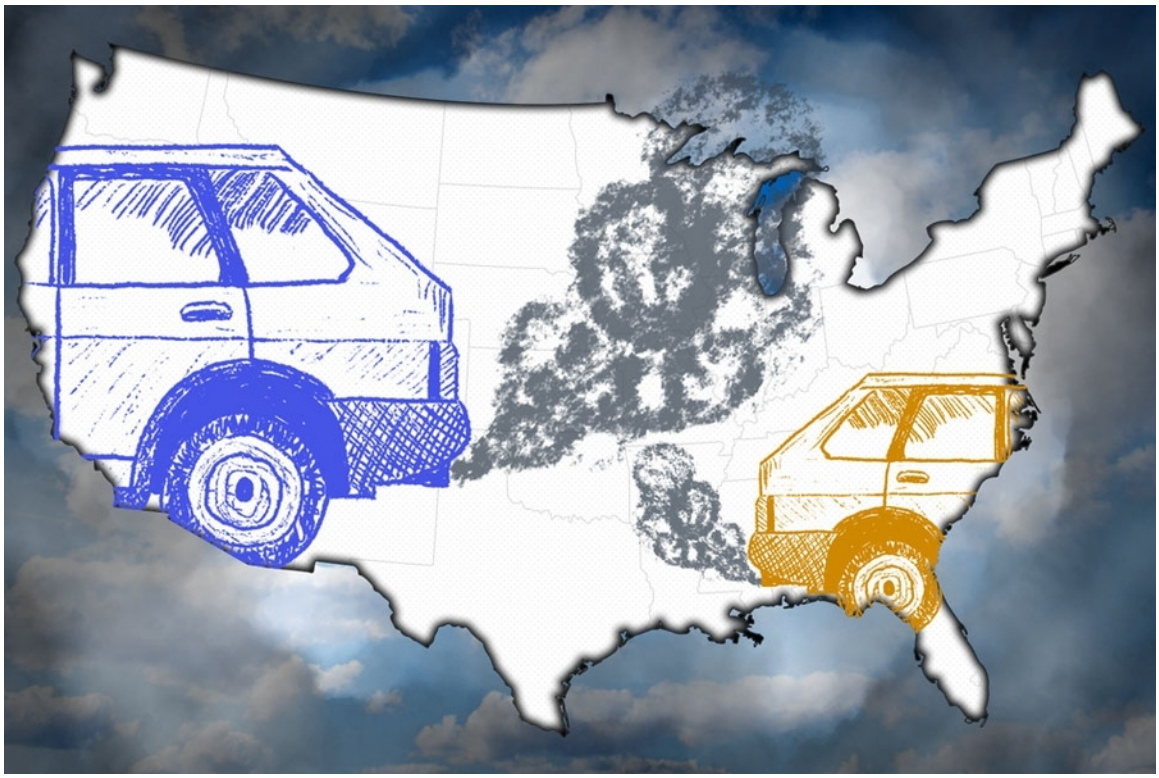


SO_2 Concentrations in the U.S.

SPATIAL AND REGRESSION ANALYSIS

Rob Young & Ammar Alzureiqi | SS4864 | Due: Dec 20/20

Contents

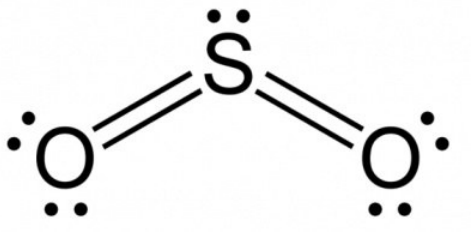
Data Description and Processing

The Study

This data gives air pollution and related values for 41 U.S. cities and were collected from U.S. government publications. The data are means over the years 1969-1971.

What is SO2?

SO_2 is the chemical formula for a molecule called Sulfur Dioxide. It is a toxic gas responsible for the smell of burnt matches. Wikipedia SO2*



This study concerns the concentrations of Sulphur Dioxide in micrograms per cubic meter in the air. The reason for studying SO_2 concentrations is because of the adverse implications on our health and the environment.

Short-term exposures to SO_2 can harm the human respiratory system and make breathing difficult. People with asthma, particularly children, are sensitive to the effects of SO_2.

SO_2 that is emitted in high concentrations into the air generally also lead to the formation of other sulfur oxides (SO_x). The following reaction is one of the ways in which sulfur dioxide can harm the environment.

$$SO_2 + \frac{1}{2}\, O_2 \rightarrow SO_3$$

This reaction occurs in the air and produces the gas sulfur trioxide which then allows for the following reaction to occur. Wikipedia*

$$SO_{3(g)} + H_2O_{(l)} \rightarrow H_2SO_{4(aq)}$$

Where *aq* is an aqueous solution. This slow but sure process creates Sulfuric Acid in the air, which may lead to acid rain. This is an example of just one the ways Sulfur Dioxide can react in the air. https://www.epa.gov/so2-pollution*

SO_x can react with other compounds in the atmosphere to form small particles. These particles contribute to particulate matter (PM) pollution. Small particles may penetrate deeply into the lungs and in sufficient quantity can contribute to health problems. https://www.epa.gov/so2-pollution*

Purpose of Our Study

The purpose of this data exploration is for two reasons, we will first look to see if it is possible to accurately predict the response given the variables with regard to the bias-variance trade off. Which is shown in the equation below,

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)$$

This expected value is the Mean Squared Error calculated from the test data. Because we can overfit training data quite easily, our test MSE is most important to us. For every model we create we will show the training and test MSE's and variance of the model.

The next most important topic of study is what variables are most important to the response. For spatial analysis, we will consider whether spatial location is important to the response variable. For multiple linear regression, we will consider all of the variables and the 5 regions of the U.S.

Acquiring the Data

First thing to be done is to set the workplace directory to wherever the modified air.dat is stored. We have to modify the original dataset, air.dat, to have no spaces in any of the names of the cities. This way we can extract the data by spaces. We call this new dataset air.dat.modif.

To read air.dat.modif we used the scan() function to simply be able to get all the information into R. While using read.table() or read.delim() functions there were random numbers of spaces between all the information and these functions could not read DAT files well enough to return everything. Once the file was scanned, we sorted all the data by the order which it came in.

```
getwd()

## [1] "C:/Users/19059/Desktop/4864 Assignments/Course Project, Air Pol
lution"

setwd("C:/Users/19059/Desktop/4864 Assignments/Course Project, Air Poll
ution")

data <- scan("air.dat", what = "character")
City <- data[seq(9, 329, by = 8)]
SO2 <- as.numeric(data[seq(10, 330, by = 8)])
Temp <- as.numeric(data[seq(11, 331, by = 8)])
Man <- as.numeric(data[seq(12, 332, by = 8)])
Pop <- as.numeric(data[seq(13, 333, by = 8)])
Wind <- as.numeric(data[seq(14, 334, by = 8)])
Rain <- as.numeric(data[seq(15, 335, by = 8)])
RainDays <- as.numeric(data[seq(16, 336, by = 8)])
air.data <- data.frame(City, SO2, Temp, Man, Pop, Wind, Rain, RainDays)
```

