Project1_Group5

Sakshi, Anshita, Shubham, Unnati

03/01/2022

Including all the required libraries

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(dlookr)
```

```
## Imported Arial Narrow fonts.
```

```
##
## Attaching package: 'dlookr'
```

```
## The following object is masked from 'package:base':
##
##     transform
```

```
library(ggplot2)
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.1.2
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:lubridate':
##
##     %--%, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(reshape)

##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename

## The following object is masked from 'package:lubridate':
##
##     stamp

library(zoo)

## Warning: package 'zoo' was built under R version 4.1.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(ggmap)

## Warning: package 'ggmap' was built under R version 4.1.2

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.1 --
```

```
## v tibble  3.1.4     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'stringr' was built under R version 4.1.2

## -- Conflicts ---------------------------------------- tidyverse_conflict
s() --
## x igraph::%--%()           masks lubridate::%--%()
## x lubridate::as.difftime() masks base::as.difftime()
## x tibble::as_data_frame()  masks igraph::as_data_frame(), dplyr::as_data_f
rame()
## x purrr::compose()         masks igraph::compose()
## x tidyr::crossing()        masks igraph::crossing()
## x lubridate::date()        masks base::date()
## x tidyr::expand()          masks reshape::expand()
## x tidyr::extract()         masks dlookr::extract()
## x dplyr::filter()          masks stats::filter()
## x igraph::groups()         masks dplyr::groups()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x reshape::rename()        masks dplyr::rename()
## x lubridate::setdiff()     masks base::setdiff()
## x purrr::simplify()        masks igraph::simplify()
## x reshape::stamp()         masks lubridate::stamp()
## x igraph::union()          masks lubridate::union(), base::union()

library(ggraph)

## Warning: package 'ggraph' was built under R version 4.1.2

library(tidygraph)

## Warning: package 'tidygraph' was built under R version 4.1.2

##
## Attaching package: 'tidygraph'

## The following object is masked from 'package:reshape':
##
##     rename

## The following object is masked from 'package:igraph':
##
##     groups

## The following object is masked from 'package:stats':
##
##     filter

library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.1.2

library(sf)

## Warning: package 'sf' was built under R version 4.1.2

## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1; sf_use_s2() is TRUE

library("wordcloud2")

## Warning: package 'wordcloud2' was built under R version 4.1.2

library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.1.2

## Loading required package: RColorBrewer

library(webshot)
library(plotly)

## Warning: package 'plotly' was built under R version 4.1.2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggmap':
##
##     wind

## The following object is masked from 'package:reshape':
##
##     rename

## The following object is masked from 'package:igraph':
##
##     groups

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library(forecast)

## Warning: package 'forecast' was built under R version 4.1.2
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'forecast'

## The following object is masked from 'package:ggpubr':
##
##     gghistogram

library(knitr)

#To display map in mac
webshot::install_phantomjs()

## It seems that the version of `phantomjs` installed is greater than or equa
l to the requested version.To install the requested version or downgrade to a
nother version, use `force = TRUE`.

#To display map in Windows
install_phantomjs(version = "2.1.1",
  baseURL = "https://github.com/wch/webshot/releases/download/v0.3.1/",
  force = FALSE)

## It seems that the version of `phantomjs` installed is greater than or equa
l to the requested version.To install the requested version or downgrade to a
nother version, use `force = TRUE`.
```

Loading dataset

```
df_bike_sharing <-  read.csv("hour.csv", header = TRUE, sep = ",")
df_bluebikes_edited <- read.csv("Bluebikes_edited.csv", header = TRUE,
                                strip.white = TRUE, sep = ",")


View(df_bike_sharing)
View(df_bluebikes_edited)
```

Renaming columns to incorporate sensible column names

```
col_names <- c("ID", "Date", "Season", "Year", "Month", "Hour", "Holiday", "W
eekday", "Working_Day",
       "Weather_Situation", "Norm_Temp", "Norm_Feels_Temp", "Norm_Humidity",
"Norm_Windspeed",
       "Users_Unregistered", "Users_Registered", "Users_Total" )
colnames(df_bike_sharing) <- col_names
```

Number of instances and attributes

```
dim(df_bike_sharing)

## [1] 17379    17
```

Checking the datatypes for the columns

```
sapply(df_bike_sharing, class)
```

```
##                  ID                Date              Season                Yea
r
##           "integer"         "character"           "integer"           "integer
"
##              Month                Hour             Holiday             Weekda
y
##           "integer"           "integer"           "integer"           "integer
"
##         Working_Day   Weather_Situation           Norm_Temp     Norm_Feels_Tem
p
##           "integer"           "integer"           "numeric"           "numeric
"
##       Norm_Humidity     Norm_Windspeed Users_Unregistered   Users_Registere
d
##           "numeric"           "numeric"           "integer"           "integer
"
##         Users_Total
##           "integer"
```

Dataset summary

