Applied Data Science with R Capstone project

Outline

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

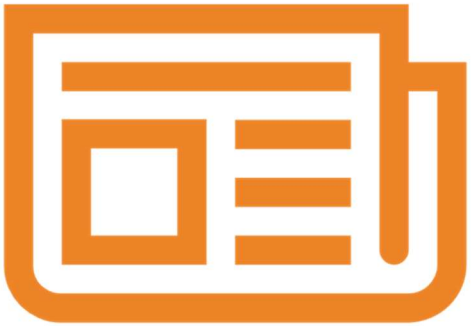
Executive Summary

- This project focuses on predicting how weather would affect bike sharing demand in urban areas by applying machine learning algorithms to a variety of weather related and bike sharing demand data.
- The datasets used for this project include:
  - The Seoul Bike Sharing Demand Data Set which contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), and the number of bikes rented per hour and date.
  - Global Bike Sharing Systems Dataset which is an HTML table on the Wikipedia page List of bicycle-sharing systems: https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems. It lists active bicycle-sharing systems around the world. Most systems listed allow users to pick up and drop off bicycles at any of the automated stations within the network.
  - World Cities Data which contains information such as name, latitude, and longitude, about major cities around the world.
- Data wrangling and pre-processing was performed on the datasets to prepare them for further analysis. This stage was especially important as it was crucial in ensuring that meaningful results were yielded from the analysis of the data.
- Exploratory Data Analysis was also carried out on the cleaned and pre-processed data in order to detect useful patterns and trends in the data sets. This was achieved by making use of the RSQLite package, which allows the data to be queried using Structured Query Language and ggplot2, a data visualization package in R, which allows for the generation of various charts and plots to show the relationships within the data.
- Based on the noted trends and strong relationships detected in the EDA phase, a linear model was built in order to predict the number of bikes rented each hour, based on the weather.
- Finally, the results of the predictive linear regression model were combined with an interactive map and visualized on a live dashboard created using the Shiny package in R. This dashboard showed select urban locations and associated visualization of the current weather and the estimated bike demand in these locations.
- This project will help to provide crucial insights that would aid supply optimization given that it would it would be beneficial to be able to predict the number of bikes required at each hour of the day based on current weather conditions in order to help optimize supply.

Introduction

- Alternative clean modes of transportation are not relevant in today's context of global warming, but also necessary. One such mode of transportation is the bike-sharing system.
- Currently, to increase transportation comfort, several cities have introduced rental bikes. It is vital to maximize rental bike supply so that they are available for public usage at all times, since this decreases waiting time. Providing a consistent supply of rental bikes to the city eventually becomes a serious challenge. The essential element is anticipating the number of bikes that will be in demand during each hour of the day to guarantee that bikes are ready to go so that the service is easily available to users and there is no lack of rental bikes.
- The entire bike-sharing rental process is heavily influenced by environmental and seasonal factors. Weather conditions, precipitation, weekday, season of the year, hour of the day, and so on can all influence rental behaviour. The goal of this project was to create a model that predicted the total number of bike rentals for any given hour of the day based on current weather conditions and then display the results in an interactive dashboard.
- Managers and users of bike sharing rental services will be able to use my predictive model at the end of this project to:
  - Aid supply optimization and,
  - Ensure that users enjoy a seamless bike sharing services as a result of regulated supply.

Methodology



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

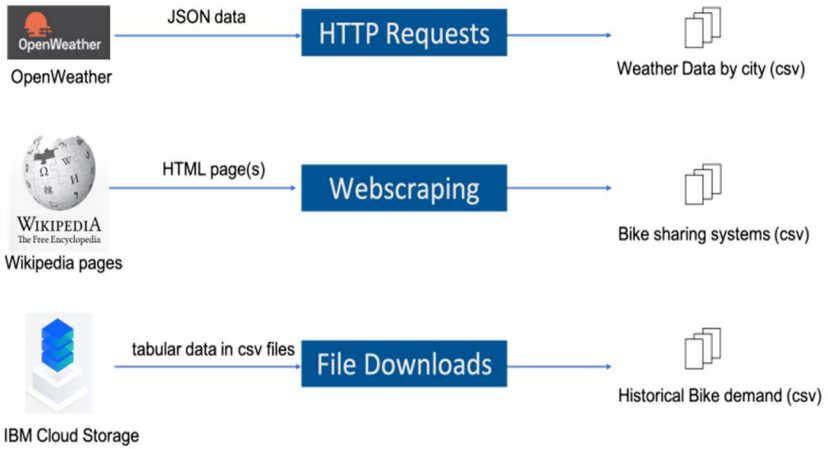
Methodology

This describes the processes taken to create an interactive dashboard for predicting how weather would effect bike sharing demand in metropolitan regions by applying machine learning algorithms to a range of weather and temporal data.

Data collection

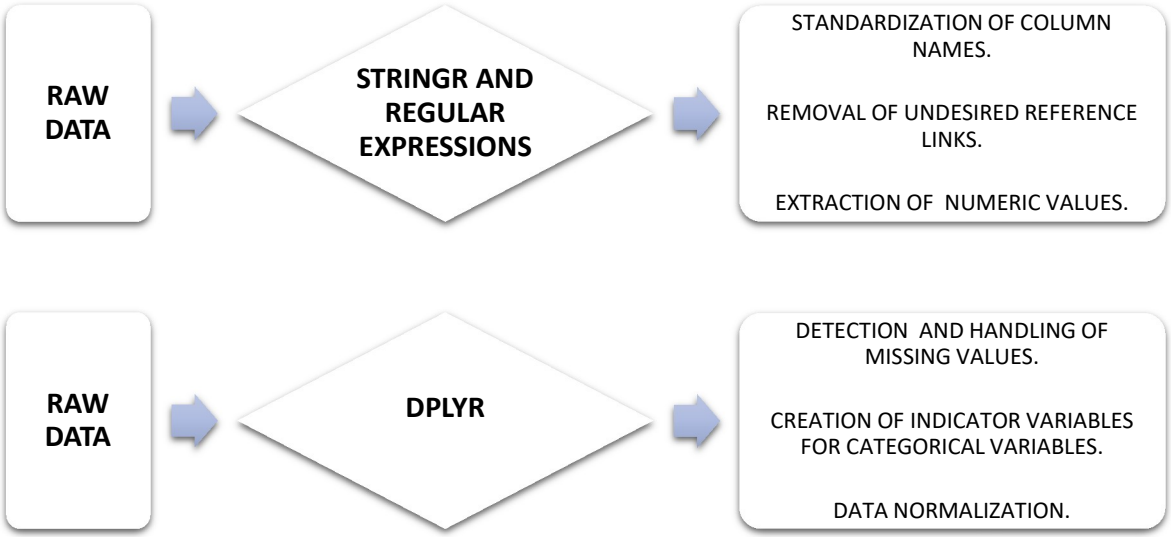
The datasets used for this project include:

- 5-day weather forecast for a list of cities from OpenWeather API.
- Global Bike Sharing Systems Dataset which is an HTML table on the Wikipedia page List of bicycle-sharing systems: https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems. It lists active bicycle-sharing systems around the world. Most systems listed allow users to pick up and drop off bicycles at any of the automated stations within the network.
- World Cities Data which contains information such as name, latitude, and longitude, about major cities around the world.
- The Seoul Bike Sharing Demand Data Set which contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), and the number of bikes rented per hour and date.



Data wrangling

- This stage involved cleaning the data by checking for missing values, mis-formatted values and/or unexpected noises.
- The first step involved the use of the R package ‘stringr’ and regular expression language to standardize column names, remove unwanted reference links from tables and extract numeric values from rows.
- The next step involved the use of the ‘dplyr’ package to detect and handle missing values in the data tables, create indicator (dummy) variables for categorical variables, and normalize the data using min-max normalization.



EDA with SQL

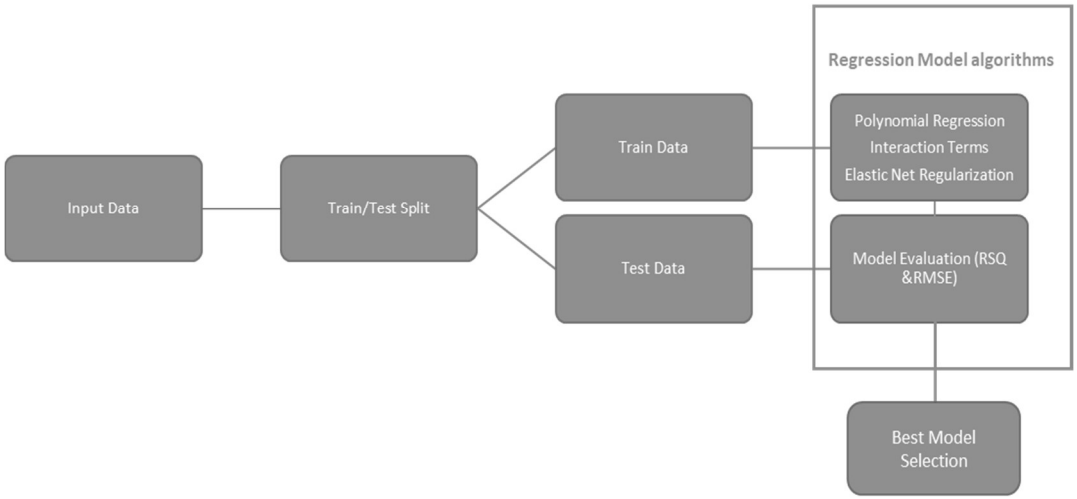
- Exploratory Data Analysis was performed on the datasets using Structured Query Language (SQL). The queries performed to;
    - Determine how many records are in the seoul_bike_sharing dataset.
    - Determine how many hours had non-zero rented bike count.
    - Query the weather forecast for Seoul 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.
    - 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.
    - Find the average hourly bike count during each season.
    - Determine the average TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season
    - Determine the Total Bike Count and City Info for Seoul
    - Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system.

EDA with data visualization

- Exploratory Data Analysis was also performed to visually inspect the datasets using the ggplot2 library. The ggplot2 library was used to;
    - 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.
    - Create a scatter plot to visualize the correlation between `RENTED_BIKE_COUNT` and `TEMPERATURE` by `SEASONS`.
    - Create a display of four boxplots of `RENTED_BIKE_COUNT` vs. `HOUR` grouped by `SEASONS`.
    - Create a line plot after grouping the data by `DATE`, and using the summarize() function to visualize the daily total rainfall and snowfall.

