Data Science with R - Capstone Project

Jing ZHANG

20 Mar 2023

IBM Developer

SKILLS NETWORK

OUTLINE

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

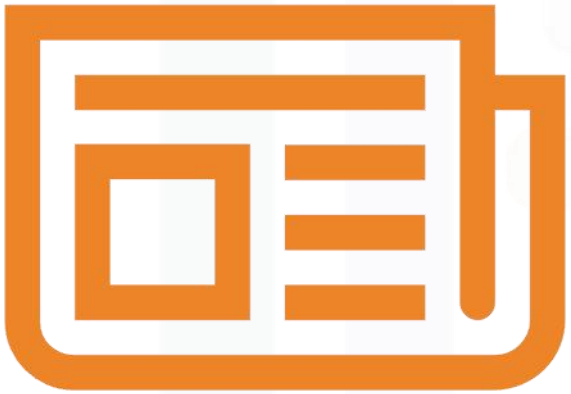
EXECUTIVE SUMMARY

- Collect relevant data from global bike sharing systems on public web pages, weather data via OpenWeather APIs, and aggregated tabular data from cloud storage
- Data wrangling with stringr and regular expressions and dplyr to remove the noise from data and convert the undesired data format
- Exploratory Data Analysis with SQL and using an R notebook to perform exploratory data analysis using tidyverse and the ggplot2 R packages
- Predicting Hourly Rented Bike Count using Basic Linear Regression Models and refineing the Baseline Regression Models
- Building a bike-sharing demand prediction app with R Shiny and Leaflet which is enhancing the Bike-Sharing Demand Prediction App with City Details Plots

INTRODUCTION

- The purpose of this project analysis on bike-sharing demand and weather data collection is to help better understand the process of building a bike-sharing demand prediction app with R Shiny and Leaflet.
- In the initial task of collecting data, used web scrape a Global Bike-Sharing Systems Wiki Page and OpenWeather APIs Calls to aggregate tabular data from cloud storage.
- In the face of the data sorting and classification task of the original dataset with missing data, Regular Expressions and dplyr are used to help with data wrangling.
- Performing Exploratory Data Analysis with SQL, Tidyverse & ggplot2 to help solve exploratory problems with datasets
- In order to solve the demand forecasting problem of shared bicycles, building a baseline Regression Model and improvements for forecasting
- For interactive R Shiny dashboard to be able to visualize and predict, build a bike-sharing demand prediction app with R Shiny and Leaflet to show the max predicted bike-sharing demand in the next 5 days.

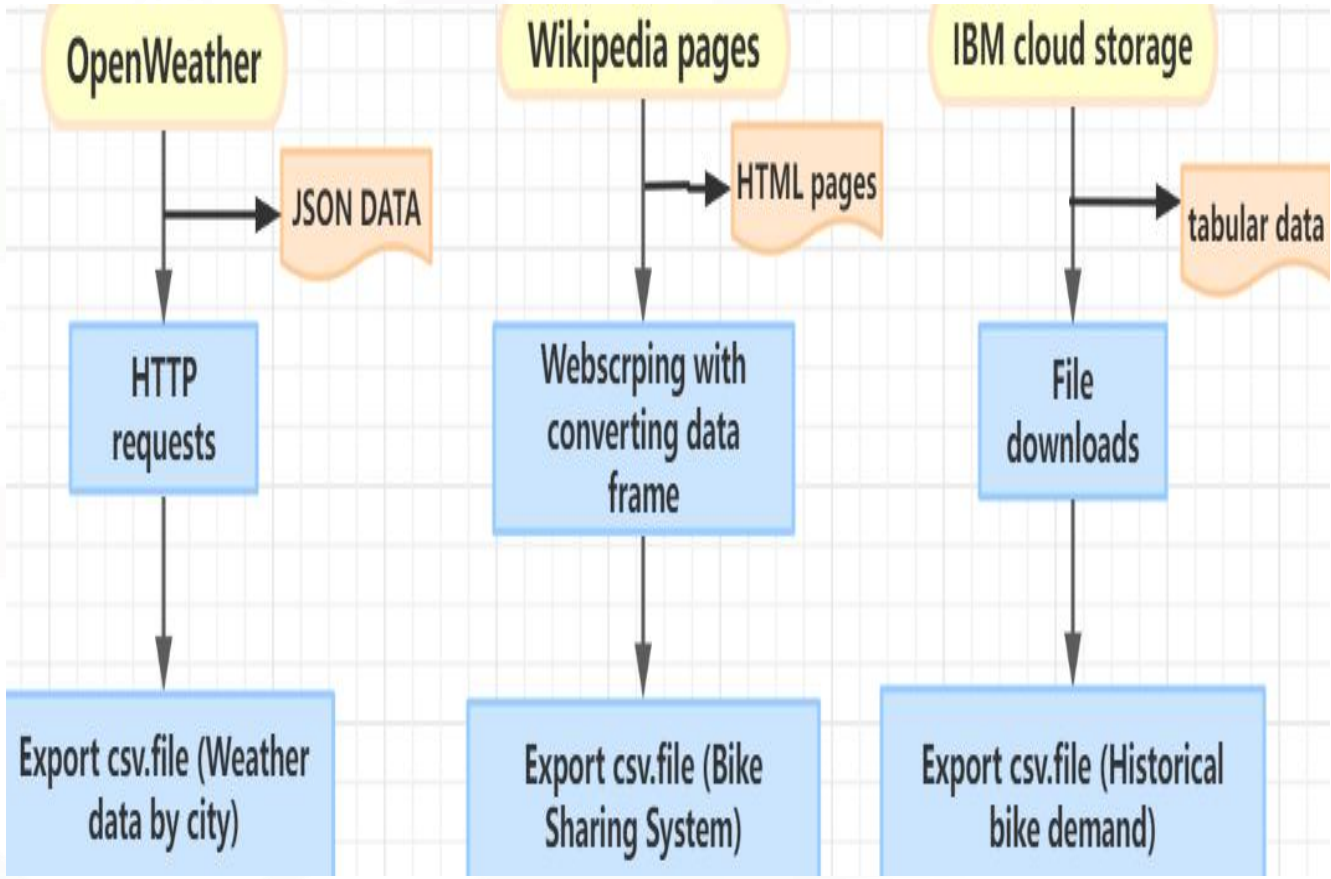
METHODOLOGY



- Data collection
- Data wrangling
- Performing Exploratory Data Analysis with SQL, Tidyverse & ggplot2
- Predictive analysis by using Regression Models and improvements
- Build a bike-sharing demand prediction app with R Shiny and Leaflet

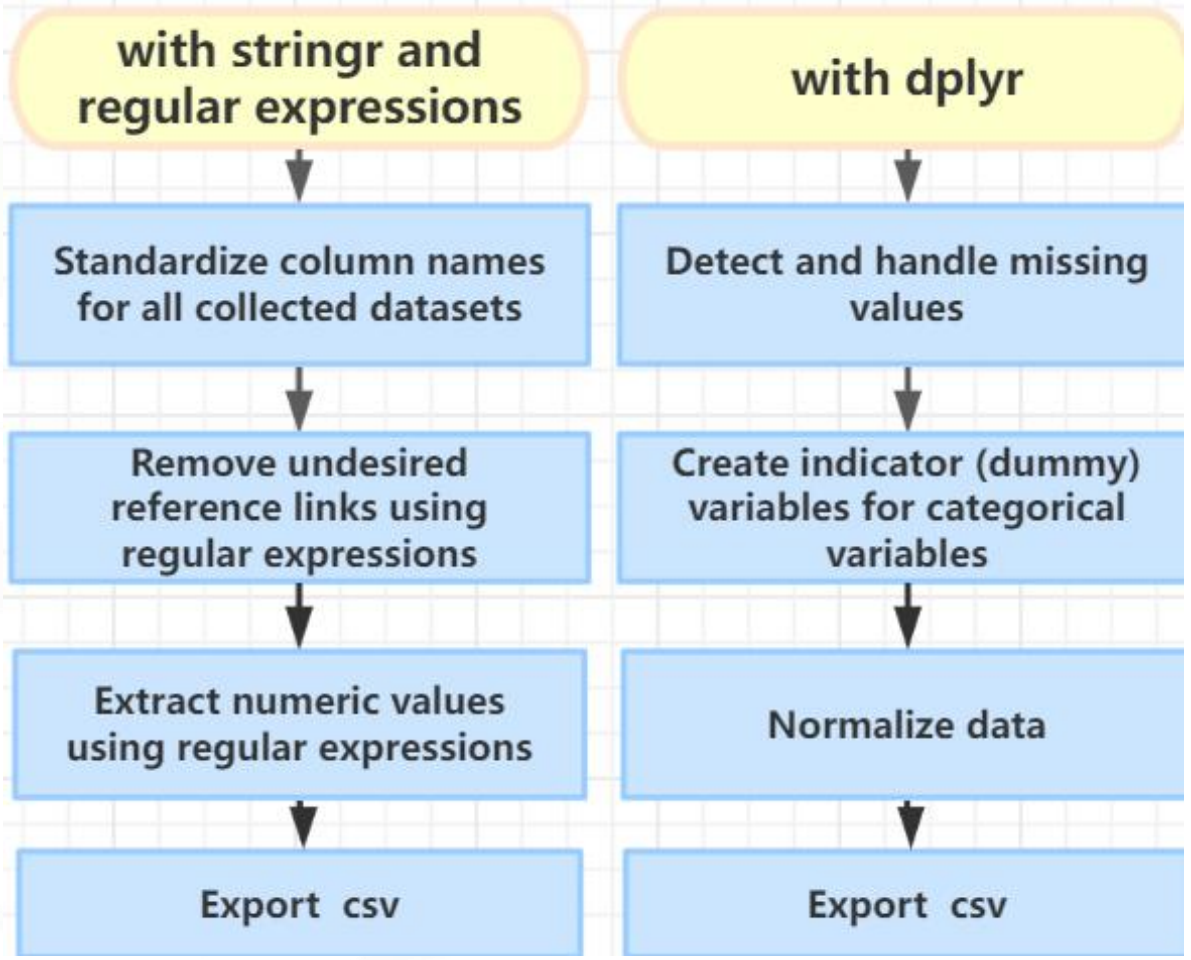
Data Collection

- Three ways of collecting relevant data from various sources:
- The use of HTTP requests to collect the JSON data of OpenWeather
- Using webscraping to collect the HTML pages from Wikipedia web
- Downloading files from IBM cloud storage to obtain tabular data



Data Wrangling

- Seperate ways for dealing with the missing, misformatted and unexpected noises:
- The use of stringr and regular expressions to standardize column and datasets
- Using dplyr to normalize data



Performing Exploratory Data Analysis with SQL

Using SQL queries with the RODBC R package by establishing Db2 connection firstly:

- Determine how many records are in the seoul_bike_sharing dataset
- For how many operational hours had non-zero rented bike count
- Query the the weather forecast for Seoul over the next 3 hours
- Find which seasons are included in the seoul bike sharing dataset
- Find the first and last dates in the Seoul Bike Sharing dataset
- Determine which date and hour had the most bike rentals
- The top ten average bike counts can be used to calculate the average hourly temperature and the average number of bike rentals per hour throughout each season
- Find the average hourly bike count during each season
- Consider the weather over each season
- Determine the total number of bikes avaialble in Seoul

Exploratory Data Analysis with SQL

Using SQL with tidyverse and the ggplot2 R packages:

- Load the dataset; Recast DATE; Cast HOURS as a categorical variable;

For Descriptive Statistics:

- Use dataset summaryto describe the seoul_bike_sharing dataset; calculate how many Holidays there are; Calculate the percentage of records that fall on a holiday; Given there is exactly a full year of data, determine how many records we expect to have; Given the observations of how many records must there be;

For drilling down:

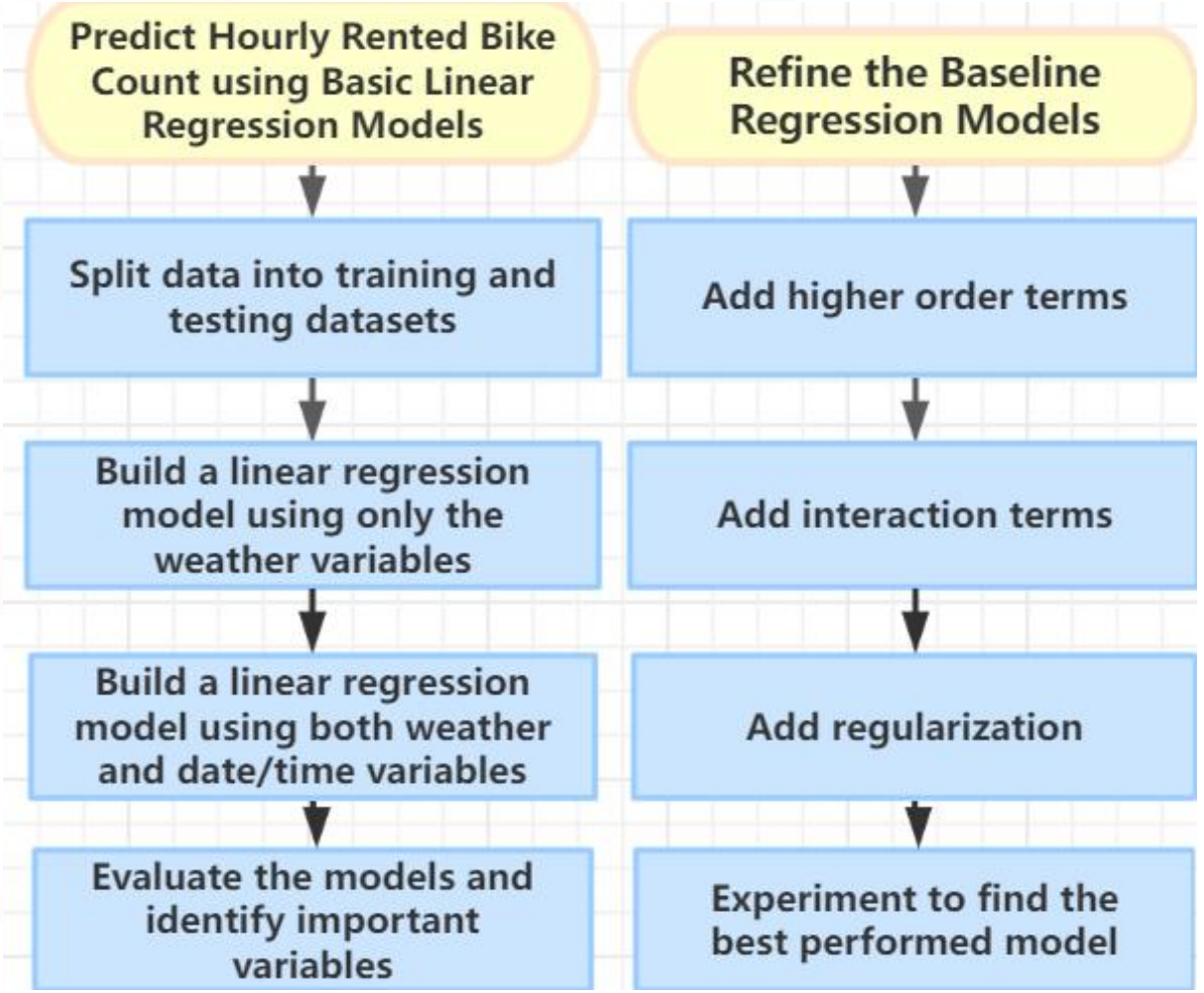
- calculate the seasonal total rainfall and snowfall

For data visluzation:

- Create a scatter plot of RENTED_BIKE_COUNT vs DATE; Create the same plot of the RENTED_BIKE_COUNT time series, but now add HOURS as the colour; Create a histogram overlaid with a kernel density curve; Use a scatter plot to visualize the correlation between RENTED_BIKE_COUNT and TEMPERATURE; Create a display of four boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS; Group the data by DATE and calculate the daily total rainfall and snowfall; Determine how many days had snowfall

Predictive Analysis

- The ways of predicting bike-sharing demand using regression models and improvements:
- Build basic linear regression models to predict the hourly rented bike count using related weather and date information
- Use methods like adding polynomial and interaction terms to refine the baseline regression models



Building a Dashboard with R Shiny

- **Build an interactive R Shiny dashboard to be able to visualize weather forecast data and predicted hourly bike-sharing demand for the following cities, New York, USA, Paris, France, Suzhou, China, and London, UK:**
- Leaflet-based interactive map that shows the max predicted bike-sharing demand in the next 5 days

- **Built a R Shiny app with leaflet to show the max bike-sharing demand predictions for each city:**
- Use ggplot to render some more detailed plots such as bike-sharing prediction trend, temperature trend, humidity and bike-sharing demand prediction correlation, when users zoom-in to a city.

Results: Data collection with webscrping and API

TASK: Extract bike sharing systems HTML table from a Wiki page and convert it into a data frame

TODO: Get the root HTML node

```
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
# Get the root HTML node by calling the `read_html()` method with URL
root_node <- read_html(url)
```

Note that this HTML page at least contains three child `<table>` nodes under the root HTML node. So, you will need to use `html_nodes(root_node, "table")` function to get all its child `<table>` nodes:

```
<html>
  <table>(table1)</table>
  <table>(table2)</table>
  <table>(table3)</table>
  ...
</html>
```

table_nodes <- html_nodes(root_node, "table")

You can use a `for` loop to print each table, and then you will see that the actual the bike sharing table is the first element `table_nodes[[1]]`.

Next, you need to convert this HTML table into a data frame using the `html_table()` function. You may choose to include `fill = TRUE` argument to fill any empty table rows/columns.

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

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

Summarize the bike sharing system data frame

```
# Summarize the dataframe
summary(data_bike_share)
```

```
   Country              City               Name              System         
 Length:549         Length:549         Length:549         Length:549        
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
   Operator           Launched         Discontinued         Stations        
 Length:549         Length:549         Length:549         Length:549        
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
   Bicycles         Daily ridership                                         
 Length:549         Length:549                                              
 Class :character   Class :character                                        
 Mode  :character   Mode  :character                                        
```

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

```
# 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 = "d49fa8896fd76b755589076219b85fef", units="metric")
        # Make HTTP GET call for the given city
        response <- 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$list
         json_list <- content(response, as="parsed")
         results <- json_list$list
        # Loop the json result
        for(result in results) {
            city <- c(city, city_name)
      weather <- c(weather, result$weather[[1]]$main)
# Get Visibility
visibility <- c(visibility, json_result$visibility)
# Get current temperature
temp <- c(temp, json_result$main$temp)
# Get min temperature
temp_min <- c(temp_min, json_result$main$temp_min)
# Get max temperature
temp_max <- c(temp_max, json_result$main$temp_max)
# Get pressure
pressure <- c(pressure, json_result$main$pressure)
# Get humidity
humidity <- c(humidity, json_result$main$humidity)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind direction
wind_deg <- c(wind_deg, json_result$wind$deg)
forecast_datetime <- c(forecast_datetime, result$dt_txt)
forecast_datetime <- c(forecast_datetime, result$dt_txt)

        }
        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)
        # Add the R Lists into a data frame
    }

    # Return a data frame
    return(df)

}
```

Complete and call `get_weather_forecaset_by_cities` function with a list of cities, and write the data frame into a csv file called `cities_weather_forecast.csv`

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

Complete and call `get_weather_forecaset_by_cities` function with a list of cities, and write the data frame into a csv file called `cities_weather_forecast.csv`

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

```
# Write cities_weather_df to `cities_weather_forecast.csv`
write.csv(cities_weather_df, "cities_weather_forecast.csv", row.names=FALSE)
```

For more details about HTTP requests with `httr`, please refer to the previous HTTP request notebook here:

HTTP request in R

TASK: Download datasets as csv files from cloud storage

The last task of this lab is straightforward: download some aggregated datasets from cloud storage

```
# 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 = "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.c
# download the file
download.file(url, destfile = "raw_seoul_bike_sharing.csv")
```

Results: Data collection with webscrping and API

TASK: Extract bike sharing systems HTML table from a Wiki page and convert it into a data frame

TODO: Get the root HTML node

```
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
# Get the root HTML node by calling the `read_html()` method with URL
root_node <- read_html(url)
```

Note that this HTML page at least contains three child `<table>` nodes under the root HTML node. So, you will need to use `html_nodes(root_node, "table")` function to get all its child `<table>` nodes:

```
<html>
  <table>(table1)</table>
  <table>(table2)</table>
  <table>(table3)</table>
  ...
</html>
```

table_nodes <- html_nodes(root_node, "table")

You can use a `for` loop to print each table, and then you will see that the actual the bike sharing table is the first element `table_nodes[[1]]`.

Next, you need to convert this HTML table into a data frame using the `html_table()` function. You may choose to include `fill = TRUE` argument to fill any empty table rows/columns.

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

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

Summarize the bike sharing system data frame

```
# Summarize the dataframe
summary(data_bike_share)
```

```
   Country              City               Name              System         
 Length:549         Length:549         Length:549         Length:549        
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
   Operator           Launched         Discontinued         Stations        
 Length:549         Length:549         Length:549         Length:549        
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
   Bicycles         Daily ridership                                         
 Length:549         Length:549                                              
 Class :character   Class :character                                        
 Mode  :character   Mode  :character                                        
```

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

```
# 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 = "d49fa8896fd76b755589076219b85fef", units="metric")
        # Make HTTP GET call for the given city
        response <- 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$list
         json_list <- content(response, as="parsed")
         results <- json_list$list
        # Loop the json result
        for(result in results) {
            city <- c(city, city_name)
       weather <- c(weather, result$weather[[1]]$main)
# Get Visibility
visibility <- c(visibility, json_result$visibility)
# Get current temperature
temp <- c(temp, json_result$main$temp)
# Get min temperature
temp_min <- c(temp_min, json_result$main$temp_min)
# Get max temperature
temp_max <- c(temp_max, json_result$main$temp_max)
# Get pressure
pressure <- c(pressure, json_result$main$pressure)
# Get humidity
humidity <- c(humidity, json_result$main$humidity)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind direction
wind_deg <- c(wind_deg, json_result$wind$deg)
forecast_datetime <- c(forecast_datetime, result$dt_txt)
forecast_datetime <- c(forecast_datetime, result$dt_txt)
        }
        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)
        # Add the R Lists into a data frame
    }

    # Return a data frame
    return(df)

}
```

Complete and call `get_weather_forecaset_by_cities` function with a list of cities, and write the data frame into a csv file called `cities_weather_forecast.csv`

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

Complete and call `get_weather_forecaset_by_cities` function with a list of cities, and write the data frame into a csv file called `cities_weather_forecast.csv`

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

```
# Write cities_weather_df to `cities_weather_forecast.csv`
write.csv(cities_weather_df, "cities_weather_forecast.csv", row.names=FALSE)
```

For more details about HTTP requests with `httr`, please refer to the previous HTTP request notebook here:

HTTP request in R

TASK: Download datasets as csv files from cloud storage

The last task of this lab is straightforward: download some aggregated datasets from cloud storage

```
# 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 = "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.c
# download the file
download.file(url, destfile = "raw_seoul_bike_sharing.csv")
```

Results: Data wrangling with regular expression

TASK: Standardize column names for all collected datasets

```
# A tibble: 6 x 10
  COUNTRY CITY  NAME  SYSTEM OPERATOR LAUNCHED DISCONTINUED STATIONS BICYCLES
  <chr>   <chr> <chr> <chr>  <chr>    <chr>    <chr>        <chr>    <chr>
1 Albania Tira… Ecov… <NA>   <NA>     March 2… <NA>         8        200
2 Argent… Mend… Metr… <NA>   <NA>     2014     <NA>         2        40
3 Argent… San … Bici… Biciu… <NA>     27 Nove… <NA>         8        80
4 Argent… Buen… Ecob… Sertt… Bike In… 2010     <NA>         400      4000
5 Argent… Rosa… Mi B… <NA>   <NA>     2 Decem… <NA>         47       480
6 Austra… Melb… Melb… PBSC … Motivate June 20… 30 November… 53       676
# … with 1 more variable: DAILY RIDERSHIP <chr>
# A tibble: 6 x 14
  DATE  RENTED_BIKE_COU…  HOUR TEMPERATURE HUMIDITY WIND_SPEED VISIBILITY
  <chr>             <dbl> <dbl>       <dbl>    <dbl>      <dbl>      <dbl>
1 01/1…               254     0        -5.2       37        2.2       2000
2 01/1…               204     1        -5.5       38        0.8       2000
3 01/1…               173     2        -6         39        1         2000
4 01/1…               107     3        -6.2       40        0.9       2000
5 01/1…                78     4        -6         36        2.3       2000
6 01/1…               100     5        -6.4       37        1.5       2000
# … with 7 more variables: DEW_POINT_TEMPERATURE <dbl>, SOLAR_RADIATION <dbl>,
#   RAINFALL <dbl>, SNOWFALL <dbl>, SEASONS <chr>, HOLIDAY <chr>,
#   FUNCTIONING DAY <chr>
# A tibble: 6 x 12
  CITY  WEATHER VISIBILITY  TEMP TEMP_MIN TEMP_MAX PRESSURE HUMIDITY WIND_SPEED
  <chr> <chr>        <dbl> <dbl>    <dbl>    <dbl>    <dbl>    <dbl>      <dbl>
1 Seoul Clear        10000  12.3     10.9     12.3     1015       50       2.18
2 Seoul Clear        10000  11.5      9.81    11.5     1016       48       1.25
3 Seoul Clouds       10000   9.99     8.82     9.99    1015       46       0.94
4 Seoul Clouds       10000   7.87     7.87     7.87    1014       46       0.83
5 Seoul Clouds       10000  10.1     10.1     10.1     1014       37       1.96
6 Seoul Rain         10000   9.74     9.74     9.74    1014       48       3.24
# … with 3 more variables: WIND_DEG <dbl>, SEASON <chr>,
#   FORECAST DATETIME <dttm>
# A tibble: 6 x 11
  CITY  CITY_ASCII   LAT   LNG COUNTRY ISO2  ISO3  ADMIN_NAME CAPITAL POPULATION
  <chr> <chr>      <dbl> <dbl> <chr>   <chr> <chr> <chr>      <chr>        <dbl>
1 Tokyo Tokyo       35.7  140.  Japan   JP    JPN   Tōkyō      primary   37977000
2 Jaka… Jakarta     -6.21 107.  Indone… ID    IDN   Jakarta    primary   34540000
3 Delhi Delhi       28.7   77.2 India   IN    IND   Delhi      admin     29617000
4 Mumb… Mumbai      19.0   72.8 India   IN    IND   Mahārāsht… admin     23355000
5 Mani… Manila      14.6  121.  Philip… PH    PHL   Manila     primary   23088000
6 Shan… Shanghai    31.2  121.  China   CN    CHN   Shanghai   admin     22120000
# … with 1 more variable: ID <dbl>
```

Process the web-scraped bike sharing system dataset

A tibble: 4 × 2

variable	class
COUNTRY	character
CITY	character
SYSTEM	character
BICYCLES	character

BICYCLES
A spec_tbl_df: 10 × 1
4115[22]
310[59]
500[72]
[75]
180[76]
600[77]
[78]
initially 800 (later 2500)
100 (220)
370[114]

CITY
A spec_tbl_df: 10 × 1
Melbourne[12]
Brisbane[14][15]
Lower Austria[18]
Namur[19]
Brussels[21]
Salvador[23]
Belo Horizonte[24]
João Pessoa[25]
(Pedro de) Toledo[26]
Rio de Janeiro[27]

SYSTEM
A spec_tbl_df: 7 × 1
EasyBike[58]
4 Gen.[61]
3 Gen. SmooveKey[113]
3 Gen. Smoove[141][142][143][139]
3 Gen. Smoove[179]
3 Gen. Smoove[181]
3 Gen. Smoove[183]

TASK: Remove undesired reference links using regular expressions

```
result
```

COUNTRY	CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>	<chr>

A spec_tbl_df: 480 × 4

```
result %>%
    select(CITY, SYSTEM, BICYCLES) %>%
    filter(find_reference_pattern(CITY) | find_reference_pattern(SYSTEM) | find_reference_pattern(BICYCLES))
```

CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>

spec_tbl_df: 25 × 3

TASK: Extract the numeric value using regular expressions

```
summary(result$BICYCLES)
```

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

SKILLS NETWORK

Results: Data wrangling with with dplyr

TASK: Detect and handle missing values

```
dim(bike_sharing_df)
```

1. 8465
2. 14

```
# Print the summary of the dataset again to make sure no missing values in all columns
bike_sharing_df %>%
  summarize(count = sum(is.na(TEMPERATURE)))
  head(bike_sharing_df)
  dim(bike_sharing_df)
```

count
<int>
0

A tibble: 1 × 1

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILIT
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl

A tibble: 6 × 14

TASK: Create indicator (dummy) variables for categorical variables

```
# Print the dataset summary again to make sure the indicator columns are created properly
head(seoul_bike_sharing_converted)
```

DATE	RENTED_BIKE_COUNT	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	...	21	
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>	<db

A tibble: 6 × 41

TASK: Normalize data

```
head(seoul_bike_sharing_converted_normalized)
```

DATE	RENTED_BIKE_COUNT	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	...	21	
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>	<db
01/12/2017	0.07090602	0.2202797	0.3775510	0.2972973	1	0.2249135	0	0	0	...	0	
01/12/2017	0.05683737	0.2150350	0.3877551	0.1081081	1	0.2249135	0	0	0	...	0	
01/12/2017	0.04811480	0.2062937	0.3979592	0.1351351	1	0.2231834	0	0	0	...	0	
01/12/2017	0.02954418	0.2027972	0.4081633	0.1216216	1	0.2249135	0	0	0	...	0	
01/12/2017	0.02138436	0.2062937	0.3673469	0.3108108	1	0.2076125	0	0	0	...	0	
01/12/2017	0.02757456	0.1993007	0.3775510	0.2027027	1	0.2058824	0	0	0	...	0	

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:

```
# 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)
}
```

Results: Exploratory Data Analysis with SQL

Task 1 - Record Count

Determine how many records are in the seoul_bike_sharing datase

Solution 1

```
# provide your solution here
query = "SELECT count(*) FROM seoul_bike_sharing dataset"
    sqlQuery(conn, query)
```

	1
	<int>
A data.frame: 1 × 1	
1	8465

Task 2 - Operational Hours

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

Solution 2

```
# provide your solution here
query = "SELECT COUNT(HOUR) FROM seoul_bike_sharing dataset
         WHERE HOUR >0"
    sqlQuery(conn, query)
```

	1
	<int>
A data.frame: 1 × 1	
1	8113

Task 4 - Seasons

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

Solution 4

```
# provide your solution here
query = "SELECT DISTINCT(SEASONS) FROM seoul_bike_sharing dataset  "
    sqlQuery(conn, query)
```

	SEASONS
	<fct>
A data.frame: 4 × 1	
1	Autumn
2	Spring
3	Summer

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

```
# provide your solution here
query = "SELECT * FROM CITIES_WEATHER_FORECAST LIMIT 1  "
    sqlQuery(conn, query)
```

	CITY	WEATHER	VISIBILITY	TEMP	TEMP_MIN	TEMP_MAX	PRESSURE	HUMIDITY	WIND_SPEED	WIND_DEG	SEASON	FORECAST_DATETIM
	<fct>	<fct>	<int>	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<int>	<fct>	<dttm
A data.frame: 1 × 12												
1	Seoul	Clear	10000	12.32	10.91	12.32	1015	50	2.18	248	Spring	2021-04-16 12:00:

Task 5 - Date Range

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

Solution 5

```
# provide your solution here
query = "SELECT MAX(DATE) as First_date, MIN(DATE) as Last_date FROM seoul_bike_sharing dataset
        "
    sqlQuery(conn, query)
```

	FIRST_DATE	LAST_DATE
	<fct>	<fct>
A data.frame: 1 × 2		
1	31/12/2017	01/01/2018

Results: Exploratory Data Analysis with SQL

- By selecting date, hour, the count of tenting bikes from Seoul to confirm the date and the hour of the most bike rentals

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

```
# provide your solution here
query = "SELECT DATE, HOUR, RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         ORDER BY RENTED_BIKE_COUNT DESC LIMIT 1
        "

    sqlQuery(conn, query)
```

	DATE	HOUR	RENTED_BIKE_COUNT
	<fct>	<int>	<int>
A data.frame: 1 × 3			
1	19/06/2018	18	3556

Results: Exploratory Data Analysis with SQL

Task 7 - Hourly popularity and temperature by season

Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.