The Response and Variables

City: the City documented in the experiment

SO2: Sulfur dioxide content of air in micrograms per cubic meter

Temp: Average annual temperature in degrees Fahrenheit

Man: Number of manufacturing enterprises employing 20 or more workers

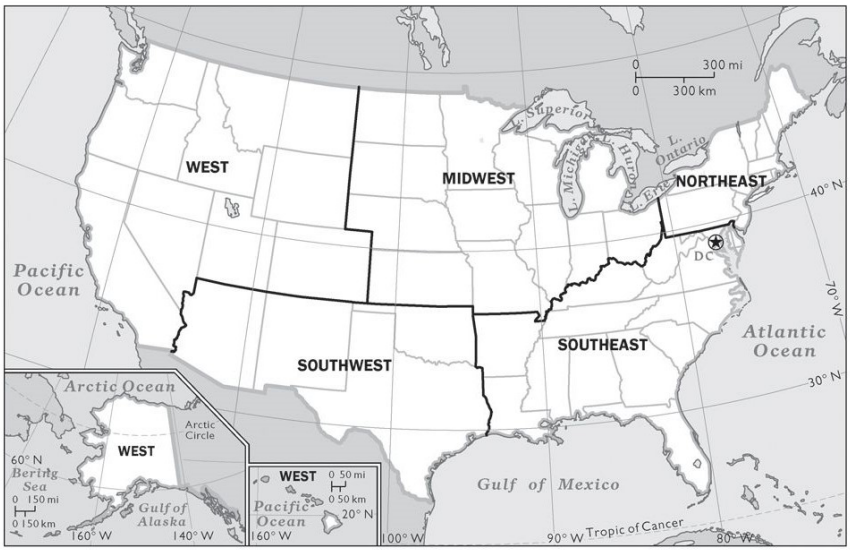
Pop: Population size in thousands from the 1970 census

Wind: Average annual wind speed in miles per hour

Rain: Average annual precipitation in inches

RainDays: Average number of days with precipitation per year

We only have one factor variable, City, and what we decide to do with it will drastically change the type of analysis we perform on the data. We can either consider City to be a categorical variable of 5 levels; the Northeast, Southwest, West, Southeast, and Midwest. These are the 5 regions that National Geographic defines the U.S. by.



By this process we allow the data to be explored through multiple linear regression and various parametric and nonparametric analysis.

```
air.data$City <- as.factor(c("SW", "SE", "W", "W", "NE", "NE", "W", "SE
", "SE", "SE", "MW", "MW", "MW", "MW", "SE", "SE", "NE", "MW", "MW", "M
W", "MW", "MW", "SW", "NE", "NE", "MW", "MW", "MW", "NE", "NE", "NE", "
SE", "SE", "SW", "SW", "W", "SE", "SE", "W", "SE", "MW"))
```

The other option we have is to consider the Cities as Longitude and Latitude coordinates and consider this as a spatial data set.

```
Latitude <- c(33.448376, 34.746483, 37.773972, 39.742043, 41.763710,
34.225727, 38.900497, 30.332184, 25.761681, 33.753746, 41.881832,
39.791000, 41.619549, 37.697948, 38.328732, 29.951065, 39.299236,
42.331429, 44.954445, 39.099724, 38.627003, 41.257160, 35.106766,
42.652580, 42.880230, 39.103119, 41.505493, 39.983334, 39.952583,
40.440624, 41.825226, 35.117500, 36.174465, 32.779167, 29.749907,
40.758701, 36.850769, 37.541290, 47.608013, 32.776566, 43.038902)
Longitude <- c(-112.074036, -92.289597, -122.431297, -104.991531, -
72.685097, -77.944710, -77.007507, -81.655647, -80.191788, -84.386330,
-87.623177, -86.148003, -93.598022, -97.314835, -85.764771, -90.071533,
-76.609383, -83.045753, -93.091301, -94.578331, -90.199402, -95.995102,
-106.629181, -73.756233, -78.878738, -84.512016, -81.681290, -
82.983330, -75.165222, -79.995888, -71.418884, -89.971107, -86.767960,
-96.808891, -95.358421, -111.876183, -76.285873, -77.434769, -
122.335167, -79.930923, -87.906471)

air.data$Longitude <- Longitude
air.data$Latitude <- Latitude
```

Training and Testing Data Split

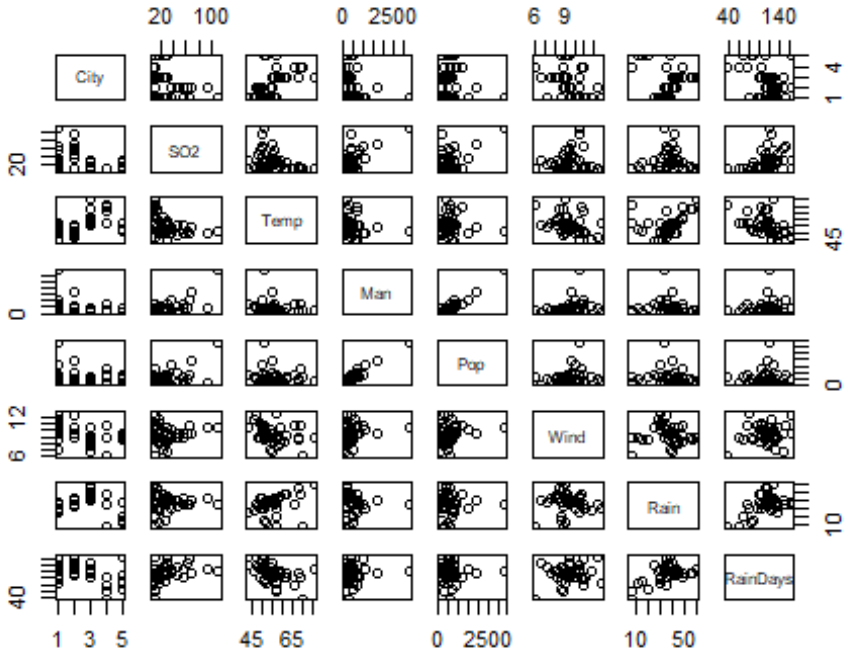
We should create training and testing datasets to be used for the whole study before we begin.

```
set.seed(192835647)
train.obs <- sample(1:41, 25)
test.obs <- c(1:41)[-train.obs]
train.data <- air.data[train.obs,]
test.data <- air.data[test.obs,]
```

