Homework8

Patrick Seebold

This homework will explore basics of predictive modeling using an online-available data set concerning bike data in Seoul, South Korea.

First, let's get our libraries set:

```r
library(tidymodels)
```

```
Warning: package 'tidymodels' was built under R version 4.3.3

-- Attaching packages -------------------------------------- tidymodels 1.2.0 --

v broom        1.0.5     v recipes      1.1.0
v dials        1.3.0     v rsample      1.2.1
v dplyr        1.1.4     v tibble       3.2.1
v ggplot2      3.5.1     v tidyr        1.3.1
v infer        1.0.7     v tune         1.2.1
v modeldata    1.4.0     v workflows    1.1.4
v parsnip      1.2.1     v workflowsets 1.1.0
v purrr        1.0.2     v yardstick    1.3.1

Warning: package 'dials' was built under R version 4.3.3

Warning: package 'scales' was built under R version 4.3.3

Warning: package 'dplyr' was built under R version 4.3.3

Warning: package 'ggplot2' was built under R version 4.3.3

Warning: package 'infer' was built under R version 4.3.3
```

```
Warning: package 'modeldata' was built under R version 4.3.3

Warning: package 'parsnip' was built under R version 4.3.3

Warning: package 'purrr' was built under R version 4.3.3

Warning: package 'recipes' was built under R version 4.3.3

Warning: package 'rsample' was built under R version 4.3.3

Warning: package 'tidyr' was built under R version 4.3.3

Warning: package 'tune' was built under R version 4.3.3

Warning: package 'workflows' was built under R version 4.3.3

Warning: package 'workflowsets' was built under R version 4.3.3

Warning: package 'yardstick' was built under R version 4.3.3

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x purrr::discard() masks scales::discard()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
x recipes::step()  masks stats::step()
* Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```r
library(dplyr)
library(ggplot2)
```

EDA

Next, let's load the data in and resolve the known error mentioned in the instructions. We can specify the fileEncoding and avoid checking names to solve the problem:

```r
data = read.csv("SeoulBikeData.csv", fileEncoding = "Latin1", check.names = F)
```

Great, now we can go ahead with our EDA to make sure everything looks good. First, lets check for missing values:

```r
sum(is.na(data))
```

```
[1] 0
```

Sweet, no missing values. makes for a good start. Next, we'll take a look at the column types, rename to avoid strange variable names, and change our categorical variables to factors. The instructions say to change our variable names later in the process, but I'd rather they be in a clean format for the rest of the EDA!

```r
summary(data)
```

```
     Date            Rented Bike Count      Hour       Temperature(°C)
 Length:8760        Min.   :   0.0     Min.   : 0.00   Min.   :-17.80
 Class :character   1st Qu.: 191.0     1st Qu.: 5.75   1st Qu.:  3.50
 Mode  :character   Median : 504.5     Median :11.50   Median : 13.70
                    Mean   : 704.6     Mean   :11.50   Mean   : 12.88
                    3rd Qu.:1065.2     3rd Qu.:17.25   3rd Qu.: 22.50
                    Max.   :3556.0     Max.   :23.00   Max.   : 39.40
  Humidity(%)    Wind speed (m/s) Visibility (10m) Dew point temperature(°C)
 Min.   : 0.00   Min.   :0.000    Min.   :  27     Min.   :-30.600
 1st Qu.:42.00   1st Qu.:0.900    1st Qu.: 940     1st Qu.: -4.700
 Median :57.00   Median :1.500    Median :1698     Median :  5.100
 Mean   :58.23   Mean   :1.725    Mean   :1437     Mean   :  4.074
 3rd Qu.:74.00   3rd Qu.:2.300    3rd Qu.:2000     3rd Qu.: 14.800
 Max.   :98.00   Max.   :7.400    Max.   :2000     Max.   : 27.200
 Solar Radiation (MJ/m2)  Rainfall(mm)     Snowfall (cm)       Seasons
```

```
 Min.   :0.0000          Min.   : 0.0000   Min.   :0.00000   Length:8760
 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   :0.5691          Mean   : 0.1487   Mean   :0.07507
 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:8760        Length:8760
 Class :character   Class :character
 Mode  :character   Mode  :character
```

```
colnames(data) = c('Date', 'NumBikes', "Hour", "Temperature", "Humidity",
                   "WindSpeed","Visibility", "DewPointTemp", "SolarRad", "Rainfall"
                   , "Snowfall", "Season", "Holiday", "FunctioningDay" )

data$Season = as.factor(data$Season)
data$Holiday = as.factor(data$Holiday)
data$FunctioningDay = as.factor(data$FunctioningDay)
levels(data$Season)
```

```
[1] "Autumn" "Spring" "Summer" "Winter"
```

```
levels(data$Holiday)
```

```
[1] "Holiday"    "No Holiday"
```

```
levels(data$FunctioningDay)
```

```
[1] "No"  "Yes"
```