```
summary(df_bike_sharing)
```

```
##        ID            Date                Season           Year
##  Min.   :    1   Length:17379       Min.   :1.000   Min.   :0.0000
##  1st Qu.: 4346   Class :character   1st Qu.:2.000   1st Qu.:0.0000
##  Median : 8690   Mode  :character   Median :3.000   Median :1.0000
##  Mean   : 8690                      Mean   :2.502   Mean   :0.5026
##  3rd Qu.:13034                      3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :17379                      Max.   :4.000   Max.   :1.0000
##      Month             Hour           Holiday           Weekday
##  Min.   : 1.000   Min.   : 0.00   Min.   :0.00000   Min.   :0.000
##  1st Qu.: 4.000   1st Qu.: 6.00   1st Qu.:0.00000   1st Qu.:1.000
##  Median : 7.000   Median :12.00   Median :0.00000   Median :3.000
##  Mean   : 6.538   Mean   :11.55   Mean   :0.02877   Mean   :3.004
##  3rd Qu.:10.000   3rd Qu.:18.00   3rd Qu.:0.00000   3rd Qu.:5.000
##  Max.   :12.000   Max.   :23.00   Max.   :1.00000   Max.   :6.000
##   Working_Day     Weather_Situation   Norm_Temp     Norm_Feels_Temp
##  Min.   :0.0000   Min.   :1.000     Min.   :0.020   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:1.000     1st Qu.:0.340   1st Qu.:0.3333
##  Median :1.0000   Median :1.000     Median :0.500   Median :0.4848
##  Mean   :0.6827   Mean   :1.425     Mean   :0.497   Mean   :0.4758
##  3rd Qu.:1.0000   3rd Qu.:2.000     3rd Qu.:0.660   3rd Qu.:0.6212
##  Max.   :1.0000   Max.   :4.000     Max.   :1.000   Max.   :1.0000
##  Norm_Humidity    Norm_Windspeed   Users_Unregistered Users_Registered
##  Min.   :0.0000   Min.   :0.0000   Min.   :  0.00     Min.   :  0.0
##  1st Qu.:0.4800   1st Qu.:0.1045   1st Qu.:  4.00     1st Qu.: 34.0
##  Median :0.6300   Median :0.1940   Median : 17.00     Median :115.0
```

```
##  Mean   :0.6272   Mean   :0.1901   Mean   : 35.68     Mean   :153.8
##  3rd Qu.:0.7800   3rd Qu.:0.2537   3rd Qu.: 48.00     3rd Qu.:220.0
##  Max.   :1.0000   Max.   :0.8507   Max.   :367.00     Max.   :886.0
##   Users_Total
##  Min.   :  1.0
##  1st Qu.: 40.0
##  Median :142.0
##  Mean   :189.5
##  3rd Qu.:281.0
##  Max.   :977.0
```

————————- Data Cleaning and Preprocessing ————————-

Diagnose dataset to look for missing values

```
diagnose(df_bike_sharing) # no missing values found
```

```
## # A tibble: 17 x 6
##    variables          types     missing_count missing_percent unique_count
unique_rate
##    <chr>              <chr>             <int>           <dbl>        <int>
<dbl>
##  1 ID                 integer               0               0        17379
1
##  2 Date               character             0               0          731
0.0421
##  3 Season             integer               0               0            4
0.000230
##  4 Year               integer               0               0            2
0.000115
##  5 Month              integer               0               0           12
0.000690
##  6 Hour               integer               0               0           24
0.00138
##  7 Holiday            integer               0               0            2
0.000115
##  8 Weekday            integer               0               0            7
0.000403
##  9 Working_Day        integer               0               0            2
0.000115
## 10 Weather_Situation  integer               0               0            4
0.000230
## 11 Norm_Temp          numeric               0               0           50
0.00288
## 12 Norm_Feels_Temp    numeric               0               0           65
0.00374
## 13 Norm_Humidity      numeric               0               0           89
0.00512
## 14 Norm_Windspeed     numeric               0               0           30
0.00173
## 15 Users_Unregistered integer               0               0          322
```

```
0.0185
## 16 Users_Registered   integer               0               0          776
0.0447
## 17 Users_Total        integer               0               0          869
0.0500
```

Changing attributes, creating new ones for more interpretability

```
df_bike_sharing$Date <- dplyr::case_when(
      substring(df_bike_sharing$Date,1,4) == "2011" ~
        sub("2011", "2020",df_bike_sharing$Date),
      substring(df_bike_sharing$Date,1,4) =="2012" ~
        sub("2012", "2021",df_bike_sharing$Date),
      TRUE ~ as.character(df_bike_sharing$Date)
)

# Changing the 'Date' column datatype to Date
df_bike_sharing$Date <- as.Date(df_bike_sharing$Date)

# Extracting the Day from Date and storing in a new column
df_bike_sharing$Day_of_Month <- format(df_bike_sharing$Date, format = "%d")

# Converting Month numerical values to their designated month names
df_bike_sharing$Month <- month.abb[df_bike_sharing$Month]

# Converting Year column from values 0 and 1 to actual year values
# The value 0 represents year 2011 and 1 represents 2012
df_bike_sharing$Year <- ifelse(df_bike_sharing$Year == 0, 2020, 2021)

# Converting Weekday numerical values to their weekday names
df_bike_sharing$Day_of_Week <- dplyr::case_when(
  df_bike_sharing$Weekday == 0 ~ "Sun",
  df_bike_sharing$Weekday == 1 ~ "Mon",
  df_bike_sharing$Weekday == 2 ~ "Tue",
  df_bike_sharing$Weekday == 3 ~ "Wed",
  df_bike_sharing$Weekday == 4 ~ "Thur",
  df_bike_sharing$Weekday == 5 ~ "Fri",
  df_bike_sharing$Weekday == 6 ~ "Sat",
  TRUE ~ as.character(df_bike_sharing$Weekday)
)