Preliminary Data Exploration

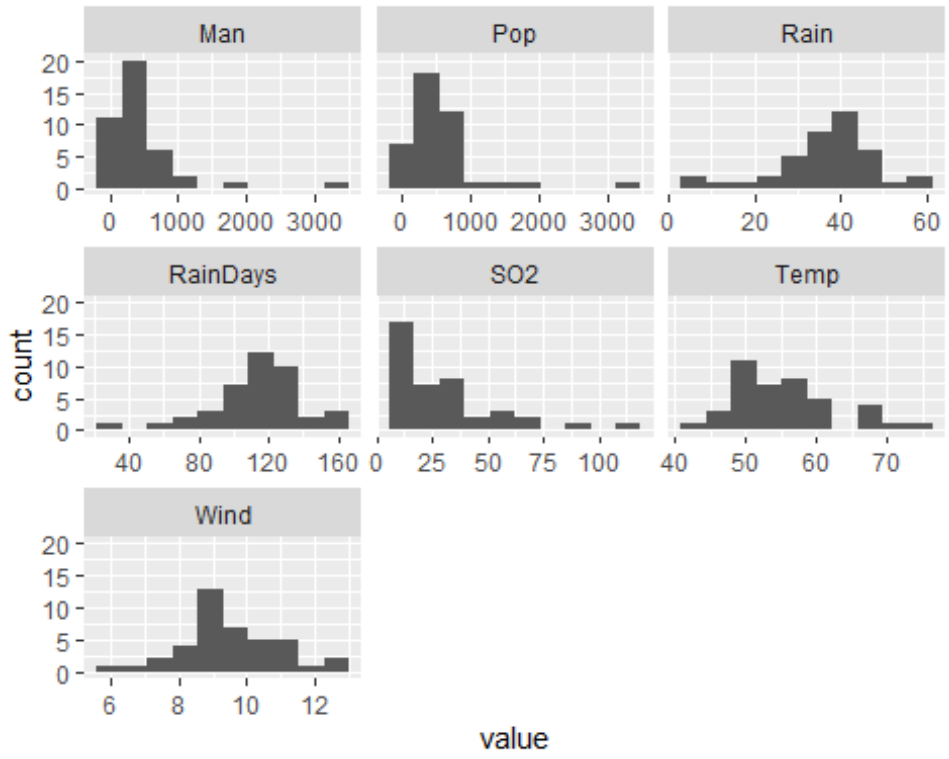
If we take a broad look at the relationships of all the variables, we can narrow it down to a select few that seem to be stronger than others.

```
library(ggplot2)

library(gridExtra)

library(tidyr)

plot(air.data)
```



We see there are potential linear relationships between Man and Pop, Temp and Rain, SO2 and Man, SO2 and Pop, SO2 and Wind, SO2 and Rain. We also take a look at the distributions of the variables below.

```
ggplot(gather(air.data[,-1]), aes(value)) + geom_histogram(bins = 10) +
facet_wrap(~key, scales = 'free_x')
```

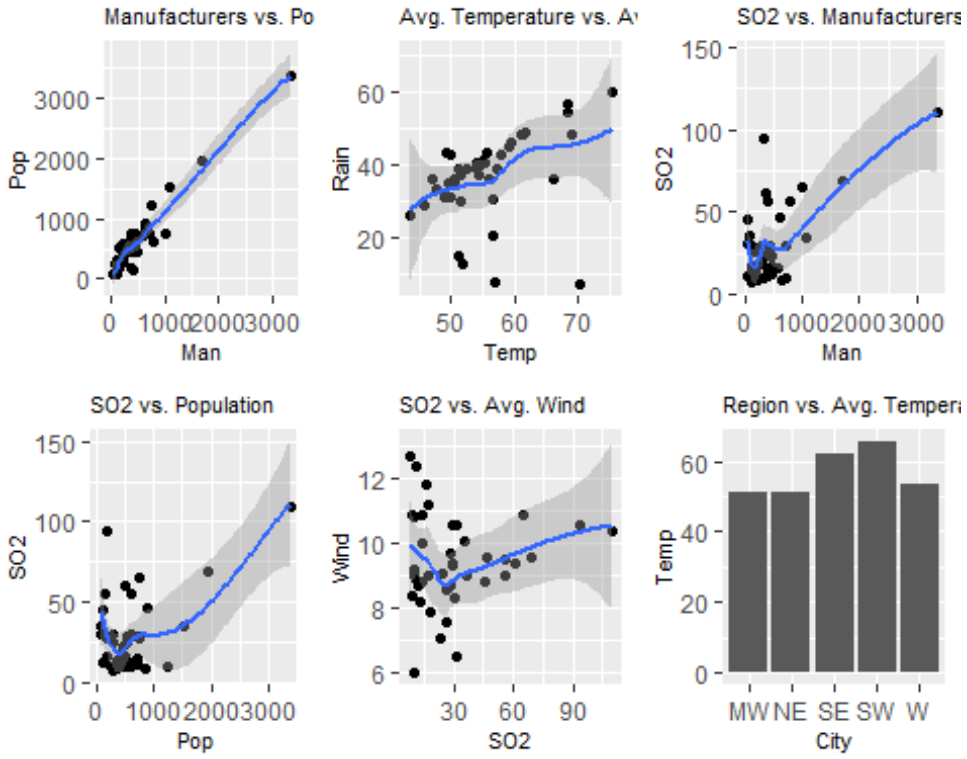


We notice that SO_2 concentration is not normal at all. Although this may not matter to us for multiple linear regression, because we only need the response to normally distributed conditional on its' predictors, this may cause trouble for spatial analysis.

Below, we take a closer look at some of the relationships that we think may be most significant in our study.

```
p1 <- ggplot(air.data, aes(Man, Pop)) + geom_point() + geom_smooth() +
ggtitle("Manufacturers vs. Population") + theme(plot.title = element_te
xt(size = 8),axis.title.x = element_text(size = 8),axis.title.y = eleme
nt_text(size = 8))
p2 <- ggplot(air.data, aes(Temp, Rain)) + geom_point() + geom_smooth()
+ ggtitle("Avg. Temperature vs. Avg. Rain") + theme(plot.title = elemen
t_text(size = 8),axis.title.x = element_text(size = 8),axis.title.y = e
lement_text(size = 8))
p3 <- ggplot(air.data, aes(Man, SO2)) + geom_point() + geom_smooth() +
ggtitle("SO2 vs. Manufacturers") + theme(plot.title = element_text(size
= 8),axis.title.x = element_text(size = 8),axis.title.y = element_text(
size = 8))
p4 <- ggplot(air.data, aes(Pop, SO2)) + geom_point() + geom_smooth() +
ggtitle("SO2 vs. Population") + theme(plot.title = element_text(size =
8),axis.title.x = element_text(size = 8),axis.title.y = element_text(si
ze = 8))
p5 <- ggplot(air.data, aes(SO2, Wind)) + geom_point() + geom_smooth() +
ggtitle("SO2 vs. Avg. Wind") + theme(plot.title = element_text(size = 8
```

```
),axis.title.x = element_text(size = 8),axis.title.y = element_text(siz
e = 8))
p6 <- ggplot(air.data, aes(x = City, y = Temp)) + stat_summary(fun.y =
"mean", geom = "bar") + ggtitle("Region vs. Avg. Temperature ") + theme
(plot.title = element_text(size = 8),axis.title.x = element_text(size =
8),axis.title.y = element_text(size = 8))

grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 3, nrow = 2)
```



We notice that we have two pairs of variables that could be correlated. Intuitively, we consider Pop and Man to be highly correlated and Rain and RainDays to be highly correlated. To be sure, lets look at the Pearson Correlation matrix of all the variables.