Great, now we can see that everything looks good with our numerical variables, our variable names are no longer likely to cause an issue, and our categorical variables are appropriately recast as factors! Next, we will change the date into a workable arithmetic form using lubridate:

```r
typeof(data$Date) # currently character
```

```
[1] "character"
```

```r
data$Date = lubridate::dmy(data$Date)
typeof(data$Date) # now it's a double!
```

```
[1] "double"
```

Now that we have cleaned the data up, we will do some summary stats, specifically looking at bike rental count, rainfall, and snowfall. We'll also examine these across some categorical variables, such as FunctioningDay, Holiday, and Season:

```r
data |> # check bike numbers by season
  group_by(FunctioningDay) |>
  summarize(mean = mean(NumBikes), sd = sd(NumBikes))# Non-functioning days mean no bikes!
```

```
# A tibble: 2 x 3
  FunctioningDay  mean    sd
  <fct>          <dbl> <dbl>
1 No                 0     0
2 Yes              729.  642.
```

```r
sum(data$FunctioningDay == "No") # 295 of the data points can be excluded
```

```
[1] 295
```

```r
sub_data = subset(data, data$FunctioningDay == "Yes") # we can ignore days that are not func


sub_data |> # check bike numbers by season
  group_by(Season) |>
  summarize(mean = mean(NumBikes), sd = sd(NumBikes))
```

```
# A tibble: 4 x 3
  Season  mean    sd
  <fct>  <dbl> <dbl>
1 Autumn  924.  618.
2 Spring  746.  619.
3 Summer 1034.  690.
4 Winter  226.  150.
```

```r
sub_data |> # check bike numbers by holiday
  group_by(Holiday) |>
  summarize(mean = mean(NumBikes), sd = sd(NumBikes))
```

```
# A tibble: 2 x 3
  Holiday     mean    sd
  <fct>      <dbl> <dbl>
1 Holiday     529.  574.
2 No Holiday  739.  644.
```

```r
sub_data |> # check rain  by season
  group_by(Season) |>
  summarize(mean = mean(Rainfall),  sd = sd(Rainfall))
```

```
# A tibble: 4 x 3
  Season   mean    sd
  <fct>   <dbl> <dbl>
1 Autumn 0.118  0.890
2 Spring 0.187  1.21
3 Summer 0.253  1.59
4 Winter 0.0328 0.423
```

```r
sub_data |> # check snowfall by season
  group_by(Season) |>
  summarize(mean = mean(Snowfall),  sd = sd(Snowfall))
```

```
# A tibble: 4 x 3
  Season   mean    sd
  <fct>   <dbl> <dbl>
1 Autumn 0.0635 0.522
2 Spring 0      0
3 Summer 0      0
4 Winter 0.248  0.698
```

We see that summer appears to have the highest number of bike rentals, and winter has the fewest. Bike rentals are more common on Non-Holidays, so they are likely being used to commute to work. Finally, we see from our tables that snowfall is most plentiful in winter, while rainfall is most plentiful in summer.