# Converting integer values of Hour to HH:MM format
df_bike_sharing$Hour <- sprintf("%02d", df_bike_sharing$Hour)

# Adding Yearly Quarter Column
df_bike_sharing$Quarter = as.yearqtr(df_bike_sharing$Date, format = "%Yq%q")

# Converting numerical Season column to categorical to denote seasons:
# 1 - Winter
# 2 - Spring
```

```
# 3 - Summer
# 4 - Fall
df_bike_sharing$Season <- as.character(factor(df_bike_sharing$Season, levels
= 1:4,
                                    labels = c("Winter", "Spring", "Summer", "Fa
ll")))

# Converting Weather Situation to its respective assigned values
# 1: Clear, Few clouds, Partly cloudy, Partly cloudy
# 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
# 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + S
cattered clouds
# 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
df_bike_sharing$Weather_Situation <- as.character(factor(df_bike_sharing$Weat
her_Situation, levels = 1:4,
                                    labels = c("Clear or Partly Cloudy",
                                               "Mist and Cloudy",
                                               "Light Rain or Snow",
                                               "Heavy Rain or Snow")))
```

Creating new columns to convert normalized values to actual values

The temperature columns Temp, Feels_like_temp are normalized and calculated using:

(t - t_min) / (t_max - t_min)

Temp : t_min = -8, t_max = +39 (Celsius scale)

Feels_like_temp : t_min = -16, t_max = +50 (Celsius scale)

Humidity and Windspeed columns are normalized to the scale:

Humidity - max value of 100

Windspeed - max value of 67

```
# Converting temperature values to their actual values and storing in new col
umns
df_bike_sharing$Temp <- (df_bike_sharing$Norm_Temp * (39 + 8)) - 8
df_bike_sharing$Feels_Temp <- (df_bike_sharing$Norm_Temp * (50 + 16)) - 16

# Converting Humidity and Windspeed
df_bike_sharing$Humidity <- df_bike_sharing$Norm_Humidity * 100
df_bike_sharing$Windspeed <- df_bike_sharing$Norm_Windspeed * 67
```

Fabricating the routes for bike sharing data

```
df_bluebikes_routes_unique <- df_bluebikes_edited %>%
  filter(start.station.name != end.station.name)
```

```
# Reading the start and end locations into a dataframe
df_bluebikes_routes <- dplyr::select(df_bluebikes_routes_unique,
                          c(start.station.name,end.station.name,
                          Source.Longitude,Source.Latitude,
                          Destination.Longitude,Destination.Latitude))

df_bluebikes_routes_dist <- distinct(df_bluebikes_routes)

# Sampling the routes to create random routes for each row of original data
df_random_routes <- sample_n(df_bluebikes_routes_dist, nrow(df_bike_sharing),
replace = TRUE)
df_bike_sharing <- cbind(df_bike_sharing, df_random_routes)
ncol(df_bike_sharing)

## [1] 30

colnames(df_bike_sharing)[c(25, 26, 27, 28, 29, 30)] <- c("Source", "Destinat
ion",
                                                          "Src_Long", "Src_Lat", "Des
t_Long", "Dest_Lat")
```

Viewing the Final Data – Ready for Visualization

```
View(df_bike_sharing)
```

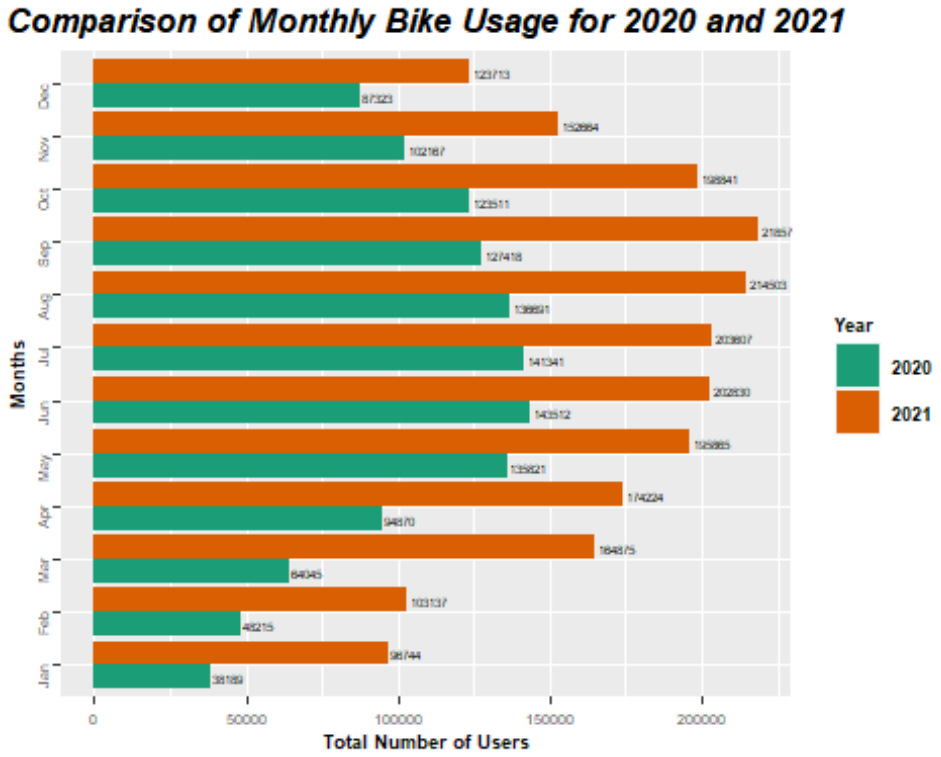
————————————- Data Visualization ————————————-

Grouped Bar Chart

This chart helps us to visually compare the monthly bike usage for years 2020 ans 2021