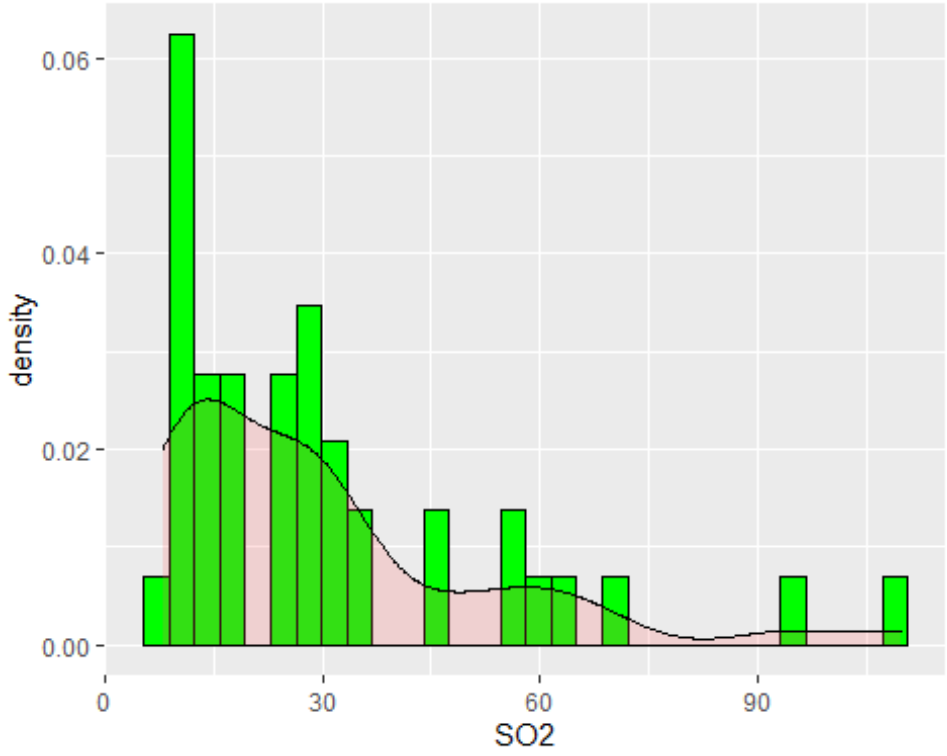
```
round(cor(air.data[,-1]), 2)
```

```
            SO2  Temp   Man   Pop  Wind  Rain RainDays
SO2        1.00 -0.43  0.64  0.49  0.09  0.05     0.37
Temp      -0.43  1.00 -0.19 -0.06 -0.35  0.39    -0.43
Man        0.64 -0.19  1.00  0.96  0.24 -0.03     0.13
Pop        0.49 -0.06  0.96  1.00  0.21 -0.03     0.04
Wind       0.09 -0.35  0.24  0.21  1.00 -0.01     0.16
Rain       0.05  0.39 -0.03 -0.03 -0.01  1.00     0.50
RainDays   0.37 -0.43  0.13  0.04  0.16  0.50     1.00
```

The only worrying correlation is the predictors Man and Pop. We should remove one of them before starting, we will delete Pop, since Man seems to have a better correlation with SO_2.

Lastly, we will take a formal look at the response variable.

```
ggplot(air.data, aes(x = SO2)) +  geom_histogram(aes(y = ..density..),
colour = "black", fill = "green") + geom_density(alpha = .2, fill = "#F
F6666")
```



We see that our response variable is not normally distributed. We should consider testing if the response variable came from a normal distribution by the Shapiro-Wilks test.

The null hypothesis of the Shapiro-Wilks test is given below,

$$H_0: Population\ is\ Normally\ Distributed$$

```
shapiro.test(air.data$SO2)

##
##  Shapiro-Wilk normality test
##
## data:  air.data$SO2
## W = 0.81165, p-value = 9.723e-06
```

There is strong evidence that the response variable is not normally distributed. It’s true distribution cannot be determined.

Multiple Linear Regression Analysis

We will begin this section by fitting a linear model that includes all of the variables, except Pop.

```
linear.model <- lm(SO2 ~ City + Temp + Man + Wind + Rain + RainDays, da
ta = train.data)
summary(linear.model)

##
## Call:
## lm(formula = SO2 ~ City + Temp + Man + Wind + Rain + RainDays,
##     data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.828  -4.725   2.007   3.932  17.929
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 98.329214  41.072437   2.394  0.03017 *
## CityNE      19.182820   6.204914   3.092  0.00744 **
## CitySE      -2.951549   8.549771  -0.345  0.73472
## CitySW       1.694257  11.694467   0.145  0.88674
## CityW        0.737298   6.944837   0.106  0.91686
## Temp        -0.959420   0.640570  -1.498  0.15494
## Man          0.018584   0.006122   3.035  0.00835 **
## Wind        -4.508091   1.658489  -2.718  0.01587 *
## Rain         0.412065   0.427793   0.963  0.35070
## RainDays    -0.034583   0.141216  -0.245  0.80986
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.762 on 15 degrees of freedom
## Multiple R-squared:  0.7717, Adjusted R-squared:  0.6347
## F-statistic: 5.633 on 9 and 15 DF,  p-value: 0.001726
```

At first glance we see that the City, Man, and Wind variables are the only significant variables. We also see that our R^2 and R^2 adjusted are adequate for a linear model. We also obtain the training and testing Mean Squared Errors of the model.

```
trainMSE <- mean(linear.model$residuals^2)
testMSE <- mean((test.data$SO2 - predict(linear.model, newdata = test.d
ata))^2)

trainMSE

## [1] 57.17618

testMSE
```

```
## [1] 437.9691
```

We will now create a model that does not include the insignificant variables.

```
##
## Call:
## lm(formula = SO2 ~ City + Man + Wind, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.5119  -5.5919   0.8806   6.3522  17.8505
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  51.365635  16.607407   3.093  0.00628 **
## CityNE       21.113615   5.983634   3.529  0.00240 **
## CitySE       -6.237589   6.389482  -0.976  0.34189
## CitySW      -14.197820   7.438620  -1.909  0.07238 .
## CityW        -3.021980   6.555777  -0.461  0.65035
## Man           0.019134   0.006135   3.119  0.00593 **
## Wind         -3.826512   1.590567  -2.406  0.02710 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.869 on 18 degrees of freedom
## Multiple R-squared:   0.72,  Adjusted R-squared:  0.6267
## F-statistic: 7.714 on 6 and 18 DF,  p-value: 0.0003259
```

Although all of our variables are significant, we have seen a large drop in R^2. This is expected because of the nature of R^2, meaning that it would be best with as many variables as possible. But using our adjusted R^2 we see a similar value, indicating that we have not discarded any variables that were of use. But we will perform a formal test of the coefficients now.

$$H_0: Temp,\ Rain,\ and\ RainDays\ have\ coefficients\ of\ zero$$

```
anova(linear.model, linear.model1)
```

```
## Analysis of Variance Table
##
## Model 1: SO2 ~ City + Temp + Man + Wind + Rain + RainDays
## Model 2: SO2 ~ City + Man + Wind
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     15 1429.4
## 2     18 1753.0 -3    -323.6 1.1319 0.3678
```

This anova test justifies our decision to discard the insignificant variables.

Now we will check the training and test MSEs for this new nested model.

```
trainMSE1 <- mean(linear.model1$residuals^2)
testMSE1 <- mean((test.data$SO2 - predict(linear.model1, newdata = test
.data))^2)

trainMSE1

## [1] 70.12028

testMSE1

## [1] 456.5554
```

We see a similar overfitting result that occurred in the full model. Either way, both of these models do not perform well. This could be attributed to the fact that we have so little data to work with. A training set of 25 observations and 16 test observations.