Predictive analysis

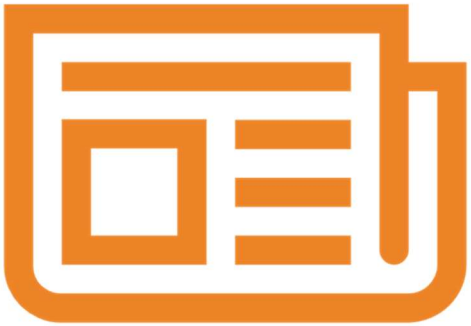
- This stage involved building and refining a regression model to predict the number of bikes that would be rented at each hour of the day by factoring in weather and non-weather conditions.
- The cleaned and processed data is divided into two sets: Train set (for model creation and refinement via polynomial regression, interaction, and regularization) and Test set (for Model Evaluation). The constructed models were evaluated using RMSE and RSQ as our assessment metrics, and the regression algorithm was chosen based on the lowest RMSE and highest RSQ values on the Test data.



Build a R Shiny dashboard

- The results of the predictive linear regression model were combined with an interactive dashboard created using the shiny package in R. This dashboard contained;
  - A basic max bike prediction overview map.
  - A static temperature trend line.
  - An interactive bike-sharing demand prediction trend line.
  - A static humidity and bike-sharing demand prediction correlation plot.

Results



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

EDA with SQL

EDA is used to examine and investigate data sets and characterize their key aspects, such as data types, measures of spread and skewness, and so on. Structured Query Language is a prominent method of carrying out EDA. It helps data scientists identify patterns, detect anomalies, test hypotheses, and confirm assumptions by understanding how to effectively manipulate data sources to acquire the desired answers.

Busiest bike rental times

In order to determine which date and hour had the most bike rentals, the following query was used;

"*dbGetQuery(conn, 'select DATE, HOUR, RENTED_BIKE_COUNT from SEOUL_BIKE where RENTED_BIKE_COUNT = (select max(RENTED_BIKE_COUNT) from SEOUL_BIKE)')"*

The result of the query showed that at 6pm on 19/06/2018, the highest ever number of bikes, a total of 3556 bikes were rented.

Hourly popularity and temperature by seasons

In order to determine the average hourly temperature and the average number of bike rentals per hour over each season, the following query was used;

“dbGetQuery(conn, 'select avg(TEMPERATURE) as "Average Hourly Temp", avg(RENTED_BIKE_COUNT) as "Average Bike Rentals/Hour", HOUR, SEASONS

from SEOUL_BIKE

group by HOUR, SEASONS

order by avg(RENTED_BIKE_COUNT) DESC

limit 10’)”

The result of the query showed that more bikes were rented during Summer, Autumn or Spring, between the hours of 5pm and 10pm and at temperatures between 15 degrees Celsius and 30 degrees Celsius which are typical Autumn, Summer and Spring temperatures.

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
dbGetQuery(conn, 'select avg(TEMPERATURE) as "Average Hourly Temp",
                  avg(RENTED_BIKE_COUNT) as "Average Bike Rentals/Hour",
                  HOUR,
                  SEASONS
                  from SEOUL_BIKE
                  group by HOUR, SEASONS
                  order by avg(RENTED_BIKE_COUNT) DESC
                  limit 10')
```

✓ 0.2s

A data.frame: 10 × 4

Average Hourly Temp	Average Bike Rentals/Hour	HOUR	SEASONS
<dbl>	<dbl>	<int>	<chr>
29.38791	2135.141	18	Summer
16.03185	1983.333	18	Autumn
28.27378	1889.250	19	Summer
27.06630	1801.924	20	Summer
26.27826	1754.065	21	Summer
15.97222	1689.311	18	Spring
25.69891	1567.870	22	Summer
17.27778	1562.877	17	Autumn
30.07691	1526.293	17	Summer
15.06346	1515.568	19	Autumn

Rental Seasonality

In order to determine the average hourly bike count during each season, the following query was used;

“dbGetQuery(conn, 'select

SEASONS,

avg(RENTED_BIKE_COUNT) AS AVERAGE,

min(RENTED_BIKE_COUNT) AS MINIMUM,

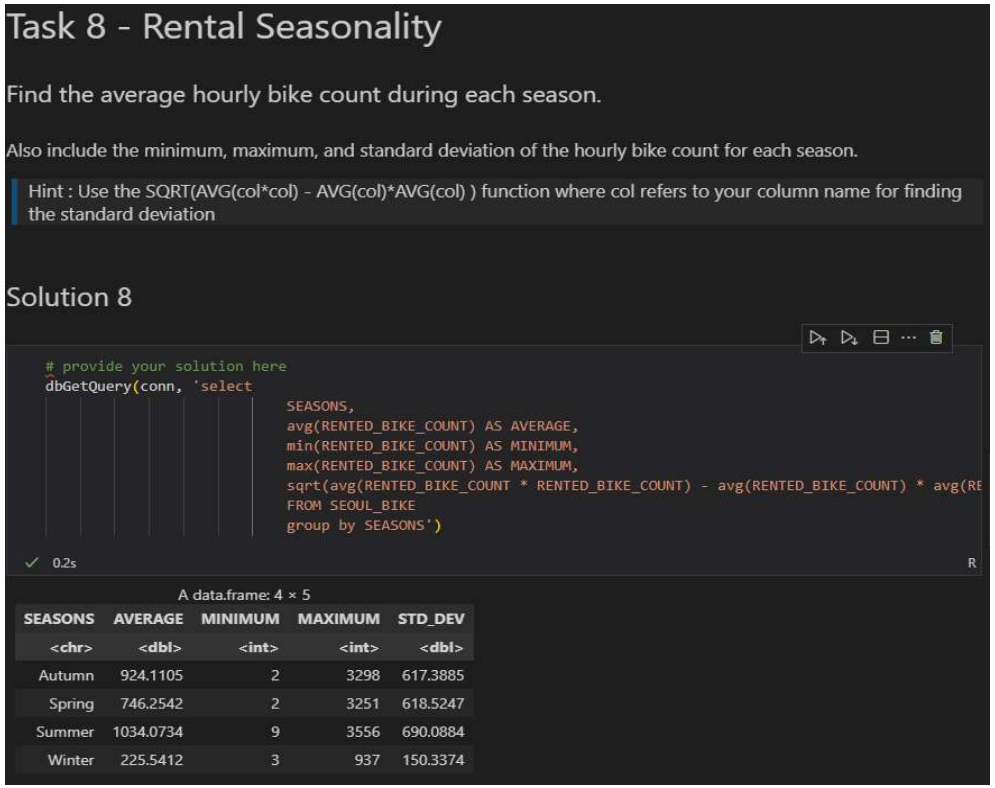
max(RENTED_BIKE_COUNT) AS MAXIMUM,

*sqrt(avg(RENTED_BIKE_COUNT * RENTED_BIKE_COUNT) - avg(RENTED_BIKE_COUNT) * avg(RENTED_BIKE_COUNT)) AS STD_DEV*

FROM SEOUL_BIKE

group by SEASONS')”

The result of the query showed that on average more bikes were rented during Summer and fewer bikes were rented during Winter.



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.

Hint : Use the SQRT(AVG(col*col) - AVG(col)*AVG(col)) function where col refers to your column name for finding the standard deviation

Solution 8

```
# provide your solution here
dbGetQuery(conn, 'select
                  SEASONS,
                  avg(RENTED_BIKE_COUNT) AS AVERAGE,
                  min(RENTED_BIKE_COUNT) AS MINIMUM,
                  max(RENTED_BIKE_COUNT) AS MAXIMUM,
                  sqrt(avg(RENTED_BIKE_COUNT * RENTED_BIKE_COUNT) - avg(RENTED_BIKE_COUNT) * avg(RE
                  FROM SEOUL_BIKE
                  group by SEASONS')
```

✓ 0.2s R

A data.frame: 4 × 5

SEASONS	AVERAGE	MINIMUM	MAXIMUM	STD_DEV
<chr>	<dbl>	<int>	<int>	<dbl>
Autumn	924.1105	2	3298	617.3885
Spring	746.2542	2	3251	618.5247
Summer	1034.0734	9	3556	690.0884
Winter	225.5412	3	937	150.3374

Weather Seasonality

In order to answer the question: On average, what were the TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season?, the following query was used;

"dbGetQuery(conn, 'select

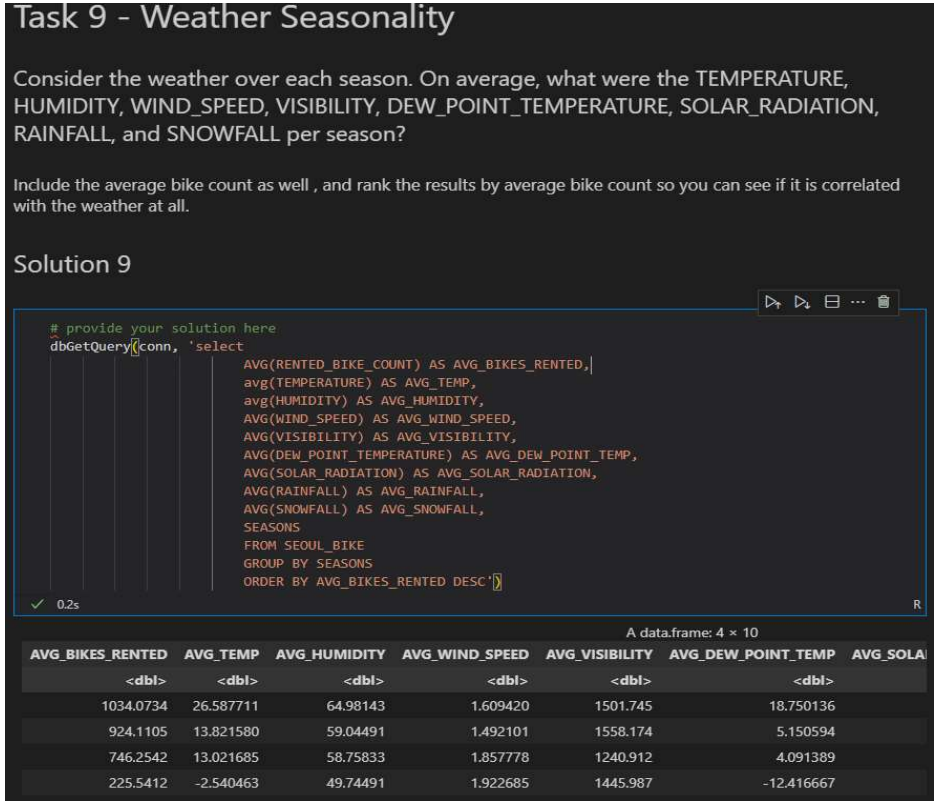
AVG(RENTED_BIKE_COUNT) AS AVG_BIKES_RENTED, avg(TEMPERATURE) AS AVG_TEMP, avg(HUMIDITY) AS AVG_HUMIDITY, AVG(WIND_SPEED) AS AVG_WIND_SPEED, AVG(VISIBILITY) AS AVG_VISIBILITY, AVG(DEW_POINT_TEMPERATURE) AS AVG_DEW_POINT_TEMP, AVG(SOLAR_RADIATION) AS AVG_SOLAR_RADIATION, AVG(RAINFALL) AS AVG_RAINFALL, AVG(SNOWFALL) AS AVG_SNOWFALL, SEASONS

FROM SEOUL_BIKE

GROUP BY SEASONS

ORDER BY AVG_BIKES_RENTED DESC')"

The results showed that the temperature, rainfall, humidity, snowfall, etc. readings were very correlated with the season of the year. For example, there was a higher average temperature and rainfall in the summer, while snowfall was confined to the winter season. It also showed a correlation between the number of rented bikes and the season of the year.



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