```
df_bike_grouped_bar <- df_bike_sharing %>%
  select(Month, Year, Users_Total) %>%
  group_by(Month, Year) %>%
  summarise(Users = sum(Users_Total))

## `summarise()` has grouped output by 'Month'. You can override using the `.
groups` argument.

df_bike_grouped_bar$Month = factor(df_bike_grouped_bar$Month, levels = month.
abb)

ggplot(df_bike_grouped_bar, aes(x = Month, y = Users, fill = as.factor(Year))
) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  geom_text(aes(label = Users),
            vjust = 0.5, hjust = -0.1, color = "black", size = 1.7, position
= position_dodge(0.9)) +
  scale_fill_brewer(palette = "Dark2") +
```

```
  ggtitle("Comparison of Monthly Bike Usage for 2020 and 2021") +
  xlab("Months") +
  ylab("Total Number of Users") +
  labs(fill = "Year") +
  scale_x_discrete(limits = ~month.abb) +
  theme(plot.title = element_text(size = 13, face = "bold.italic", color = "b
lack", hjust = 0.5),
        axis.title.x = element_text(size = 7, face = "bold"),
        axis.title.y = element_text(size = 7, face = "bold"),
        axis.text.x = element_text(size = 5, angle = 0),
        axis.text.y = element_text(size = 5, angle = 90),
        legend.position = "right",
        legend.title = element_text(size = 7, face = "bold"),
        legend.text = element_text(size = 7, face = "bold"),
        strip.text = element_text(size = 7))
```



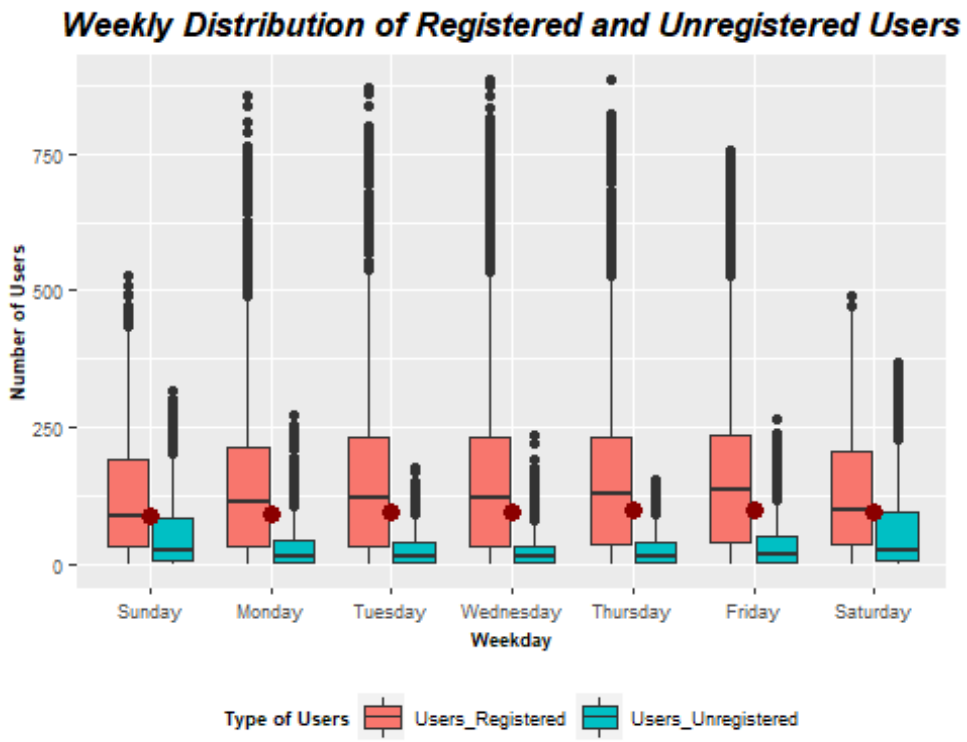
Box Plot

This plot shows the weekly distribution of registered and unregistered users

```
df_bike_boxplot <- df_bike_sharing %>%
  select(Weekday, Users_Unregistered, Users_Registered)

df_bike_boxplot_long <- df_bike_boxplot %>%
  pivot_longer(-Weekday,
               names_to = "Type_of_User",
               values_to = "Users")
```

```r
ggplot(data = df_bike_boxplot_long, aes(x = as.character(Weekday), y = Users)
) +
  geom_boxplot(aes(fill = Type_of_User)) +
  scale_x_discrete(labels = c("0" = "Sunday",
                              "1" = "Monday",
                              "2" = "Tuesday",
                              "3" = "Wednesday",
                              "4" = "Thursday",
                              "5" = "Friday",
                              "6" = "Saturday")) +
  ggtitle("Weekly Distribution of Registered and Unregistered Users") +
  xlab("Weekday") +
  ylab("Number of Users") +
  labs(fill = "Type of Users") +
  theme(plot.title = element_text(size = 13, face = "bold.italic", color = "b
lack", hjust = 0.5),
        axis.title.x = element_text(size = 7, face = "bold"),
        axis.title.y = element_text(size = 7, face = "bold"),
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        legend.position = "bottom",
        legend.title = element_text(size = 7, face = "bold"),
        legend.text = element_text(size = 7),
        strip.text = element_text(size = 7)) +
  stat_summary(fun.y = "mean", colour = "darkred", geom = "point",
               shape = 19, size = 3, show.legend = FALSE)
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