We will test our theory of overfitting on a second training and test set. If our theory is correct, we will see a transfer of error between the training and test MSE either a very small training MSE and huge test MSE or a larger training MSE and smaller test MSE. Both situations mean that the variance of the model is very large.

```
set.seed(123456)
train.obs1 <- sample(1:41, 25)
test.obs1 <- c(1:41)[-train.obs1]
train.data1 <- air.data[train.obs1,]
test.data1 <- air.data[test.obs1,]

linear.model2 <- lm(SO2 ~ City + Man + Wind, data = train.data1)
trainMSE1f <- mean(linear.model2$residuals^2)
testMSE1f <- mean((test.data1$SO2 - predict(linear.model2, newdata = te
st.data1))^2)
trainMSE1f

## [1] 166.325

testMSE1f

## [1] 283.0285

linear.model3 <- lm(SO2 ~ City + Temp + Man + Wind + Rain + RainDays, d
ata = train.data1)
trainMSEf <- mean(linear.model3$residuals^2)
testMSEf <- mean((test.data1$SO2 - predict(linear.model3, newdata = tes
t.data1))^2)
trainMSEf

## [1] 118.1003

testMSEf
```

```
## [1] 475.0163
```

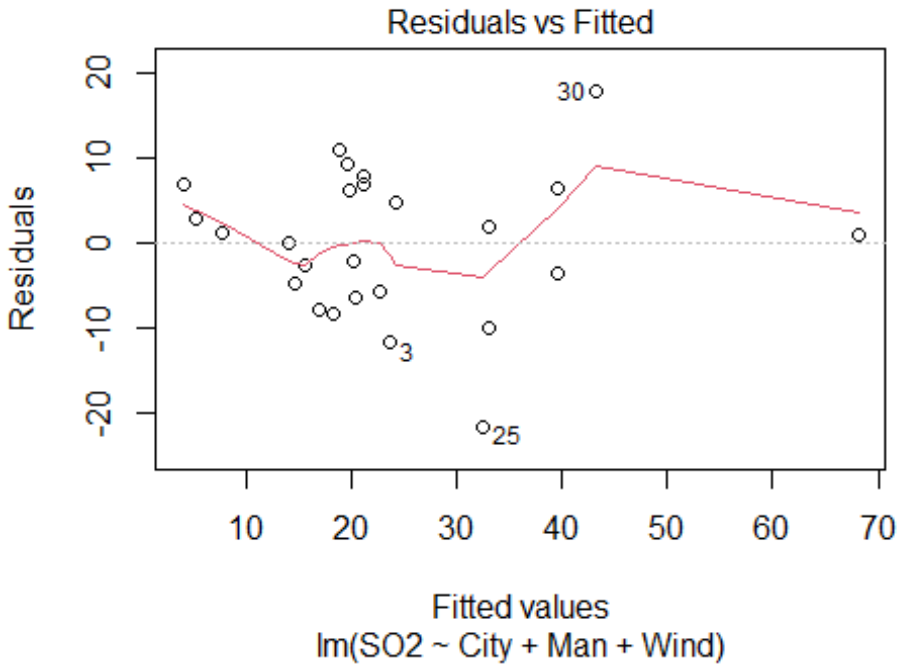
What we see are results that are inconsistent with our first MSEs. But they have the same message, that our models are overfitting the training data by a margin that cannot be ignored.

We consider our nested model to be our best model thus far and so we will have to consider whether it is even a valid linear model.

Residual Diagnostic Tests

We will begin this by looking at the residuals and fitted values of the nested model, this will tell us if the residuals have a constant variance and have a mean of zero.

```
plot(linear.model1, which = 1)
```



The residuals are centered around zero, but our variance does not seem to remain constant but actually disperses as the fitted values grow. Our constant variance (homoscedasticity assumption) may be violated. We should consider a formal test to verify our visual assumptions.

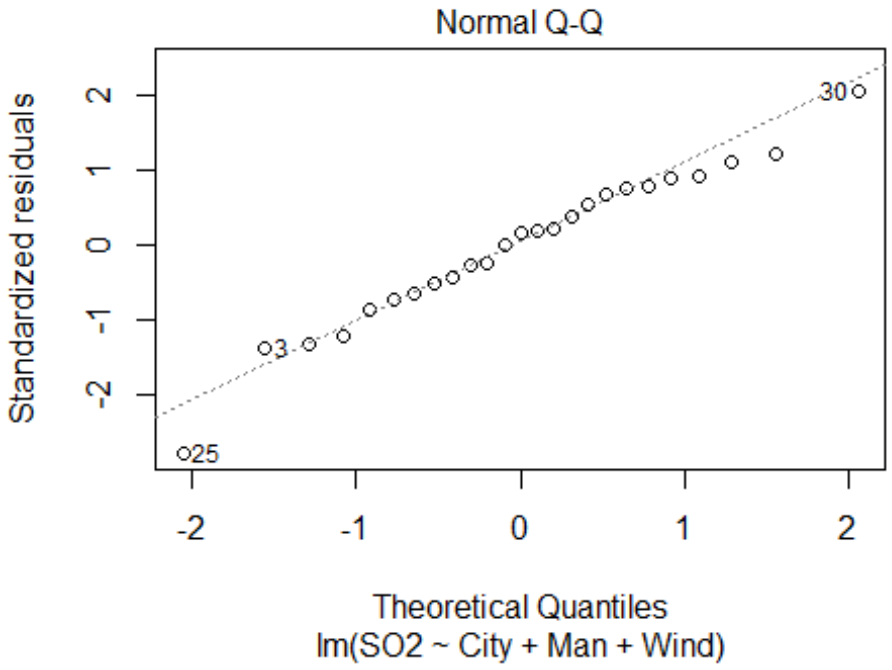
$$H_0: Residual\ Variance\ is\ Homoscedastic$$

```
library(lmtest)
```

```
bptest(linear.model1)

##
##  studentized Breusch-Pagan test
##
## data:  linear.model1
## BP = 7.5851, df = 6, p-value = 0.2701
```

This is the Breusch-Pagan test, which tests whether the constant variance assumption is true. A key assumption made by this test is that the underlying model in linear, such that the response and predictors have a linear relationship. If this is not the case and the true relationship is otherwise, then this test may be invalid. We have not tested that yet and so we will take the test for what it is, this provides sufficient evidence that our nested model has residuals whose variance is constant.

```
plot(linear.model1, which = 2)
```



Our normal QQ-plot of the studentized residuals turns out great. Except for point 25.

Now we will dive into the individual characteristics of the residuals to ensure that our model is not being greatly swayed by certain points.

We will first look at the VIF, variance inflation factors, to determine if any of our variables are highly correlated.

```
library(car)
```

```
vif(linear.model1)
```

```
##          GVIF Df GVIF^(1/(2*Df))
## City 1.486554  4        1.050806
## Man  1.176483  1        1.084658
## Wind 1.335708  1        1.155728
```

From our variance inflation factors we see that we do not have correlated factors and that we dealt with them appropriately at the beginning.

Now we noticed that points 25 and 30 seemed to be largest outliers in the residuals and fitted plot and point 25 appeared to be off the normal QQ-plot. We will check to see if these points have influential properties.

```
hatvalues(linear.model1)
```

```
##         2         3        30        29        16         9
1        33
## 0.1476744 0.2544540 0.2146695 0.7436606 0.1430504 0.1550423 0.509573
0 0.1497958
##        39        24        20         7        34         8        1
2         4
## 0.2534983 0.3074025 0.1736252 0.2510372 0.5077514 0.1508916 0.178238
3 0.2505067
##        18        26        38        15        14         6        2
3        19
## 0.2827689 0.3897006 0.1560919 0.1460664 0.4208314 0.2863974 0.361380
4 0.1869925
##        25
## 0.3788993
```

```
hatvalues(linear.model1) > 3*mean((hatvalues(linear.model1)))
```

```
##     2     3    30    29    16     9     1    33    39    24    20
7    34
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
##     8    12     4    18    26    38    15    14     6    23    19
25
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE
```

We see that from the diagonals of the hat matrix, we do not have any high influence points. Specifically points 30 and 25 are not very close to the highest influence.

