Code

Project 2: Regression

Link to “Hourly Weather Surface - Brazil (Southeast region)”: https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region (https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region)

Initial Data Exploration and Cleaning

Hide

```
#load the data
sude <- read.csv("sudeste.csv")
sude <- sude[ which(sude$yr>=2009 & sude$yr<2016), ]
attach(sude)
```

```
The following objects are masked _by_ .GlobalEnv:

    city, date
```

Check out the Kaggle link for more information on the data set and its variables

Hide

```
#exploration function 1
str(sude)
```

```
'data.frame':	6788624 obs. of  31 variables:
 $ wsid: int  178 178 178 178 178 178 178 178 178 178 ...
 $ wsnm: chr  "SÃƒO GONÃ‡ALO" "SÃƒO GONÃ‡ALO" "SÃƒO GONÃ‡ALO" "SÃƒO GONÃ‡ALO" ...
 $ elvt: num  237 237 237 237 237 237 237 237 237 237 ...
 $ lat : num  -6.84 -6.84 -6.84 -6.84 -6.84 ...
 $ lon : num  -38.3 -38.3 -38.3 -38.3 -38.3 ...
 $ inme: chr  "A333" "A333" "A333" "A333" ...
 $ city: chr  "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" ...
 $ prov: chr  "RJ" "RJ" "RJ" "RJ" ...
 $ mdct: chr  "2009-01-01 00:00:00" "2009-01-01 01:00:00" "2009-01-01 02:00:00" "2009-01-01 03:0
0:00" ...
 $ date: chr  "2009-01-01" "2009-01-01" "2009-01-01" "2009-01-01" ...
 $ yr  : int  2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
 $ mo  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ da  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ hr  : int  0 1 2 3 4 5 6 7 8 9 ...
 $ prcp: num  NA NA NA NA NA NA NA NA NA NA ...
 $ stp : num  983 983 984 984 983 ...
 $ smax: num  983 983 984 984 984 ...
 $ smin: num  982 983 983 984 983 ...
 $ gbrd: num  NA NA NA NA NA ...
 $ temp: num  22.9 25.9 27.8 27 24.1 22.8 22.5 24.6 23.6 23.7 ...
 $ dewp: num  19.5 16.3 19.3 19.5 20 20.6 21 18.5 18.2 17.8 ...
 $ tmax: num  23.7 26 28.2 27.9 27 24.1 22.9 24.9 24.8 23.8 ...
 $ dmax: num  19.8 20 19.3 19.6 20.1 20.7 21.1 21.1 18.5 18.3 ...
 $ tmin: num  22.5 22.2 25.9 26.7 24.1 22.8 22.1 22.5 23.6 23 ...
 $ dmin: num  17.2 15.5 15.8 19.3 19.4 20.1 20.5 18.5 17.9 17.8 ...
 $ hmdy: num  81 56 60 64 78 87 91 69 72 70 ...
 $ hmax: num  85 87 60 65 78 88 93 91 72 74 ...
 $ hmin: num  70 54 52 59 64 78 87 68 67 69 ...
 $ wdsp: num  0.6 1.8 2.3 1.1 0.3 NA 0.9 2.7 1.8 2.1 ...
 $ wdct: num  196 92 84 87 146 111 111 109 98 109 ...
 $ gust: num  1.7 3.9 6.4 5.2 3.1 1.1 2.4 4.6 5 5.3 ...
```

This data set is hourly weather data from 122 weather stations in Southeast (Sudeste) Brazil. I am aiming to predict the temperature given other variables with my regression models. I decided to start off exploring the data with the str() function to get an idea of the structure of the data and see a list of all the columns. We can see that this is a data set containing 9779168 observations (cut down to) and 31 attributes. We also see the data types for each column and they seem to all make sense. Looking closer at the some of the data preview we can notice that certain columns seem to have NAs right off the bat; we'll tackle this in just a bit. For now, given the great number of columns, I want to remove a few unnecessary columns so we can work with a more clean and concise set of data.

Hide

```
#cleaning: dropping unnecessary columns mdct, yr, mo, da
sude <- subset(sude, select = -c(mdct, yr, mo, da))
```

There are many columns related to time: mdct, yr, mo, da, and hr. Date contains yr, mo, and da so I will discard those along with mdct which doesn't really add much as it is just date and hr. So I'll keep just date and hr separately.

Hide

```
#cleaning: dropping unnecessary columns wsid, inme, wsnm, prov
sude <- subset(sude, select = -c(wsid, inme, wsnm, prov))
```

Now there seems to be other multiple variables in regards to weather stations and locations: wsid, wsnm, inme, city, and prov. Weather station id, wsid, and inme, station number, are essentially the same idea of identifying stations which I don’t think is too useful. Also, wsnm, name of the station, usually the city location, is redundant as we have city so I will drop this as well. Finally, though province might be helpful, city is more specific and useful so this will be dropped as well.