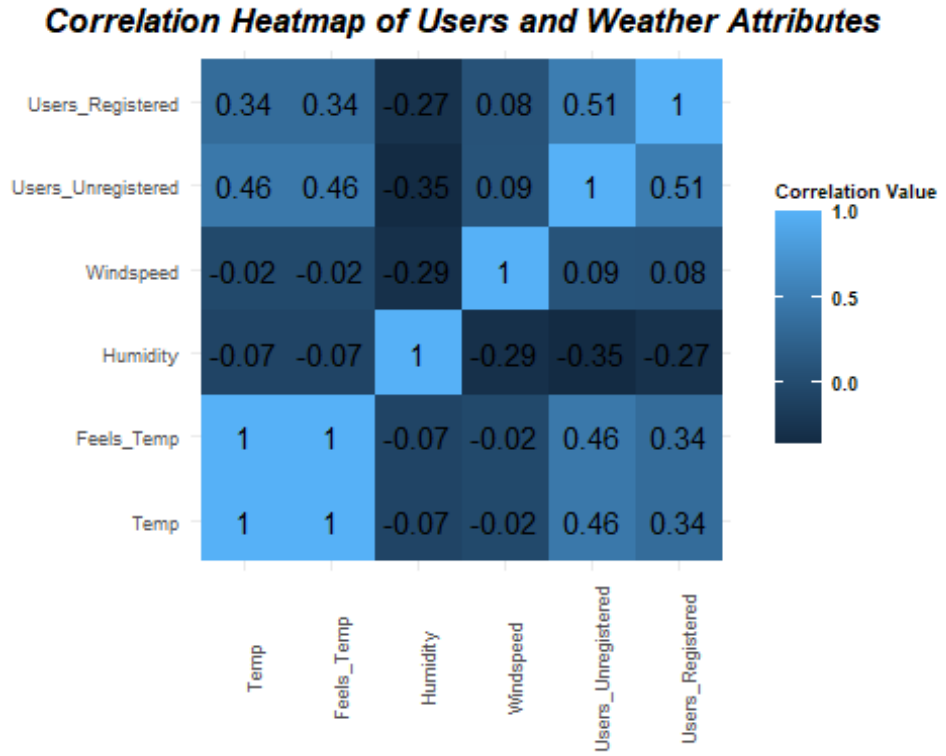


Heatmap

This heatmap shows the correlation between Registerd and Unregistered Users with the Weather Attributes

```
# Create a dataframe with numerical columns
df_numeric <- df_bike_sharing %>%
  select(Temp, Feels_Temp, Humidity, Windspeed, Users_Unregistered, Users_Reg
istered)

# Correlation Matrix of all numerical columns
df_bike_heatmap <- round(cor(df_numeric), 2)
df_bike_heatmap <- melt(df_bike_heatmap)

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified
by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified
by the
## caller; using TRUE

# Plotting heatmap
ggplot(data = df_bike_heatmap, aes(x = X1, y = X2, fill = value)) +
  geom_tile() +
  geom_text(aes(X1, X2, label = value), color = "black", size = 4) +
  ggtitle("Correlation Heatmap of Users and Weather Attributes") +
```

```
  theme_minimal() +
  labs(fill = "Correlation Value") +
  theme(plot.title = element_text(size = 13, face = "bold.italic", hjust = 0.
5),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_text(size = 7, angle = 90),
        axis.text.y = element_text(size = 7),
        legend.position = "right",
        legend.title = element_text(size = 7, face = "bold"),
        legend.text = element_text(size = 7, face = "bold"))
```



