorical variable

Also, coerce its levels to be an ordered sequence. This will ensure your visualizations correctly utilize `HOURS` as a discrete variable with the expected ordering.

Solution 3

[3]:
```
# provide your solution here
seoul_dataset$HOUR <-factor(seoul_dataset$HOUR, ordered = TRUE)
class(seoul_dataset$HOUR)
```

'ordered'
'factor'

Task 4 - Dataset Summary

Use the base R `sumamry()` function to describe the `seoul_bike_sharing` dataset.

Solution 4

[6]:
```
# provide your solution here
summary(seoul_dataset)
```

```
      DATE            RENTED_BIKE_COUNT       HOUR         TEMPERATURE
 Min.   :2017-12-01   Min.   :   2.0     7      : 353   Min.   :-17.80
 1st Qu.:2018-02-27   1st Qu.: 214.0     8      : 353   1st Qu.:  3.00
 Median :2018-05-28   Median : 542.0     9      : 353   Median : 13.50
 Mean   :2018-05-28   Mean   : 729.2     10     : 353   Mean   : 12.77
 3rd Qu.:2018-08-24   3rd Qu.:1084.0     11     : 353   3rd Qu.: 22.70
 Max.   :2018-11-30   Max.   :3556.0     12     : 353   Max.   : 39.40
                                         (Other):6347
    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.: 935   1st Qu.: -5.100
 Median :57.00   Median :1.500   Median :1690   Median :  4.700
 Mean   :58.15   Mean   :1.726   Mean   :1434   Mean   :  3.945
 3rd Qu.:74.00   3rd Qu.:2.300   3rd Qu.:2000   3rd Qu.: 15.200
 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   Autumn:1937
 1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Spring:2160
 Median :0.0100   Median : 0.0000   Median :0.00000   Summer:2208
 Mean   :0.5679   Mean   : 0.1491   Mean   :0.07769   Winter:2160
 3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000
 Max.   :3.5200   Max.   :35.0000   Max.   :8.80000

       HOLIDAY      FUNCTIONING_DAY
 Holiday   : 408   Yes:8465
 No Holiday:8057
```

APPENDIX

- EDA with Data Visualization

Solution 5:

```
[7]: # provide your solution here
holiday <- seoul_dataset %>% filter(HOLIDAY == 'Holiday') %>% count()
holiday
```

A tibble: 1 × 1

n
<int>
408

Task 6 - Calculate the percentage of records that fall on a holiday.

Solution 6

```
[8]: # provide your solution here
record <- seoul_dataset %>% filter(HOLIDAY == "Holiday") %>% count()
total <- seoul_dataset %>% count()
percentage <- (record/total) * 100
percentage
```

A data.frame: 1 × 1

n
<dbl>
4.819846

Task 7 - Given there is exactly a full year of data, determine how many records we expect to have.

Solution 7

```
[9]: # provide your solution here
one_day <- seoul_dataset %>% filter(DATE == "2017-12-01") %>% count()
predicted <- 365 * one_day
predicted
```

A data.frame: 1 × 1

n

Solution 8

```
[10]: # provide your solution here
function_day <- seoul_dataset %>% filter(FUNCTIONING_DAY == "Yes") %>% count()
function_day
```

A tibble: 1 × 1

n
<int>
8465

Drilling Down

Let's calculate some seasonally aggregated measures to help build some more context.

Task 9 - Load the dplyr package, group the data by `SEASONS`, and use the `summarize()` function to calculate the seasonal total rainfall and snowfall.

Solution 9

```
[11]: # provide your solution here
library(dplyr)
group <- seoul_dataset %>% group_by(SEASONS) %>%
summarize(rainfall = sum(RAINFALL), snowfall = sum(SNOWFALL))
group
```

A tibble: 4 × 3

SEASONS	rainfall	snowfall
<fct>	<dbl>	<dbl>
Autumn	227.9	123.0
Spring	403.8	0.0
Summer	559.7	0.0
Winter	70.9	534.6