Hide

```
#cleaning: dropping unnecessary columns smax, smin, dmax, tmax, tmin, dmin, hmax, hmin
sude <- subset(sude, select = -c(smax, smin, dmax, tmax, tmin, dmin, hmax, hmin))
str(sude)
```

```
'data.frame':   6788624 obs. of  15 variables:
 $ elvt: num  237 237 237 237 237 237 237 237 237 237 ...
 $ lat : num  -6.84 -6.84 -6.84 -6.84 -6.84 ...
 $ lon : num  -38.3 -38.3 -38.3 -38.3 -38.3 ...
 $ city: chr  "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" "SÃ£o GonÃ§alo" ...
 $ date: chr  "2009-01-01" "2009-01-01" "2009-01-01" "2009-01-01" ...
 $ hr  : int  0 1 2 3 4 5 6 7 8 9 ...
 $ prcp: num  NA NA NA NA NA NA NA NA NA NA ...
 $ stp : num  983 983 984 984 983 ...
 $ gbrd: num  NA NA NA NA NA ...
 $ temp: num  22.9 25.9 27.8 27 24.1 22.8 22.5 24.6 23.6 23.7 ...
 $ dewp: num  19.5 16.3 19.3 19.5 20 20.6 21 18.5 18.2 17.8 ...
 $ hmdy: num  81 56 60 64 78 87 91 69 72 70 ...
 $ wdsp: num  0.6 1.8 2.3 1.1 0.3 NA 0.9 2.7 1.8 2.1 ...
 $ wdct: num  196 92 84 87 146 111 111 109 98 109 ...
 $ gust: num  1.7 3.9 6.4 5.2 3.1 1.1 2.4 4.6 5 5.3 ...
```

There are still many variables left, 23 actually, and I think there is still some attributes that are unecessary for our modeling. Moving into the actual weather data, we see that for air pressure, dew point temperature, temperature, and relative humid temperature there are maximum and minimum columns in regards to each recording’s max and min within the last hour. I believe the instant recording of each type of weather data is sufficient so I will drop smax, smin, dmax, dmin, hmax, and hmin. Tmax and tmin, which are the max and min temperatures of the last hour, will be dropped for another reason; they are too highly correlated with temperature, our target variable.

Now we have a decent amount of valuable variables to work with so we are efficient with our resources. Let’s take a deeper look into our variables with the summary() function.

Hide

```
#exploration function 2
summary(sude)
```

```
     elvt             lat              lon              city          
 Min.   :   0.0   Min.   :-24.96   Min.   :-51.41   Length:6788624    
 1st Qu.: 288.0   1st Qu.:-22.38   1st Qu.:-47.43   Class :character  
 Median : 573.0   Median :-20.64   Median :-44.45   Mode  :character  
 Mean   : 592.7   Mean   :-20.13   Mean   :-44.53                     
 3rd Qu.: 873.0   3rd Qu.:-18.79   3rd Qu.:-42.39                     
 Max.   :1758.0   Max.   :  0.00   Max.   :  0.00                     
                                                                      
    date                 hr             prcp              stp        
 Length:6788624     Min.   : 0.0   Min.   :  0       Min.   :   0.0  
 Class :character   1st Qu.: 5.0   1st Qu.:  0       1st Qu.: 912.9  
 Mode  :character   Median :12.0   Median :  0       Median : 945.3  
                    Mean   :11.5   Mean   :  1       Mean   : 893.3  
                    3rd Qu.:18.0   3rd Qu.:  1       3rd Qu.: 971.8  
                    Max.   :23.0   Max.   :100       Max.   :1050.0  
                                   NA's   :5870742                   
     gbrd              temp            dewp              hmdy       
 Min.   :    0.0   Min.   :-3.80   Min.   :-10.00   Min.   :  0.00  
 1st Qu.:   93.4   1st Qu.:18.00   1st Qu.: 12.60   1st Qu.: 55.00  
 Median :  926.0   Median :21.50   Median : 16.30   Median : 75.00  
 Mean   : 1212.8   Mean   :20.79   Mean   : 15.05   Mean   : 68.35  
 3rd Qu.: 2142.9   3rd Qu.:25.00   3rd Qu.: 18.90   3rd Qu.: 89.00  
 Max.   :11586.5   Max.   :44.90   Max.   : 42.70   Max.   :100.00  
 NA's   :2850097   NA's   :15      NA's   :325                      
     wdsp             wdct            gust       
 Min.   : 0.0     Min.   :  0.0   Min.   : 0.00  
 1st Qu.: 0.8     1st Qu.: 59.0   1st Qu.: 2.40  
 Median : 1.8     Median :117.0   Median : 4.30  
 Mean   : 2.0     Mean   :140.7   Mean   : 4.58  
 3rd Qu.: 2.9     3rd Qu.:217.0   3rd Qu.: 6.40  
 Max.   :19.8     Max.   :360.0   Max.   :50.00  
 NA's   :710413                   NA's   :251170 
```

We can clearly see that city and date aren't numerical variables as they don't have statistics like the other variables which is appropriate but they don't seem to be the right type still; city would be more useful as a factor and date as a date type.