Next up, let's summarize across hours so that we can collapse each day into a single observation.

```r
clean_data = sub_data |>
  group_by(Date, Season, Holiday) |>
  summarize(TotBikes = sum(NumBikes), TotRain = sum(Rainfall), TotSnow = sum(Snowfall), Mean
```

```
`summarise()` has grouped output by 'Date', 'Season'. You can override using
the `.groups` argument.
```

```r
head(clean_data)
```

```
# A tibble: 6 x 12
# Groups:   Date, Season [6]
  Date       Season Holiday    TotBikes TotRain TotSnow MeanTemp MeanHumid
  <date>     <fct>  <fct>         <int>   <dbl>   <dbl>    <dbl>     <dbl>
1 2017-12-01 Winter No Holiday     9539     0       0    -2.45       45.9
2 2017-12-02 Winter No Holiday     8523     0       0     1.32       62.0
3 2017-12-03 Winter No Holiday     7222     4       0     4.88       81.5
4 2017-12-04 Winter No Holiday     8729     0.1     0    -0.304      52.5
5 2017-12-05 Winter No Holiday     8307     0       0    -4.46       36.4
6 2017-12-06 Winter No Holiday     6669     1.3     8.6   0.0458     70.8
# i 4 more variables: MeanWindSpeed <dbl>, MeanVis <dbl>, MeanDewPoint <dbl>,
#   MeanSolar <dbl>
```

Great, now we have our final data set for training. Let's recreate our basic summaries and do some plots on this cleaned data. We'll also report some correlations. Let's first recreate our summary stats:

```r
clean_data |> # check bike numbers by season
  group_by(Season) |>
  summarize(mean = mean(TotBikes), sd = sd(TotBikes))
```

```
# A tibble: 4 x 3
  Season   mean    sd
  <fct>   <dbl> <dbl>
1 Autumn 22099. 6711.
2 Spring 17910. 8357.
3 Summer 24818. 7297.
4 Winter  5413. 1808.
```

```r
clean_data |> # check bike numbers by holiday
  group_by(Holiday) |>
  summarize(mean = mean(TotBikes), sd = sd(TotBikes))
```

```
# A tibble: 2 x 3
  Holiday      mean     sd
  <fct>       <dbl>  <dbl>
1 Holiday    12700. 10504.
2 No Holiday 17727.  9862.
```

```r
clean_data |> # check rain  by season
  group_by(Season) |>
  summarize(mean = mean(TotRain),  sd = sd(TotRain))
```

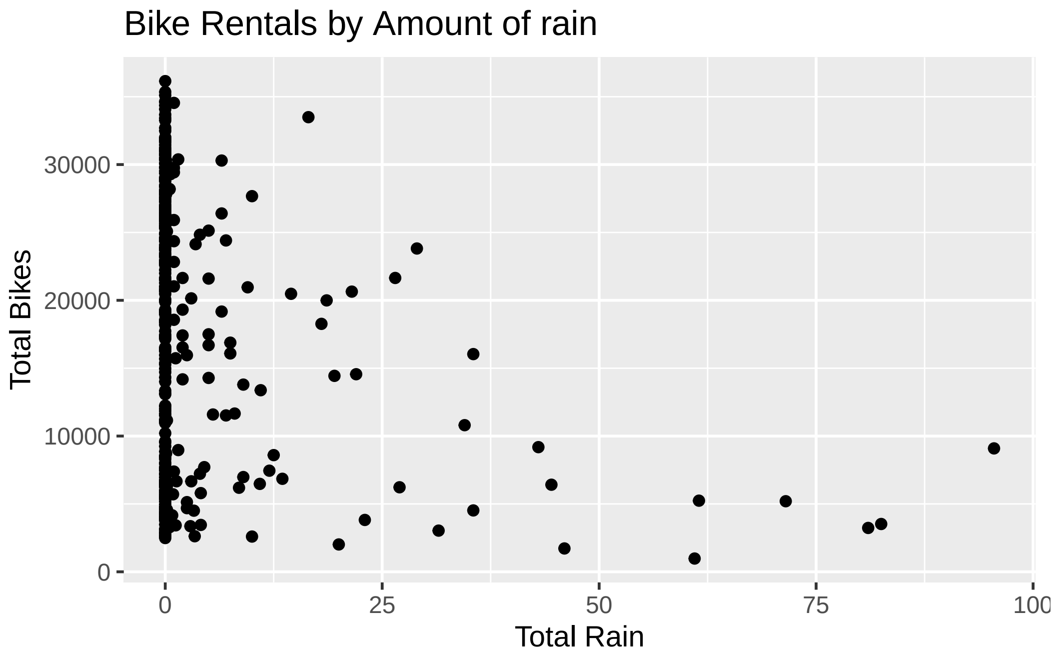
```
# A tibble: 4 x 3
  Season  mean    sd
  <fct>  <dbl> <dbl>
1 Autumn 2.81   8.61
2 Spring 4.49  12.7
3 Summer 6.08  17.0
4 Winter 0.788  3.28
```

```r
clean_data |> # check snowfall by season
  group_by(Season) |>
  summarize(mean = mean(TotSnow),  sd = sd(TotSnow))
```