```
# provide your solution here
dbGetQuery(conn, 'select
                  AVG(RENTED_BIKE_COUNT) AS AVG_BIKES_RENTED,
                  avg(TEMPERATURE) AS AVG_TEMP,
                  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_TEMP,
                  AVG(SOLAR_RADIATION) AS AVG_SOLAR_RADIATION,
                  AVG(RAINFALL) AS AVG_RAINFALL,
                  AVG(SNOWFALL) AS AVG_SNOWFALL,
                  SEASONS
                  FROM SEOUL_BIKE
                  GROUP BY SEASONS
                  ORDER BY AVG_BIKES_RENTED DESC')
```

✓ 0.2s R

A data.frame: 4 × 10

AVG_BIKES_RENTED	AVG_TEMP	AVG_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT_TEMP	AVG_SOLA
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
1034.0734	26.587711	64.98143	1.609420	1501.745	18.750136	
924.1105	13.821580	59.04491	1.492101	1558.174	5.150594	
746.2542	13.021685	58.75833	1.857778	1240.912	4.091389	
225.5412	-2.540463	49.74491	1.922685	1445.987	-12.416667	

Bike-sharing info in Seoul

In order to determine the total number of bikes available in Seoul, the following query was used;

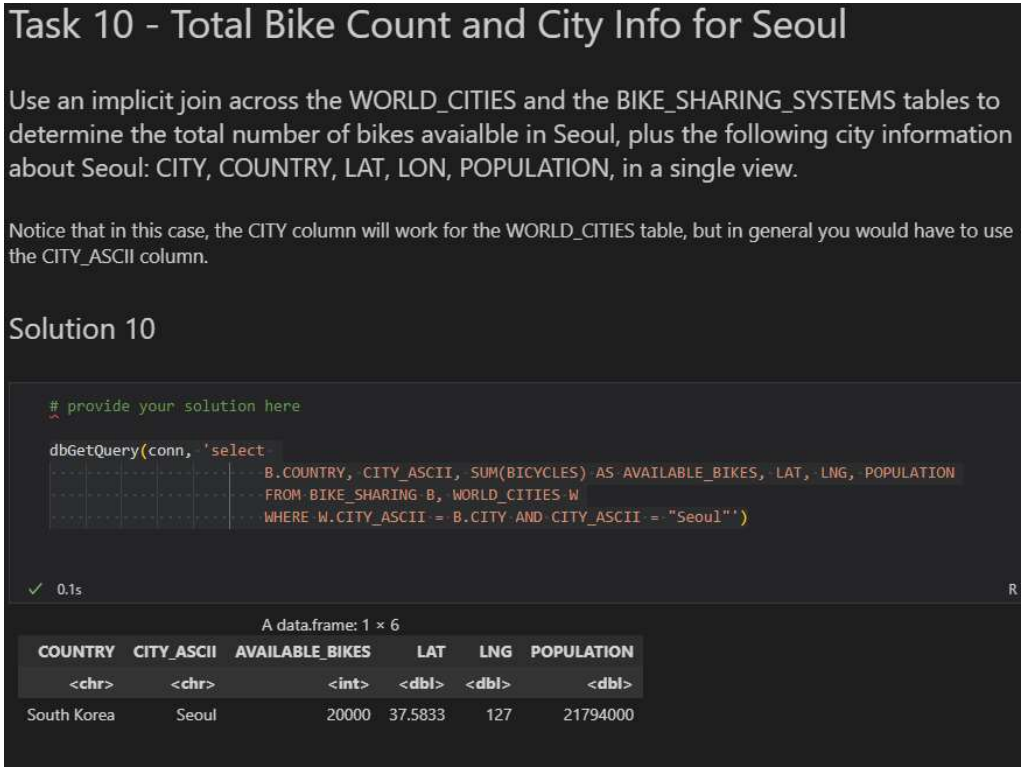
“dbGetQuery(conn, 'select

B.COUNTRY, CITY_ASCII, SUM(BICYCLES) AS AVAILABLE_BIKES, LAT, LNG, POPULATION

FROM BIKE_SHARING B, WORLD_CITIES W

WHERE W.CITY_ASCII = B.CITY AND CITY_ASCII = "Seoul"')”

The result of the query showed that there were 20,000 bikes available for rent in the city of Seoul.



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

dbGetQuery(conn, 'select
                  B.COUNTRY, CITY_ASCII, SUM(BICYCLES) AS AVAILABLE_BIKES, LAT, LNG, POPULATION
                  FROM BIKE_SHARING B, WORLD_CITIES W
                  WHERE W.CITY_ASCII = B.CITY AND CITY_ASCII = "Seoul"')
```

✓ 0.1s R

A data.frame: 1 × 6

COUNTRY	CITY_ASCII	AVAILABLE_BIKES	LAT	LNG	POPULATION
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>
South Korea	Seoul	20000	37.5833	127	21794000

Cities similar to Seoul

In order to determine all cities having a similar number of bikes for rental like Seoul(i.e. total bike counts between 15000 and 20000), the following query was used;

“dbGetQuery(conn, 'select

B.COUNTRY, CITY_ASCII, BICYCLES AS AVAILABLE_BIKES, LAT, LNG, POPULATION

FROM BIKE_SHARING B, WORLD_CITIES W

WHERE W.CITY_ASCII = B.CITY AND AVAILABLE_BIKES BETWEEN 15000 AND 20000

GROUP BY CITY_ASCII

ORDER BY by AVAILABLE_BIKES DESC')”

The result of the query showed that six cities in China namely; Zhuzhou, Xi’an, Weifang, Shanghai, Beijing and Ningbo had similar number of rental bikes like Seoul.