Hide

```
#cleaning: changing variable data types for city and date
sude$city <- as.factor(sude$city)
sude$date <- as.Date(sude$date)
summary(sude)
```

```
     elvt            lat             lon        
 Min.   :   0.0   Min.   :-24.96   Min.   :-51.41  
 1st Qu.: 288.0   1st Qu.:-22.38   1st Qu.:-47.43  
 Median : 573.0   Median :-20.64   Median :-44.45  
 Mean   : 592.7   Mean   :-20.13   Mean   :-44.53  
 3rd Qu.: 873.0   3rd Qu.:-18.79   3rd Qu.:-42.39  
 Max.   :1758.0   Max.   :  0.00   Max.   :  0.00  

                   city              date                  hr      
 Rio de Janeiro       : 192792   Min.   :2009-01-01   Min.   : 0.0  
 Campos dos Goytacazes: 122688   1st Qu.:2010-10-28   1st Qu.: 5.0  
 Belo Horizonte       :  78984   Median :2012-07-30   Median :12.0  
 Ã\u0081guas Vermelhas     :  61344   Mean   :2012-07-21   Mean   :11.5  
 AimorÃ©s             :  61344   3rd Qu.:2014-04-23   3rd Qu.:18.0  
 Alegre               :  61344   Max.   :2015-12-31   Max.   :23.0  
 (Other)              :6210128                                      
     prcp              stp              gbrd              temp      
 Min.   :  0       Min.   :   0.0   Min.   :    0.0   Min.   :-3.80  
 1st Qu.:  0       1st Qu.: 912.9   1st Qu.:   93.4   1st Qu.:18.00  
 Median :  0       Median : 945.3   Median :  926.0   Median :21.50  
 Mean   :  1       Mean   : 893.3   Mean   : 1212.8   Mean   :20.79  
 3rd Qu.:  1       3rd Qu.: 971.8   3rd Qu.: 2142.9   3rd Qu.:25.00  
 Max.   :100       Max.   :1050.0   Max.   :11586.5   Max.   :44.90  
 NA's   :5870742                    NA's   :2850097   NA's   :15     
     dewp             hmdy             wdsp             wdct      
 Min.   :-10.00   Min.   :  0.00   Min.   : 0.0     Min.   :  0.0  
 1st Qu.: 12.60   1st Qu.: 55.00   1st Qu.: 0.8     1st Qu.: 59.0  
 Median : 16.30   Median : 75.00   Median : 1.8     Median :117.0  
 Mean   : 15.05   Mean   : 68.35   Mean   : 2.0     Mean   :140.7  
 3rd Qu.: 18.90   3rd Qu.: 89.00   3rd Qu.: 2.9     3rd Qu.:217.0  
 Max.   : 42.70   Max.   :100.00   Max.   :19.8     Max.   :360.0  
 NA's   :325                       NA's   :710413                  
     gust       
 Min.   : 0.00  
 1st Qu.: 2.40  
 Median : 4.30  
 Mean   : 4.58  
 3rd Qu.: 6.40  
 Max.   :50.00  
 NA's   :251170 
```

Now the last aspect of data cleaning, dealing with missing values. It seems that prcp, gbrd, temp, dewp, wdsp, and gust all have missing values with temp and dewp with the relatively least. Prcp is the amount of precipitation in the last hour and may have NAs here because there was no rainfall; this will be set to 0. Similarly, the NAs in gbrd, the measure of solar radiation, are most likely due to times of little sunlight, such as nighttime; this pattern can be easily verified in the graph below. So missing values in grbd will also be set to 0.

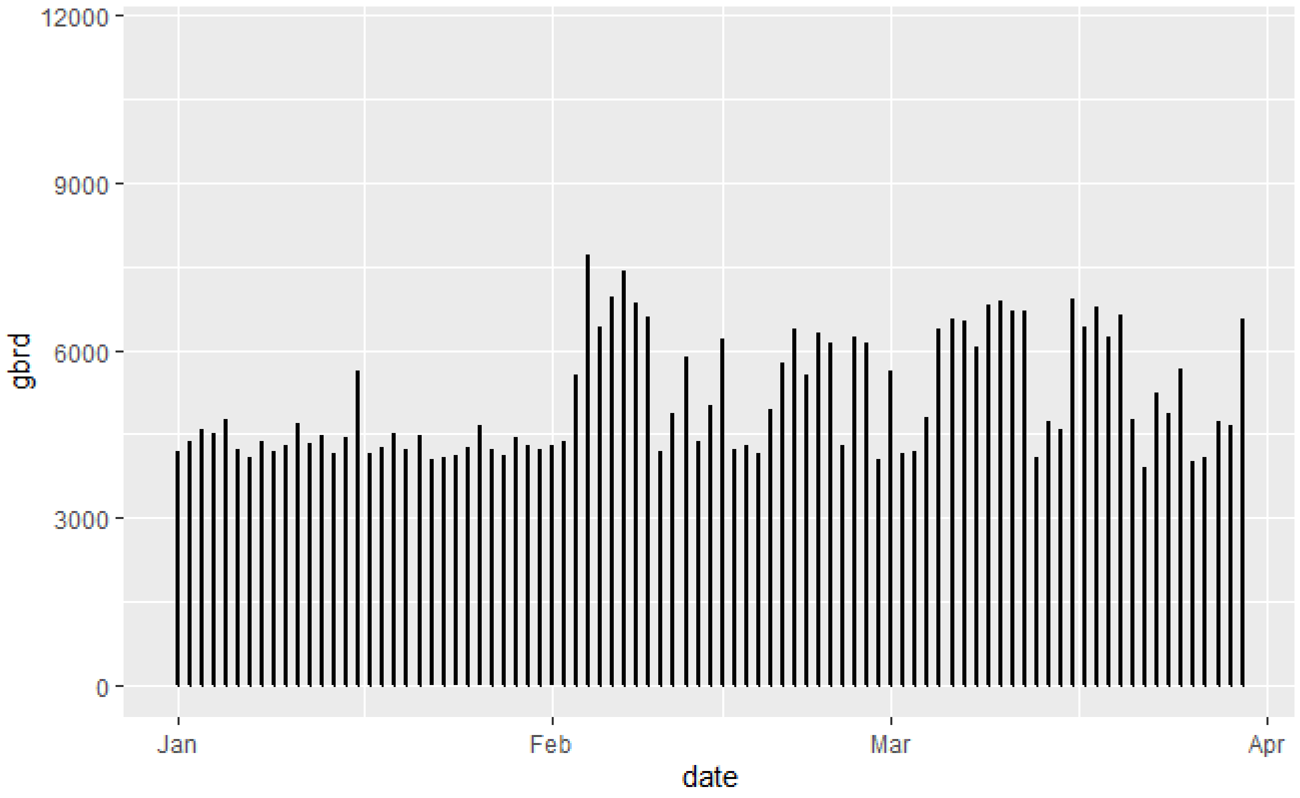
Hide

```
#exploring gbrd for missing value patterns
library(ggplot2)
```

```
package 惁梔ggplot2惁怍 was built under R version 4.0.4
```