Bar Plot

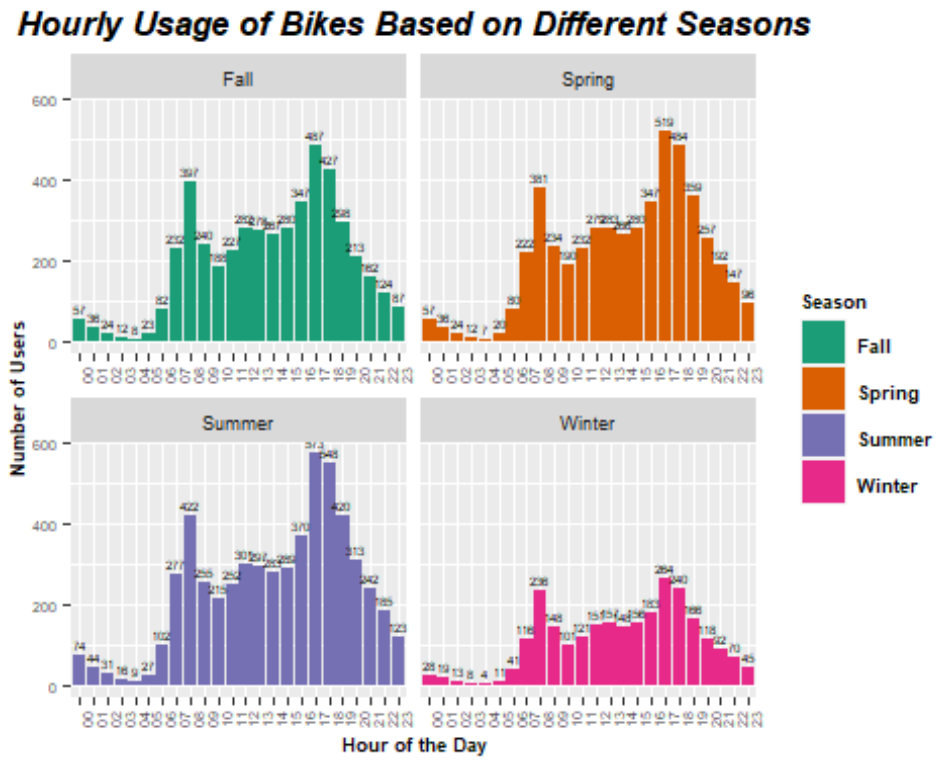
This chart shows the hourly usage of bikes based on different seasons

```
df_bike_bar <- df_bike_sharing %>%
  select(Hour, Season, Users_Total) %>%
  group_by(Hour, Season) %>%
  summarise(Users = ceiling(mean(Users_Total)))
```

```
## `summarise()` has grouped output by 'Hour'. You can override using the `.g
roups` argument.
```

```
# Plotting bar graph

# Since the x label used in the bar graph is ordinal categorical (hours of th
e day), hence we have not sorted the bars in decreasing order of their height
```

```
s.
ggplot(df_bike_bar, aes(x = Hour, y = Users)) +
  geom_bar(stat = "identity", position="dodge", aes(fill = Season)) +
  facet_wrap(~Season, scales = "free_x") +
  geom_text(aes(label = Users), angle = 0, vjust = -0.5,
            color = "black", size = 1.5, angle = 90) +
  scale_fill_brewer(palette = "Dark2") +
  ggtitle("Hourly Usage of Bikes Based on Different Seasons") +
  xlab("Hour of the Day") +
  ylab("Number of Users") +
  theme(plot.title = element_text(size = 13, face = "bold.italic", color = "b
lack", hjust = 0.5),
        axis.title.x = element_text(size = 7, face = "bold"),
        axis.title.y = element_text(size = 7, face = "bold"),
        axis.text.x = element_text(size = 5, angle = 90),
        axis.text.y = element_text(size = 5),
        legend.position = "right",
        legend.title = element_text(size = 7, face = "bold"),
        legend.text = element_text(size = 7, face = "bold"),
        strip.text = element_text(size = 7))

## Warning: Duplicated aesthetics after name standardisation: angle
```