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
dbGetQuery(conn, 'select
                  B.COUNTRY, CITY_ASCII, BICYCLES AS AVAILABLE_BIKES, LAT, LNG, POPULATION
                  FROM BIKE_SHARING B, WORLD_CITIES W
                  WHERE W.CITY_ASCII = B.CITY
                  AND CITY_ASCII != "Seoul"
                  AND AVAILABLE_BIKES BETWEEN 15000 AND 20000
                  GROUP BY CITY_ASCII
                  order by AVAILABLE_BIKES DESC')
```

✓ 0.4s

A data.frame: 6 × 6

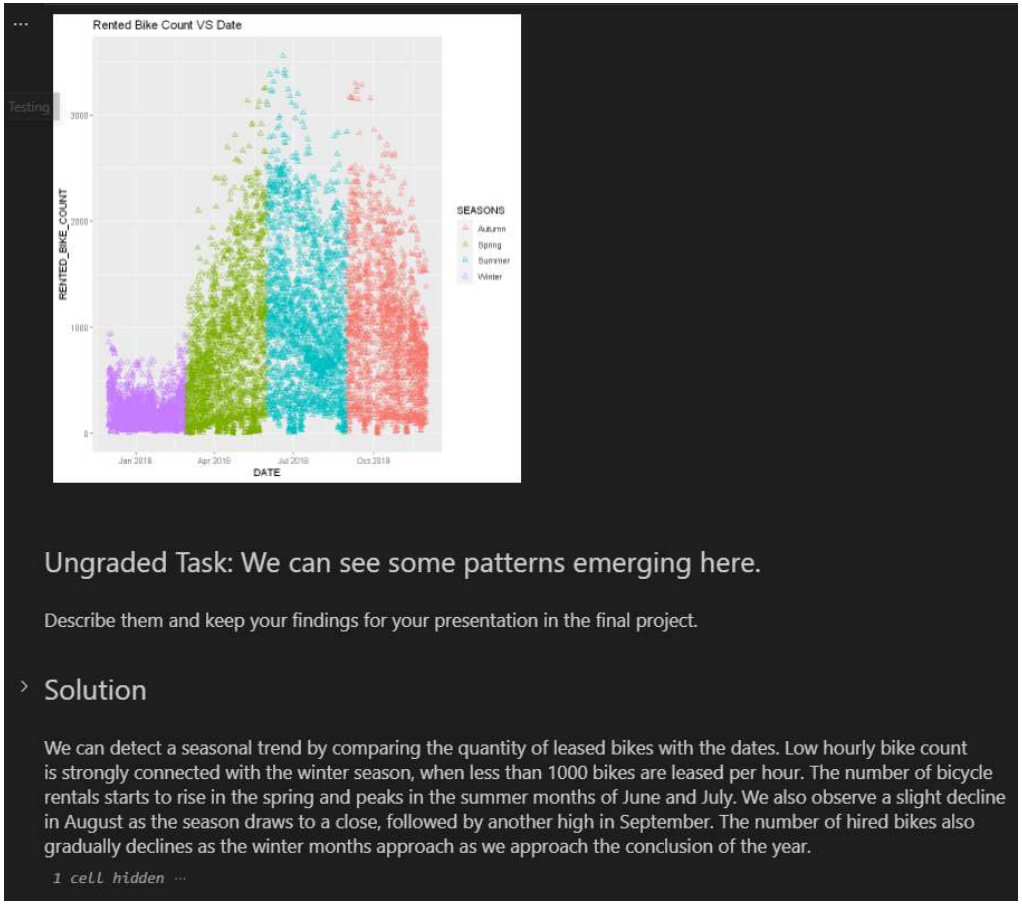
COUNTRY	CITY_ASCII	AVAILABLE_BIKES	LAT	LNG	POPULATION
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>
China	Zhuzhou	20000	27.8407	113.1469	3855609
China	Xi'an	20000	34.2667	108.9000	7135000
China	Weifang	20000	36.7167	119.1000	9373000
China	Shanghai	19165	31.1667	121.4667	22120000
China	Beijing	16000	39.9050	116.3914	19433000
China	Ningbo	15000	29.8750	121.5492	7639000

EDA with Visualization

In addition to SQL for EDA, visualizations may be utilized to get graphical insights into datasets. It is a straightforward categorization strategy based on visual cues. Visualizations also aid in the depiction of descriptive statistics, as well as outliers and correlations found in the dataset.

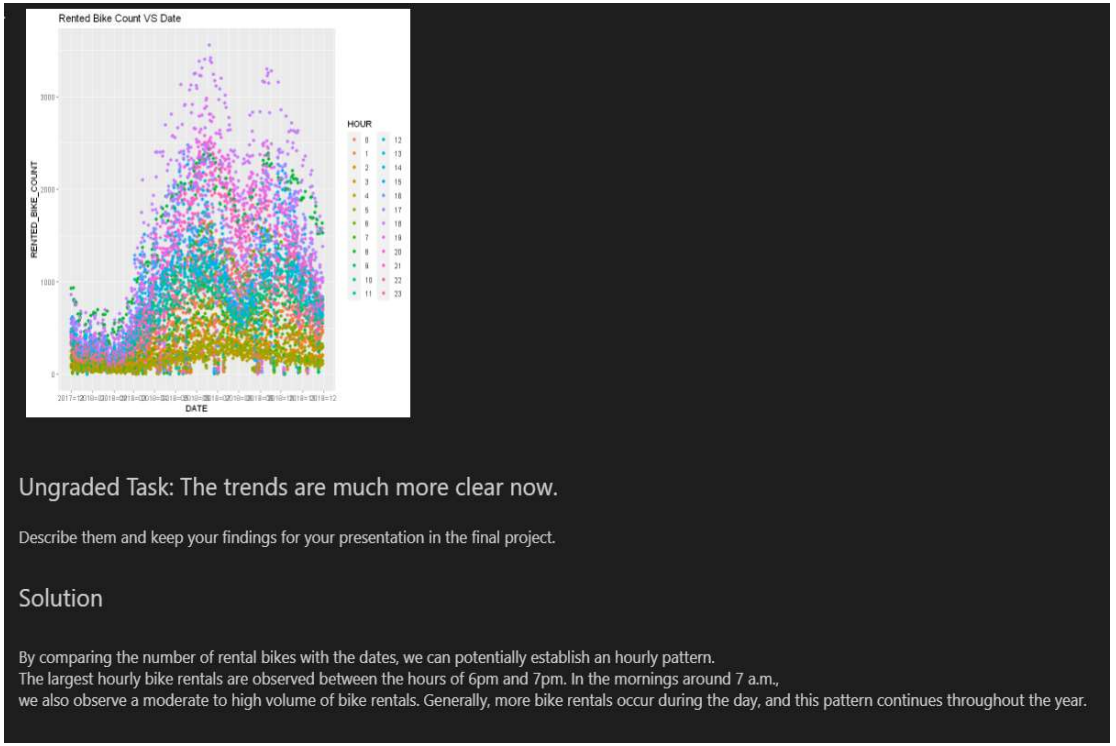
Bike rental vs. Date

We can detect a seasonal trend by comparing the quantity of leased bikes with the dates. Low hourly bike count is strongly connected with the winter season, when less than 1000 bikes are leased per hour. The number of bicycle rentals starts to rise in the spring and peaks in the summer months of June and July. We also observe a slight decline in August as the season draws to a close, followed by another high in September. The number of hired bikes also gradually declines as the winter months approach as we approach the conclusion of the year.



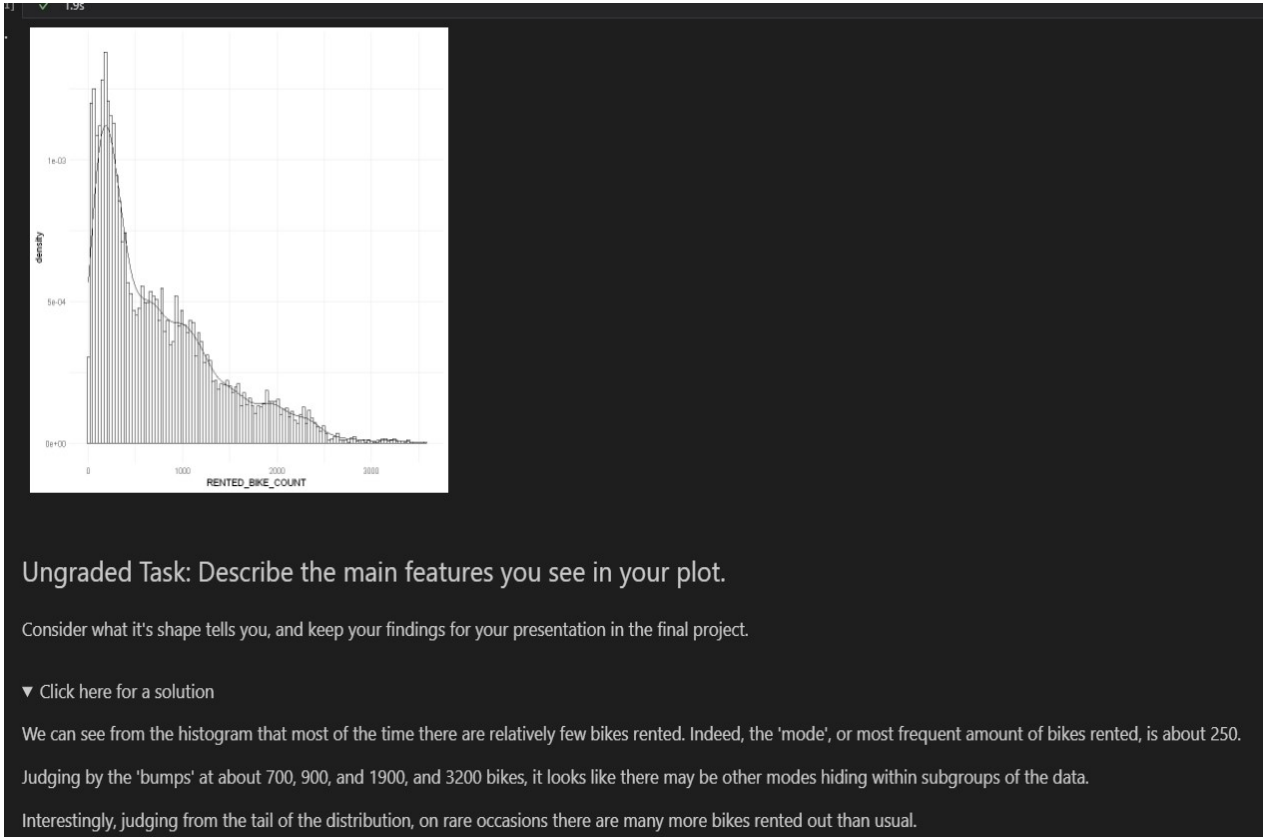
Bike rental vs. Datetime

By comparing the number of rental bikes with the dates, we can potentially establish an hourly pattern. The largest hourly bike rentals are observed between the hours of 6pm and 7pm. In the mornings around 7 a.m., we also observe a moderate to high volume of bike rentals. Generally, more bike rentals occur during the day, and this pattern continues throughout the year.



Bike rental histogram

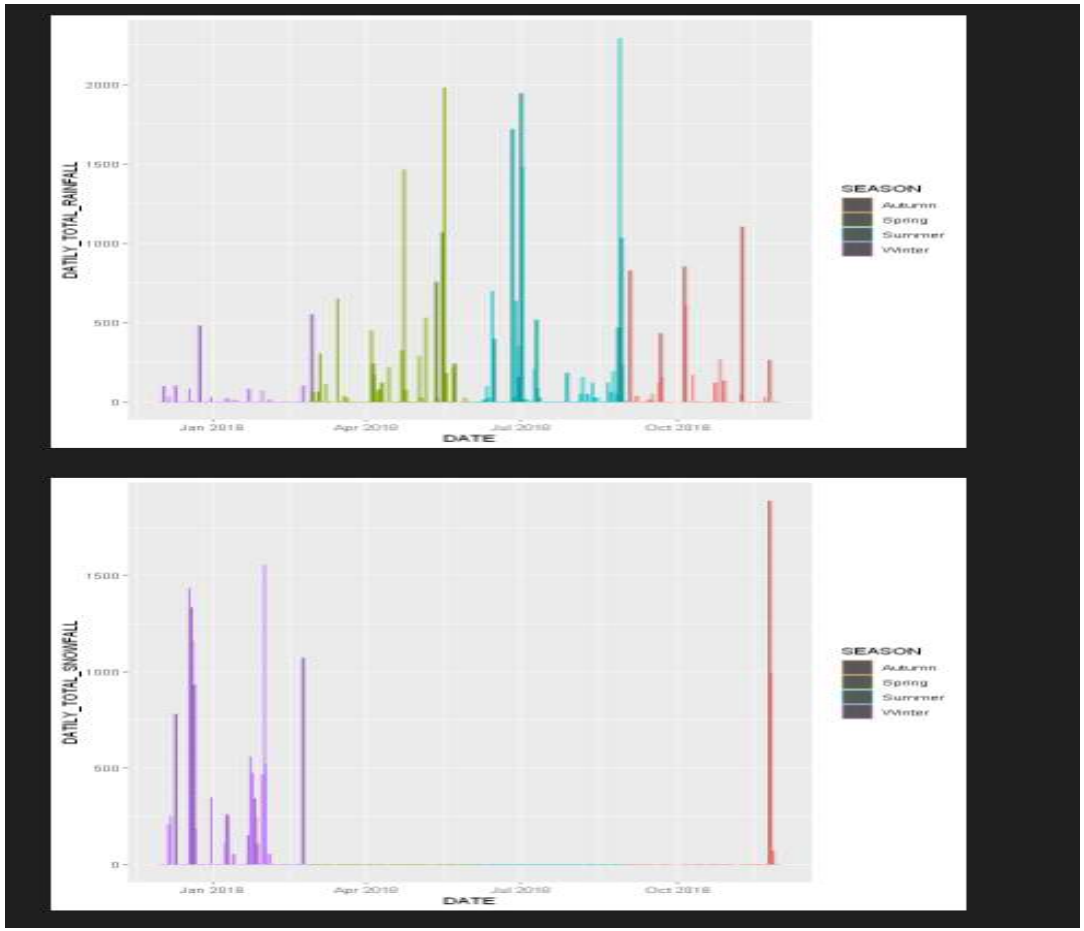