Hide

```
ggplot(sude, aes(x = date, y = gbrd)) + geom_line(size = 1)  + scale_x_date(limits = c(as.Date(
"2010-1-1"), as.Date("2010-3-30")))
```



Hide

```
#cleaning: dealing with missing values for prcp, gbrd, set to 0
sude$prcp[is.na(sude$prcp)] <- 0
sude$gbrd[is.na(sude$gbrd)] <- 0
```

Hide

```
#cleaning: dealing with missing values for temp, dewp, wdsp, and gust with interpolation
library(zoo)
```

```
package 惁梔zoo惁怍 was built under R version 4.0.4
Attaching package: 惁梔zoo惁怍

The following objects are masked from 惁梔package:base惁怍:

    as.Date, as.Date.numeric
```

Hide

```
sude$temp <- na.approx(sude$temp)
sude$dewp <- na.approx(sude$dewp)
sude$wdsp <- na.approx(sude$wdsp)
sude$gust <- na.approx(sude$gust)
```

Now for temp, dewp, wdsp, and gust, I originally thought about simply replacing the missing values with mean values but I'm not sure if a mean value across 16 years would be a good idea for so many NA values; it might muddle the data. Instead, after doing a little research on other cleaning techniques, I learned about interpolation which linearly approximates missing values between values the data already has. For this we need the zoo package

Hide

```
#exploration function 3
head(sude)
```

	elvt <dbl>	lat <dbl>	lon <dbl>	city <fctr>	date <date>	hr <int>	prcp <dbl>	stp <dbl>	gbrd <dbl>	▶
10129	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	0	0	982.6	0	
10130	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	1	0	983.1	0	
10131	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	2	0	983.5	0	
10132	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	3	0	983.6	0	
10133	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	4	0	983.3	0	
10134	237	-6.835777	-38.31158	SÃ£o GonÃ§alo	2009-01-01	5	0	983.1	0	

6 rows | 1-10 of 15 columns

Hide

```
tail(sude)
```

	elvt <dbl>	lat <dbl>	lon <dbl>	city <fctr>	date <date>	hr <int>	prcp <dbl>	stp <dbl>	gbrd <dbl>	▶
9772587	777	-23.52389	-46.86945	Barueri	2015-12-31	18	0	924.2	0	
9772588	777	-23.52389	-46.86945	Barueri	2015-12-31	19	0	923.8	0	
9772589	777	-23.52389	-46.86945	Barueri	2015-12-31	20	0	923.3	0	
9772590	777	-23.52389	-46.86945	Barueri	2015-12-31	21	0	923.6	0	
9772591	777	-23.52389	-46.86945	Barueri	2015-12-31	22	0	923.9	0	
9772592	777	-23.52389	-46.86945	Barueri	2015-12-31	23	0	924.9	0	

6 rows | 1-10 of 15 columns

I believe most of the cleaning is now complete and now to further familiarize ourselves with the data. With the head()/tail() functions, we can see a preview of the beginning and end of the data so we can easily understand what one instance of the data looks like. An instance in this context is a snapshot of weather data every hour. This is a good time to see if there are any unreasonable data points at the ends of the data. We seem to be fine in this case. One interesting observation is that prcp and gbrd are 0s on the first hour and prcp, wdsp, wdct, and gust are 0 on the last day.

Hide

```
#exploration function 4
head(sort(table(city)))
```

```
city
   JanuÃ¡ria  Sete Lagoas   TrÃªs Rios    Rio Claro    Saquarema Silva Jardim
        2400         2712         2760         2880         9480         9600
```

Hide

```
tail(sort(table(city)))
```

```
city
      Duque de Caxias     Campos do JordÃ£o                 Bauru
               122256                127584                132264
          SeropÃ©dica Campos dos Goytacazes        Rio de Janeiro
               143376                160584                302352
```

Next, I originally wanted to use the table() function to explore the conditional relationships between variables but due to the number of columns and magnitude of this data set, I decided to explore just one variable, city. Using table(), we are able to see how many instances are from each city. There is still a large number of cities, however, so I looked at the ones with the least and most amount of instances with a combination of sort and the head and tail functions. This was we are able to see which cities the data is most and least representative of. JanuÃ¡ria is at the bottom of this with 2400 observations and, as expected, Rio de Janeiro is at the top with 302352.