Solution 7

```
# provide your solution here
query = "SELECT  HOUR AS HOUR_Autumn, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         WHERE SEASONS=  'Autumn'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10

        "
    sqlQuery(conn, query)


query = "SELECT  HOUR AS HOUR__Spring, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         WHERE SEASONS=  'Spring'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)

query = "SELECT  HOUR AS HOUR__Summer, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         WHERE SEASONS=  'Summer'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)

query = "SELECT  HOUR AS HOUR__Winter, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         WHERE SEASONS=  'Winter'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)
```

HOUR_AUTUMN	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
<int>	<dbl>	<int>

A data.frame: 10 × 3

HOUR__SPRING	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
<int>	<dbl>	<int>

A data.frame: 10 × 3

HOUR__SUMMER	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
<int>	<dbl>	<int>

A data.frame: 10 × 3

HOUR__WINTER	AVG_TEMPERATURE	AVG_RENTED_BIKE_
<int>	<dbl>	

A data.frame: 10 × 3

```
query = "SELECT  HOUR AS HOUR__Summer, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         WHERE SEASONS=  'Summer'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)
```

	HOUR_SUMMER	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	29.38696	2135
2	19	28.27283	1889
3	20	27.06630	1801
4	21	26.27826	1754
5	22	25.69891	1567
6	17	30.07500	1526
7	8	24.53587	1418
8	16	30.48804	1174
9	23	25.23043	1153
10	15	30.49022	1009

- By selecting the average hourly temperature and the average number of bike rentals per hour over each season
- Next ordering the top ten results by average bike count

SKILLS NETWORK

Results: Exploratory Data Analysis with SQL

- By selecting the average hourly bike count during each season.
- Then also selecting the minimum, maximum, and standard deviation of the hourly bike count for each season

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
query = "SELECT   SEASONS,  AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT ,
                  MAX(RENTED_BIKE_COUNT) AS Max_Rented_BIKE,
                  MIN(RENTED_BIKE_COUNT) AS MIN_Rented_BIKE,
                  STDDEV(RENTED_BIKE_COUNT) AS standard_Deviation

                  FROM seoul_bike_sharing dataset
                  GROUP BY SEASONS

        "
    sqlQuery(conn, query)
```

	SEASONS	AVG_RENTED_BIKE_COUNT	MAX_RENTED_BIKE	MIN_RENTED_BIKE	STANDARD_DEVIATION
	<fct>	<int>	<int>	<int>	<dbl>
A data.frame: 4 × 5					
1	Autumn	924	3298	2	617.3885
2	Spring	746	3251	2	618.5247
3	Summer	1034	3556	9	690.0884
4	Winter	225	937	3	150.3374

Results: Exploratory Data Analysis with SQL

- Selecting then ranking the results about the average count of TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, SNOWFALL for each season

Task 9 - Weather Seasonality

Consider the weather over each season. On average, what were the TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season?

Include the average bike count as well , and rank the results by average bike count so you can see if it is correlated with the weather at all.

Solution 9

```
query = "SELECT  SEASONS,
                 AVG(RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT,
                 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 dataset
         GROUP BY SEASONS

         "
    sqlQuery(conn, query)
```

A data.frame: 4 × 10

	SEASONS	AVG_RENTED_BIKE_COUNT	AVG_TEMPERATURE	AVG_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT_TEMPERATURE	AVG_SOLAR_RADIATION
	<fct>	<int>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>
1	Autumn	924	13.821167	59	1.492101	1558	5.150594	0.5227827
2	Spring	746	13.021389	58	1.857778	1240	4.091389	0.6803009
3	Summer	1034	26.587274	64	1.609420	1501	18.750136	0.7612545
4	Winter	225	-2.540463	49	1.922685	1445	-12.416667	0.2981806

Results: Exploratory Data Analysis with SQL

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, POPULATION, in a single view.

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
query = "SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
        WHERE B.CITY=W.CITY_ASCII AND B.CITY='Seoul'
"
    sqlQuery(conn, query)
```

	CITY	COUNTRY	LAT	LNG	POPULATION	BICYCLES
	<fct>	<fct>	<dbl>	<dbl>	<int>	<int>
A data.frame: 1 × 6						
1	Seoul	Korea, South	37.58	127	21794000	20000

- Selecting an implicit join across the WORLD_CITIES and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes avaialble in Seoul, with the city information about Seoul: CITY, COUNTRY, LAT, LON, POPULATION, with the CITY_ASCII column

Results: Exploratory Data Analysis with SQL

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

Solution 11

- Selecting 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.

```
# provide your solution here
query = " WITH comparable_cities AS
          (
          SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES AS BICYCLES
          FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
          WHERE   B.CITY=W.CITY_ASCII AND BICYCLES != 'NA'
         )

        SELECT *
        FROM comparable_cities
        WHERE BICYCLES BETWEEN 15000 AND 20000
        ORDER BY BICYCLES DESC

"
    sqlQuery(conn, query)
```

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

```
query = " WITH comparable_cities AS
          (
          SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES AS BICYCLES
          FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
          WHERE   B.CITY=W.CITY_ASCII AND BICYCLES != 'NA'
         )

        SELECT CITY,  SUM(BICYCLES)  AS BICYCLES
        FROM comparable_cities
        GROUP BY CITY
        HAVING  SUM(BICYCLES) BETWEEN 15000 AND 20000
        ORDER BY BICYCLES DESC

"
    sqlQuery(conn, query)
```

	CITY	BICYCLES
	<fct>	<int>
A data.frame: 7 × 2		
1	Seoul	20000
2	Weifang	20000
3	Xi'an	20000
4	Zhuzhou	20000
5	Shanghai	19165
6	Beijing	16000
7	Ningbo	15000

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 1 - Load the dataset

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>

A tibble: 6 × 14

Task 2 - Recast `DATE` as a date

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

Solution 2

```
# provide your solution here
seoul_bike_sharing$DATE<-as.Date(seoul_bike_sharing$DATE, format="%d/%m/%Y")
```

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HO
<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	H
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	H
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	H
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	H
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	H
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	H

A tibble: 6 × 14

SKILLS NETWORK

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

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

```
# provide your solution here
seoul_bike_sharing$HOUR <- factor(seoul_bike_sharing$HOUR, ordered=TRUE)
```

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HO
<date>	<dbl>	<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
A tibble: 6 x 14												
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	H
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	H
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	H
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	H
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	H
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	H

```
str(seoul_bike_sharing)
```

```
tibble [8,465 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ DATE                 : Date[1:8465], format: "2017-12-01" "2017-12-01" ...
 $ RENTED_BIKE_COUNT    : num [1:8465] 254 204 173 107 78 100 181 460 930 490 ...
 $ HOUR                 : Ord.factor w/ 24 levels "0"<"1"<"2"<"3"<..: 1 2 3 4 5 6 7 8 9 10 ...
 $ TEMPERATURE          : num [1:8465] -5.2 -5.5 -6 -6.2 -6 -6.4 -6.6 -7.4 -7.6 -6.5 ...
 $ HUMIDITY             : num [1:8465] 37 38 39 40 36 37 35 38 37 27 ...
 $ WIND_SPEED           : num [1:8465] 2.2 0.8 1 0.9 2.3 1.5 1.3 0.9 1.1 0.5 ...
 $ VISIBILITY           : num [1:8465] 2000 2000 2000 2000 2000 ...
 $ DEW_POINT_TEMPERATURE: num [1:8465] -17.6 -17.6 -17.7 -17.6 -18.6 -18.7 -19.5 -19.3 -19.8 -22.4 ...
 $ SOLAR_RADIATION      : num [1:8465] 0 0 0 0 0 0 0 0 0.01 0.23 ...
 $ RAINFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SNOWFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SEASONS              : chr [1:8465] "Winter" "Winter" "Winter" "Winter" ...
 $ HOLIDAY              : chr [1:8465] "No Holiday" "No Holiday" "No Holiday" "No Holiday" ...
 $ FUNCTIONING_DAY      : chr [1:8465] "Yes" "Yes" "Yes" "Yes" ...
 - attr(*, "spec")=
  .. cols(
  ..   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()
  .. )
```

Finally, ensure there are no missing values

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

0

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 4 - Dataset Summary

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

Solution 4

```
# provide your solution here
summary(seoul_bike_sharing)
```

```
      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   Length:8465  
 1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Class :character
 Median :0.0100   Median : 0.0000   Median :0.00000   Mode  :character
 Mean   :0.5679   Mean   : 0.1491   Mean   :0.07769                  
 3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000                  
 Max.   :3.5200   Max.   :35.0000   Max.   :8.80000                  

  HOLIDAY          FUNCTIONING_DAY   
 Length:8465        Length:8465       
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

Task 5 - Based on the above stats, calculate how many Holidays there are.

Solution 5:

```
# provide your solution here
count(filter(seoul_bike_sharing['HOLIDAY'], HOLIDAY=="Holiday" ))/24
```

n
<dbl>
17

A data.frame: 1 × 1

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

Solution 6

```
# provide your solution here
(count(filter(seoul_bike_sharing['HOLIDAY'], HOLIDAY=="Holiday" ))/count(seoul_bike_sharing))*100
```

n
<dbl>
4.819846

A data.frame: 1 × 1

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

Solution 7

```
# provide your solution here
365*24
```

8760

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 8 - Given the observations for the 'FUNCTIONING_DAY' how many records must there be?

Solution 8

```
# provide your solution here
count(seoul_bike_sharing['FUNCTIONING_DAY'])
```

n
<int>
A tibble: 1 × 1
8465

```
distinct(seoul_bike_sharing['FUNCTIONING_DAY'])
```

FUNCTIONING_DAY
<chr>
A tibble: 1 × 1
Yes

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

```
# provide your solution here
library(dplyr)
```

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HC
<date>	<dbl>	<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
A tibble: 6 × 14												
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	

```
seoul_bike_sharing %>% group_by (SEASONS) %>% summarize(RAINFALL = sum(RAINFALL), SNOWFALL = sum(SNOWFALL))
```

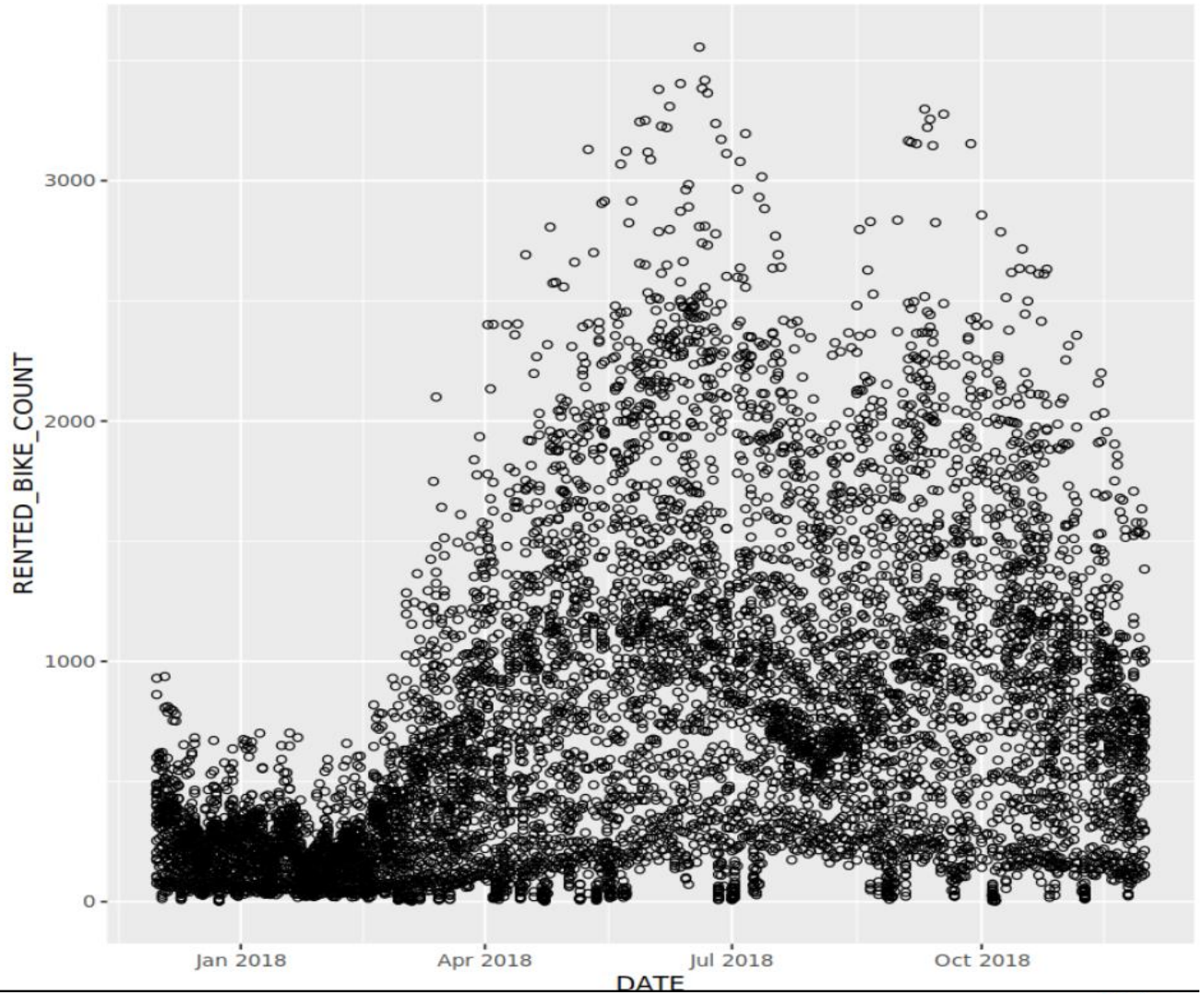
SEASONS	RAINFALL	SNOWFALL
<chr>	<dbl>	<dbl>
A tibble: 4 × 3		
Autumn	227.9	123.0
Spring	403.8	0.0
Summer	559.7	0.0
Winter	70.9	534.6

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

- Create a scatter plot of RENTED_BIKE_COUNT vs DATE

- This scatter plot explains that there appears the state of fluctuations of the count of renting bikes from Jan 2018 to Oct 2018

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT), alpha=0.5) +
  geom_point(shape=1)
```

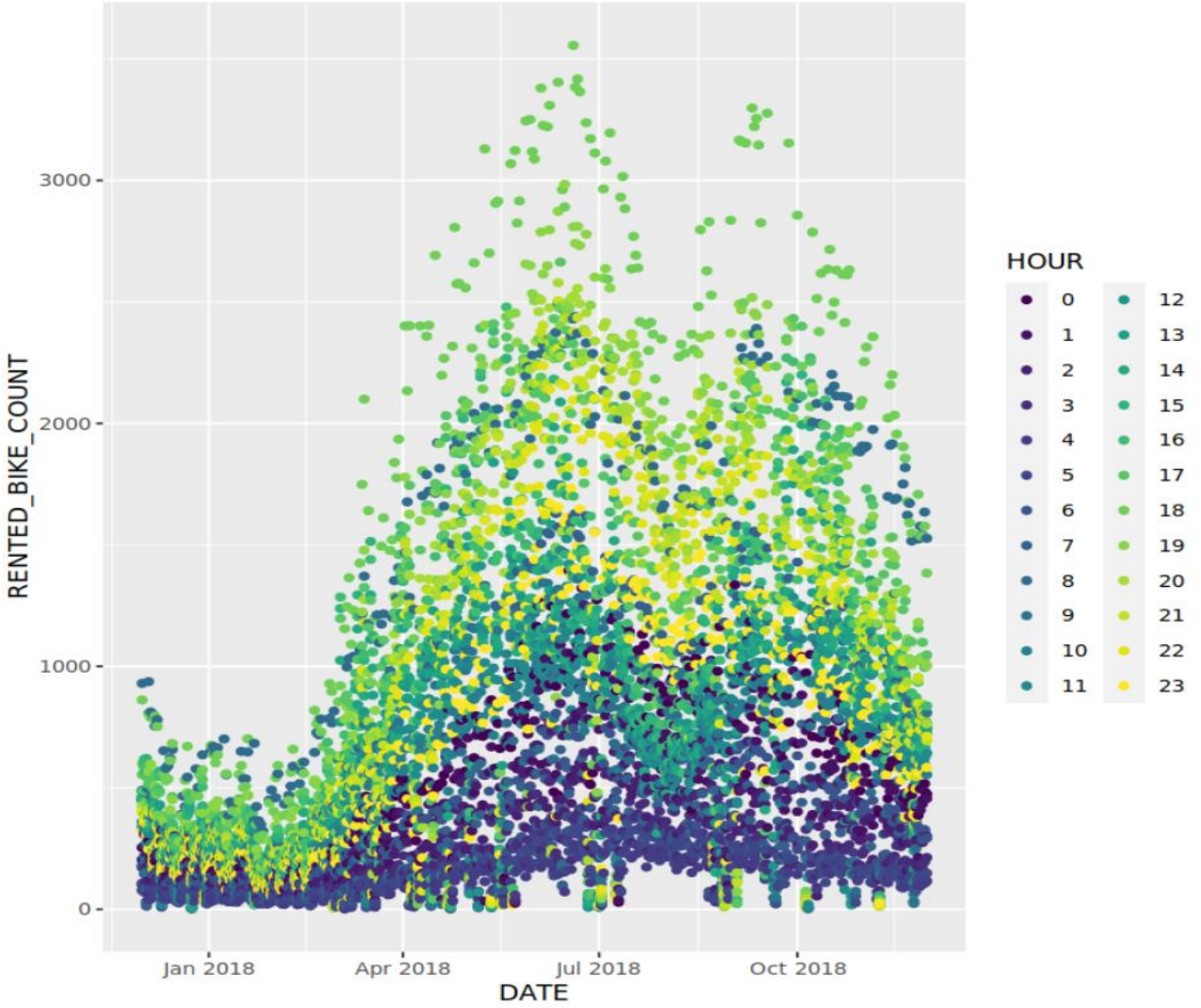


IBM Developer

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT), alpha=0.5) +
geom_point(aes(color=HOUR))
```

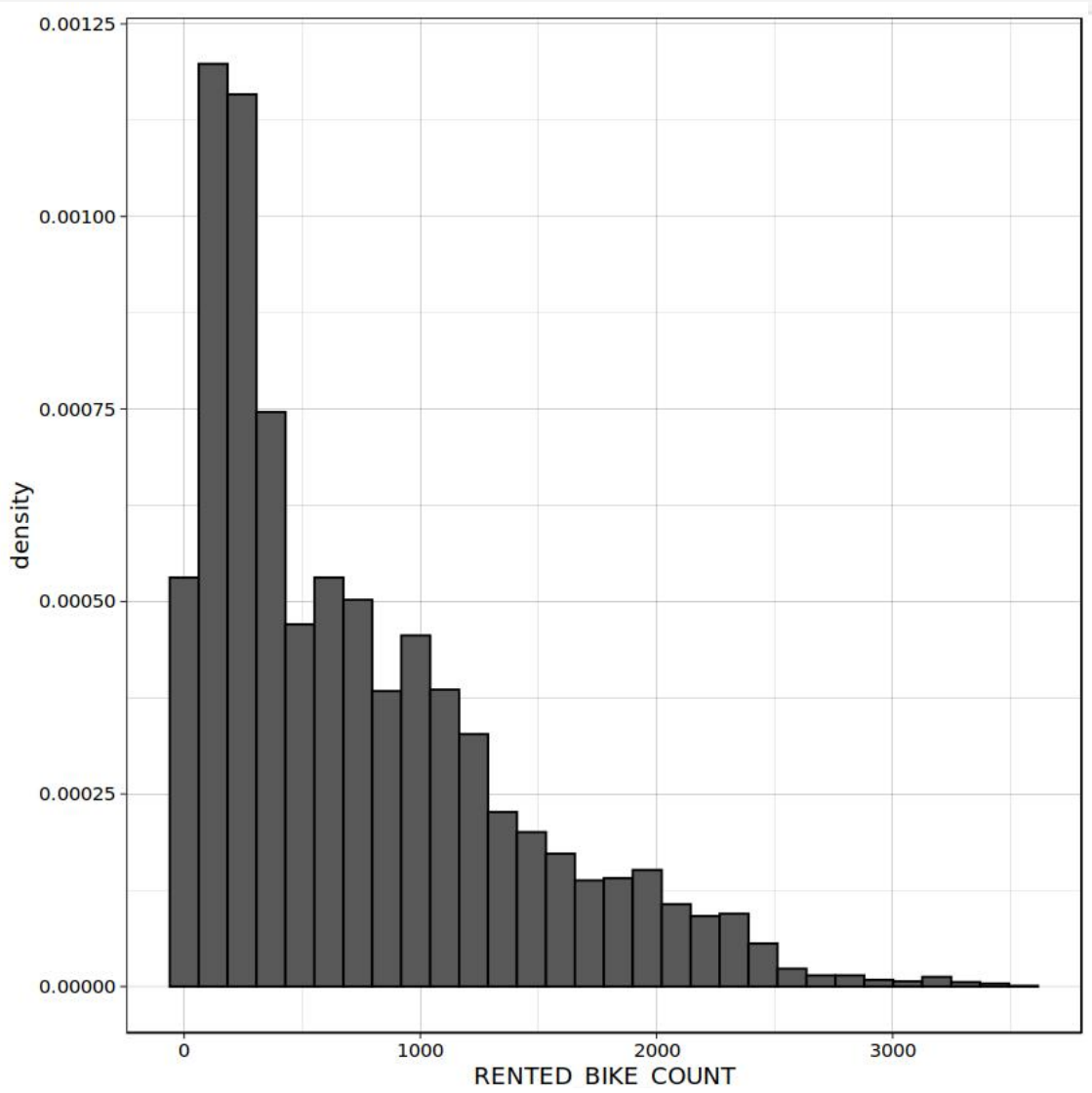
- Create the same plot of the RENTED_BIKE_COUNT time series and add HOURS as the colour.
- The scatter plot marked with HOUR as the color proves that there is a huge difference in the number of rented bicycles at different times. And from 17 to 19 in July 2018, the number of rented bicycles reached its peak.



Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x=RENTED_BIKE_COUNT ), alpha=0.5 )  + geom_histogram(bins=30, col="black",  aes(y=..density..)) +theme_linedraw()
```

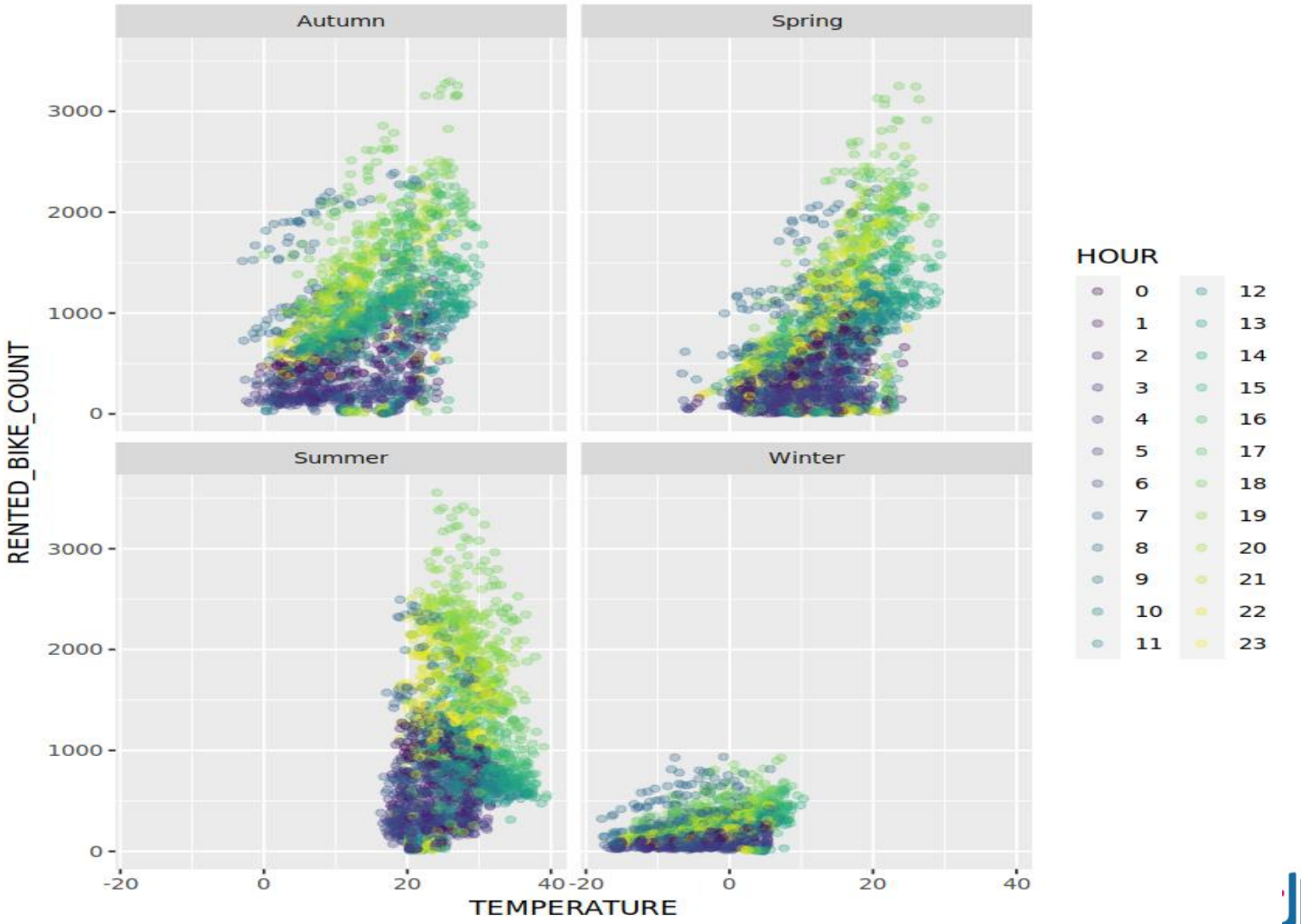
- Create a histogram overlaid with a kernel density curve
- The histogram of the number of rented bikes shows that the time period of renting bikes to the maximum number of 3000 is the least. The time period of renting 0 to 2000 bikes range is at its peak.



Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

- Create a scatter plot to visualize the correlation between RENTED_BIKE_COUNT and TEMPERATURE by SEASONS.
- The scatter plot of the season classification shows that except for the time when the temperature is low in winter, the number of rented bicycles in the other three seasons is relatively scattered. And in winter, almost no one rents more than 1,000 bicycles.

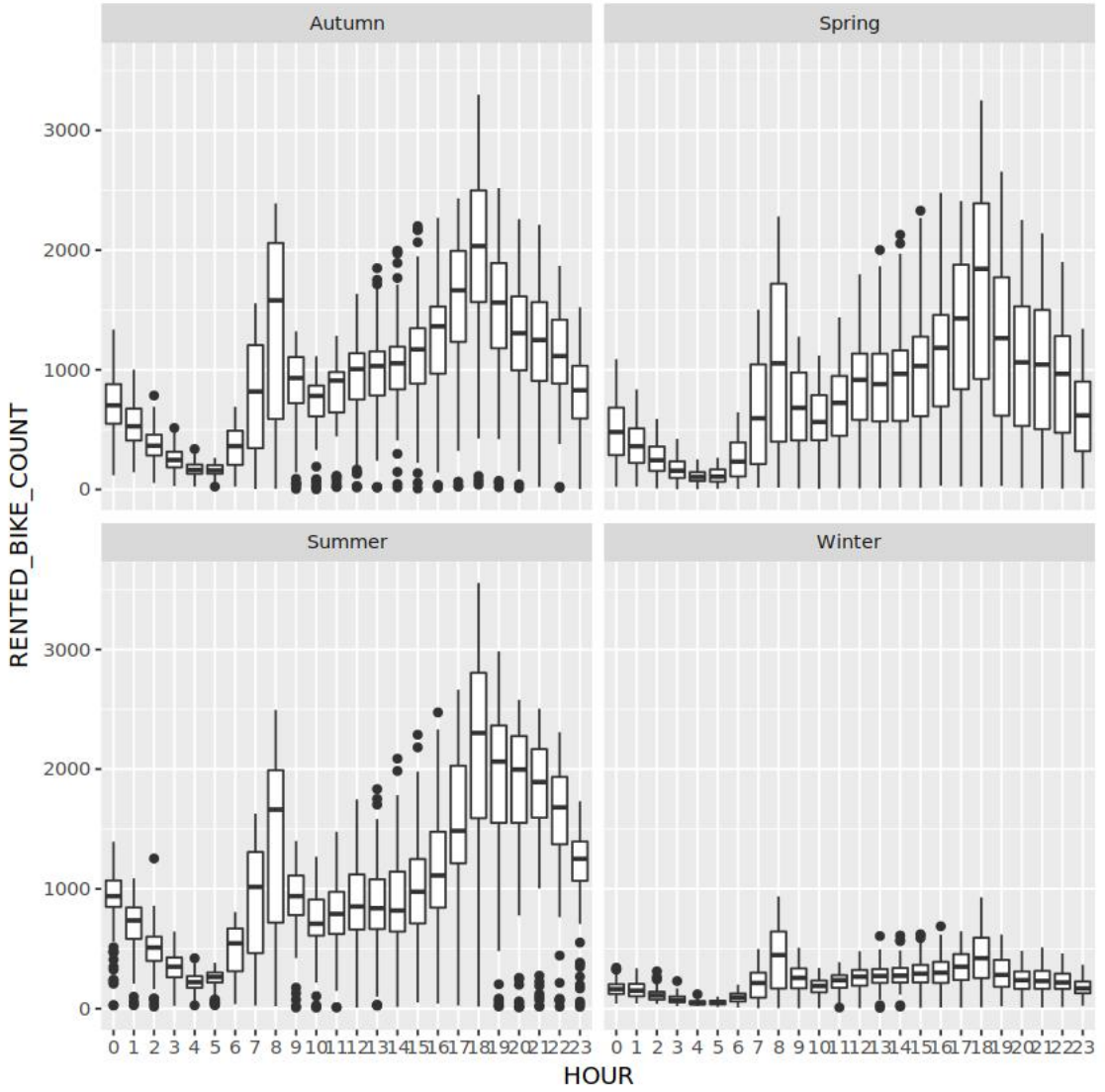
```
# provide your solution here
ggplot(seoul_bike_sharing) +
  geom_point(aes(x=TEMPERATURE,y=RENTED_BIKE_COUNT,colour=HOUR),alpha=0.3) +facet_wrap(.~SEASONS)
```



Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

- Create a display of four boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS.
- From the first three boxplots except winter, it can be seen that the peaks of the number of rental bicycles tend to be consistent. Only in winter there is almost no peak due to the temperature.

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x=HOUR,y=RENTED_BIKE_COUNT)) +
    geom_boxplot() +facet_wrap(.~SEASONS)
```



Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 15 - Group the data by `DATE`, and use the summarize() function to calculate the daily total rainfall and snowfall.

Also, go ahead and plot the results if you wish.

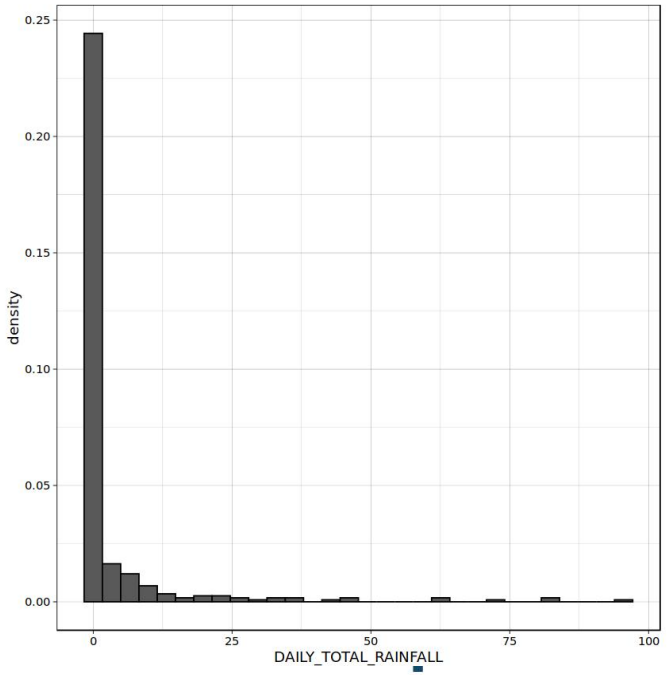
Solution 15

```
# provide your solution here
seoul_daily_rainfall_snowfall<-seoul_bike_sharing %>% group_by (DATE) %>% summarize(DAILY_TOTAL_RAINFALL = sum(RAINFALL), DAILY_TOTAL_SNOW
seoul_daily_rainfall_snowfall
```

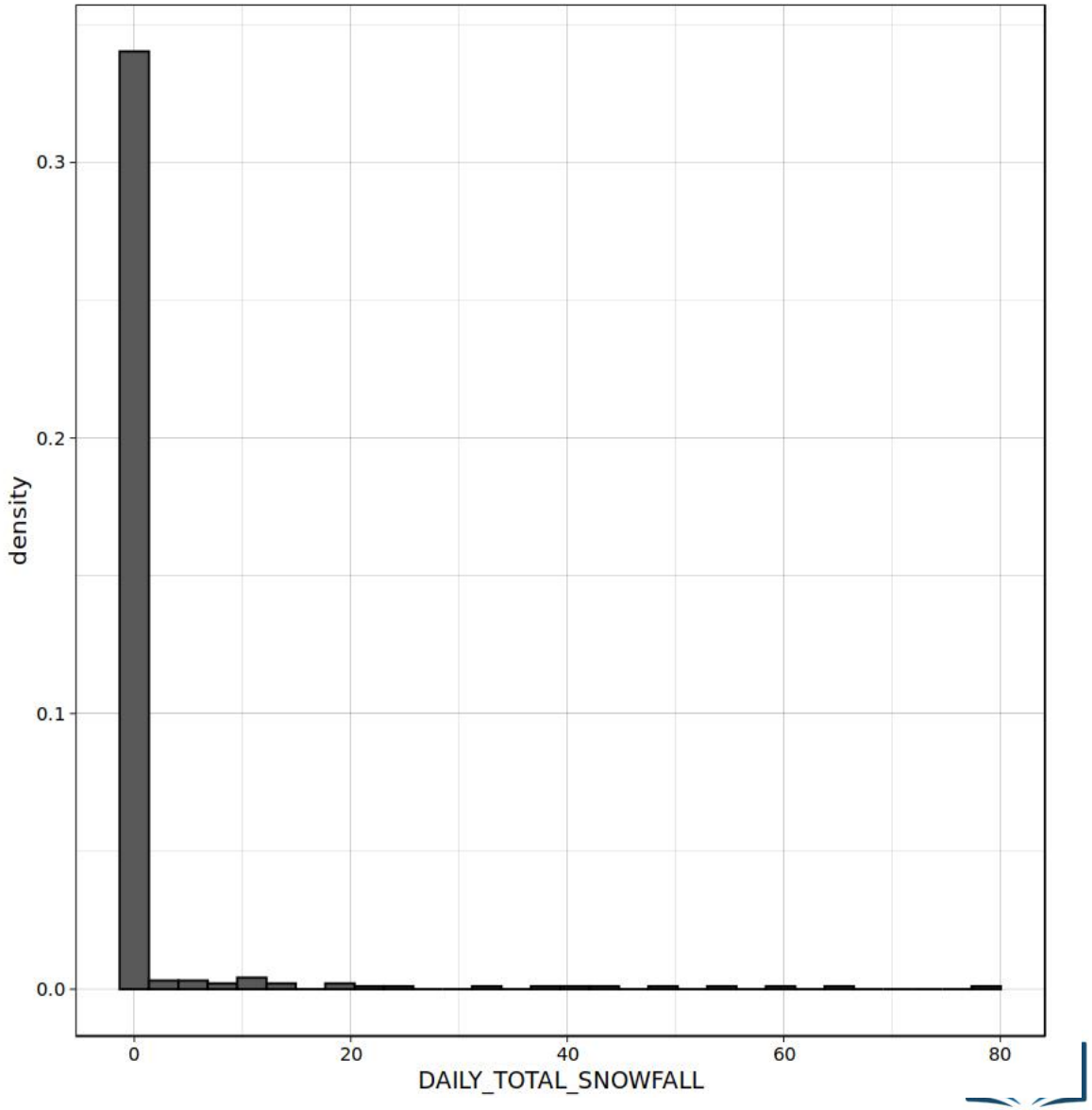
DATE	DAILY_TOTAL_RAINFALL	DAILY_TOTAL_SNOWFALL
<date>	<dbl>	<dbl>

A tibble: 353 × 3