The kernel density plot shows that the dataset's distribution is skewed to the left. It also demonstrates that rental bike counts ranging from 0 to 500 occur most frequently in the sample. The 'mode,' or most often rented number of bikes, is about 250. Because the density curve is tilted to the left, the mean is smaller than the median.



Daily total rainfall and snowfall

This bar chart shows that peak rainfall occurs around September with little or no rainfall during the months of winter like December and January.

This bar chart shows snowfall only occurs during winter between December and January and ends in March.

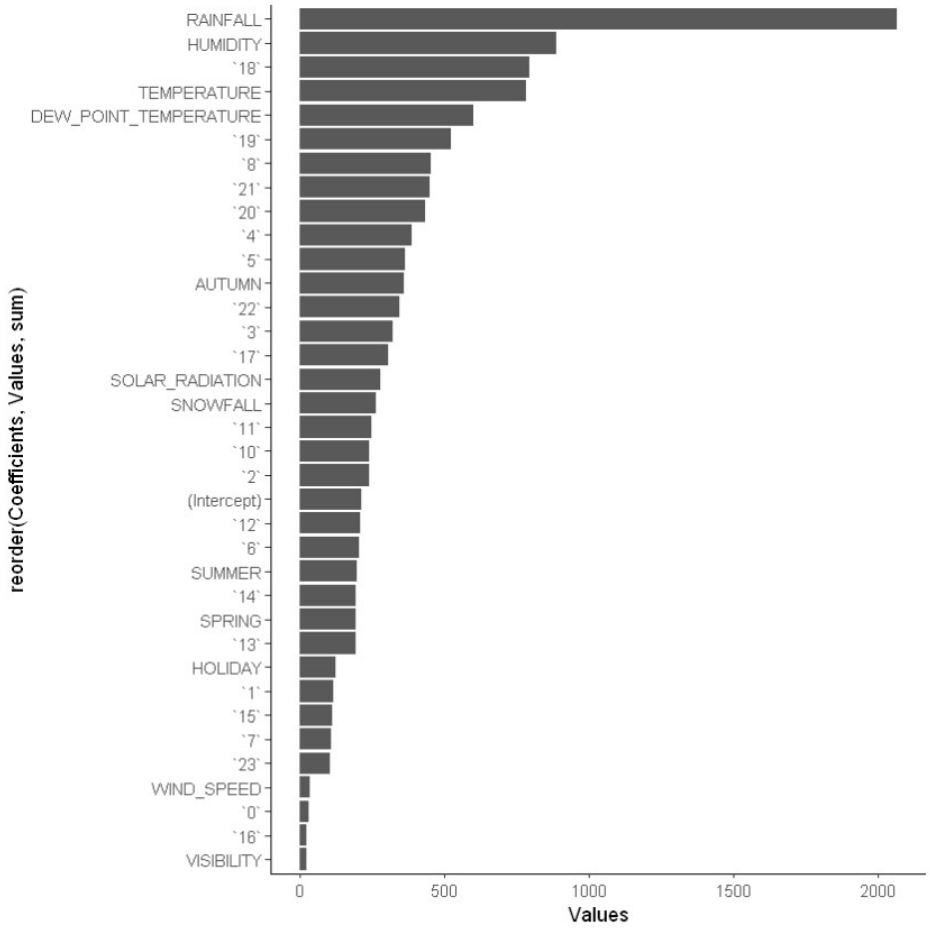


Predictive analysis

This phase of the project entailed developing and refining linear models to forecast the hourly number of rental bikes given a certain set of meteorological and temporal parameters.

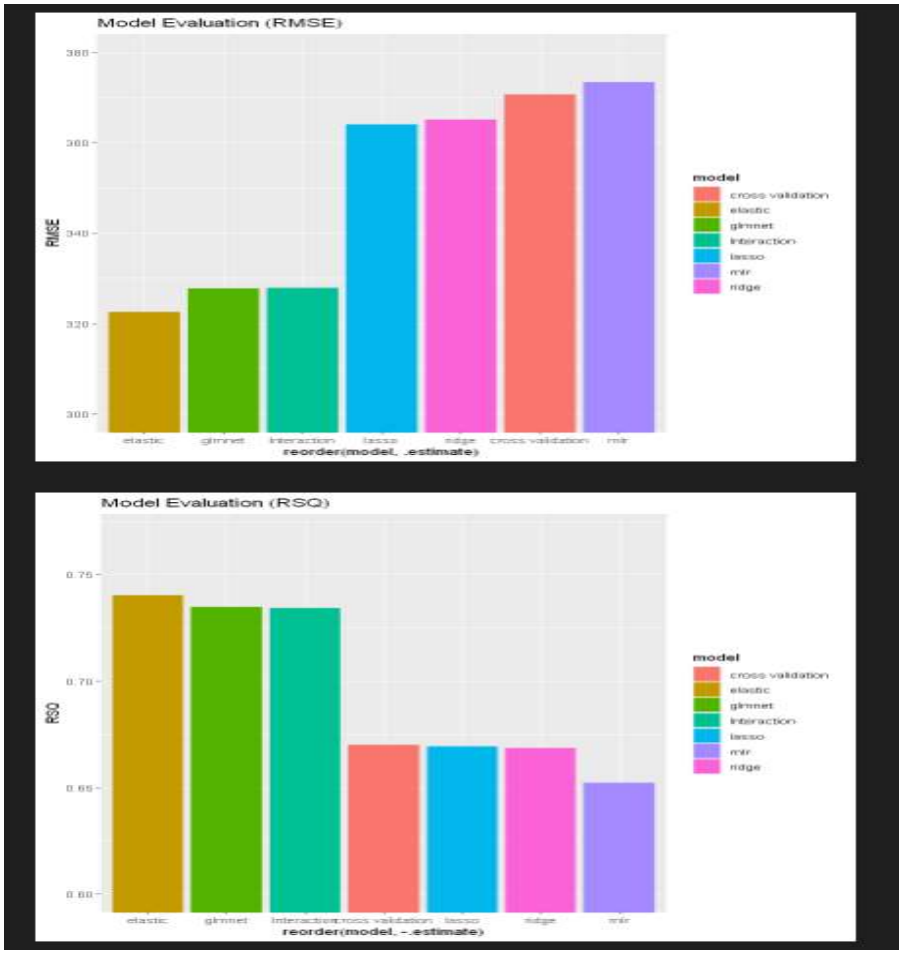
Ranked coefficients

According to the bar chart on the right, the top weather-related variables with the highest coefficients are rainfall, humidity, and temperature. Another significant variable is the hour of 6 p.m., which has a large magnitude as well. However, it should be emphasized that many weather-related factors, such as rainfall and temperature, are correlated, and so not all weather variables are suitable for predicting bike rentals. Furthermore, rainfall and snowfall are seasonally dependent and hence unsuitable for our model. Finally, because bike rental services are offered all year, the variables 'functional day' and 'holiday' were not included in the final model.



Model evaluation

After taking into account all of the variables, I developed several models using polynomial terms, interaction terms, and regularizations. The Root Mean Squared Error (RMSE) and R Squared (RSQ) metrics were used to assess the performance of each model. The bar charts on the right show the RMSE and RSQ performance of all models.



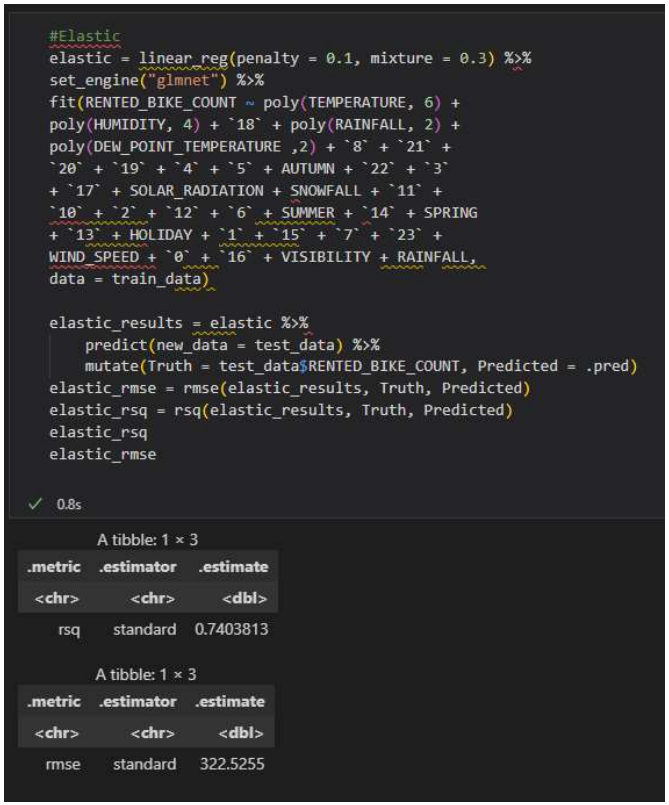
Find the best performing model

By comparing all models, it was deduced that overall, the elastic net regularized model performed best as it has the lowest RMSE (322.5255) and the highest RSQ(0.7403813).

Its model formula was :

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

The screenshot on the right highlights the model's performance