Hide

```
#exploration function 5
sort(cor(sude[sapply(sude,is.numeric)])[,8])
```

```
       elvt         lon        prcp        hmdy         lat        wdct 
-0.21429678 -0.02427632 -0.01273416  0.06814771  0.07424163  0.20797740 
         hr        wdsp        gust        gbrd        dewp         stp 
 0.30582731  0.32214068  0.43146903  0.53082846  0.62560459  0.67515978 
       temp 
 1.00000000 
```

To aid with feature selection the final point of data exploration was with the cor() function which allows us to see the associations between numerical variables in the data set. There are 15 columns making it quite large of an output and difficult to focus on temperature, our target, I decided to print only temp’s correlations with the other variables. By sorting this output, we can see the variables with the highest magnitude, excluding temp of course, are stp, dewp, gbrd, and gust which just slightly falls below 0.5. We will focus mostly on these variables.

We have gotten a great look and feel of the numbers but let’s dive deeper into how the relate to one another with some visualizations.

Hide

```
#exploration graph 1
top_cities = subset(sude, city == "Rio de Janeiro"| city == "Campos dos Goytacazes" | city == "S
eropÃ©dica" | city == "Bauru" | city == "Campos do JordÃ£o")

require(tidyr)
```

```
Loading required package: tidyr
package 惢拸tidyr惢拃 was built under R version 4.0.4
```

Hide

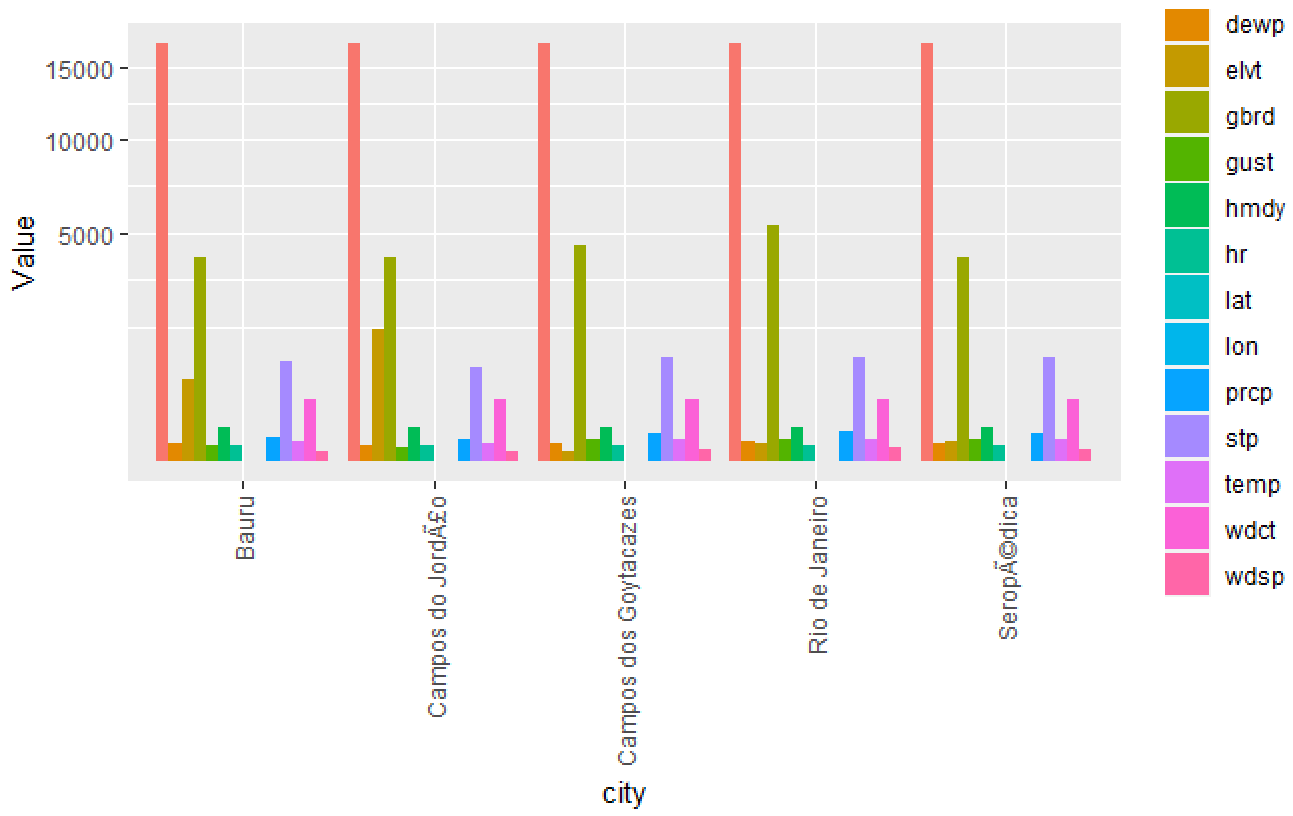
```
dat2 <- top_cities %>%
  gather(Total, Value, -city)
```

```
attributes are not identical across measure variables;
they will be dropped
```