```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DAILY_TOTAL_RAINFALL ), alpha=0.5 ) + geom_histogram(bins=30, col="black",
        aes(y=..density..)) +theme_linedraw()
```

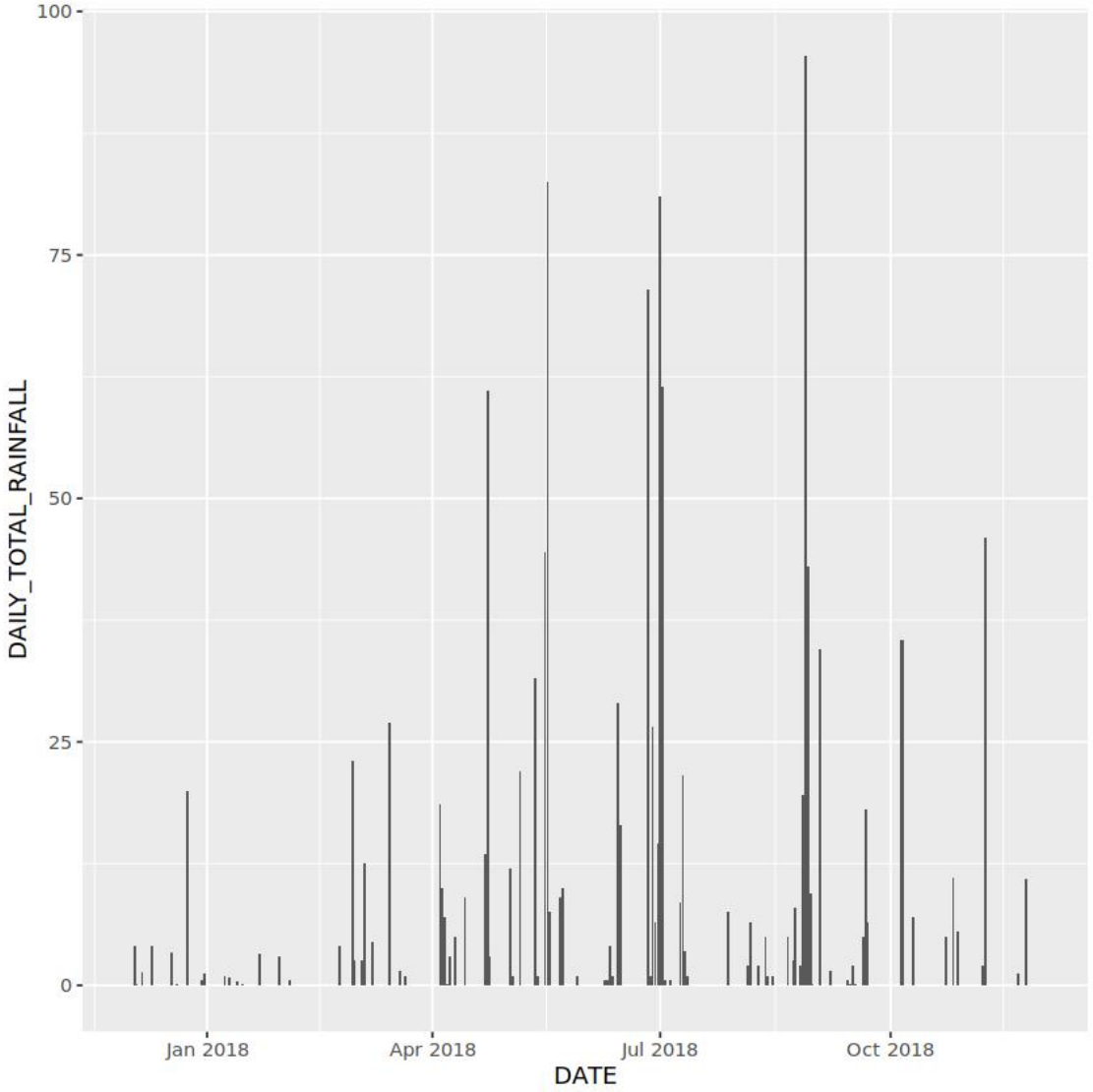



```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DAILY_TOTAL_SNOWFALL ), alpha=0.5 ) + geom_histogram(bins=30, col="black",
        aes(y=..density..)) +theme_linedraw()
```

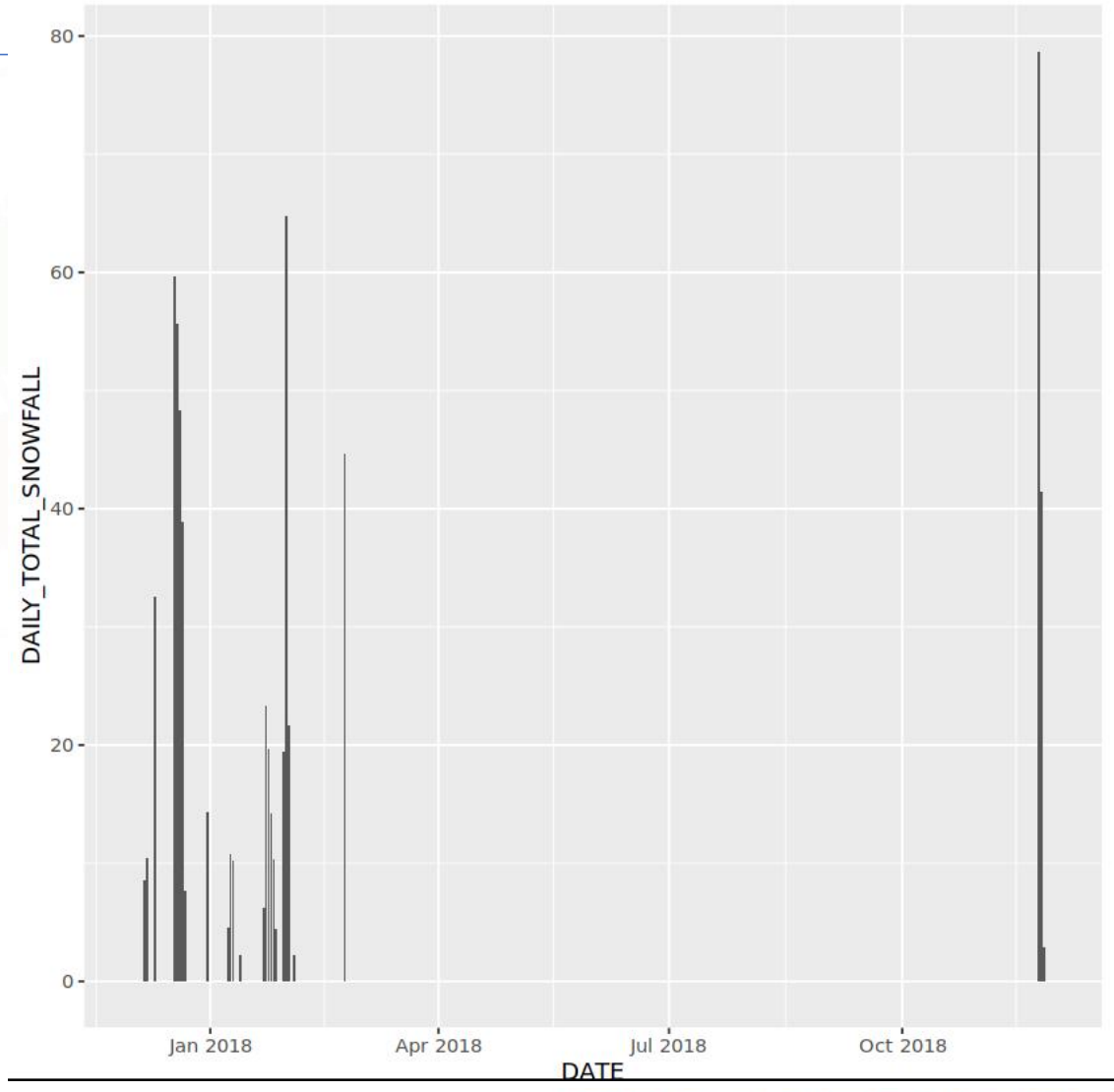


Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DATE, y=DAILY_TOTAL_RAINFALL)) +
  geom_bar(stat = "identity")
```




```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DATE, y=DAILY_TOTAL_SNOWFALL)) +
  geom_bar(stat = "identity")
```



IBM Developer

SKILLS NETWORK

Results: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 16 - Determine how many days had snowfall.

Solution 16

```
# provide your solution here
head(seoul_daily_rainfall_snowfall)
filter(seoul_daily_rainfall_snowfall, DAILY_TOTAL_SNOWFALL>0) %>% count()
```

DATE	DAILY_TOTAL_RAINFALL	DAILY_TOTAL_SNOWFALL
<date>	<dbl>	<dbl>
A tibble: 6 × 3		
2017-12-01	0.0	0.0
2017-12-02	0.0	0.0
2017-12-03	4.0	0.0
2017-12-04	0.1	0.0
2017-12-05	0.0	0.0
2017-12-06	1.3	8.6

n
<int>
A tibble: 1 × 1
27

Results: Predictive Analysis with building a Baseline Regression Model

TASK: Build a linear regression model using weather variables only

```
# print(lm_model_weather$fit)
print(lm_model_weather$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY +
    WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION +
    RAINFALL + SNOWFALL, data = data)

Coefficients:
         (Intercept)           TEMPERATURE              HUMIDITY
             171.370              2210.928             -1003.792
        WIND_SPEED            VISIBILITY  DEW_POINT_TEMPERATURE
             422.516                 4.582              -100.903
    SOLAR_RADIATION              RAINFALL              SNOWFALL
            -407.845             -2090.642               331.864
```

TASK: Build a linear regression model using all variables

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ ., data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-1410.1  -217.0    -8.7   200.7  2025.1

Coefficients: (3 not defined because of singularities)
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                254.625     50.277   5.064 4.21e-07 ***
TEMPERATURE                590.970    208.972   2.828 0.004699 **
HUMIDITY                  -974.192     97.813  -9.960  < 2e-16 ***
WIND_SPEED                   3.264     39.813   0.082 0.934662
VISIBILITY                   3.338     19.826   0.168 0.866281
DEW_POINT_TEMPERATURE      796.057    217.519   3.660 0.000255 ***
SOLAR_RADIATION            291.995     41.101   7.104 1.34e-12 ***
RAINFALL                 -2317.209    166.583 -13.910  < 2e-16 ***
SNOWFALL                   263.825     96.700   2.728 0.006384 **
`0`                        -16.895     33.520  -0.504 0.614249
`1`                       -128.323     33.375  -3.845 0.000122 ***
`10`                      -220.389     32.375  -6.807 1.09e-11 ***
`11`                      -220.593     33.175  -6.649 3.19e-11 ***
`12`                      -203.231     34.280  -5.928 3.22e-09 ***
`13`                      -179.463     34.248  -5.240 1.66e-07 ***
`14`                      -183.466     33.911  -5.410 6.52e-08 ***
`15`                      -108.625     33.647  -3.228 0.001251 **
`16`                        34.530     33.664   1.026 0.305062
`17`                       336.930     33.580  10.034  < 2e-16 ***
`18`                       797.708     34.028  23.443  < 2e-16 ***
`19`                       495.566     34.124  14.522  < 2e-16 ***
`2`                       -220.132     33.173  -6.636 3.49e-11 ***
`20`                       449.443     33.551  13.396  < 2e-16 ***
`21`                       454.048     33.940  13.378  < 2e-16 ***
`22`                       344.139     33.625  10.234  < 2e-16 ***
`23`                       103.665     33.697   3.076 0.002104 **
`3`                       -307.956     33.413  -9.217  < 2e-16 ***
`4`                       -387.267     33.047 -11.719  < 2e-16 ***
`5`                       -367.805     33.257 -11.059  < 2e-16 ***
`6`                       -203.367     33.188  -6.128 9.45e-10 ***
`7`                         92.548     33.193   2.788 0.005316 **
`8`                        454.810     32.460  14.012  < 2e-16 ***
`9`                             NA         NA      NA       NA
AUTUMN                     360.852     20.138  17.919  < 2e-16 ***
SPRING                     202.465     19.188  10.552  < 2e-16 ***
SUMMER                     212.849     28.906   7.364 2.02e-13 ***
WINTER                          NA         NA      NA       NA
HOLIDAY                   -99.749     21.812  -4.573 4.89e-06 ***
NO_HOLIDAY                      NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 372.8 on 6312 degrees of freedom
```

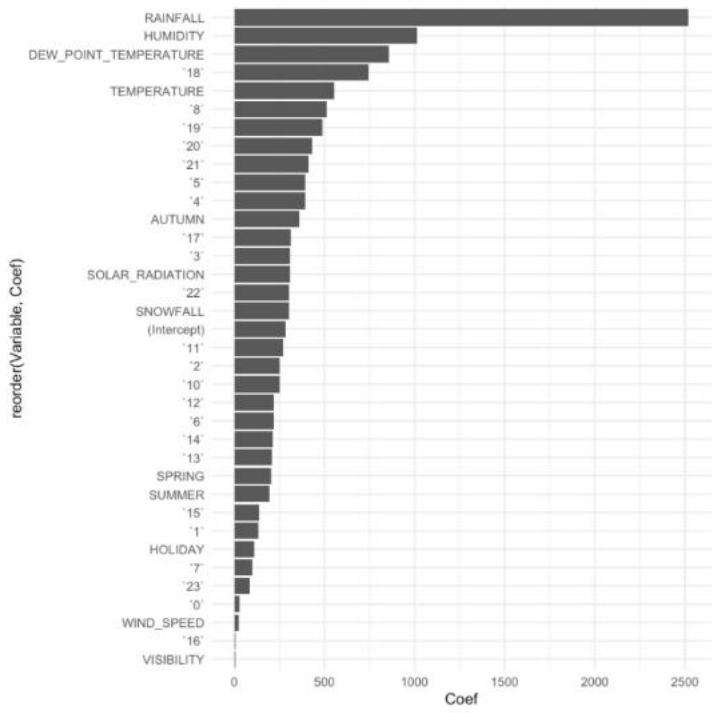
Results: Predictive Analysis with building a Baseline Regression Model

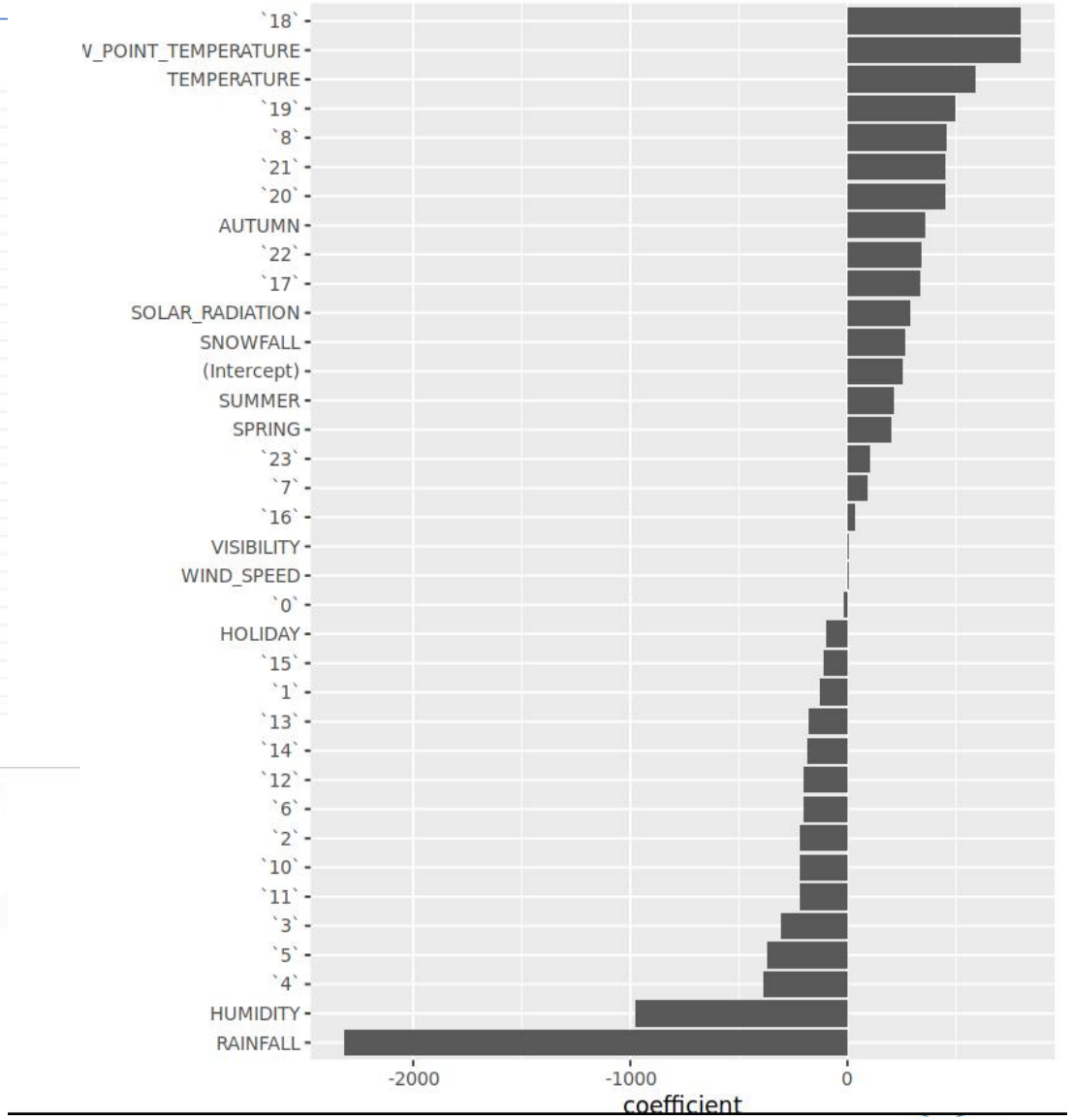
TASK: Model evaluation and identification of important variables

```
# Visualize the list using ggplot and geom_bar
ggplot(Coefficients_df, aes(reorder(Variables, coefficient) , coefficient))+ geom_col()+  coord_flip()
```

- Compare the performance of lm_model_weather and lm_model_all by using R^2 / R-squared and Root Mean Squared Error (RMSE)
- It can be seen from the graphs that tainfall is the most coefficient as variables after ranking coefficients.

You should see a sorted coefficient bar chart like the following example:





IBM Developer

Results: Predictive Analysis with improving the linear model

TASK: Add polynomial terms

```
# Fit a linear model with higher order polynomial on some important variables

# #HINT: Use ploy function to build polynomial terms, lm_poly <- RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) .....
lm_poly <- RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY, 4)+ poly(WIND_SPEED, 4)+ poly(VISIBILITY, 3)+
                               poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 4) +
                               `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `18` + `19` + `2` + `20` +
                               `21` + `22` + `23` + `3` + `4` + `5` + `6` + `7` + `8`+ `9` + AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY

train_fit <- lm_spec %>%
  fit(lm_poly,  data=train_data)
```

```
# Print model summary

# summary(lm_poly$fit)
summary(train_fit$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) +
    poly(HUMIDITY, 4) + poly(WIND_SPEED, 4) + poly(VISIBILITY,
    3) + poly(DEW_POINT_TEMPERATURE, 6) + poly(SOLAR_RADIATION,
    5) + poly(RAINFALL, 6) + poly(SNOWFALL, 4) + `0` + `1` +
    `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `18` +
    `19` + `2` + `20` + `21` + `22` + `23` + `3` + `4` + `5` +
    `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER + WINTER +
    HOLIDAY + NO_HOLIDAY, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-1740.41  -181.03     5.47   166.41  1273.15

Coefficients: (3 not defined because of singularities)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    268.266     25.716  10.432  < 2e-16 ***
poly(TEMPERATURE, 6)1        38602.565   4229.867   9.126  < 2e-16 ***
poly(TEMPERATURE, 6)2         9664.062    709.405  13.623  < 2e-16 ***
poly(TEMPERATURE, 6)3       -10741.230    452.201 -23.753  < 2e-16 ***
poly(TEMPERATURE, 6)4        -7261.710    429.991 -16.888  < 2e-16 ***
poly(TEMPERATURE, 6)5           28.176    373.887   0.075 0.939930
poly(TEMPERATURE, 6)6         1929.759    359.981   5.361 8.57e-08 ***
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
127.102562	254
-3.939999	204
-232.061186	100
80.637429	460
315.503696	930
539.709057	398

Another minor improvement we could do here is to convert all negative prediction results to zero, be

```
# e.g., test_results[test_results<0] <- 0
test_results[test_results<0] <- 0
```

Now, calculate R-squared and RMSE for the test results generated by `lm_ploy` model

```
# Calculate R-squared and RMSE from the test results
rsq_1<-rsq(test_results, truth = truth, estimate = .pred)

rmse_1<-rmse(test_results, truth = truth, estimate = .pred)


model_1_results<-c( rsq_1, rmse_1)
model_1_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.771285774483661
$.metric	'rmse'
$.estimator	'standard'

Results: Predictive Analysis with improving the linear model

TASK: Add interaction terms

```
parsnip model object


Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ `18` * TEMPERATURE +
    DEW_POINT_TEMPERATURE + RAINFALL * HUMIDITY * `4` + SOLAR_RADIATION *
    SNOWFALL + WIND_SPEED * VISIBILITY + `18` * TEMPERATURE +
    +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) + poly(WIND_SPEED,
    4) + poly(VISIBILITY, 3) + poly(DEW_POINT_TEMPERATURE, 6) +
    poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6) + poly(SNOWFALL,
    4) + `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` +
    `16` + `17` + `19` + `2` + `20` + `21` + `22` + `23` + `3` +
    `5` + `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER +
    WINTER + HOLIDAY + NO_HOLIDAY, data = data)

Coefficients:
                      (Intercept)                               `18`
                          1905.03                             -54.45
                      TEMPERATURE            DEW_POINT_TEMPERATURE
                         -1745.35                           -7335.56
                         RAINFALL                           HUMIDITY
                        -11539.95                             587.40
                              `4`                     SOLAR_RADIATION
                          -21.80                             503.07
                         SNOWFALL                         WIND_SPEED
                          -62.54                             -183.19
                       VISIBILITY                 poly(TEMPERATURE, 6)1
                          -91.16                                 NA
      poly(TEMPERATURE, 6)2      poly(TEMPERATURE, 6)3
                        -9084.46                          -10880.02
      poly(TEMPERATURE, 6)4      poly(TEMPERATURE, 6)5
                        -7137.86                           -1022.73
      poly(TEMPERATURE, 6)6         poly(HUMIDITY, 4)1
                         -432.87                                 NA
         poly(HUMIDITY, 4)2         poly(HUMIDITY, 4)3
                        -1751.93                           -3850.38
         poly(HUMIDITY, 4)4        poly(WIND_SPEED, 4)1
                         -897.96                                 NA
        poly(WIND_SPEED, 4)2        poly(WIND_SPEED, 4)3
                        -1285.05                           -697.99
        poly(WIND_SPEED, 4)4        poly(VISIBILITY, 3)1
                          368.81                                 NA
       poly(VISIBILITY, 3)2        poly(VISIBILITY, 3)3
                          388.98                           -1391.17
poly(DEW_POINT_TEMPERATURE, 6)1 poly(DEW_POINT_TEMPERATURE, 6)2
```

```
# Print model summary
summary(train_fit_2$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ `18` * TEMPERATURE +
    DEW_POINT_TEMPERATURE + RAINFALL * HUMIDITY * `4` + SOLAR_RADIATION *
    SNOWFALL + WIND_SPEED * VISIBILITY + `18` * TEMPERATURE +
    +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) + poly(WIND_SPEED,
    4) + poly(VISIBILITY, 3) + poly(DEW_POINT_TEMPERATURE, 6) +
    poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6) + poly(SNOWFALL,
    4) + `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` +
    `16` + `17` + `19` + `2` + `20` + `21` + `22` + `23` + `3` +
    `5` + `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER +
    WINTER + HOLIDAY + NO_HOLIDAY, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-2069.64  -175.47     5.46   165.68  1260.71

Coefficients: (22 not defined because of singularities)
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      1905.03     781.06   2.439 0.014753
`18`                              -54.45     288.23  -0.189 0.850175
TEMPERATURE                     -1745.35    1473.33  -1.185 0.236205
DEW_POINT_TEMPERATURE           -7335.56    2505.56  -2.928 0.003426
RAINFALL                       -11539.95    6391.21  -1.806 0.071026
HUMIDITY                         587.40     304.09   1.932 0.053438
`4`                              -21.80      79.83  -0.273 0.784778
SOLAR_RADIATION                  503.07      45.49  11.059  < 2e-16
SNOWFALL                         -62.54      96.50  -0.648 0.516956
WIND_SPEED                      -183.19      79.64  -2.300 0.021466
VISIBILITY                       -91.16      26.51  -3.438 0.000589
poly(TEMPERATURE, 6)1                NA         NA      NA       NA
poly(TEMPERATURE, 6)2           -9084.46    6760.65  -1.344 0.179082
poly(TEMPERATURE, 6)3          -10880.02     488.43 -22.276  < 2e-16
poly(TEMPERATURE, 6)4           -7137.86     433.18 -16.478  < 2e-16
poly(TEMPERATURE, 6)5           -1022.73     496.35  -2.061 0.039388
poly(TEMPERATURE, 6)6            -432.87     783.10  -0.553 0.580438
poly(HUMIDITY, 4)1                   NA         NA      NA       NA
poly(HUMIDITY, 4)2              -1751.93     975.10  -1.797 0.072436
poly(HUMIDITY, 4)3              -3850.38     643.07  -5.987 2.24e-09
poly(HUMIDITY, 4)4               -897.96     470.62  -1.908 0.056429
poly(WIND_SPEED, 4)1                 NA         NA      NA       NA
poly(WIND_SPEED, 4)2            -1285.05     324.56  -3.959 7.59e-05
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
115.35980	254
-25.03473	204
-246.11749	100
85.92372	460
352.11412	930
538.93776	398

```
rsq_2<-rsq(test_results_2, truth = truth, estimate = .pred)

rmse_2<- rmse(test_results_2, truth = truth, estimate = .pred)

model_2_results<-c(rsq_2, rmse_2)
model_2_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.772629482653108
$.metric	'rmse'
$.estimator	'standard'
$.estimate	304.46639380277

SKILLS NETWORK

Results: Predictive Analysis with improving the linear model

TASK: Add regularization

```
test_results_glment <- glmnet_fit %>%
predict(new_data = test_data) %>%
mutate(truth = test_data$RENTED_BIKE_COUNT)

head(test_results_glment)
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
125.97474	254
-13.93975	204
-235.76937	100
97.92126	460
364.40560	930
527.00551	398

```
test_results_glment[test_results_glment<0] <- 0
rsq_3<-rsq(test_results_glment, truth = truth, estimate = .pred)

rmse_3<-rmse(test_results_glment, truth = truth, estimate = .pred)

model_3_results<-c(rsq_3, rmse_3)
model_3_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.776136730923099
$.metric	'rmse'
$.estimator	'standard'
$.estimate	302.333828956636

TASK: Experiment to search for improved models

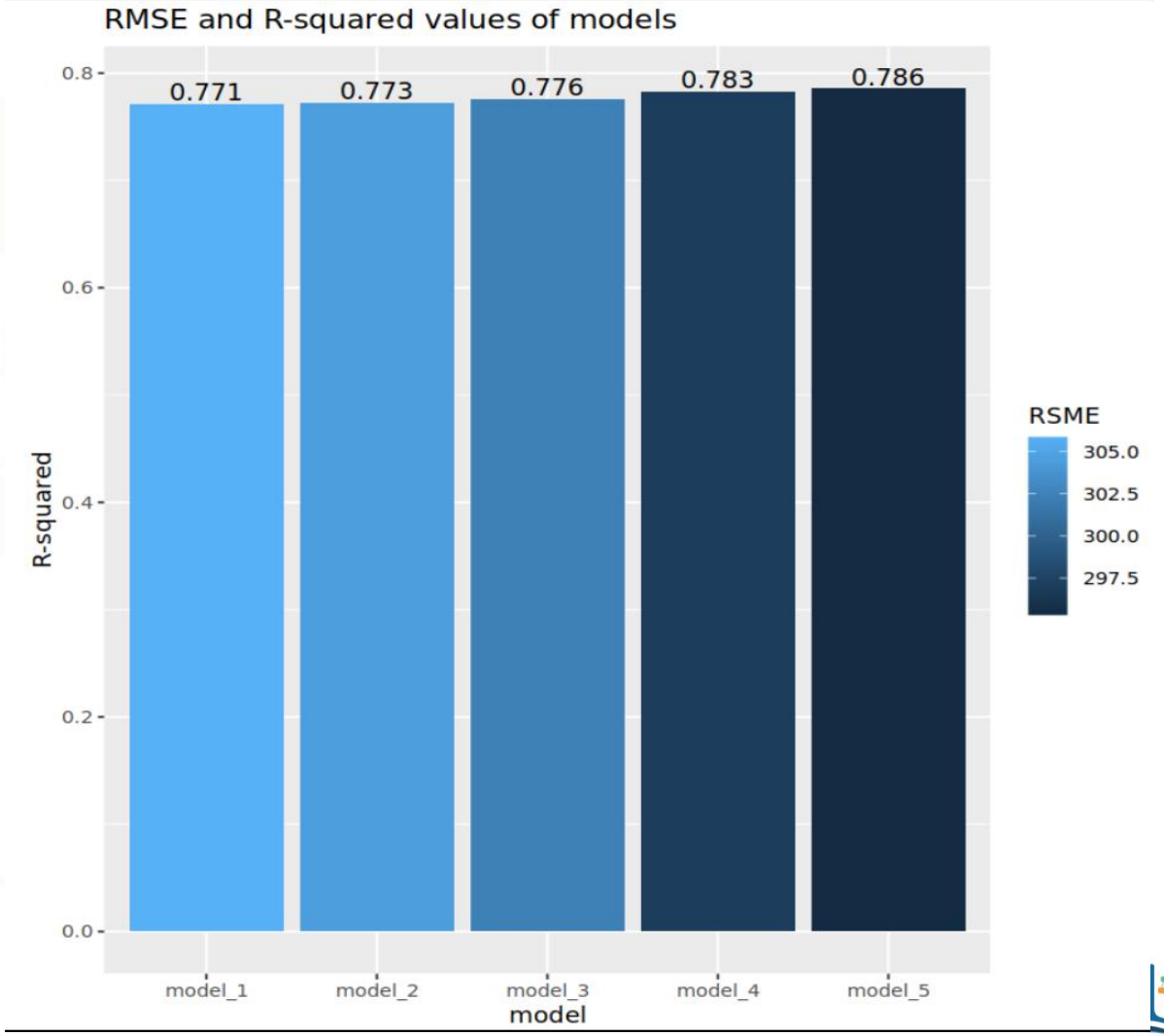
```
rsq_rsme_data<-data.frame(model_1_results)
rsq_rsme_data<-rbind(rsq_rsme_data, model_2_results, model_3_results, model_4_results, model_5_results)
rsq_rsme_data['model']<-c("model_1", "model_2", "model_3", "model_4", "model_5")
colnames(rsq_rsme_data)[6]<-"RSME"
colnames(rsq_rsme_data)[3]<-"Rsquared"
rsq_rsme_data
```

	.metric	.estimator	Rsquared	.metric.1	.estimator.1	RSME	model
	<fct>	<fct>	<dbl>	<fct>	<fct>	<dbl>	<chr>
A data.frame: 5 × 7							
1	rsq	standard	0.7712858	rmse	standard	305.8488	model_1
2	rsq	standard	0.7726295	rmse	standard	304.4664	model_2
3	rsq	standard	0.7761367	rmse	standard	302.3338	model_3
4	rsq	standard	0.7829326	rmse	standard	296.9778	model_4
5	rsq	standard	0.7857093	rmse	standard	295.3345	model_5

Results: Predictive Analysis with improving the linear model

- It can be seen from the bar chart that model_5 of polynominal has the lowest RSME scores and highest R-squared values

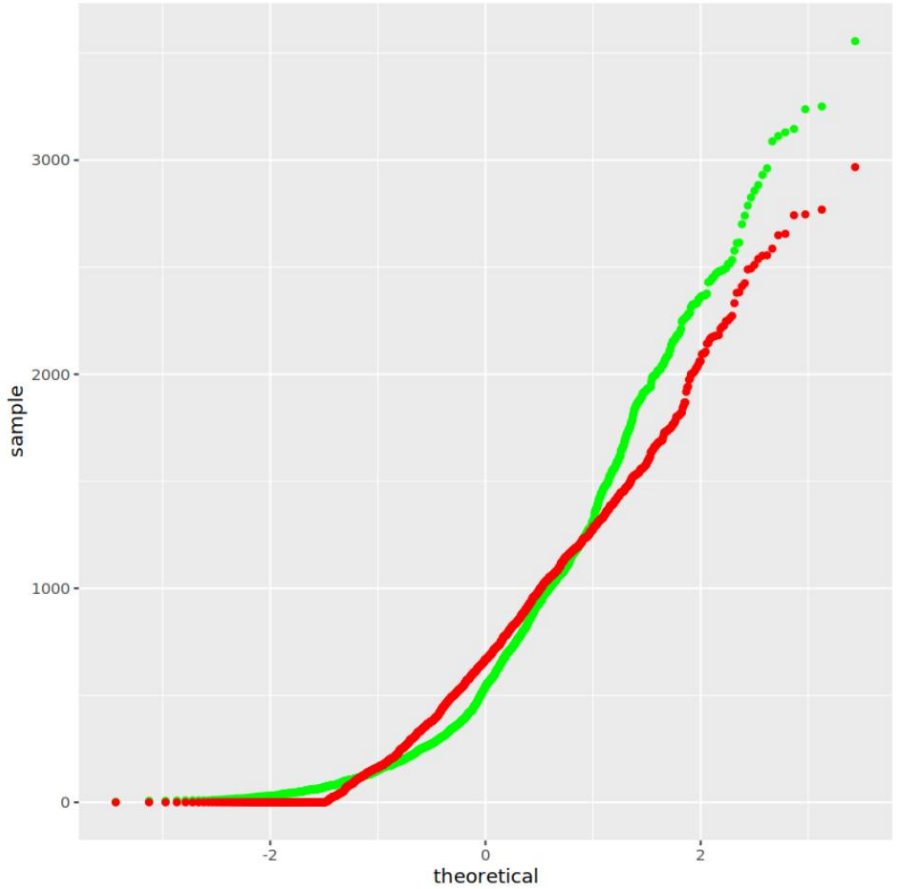
```
# HINT: Use ggplot() + geom_bar()
library(ggplot2)

ggplot(rsq_rsme_data, aes(fill=RSME, x=model, y=Rsquared))+
  geom_bar(position="stack", stat="identity")+
  geom_text(aes(label = format(round(Rsquared, 3), nsmall = 2)), vjust = -0.2, size = 4)+
  ylab("R-squared")+
  ggtitle("RMSE and R-squared values of models ")
```



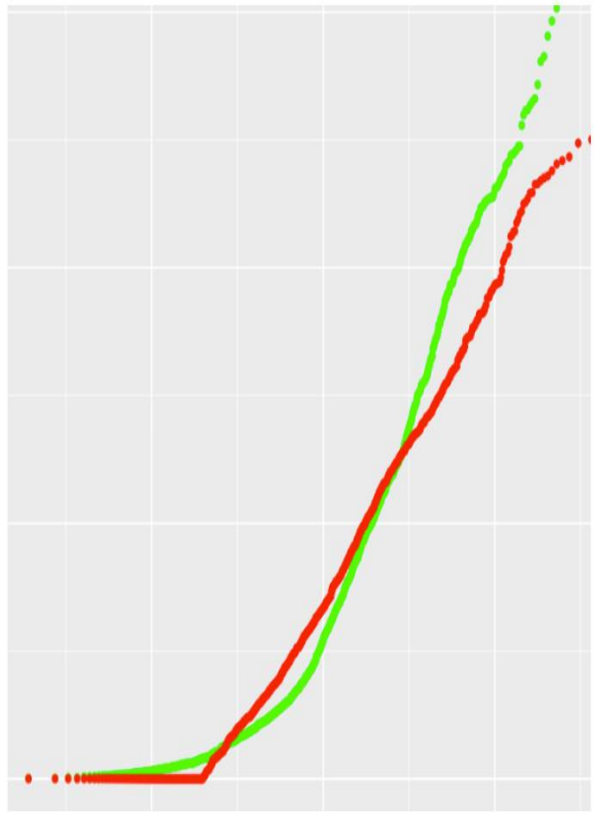
Results: Predictive Analysis with improving the linear model