```
#Elastic
elastic = linear_reg(penalty = 0.1, mixture = 0.3) %>%
set_engine("glmnet") %>%
fit(RENTED_BIKE_COUNT ~ poly(TEMPERATURE, 6) +
poly(HUMIDITY, 4) + `18` + poly(RAINFALL, 2) +
poly(DEW_POINT_TEMPERATURE ,2) + `8` + `21` +
`20` + `19` + `4` + `5` + AUTUMN + `22` + `3`
+ `17` + SOLAR_RADIATION + SNOWFALL + `11` +
`10` + `2` + `12` + `6` + SUMMER + `14` + SPRING
+ `13` + HOLIDAY + `1` + `15` + `7` + `23` +
WIND_SPEED + `0` + `16` + VISIBILITY + RAINFALL,
data = train_data)

elastic_results = elastic %>%
    predict(new_data = test_data) %>%
    mutate(Truth = test_data$RENTED_BIKE_COUNT, Predicted = .pred)
elastic_rmse = rmse(elastic_results, Truth, Predicted)
elastic_rsq = rsq(elastic_results, Truth, Predicted)
elastic_rsq
elastic_rmse
```

✓ 0.8s

A tibble: 1 × 3

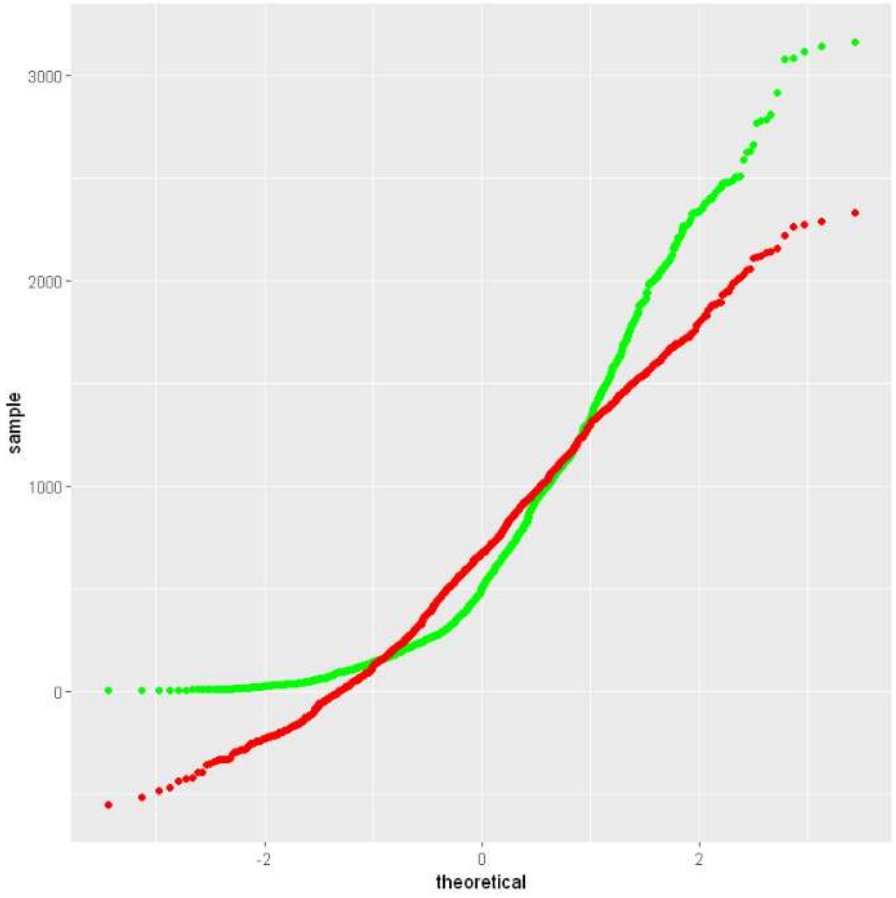
.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rsq	standard	0.7403813

A tibble: 1 × 3

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	322.5255

Q-Q plot of the best model

Q-Q plot of the best model's test results vs the truths

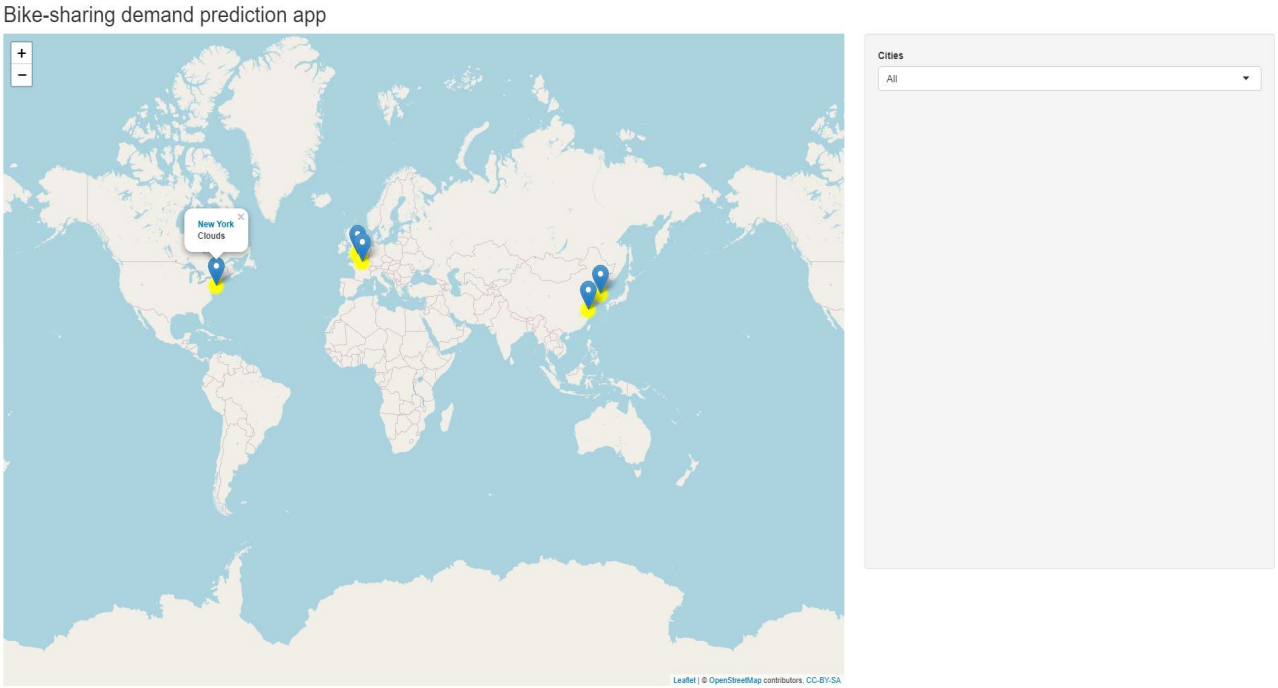


Dashboard

The goal of the last part of this project was to create an interactive R Shiny dashboard that could visualize weather prediction data and estimated hourly bike-sharing demand for the following cities: New York (USA), Paris (France), Suzhou (China), and London (UK) (UK). These cities feature bike rental sizes that are comparable to Seoul.

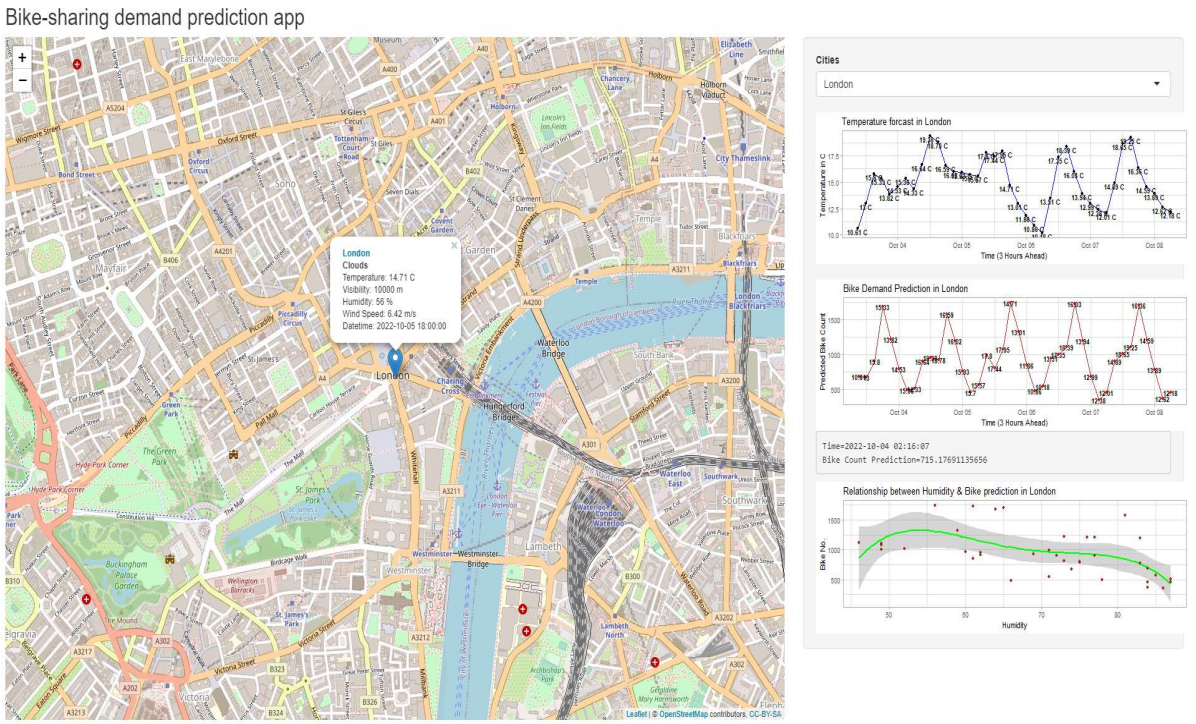
Bike-sharing Demand Dashboard Overview

The screenshot on the right depicts a basic max bike forecast overview map as well as a dropdown list for selecting a specific city.



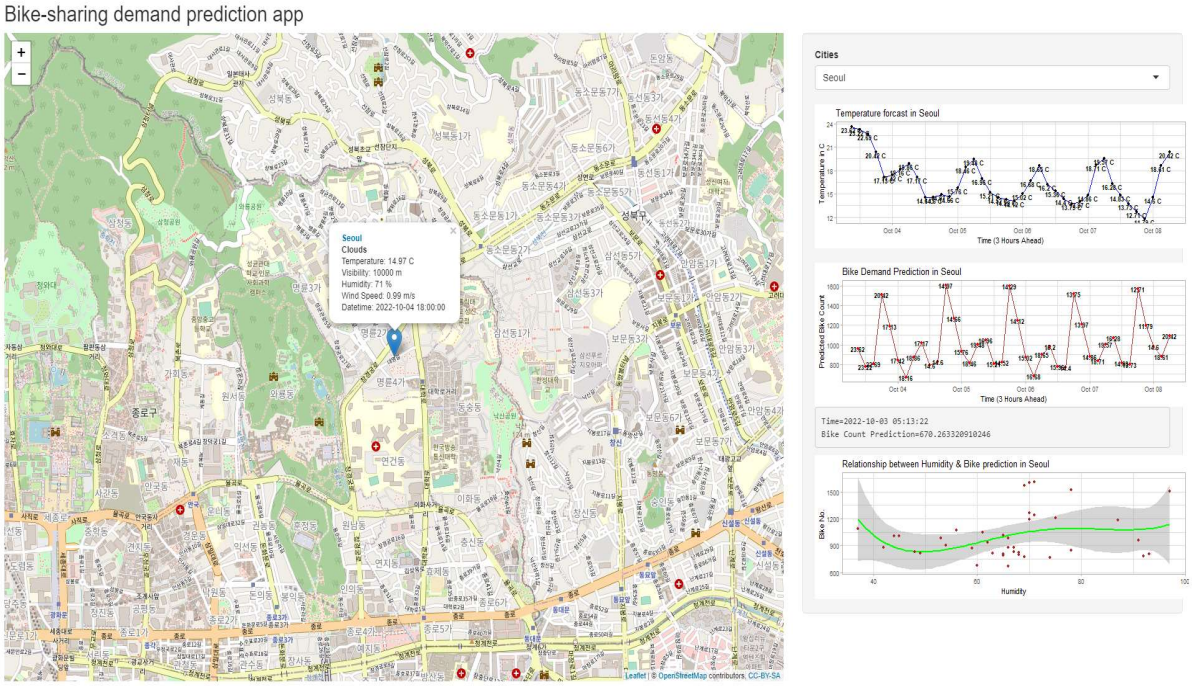
Bike-sharing Demand Overview for London

When the city of London is selected, the screenshot on the right displays an overview map and basic max bike estimate. It also displays the temperature forecast for London, as well as the relationship between humidity and bike prediction.



Bike-sharing Demand Overview for Seoul

The screenshot on the right shows an overview map and basic max bike prediction when the city of Seoul is selected. It also shows the temperature forecast as well as the relationship between humidity and bike prediction for Seoul.

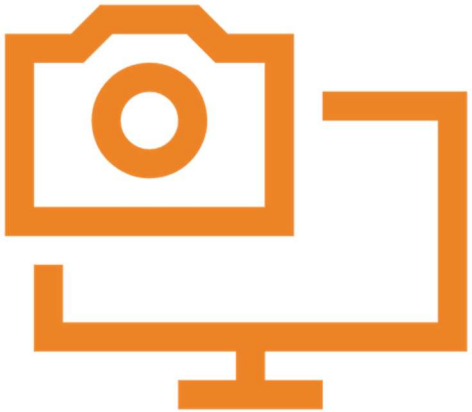


CONCLUSION

Conclusion based on our study

- The summer months of June and July saw the biggest demand for bike sharing.
- During the day, the peak demand for rental bikes is between 6 and 7 p.m.
- During the winter, there is virtually little demand for bicycles.
- The quantity of bikes hired throughout each hour of any given day is heavily influenced by temperature and humidity.

APPENDIX



- Appendix 1.0: Web scraping a Global Bike-Sharing Systems Wiki Page.
- Appendix 2.0: Obtaining a 5-day weather forecast for a list of cities using OpenWeather API.
- Appendix 3.0: Data wrangling using ‘stringr’ and regular expressions.
- Appendix 4.0: Data wrangling using ‘dplyr’.
- Appendix 5.0: SQL Queries.
- Appendix 5.1: SQL Queries cont’d.
- Appendix 6.0: EDA with visualizations.
- Appendix 6.1: EDA with visualizations cont’d.