Hide

```
ggplot(dat2, aes(x = city, y = Value, group = Total, fill = Total)) +
  geom_col(position = "dodge") + scale_y_continuous() +
  scale_y_sqrt() + theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) + scale_x_discret
e()
```

```
Scale for 'y' is already present. Adding another scale for 'y', which will
replace the existing scale.
```



The first way I decided to explore the data was to look at the various weather factors for cities that had the highest number of observations, which we found earlier. These are Rio de Janeiro, Campos dos Goytacazes, SeropÃ©dica, Bauru, and Campos do JordÃ£o. This visualization is useful as it can help us compare some of the most representative cities of the data set. We see that variables such average temperatures or lon/lat are generally the same but the other variables vary a little. Let's look at the most noticable. Rio, has the highest gbrd, Campos do JordÃ£o has the highest elevation, and Rio and Campos dos Goytacazes seem to be very close for stp. Though these cities did not have high variance for comparison, this is a good thing as this means the data is more uniform and balanced for modeling.

Hide

```
#exploration graph 2
# for aggregation
library(lubridate)
```

```


Attaching package: 拹牠lubridate拹牪

The following objects are masked from 拹牠package:base拹牪:

    date, intersect, setdiff, union
```

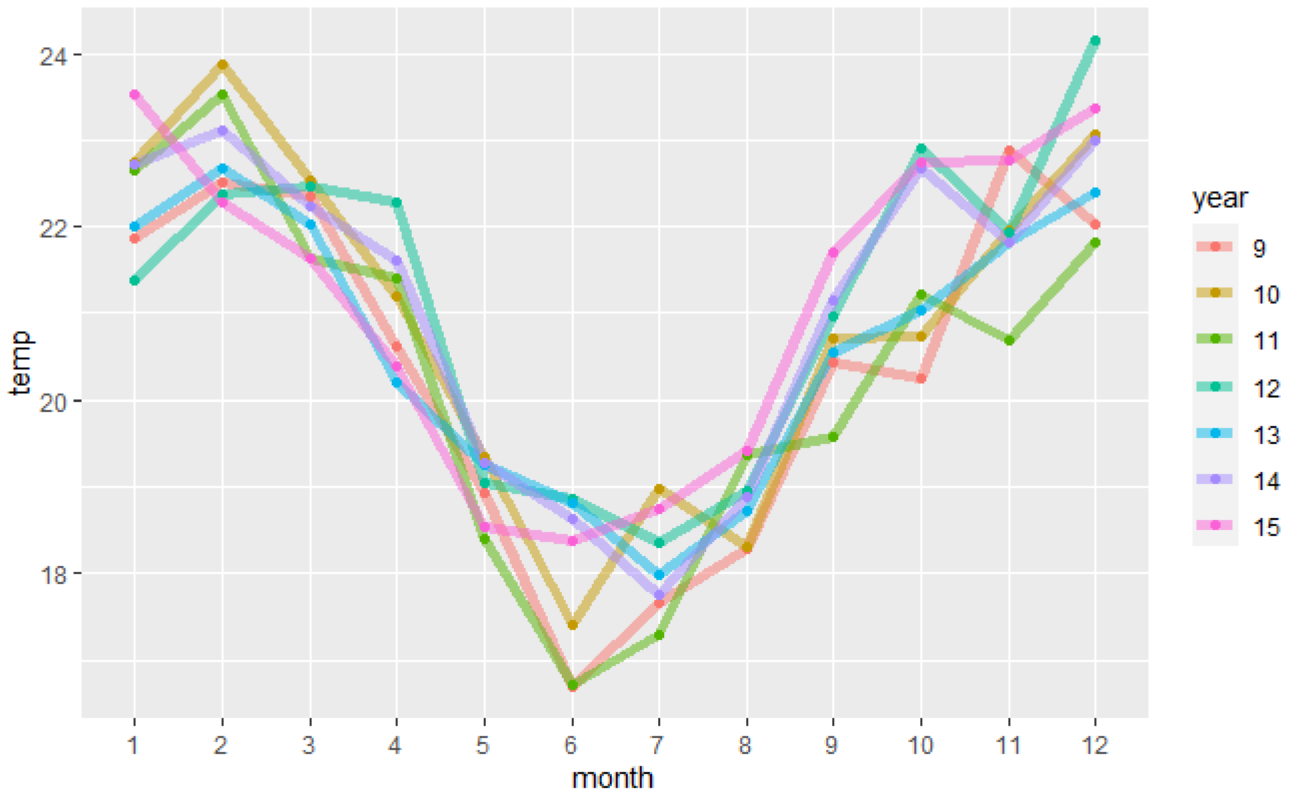
Hide

```
library(dplyr)
```

```
package 惚梔dplyr惚怍 was built under R version 4.0.4
Attaching package: 惚梔dplyr惚怍

The following objects are masked from 惚梔package:stats惚怍:

    filter, lag

The following objects are masked from 惚梔package:base惚怍:

    intersect, setdiff, setequal, union
```

Hide

```
year<-as.numeric(format(sude$date, "%y"))
month<-as.numeric(format(sude$date, "%m"))

for_plot=aggregate(temp ~ + month + year, data =sude, FUN=mean)
for_plot$month = as.factor(for_plot$month)
for_plot$year = as.factor(for_plot$year)

ggplot(for_plot, aes(x=month, y=temp)) + geom_line(aes(group=year, color=year),size=2,alpha=0.5)
+ geom_point(aes(group=year, color=year))
```



Another aspect I wanted to explore was how the average temperatures changed over time. For this I plotted the temperature as a line graph for each year, from 2009 to 2015, against the months of each respective year. It seems that the coolest year was arguably 2011 and the hottest for the beginning 2010, the middle 2012 or 2013, and the end 2012. We see a clear pattern of high temperatures at the beginning of the year and ends and low temperatures where it dip towards June and July.