```
# HINT: Use ggplot() +
    # stat_qq(aes(sample=truth), color='green') +
    # stat_qq(aes(sample=prediction), color='red')
    ggplot(results_model_5)+
  stat_qq(aes(sample=truth), color='green') +
  stat_qq(aes(sample=.pred), color='red')
```

- Create a Q-Q plot by plotting the distribution difference between the predictions generated by the best model and the true values on the test dataset.

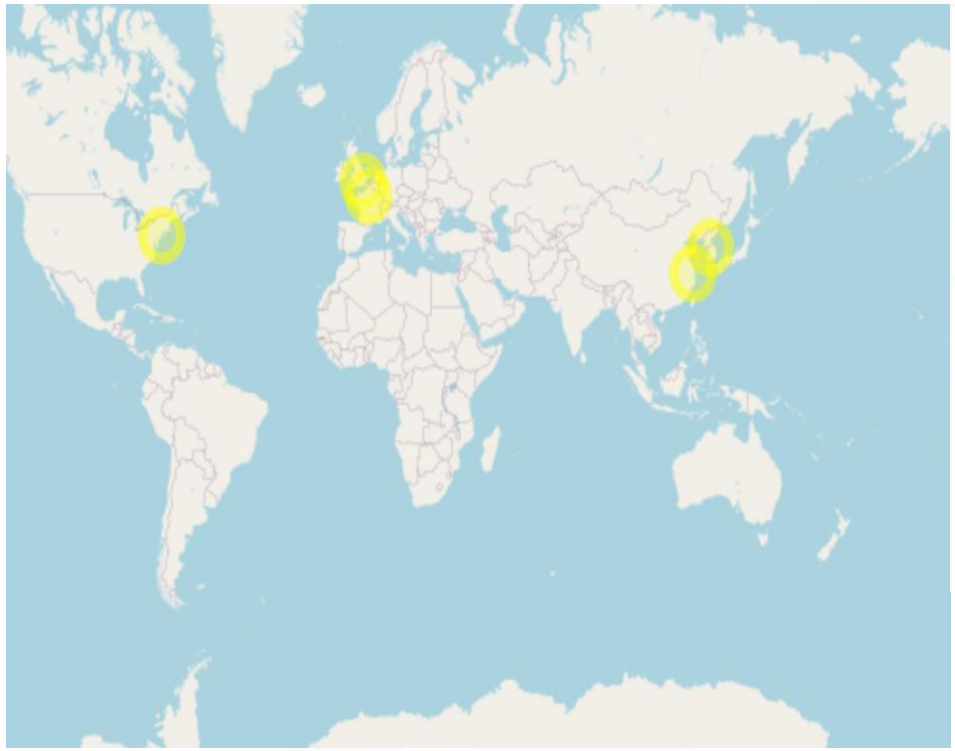


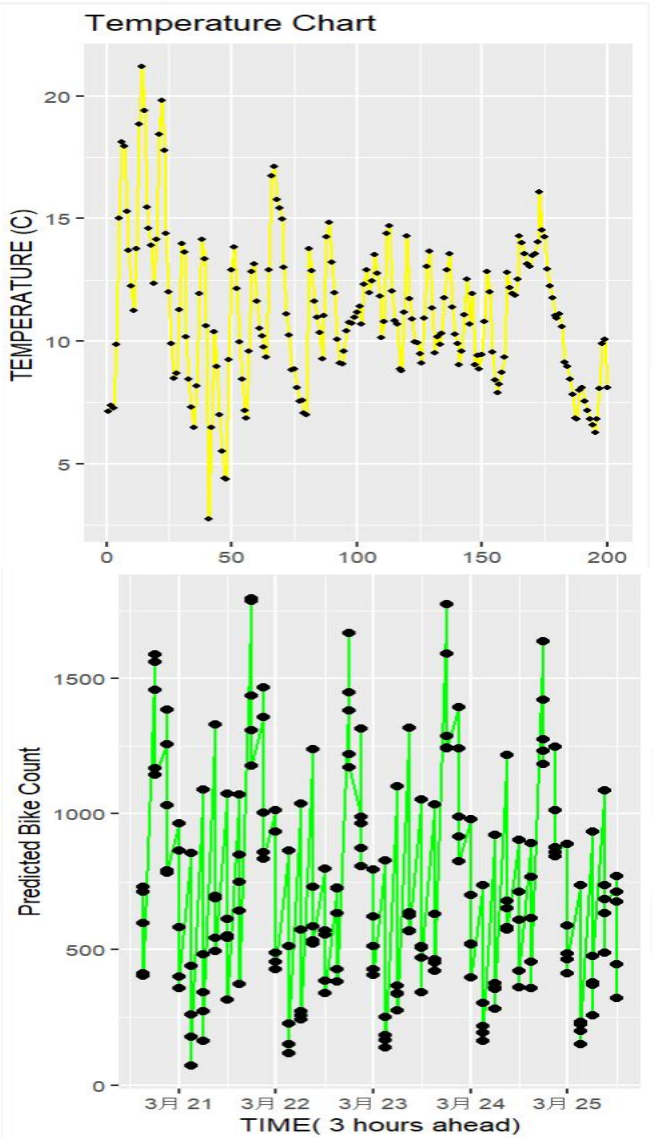
One example of such Q-Q plot may look like this:



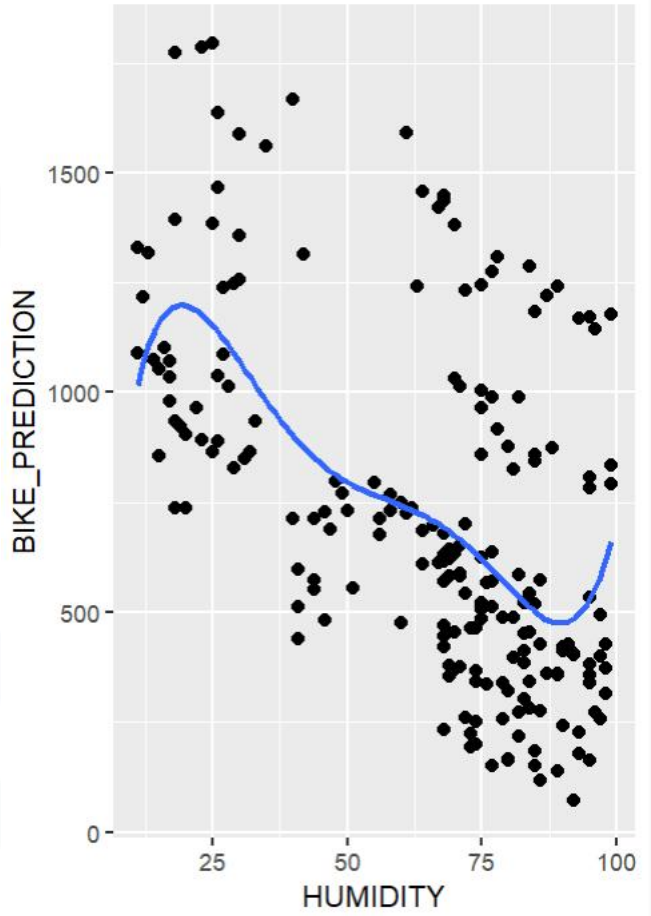
Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for the **all** 5 cities






```
Time=
BikeCountPred=
```

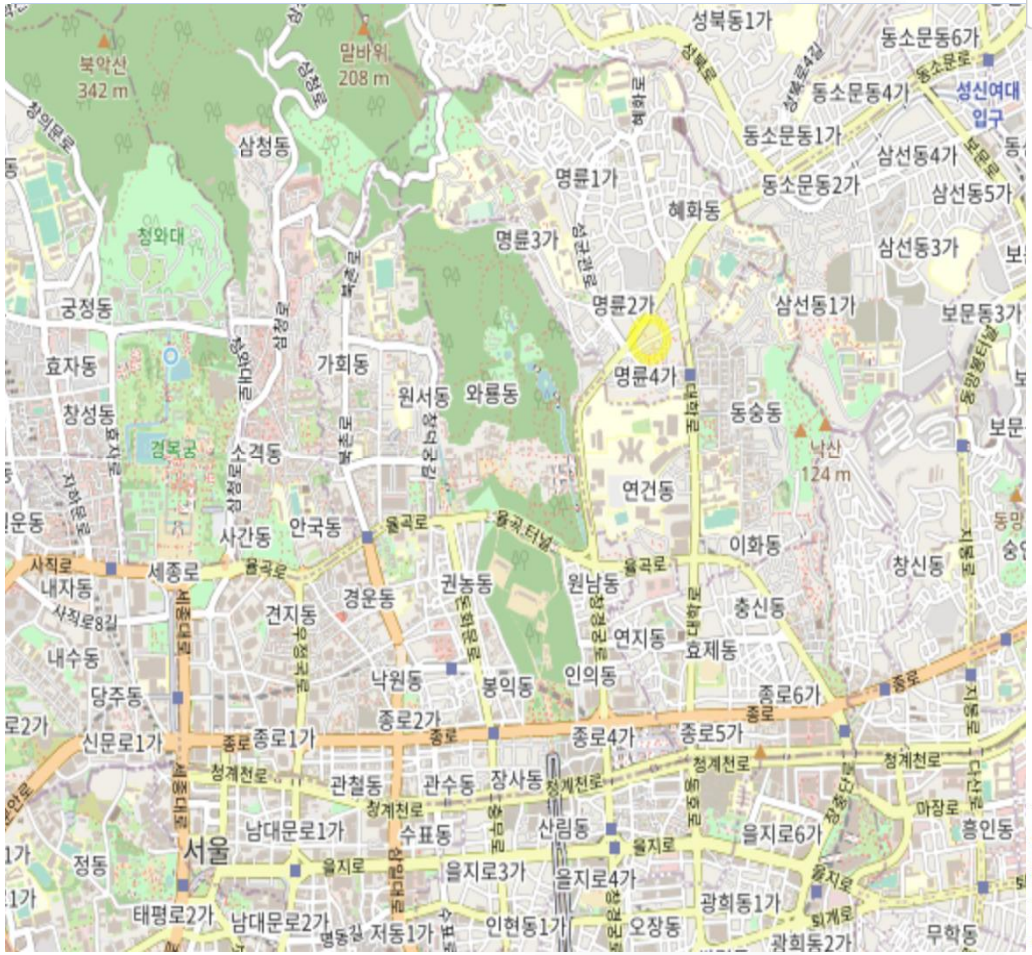


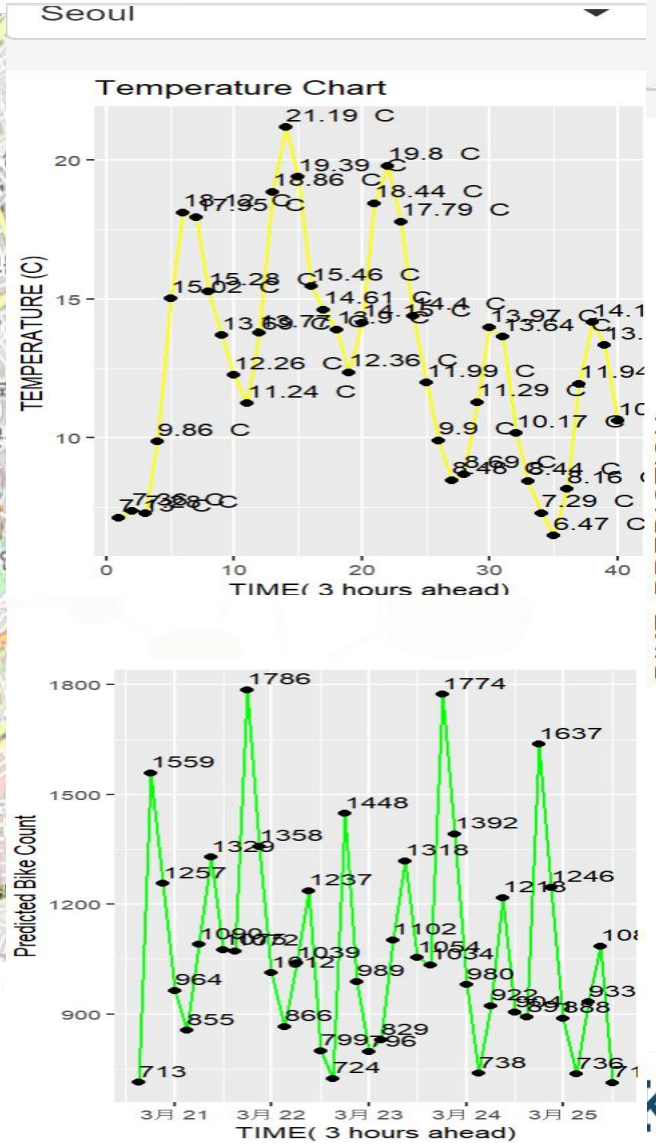
IBM Developer

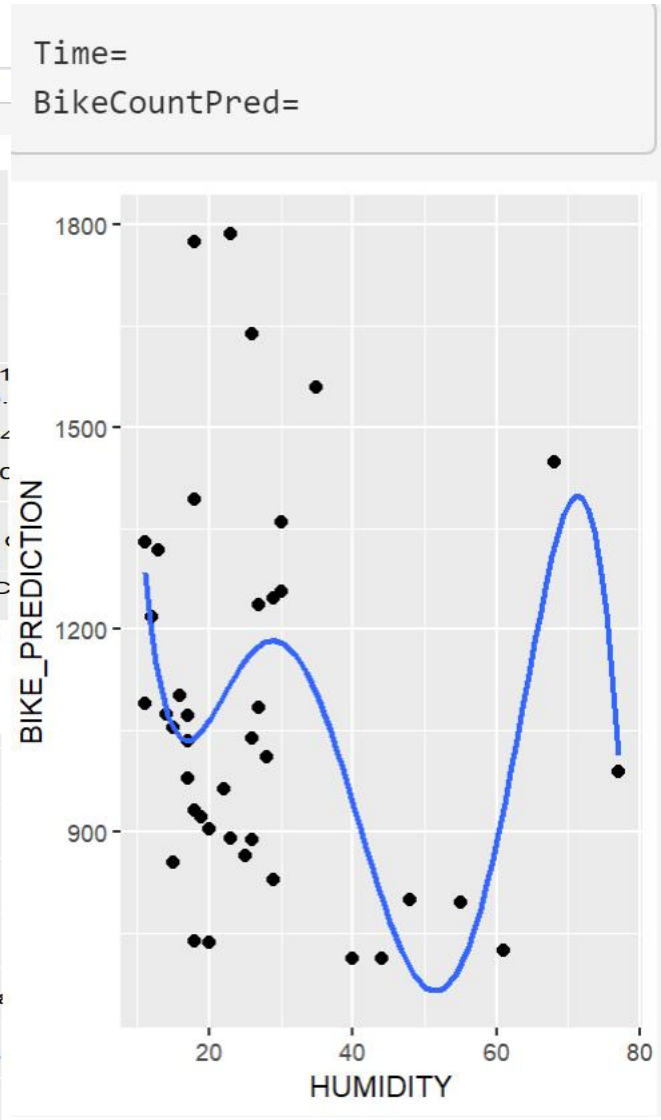
SKILLS NETWORK

Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for **Seoul**





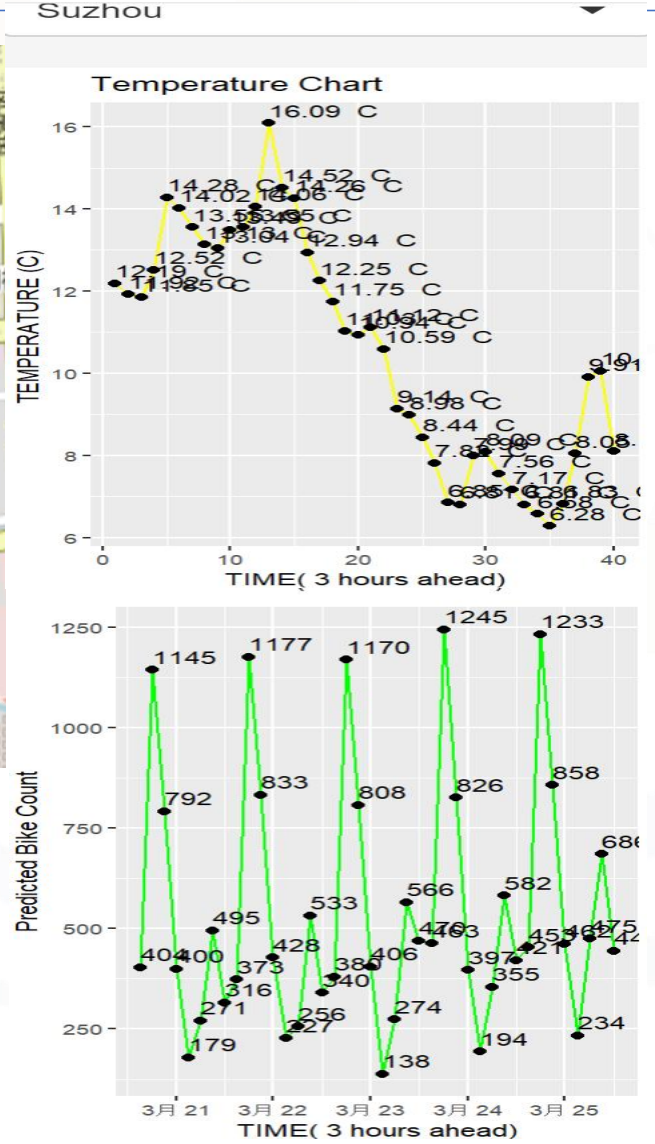


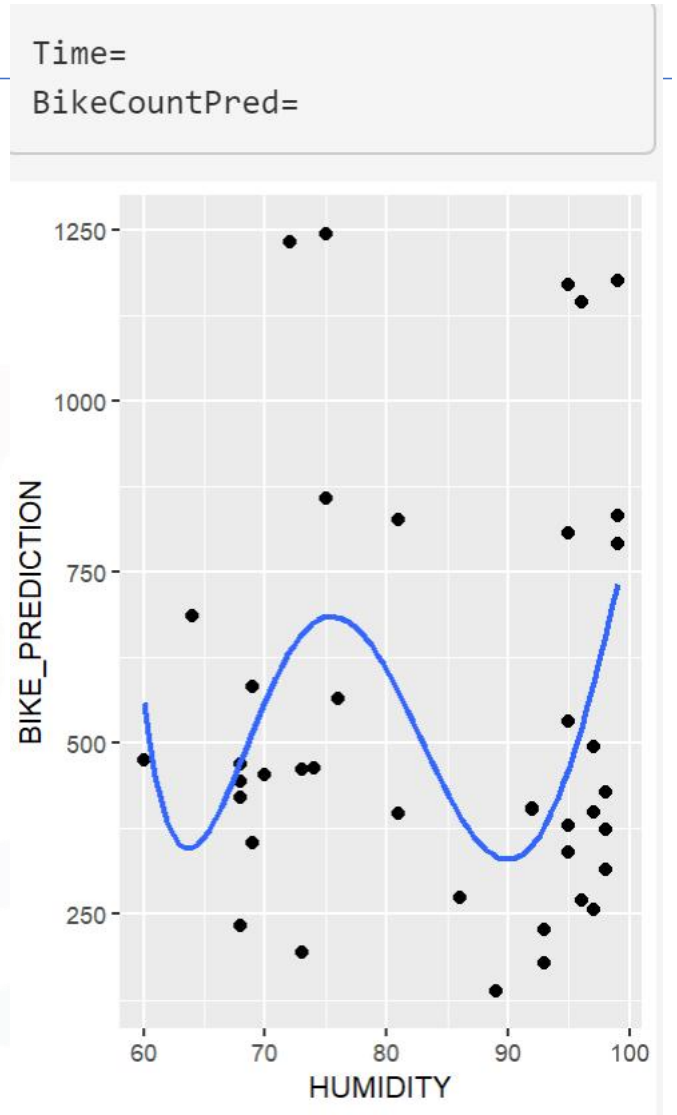
IBM Developer

SKILLS NETWORK

Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for **Suzhou**



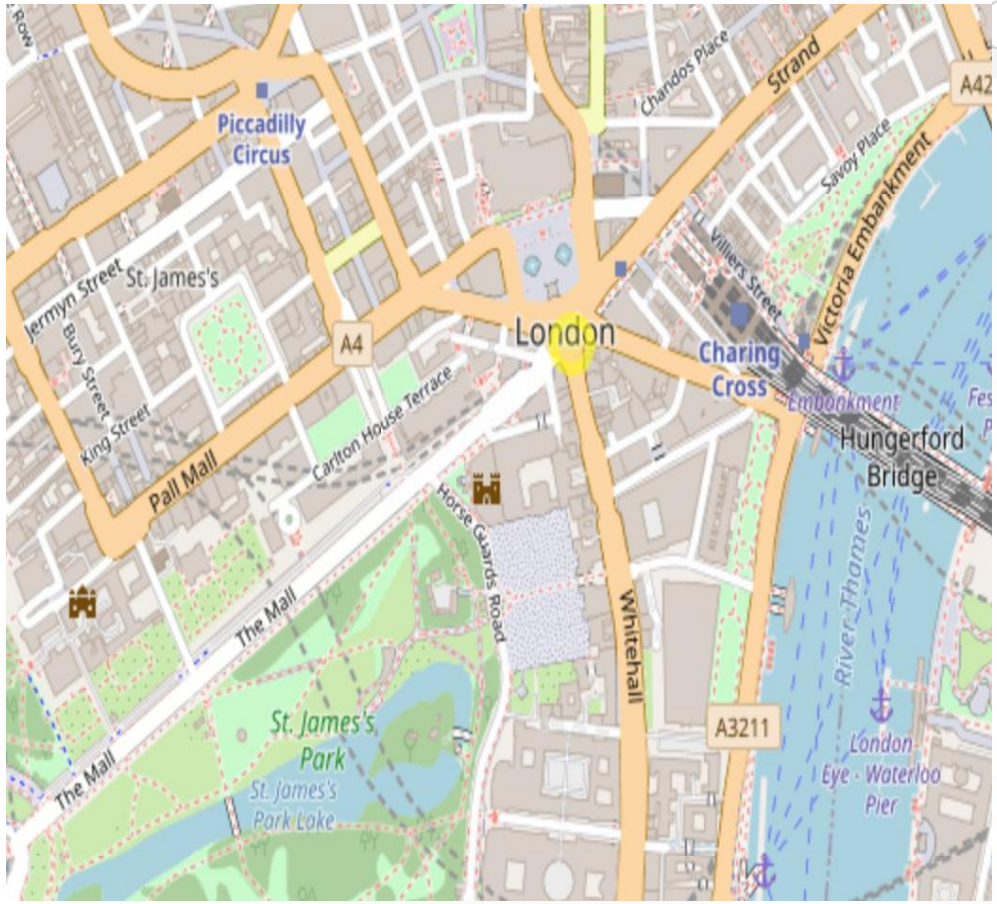


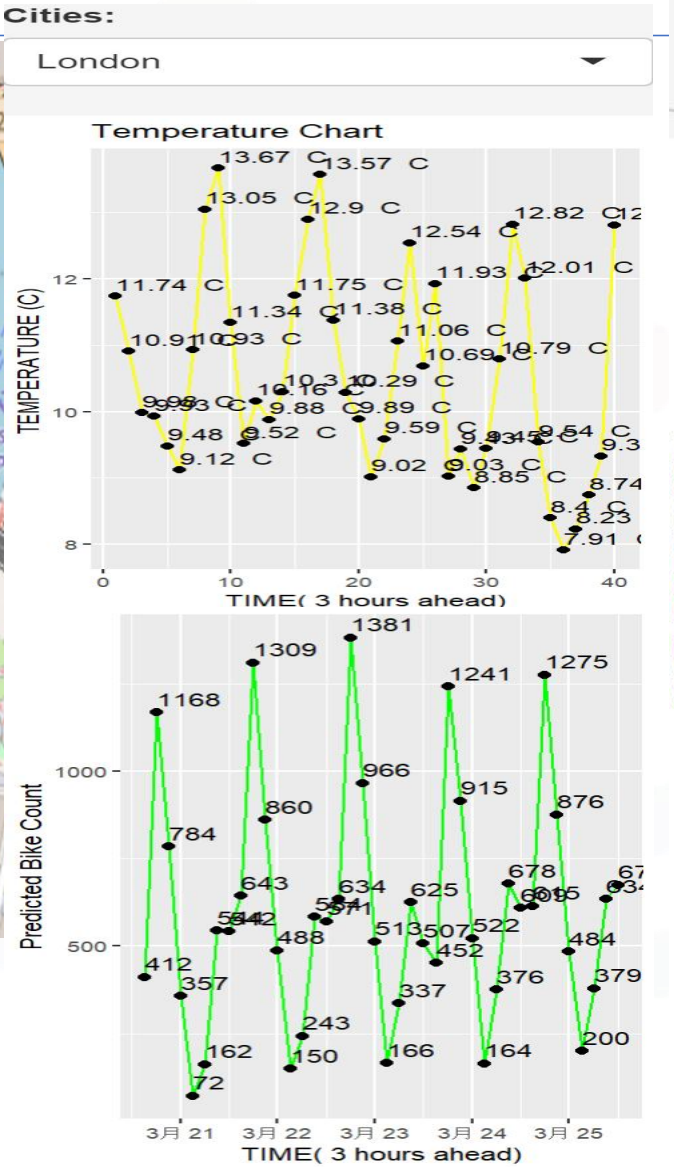
IBM Developer

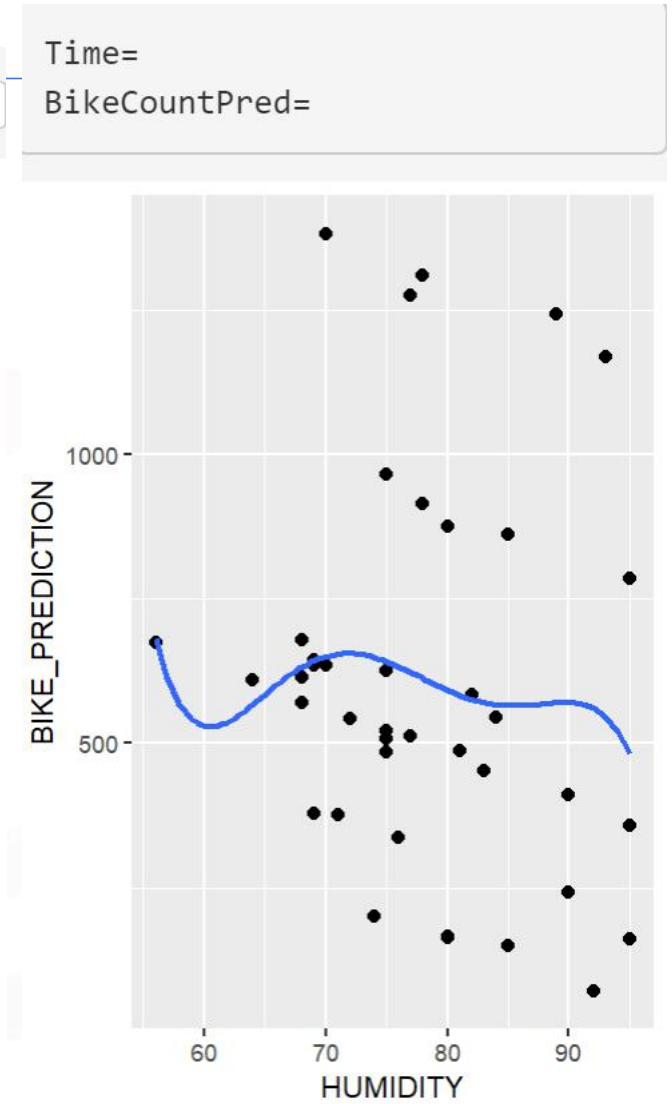
SKILLS NETWORK

Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for **London**





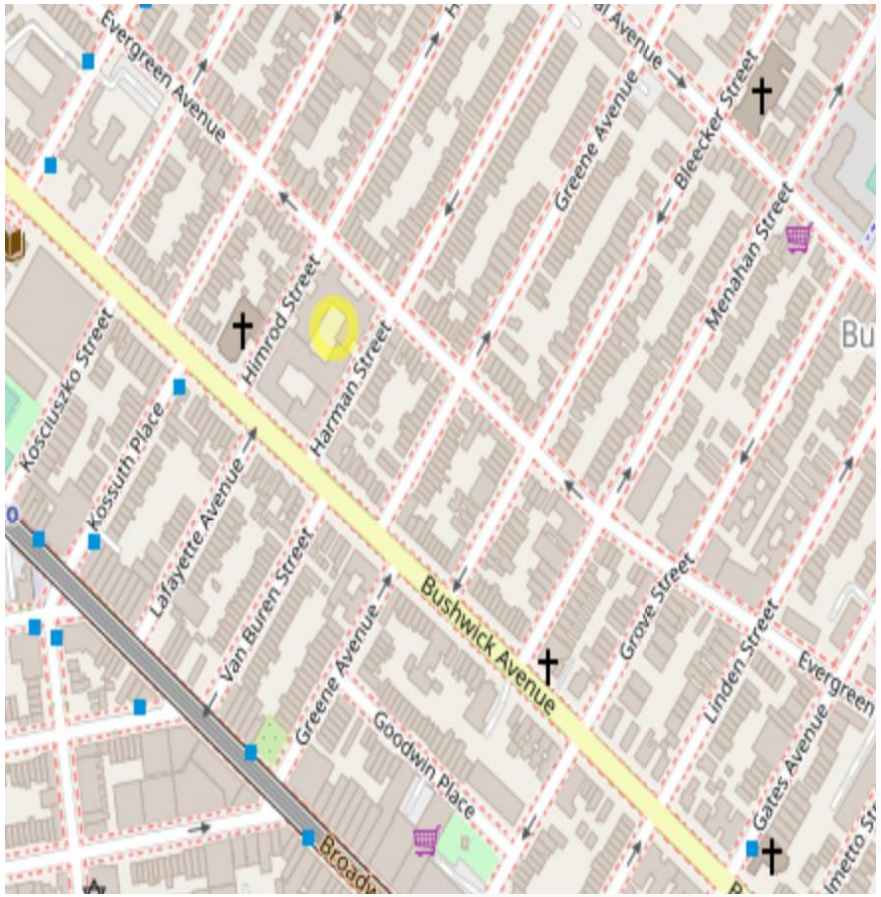


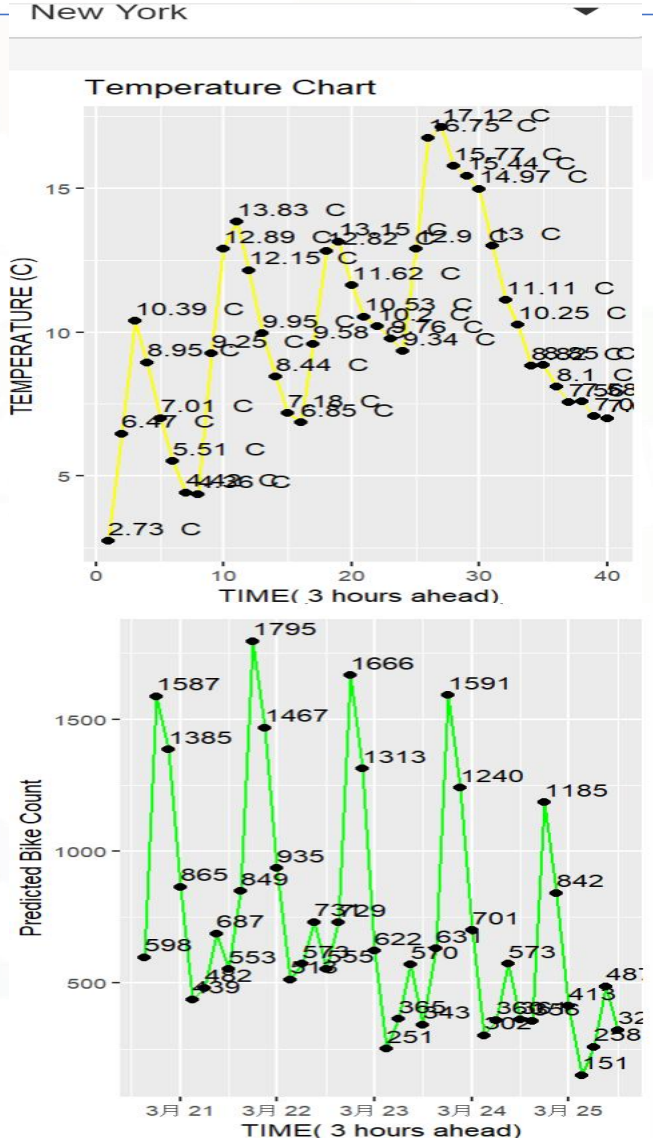
IBM Developer

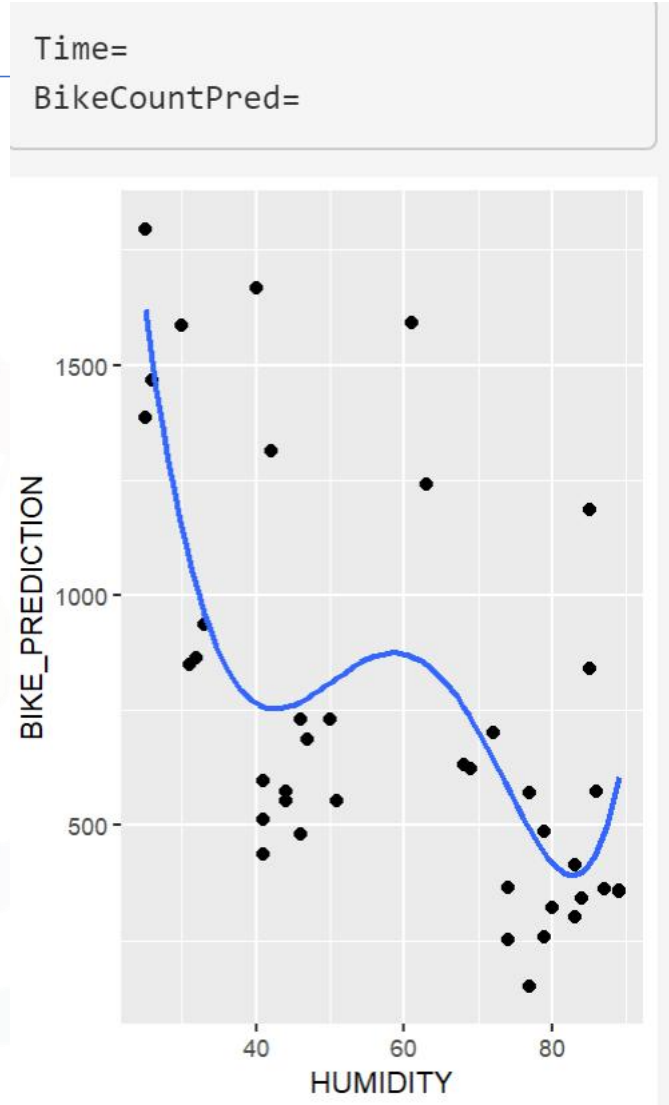
SKILLS NETWORK

Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for **New York**





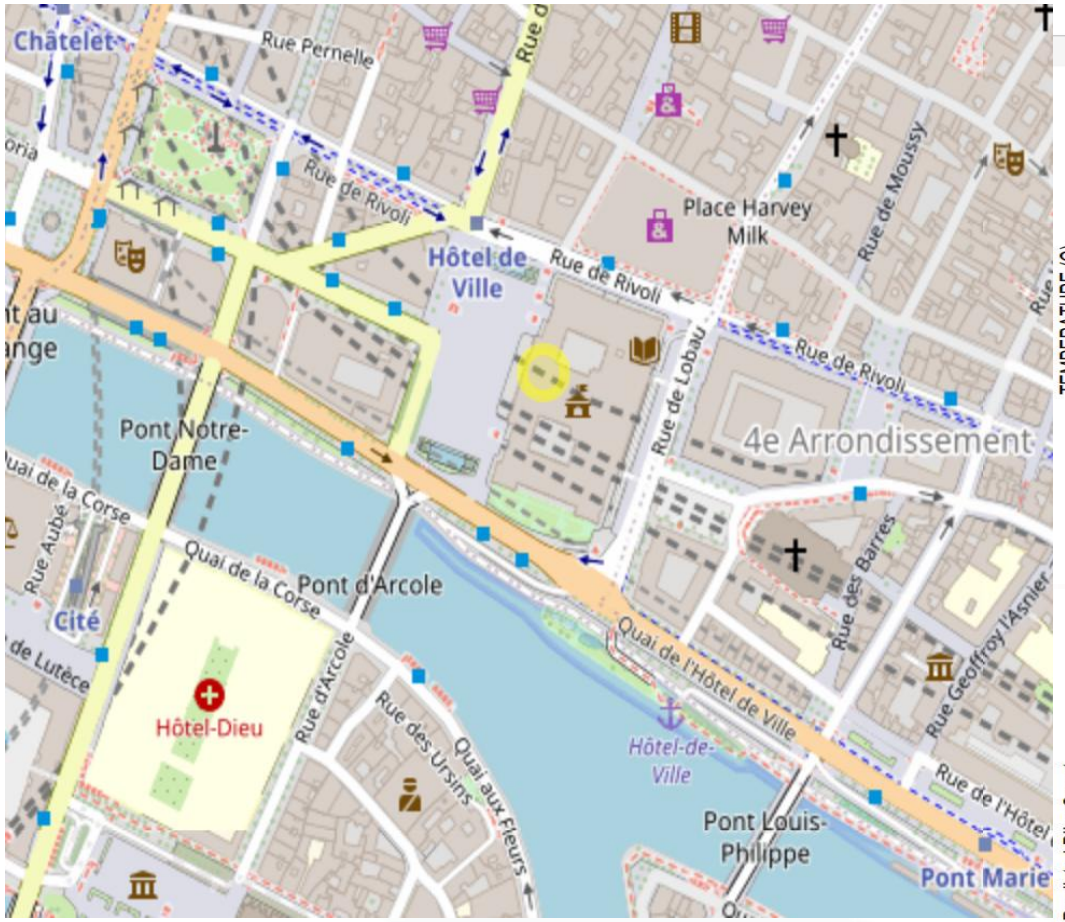


IBM Developer

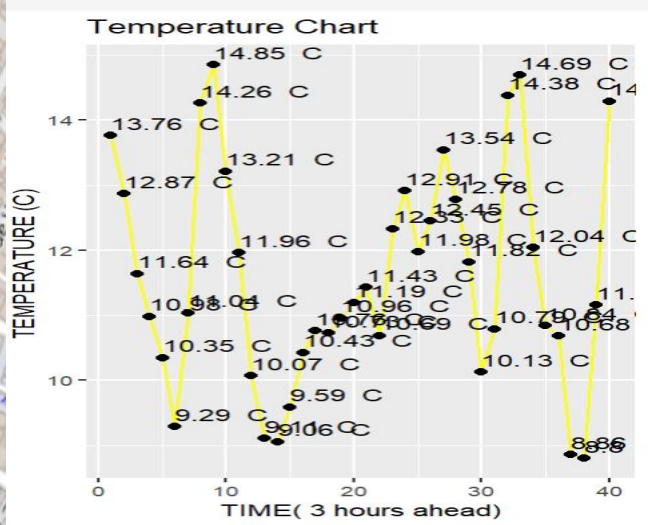
SKILLS NETWORK

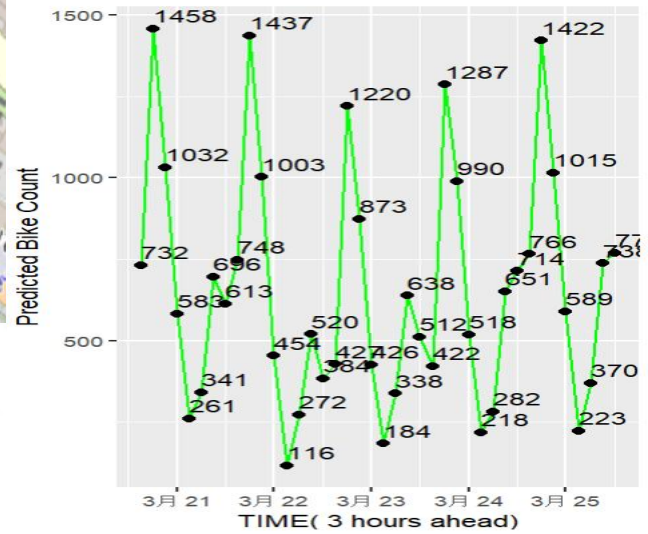
Results: Build a bike-sharing demand prediction app with R Shiny and Leaflet

- Build an interactive R Shiny dashboard to visualize weather forecast data and predicted hourly bike-sharing demand for **Paris**

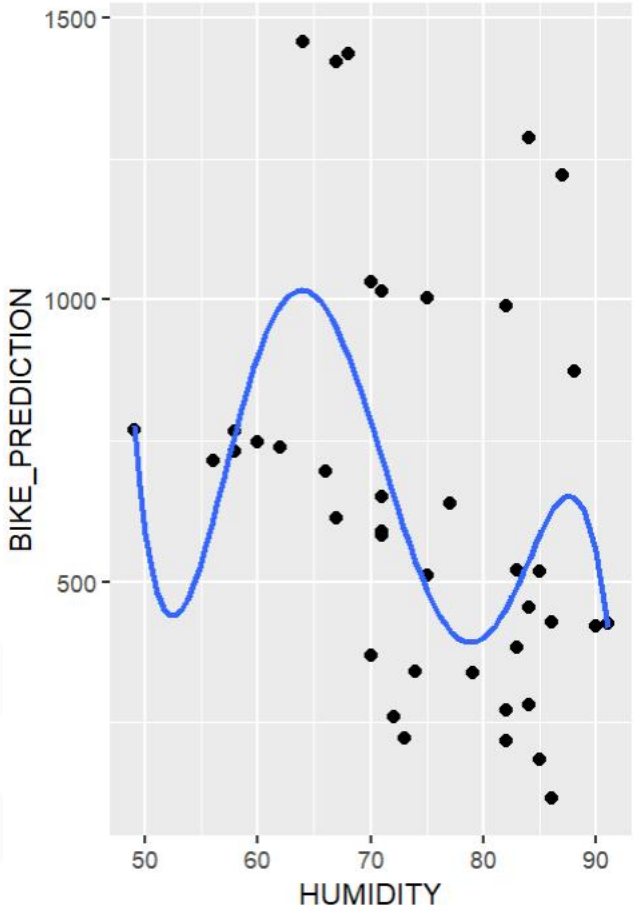


Paris