```r
# Convert the bike-sharing system table into a dataframe
bike_sharing_df = html_table(table_nodes[[2]], fill = TRUE)
head(bike_sharing_df)
```

A tibble: 6 × 10

Country	City	Name	System	Operator	Launched	Discontinued	Stations	Bicycles	Daily ridership
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
Albania	Tirana[5]	Ecovolis			March 2011		8	200	
Argentina	Buenos Aires[6][7]	Ecobici	Serttel Brasil[8]	Bike In Baires Consortium.[9]	2010		400	4000	21917
Argentina	Mendoza[10]	Metrobici			2014		2	40	
Argentina	Rosario	Mi Bici Tu Bici[11]			2 December 2015		47	480	
Argentina	San Lorenzo, Santa Fe	Biciudad	Biciudad		27 November 2016		8	80	
Australia	Melbourne[12]	Melbourne Bike Share	PBSC & 8D	Motivate	June 2010	30 November 2019[13]	53	676	

Summarize the bike sharing system data frame

```r
# Summarize the dataframe

summary(bike_sharing_df
)
```

```r
# Check if need to install rvest` library
require("rvest")

library(rvest)
```

```
Loading required package: rvest
```

TASK: Extract bike sharing systems HTML table

TODO: Get the root HTML node

```r
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)
table_nodes = html_nodes(root_node, "table")
table_nodes
for (tables in table_nodes)
print(tables)
```

Appendix 2.0: Obtaining a 5-day weather forecast for a list of cities using OpenWeather API

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

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

[13]

A data.frame: 6 × 12

	city	weather	visibility	temp	temp_min	temp_max	pressure	humidity	wind_speed	wind_deg	forecast_datetime	season
	<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<int>	<chr>	<chr>
1	Seoul	Clouds	10000	25.81	24.10	25.81	1006	52	1.36	299	2022-08-23 12:00:00	Summer
2	Seoul	Clouds	10000	24.12	22.85	24.12	1007	61	0.79	89	2022-08-23 15:00:00	Summer
3	Seoul	Clouds	10000	21.98	21.98	21.98	1007	73	0.77	99	2022-08-23 18:00:00	Summer
4	Seoul	Clouds	10000	21.64	21.64	21.64	1008	78	0.91	96	2022-08-23 21:00:00	Summer
5	Seoul	Clouds	10000	23.04	23.04	23.04	1009	75	1.31	90	2022-08-24 00:00:00	Summer
6	Seoul	Rain	10000	24.99	24.99	24.99	1009	70	2.26	94	2022-08-24 03:00:00	Summer

Now assign the values in the `json_result` list into different vectors

```
# $weather is also a list with one element, its $main element indicates the we
weather <- c(weather, json_result$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)
```

[13]

```
# 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'
        api_key = "[redacted]"
        # Create query parameters
        forecast_query <- list(q = city_name, appid = api_key, units="metric")
        # Make HTTP GET call for the given city
        response <- GET(forecast_url, query=forecast_query)
        #print(response)
        json_list <- content(response, as="parsed")
        #print(json_list$list)
        # Note that the 5-day forecast JSON result is a list of lists. You can print the reponse to check the results
        results <- json_list$list
        #print(results)
        # Loop the json result
        for(result in results) {
            city <- c(city, city_name)
            # Weather column, rainy or cloudy, etc
            weather <- c(weather, result$weather[[1]]$main)
            # Sky visibility column
            visibility <- c(visibility, result$visibility)
            # Current temperature column
            temp <- c(temp, result$main$temp)
            # Max temperature column
            temp_min <- c(temp_min, result$main$temp_min)
            # Min temperature column
            temp_max <- c(temp_max, result$main$temp_max)
            # Pressure column
            pressure <- c(pressure, result$main$pressure)
            # Humidity column
            humidity <- c(humidity, result$main$humidity)
            # Wind speed column
            wind_speed <- c(wind_speed, result$wind$speed)
            # Wind direction column
            wind_deg <- c(wind_deg, result$wind$deg)
            # Forecast timestamp
            forecast_datetime <- c(forecast_datetime, result$dt_txt)
            # 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(season, "Summer")

        }
```

Appendix 3.0: Data wrangling using ‘stringr’ and regular expressions

TASK: Remove undesired reference links using regular expressions

TODO: Write a custom function using `stringr::str_replace_all` to replace all reference links with an empty character for columns `CITY` and `SYSTEM`

```
# remove reference link
remove_ref <- function(strings) {
    ref_pattern <- "\\[[A-z0-9]+\\]"
    # Replace all matched substrings with a white space using str_replace_all()
    replace = str_replace_all(strings, pattern = ref_pattern, replacement = " ")
    # Trim the reslt if you want
    result = str_trim(replace, side = c("both"))
    return(result)
}
```

TODO: Use the `dplyr::mutate()` function to apply the `remove_ref` function to the `CITY` and `SYSTEM` columns

```
#
result = sub_bike_sharing_df %>% mutate(CITY=remove_ref(CITY), SYSTEM = remove_ref(SYSTEM))
result
```

A tibble: 520 × 4

COUNTRY	CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>	<chr>
Albania	Tirana	NA	200
Argentina	Buenos Aires	Serttel Brasil	4000
Argentina	Mendoza	NA	40

TASK: Extract the numeric value using regular expressions

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

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

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

```
# Use the mutate() function on the BICYCLES column

result = result %>% mutate(BICYCLES=extract_num(BICYCLES))
head(result)
```

A tibble: 6 × 4

COUNTRY	CITY	SYSTEM	BICYCLES
<chr>	<chr>	<chr>	<dbl>
Albania	Tirana	NA	200
Argentina	Buenos Aires	Serttel Brasil	4000

Appendix 4.0: Data wrangling using ‘dplyr’

```
# Using mutate() function to convert HOUR column into character type
hour_char = temp_replace_na %>%
mutate(HOUR = as.character(HOUR))
summary(hour_char)
```

SEASONS, HOLIDAY, FUNCTIONING_DAY, HOUR are all character columns now and are ready to be converted into in

For example, SEASONS has four categorical values: Spring, Summer, Autumn, Winter. We thus need to create four

So, given a data entry with the value Spring in the SEASONS column, the values for the four new columns Spring

Spring	Summer	Autumn	Winter
1	0	0	0

TODO: Convert SEASONS, HOLIDAY, FUNCTIONING_DAY, and HOUR columns into indicator columns.

Note that if FUNCTIONING_DAY only contains one categorical value after missing values removal, then you don't n

```
# Convert SEASONS, HOLIDAY, FUNCTIONING_DAY, and HOUR columns into indicator columns.
seasons_ind = hour_char %>%
    mutate(dummy = 1) %>%
    spread(
        key = SEASONS,
        value = dummy,
        fill = 0
    )

holiday_ind = seasons_ind %>%
    mutate(dummy = 1) %>%
    spread(
        key = HOLIDAY,
        value = dummy,
        fill = 0
    )

hour_ind = holiday_ind %>%
    mutate(dummy = 1) %>%
    spread(
        key = HOUR,
        value = dummy,
        fill = 0
```

TODO: Impute missing values for the TEMPERATURE column using its mean value.

```
[25]
# Calculate the summer average temperature
summer_mean = mean(drop_rows$TEMPERATURE[drop_rows$SEASONS == 'Summer'], na.rm = TRUE)
summer_mean
```

```
26.5877105143377
```

```
[33]
# Impute missing values for TEMPERATURE column with summer average temperature
temp_replace_na = drop_rows %>% replace_na(list(TEMPERATURE = summer_mean))
```

```
[39]
# Print the summary of the dataset again to make sure no missing values in all columns
sum(is.na(temp_replace_na))
summary(temp_replace_na)
```

```
0
```

TASK: Normalize data

Columns RENTED_BIKE_COUNT, TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, SNOWFALL are numerical variables/columns wi
values may adversely influence (bias) the predictive models and degrade model accuracy. Thus, we need to perform normalization on these numeric columns to transfer them into a similar ra

In this project, you are asked to use Min-max normalization:

Min-max rescales each value in a column by first subtracting the minimum value of the column from each value, and then divides the result by the difference between the maximum and mi
scaled such that the minimum becomes 0 and the maximum becomes 1.

$$x\_new = \frac{x\_old - x\_min}{x\_max - x\_min}$$

TODO: Apply min-max normalization on RENTED_BIKE_COUNT, TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, SNOWFALL

```
# Use the `mutate()` function to apply min-max normalization on columns
# `RENTED_BIKE_COUNT`, `TEMPERATURE`, `HUMIDITY`, `WIND_SPEED`, `VISIBILITY`, `DEW_POINT_TEMPERATURE`, `SOLAR_RADIATION`, `RAINFALL`, `SNOWFALL`
normalized = hour_ind %>%
                    mutate(
                        RENTED_BIKE_COUNT = (RENTED_BIKE_COUNT - min(RENTED_BIKE_COUNT)) / (max(RENTED_BIKE_COUNT) - min(RENTED_BIKE_COUNT)),
                        TEMPERATURE = (TEMPERATURE - min(TEMPERATURE)) / (max(TEMPERATURE) - min(TEMPERATURE)),
                        HUMIDITY = (HUMIDITY - min(HUMIDITY)) / (max(HUMIDITY) - min(HUMIDITY)),
                        WIND_SPEED = (WIND_SPEED - min(WIND_SPEED)) / (max(WIND_SPEED) -    min(WIND_SPEED)),
                        VISIBILITY = (VISIBILITY - min(VISIBILITY)) / (max(VISIBILITY) - min(VISIBILITY)),
                        DEW_POINT_TEMPERATURE = (DEW_POINT_TEMPERATURE - min(DEW_POINT_TEMPERATURE)) / (max(DEW_POINT_TEMPERATURE) - min(DEW_POINT_TEMPERATURE)),
                        SOLAR_RADIATION = (SOLAR_RADIATION - min(SOLAR_RADIATION)) / (max(SOLAR_RADIATION) - min(SOLAR_RADIATION)),
                        RAINFALL = (RAINFALL - min(RAINFALL)) / (max(RAINFALL) - min(RAINFALL)),
                        SNOWFALL = (SNOWFALL - min(SNOWFALL)) / (max(SNOWFALL) - min(SNOWFALL)))

head(normalized)
```

Appendix 5.0: SQL Queries

Task 1 - Record Count

Determine how many records are in the seoul_bike_sharing dataset.

Solution 1