Now that we have explored the data, let’s select our features and build our models.

Modeling (Linear regression, kNN Regression,)

Hide

```
#feature selecting decisions
library(caret)
```

```
Loading required package: lattice
```

Hide

```
library(mlbench)
corMatrix <- cor(cor(sude[sapply(sude,is.numeric)]))
findCorrelation(corMatrix, cutoff=0.5, verbose=TRUE)
```

```
Compare row 8  and column  6 with corr  0.734 
  Means:  0.397 vs 0.301 so flagging column 8 
Compare row 6  and column  10 with corr  0.687 
  Means:  0.361 vs 0.283 so flagging column 6 
Compare row 10  and column  9 with corr  0.77 
  Means:  0.367 vs 0.279 so flagging column 10 
Compare row 9  and column  1 with corr  0.546 
  Means:  0.219 vs 0.265 so flagging column 1 
Compare row 2  and column  3 with corr  0.878 
  Means:  0.325 vs 0.268 so flagging column 2 
Compare row 13  and column  7 with corr  0.523 
  Means:  0.352 vs 0.246 so flagging column 13 
Compare row 7  and column  4 with corr  0.656 
  Means:  0.315 vs 0.216 so flagging column 7 
All correlations <= 0.5 
[1]  8  6 10  2 13  7  1
```

Previously, some early feature selection with the correlation function which helped us pick out stp, dewp, gbrd, and gust as the highest correlated variables with temperature, all above 0.4. However, before we move on, let’s perform some feature selection with caret for more input on feature selection. The findCorrelation() function returns a list of columns that are suggested to be removed due to multicollinearity. I tried exploring other feature selection options such as recursive or FSelector but for some reason, most likely the size of the data, the RStudio environment either started to hang or froze. Utilizing the output of the findCorrelation function, however, we can still narrow down our features. The the features with the highest correlations are 6 with 8 and 9, 2 with 3, and 13 with 7. I will drop 6 and not 8 and 9 as hr doesn’t seem very useful to predict temp, 9 is fine because it is correlated with temp, will drop 2 and 3, and finally will drop 7 instead of 13 as prcp had many more NAs during data cleaning than wdsp. So our best predictors are stp, dewp, gbrd, gust but more weather factors can always add a little significance to the model so I decided to include wdsp, wdct, and hmdy as well.

Note: kNN and SVM were attempted with the large data set of 6 million observations but due to hardware limitations and the fact the these model bog down at higher dimension it simply wasn’t feasible. I truncated the sample to 30000 for train and test, which is still well above the requirement. Though the 6 million worked for linear regression and decision tree, the same train/test vectors are used for comparison’s sake.

Hide

```
#divide train and test
set.seed(1234)
i <- sample(1:30000, 30000*0.75, replace=FALSE)
train <- sude[i,]
test <- sude[-i,]

#build models

#mulitple linear regression model
lm1 <- lm(temp~stp+dewp+gbrd+gust+wdsp+wdct+hmdy, data=train)

#SVM regression model
library(e1071)
svm1 <- svm(temp~stp+dewp+gbrd+gust+wdsp+wdct+hmdy, data=train, kernel="linear", cost=10, scale=
FALSE)
```

```


WARNING: reaching max number of iterations
```

Hide

```
#decision tree model
library(tree)
tree1 <- tree(temp~stp+dewp+gbrd+gust+wdsp+wdct+hmdy, data=train)
```

Metrics/Evaluation

Hide

```
#evaluate linear regression
summary(lm1)
```

```

Call:
lm(formula = temp ~ stp + dewp + gbrd + gust + wdsp + wdct + 
    hmdy, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.8385 -0.4546 -0.0021  0.1898  6.5836 

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)    
(Intercept)  2.087e-03  1.015e-02    0.206   0.8371    
stp          2.808e-02  5.678e-05  494.490  < 2e-16 ***
dewp         8.548e-01  3.851e-03  221.977  < 2e-16 ***
gbrd         3.308e-04  8.118e-06   40.747  < 2e-16 ***
gust         2.330e-02  5.150e-03    4.525 6.08e-06 ***
wdsp        -2.933e-01  1.062e-02  -27.622  < 2e-16 ***
wdct         2.284e-04  1.084e-04    2.107   0.0351 *  
hmdy        -2.596e-01  8.002e-04 -324.447  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8482 on 22492 degrees of freedom
Multiple R-squared:  0.9958,	Adjusted R-squared:  0.9958 
F-statistic: 7.542e+05 on 7 and 22492 DF,  p-value: < 2.2e-16
```

Hide

```
pred1 <- predict(lm1, newdata=test)
mse1 <- mean((pred1 - test$temp)^2)
cor1 <- cor(pred1, test$temp)
print(paste("lm1 mse: ", mse1))
```

```
[1] "lm1 mse:  8.09477965425756"
```