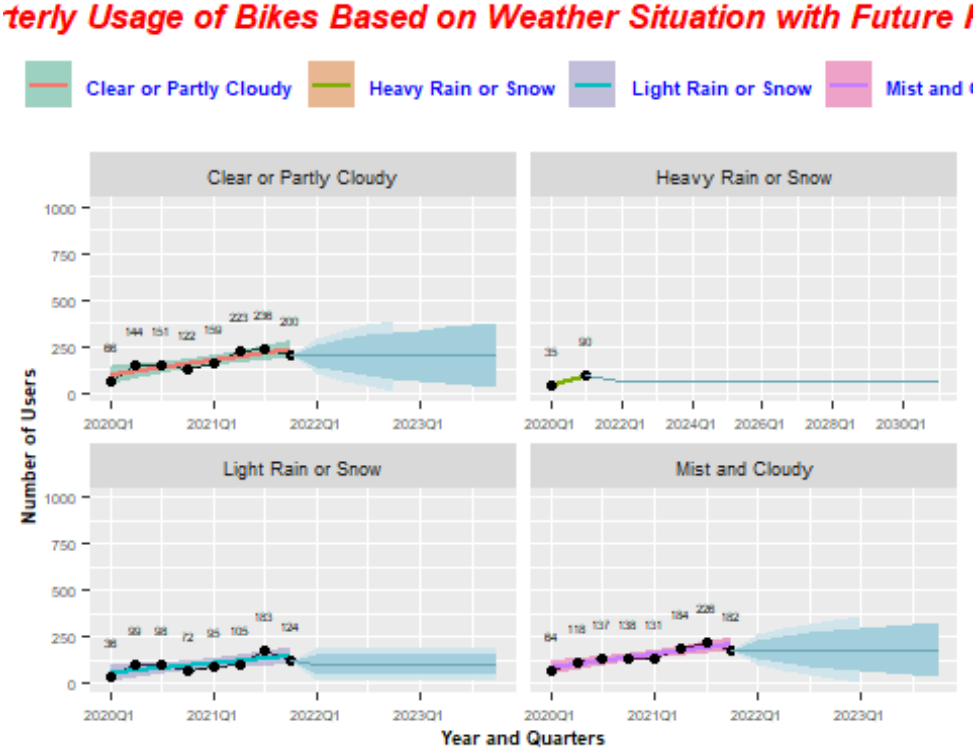
Line Chart

This chart shows the quarterlu usage of bikes based on weather situation

We are also predicting the number of users for next quarters

```
df_bike_line <- df_bike_sharing %>%
  group_by(Quarter, Weather_Situation) %>%
  summarise(Users = ceiling(mean(Users_Registered)))

## `summarise()` has grouped output by 'Quarter'. You can override using the
`.groups` argument.

ggplot(df_bike_line, aes(x = Quarter,  y = Users)) +
  geom_line() +
  geom_smooth(aes(color = Weather_Situation,
                  fill = Weather_Situation), method = "lm") +
  geom_point() +
  ylim(0, 1000) +
  facet_wrap(~ Weather_Situation, scales = "free_x") +
  geom_forecast(stat = "forecast", position = "identity",
                colour = "lightblue", showgap = FALSE) +
  geom_text(aes(label = Users),
            vjust = -2.5, color = "black", size = 1.7) +
  scale_fill_brewer(palette = "Dark2") +
  ggtitle("Quarterly Usage of Bikes Based on Weather Situation with Future Pr
ediction") +
  xlab("Year and Quarters") +
  ylab("Number of Users") +
  scale_x_yearqtr(format = "%YQ%q") +
  theme(plot.title = element_text(size = 13, face = "bold.italic", color = "r
ed", hjust = 0.5),
        axis.title.x = element_text(size = 7, face = "bold"),
        axis.title.y = element_text(size = 7, face = "bold"),
        axis.text.x = element_text(size = 5),
        axis.text.y = element_text(size = 5),
        legend.position = "top",
        legend.title = element_blank(),
        legend.text = element_text(colour = "blue", size = 7, face = "bold"),
        strip.text = element_text(size = 7))

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 3 rows containing non-finite values (stat_smooth).

## Warning in qt((1 - level)/2, df): NaNs produced

## Warning: Removed 3 rows containing non-finite values (stat_forecast).

## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; return
ing
## -Inf

## Warning: Removed 3 rows containing missing values (geom_point).

## Warning: Removed 3 rows containing missing values (geom_text).
```



Network Graph

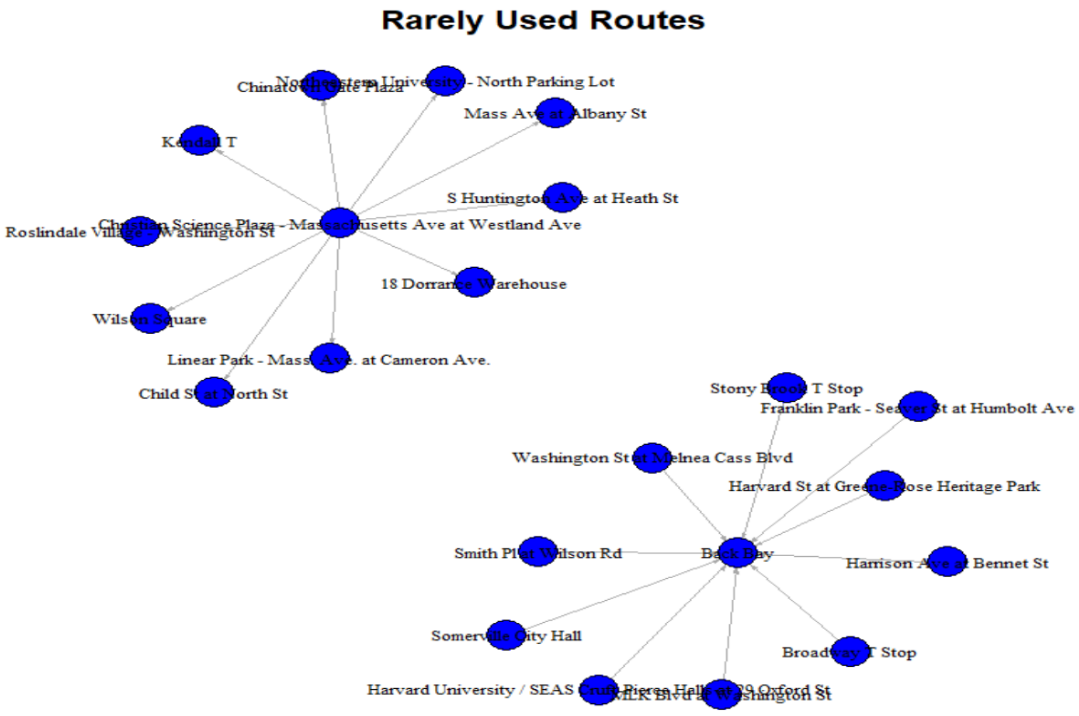
The graph is showcasing the 20 minimally used routes.

```
# Selecting the source and destination and summarizing the total users
plot_net <- dplyr::select(df_bike_sharing, c(Source, Destination, Users_Total
))

plot_net_filtered <- plot_net %>%
  group_by(Source, Destination) %>%
  summarise(Users_Avg = mean(Users_Total)) %>%
  arrange(Users_Avg) %>%
  head(20)

## `summarise()` has grouped output by 'Source'. You can override using the `
.groups` argument.

# Creating a network
grph_net <- graph.data.frame(plot_net_filtered[c(1, 2, 3)], directed = TRUE)
```

```
V(grph_net)$size <- plot_net_filtered$Users_Avg
```

```
## Warning in vattrs[[name]][index] <- value: number of items to replace is n
ot a
## multiple of replacement length
```

```
# Plotting the network
plot(grph_net,
     layout = layout.auto,
     vertex.size = 10,
     vertex.color = "blue",
     vertex.label.cex = 0.8,
     vertex.label.color = "black",
     edge.arrow.size = 0.1,
)
title(main =list("Rarely Used Routes", cex=1.5))
```



Map View of stations

##The map shows network of various bike stations and relative locations in Boston

```r
# Selecting the source and destination with respective longitude and latitude
columns
plot_map_filtered <- df_bike_sharing %>%
  dplyr::select(c(Source, Destination, Src_Long,
                  Src_Lat, Dest_Long, Dest_Lat, Users_Total)) %>%
  group_by(Source, Destination, Src_Long,
                  Src_Lat, Dest_Long, Dest_Lat) %>%
  summarise(Users = round(mean(Users_Total), 0))
```

```
## `summarise()` has grouped output by 'Source', 'Destination', 'Src_Long', '
Src_Lat', 'Dest_Long'. You can override using the `.groups` argument.
```

```r
# Using Google API to plot the mapview
api_key <- register_google(key = "AIzaSyCO_xreF1k7Gx-grRe5Dzz17BdzRca5398")
map_canvas <- get_map(c(left = min(plot_map_filtered$Src_Long),
                  bottom = min(plot_map_filtered$Src_Lat),
                  right = max(plot_map_filtered$Dest_Long),
                  top = max(plot_map_filtered$Dest_Lat)),
                  maptype = "satellite",
                  source = "google",
                  zoom = 10)
```