```
Time=
BikeCountPred=
```



IBM Developer

SKILLS NETWORK

CONCLUSION

- For data collection, fetching bike and weather datasets has been done
- For data wrangling, the missing data has been processed
- Using SQL, Tidyverse & ggplot2 has explored data analysis
- A baseline Regression Model and improvements have solved the demand forecasting problem of shared bicycles,
- For the five selected cities, the temperature, humidity and average time of day all affect the change of the number of rented bicycles.

APPENDIX: Data collection with webscrping

TASK: Extract bike sharing systems HTML table from a Wiki page and convert it into a data frame

TODO: Get the root HTML node

```
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
# Get the root HTML node by calling the `read_html()` method with URL
root_node <- read_html(url)
```

Note that this HTML page at least contains three child `<table>` nodes under the root HTML node. So, you will need to use `html_nodes(root_node, "table")` function to get all its child `<table>` nodes:

```
<html>
  <table>(table1)</table>
  <table>(table2)</table>
  <table>(table3)</table>
  ...
</html>
```

```
table_nodes <- html_nodes(root_node, "table")
```

You can use a `for` loop to print each table, and then you will see that the actual the bike sharing table is the first element `table_nodes[[1]]`.

Next, you need to convert this HTML table into a data frame using the `html_table()` function. You may choose to include `fill = TRUE` argument to fill any empty table rows/columns.

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

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

Summarize the bike sharing system data frame

```
# Summarize the dataframe
summary(data_bike_share)
```

```
   Country              City               Name              System
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Operator           Launched          Discontinued         Stations
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Bicycles           Daily ridership
 Length:549         Length:549
 Class :character   Class :character
 Mode  :character   Mode  :character
```

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

```
# 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:

APPENDIX: Data collection with openAPI

TASK: Get 5-day weather forecasts for a list of cities using the OpenWeather API

Now you should be familiar with the usage of OpenWeather API. Next, you need to complete a task to get 5-day weather forecasts for a list of cities

TODO: Write a function to return a data frame containing 5-day weather forecasts for a list of cities

In [13]:

```
# Create some empty vectors to hold data temporarily

# City name column
city <- c()
# Weather column, rainy or cloudy, etc
weather <- c()
# Sky visibility column
visibility <- c()
# Current temperature column
temp <- c()
# Max temperature column
temp_min <- c()
# Min temperature column
temp_max <- c()
# Pressure column
pressure <- c()
# Humidity column
humidity <- c()
# Wind speed column
wind_speed <- c()
# Wind direction column
wind_deg <- c()
# Forecast timestamp
forecast_datetime <- c()
# Season column
# Note that for season, you can hard code a season value from levels Spring, Summer, Autumn, and Winter based on your current month.
season <- c('summer')
```

```
# 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 = "d49fa8896fd76b755589076219b85fef", units="metric")
        # Make HTTP GET call for the given city
        response <- 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$list
         json_list <- content(response, as="parsed")
         results <- json_list$list
        # Loop the json result
        for(result in results) {
           city <- c(city, city_name)
      weather <- c(weather, result$weather[[1]]$main)
# Get Visibility
visibility <- c(visibility, json_result$visibility)
# Get current temperature
temp <- c(temp, json_result$main$temp)
# Get min temperature
temp_min <- c(temp_min, json_result$main$temp_min)
# Get max temperature
temp_max <- c(temp_max, json_result$main$temp_max)
# Get pressure
pressure <- c(pressure, json_result$main$pressure)
# Get humidity
humidity <- c(humidity, json_result$main$humidity)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind direction
wind_deg <- c(wind_deg, json_result$wind$deg)
forecast_datetime <- c(forecast_datetime, result$dt_txt)forecast_datetime < −c(forecast_datetime, result$dt_txt)

        }
        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)
        # Add the R Lists into a data frame
    }

    # Return a data frame
    return(df)

}
```

IBM Developer

APPENDIX: Data collection with openAPI

Complete and call `get_weather_forecaset_by_cities` function with a list of cities, and write the data frame into a csv file called `cities_weather_forecast.csv`

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

```
# Write cities_weather_df to `cities_weather_forecast.csv`
write.csv(cities_weather_df, "cities_weather_forecast.csv", row.names=FALSE)
```

For more details about HTTP requests with `httr`, please refer to the previous HTTP request notebook here:

HTTP request in R

TASK: Download datasets as csv files from cloud storage

The last task of this lab is straightforward: download some aggregated datasets from cloud storage

```
# 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_worldcitie
# download the file
download.file(url, destfile = "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
# download the file
download.file(url, destfile = "raw_seoul_bike_sharing.csv")
```

Appendix: Data wrangling with regular expression

TASK: Standardize column names for all collected datasets

```
# A tibble: 6 x 10
  COUNTRY CITY  NAME  SYSTEM OPERATOR LAUNCHED DISCONTINUED STATIONS BICYCLES
  <chr>   <chr> <chr> <chr>  <chr>    <chr>    <chr>        <chr>    <chr>
1 Albania Tira… Ecov… <NA>   <NA>     March 2… <NA>         8        200
2 Argent… Mend… Metr… <NA>   <NA>     2014     <NA>         2        40
3 Argent… San … Bici… Biciu… <NA>     27 Nove… <NA>         8        80
4 Argent… Buen… Ecob… Sertt… Bike In… 2010     <NA>         400      4000
5 Argent… Rosa… Mi B… <NA>   <NA>     2 Decem… <NA>         47       480
6 Austra… Melb… Melb… PBSC … Motivate June 20… 30 November… 53       676
# … with 1 more variable: DAILY RIDERSHIP <chr>
# A tibble: 6 x 14
  DATE  RENTED_BIKE_COU…  HOUR TEMPERATURE HUMIDITY WIND_SPEED VISIBILITY
  <chr>             <dbl> <dbl>       <dbl>    <dbl>      <dbl>      <dbl>
1 01/1…               254     0        -5.2       37        2.2       2000
2 01/1…               204     1        -5.5       38        0.8       2000
3 01/1…               173     2        -6         39        1         2000
4 01/1…               107     3        -6.2       40        0.9       2000
5 01/1…                78     4        -6         36        2.3       2000
6 01/1…               100     5        -6.4       37        1.5       2000
# … with 7 more variables: DEW_POINT_TEMPERATURE <dbl>, SOLAR_RADIATION <dbl>,
#   RAINFALL <dbl>, SNOWFALL <dbl>, SEASONS <chr>, HOLIDAY <chr>,
#   FUNCTIONING DAY <chr>
# A tibble: 6 x 12
  CITY  WEATHER VISIBILITY  TEMP TEMP_MIN TEMP_MAX PRESSURE HUMIDITY WIND_SPEED
  <chr> <chr>        <dbl> <dbl>    <dbl>    <dbl>    <dbl>    <dbl>      <dbl>
1 Seoul Clear        10000  12.3     10.9     12.3     1015       50       2.18
2 Seoul Clear        10000  11.5      9.81    11.5     1016       48       1.25
3 Seoul Clouds       10000   9.99     8.82     9.99    1015       46       0.94
4 Seoul Clouds       10000   7.87     7.87     7.87    1014       46       0.83
5 Seoul Clouds       10000  10.1     10.1     10.1     1014       37       1.96
6 Seoul Rain         10000   9.74     9.74     9.74    1014       48       3.24
# … with 3 more variables: WIND_DEG <dbl>, SEASON <chr>,
#   FORECAST DATETIME <dttm>
# A tibble: 6 x 11
  CITY  CITY_ASCII   LAT   LNG COUNTRY ISO2  ISO3  ADMIN_NAME CAPITAL POPULATION
  <chr> <chr>      <dbl> <dbl> <chr>   <chr> <chr> <chr>      <chr>        <dbl>
1 Tokyo Tokyo       35.7  140.  Japan   JP    JPN   Tōkyō      primary   37977000
2 Jaka… Jakarta     -6.21 107.  Indone… ID    IDN   Jakarta    primary   34540000
3 Delhi Delhi       28.7   77.2 India   IN    IND   Delhi      admin     29617000
4 Mumb… Mumbai      19.0   72.8 India   IN    IND   Mahārāsht… admin     23355000
5 Mani… Manila      14.6  121.  Philip… PH    PHL   Manila     primary   23088000
6 Shan… Shanghai    31.2  121.  China   CN    CHN   Shanghai   admin     22120000
# … with 1 more variable: ID <dbl>
```

Process the web-scraped bike sharing system dataset

A tibble: 4 × 2

variable	class
COUNTRY	character
CITY	character
SYSTEM	character
BICYCLES	character

A spec_tbl_df: 10 × 1

BICYCLES
4115[22]
310[59]
500[72]
[75]
180[76]
600[77]
[78]
initially 800 (later 2500)
100 (220)
370[114]

A spec_tbl_df: 10 × 1

CITY
Melbourne[12]
Brisbane[14][15]
Lower Austria[18]
Namur[19]
Brussels[21]
Salvador[23]
Belo Horizonte[24]
João Pessoa[25]
(Pedro de) Toledo[26]
Rio de Janeiro[27]

A spec_tbl_df: 7 × 1

SYSTEM
EasyBike[58]
4 Gen.[61]
3 Gen. SmooveKey[113]
3 Gen. Smoove[141][142][143][139]
3 Gen. Smoove[179]
3 Gen. Smoove[181]
3 Gen. Smoove[183]

TASK: Remove undesired reference links using regular expressions

```
result
```

A spec_tbl_df: 480 × 4

COUNTRY	CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>	<chr>

```
result %>%
    select(CITY, SYSTEM, BICYCLES) %>%
    filter(find_reference_pattern(CITY) | find_reference_pattern(SYSTEM) | find_reference_pattern(BICYCLES))
```

CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>

spec_tbl_df: 25 × 3

TASK: Extract the numeric value using regular expressions

```
summary(result$BICYCLES)
```

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

SKILLS NETWORK

Appendix: Data wrangling with with dplyr

TASK: Detect and handle missing values

```
dim(bike_sharing_df)
```

1. 8465
2. 14

```
# Print the summary of the dataset again to make sure no missing values in all columns
bike_sharing_df %>%
  summarize(count = sum(is.na(TEMPERATURE)))
  head(bike_sharing_df)
  dim(bike_sharing_df)
```

count
<int>
0

A tibble: 1 × 1

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILIT
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl

A tibble: 6 × 14

TASK: Create indicator (dummy) variables for categorical variables

```
# Print the dataset summary again to make sure the indicator columns are created properly
head(seoul_bike_sharing_converted)
```

DATE	RENTED_BIKE_COUNT	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	...	21	
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>	<db

A tibble: 6 ×

TASK: Normalize data

```
head(seoul_bike_sharing_converted_normalized)
```

DATE	RENTED_BIKE_COUNT	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	...	21	
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>	<db
01/12/2017	0.07090602	0.2202797	0.3775510	0.2972973	1	0.2249135	0	0	0	...	0	
01/12/2017	0.05683737	0.2150350	0.3877551	0.1081081	1	0.2249135	0	0	0	...	0	
01/12/2017	0.04811480	0.2062937	0.3979592	0.1351351	1	0.2231834	0	0	0	...	0	
01/12/2017	0.02954418	0.2027972	0.4081633	0.1216216	1	0.2249135	0	0	0	...	0	
01/12/2017	0.02138436	0.2062937	0.3673469	0.3108108	1	0.2076125	0	0	0	...	0	
01/12/2017	0.02757456	0.1993007	0.3775510	0.2027027	1	0.2058824	0	0	0	...	0	

A tibble: 6 × 41

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:

```
# 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)
}
```

IBM Developer

Appendix: Exploratory Data Analysis with SQL

Task 1 - Record Count

Determine how many records are in the seoul_bike_sharing datase

Solution 1

```
# provide your solution here
query = "SELECT count(*) FROM seoul_bike_sharing dataset"
    sqlQuery(conn, query)
```

	1
	<int>
A data.frame: 1 × 1	
1	8465

Task 2 - Operational Hours

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

Solution 2

```
# provide your solution here
query = "SELECT COUNT(HOUR) FROM seoul_bike_sharing dataset
         WHERE HOUR >0"
    sqlQuery(conn, query)
```

	1
	<int>
A data.frame: 1 × 1	
1	8113

Task 4 - Seasons

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

Solution 4

```
# provide your solution here
query = "SELECT DISTINCT(SEASONS) FROM seoul_bike_sharing dataset  "
    sqlQuery(conn, query)
```

	SEASONS
	<fct>
A data.frame: 4 × 1	
1	Autumn
2	Spring
3	Summer

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

```
# provide your solution here
query = "SELECT * FROM CITIES_WEATHER_FORECAST LIMIT 1  "
    sqlQuery(conn, query)
```

	CITY	WEATHER	VISIBILITY	TEMP	TEMP_MIN	TEMP_MAX	PRESSURE	HUMIDITY	WIND_SPEED	WIND_DEG	SEASON	FORECAST_DATETIM
	<fct>	<fct>	<int>	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<int>	<fct>	<dttm
A data.frame: 1 × 12												
1	Seoul	Clear	10000	12.32	10.91	12.32	1015	50	2.18	248	Spring	2021-04-16 12:00:

Task 5 - Date Range

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

Solution 5

```
# provide your solution here
query = "SELECT MAX(DATE) as First_date, MIN(DATE) as Last_date FROM seoul_bike_sharing dataset
        "
    sqlQuery(conn, query)
```

	FIRST_DATE	LAST_DATE
	<fct>	<fct>
A data.frame: 1 × 2		
1	31/12/2017	01/01/2018

Appendix: Exploratory Data Analysis with SQL

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

```
# provide your solution here
query = "SELECT DATE, HOUR, RENTED_BIKE_COUNT FROM seoul_bike_sharing dataset
         ORDER BY RENTED_BIKE_COUNT DESC LIMIT 1
        "
    sqlQuery(conn, query)
```

	DATE	HOUR	RENTED_BIKE_COUNT
	<fct>	<int>	<int>
A data.frame: 1 × 3			
1	19/06/2018	18	3556

	HOUR__WINTER	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	-0.8355556	438
2	8	-5.3266667	422
3	17	0.1400000	342
4	16	0.9822222	308
5	19	-1.5166667	304
6	15	1.1666667	298
7	14	0.8711111	284
8	13	0.2166667	275
9	12	-0.5666667	263
10	9	-4.8022222	254

	HOUR__SUMMER	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	29.38696	2135
2	19	28.27283	1889
3	20	27.06630	1801
4	21	26.27826	1754
5	22	25.69891	1567
6	17	30.07500	1526
7	8	24.53587	1418
8	16	30.48804	1174
9	23	25.23043	1153
10	15	30.49022	1009

Task 7 - Hourly popularity and temperature by season

Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.

Solution 7

```
# provide your solution here
query = "SELECT  HOUR AS HOUR_Autumn, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing data
         WHERE SEASONS=  'Autumn'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10

        "
    sqlQuery(conn, query)


query = "SELECT  HOUR AS HOUR__Spring, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dat
         WHERE SEASONS=  'Spring'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)

query = "SELECT  HOUR AS HOUR__Summer, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dat
         WHERE SEASONS=  'Summer'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)

query = "SELECT  HOUR AS HOUR__Winter, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dat
         WHERE SEASONS=  'Winter'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)
```

	HOUR_AUTUMN	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	16.03086	1983
2	17	17.27778	1562
3	19	15.06049	1515
4	8	10.55679	1345
5	20	14.32469	1321
6	16	18.08025	1293
7	21	13.69012	1253
8	15	18.34321	1156
9	22	13.09012	1130
10	14	18.15185	1058

	HOUR_SPRING	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	15.97222	1689
2	17	16.87889	1307
3	19	14.67667	1247
4	16	17.35889	1101
5	20	13.57444	1060
6	8	10.30778	1036
7	21	12.83778	1021
8	15	17.51111	990
9	14	17.36000	925
10	22	12.25222	897

Appendix: Exploratory Data Analysis with SQL

```
query = "SELECT  HOUR AS HOUR__Summer, AVG(TEMPERATURE) AS AVG_TEMPERATURE, AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT FROM seoul_bike_sharing dat
         WHERE SEASONS=  'Summer'
         GROUP BY HOUR
         ORDER BY AVG_RENTED_BIKE_COUNT DESC LIMIT 10
        "
    sqlQuery(conn, query)
```

	HOUR_SUMMER	AVG_TEMPERATURE	AVG_RENTED_BIKE_COUNT
	<int>	<dbl>	<int>
A data.frame: 10 × 3			
1	18	29.38696	2135
2	19	28.27283	1889
3	20	27.06630	1801
4	21	26.27826	1754
5	22	25.69891	1567
6	17	30.07500	1526
7	8	24.53587	1418
8	16	30.48804	1174
9	23	25.23043	1153
10	15	30.49022	1009

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
query = "SELECT    SEASONS,   AVG (RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT ,
                   MAX(RENTED_BIKE_COUNT) AS Max_Rented_BIKE,
                   MIN(RENTED_BIKE_COUNT) AS MIN_Rented_BIKE,
                   STDDEV(RENTED_BIKE_COUNT) AS standard_Deviation

                   FROM seoul_bike_sharing dataset
                   GROUP BY SEASONS

        "
    sqlQuery(conn, query)
```

	SEASONS	AVG_RENTED_BIKE_COUNT	MAX_RENTED_BIKE	MIN_RENTED_BIKE	STANDARD_DEVIATION
	<fct>	<int>	<int>	<int>	<dbl>
A data.frame: 4 × 5					
1	Autumn	924	3298	2	617.3885
2	Spring	746	3251	2	618.5247
3	Summer	1034	3556	9	690.0884
4	Winter	225	937	3	150.3374

Appendix: Exploratory

Task 9 - Weather Seas

Consider the weather over each
DEW_POINT_TEMPERATURE, SC

Include the average bike count as well

Solution 9

```
query = "SELECT  SEASONS,
                 AVG(RENTED_BIKE_COUNT) AS AVG_RENTED_BIKE_COUNT,
                 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 dataset
         GROUP BY SEASONS

         "
    sqlQuery(conn, query)
```

	SEASONS	AVG_RENTED_BIKE_COUNT	AVG_TEMPERATURE	AVG_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT_TEMPERATURE	AVG_SOLAR_RADIATION
	<fct>	<int>	<dbl>	<int>	<dbl>	<int>	<dbl>	<dbl>
A data.frame: 4 × 10								
1	Autumn	924	13.821167	59	1.492101	1558	5.150594	0.5227827
2	Spring	746	13.021389	58	1.857778	1240	4.091389	0.6803009
3	Summer	1034	26.587274	64	1.609420	1501	18.750136	0.7612545
4	Winter	225	-2.540463	49	1.922685	1445	-12.416667	0.2981806

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, POPULATION, in a single view.

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
query = "SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
         WHERE B.CITY=W.CITY_ASCII AND B.CITY='Seoul'
"
    sqlQuery(conn, query)
```

	CITY	COUNTRY	LAT	LNG	POPULATION	BICYCLES
	<fct>	<fct>	<dbl>	<dbl>	<int>	<int>
A data.frame: 1 × 6						
1	Seoul	Korea, South	37.58	127	21794000	20000

Appendix: Exploratory Data Analysis with SQL

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, POPULATION, in a single view.

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
query = "SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
         WHERE B.CITY=W.CITY_ASCII AND B.CITY='Seoul'
"
    sqlQuery(conn, query)
```

	CITY	COUNTRY	LAT	LNG	POPULATION	BICYCLES
	<fct>	<fct>	<dbl>	<dbl>	<int>	<int>
A data.frame: 1 × 6						
1	Seoul	Korea, South	37.58	127	21794000	20000

Appendix: Exploratory Data Analysis with SQL

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
query = " WITH comparable_cities AS
         (
         SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES AS BICYCLES
         FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
         WHERE   B.CITY=W.CITY_ASCII AND BICYCLES != 'NA'
        )

       SELECT *
       FROM comparable_cities
       WHERE BICYCLES BETWEEN 15000 AND 20000
       ORDER BY BICYCLES DESC

"
    sqlQuery(conn, query)
```

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

In [18]:

```
query = " WITH comparable_cities AS
         (
         SELECT  B.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,  B.BICYCLES AS BICYCLES
         FROM BIKE_SHARING_SYSTEMS B,  WORLD_CITIES W
         WHERE   B.CITY=W.CITY_ASCII AND BICYCLES != 'NA'
        )

       SELECT CITY,  SUM(BICYCLES)  AS BICYCLES
       FROM comparable_cities
       GROUP BY CITY
       HAVING  SUM(BICYCLES) BETWEEN 15000 AND 20000
       ORDER BY BICYCLES DESC

"
    sqlQuery(conn, query)
```

	CITY	BICYCLES
	<fct>	<int>
A data.frame: 7 × 2		
1	Seoul	20000
2	Weifang	20000
3	Xi'an	20000
4	Zhuzhou	20000
5	Shanghai	19165
6	Beijing	16000
7	Ningbo	15000

In [19]:

```
close(conn)
```

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 1 - Load the dataset

Ensure you read `DATE` as type `character`.

Solution 1

```
# provide your solution here
seoul_bike_sharing <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/labs/datasets
```

```
library(tidyverse)
```

```
── 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()
```

```
seoul_bike_sharing<-read_csv(seoul_bike_sharing)
```

```
Parsed with column specification:
cols(
  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()
)
```

```
head(seoul_bike_sharing)
```

A tibble: 6 × 14

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter
01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter
01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter
01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter
01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter
01/12/2017	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter

IBM Developer

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 2 - Recast `DATE` as a date

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

Solution 2

In [5]:
```
# provide your solution here
seoul_bike_sharing$DATE<-as.Date(seoul_bike_sharing$DATE, format="%d/%m/%Y")
```

In [6]:
```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HC
<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
A tibble: 6 × 14												
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

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

```
# provide your solution here
seoul_bike_sharing$HOUR <- factor(seoul_bike_sharing$HOUR, ordered=TRUE)
```

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HO
<date>	<dbl>	<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
A tibble: 6 × 14												
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	

Check the structure of the dataframe

```
str(seoul_bike_sharing)
```

```
tibble [8,465 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ DATE                 : Date[1:8465], format: "2017-12-01" "2017-12-01" ...
 $ RENTED_BIKE_COUNT    : num [1:8465] 254 204 173 107 78 100 181 460 930 490 ...
 $ HOUR                 : Ord.factor w/ 24 levels "0"<"1"<"2"<"3"<..: 1 2 3 4 5 6 7 8 9 10 ...
 $ TEMPERATURE          : num [1:8465] -5.2 -5.5 -6 -6.2 -6 -6.4 -6.6 -7.4 -7.6 -6.5 ...
 $ HUMIDITY             : num [1:8465] 37 38 39 40 36 37 35 38 37 27 ...
 $ WIND_SPEED           : num [1:8465] 2.2 0.8 1 0.9 2.3 1.5 1.3 0.9 1.1 0.5 ...
 $ VISIBILITY           : num [1:8465] 2000 2000 2000 2000 2000 ...
 $ DEW_POINT_TEMPERATURE: num [1:8465] -17.6 -17.6 -17.7 -17.6 -18.6 -18.7 -19.5 -19.3 -19.8 -22.4 ...
 $ SOLAR_RADIATION      : num [1:8465] 0 0 0 0 0 0 0 0 0.01 0.23 ...
 $ RAINFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SNOWFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SEASONS              : chr [1:8465] "Winter" "Winter" "Winter" "Winter" ...
 $ HOLIDAY              : chr [1:8465] "No Holiday" "No Holiday" "No Holiday" "No Holiday" ...
 $ FUNCTIONING_DAY      : chr [1:8465] "Yes" "Yes" "Yes" "Yes" ...
 - attr(*, "spec")=
  .. cols(
  ..   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()
  .. )
```

Finally, ensure there are no missing values

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

```
0
```

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Descriptive Statistics

Now you are all set to take a look at some high level statistics of the `seoul_bike_sharing` dataset.

Task 4 - Dataset Summary

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

Solution 4

```
# provide your solution here
summary(seoul_bike_sharing)
```

```
     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   Length:8465
1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Class :character
Median :0.0100   Median : 0.0000   Median :0.00000   Mode  :character
Mean   :0.5679   Mean   : 0.1491   Mean   :0.07769
3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000
Max.   :3.5200   Max.   :35.0000   Max.   :8.80000

   HOLIDAY          FUNCTIONING_DAY
Length:8465        Length:8465
Class :character   Class :character
Mode  :character   Mode  :character
```

Task 5 - Based on the above stats, calculate how many Holidays there are.

Solution 5:

```
# provide your solution here
count(filter(seoul_bike_sharing['HOLIDAY'], HOLIDAY=="Holiday" ))/24
```

n
<dbl>
A data.frame: 1 × 1
17

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

Solution 6

```
# provide your solution here
(count(filter(seoul_bike_sharing['HOLIDAY'], HOLIDAY=="Holiday" ))/count(seoul_bike_sharing))*100
```

n
<dbl>
A data.frame: 1 × 1
4.819846

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

Solution 7

```
# provide your solution here
365*24
```

8760

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 8 - Given the observations for the 'FUNCTIONING_DAY' how many records must there be?

Solution 8

In [15]:
```
# provide your solution here
count(seoul_bike_sharing['FUNCTIONING_DAY'])
```

n
<int>
8465

A tibble: 1 × 1

In [16]:
```
distinct(seoul_bike_sharing['FUNCTIONING_DAY'])
```

FUNCTIONING_DAY
<chr>
Yes

A tibble: 1 × 1

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

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

```
# provide your solution here
library(dplyr)
```

```
head(seoul_bike_sharing)
```

DATE	RENTED_BIKE_COUNT	HOUR	TEMPERATURE	HUMIDITY	WIND_SPEED	VISIBILITY	DEW_POINT_TEMPERATURE	SOLAR_RADIATION	RAINFALL	SNOWFALL	SEASONS	HC
<date>	<dbl>	<ord>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	
A tibble: 6 × 14												
2017-12-01	254	0	-5.2	37	2.2	2000	-17.6	0	0	0	Winter	
2017-12-01	204	1	-5.5	38	0.8	2000	-17.6	0	0	0	Winter	
2017-12-01	173	2	-6.0	39	1.0	2000	-17.7	0	0	0	Winter	
2017-12-01	107	3	-6.2	40	0.9	2000	-17.6	0	0	0	Winter	
2017-12-01	78	4	-6.0	36	2.3	2000	-18.6	0	0	0	Winter	
2017-12-01	100	5	-6.4	37	1.5	2000	-18.7	0	0	0	Winter	