```
# provide your solution here

dbGetQuery(conn, 'SELECT COUNT(*) FROM SEOUL_BIKE')
```

A data.frame: 1 × 1

COUNT(*)
<int>
8465

Task 2 - Operational Hours

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

Solution 2

```
# provide your solution here

dbGetQuery(conn, 'SELECT COUNT(HOUR) FROM SEOUL_BIKE WHERE RENTED_BIKE_COUNT != 0')
```

✓ 0.3s

A data.frame: 1 × 1

COUNT(HOUR)
<int>
8465

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
dbGetQuery(conn, 'SELECT * FROM WEATHER where CITY = "Seoul" order by FORECAST_DATETIME DESC LIMIT 1')
```

A data.frame: 1 × 12

CITY	WEATHER	VISIBILITY	TEMP	TEMP_MIN	TEMP_MAX	PRESSURE	HUMIDITY	WIND_SPEED	WIND_DEG	SEASON	FORECAST_DATETIME
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>	<int>	<chr>	<chr>
Seoul	Clouds	10000	25.1	25.1	25.1	1020	17	1.54	262	Spring	2021-04-21 09:00:00

Task 4 - Seasons

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

Solution 4

```
# provide your solution here
dbGetQuery(conn, 'select distinct(SEASONS) from SEOUL_BIKE')
```

A data.frame: 4 × 1

SEASONS
<chr>
Winter
Spring
Summer
Autumn

Task 5 - Date Range

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

Solution 5

```
# provide your solution here
dbGetQuery(conn, 'select min(DATE) AS "First Date", max(DATE) AS "Last Date" from SEOUL_BIKE')
```

✓ 0.1s

A data.frame: 1 × 2

First Date	Last Date
<chr>	<chr>
01/01/2018	31/12/2017

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

```
# provide your solution here
dbGetQuery(conn, 'select DATE, HOUR,
                  RENTED_BIKE_COUNT from SEOUL_BIKE
                  where RENTED_BIKE_COUNT = (select max(RENTED_BIKE_COUNT) from SEOUL_BIKE)')
```

A data.frame: 1 × 3

DATE	HOUR	RENTED_BIKE_COUNT
<chr>	<int>	<int>
19/06/2018	18	3556

Task 7 - Hourly popularity and temperature by season

Determine the average hourly temperature and the average number of bike rentals

Solution 7

```
# provide your solution here
dbGetQuery(conn, 'select avg(TEMPERATURE) as "Average Hourly Temp",
                  avg(RENTED_BIKE_COUNT) as "Average Bike Rentals/Hour",
                  HOUR,
                  SEASONS
                  from SEOUL_BIKE
                  group by HOUR, SEASONS
                  order by avg(RENTED_BIKE_COUNT) DESC
                  limit 10')
```

A data.frame: 10 × 4

Average Hourly Temp	Average Bike Rentals/Hour	HOUR	SEASONS
<dbl>	<dbl>	<int>	<chr>
29.38791	2135.141	18	Summer
16.03185	1983.333	18	Autumn
28.27378	1889.250	19	Summer
27.06630	1801.924	20	Summer
26.27826	1754.065	21	Summer
15.97222	1689.311	18	Spring
25.69891	1567.870	22	Summer
17.27778	1562.877	17	Autumn
30.07691	1526.293	17	Summer
15.06346	1515.568	19	Autumn

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.

Hint : Use the SQRT(AVG(col*col) - AVG(col)*AVG(col)) function where col refers to your column name for finding the standard deviation

Solution 8

```
# provide your solution here
dbGetQuery(conn, 'select
                  SEASONS,
                  avg(RENTED_BIKE_COUNT) AS AVERAGE,
                  min(RENTED_BIKE_COUNT) AS MINIMUM,
                  max(RENTED_BIKE_COUNT) AS MAXIMUM,
                  sqrt(avg(RENTED_BIKE_COUNT * RENTED_BIKE_COUNT) - avg(RENTED_BIKE_COUNT) * avg(RENTED_BIKE_COUNT)) AS STD_DEV
                  FROM SEOUL_BIKE
                  group by SEASONS')
```

A data.frame: 4 × 5

SEASONS	AVERAGE	MINIMUM	MAXIMUM	STD_DEV
<chr>	<dbl>	<int>	<int>	<dbl>
Autumn	924.1105	2	3298	617.3885
Spring	746.2542	2	3251	618.5247
Summer	1034.0734	9	3556	690.0884
Winter	225.5412	3	937	150.3374

Appendix 5.1: SQL Queries cont'd

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

```
# provide your solution here
dbGetQuery(conn, 'select
                  AVG(RENTED_BIKE_COUNT) AS AVG_BIKES_RENTED,
                  avg(TEMPERATURE) AS AVG_TEMP,
                  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_TEMP,
                  AVG(SOLAR_RADIATION) AS AVG_SOLAR_RADIATION,
                  AVG(RAINFALL) AS AVG_RAINFALL,
                  AVG(SNOWFALL) AS AVG_SNOWFALL,
                  SEASONS
                  FROM SEOUL_BIKE
                  GROUP BY SEASONS
                  ORDER BY AVG_BIKES_RENTED DESC')
```

A data.frame: 4 × 10

AVG_BIKES_RENTED	AVG_TEMP	AVG_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT_TEMP	AVG_SOLAR_RADIATION	AVG_RAINFALL	AVG_SNOWFALL	SEASONS
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1034.0734	26.587711	64.98143	1.609420	1501.745	18.750136	0.7612545	0.25348732	0.00000000	Summer
924.1105	13.821580	59.04491	1.492101	1558.174	5.150594	0.5227827	0.11765617	0.06350026	Autumn
746.2542	13.021685	58.75833	1.857778	1240.912	4.091389	0.6803009	0.18694444	0.00000000	Spring
225.5412	-2.540463	49.74491	1.922685	1445.987	-12.416667	0.2981806	0.03282407	0.24750000	Winter

Task 10 - Total Bike Count and City Info for Seoul

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

dbGetQuery(conn, 'select
                  B.COUNTRY, CITY_ASCII, SUM(BICYCLES) AS AVAILABLE_BIKES, LAT, LNG, POPULATION
                  FROM BIKE_SHARING B, WORLD_CITIES W
                  WHERE W.CITY_ASCII = B.CITY AND CITY_ASCII = "Seoul"')
```

0.2s

A data.frame: 1 × 6

COUNTRY	CITY_ASCII	AVAILABLE_BIKES	LAT	LNG	POPULATION
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>
South Korea	Seoul	20000	37.5833	127	21794000

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

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

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

Solution 11

```
# provide your solution here
dbGetQuery(conn, 'select
                  B.COUNTRY, CITY_ASCII, BICYCLES AS AVAILABLE_BIKES, LAT, LNG, POPULATION
                  FROM BIKE_SHARING B, WORLD_CITIES W
                  WHERE W.CITY_ASCII = B.CITY AND AVAILABLE_BIKES BETWEEN 15000 AND 20000
                  GROUP BY CITY_ASCII
                  order by AVAILABLE_BIKES DESC')
```

A data.frame: 7 × 6

COUNTRY	CITY_ASCII	AVAILABLE_BIKES	LAT	LNG	POPULATION
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>
China	Zhuzhou	20000	27.8407	113.1469	3855609
China	Xi'an	20000	34.2667	108.9000	7135000
China	Weifang	20000	36.7167	119.1000	9373000
South Korea	Seoul	20000	37.5833	127.0000	21794000
China	Shanghai	19165	31.1667	121.4667	22120000
China	Beijing	16000	39.9050	116.3914	19433000
China	Ningbo	15000	29.8750	121.5492	7639000

Appendix 6.0: EDA with visualizations

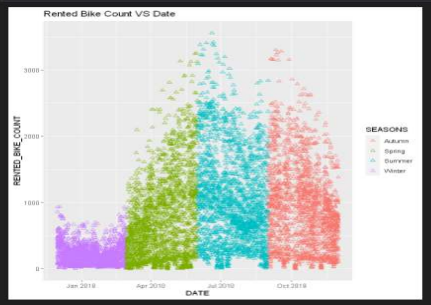
Task 10 - Create a scatter plot of RENTED_BIKE_COUNT vs DATE.

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, mapping = aes(y = RENTED_BIKE_COUNT, x = DATE, color = SEASONS), alpha = 0.3) +
geom_point(shape = 2) +
ggtitle("Rented Bike Count VS Date")
```

[32]

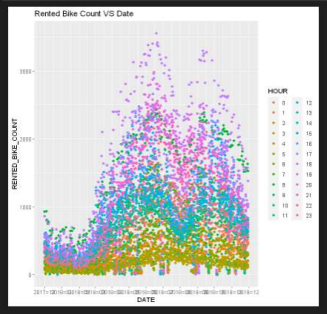


Task 11 - Create the same plot of the RENTED_BIKE_COUNT time series, but now add HOURS as the colour.

Solution 11

```
# provide your solution here
ggplot(seoul_bike_sharing, mapping = aes(y = RENTED_BIKE_COUNT, x = DATE, color = HOUR)) +
geom_point() +
scale_x_date(breaks="month",date_labels="%Y-%m") +
ggtitle("Rented Bike Count VS Date")
```

[34]



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

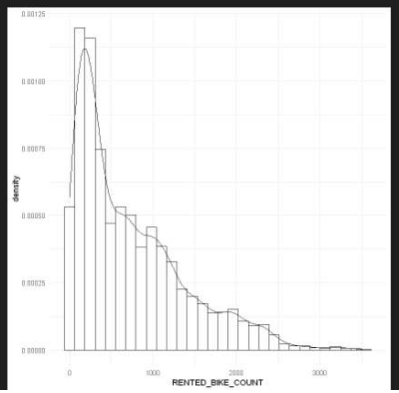
▶ Click here for another hint

Solution 12

```
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = RENTED_BIKE_COUNT)) +
geom_histogram(aes(y = ..density..), color = 1, fill = '#ffffff') +
geom_density(alpha = 0.9) +
theme_minimal()
```

[19]

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



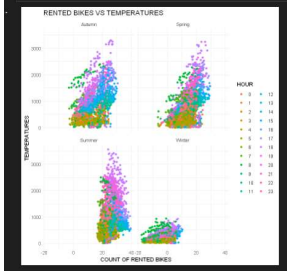
Appendix 6.1: EDA with visualizations cont'd

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(y = RENTED_BIKE_COUNT, x = TEMPERATURE, color = HOUR)) +
labs(x = "COUNT OF RENTED BIKES",
     y = "TEMPERATURES",
     title = "RENTED BIKES VS TEMPERATURES") +
facet_wrap(~SEASONS) +
theme_minimal()
```



Outliers (boxplot)

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) +
geom_boxplot(aes(y = RENTED_BIKE_COUNT, x = HOUR, fill = HOUR)) +
labs(x = "COUNT OF RENTED BIKES",
     y = "HOUR OF THE DAY",
     title = "RENTED BIKES VS HOUR OF THE DAY") +
facet_wrap(~SEASONS) +
theme_minimal()
```

[41]