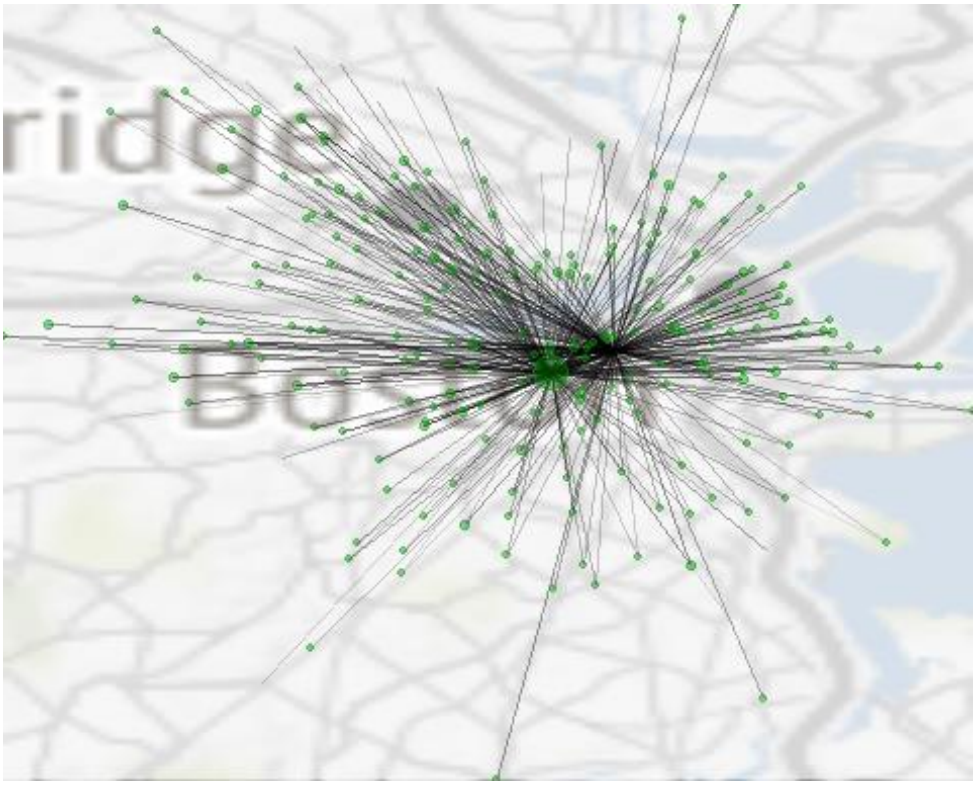
```
## Source : http://tile.stamen.com/terrain/10/309/378.png
```

```
## Source : http://tile.stamen.com/terrain/10/309/379.png
```

```r
# Plotting using ggmap
ggmap(map_canvas, darken = c(0.6, "white")) +
  geom_segment(data = plot_map_filtered,
               aes(x = Src_Long,
                   y = Src_Lat,
                   xend = Dest_Long,
                   yend = Dest_Lat,
                   alpha = sqrt(Users)),
               color = "#000000") +
  coord_cartesian() +
  scale_alpha(range = c(0.0001, .5)) +
  geom_point(data = plot_map_filtered %>%
               group_by(longitude = Src_Long,
                        latitude = Src_Lat) %>%
               summarize(rides = sum(Users)),
             aes(x = longitude, y = latitude, size = rides),
             color = "#009900", alpha = .4) +
  scale_size_continuous(range(4, 100)) +
  scale_color_viridis_c() +
  scale_fill_viridis_c() +
  theme_nothing()
```

```
## Coordinate system already present. Adding new coordinate system, which wil
l replace the existing one.
```

```
## `summarise()` has grouped output by 'longitude'. You can override using th
e `.groups` argument.
```



```
# Saving the html output as png in local drive
ggsave(filename = "station-network.jpg", width = 8, units = "in")

## Saving 8 x 4 in image
```

Word Cloud

This visualization is displaying the routes, the size of the text depends on the number of users

travelling on the route.

```
# Aggregating Total Number of users on a particular route using Group by

plot_cloud_filtered <- df_bike_sharing %>%
  dplyr::select(c(Source, Destination, Users_Total)) %>%
  group_by(Source, Destination) %>%
  summarise(Users = ceiling(mean(Users_Total)))

## `summarise()` has grouped output by 'Source'. You can override using the `
.groups` argument.

# Concatenating source and destination
plot_cloud_filtered$concat <- paste(plot_cloud_filtered$Source,
```

```
                                    plot_cloud_filtered$Destination,
                                    sep = " --> ")

set.seed(1234)
word_cloud <- wordcloud2(data = plot_cloud_filtered[c(4,3)],
                         size = 0.5, color = 'random-dark', gridSize = 15,
                         shape = "diamond", ellipticity = 0.2,
                         fontWeight = "normal", shuffle = FALSE)
word_cloud
```