Next we will check the standardized residuals to see if any points can be classified as any outlier. We can find the values through a function referenced in the textbook for this class.

```
library(MASS)
```

```r
stdres(linear.model1)
```

```
##            2            3           30           29           16
9
## -0.273502531 -1.375498956  2.041117814  0.176244101 -0.863516710 -0.
512620002
##            1           33           39           24           20
7
## -1.198703734 -0.237424266  1.099370490  0.773438057 -0.712351666  0.
929538410
##           34            8           12            4           18
26
##  0.184333195 -0.006263555  0.765054470 -0.654511612  0.230159781 -1.
301935478
##           38           15           14            6           23
19
##  0.682134836  1.213236243  0.378056507 -0.428400183  0.888620253  0.
541766257
##           25
## -2.765934001
```

```r
stdres(linear.model1) > 3
```

```
##     2     3    30    29    16     9     1    33    39    24    20
7    34
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
##     8    12     4    18    26    38    15    14     6    23    19
25
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE
```

We see that none of the points get near the general cutoff.

Now let's look to see how much our fitted values would change if we were to remove the i^{th} observation. ie: the Cook's Distance.

```r
cooks.distance(linear.model1) > 1
```

```
##     2     3    30    29    16     9     1    33    39    24    20
7    34
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
##     8    12     4    18    26    38    15    14     6    23    19
25
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE
```

```r
cooks.distance(linear.model1) > 0.5
```

```
##     2     3    30    29    16     9     1    33    39    24    20
7    34
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
##     8    12     4    18    26    38    15    14     6    23    19
25
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  T
RUE
```

The general cutoff for a point that is absolutely influential is 1 and none of the points exceed 1. We see that point 25 meets the cutoff to be considered as an influential point.

After our residual analysis we should be wary of point 25 because of it's influential properties on the model. But we must be aware of how small the sample size is. Can we afford to remove a point that is potentially part of a larger population that we do not see? Our analysis suggests nothing out of the ordinary except an arbitrary outlier test that is relative to the other points. We should delve deeper into this to truly understand what should be done about these potential outliers.

Model Transformations

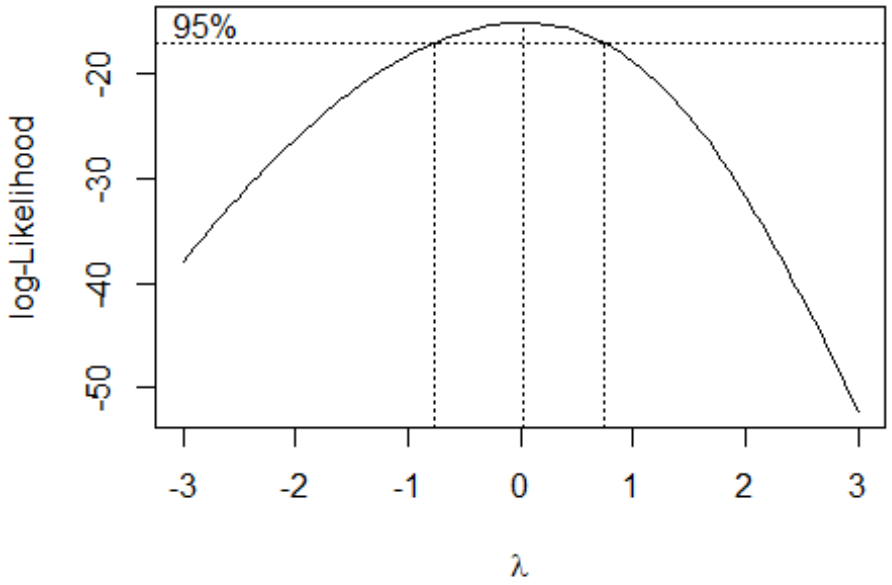
As a precaution, we will attempt to find a transformation that fits the model best, to see if it betters the residual analysis we just did.

We will use the Box-Cox method to see if we can produce better results. Using textbook Regression: Linear Models in Statistics (Springer Undergraduate Mathematics Series) we found a similar model to the function used in R. The function will find the best Sum of Squares Error of a model that is transformed by the following functions below. The lambda is optimized according to the SSE.

$$y^{(\lambda)} = \frac{y^{\lambda}-1}{\lambda}, if \lambda \neq 0$$

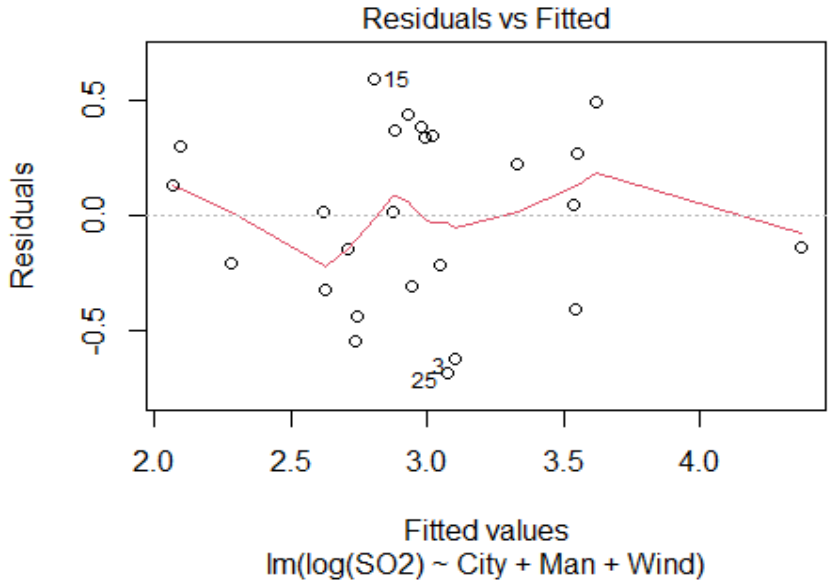
$$y^{(\lambda)} = ln(y), if \lambda = 0$$

```
library(MASS)

boxcox(linear.model1, lambda = seq(-3, 3, by = 0.1))
```



Our best transformation is lambda = 0.

```
linear.model.log <- lm(log(SO2) ~ City + Man + Wind, data = train.data)
plot(linear.model.log, which = 1)
```



```
summary(linear.model.log)

##
## Call:
## lm(formula = log(SO2) ~ City + Man + Wind, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67811 -0.30647  0.01819  0.34164  0.59462
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.6147279  0.7286361   6.333 5.74e-06 ***
## CityNE       0.5789315  0.2625269   2.205   0.0407 *
## CitySE      -0.4078873  0.2803332  -1.455   0.1629
## CitySW      -0.8637302  0.3263633  -2.647   0.0164 *
## CityW       -0.1315119  0.2876292  -0.457   0.6530
## Man          0.0005875  0.0002692   2.183   0.0426 *
## Wind        -0.1893043  0.0697848  -2.713   0.0143 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.433 on 18 degrees of freedom
## Multiple R-squared:  0.6422, Adjusted R-squared:  0.523
## F-statistic: 5.385 on 6 and 18 DF,  p-value: 0.002435
```

We see that the log transformation is more desirable than the square root transformation in the residuals vs. fitted values and the p-value created when testing the models significance. But we now have an adjusted R^2 that suggests our model is just as valuable as flipping a coin.