Hide

```
print(paste("lm1 rmse: ", sqrt(mse1)))
```

```
[1] "lm1 rmse:  2.84513262507349"
```

Hide

```
print(paste("lm1 cor: ", cor1))
```

```
[1] "lm1 cor:  0.928043837974655"
```

Looking at summary output of the multiple linear regression model, we see that all of our predictors except wdct were statistically significant. The RSE in this case was 0.8482 which is actually quite smasll which is good. The Adjusted-R^2 is 0.9958 which mean the model fit the data very well. The F-statistic is very large and the associated p-value is very small which means the predictors and temp are veryr elated and the relationship is significant. Finally, the mse is quite small but as it difficult to predict let's look at the rmse, which is in units of temperature: 2.85. This mean the temperature was off by 2.85 degrees, which is very good in the context. The correlation is 0.928, a very high value close to 1, which indicates a strong positive relationship.

Hide

```
summary(svm1)
```

```


Call:
svm(formula = temp ~ stp + dewp + gbrd + gust + wdsp + wdct + hmdy, 
    data = train, kernel = "linear", cost = 10, scale = FALSE)


Parameters:
   SVM-Type:  eps-regression 
 SVM-Kernel:  linear 
       cost:  10 
      gamma:  0.1428571 
    epsilon:  0.1 


Number of Support Vectors:  15583
```

Hide

```
#pred2 <- predict(svm1, newdata=test)
mse2 <- mean((pred2 - test$temp)^2)
cor2 <- cor(pred2, test$temp)
print(paste("svm1 mse: ", mse2))
```

```
[1] "svm1 mse:  1152.92814232048"
```

Hide

```
print(paste("svm1 rmse: ", sqrt(mse2)))
```

```
[1] "svm1 rmse:  33.9547955717669"
```

Hide

```
print(paste("svm1 cor: ", cor2))
```

```
[1] "svm1 cor:  0.525937361067351"
```

SVM took a turn and didn't perform so well. It took the longest to build and predicting took even longer; for this reason it is the worst in terms of efficiency but also left little time for hyperparameter optimization. It could have been better but as memory has to scale quadratically with the number of data points, this algorithm wasn't the best option for a large data set. The mse is outrageous here and so the rmse was also very bad at 33.95; this means the model was off by 33.95 degrees on average. The correlation of 52.59% isn't terrible but isn't strong either; it's just okay.

Hide

```
#evaluate decision  tree
summary(tree1)
```

```

Regression tree:
tree(formula = temp ~ stp + dewp + gbrd + gust + wdsp + wdct + 
    hmdy, data = train)
Variables actually used in tree construction:
[1] "stp"  "hmdy"
Number of terminal nodes:  3 
Residual mean deviance:  4.302 = 96790 / 22500 
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
-10.5000  -0.8337   0.0000   0.0000   0.9025   7.2660 
```

Hide

```
pred3 <- predict(tree1, newdata=test)
mse3 <- mean((pred3 - test$temp)^2)
cor3 <- cor(pred3, test$temp)
print(paste("tree1 mse: ", mse3))
```

```
[1] "tree1 mse:  39.4890377263862"
```

Hide

```
print(paste("tree1 rmse: ", sqrt(mse3)))
```

```
[1] "tree1 rmse:  6.28403037280901"
```

Hide

```
print(paste("tree1 cor: ", cor3))
```

```
[1] "tree1 cor:  0.741186855651097"
```

Finally, the decision tree performed better was not as good as the linear regression. Looking at the summary output we see that the model decided to use stp and hmdy, some of best features. The mse here was lower than SVM's but higher than linear regression. An mse of 39.49 translates to an rmse of 6.284 which means the model was off by 8.284 degrees on average. Finally, the correlation of 0.741 is moderately high which indicates a moderately strong positive relationship.

Ranking the algorithms from best to worst correlation and rmse, we have multiple linear regression, the decision tree, and support vector machines. Linear regression assumes the relationship the relationship between the target and predictors is linear and in this case that excelled. SVM performs dot product of training samples so having a more predictors increased the complexity and decreased the run time cause it to perform worse efficiency-wise than the other algorithms. More time spent on hyperparameter optimization may have helped but with insane run times, this time was unavailable. The decision tree wasn't too bad but linear regression outperforms regression trees when the underlying function is linear and in the case of temperature this might be why linear regression performed much better. The data here was linear and not too complex so the tree may have overfit. Pruning may have helped but I wanted to compare the base tree.

In the end multiple linear regression won being the relatively simple linear algorithm. This project in the end was a great opportunity to not only compare various regression algorithms but also explore the southeast region of Brazil. Though the data is localized to Brazil, the application of temperature prediction is known to be valuable without any explanation. We already have amazing model but the more variables are explored the more we can fine tune weather models. I think the biggest takeaway from exploring is data is the weight of air pressure in predicting temperature. Moreover, in exploring multicolinearity we also learned that precipitation is related to wind speed, hour of day is strongly related to solar radiation and air pressure, and unsurprisingly, but refreshingly, that longitude is correlated with latitude. This project also was good practice for data visualization using ggplot2. All in all, I learned about Southeast Brazil's weather patterns and see how it is important to study various weather conditions to build more accurate models for the future.