```
seoul_bike_sharing %>% group_by (SEASONS) %>% summarize(RAINFALL = sum(RAINFALL), SNOWFALL = sum(SNOWFALL))
```

SEASONS	RAINFALL	SNOWFALL
<chr>	<dbl>	<dbl>
A tibble: 4 × 3		
Autumn	227.9	123.0
Spring	403.8	0.0
Summer	559.7	0.0
Winter	70.9	534.6

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Load the ggplot2 package so we can generate some data visualization

```
# provide your solution here
library(ggplot2)
```

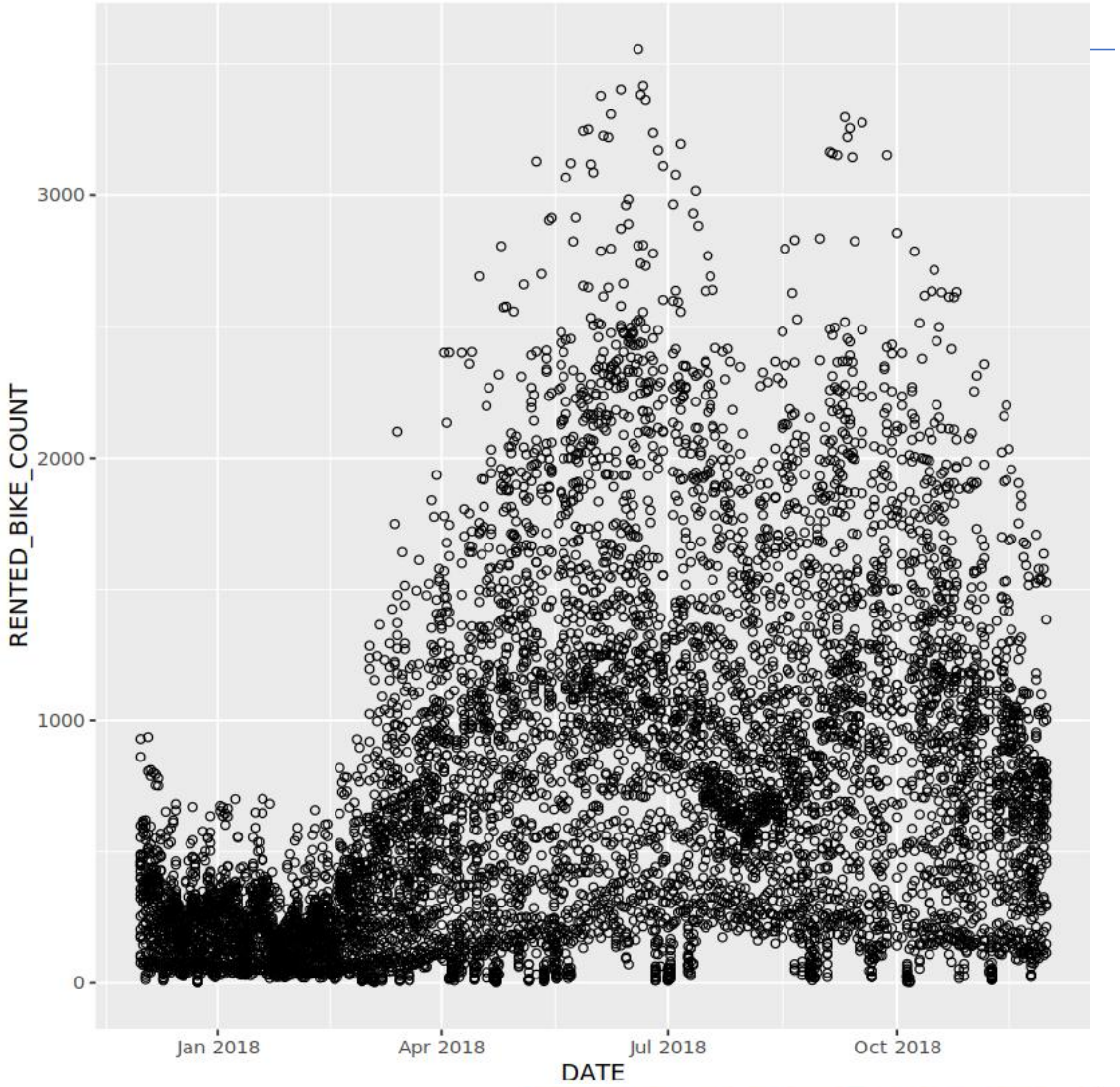
Our variable of interest is a time series, so why not start by taking a look at it in it's natural form?

Task 10 - Create a scatter plot of `RENTED_BIKE_COUNT` vs `DATE`.

Tune the opacity using the `alpha` parameter such that the points don't obscure each other too much.

Solution 10

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT), alpha=0.5) +
  geom_point(shape=1)
```

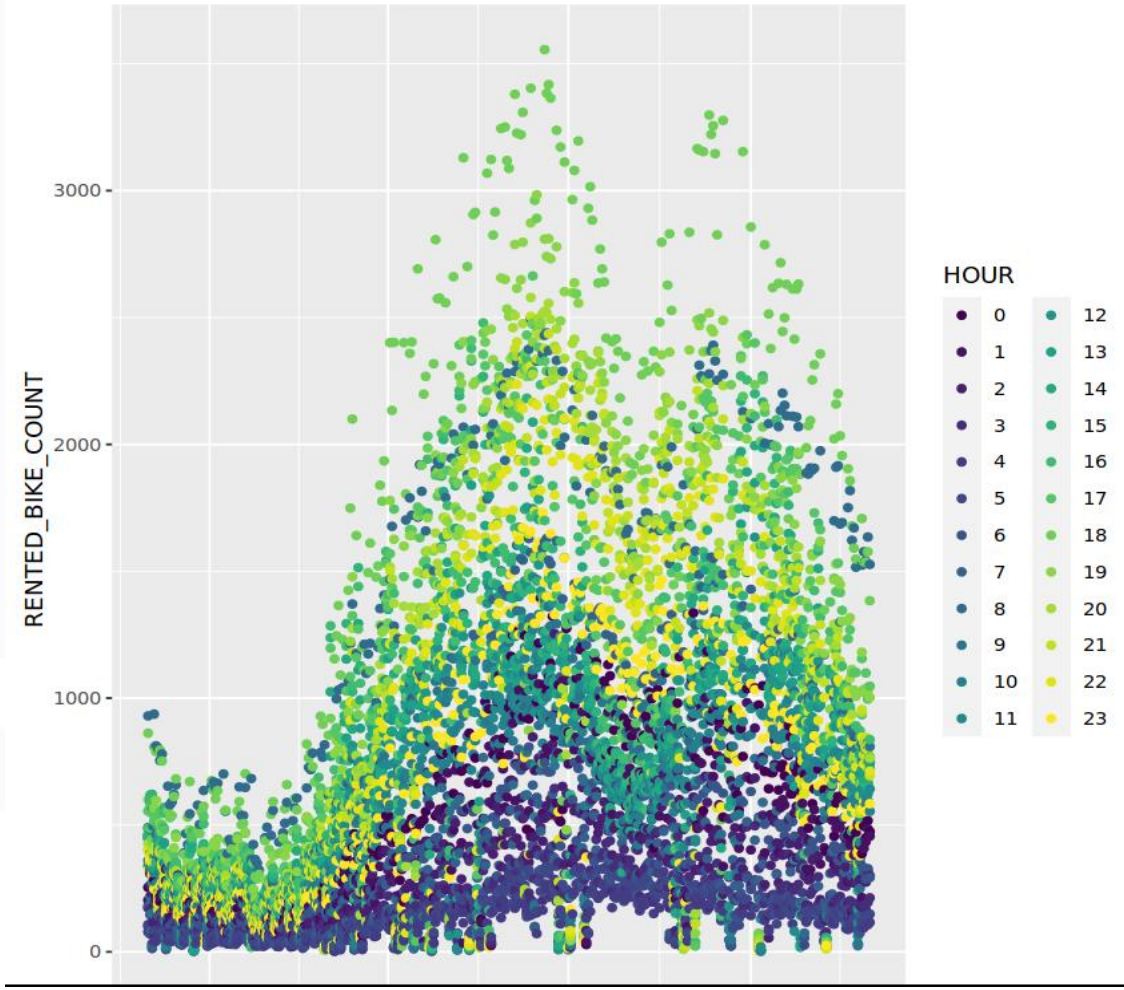


Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 11 - Create the same plot of the `RENTED_BIKE_COUNT` time series, but now add `HOURS` as the colour.

Solution 11

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT), alpha=0.5) +
geom_point(aes(color=HOUR))
```



Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 12 - Create a histogram overlaid with a kernel density curve

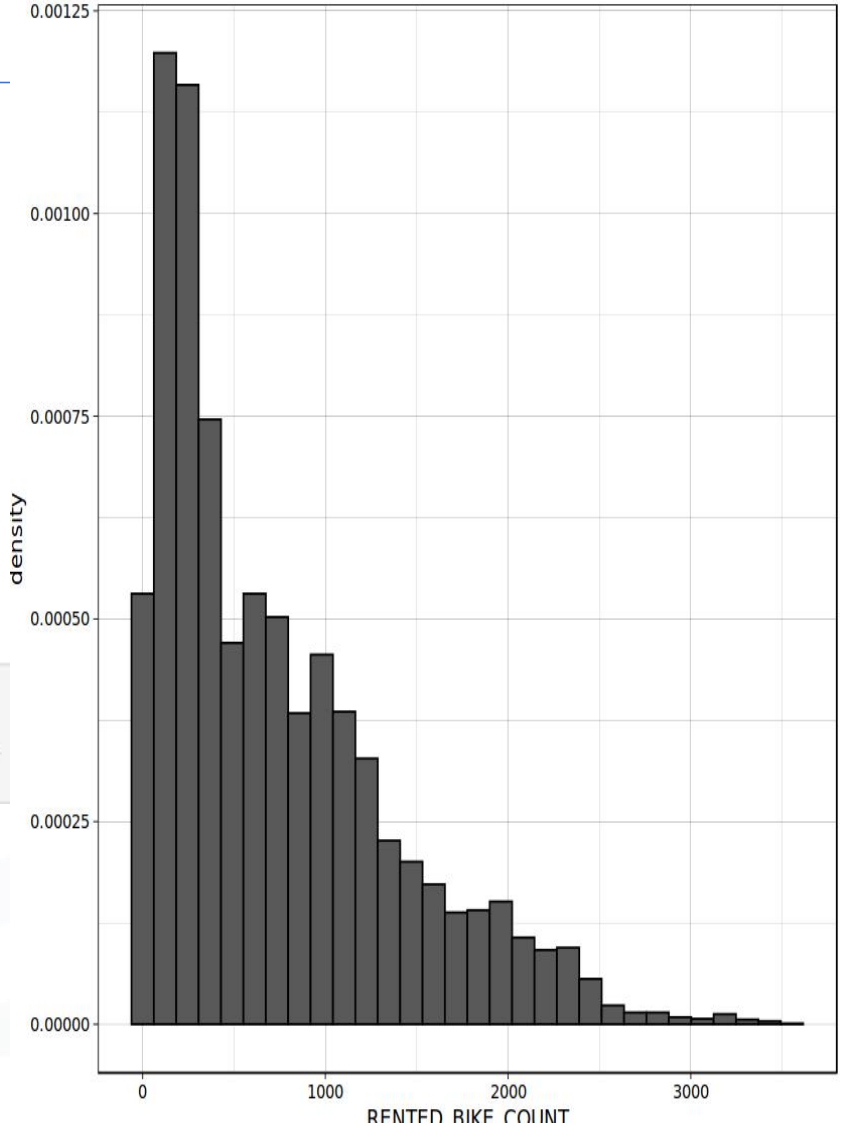
Normalize the histogram so the y axis represents 'density'. This can be done by setting `y=..density..` in the aesthetics of the histogram.

Click here for a hint Set the colour to something like black and the fill to white so you can see the kernel density plot layer better.

Click here for another hint Set the color and alpha such that your denstiy plot is clearly visible, without obscuring the histogram.

Solution 12

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x=RENTED_BIKE_COUNT ), alpha=0.5 ) + geom_histogram(bins=30, col="black",  aes(y=..density..)) +theme_linedraw()
```



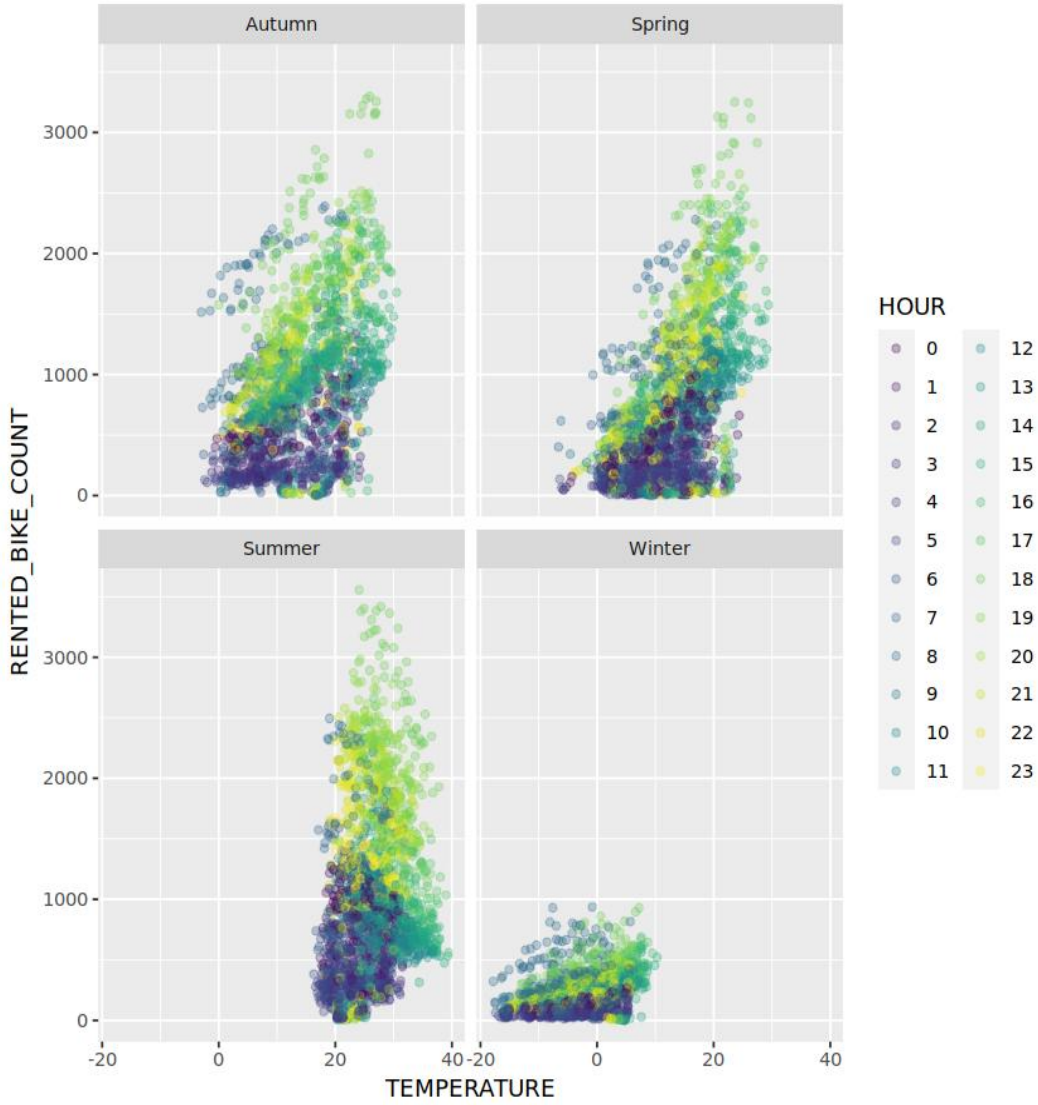
Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 13 - Use a scatter plot to visualize the correlation between `RENTED_BIKE_COUNT` and `TEMPERATURE` by `SEASONS`.

Start with `RENTED_BIKE_COUNT` vs. `TEMPERATURE`, then generate four plots corresponding to the `SEASONS` by adding a `facet_wrap()` layer. Also, make use of colour and opacity to emphasize any patterns that emerge. Use `HOUR` as the color.

Solution 13

```
# provide your solution here
ggplot(seoul_bike_sharing) +
    geom_point(aes(x=TEMPERATURE, y=RENTED_BIKE_COUNT, colour=HOUR), alpha=0.3) +facet_wrap(.~SEASONS)
```

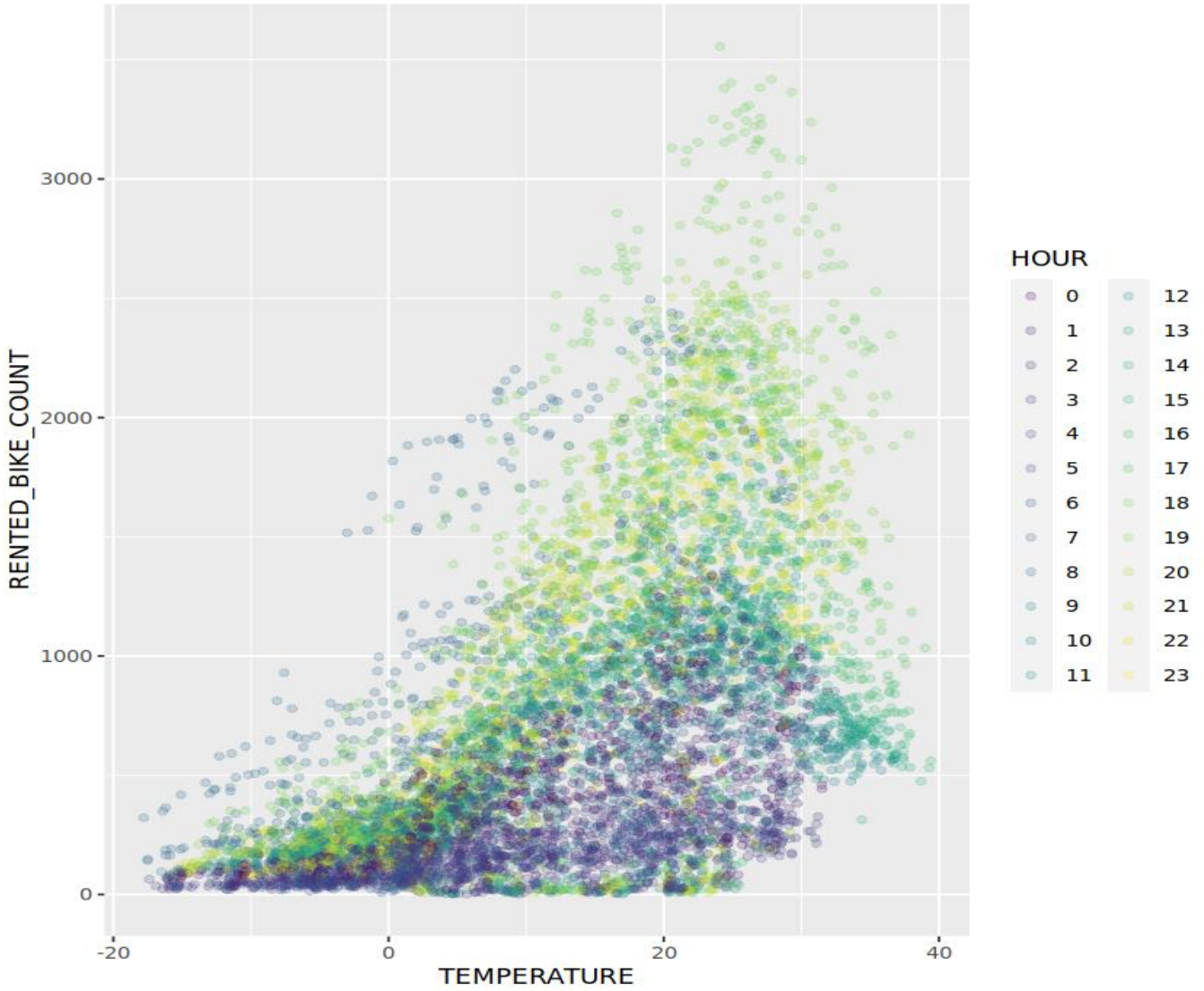


IBM Developer

SKILLS NETWORK

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
ggplot(seoul_bike_sharing) +
   geom_point(aes(x=TEMPERATURE, y=RENTED_BIKE_COUNT, colour=HOUR), alpha=1/5)
```



SKILLS NETWORK

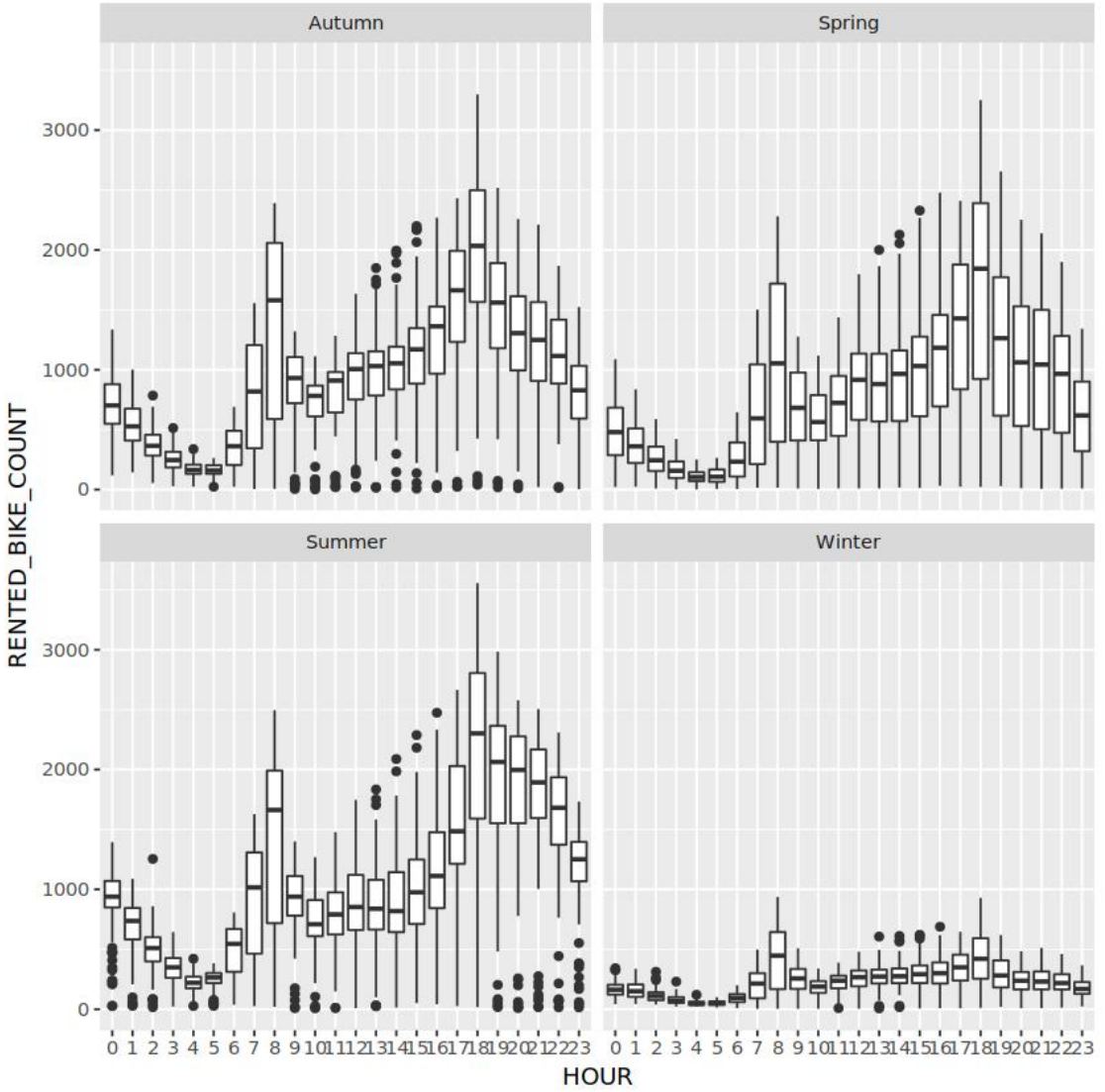
Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 14 - Create a display of four boxplots of `RENTED_BIKE_COUNT` vs. `HOUR` grouped by `SEASONS`

Use `facet_wrap` to generate four plots corresponding to the seasons.

Solution 14

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x=HOUR,y=RENTED_BIKE_COUNT)) +
    geom_boxplot() +facet_wrap(.~SEASONS)
```



Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 15 - Group the data by `DATE`, and use the summarize() function to calculate the daily total rainfall and snowfall.

Also, go ahead and plot the results if you wish.

Solution 15

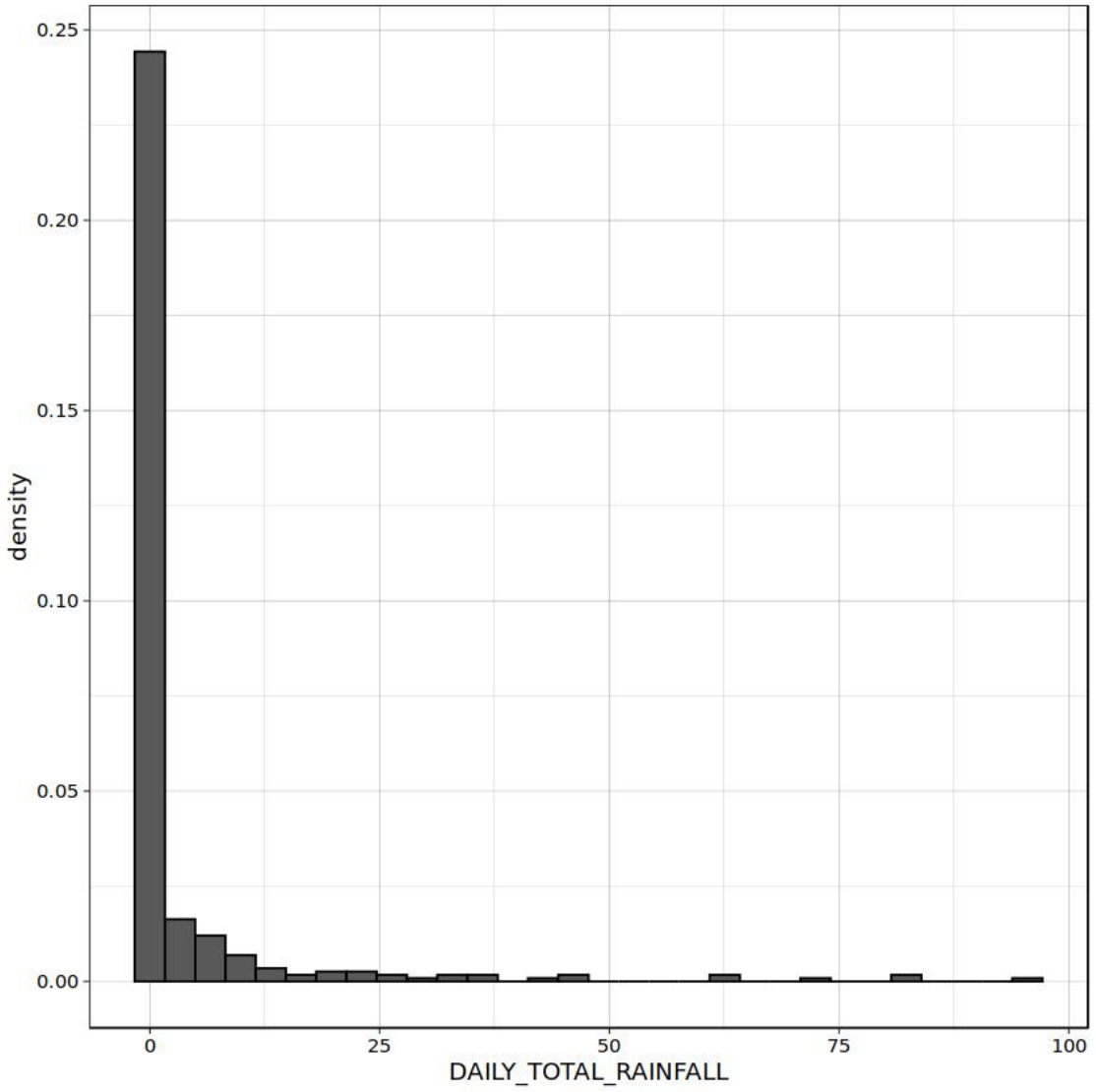
```
# provide your solution here
seoul_daily_rainfall_snowfall<-seoul_bike_sharing %>% group_by (DATE) %>% summarize(DAILY_TOTAL_RAINFALL = sum(RAINFALL), DAILY_TOTAL_SNOWFALL = sum(SNO
seoul_daily_rainfall_snowfall
```

A tibble: 353 × 3

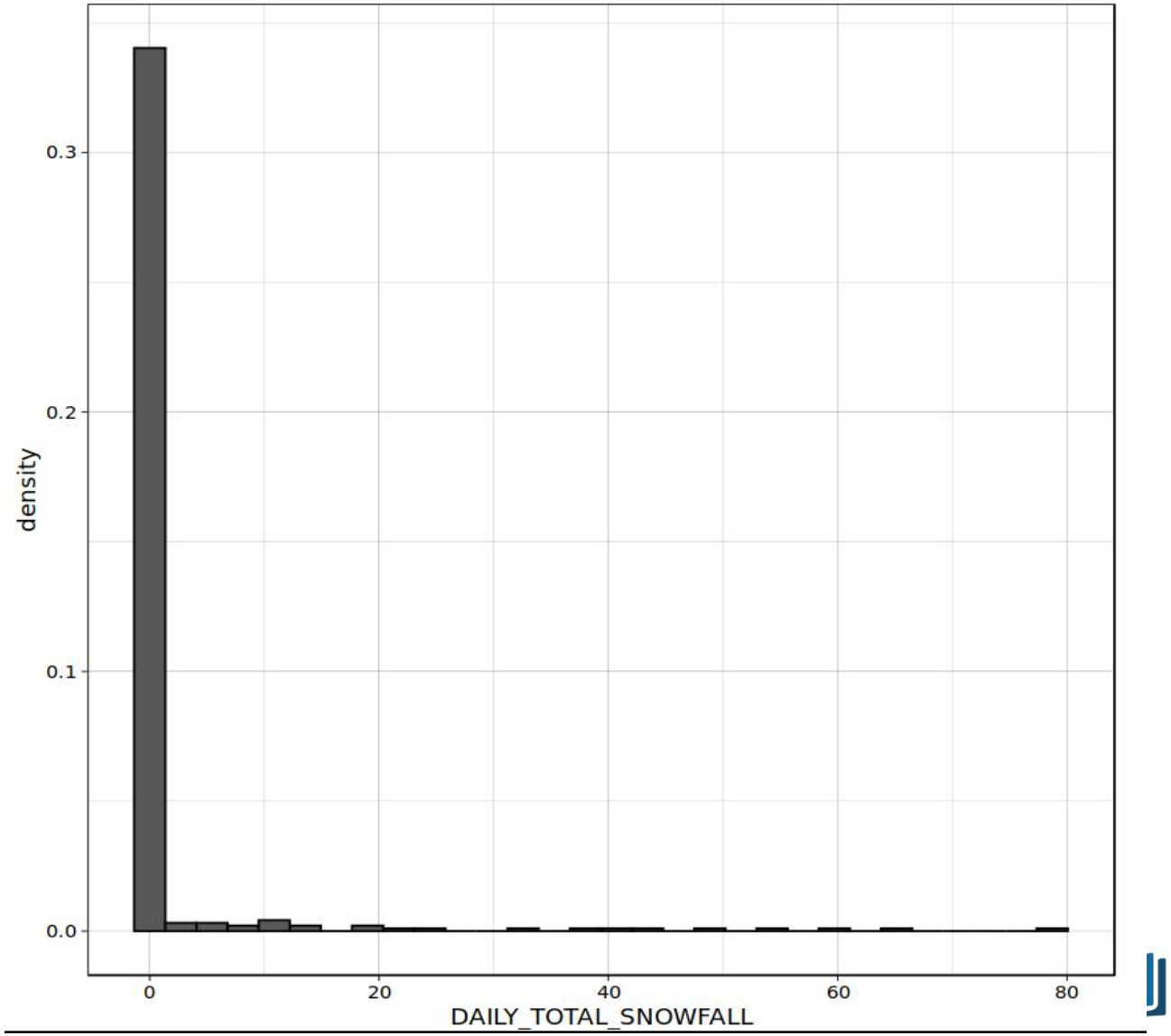
DATE	DAILY_TOTAL_RAINFALL	DAILY_TOTAL_SNOWFALL
<date>	<dbl>	<dbl>
2017-12-01	0.0	0.0
2017-12-02	0.0	0.0
2017-12-03	4.0	0.0
2017-12-04	0.1	0.0
2017-12-05	0.0	0.0
2017-12-06	1.3	8.6
2017-12-07	0.0	10.4
2017-12-08	0.0	0.0
2017-12-09	0.0	0.0
2017-12-10	4.1	32.5
2017-12-11	0.0	0.0
2017-12-12	0.0	0.0
2017-12-13	0.0	0.0
2017-12-14	0.0	0.0
2017-12-15	0.0	0.0
2017-12-16	0.0	0.0
2017-12-17	0.0	0.0
2017-12-18	3.4	59.7
2017-12-19	0.0	55.6
2017-12-20	0.2	48.3
2017-12-21	0.0	38.9
2017-12-22	0.0	7.7
2017-12-23	0.0	0.0
2017-12-24	20.0	0.0
2017-12-25	0.0	0.0
2017-12-26	0.0	0.0
2017-12-27	0.0	0.0
2017-12-28	0.0	0.0

Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DAILY_TOTAL_RAINFALL ), alpha=0.5 )  + geom_histogram(bins=30, col="black",
        aes(y=..density..)) +theme_linedraw()
```

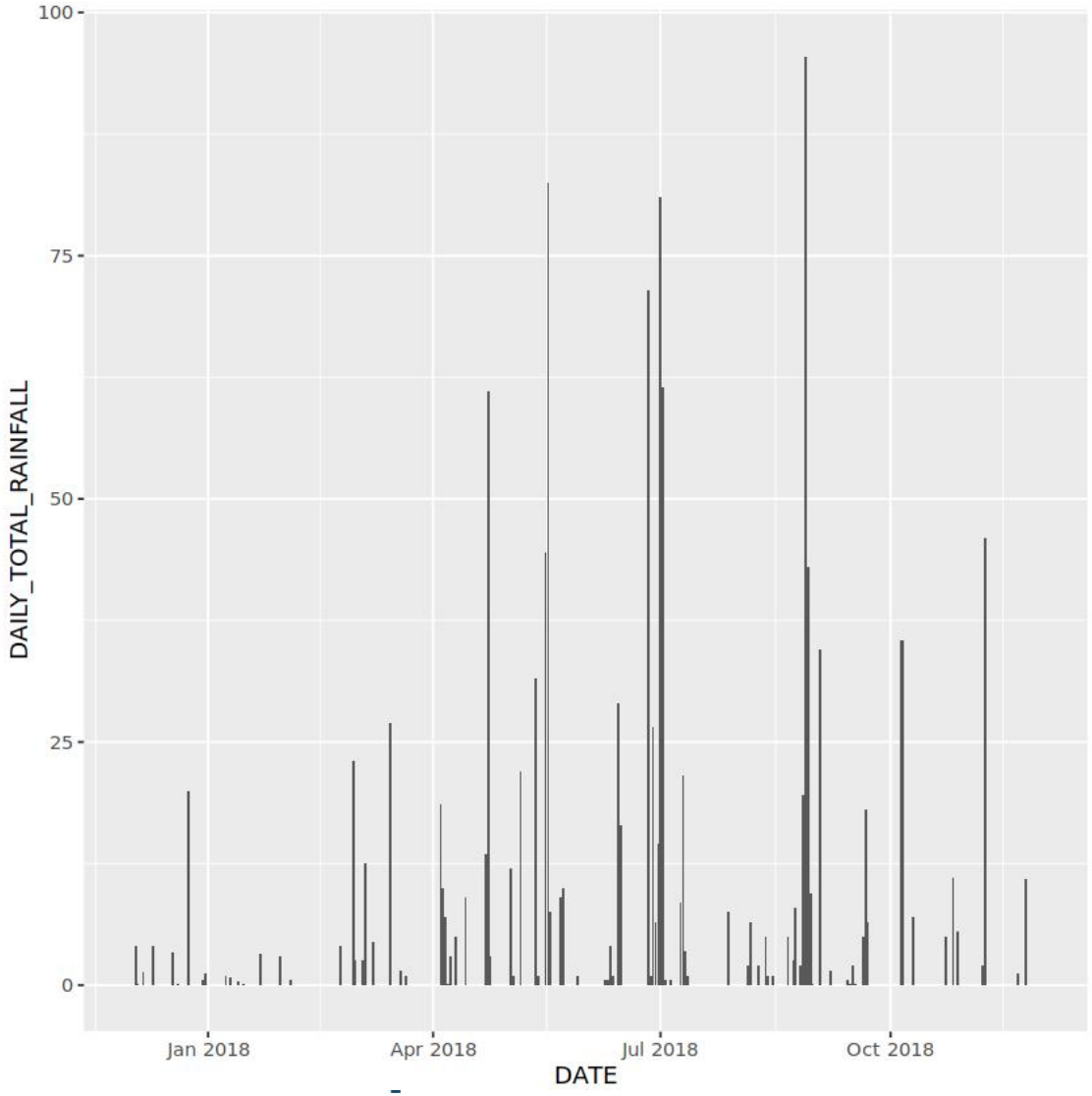



```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DAILY_TOTAL_SNOWFALL ), alpha=0.5 )  + geom_histogram(bins=30, col="black",
        aes(y=..density..)) +theme_linedraw()
```

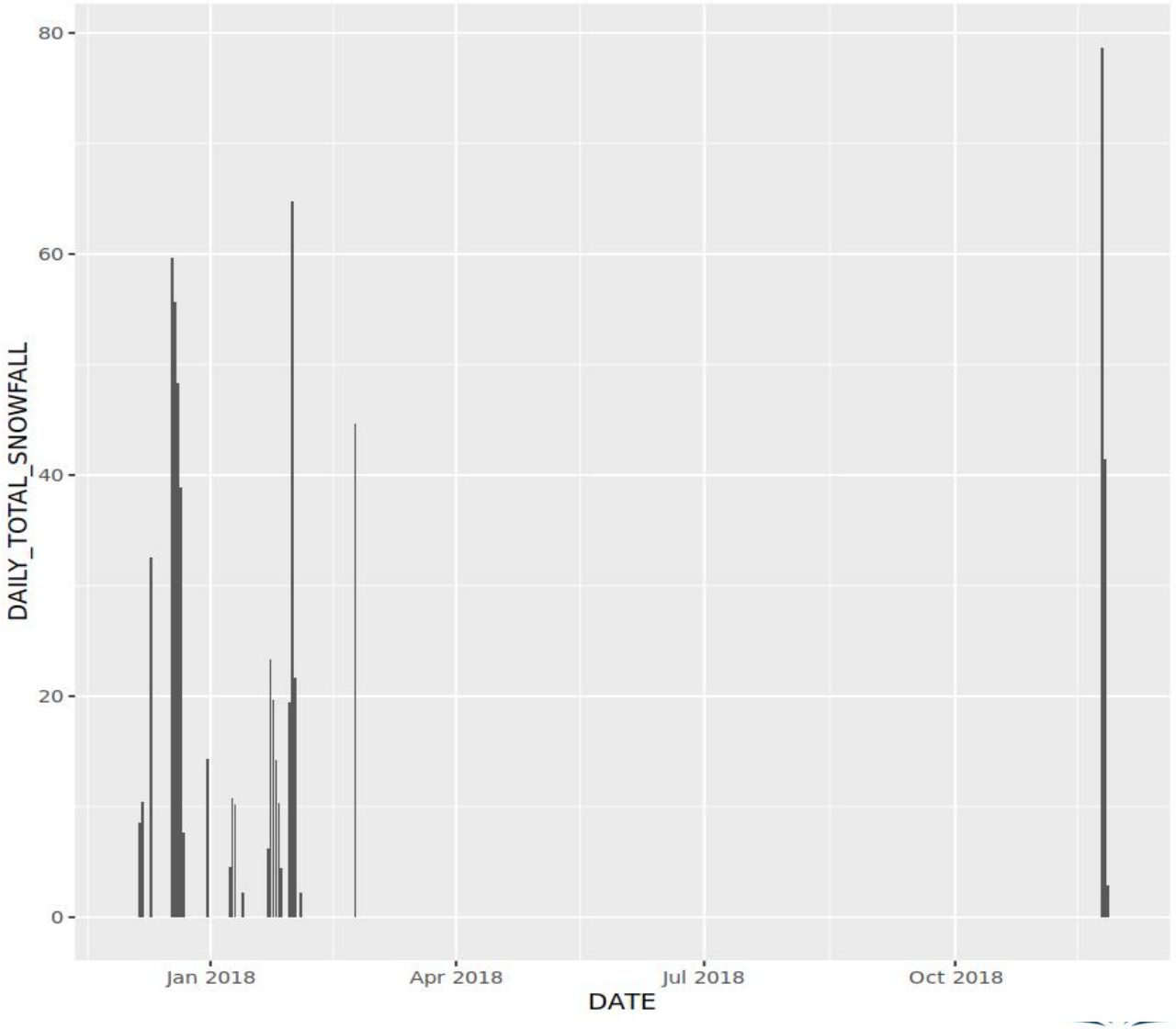


Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DATE, y=DAILY_TOTAL_RAINFALL)) +
  geom_bar(stat = "identity")
```




```
ggplot(seoul_daily_rainfall_snowfall, aes(x=DATE, y=DAILY_TOTAL_SNOWFALL)) +
  geom_bar(stat = "identity")
```



Appendix: Exploratory Data Analysis with tidyverse and ggplot2 data visualization

Task 16 - Determine how many days had snowfall.

Solution 16

In [32]:
```
# provide your solution here
head(seoul_daily_rainfall_snowfall)
filter(seoul_daily_rainfall_snowfall, DAILY_TOTAL_SNOWFALL>0) %>% count()
```

DATE	DAILY_TOTAL_RAINFALL	DAILY_TOTAL_SNOWFALL
<date>	<dbl>	<dbl>
A tibble: 6 × 3		
2017-12-01	0.0	0.0
2017-12-02	0.0	0.0
2017-12-03	4.0	0.0
2017-12-04	0.1	0.0
2017-12-05	0.0	0.0
2017-12-06	1.3	8.6

n
<int>
A tibble: 1 × 1
27

Appendix: Predictive Analysis with building a Baseline Regression Model

TASK: Split training and testing data

First, we need to split the full dataset into training and testing datasets.

The training dataset will be used for fitting regression models, and the testing dataset will be used to evaluate the trained models.

TODO: Use the `initial_split()`, `training()`, and `testing()` functions to generate a training dataset consisting of 75% of the original dataset, and a testing dataset using the remaining 25%.

```
# Use the `initial_split()`, `training()`, and `testing()` functions to split the dataset
# With seed 1234
set.seed(1234)
bike_sharing_split<- initial_split(bike_sharing_df, prop=3/4)
# prop = 3/4
# train_data
train_data<-training(bike_sharing_split)
# test_data
test_data<-testing(bike_sharing_split)
```

TASK: Build a linear regression model using weather variables only

Define a linear regression model specification.

```
# Use `linear_reg()` with engine `lm` and mode `regression`
lm_spec<- linear_reg() %>% set_engine(engine="lm")
lm_spec
```

```
Linear Regression Model Specification (regression)

Computational engine: lm
```

Fit a model with the response variable `RENTED_BIKE_COUNT` and predictor variables `TEMPERATURE + HUMIDITY + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL`

```
# Fit the model called `lm_model_weather`
# RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL,  with the traini
lm_model_weather<- lm_spec %>% fit(RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAIN
```

Print the fit summary for the `lm_model_weather` model.

```
# print(lm_model_weather$fit)
print(lm_model_weather$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY +
    WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION +
    RAINFALL + SNOWFALL, data = data)

Coefficients:
          (Intercept)           TEMPERATURE              HUMIDITY
              171.370              2210.928             -1003.792
           WIND_SPEED            VISIBILITY  DEW_POINT_TEMPERATURE
              422.516                 4.582              -100.903
      SOLAR_RADIATION              RAINFALL              SNOWFALL
             -407.845             -2090.642               331.864
```

Appendix: Predictive Analysis with building a Baseline Regression Model

TASK: Build a linear regression model using all variables

In addition to weather, there could be other factors that may affect bike rental demand, such as the time of a day or if today is a holiday or not.

Next, let's build a linear regression model using all variables (weather + date/time) in this task.

TODO: Build a linear regression model called `lm_model_all` using all variables `RENTED_BIKE_COUNT ~ .`

```
# Fit the model called `lm_model_all`
# `RENTED_BIKE_COUNT ~ .` means use all other variables except for the response variable
lm_model_all<- lm_spec %>% fit(RENTED_BIKE_COUNT ~ ., data=train_data)
```

Print the fit summary for `lm_model_all`.

```
# summary(lm_model_all$fit)
summary(lm_model_all$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ ., data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-1410.1  -217.0    -8.7   200.7  2025.1

Coefficients: (3 not defined because of singularities)
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            254.625     50.277   5.064 4.21e-07 ***
TEMPERATURE            590.970    208.972   2.828 0.004699 **
HUMIDITY              -974.192     97.813  -9.960  < 2e-16 ***
WIND_SPEED               3.264     39.813   0.082 0.934662
VISIBILITY               3.338     19.826   0.168 0.866281
DEW_POINT_TEMPERATURE  796.057    217.519   3.660 0.000255 ***
SOLAR_RADIATION        291.995     41.101   7.104 1.34e-12 ***
RAINFALL             -2317.209    166.583 -13.910  < 2e-16 ***
SNOWFALL               263.825     96.700   2.728 0.006384 **
`0`                    -16.895     33.520  -0.504 0.614249
`1`                   -128.323     33.375  -3.845 0.000122 ***
`10`                  -220.389     32.375  -6.807 1.09e-11 ***
`11`                  -220.593     33.175  -6.649 3.19e-11 ***
`12`                  -203.231     34.280  -5.928 3.22e-09 ***
`13`                  -179.463     34.248  -5.240 1.66e-07 ***
`14`                  -183.466     33.911  -5.410 6.52e-08 ***
`15`                  -108.625     33.647  -3.228 0.001251 **
`16`                    34.530     33.664   1.026 0.305062
`17`                   336.930     33.580  10.034  < 2e-16 ***
`18`                   797.708     34.028  23.443  < 2e-16 ***
`19`                   495.566     34.124  14.522  < 2e-16 ***
`2`                   -220.132     33.173  -6.636 3.49e-11 ***
`20`                   449.443     33.551  13.396  < 2e-16 ***
`21`                   454.048     33.940  13.378  < 2e-16 ***
`22`                   344.139     33.625  10.234  < 2e-16 ***
`23`                   103.665     33.697   3.076 0.002104 **
`3`                   -307.956     33.413  -9.217  < 2e-16 ***
`4`                   -387.267     33.047 -11.719  < 2e-16 ***
`5`                   -367.805     33.257 -11.059  < 2e-16 ***
`6`                   -203.367     33.188  -6.128 9.45e-10 ***
`7`                     92.548     33.193   2.788 0.005316 **
`8`                    454.810     32.460  14.012  < 2e-16 ***
`9`                         NA         NA      NA       NA
AUTUMN                 360.852     20.138  17.919  < 2e-16 ***
SPRING                 202.465     19.188  10.552  < 2e-16 ***
SUMMER                 212.849     28.906   7.364 2.02e-13 ***
WINTER                      NA         NA      NA       NA
HOLIDAY                -99.749     21.812  -4.573 4.89e-06 ***
NO_HOLIDAY                  NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 372.8 on 6312 degrees of freedom
Multiple R-squared:  0.6671,    Adjusted R-squared:  0.6653
F-statistic: 361.4 on 35 and 6312 DF,  p-value: < 2.2e-16
```

Appendix: Predictive Analysis with building a Baseline Regression Model

TASK: Model evaluation and identification of important variables

```
# Use predict() function to generate test results for `lm_model_weather` and `lm_model_all`
# and generate two test results dataframe with a truth column:

# test_results_weather for lm_model_weather model
test_results_weather <- lm_model_weather %>% predict(new_data=test_data) %>% mutate(truth=test_data$RENTED_BIKE_COUNT)
# test_results_all for lm_model_all
test_results_all <- lm_model_all %>% predict(new_data=test_data) %>% mutate(truth=test_data$RENTED_BIKE_COUNT)
```

```
Warning message in predict.lm(object = object$fit, newdata = new_data, type = "response"):
"prediction from a rank-deficient fit may be misleading"
```

NOTE: if you happen to see a warning like : `prediction from a rank-deficient fit may be misleading`, it may be casued by collinearity in the predictor variables. Collinearity means that one predictor variable can be predicted from other predictor variables to some degree. For example, `RAINFALL` could be predicted by `HUMIDITY`.

But dont worry, you will address `glmnet` models (Lasso and Elastic-Net Regularized Generalized Linear Models) instead of regular `regression` models to solve this issue and futher improve the model performance.

Next, let's calculate and print the R-squared and RMSE for the two test results

TODO: Use `rsq()` and `rmse()` functions to calculate R-squared and RMSE metrics for the two test results

```
# rsq_weather <- rsq(...)
# rsq_all <- rsq(...)
rsq_weather <- rsq(test_results_weather, truth=truth, estimate=.pred)
rsq_all <- rsq(test_results_all, truth=truth, estimate=.pred)
# rmse_weather <- rmse(...)
# rmse_all <- rmse(...)
rmse_weather <- rmse(test_results_weather, truth=truth, estimate=.pred)
rmse_all <- rmse(test_results_all, truth=truth, estimate=.pred)
```

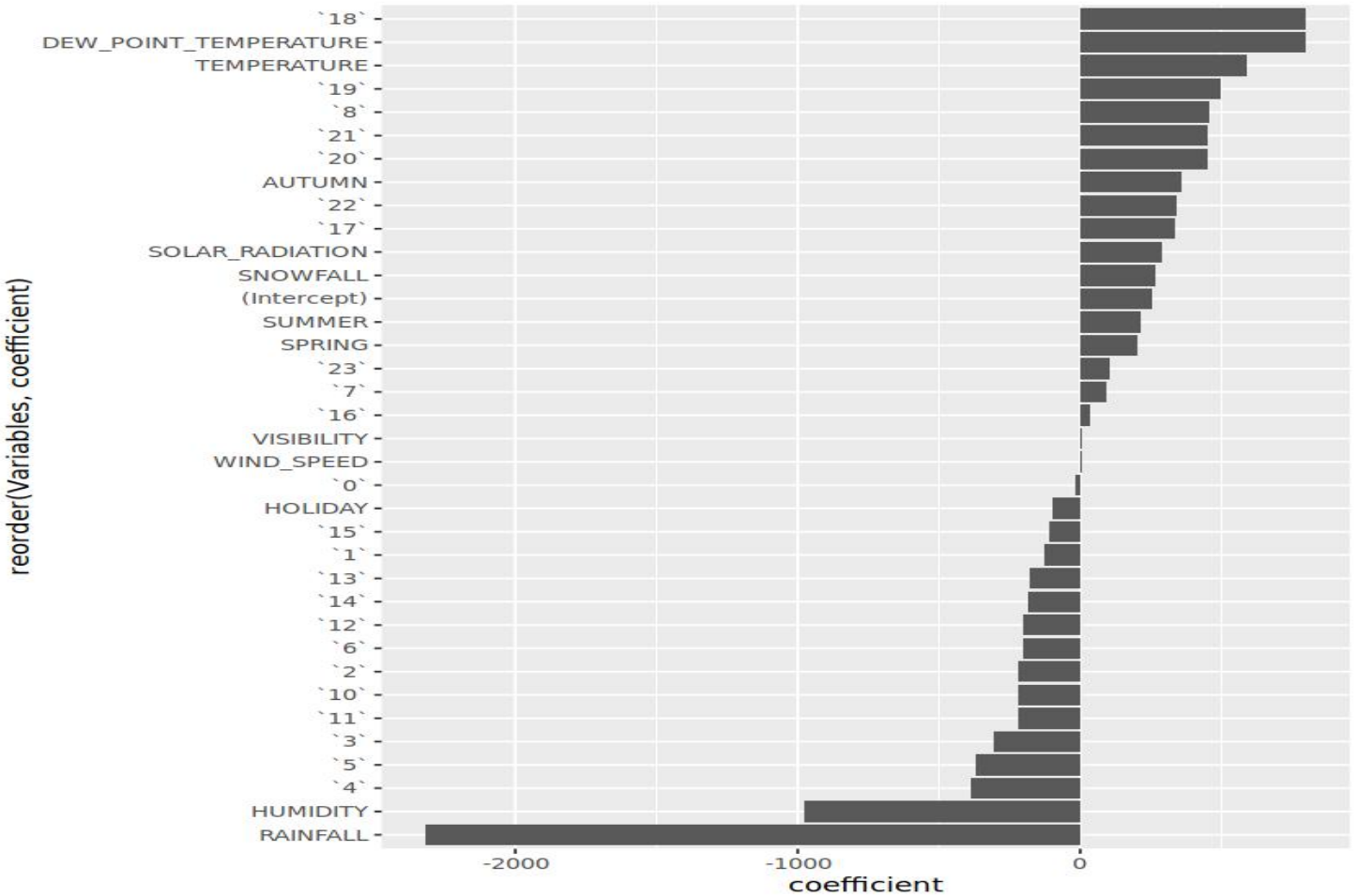
Appendix: Predictive Analysis with building a Baseline Regression Model

```
lm_model_allfitcoefficients
```

(Intercept)	254.624713841887
TEMPERATURE	590.970013169704
HUMIDITY	-974.191665853887
WIND_SPEED	3.26403322110029
VISIBILITY	3.33846758815491
DEW_POINT_TEMPERATURE	796.057154996279
SOLAR_RADIATION	291.99491187606
RAINFALL	-2317.20898113899
SNOWFALL	263.824807690223
`0`	-16.895308187225
`1`	-128.323156930108
`10`	-220.388845946249
`11`	-220.593196981749
`12`	-203.230668526479
`13`	-179.463293490874
`14`	-183.46633365467
`15`	-108.625201366895
`16`	34.5303135068688
`17`	336.929992543532
`18`	797.708010787832
`19`	495.566233185417
`2`	-220.132469200938
`20`	449.442738955973
`21`	454.047517540653
`22`	344.138641820235
`23`	103.665012190421
`3`	-307.955780172193
`4`	-387.267047105129
`5`	-367.804644005323
`6`	-203.366761849977
`7`	92.5484678840224
`8`	454.810323199464
`9`	<NA>
AUTUMN	360.852377661853
SPRING	202.464657365026
SUMMER	212.849374276925
WINTER	<NA>
HOLIDAY	-99.7491194802854
NO_HOLIDAY	<NA>

```
# Sort coefficient list
coefficient<-sort(lm_model_allfitcoefficients, decreasing=TRUE)
Coefficients_df<-data.frame(coefficient)
Variables<-factor(row.names(Coefficients_df))
```

```
# Visualize the list using ggplot and geom_bar
ggplot(Coefficients_df,aes(reorder(Variables,coefficient) , coefficient))+ geom_col()+  coord_flip()
```



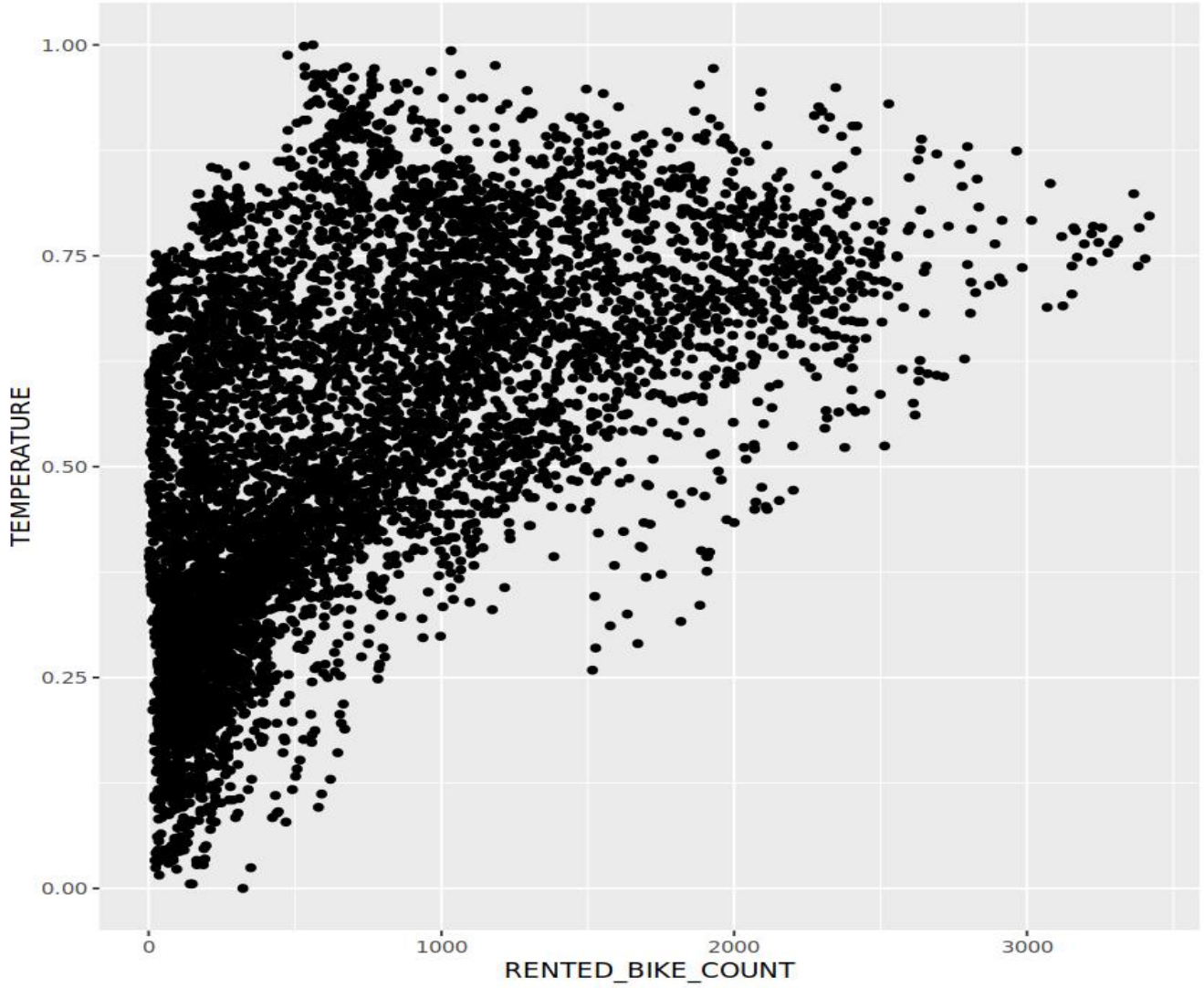
Appendix: Predictive Analysis with improving the linear model

TASK: Add polynomial terms

Linear regression models are the most suitable models to capture the linear correlations among variables. However, in real world data, many relationships may be non-linear.

For example, the correlation between `RENTED_BIKE_COUNT` and `TEMPERATURE` does not look like linear:

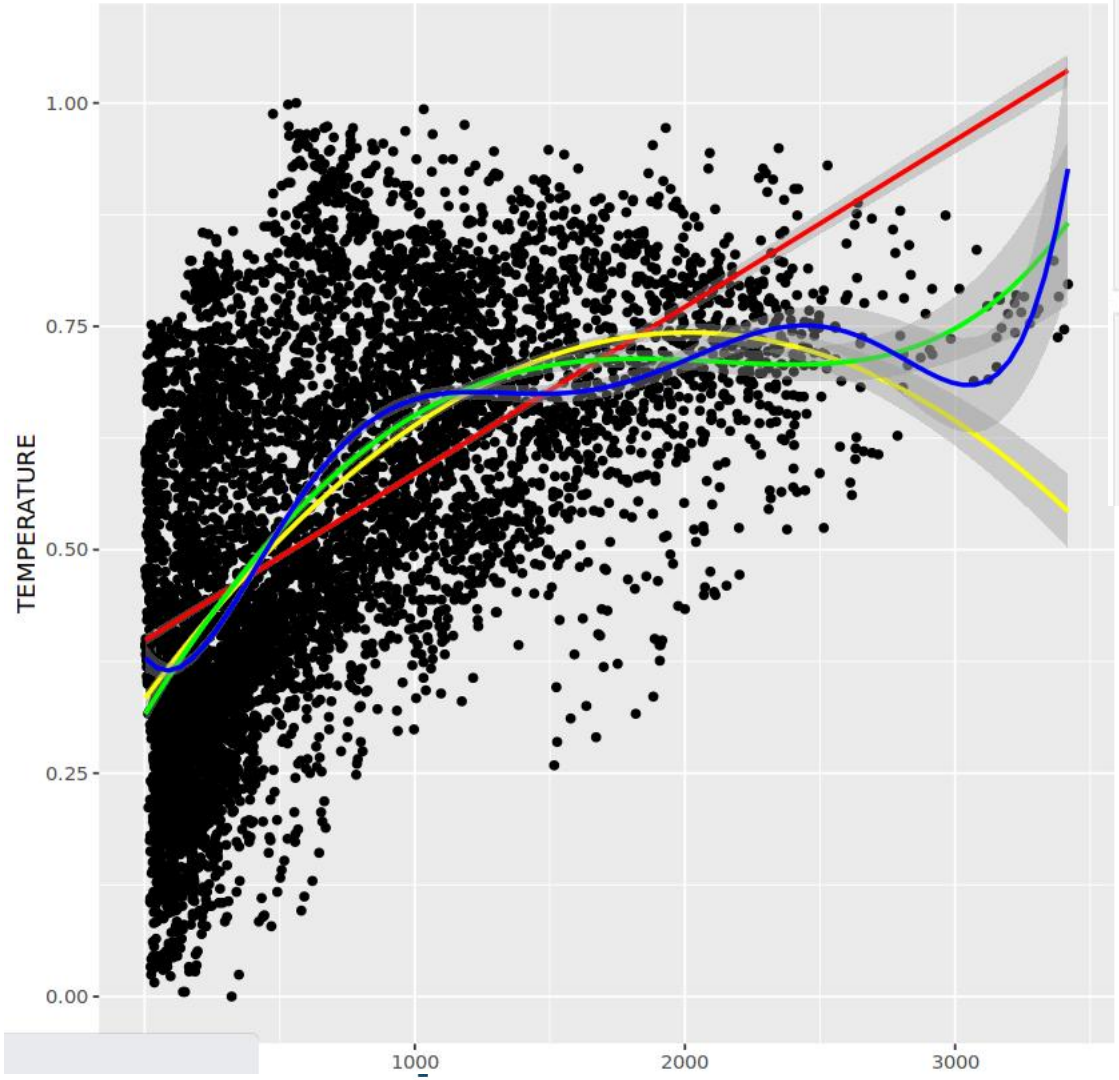
```
ggplot(data = train_data, aes(RENTED_BIKE_COUNT, TEMPERATURE)) +
    geom_point()
```



IBM Developer

Appendix: Predictive Analysis with improving the linear model

```
# Plot the higher order polynomial fits
ggplot(data=train_data, aes(RENTED_BIKE_COUNT, TEMPERATURE)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, color="red") +
    geom_smooth(method = "lm", formula = y ~ poly(x, 2), color="yellow") +
    geom_smooth(method = "lm", formula = y ~ poly(x, 4), color="green") +
    geom_smooth(method = "lm", formula = y ~ poly(x, 6), color="blue")
```




```
# Fit a linear model with higher order polynomial on some important variables

# #HINT: Use ploy function to build polynomial terms, lm_poly <- RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) .....
lm_poly <- RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) + poly(HUMIDITY, 4)+ poly(WIND_SPEED, 4)+ poly(VISIBILITY, 3)+
                               poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 4) +
                               `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `18` + `19` + `2` + `20` +
                               `21` + `22` + `23` + `3` + `4` + `5` + `6` + `7` + `8`+ `9` + AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY


train_fit <- lm_spec %>%
  fit(lm_poly,  data=train_data)
```

```
# Print model summary

# summary(lm_poly$fit)
summary(train_fit$fit)
```

SKILLS NETWORK

Appendix: Predictive Analysis with improving the linear model

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) +
    poly(HUMIDITY, 4) + poly(WIND_SPEED, 4) + poly(VISIBILITY,
    3) + poly(DEW_POINT_TEMPERATURE, 6) + poly(SOLAR_RADIATION,
    5) + poly(RAINFALL, 6) + poly(SNOWFALL, 4) + `0` + `1` +
    `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `18` +
    `19` + `2` + `20` + `21` + `22` + `23` + `3` + `4` + `5` +
    `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER + WINTER +
    HOLIDAY + NO_HOLIDAY, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-1740.41  -181.03     5.47   166.41  1273.15

Coefficients: (3 not defined because of singularities)
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                        268.266     25.716  10.432  < 2e-16 ***
poly(TEMPERATURE, 6)1            38602.565   4229.867   9.126  < 2e-16 ***
poly(TEMPERATURE, 6)2             9664.062    709.405  13.623  < 2e-16 ***
poly(TEMPERATURE, 6)3           -10741.230    452.201 -23.753  < 2e-16 ***
poly(TEMPERATURE, 6)4            -7261.710    429.991 -16.888  < 2e-16 ***
poly(TEMPERATURE, 6)5               28.176    373.887   0.075 0.939930
poly(TEMPERATURE, 6)6             1929.759    359.981   5.361 8.57e-08 ***
poly(HUMIDITY, 4)1                -635.410   2020.776  -0.314 0.753198
poly(HUMIDITY, 4)2               -3445.652    511.522  -6.736 1.76e-11 ***
poly(HUMIDITY, 4)3               -2318.015    371.292  -6.243 4.55e-10 ***
poly(HUMIDITY, 4)4               -1137.423    427.066  -2.663 0.007755 **
poly(WIND_SPEED, 4)1               -434.552    376.046  -1.156 0.247894
poly(WIND_SPEED, 4)2              -1196.774    323.654  -3.698 0.000219 ***
poly(WIND_SPEED, 4)3               -741.600    318.185  -2.331 0.019798 *
poly(WIND_SPEED, 4)4                457.772    315.694   1.450 0.147092
poly(VISIBILITY, 3)1              -1455.631    438.722  -3.318 0.000912 ***
poly(VISIBILITY, 3)2                449.168    347.159   1.294 0.195765
poly(VISIBILITY, 3)3              -1427.409    326.985  -4.365 1.29e-05 ***
poly(DEW_POINT_TEMPERATURE, 6)1 -22734.324   4969.154  -4.575 4.85e-06 ***
poly(DEW_POINT_TEMPERATURE, 6)2 -14046.254    716.295 -19.610  < 2e-16 ***
poly(DEW_POINT_TEMPERATURE, 6)3  -1833.481    457.249  -4.010 6.14e-05 ***
poly(DEW_POINT_TEMPERATURE, 6)4    964.956    417.751   2.310 0.020925 *
poly(DEW_POINT_TEMPERATURE, 6)5    775.914    370.171   2.096 0.036111 *
poly(DEW_POINT_TEMPERATURE, 6)6    830.193    331.322   2.506 0.012245 *
poly(SOLAR_RADIATION, 5)1        12009.756    918.417  13.077  < 2e-16 ***
poly(SOLAR_RADIATION, 5)2       -11033.039    484.377 -22.778  < 2e-16 ***
poly(SOLAR_RADIATION, 5)3         5766.435    384.602  14.993  < 2e-16 ***
poly(SOLAR_RADIATION, 5)4        -3123.697    348.673  -8.959  < 2e-16 ***
poly(SOLAR_RADIATION, 5)5         2525.991    336.015   7.517 6.31e-14 ***
poly(RAINFALL, 6)1               -3905.280    369.401 -10.572  < 2e-16 ***
poly(RAINFALL, 6)2                2660.767    340.569   7.813 6.45e-15 ***
poly(RAINFALL, 6)3               -1524.905    331.201  -4.604 4.22e-06 ***
poly(RAINFALL, 6)4                2746.691    322.789   8.509  < 2e-16 ***
poly(RAINFALL, 6)5               -2155.922    319.000  -6.758 1.51e-11 ***
poly(RAINFALL, 6)6                1742.463    316.481   5.506 3.81e-08 ***
poly(SNOWFALL, 4)1                -500.801    350.361  -1.429 0.152939
poly(SNOWFALL, 4)2                 777.141    329.531   2.358 0.018386 *
poly(SNOWFALL, 4)3                -276.317    334.489  -0.826 0.408783
poly(SNOWFALL, 4)4                -130.617    332.264  -0.393 0.694248
`0`                                321.528     32.344   9.941  < 2e-16 ***
`1`                                205.454     32.406   6.340 2.45e-10 ***
`10`                              -209.025     26.363  -7.929 2.58e-15 ***
`11`                              -158.828     27.339  -5.810 6.55e-09 ***
`12`                               -56.458     28.253  -1.998 0.045724 *
`13`                               -46.091     28.441  -1.621 0.105157
`14`                               -34.976     27.918  -1.253 0.210318
`15`                                 6.625     27.643   0.240 0.810596
`16`                               118.001     27.527   4.287 1.84e-05 ***
`17`                               401.130     27.289  14.700  < 2e-16 ***
`18`                               919.057     28.280  32.498  < 2e-16 ***
`19`                               741.880     30.010  24.721  < 2e-16 ***
`2`                                105.017     32.031   3.279 0.001049 **
`20`                               796.528     32.108  24.808  < 2e-16 ***
`21`                               820.118     32.791  25.010  < 2e-16 ***
`22`                               689.736     32.585  21.167  < 2e-16 ***
`23`                               462.290     32.495  14.227  < 2e-16 ***
`3`                                 31.572     32.148   0.982 0.326097
`4`                                -47.965     31.901  -1.504 0.132735
`5`                                -20.809     32.081  -0.649 0.516594
`6`                                130.379     31.658   4.118 3.86e-05 ***
`7`                                299.601     29.297  10.226  < 2e-16 ***
`8`                                519.212     26.796  19.377  < 2e-16 ***
`9`                                     NA         NA      NA        NA
AUTUMN                             371.833     19.516  19.053  < 2e-16 ***
SPRING                             208.836     18.876  11.064  < 2e-16 ***
SUMMER                             300.650     24.712  12.166  < 2e-16 ***
WINTER                                  NA         NA      NA        NA
HOLIDAY                            -84.725     18.030  -4.699 2.66e-06 ***
NO_HOLIDAY                              NA         NA      NA        NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Appendix: Predictive Analysis with improving the linear model

```
# Use predict() function to generate test results for `lm_poly`
test_results <- train_fit %>%
predict(new_data = test_data) %>%
mutate(truth = test_data$RENTED_BIKE_COUNT)
head(test_results)
```

```
Warning message in predict.lm(object = object$fit, newdata = new_
"prediction from a rank-deficient fit may be misleading"
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
127.102562	254
-3.939999	204
-232.061186	100
80.637429	460
315.503696	930
539.709057	398

```
# e.g., test_results[test_results<0] <- 0
test_results[test_results<0] <- 0
```

Now, calculate R-squared and RMSE for the test results generated by `lm_ploy` model

```
# Calculate R-squared and RMSE from the test results
rsq_1<-rsq(test_results, truth = truth, estimate = .pred)

rmse_1<-rmse(test_results, truth = truth, estimate = .pred)


model_1_results<-c( rsq_1, rmse_1)
model_1_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.771285774483661
$.metric	'rmse'
$.estimator	'standard'
$.estimate	305.848811787698

Appendix: Predictive Analysis with improving the linear model

TASK: Add interaction terms

In real-world scenarios, in addition to non-linear relationships between response variables and predictor variables, you may also encounter relationships among variables called `interaction effects`.