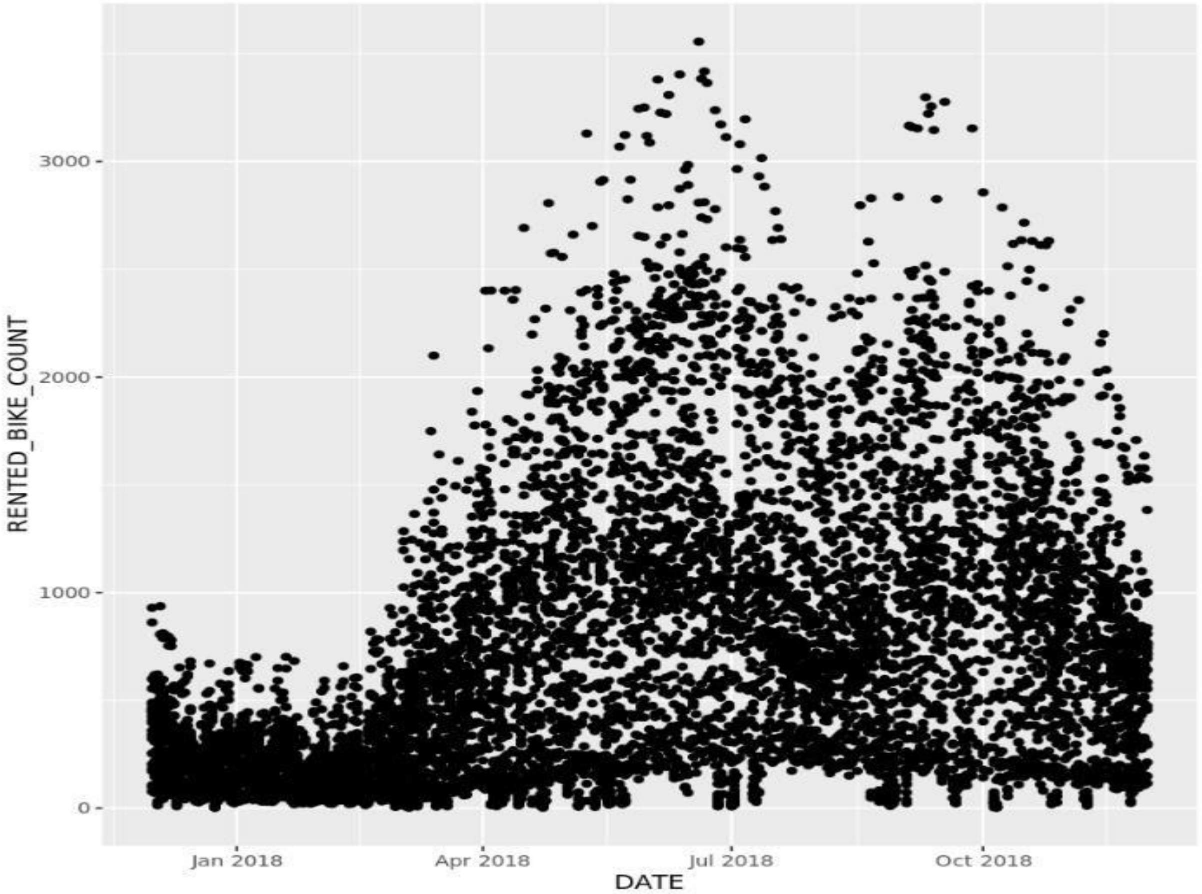
Wow, that seems like a lot of snow.

Now that you have some ideas about what sorts of questions can be answered through descriptive statistics, let's start visualizing the data.