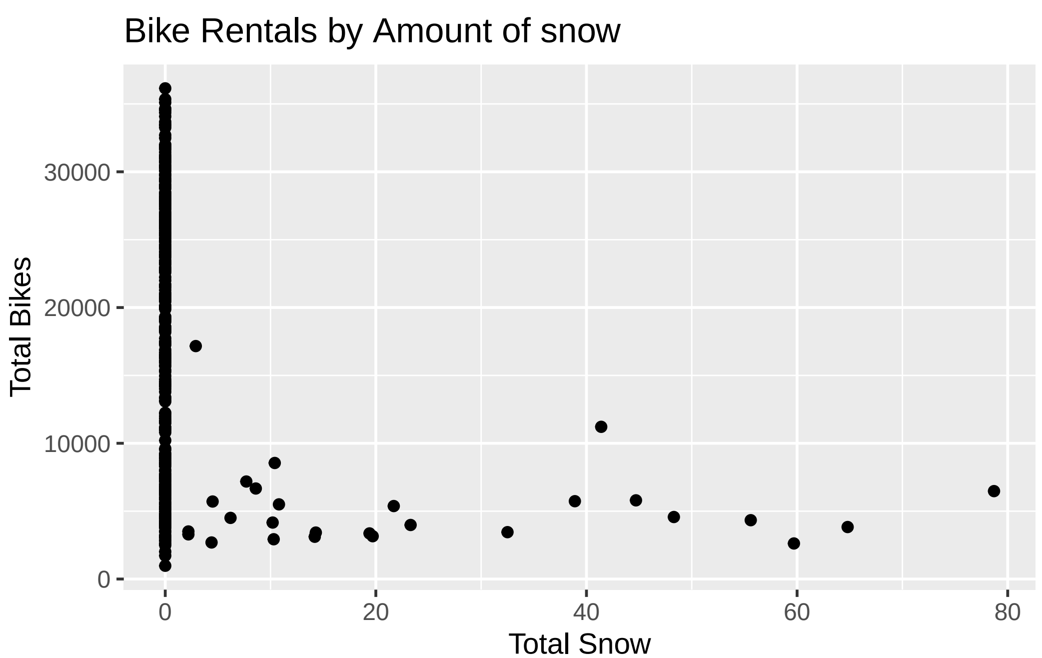
```
# A tibble: 4 x 3
  Season  mean    sd
  <fct>  <dbl> <dbl>
1 Autumn  1.52  9.83
2 Spring  0     0
3 Summer  0     0
4 Winter  5.94 14.0
```

We see that the same trends hold; more bikes in summer, more bikes on non-holidays, more rain in summer, and more snow in winter. Next, let's plot some of the numerical variables:

```r
ggplot(clean_data, aes(x = TotRain, y = TotBikes)) +
       geom_point() +
       labs(x = 'Total Rain', y = 'Total Bikes', title = 'Bike Rentals by Amount of rain')
```