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

To capture such interaction effects, we could add some interaction terms such as `RAINFALL*HUMIDITY` to the regression model, similar to what we did with polynominal terms. In this task, you will explore and conduct some experiments to search for interaction terms which will improve the model performance.

TODO: Try adding some interaction terms to the previous polynomial models.

```
# Add interaction terms to the poly regression built in previous step

# HINT: You could use `*` operator to create interaction terms such as HUMIDITY*TEMPERATURE and make the formula look like:
# RENTED_BIKE_COUNT ~ RAINFALL*HUMIDITY ...
lm_interactive <- RENTED_BIKE_COUNT ~ `18`*TEMPERATURE*DEW_POINT_TEMPERATURE + RAINFALL*HUMIDITY*`4` +SOLAR_RADIATION*SNOWFALL+
                                    WIND_SPEED*VISIBILITY +`18`*TEMPERATURE* +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4)+ poly(WIND_SPEED, 4)+ poly(VISI
  poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 4) +
  `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17`  + `19` + `2` + `20` +
  `21` + `22` + `23` + `3`  + `5` + `6` + `7` + `8`+ `9` + AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY

train_fit_2 <- lm_spec %>%
  fit(lm_interactive,  data=train_data)

train_fit_2
```

Appendix: Predictive Analysis with improving the linear model

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ `18` * TEMPERATURE *
    DEW_POINT_TEMPERATURE + RAINFALL * HUMIDITY * `4` + SOLAR_RADIATION *
    SNOWFALL + WIND_SPEED * VISIBILITY + `18` * TEMPERATURE *
    +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) + poly(WIND_SPEED,
    4) + poly(VISIBILITY, 3) + poly(DEW_POINT_TEMPERATURE, 6) +
    poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6) + poly(SNOWFALL,
    4) + `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` +
    `16` + `17` + `19` + `2` + `20` + `21` + `22` + `23` + `3` +
    `5` + `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER +
    WINTER + HOLIDAY + NO_HOLIDAY, data = data)

Coefficients:
                                  (Intercept)                                           `18`
                                      1905.03                                         -54.45
                                  TEMPERATURE                            DEW_POINT_TEMPERATURE
                                     -1745.35                                       -7335.56
                                     RAINFALL                                       HUMIDITY
                                    -11539.95                                         587.40
                                          `4`                                 SOLAR_RADIATION
                                       -21.80                                         503.07
                                     SNOWFALL                                     WIND_SPEED
                                       -62.54                                        -183.19
                                   VISIBILITY                           poly(TEMPERATURE, 6)1
                                       -91.16                                             NA
                        poly(TEMPERATURE, 6)2                           poly(TEMPERATURE, 6)3
                                     -9084.46                                       -10880.02
                        poly(TEMPERATURE, 6)4                           poly(TEMPERATURE, 6)5
                                     -7137.86                                        -1022.73
                        poly(TEMPERATURE, 6)6                              poly(HUMIDITY, 4)1
                                      -432.87                                             NA
                           poly(HUMIDITY, 4)2                              poly(HUMIDITY, 4)3
                                     -1751.93                                        -3850.38
                           poly(HUMIDITY, 4)4                           poly(WIND_SPEED, 4)1
                                      -897.96                                             NA
                         poly(WIND_SPEED, 4)2                            poly(WIND_SPEED, 4)3
                                     -1285.05                                         -697.99
                         poly(WIND_SPEED, 4)4                            poly(VISIBILITY, 3)1
                                       368.81                                             NA
                         poly(VISIBILITY, 3)2                            poly(VISIBILITY, 3)3
                                       388.98                                        -1391.17
      poly(DEW_POINT_TEMPERATURE, 6)1                 poly(DEW_POINT_TEMPERATURE, 6)2
                                           NA                                       -34162.01
      poly(DEW_POINT_TEMPERATURE, 6)3                 poly(DEW_POINT_TEMPERATURE, 6)4
                                     -1529.42                                         929.40
      poly(DEW_POINT_TEMPERATURE, 6)5                 poly(DEW_POINT_TEMPERATURE, 6)6
                                       619.69                                        1094.87
            poly(SOLAR_RADIATION, 5)1                       poly(SOLAR_RADIATION, 5)2
                                           NA                                       -8889.49
            poly(SOLAR_RADIATION, 5)3                       poly(SOLAR_RADIATION, 5)4
                                      4583.15                                       -2649.33
            poly(SOLAR_RADIATION, 5)5                              poly(RAINFALL, 6)1
                                      2446.66                                             NA
                  poly(RAINFALL, 6)2                              poly(RAINFALL, 6)3
                                      2514.35                                       -1366.80
                  poly(RAINFALL, 6)4                              poly(RAINFALL, 6)5
                                      2581.49                                       -2002.60
                                     -202.87                                          248.42
                                          `1`                                          `10`
                                       130.65                                       -202.49
                                          `11`                                         `12`
                                     -151.21                                        -47.55
                                          `13`                                         `14`
                                     -39.29                                         -25.55
                                          `15`                                         `16`
                                      21.26                                         134.71
                                          `17`                                         `19`
                                      411.22                                         701.09
                                          `2`                                          `20`
                                      29.55                                          731.26
                                          `21`                                         `22`
                                      749.77                                         618.49
                                          `23`                                         `3`
                                      389.94                                        -42.22
                                          `5`                                          `6`
                                     -95.23                                         57.81
                                          `7`                                          `8`
                                      245.01                                         495.50
                                          `9`                                      AUTUMN
                                          NA                                         362.02
                                      SPRING                                       SUMMER
                                      206.31                                         301.28
                                      WINTER                                      HOLIDAY
                                          NA                                        -78.59
                                  NO_HOLIDAY                          `18`:TEMPERATURE
                                          NA                                       2578.48
          `18`:DEW_POINT_TEMPERATURE          TEMPERATURE:DEW_POINT_TEMPERATURE
                                     -210.98                                       9272.40
                  RAINFALL:HUMIDITY                                RAINFALL:`4`
                                    10139.20                                     -35728.97
                        HUMIDITY:`4`                          SOLAR_RADIATION:SNOWFALL
                                     -157.40                                       -876.79
              WIND_SPEED:VISIBILITY                `18`:poly(TEMPERATURE, 6)1
                                      186.09                                             NA
          `18`:poly(TEMPERATURE, 6)2                `18`:poly(TEMPERATURE, 6)3
                                     2043.88                                      -1605.91
          `18`:poly(TEMPERATURE, 6)4                `18`:poly(TEMPERATURE, 6)5
                                    -2447.71                                      3610.29
          `18`:poly(TEMPERATURE, 6)6           TEMPERATURE:poly(TEMPERATURE, 6)1
                                    -3917.69                                            NA
TEMPERATURE:poly(TEMPERATURE, 6)2           TEMPERATURE:poly(TEMPERATURE, 6)3
                                          NA                                            NA
TEMPERATURE:poly(TEMPERATURE, 6)4           TEMPERATURE:poly(TEMPERATURE, 6)5
                                          NA                                            NA
TEMPERATURE:poly(TEMPERATURE, 6)6 `18`:TEMPERATURE:DEW_POINT_TEMPERATURE
                                     4565.48                                      -1019.59
        RAINFALL:HUMIDITY:`4`      `18`:TEMPERATURE:poly(TEMPERATURE, 6)1
                                    40581.51                                            NA
```

SKILLS NETWORK

Appendix: Predictive Analysis with improving the linear model

```
# Print model summary
summary(train_fit_2$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ `18` * TEMPERATURE *
    DEW_POINT_TEMPERATURE + RAINFALL * HUMIDITY * `4` + SOLAR_RADIATION *
    SNOWFALL + WIND_SPEED * VISIBILITY + `18` * TEMPERATURE *
    +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4) + poly(WIND_SPEED,
    4) + poly(VISIBILITY, 3) + poly(DEW_POINT_TEMPERATURE, 6) +
    poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6) + poly(SNOWFALL,
    4) + `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` +
    `16` + `17` + `19` + `2` + `20` + `21` + `22` + `23` + `3` +
    `5` + `6` + `7` + `8` + `9` + AUTUMN + SPRING + SUMMER +
    WINTER + HOLIDAY + NO_HOLIDAY, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-2069.64  -175.47     5.46   165.68  1260.71

Coefficients: (22 not defined because of singularities)
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         1905.03     781.06   2.439 0.014753
`18`                                 -54.45     288.23  -0.189 0.850175
TEMPERATURE                        -1745.35    1473.33  -1.185 0.236205
DEW_POINT_TEMPERATURE              -7335.56    2505.56  -2.928 0.003426
RAINFALL                          -11539.95    6391.21  -1.806 0.071026
HUMIDITY                             587.40     304.09   1.932 0.053438
`4`                                  -21.80      79.83  -0.273 0.784778
SOLAR_RADIATION                      503.07      45.49  11.059  < 2e-16
SNOWFALL                             -62.54      96.50  -0.648 0.516956
WIND_SPEED                          -183.19      79.64  -2.300 0.021466
VISIBILITY                           -91.16      26.51  -3.438 0.000589
poly(TEMPERATURE, 6)1                    NA         NA      NA       NA
poly(TEMPERATURE, 6)2                -9084.46    6760.65  -1.344 0.179082
poly(TEMPERATURE, 6)3               -10880.02     488.43 -22.276  < 2e-16
poly(TEMPERATURE, 6)4                -7137.86     433.18 -16.478  < 2e-16
poly(TEMPERATURE, 6)5                -1022.73     496.35  -2.061 0.039388
poly(TEMPERATURE, 6)6                 -432.87     783.10  -0.553 0.580438
poly(HUMIDITY, 4)1                       NA         NA      NA       NA
poly(HUMIDITY, 4)2                   -1751.93     975.10  -1.797 0.072436
poly(HUMIDITY, 4)3                   -3850.38     643.07  -5.987 2.24e-09
poly(HUMIDITY, 4)4                    -897.96     470.62  -1.908 0.056429
poly(WIND_SPEED, 4)1                     NA         NA      NA       NA
poly(WIND_SPEED, 4)2                 -1285.05     324.56  -3.959 7.59e-05
poly(WIND_SPEED, 4)3                  -697.99     314.58  -2.219 0.026532
poly(WIND_SPEED, 4)4                   368.81     311.34   1.185 0.236233
poly(VISIBILITY, 3)1                     NA         NA      NA       NA
poly(VISIBILITY, 3)2                   388.98     344.62   1.129 0.259055
poly(VISIBILITY, 3)3                 -1391.17     322.56  -4.313 1.63e-05
poly(DEW_POINT_TEMPERATURE, 6)1          NA         NA      NA       NA
poly(DEW_POINT_TEMPERATURE, 6)2      -34162.01    8341.90  -4.095 4.27e-05
poly(DEW_POINT_TEMPERATURE, 6)3       -1529.42     493.01  -3.102 0.001929
poly(DEW_POINT_TEMPERATURE, 6)4         929.40     413.88   2.246 0.024764
poly(DEW_POINT_TEMPERATURE, 6)5         619.69     369.59   1.677 0.093654
poly(DEW_POINT_TEMPERATURE, 6)6        1094.87     336.83   3.250 0.001158
poly(SOLAR_RADIATION, 5)1                NA         NA      NA       NA
poly(SOLAR_RADIATION, 5)2            -8889.49     504.53 -17.620  < 2e-16
poly(SOLAR_RADIATION, 5)3             4583.15     389.84  11.756  < 2e-16
poly(SOLAR_RADIATION, 5)4            -2649.33     346.24  -7.652 2.26e-14
poly(SOLAR_RADIATION, 5)5             2446.66     332.23   7.364 1.99e-13
poly(RAINFALL, 6)1                       NA         NA      NA       NA
poly(RAINFALL, 6)2                    2514.35     387.09   6.495 8.87e-11
poly(RAINFALL, 6)3                   -1366.80     351.87  -3.884 0.000104
poly(RAINFALL, 6)4                    2581.49     322.31   8.009 1.35e-15
poly(RAINFALL, 6)5                   -2002.60     324.13  -6.178 6.86e-10
poly(RAINFALL, 6)6                    1727.17     314.34   5.495 4.06e-08
poly(SNOWFALL, 4)1                       NA         NA      NA       NA
poly(SNOWFALL, 4)2                    619.52     328.76   1.884 0.059556
poly(SNOWFALL, 4)3                   -212.48     329.57  -0.645 0.519131
poly(SNOWFALL, 4)4                   -202.87     331.85  -0.611 0.540989
`0`                                   248.42      32.28   7.696 1.61e-14
`1`                                   130.65      32.35   4.038 5.44e-05
`10`                                 -202.49      25.97  -7.797 7.32e-15
`11`                                 -151.21      26.95  -5.610 2.11e-08
`12`                                  -47.55      27.92  -1.703 0.088585
`13`                                  -39.29      28.10  -1.398 0.162053
`14`                                  -25.55      27.58  -0.926 0.354372
`15`                                   21.26      27.32   0.778 0.436488
`16`                                  134.71      27.22   4.948 7.66e-07
`17`                                  411.22      26.93  15.272  < 2e-16
`19`                                  701.09      29.71  23.598  < 2e-16
`2`                                    29.55      32.00   0.924 0.355744
`20`                                  731.26      31.96  22.880  < 2e-16
`21`                                  749.77      32.69  22.939  < 2e-16
`22`                                  618.49      32.50  19.032  < 2e-16
`23`                                  389.94      32.41  12.030  < 2e-16
`3`                                   -42.22      32.10  -1.315 0.188461
`5`                                   -95.23      32.05  -2.971 0.002974
`6`                                    57.81      31.61   1.829 0.067461
`7`                                   245.01      29.11   8.417  < 2e-16
`8`                                   495.50      26.44  18.741  < 2e-16
`9`                                      NA         NA      NA       NA
AUTUMN                                362.02      19.37  18.694  < 2e-16
SPRING                                206.31      18.68  11.043  < 2e-16
SUMMER                                301.28      24.59  12.250  < 2e-16
WINTER                                   NA         NA      NA       NA
HOLIDAY                               -78.59      17.81  -4.413 1.04e-05
NO_HOLIDAY                               NA         NA      NA       NA
`18`:TEMPERATURE                      2578.48     622.33   4.143 3.47e-05
`18`:DEW_POINT_TEMPERATURE            -210.98     725.96  -0.291 0.771352
TEMPERATURE:DEW_POINT_TEMPERATURE     9272.40    3631.12   2.554 0.010684
RAINFALL:HUMIDITY                    10139.20    6518.31   1.555 0.119876
RAINFALL:`4`                        -35728.97   37984.48  -0.941 0.346933
HUMIDITY:`4`                          -157.40     105.50  -1.492 0.135784
SOLAR_RADIATION:SNOWFALL              -876.79     824.86  -1.063 0.287836
WIND_SPEED:VISIBILITY                  186.09      98.37   1.892 0.058582
`18`:poly(TEMPERATURE, 6)1                NA         NA      NA       NA
`18`:poly(TEMPERATURE, 6)2             2043.88    5299.52   0.386 0.699751
`18`:poly(TEMPERATURE, 6)3            -1605.91    3075.27  -0.522 0.601546
`18`:poly(TEMPERATURE, 6)4            -2447.71    3302.25  -0.741 0.458582
`18`:poly(TEMPERATURE, 6)5             3610.29    2503.74   1.442 0.149361
`18`:poly(TEMPERATURE, 6)6            -3917.69    7334.37  -0.534 0.593252
RAINFALL:HUMIDITY:`4`                 40581.51   39601.33   1.025 0.305518
`18`:TEMPERATURE:poly(TEMPERATURE, 6)1       NA         NA      NA       NA
`18`:TEMPERATURE:poly(TEMPERATURE, 6)2       NA         NA      NA       NA
`18`:TEMPERATURE:poly(TEMPERATURE, 6)3       NA         NA      NA       NA
`18`:TEMPERATURE:poly(TEMPERATURE, 6)4       NA         NA      NA       NA
`18`:TEMPERATURE:poly(TEMPERATURE, 6)5       NA         NA      NA       NA
`18`:TEMPERATURE:poly(TEMPERATURE, 6)6   1560.88    9727.28   0.160 0.872520

(Intercept)                         *
`18`
TEMPERATURE
DEW_POINT_TEMPERATURE               **
RAINFALL                            .
HUMIDITY                            .
`4`
SOLAR_RADIATION                     ***
SNOWFALL
WIND_SPEED                          *
VISIBILITY                          ***
poly(TEMPERATURE, 6)1
poly(TEMPERATURE, 6)2
poly(TEMPERATURE, 6)3               ***
poly(TEMPERATURE, 6)4               ***
poly(TEMPERATURE, 6)5               *
poly(TEMPERATURE, 6)6
poly(HUMIDITY, 4)1
poly(HUMIDITY, 4)2                  .
poly(HUMIDITY, 4)3                  ***
poly(HUMIDITY, 4)4                  .
poly(WIND_SPEED, 4)1
poly(WIND_SPEED, 4)2                ***
poly(WIND_SPEED, 4)3                *
poly(WIND_SPEED, 4)4
poly(VISIBILITY, 3)1
poly(VISIBILITY, 3)2
poly(VISIBILITY, 3)3                ***
poly(DEW_POINT_TEMPERATURE, 6)1
poly(DEW_POINT_TEMPERATURE, 6)2     ***
poly(DEW_POINT_TEMPERATURE, 6)3     **
poly(DEW_POINT_TEMPERATURE, 6)4     *
poly(DEW_POINT_TEMPERATURE, 6)5     .
poly(DEW_POINT_TEMPERATURE, 6)6     **
poly(SOLAR_RADIATION, 5)1
poly(SOLAR_RADIATION, 5)2           ***
poly(SOLAR_RADIATION, 5)3           ***
poly(SOLAR_RADIATION, 5)4           ***
poly(SOLAR_RADIATION, 5)5           ***
poly(RAINFALL, 6)1
poly(RAINFALL, 6)2                  ***
poly(RAINFALL, 6)3                  ***
poly(RAINFALL, 6)4                  ***
poly(RAINFALL, 6)5                  ***
poly(RAINFALL, 6)6                  ***
poly(SNOWFALL, 4)1
poly(SNOWFALL, 4)2                  .
poly(SNOWFALL, 4)3
poly(SNOWFALL, 4)4
`0`                                 ***
```

Appendix: Predictive Analysis with improving the linear model

```
`0`                                    ***
`1`                                    ***
`10`                                   ***
`11`                                   ***
`12`                                   .
`13`
`14`
`15`
`16`                                   ***
`17`                                   ***
`19`                                   ***
`2`
`20`                                   ***
`21`                                   ***
`22`                                   ***
`23`                                   ***
`3`
`5`                                    **
`6`                                    .
`7`                                    ***
`8`                                    ***
`9`
AUTUMN                                 ***
SPRING                                 ***
SUMMER                                 ***
WINTER
HOLIDAY                                ***
NO_HOLIDAY
`18`:TEMPERATURE                       ***
`18`:DEW_POINT_TEMPERATURE
TEMPERATURE:DEW_POINT_TEMPERATURE      *
RAINFALL:HUMIDITY
RAINFALL:`4`
HUMIDITY:`4`
SOLAR_RADIATION:SNOWFALL
WIND_SPEED:VISIBILITY                  .
`18`:poly(TEMPERATURE, 6)1
`18`:poly(TEMPERATURE, 6)2
`18`:poly(TEMPERATURE, 6)3
`18`:poly(TEMPERATURE, 6)4
`18`:poly(TEMPERATURE, 6)5
`18`:poly(TEMPERATURE, 6)6
TEMPERATURE:poly(TEMPERATURE, 6)1
TEMPERATURE:poly(TEMPERATURE, 6)2
TEMPERATURE:poly(TEMPERATURE, 6)3
TEMPERATURE:poly(TEMPERATURE, 6)4
TEMPERATURE:poly(TEMPERATURE, 6)5
TEMPERATURE:poly(TEMPERATURE, 6)6      ***
`18`:TEMPERATURE:DEW_POINT_TEMPERATURE
RAINFALL:HUMIDITY:`4`
`18`:TEMPERATURE:poly(TEMPERATURE, 6)1
`18`:TEMPERATURE:poly(TEMPERATURE, 6)2
`18`:TEMPERATURE:poly(TEMPERATURE, 6)3
`18`:TEMPERATURE:poly(TEMPERATURE, 6)4
`18`:TEMPERATURE:poly(TEMPERATURE, 6)5
`18`:TEMPERATURE:poly(TEMPERATURE, 6)6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 306.6 on 6689 degrees of freedom
Multiple R-squared:  0.7761,    Adjusted R-squared:  0.7733
F-statistic: 282.7 on 82 and 6689 DF,  p-value: < 2.2e-16
```

16]:

```
# Calculate R-squared and RMSE for the new model to see if performance has improved
test_results_2 <- train_fit_2 %>%
predict(new_data = test_data) %>%
mutate(truth = test_data$RENTED_BIKE_COUNT)

head(test_results_2)
test_results_2[test_results_2<0] <- 0
```

```
Warning message in predict.lm(object = object$fit, newdata = new_data, type = "response"):
 "prediction from a rank-deficient fit may be misleading"
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
115.35980	254
-25.03473	204
-246.11749	100
85.92372	460
352.11412	930
538.93776	398

17]:

```
rsq_2<-rsq(test_results_2, truth = truth, estimate = .pred)

rmse_2<- rmse(test_results_2, truth = truth, estimate = .pred)

model_2_results<-c(rsq_2, rmse_2)
model_2_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.772629482653108
$.metric	'rmse'
$.estimator	'standard'
$.estimate	304.46639380277

Appendix: Predictive Analysis with improving the linear model

TASK: Add regularization

```
# Fit a glmnet model using the fit() function
glmnet_spec <- linear_reg(penalty = 0.2, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
```

```
# Report rsq and rmse of the `lm_glmnet` model
glmnet_fit<-glmnet_spec %>% fit(RENTED_BIKE_COUNT ~ `18`*TEMPERATURE*DEW_POINT_TEMPERATURE + RAINFALL*HUMIDITY*`4` +SOLAR_RADIATION*SNOWFALL +
                                  WIND_SPEED*VISIBILITY +`18`*TEMPERATURE* +poly(TEMPERATURE, 6) + poly(HUMIDITY, 4)+ poly(WIND_SPEED, 4)+ poly(VISIBILI
                                  poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 4) +
                                  `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `19` + `2` + `20` +
                                  `21` + `22` + `23` + `3` + `5` + `6` + `7` + `8`+ `9` + AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY ,
                                data = train_data )
```

```
test_results_glment <- glmnet_fit %>%
predict(new_data = test_data) %>%
mutate(truth = test_data$RENTED_BIKE_COUNT)

head(test_results_glment)
```

.pred	truth
<dbl>	<dbl>
A tibble: 6 × 2	
125.97474	254
-13.93975	204
-235.76937	100
97.92126	460
364.40560	930
527.00551	398

```
test_results_glment[test_results_glment<0] <- 0
rsq_3<-rsq(test_results_glment, truth = truth, estimate = .pred)

rmse_3<-rmse(test_results_glment, truth = truth, estimate = .pred)

model_3_results<-c(rsq_3, rmse_3)
model_3_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.776136730923099
$.metric	'rmse'
$.estimator	'standard'
$.estimate	302.333828956636

Appendix: Predictive Analysis with improving the linear model

TASK: Experiment to search for improved models

Now you understand some of the methods that you can use to try to improve your models.

TODO: Experiment by building and testing at least five different models. For each of your experiments, include polynomial terms, interaction terms, and one of the three regularizations we introduced.

```
# Build at least five different models using polynomial terms, interaction terms, and regularizations.

# Save their rmse and rsq values
```

```
model_prediction<- function(model_fit, test_data )  {
  test_results<-model_fit %>%
    predict(new_data = test_data) %>%
    mutate(truth = test_data$RENTED_BIKE_COUNT)
  test_results[test_results<0] <- 0
  return(test_results)}
```

```
model_evaluation <- function(test_results){
    rsq_model<-rsq(test_results, truth = truth, estimate = .pred)
    rmse_model<-rmse(test_results, truth = truth, estimate = .pred)
    results<-c(rsq_model, rmse_model)
    return(results)
}
```

```
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

Appendix: Predictive Analysis with improving the linear model

```
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

```
model_4_fit <- lm_spec %>% fit(RENTED_BIKE_COUNT ~ `18`*TEMPERATURE*DEW_POINT_TEMPERATURE + RAINFALL*HUMIDITY*`4` +SOLAR_RADIATION*SNOWFALL+
                               WIND_SPEED*VISIBILITY +`18`*TEMPERATURE+RAINFALL*HUMIDITY*`18` +poly(TEMPERATURE, 6) + poly(HUMIDITY, 6)+ poly(WIND_SPEED,
                               poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 2) + `18`+`4` +
                               `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `19` + `2` + `20` +
                               `21` + `22` + `23` + `3` + `5` + `6` + `7` + `8`+ `9` + AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY+
                               AUTUMN*SPRING*SUMMER*WINTER*HOLIDAY*`18`*`4`*`19`, data = train_data )

results_model_4<-model_prediction(model_4_fit, test_data)
model_4_results<-model_evaluation(results_model_4)
model_4_results
```

```
Warning message in predict.lm(object = object$fit, newdata = new_data, type = "response"):
"prediction from a rank-deficient fit may be misleading"
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.782932552014311
$.metric	'rmse'
$.estimator	'standard'
$.estimate	296.977827428816

```
glmnet_spec <- linear_reg(penalty = 0.2, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
```

Appendix: Predictive Analysis with improving the linear model

```
glmnet_fit<-glmnet_spec %>% fit(RENTED_BIKE_COUNT ~ `18`*TEMPERATURE*DEW_POINT_TEMPERATURE + RAINFALL*HUMIDITY*`4` +
                                SOLAR_RADIATION*SNOWFALL+ WIND_SPEED*VISIBILITY +`18`*TEMPERATURE+RAINFALL*HUMIDITY*`18` +
                                poly(TEMPERATURE, 6) + poly(HUMIDITY, 6)+ poly(WIND_SPEED, 2)+ poly(VISIBILITY, 3)+
                                poly(DEW_POINT_TEMPERATURE,6)+ poly(SOLAR_RADIATION, 5) + poly(RAINFALL, 6)+ poly(SNOWFALL, 2) +
                                `18`+`4`+ `0` + `1` + `10` + `11` + `12` + `13` + `14` + `15` + `16` + `17` + `19` +
                                `2` + `20` + `21` + `22` + `23` + `3` + `5` + `6` + `7` + `8`+ `9` +
                                AUTUMN+ SPRING + SUMMER + WINTER + HOLIDAY + NO_HOLIDAY+
                                AUTUMN*SPRING*SUMMER*WINTER*HOLIDAY*`18`*`4`*`19`, data = train_data )


results_model_5<-model_prediction(glmnet_fit, test_data)
model_5_results<-model_evaluation(results_model_5)
model_5_results
```

$.metric	'rsq'
$.estimator	'standard'
$.estimate	0.785709328328774
$.metric	'rmse'
$.estimator	'standard'
$.estimate	295.334507497087

```
rsq_rsme_data<-data.frame(model_1_results)
rsq_rsme_data<-rbind(rsq_rsme_data, model_2_results, model_3_results, model_4_results, model_5_results)
rsq_rsme_data['model']<-c("model_1", "model_2", "model_3", "model_4", "model_5")
colnames(rsq_rsme_data)[6]<-"RSME"
colnames(rsq_rsme_data)[3]<-"Rsquared"
rsq_rsme_data
```

	.metric	.estimator	Rsquared	.metric.1	.estimator.1	RSME	model
	<fct>	<fct>	<dbl>	<fct>	<fct>	<dbl>	<chr>
A data.frame: 5 × 7							
1	rsq	standard	0.7712858	rmse	standard	305.8488	model_1
2	rsq	standard	0.7726295	rmse	standard	304.4664	model_2
3	rsq	standard	0.7761367	rmse	standard	302.3338	model_3
4	rsq	standard	0.7829326	rmse	standard	296.9778	model_4
5	rsq	standard	0.7857093	rmse	standard	295.3345	model_5

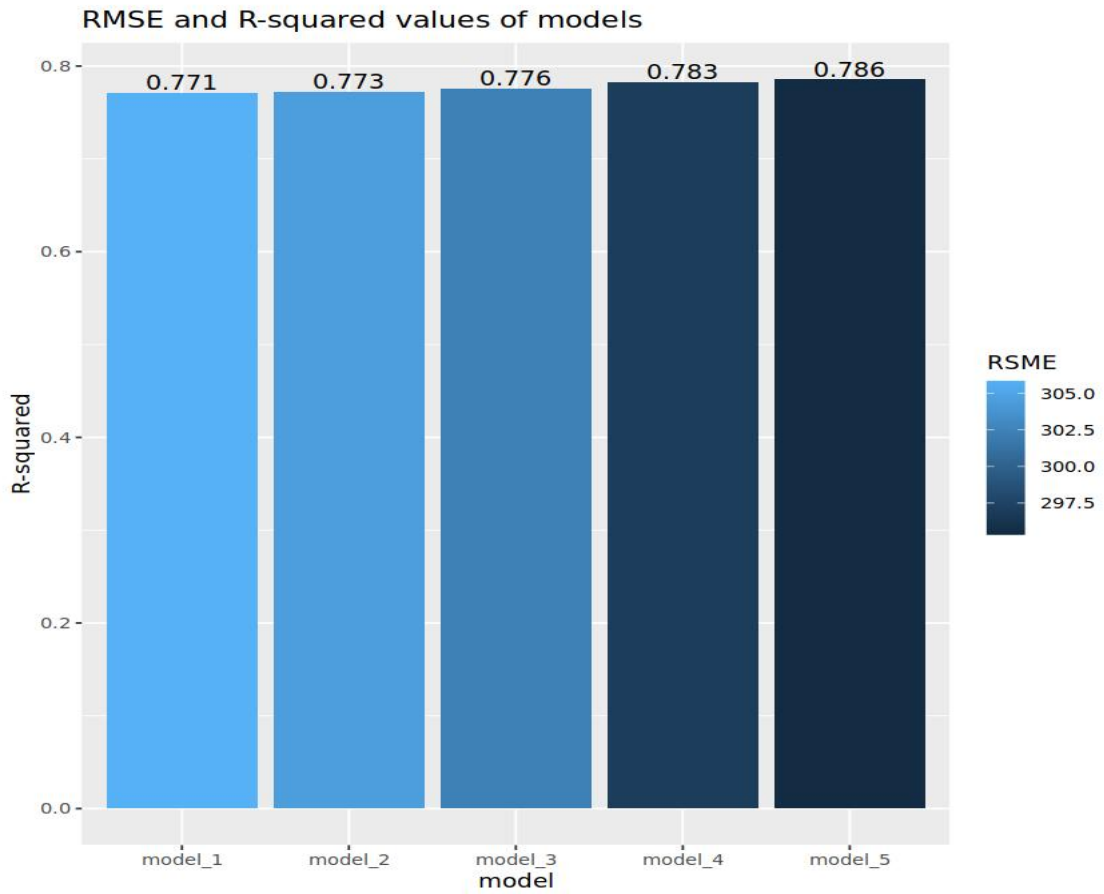
Here are the performance requirements for your best model:

- The RMSE should be less than 330 (rougly 10% of the max value in test dataset)
- R-squared should be greater than 0.72

TODO: Visualize the saved RMSE and R-squared values using a grouped barchart

Appendix: Predictive Analysis with improving the linear model

```
# HINT: Use ggplot() + geom_bar()
library(ggplot2)

ggplot(rsq_rsme_data, aes(fill=RSME, x=model, y=Rsquared))+
  geom_bar(position="stack", stat="identity")+
  geom_text(aes(label = format(round(Rsquared, 3), nsmall = 2)), vjust = -0.2, size = 4)+
  ylab("R-squared")+
  ggtitle("RMSE and R-squared values of models ")
```



Appendix: Predictive Analysis with improving the linear model

```
# HINT: Use ggplot() +
    # stat_qq(aes(sample=truth), color='green') +
    # stat_qq(aes(sample=prediction), color='red')
    ggplot(results_model_5)+
  stat_qq(aes(sample=truth), color='green') +
  stat_qq(aes(sample=.pred), color='red')
```