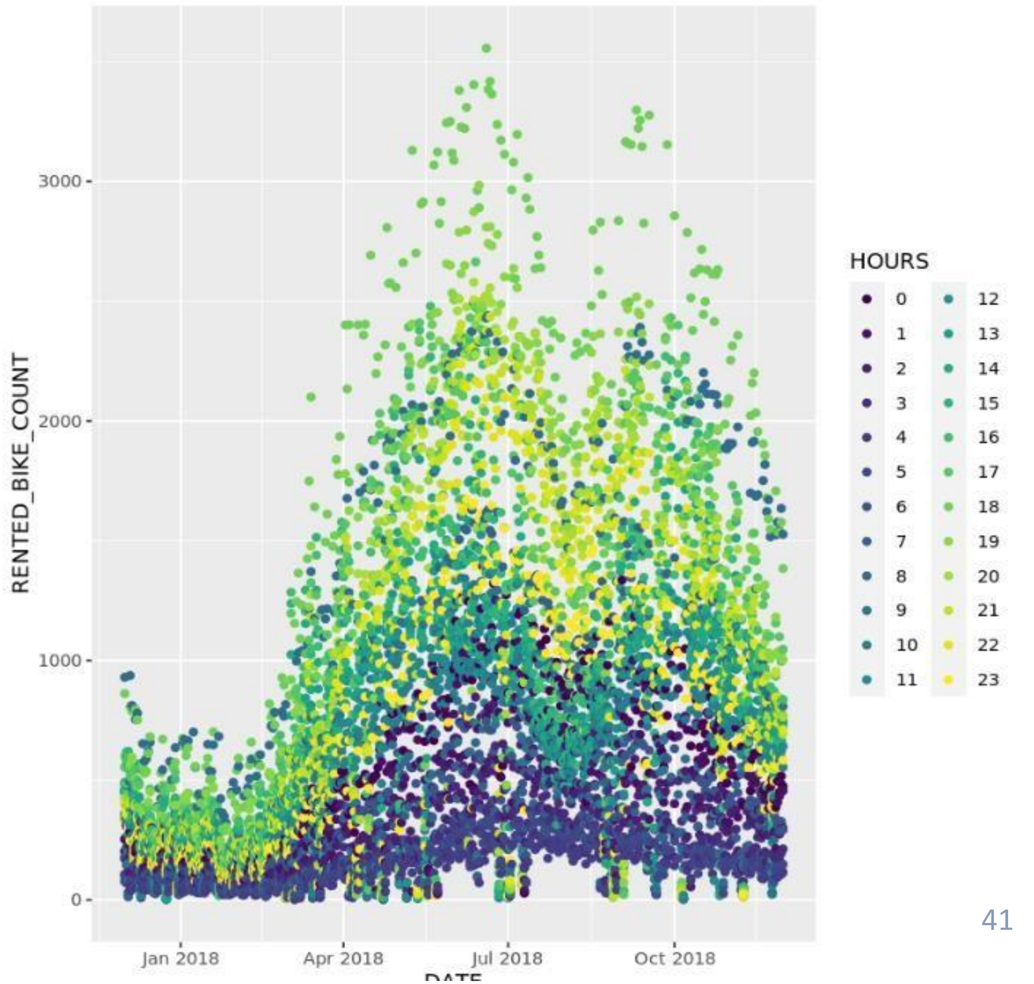
APPENDIX

- EDA with Data Visualization

```
[13]: # provide your solution here
qplot(DATE, RENTED_BIKE_COUNT, data = seoul_dataset, alpha = I(50))
```



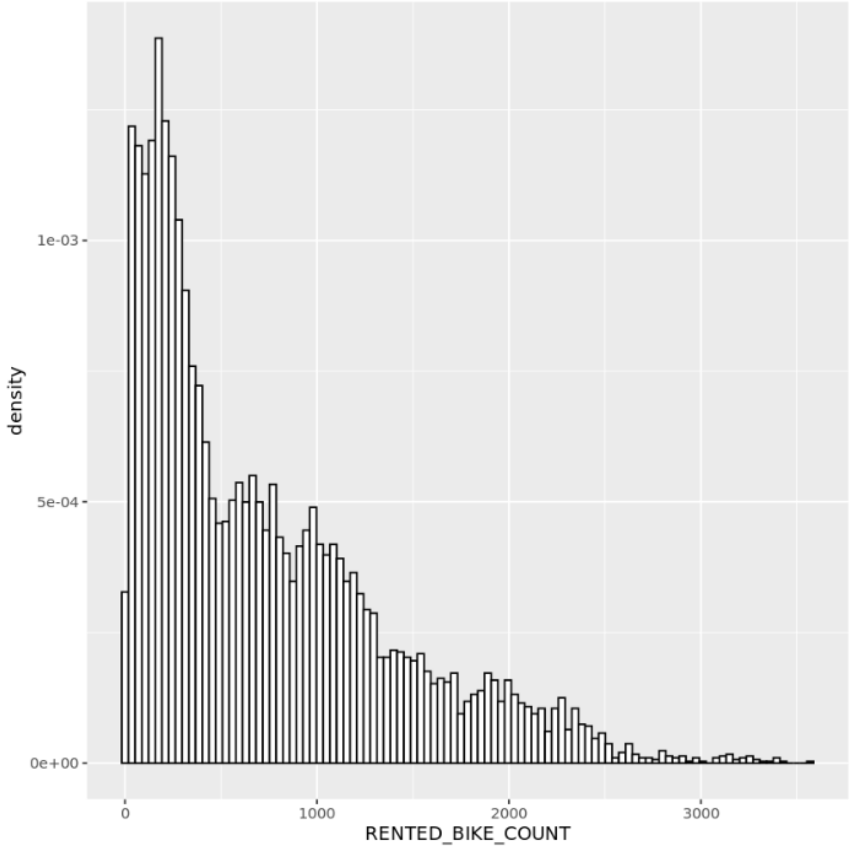

```
[14]: # provide your solution here
ggplot(seoul_dataset, aes(x = DATE, y = RENTED_BIKE_COUNT, color = HOUR), alpha = I(50)) + geom_point() + labs(color = "HOURS")
```