We will still test the predictive accuracy of this model.

```
trainMSE3 <- mean((train.data$SO2 - exp(predict(linear.model.log)))^2)
trainMSE3

## [1] 76.67647

testMSE3 <- mean((test.data$SO2 - exp(predict(linear.model.log, newdata
= test.data)))^2)
testMSE3

## [1] 493.0604
```

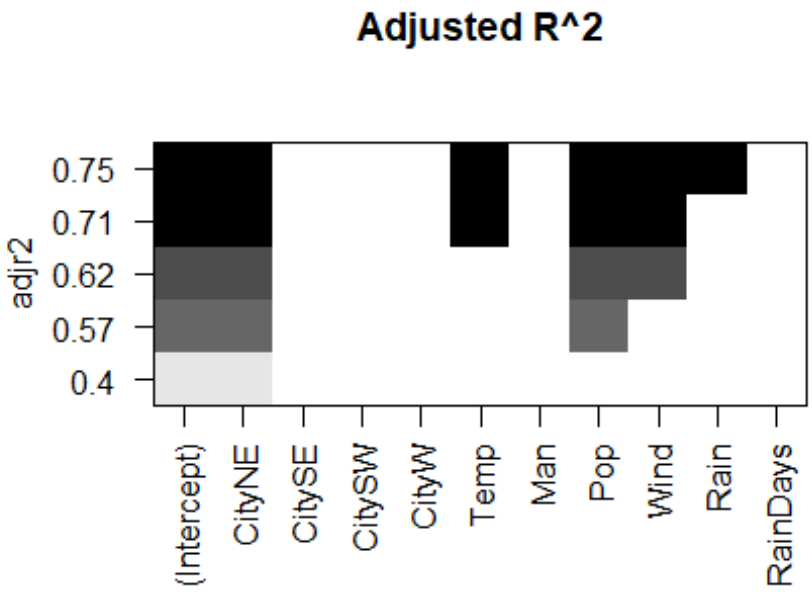
The training and test MSEs are like the other models we have created. We have sacrificed the adjusted R-squared for nothing.

We look to another algorithm for finding the best predictors.

Forward and Backward Selection

Let's go back to the linear regression models, with no transformation to the response, and consider forward and backwards variable selection to see if we stepped over something that was worth looking at.

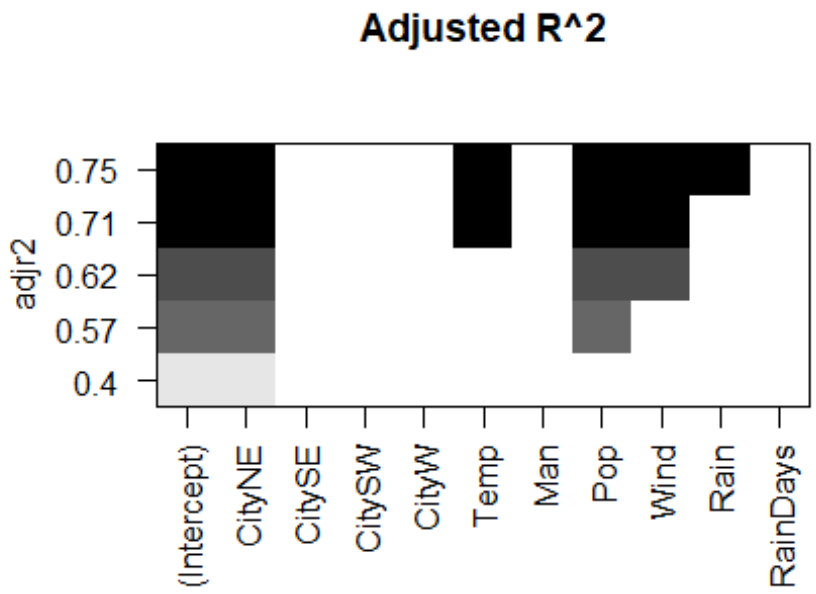
```
library(leaps)

back.models <- regsubsets(SO2 ~., data = train.data, nvmax = 5, method
= "backward")
summary(back.models)

## Subset selection object
## Call: regsubsets.formula(SO2 ~ ., data = train.data, nvmax = 5, meth
od = "backward")
## 10 Variables  (and intercept)
##          Forced in Forced out
## CityNE       FALSE      FALSE
## CitySE       FALSE      FALSE
## CitySW       FALSE      FALSE
## CityW        FALSE      FALSE
## Temp         FALSE      FALSE
## Man          FALSE      FALSE
## Pop          FALSE      FALSE
## Wind         FALSE      FALSE
## Rain         FALSE      FALSE
## RainDays     FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: backward
##          CityNE CitySE CitySW CityW Temp Man Pop Wind Rain RainDays
## 1  ( 1 ) "*"    " "    " "    " "   " "  " " " " " " " "  " "
## 2  ( 1 ) "*"    " "    " "    " "   " "  " " "*" " " " "  " "
## 3  ( 1 ) "*"    " "    " "    " "   " "  " " "*" "*"  " "  " "
## 4  ( 1 ) "*"    " "    " "    " "   "*"  " " "*" "*"  " "  " "
## 5  ( 1 ) "*"    " "    " "    " "   "*"  " " "*" "*"  "*"  " "

plot(back.models, scale = "adjr2", main = "Adjusted R^2")
```



The backwards algorithm shows us that our best model, according to adjusted R^2 has variables City, Temp, Pop, Wind, and Rain. Notice that this algorithm has suggested that Population be used and not Man. We will verify these results with a forward selection algorithm as well.

```
forward.models <- regsubsets(SO2 ~., data = train.data, nvmax = 5, meth
od = "forward")
summary(forward.models)

## Subset selection object
## Call: regsubsets.formula(SO2 ~ ., data = train.data, nvmax = 5, meth
od = "forward")
## 10 Variables  (and intercept)
##          Forced in Forced out
## CityNE       FALSE      FALSE
## CitySE       FALSE      FALSE
## CitySW       FALSE      FALSE
## CityW        FALSE      FALSE
## Temp         FALSE      FALSE
## Man          FALSE      FALSE
## Pop          FALSE      FALSE
## Wind         FALSE      FALSE
## Rain         FALSE      FALSE
## RainDays     FALSE      FALSE
## 1 subsets of each size up to 5
```

```
## Selection Algorithm: forward
##          CityNE CitySE CitySW CityW Temp Man Pop Wind Rain RainDays
## 1  ( 1 ) "*"    " "    " "    " "   " "  " " " " " "  " "  " "
## 2  ( 1 ) "*"    " "    " "    " "   " "  " " "*" " "  " "  " "
## 3  ( 1 ) "*"    " "    " "    " "   " "  " " "*" "*"  " "  " "
## 4  ( 1 ) "*"    " "    " "    " "   "*"  " " "*" "*"  " "  " "
## 5  ( 1 ) "*"    " "    " "    " "   "*"  " " "*" "*"  "*"  " "

plot(forward.models, scale = "adjr2", main = "Adjusted R^2")
```