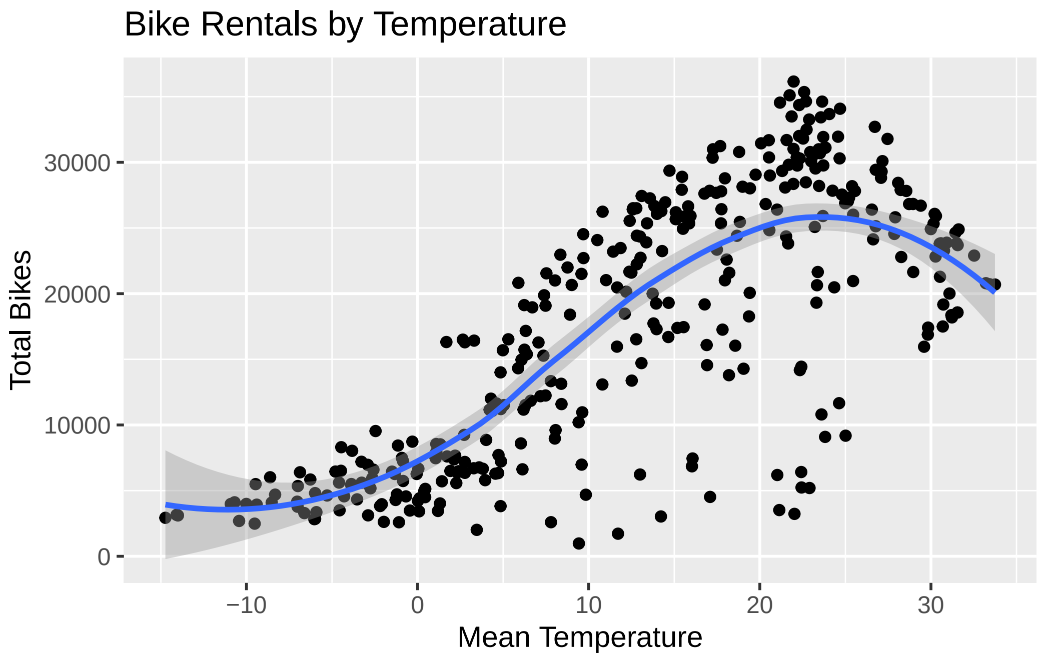



```
ggplot(clean_data, aes(x = TotSnow, y = TotBikes)) +
       geom_point() +
       labs(x = 'Total Snow', y = 'Total Bikes', title = 'Bike Rentals by Amount of snow')
```



```
ggplot(clean_data, aes(x = MeanTemp, y = TotBikes)) +
       geom_point() +
       geom_smooth() +
       labs(x = 'Mean Temperature', y = 'Total Bikes', title = 'Bike Rentals by Temperature')
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



These plots make sense. We see a lot more bike rentals when there is low snow and low rain. Also, we see that the number of bikes increase as temperature increases, until a certain point. These plots appear to be accurately capturing info about how weather influence bike riding behavior!

Finally, let's plot some correlations:

```
cor(clean_data[sapply(clean_data, is.numeric)])
```

```
              TotBikes     TotRain     TotSnow     MeanTemp   MeanHumid
TotBikes    1.00000000 -0.23910905 -0.26529110  0.753076732  0.03588697
TotRain    -0.23910905  1.00000000 -0.02313404  0.144517274  0.52864263
TotSnow    -0.26529110 -0.02313404  1.00000000 -0.266963662  0.06539191
MeanTemp    0.75307673  0.14451727 -0.26696366  1.000000000  0.40416749
MeanHumid   0.03588697  0.52864263  0.06539191  0.404167486  1.00000000
```

```
MeanWindSpeed -0.19288142 -0.10167578  0.02088156 -0.260721792 -0.23425778
MeanVis        0.16599375 -0.22199387 -0.10188902  0.002336683 -0.55917733
MeanDewPoint   0.65047655  0.26456621 -0.20955286  0.962796255  0.63204729
MeanSolar      0.73589290 -0.32270413 -0.23343056  0.550274301 -0.27444967
              MeanWindSpeed      MeanVis MeanDewPoint   MeanSolar
TotBikes        -0.19288142  0.165993749    0.6504765  0.73589290
TotRain         -0.10167578 -0.221993866    0.2645662 -0.32270413
TotSnow          0.02088156 -0.101889019   -0.2095529 -0.23343056
MeanTemp        -0.26072179  0.002336683    0.9627963  0.55027430
MeanHumid       -0.23425778 -0.559177334    0.6320473 -0.27444967
MeanWindSpeed    1.00000000  0.206022636   -0.2877032  0.09612635
MeanVis          0.20602264  1.000000000   -0.1535516  0.27139591
MeanDewPoint    -0.28770322 -0.153551591    1.0000000  0.38315713
MeanSolar        0.09612635  0.271395906    0.3831571  1.00000000
```

We can see some impressive correlations here. For example, Mean Temp, Mean Dew Point, and Mean Solar are all quite positively correlated with total number of bike rentals. These are likely all tied into the weather we commented on above - when the weather is nice, people are more likely to rent bikes, thus leading to higher correlation values! We should be able to do some nice prediction on this data given the strengths of these correlations, although we might be at risk of overfitting if we aren't careful.

Modeling the Data