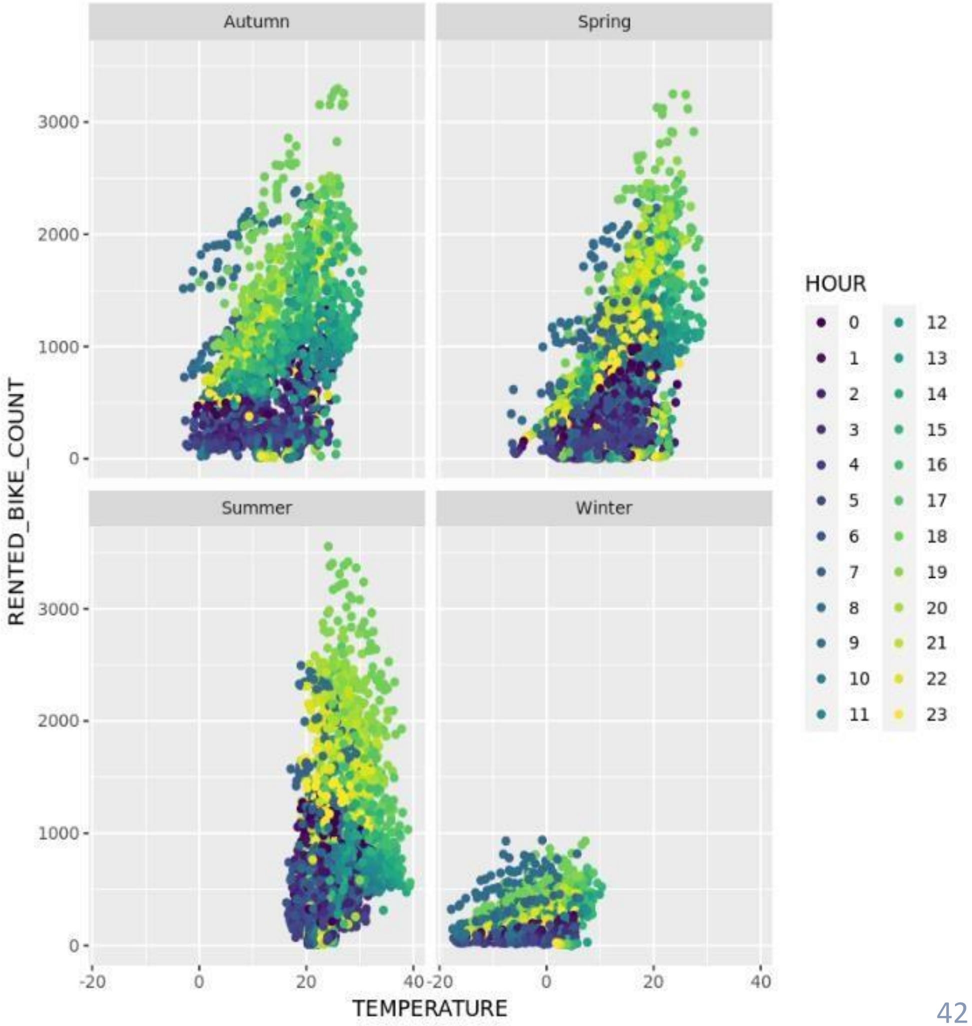
APPENDIX

- EDA with Data Visualization

```
[24]: # provide your solution here
ggplot(seoul_dataset, aes(x = RENTED_BIKE_COUNT)) + geom_histogram(binwidth = 35, color = "black", alpha = I(50), fill = I("white"), aes(y = ..density..))
```



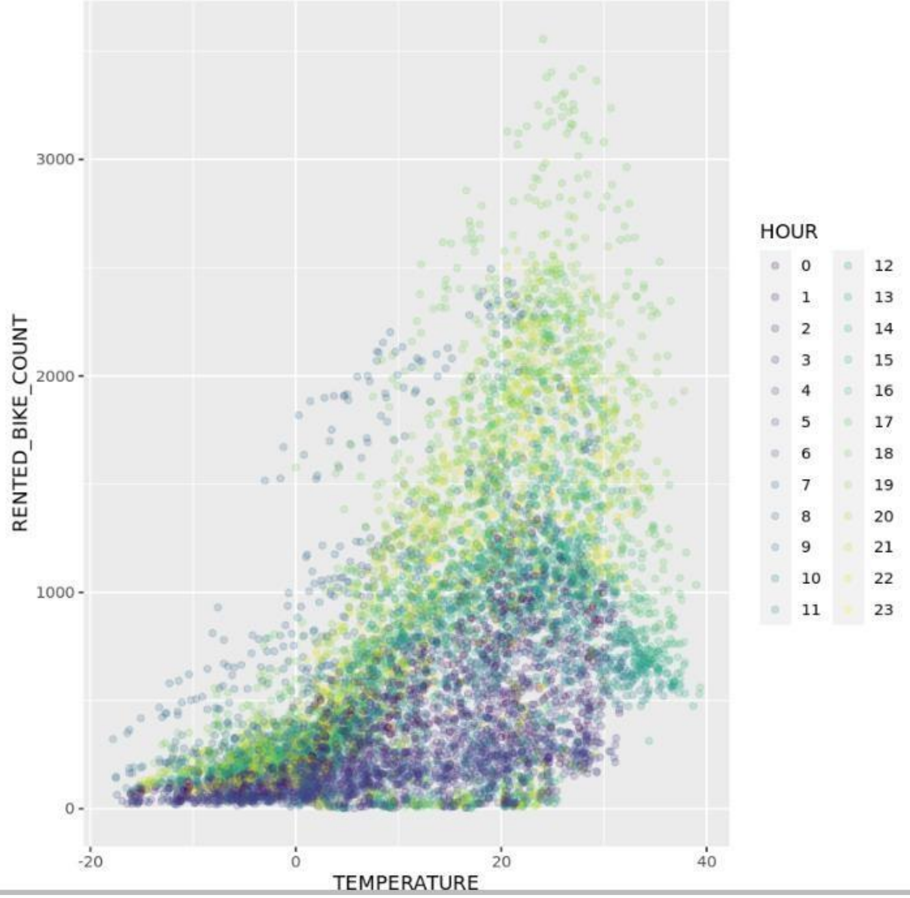

```
30]: # provide your solution here
ggplot(seoul_dataset, aes(x=TEMPERATURE, y=RENTED_BIKE_COUNT), alpha = I(50)) + geom_point(aes(color = HOUR)) + facet_wrap(~SEASONS)
```



APPENDIX

- EDA with Data Visualization

```
[31]: ggplot(seoul_dataset) +
        geom_point(aes(x=TEMPERATURE,y=RENTED_BIKE_COUNT,colour=HOUR),alpha=1/5)
```



Solution 14

```
[32]: # provide your solution here
ggplot(seoul_dataset, aes(x=HOUR, y=RENTED_BIKE_COUNT), alpha = 1/5) + geom_boxplot() + facet_wrap(~SEASONS)
```