Surprisingly, we receieve the same output from a forward selection of parameters. Lets use this model to see whether it has a quality of fit worth pursuing.

```
select.model <- lm(SO2 ~ City + Temp + Pop + Wind + Rain, data = train.
data)
summary(select.model)

##
## Call:
## lm(formula = SO2 ~ City + Temp + Pop + Wind + Rain, data = train.dat
a)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.2522  -3.1445  -0.1161   3.9324  16.2635
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 92.719645  25.199886   3.679  0.00203 **
## CityNE      20.057423   5.513744   3.638  0.00222 **
## CitySE      -2.556305   7.676734  -0.333  0.74346
## CitySW       1.181098  10.381144   0.114  0.91083
## CityW        0.439233   6.135035   0.072  0.94381
## Temp        -0.942012   0.423030  -2.227  0.04067 *
## Pop          0.017323   0.004641   3.733  0.00181 **
## Wind        -4.522094   1.491436  -3.032  0.00793 **
## Rain         0.357175   0.274198   1.303  0.21114
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.795 on 16 degrees of freedom
## Multiple R-squared:  0.8023, Adjusted R-squared:  0.7035
## F-statistic: 8.117 on 8 and 16 DF,  p-value: 0.0002159
```

Now we will take a look at the predictive capabilities of this model.

```
trainMSE4 <- mean(select.model$residuals^2)
trainMSE4

## [1] 49.50471

testMSE4 <- mean((test.data$SO2 - predict(select.model, newdata = test.
data))^2)
testMSE4

## [1] 583.0676
```

If anything, we have found a greater overfitted model and not a model that can predict new data.

We deduce the given model to a new nested model of the selection we have above. We find that the model is satisfactory without Rain and Temp and arrive at a model that has City, Pop, and Wind. We try this model too.

```
select.model2 <- lm(SO2 ~ City + Pop + Wind, data = train.data)
summary(select.model2)

##
## Call:
## lm(formula = SO2 ~ City + Pop + Wind, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.0413  -3.7569  -0.2314   6.1834  16.5377
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  48.973520  16.071706   3.047  0.00693 **
## CityNE       22.417236   5.772132   3.884  0.00109 **
## CitySE       -6.731159   6.084180  -1.106  0.28315
## CitySW      -15.725799   7.094242  -2.217  0.03976 *
## CityW        -3.323083   6.302957  -0.527  0.60447
## Pop           0.017164   0.004969   3.454  0.00283 **
## Wind         -3.814218   1.530470  -2.492  0.02267 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.498 on 18 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.6542
## F-statistic: 8.566 on 6 and 18 DF,  p-value: 0.0001719
```

```
trainMSE5 <- mean(select.model2$residuals^2)
trainMSE5
```

```
## [1] 64.95552
```

```
testMSE5 <- mean((test.data$SO2 - predict(select.model2, newdata = test
.data))^2)
testMSE5
```

```
## [1] 614.883
```

This model equally overfits the best adjusted R^2 model and so we can conclude that this algorithm does not help.

Our next model attempt is simply done out of curiosity and trial and error.

Polynomial Regression

Using the first nested model we created, with variables City, Man, and Wind, we find that a quadratic coefficient is significant to the process.

```
poly.model <- lm(SO2 ~ City + Man + poly(Wind, 2), data = train.data)
summary(poly.model)
```

```
##
## Call:
## lm(formula = SO2 ~ City + Man + poly(Wind, 2), data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.4043  -6.3178   0.0521   3.3265  15.0177
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     18.689858   4.920220   3.799  0.00144 **
```

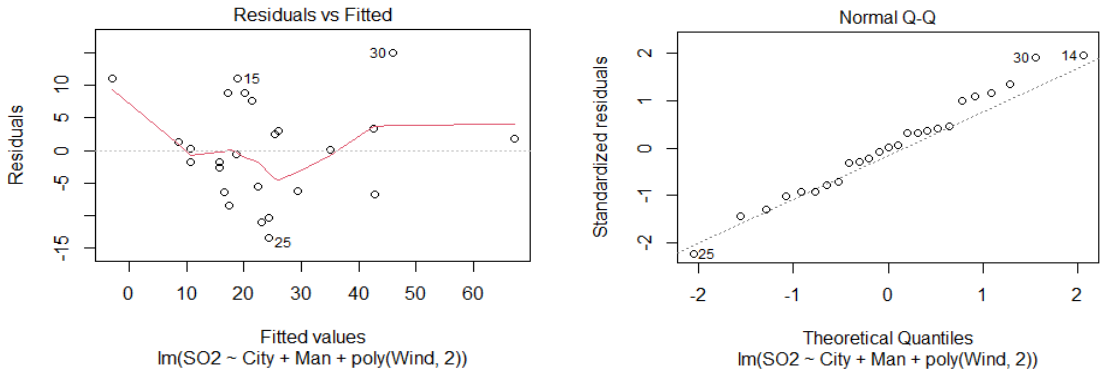
```
## CityNE          19.613729   5.492881   3.571  0.00235 **
## CitySE          -8.812762   5.938675  -1.484  0.15612
## CitySW         -12.239184   6.834130  -1.791  0.09113 .
## CityW           -6.799400   6.219065  -1.093  0.28951
## Man              0.016271   0.005741   2.834  0.01145 *
## poly(Wind, 2)1 -27.541528  10.386795  -2.652  0.01679 *
## poly(Wind, 2)2 -21.807034  10.054493  -2.169  0.04456 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.987 on 17 degrees of freedom
## Multiple R-squared:  0.7807, Adjusted R-squared:  0.6904
## F-statistic: 8.645 on 7 and 17 DF,  p-value: 0.0001464
```

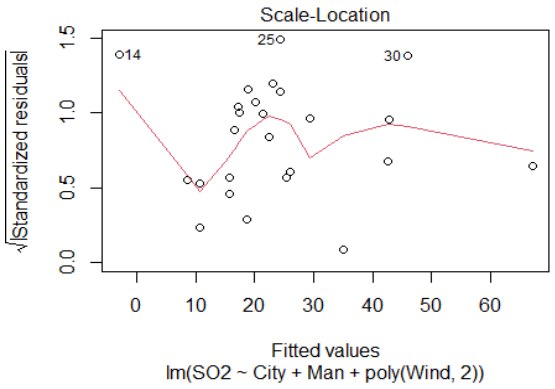
This model works surprisingly well. Once again we have shown that wind is certainly an essential part of this process and that it can contribute on a quadratic

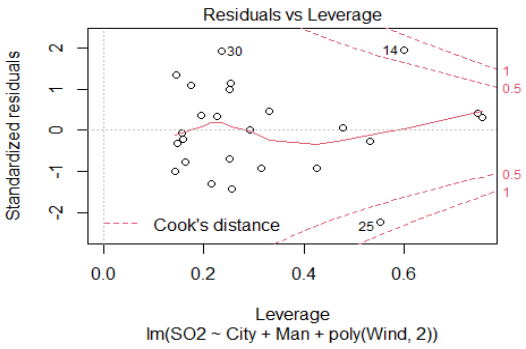
```
trainMSE6 <- mean(poly.model$residuals^2)
trainMSE6

## [1] 54.92266

testMSE6 <- mean((test.data$SO2 - predict(poly.model, newdata = test.da
ta))^2)
testMSE6

## [1] 473.5411
```

We find that these training and test MSEs rival the early nested models'. This is still not a good fit but we should note that the addition of the quadratic Wind term does not affect the performance of the nested model and has only greatly enhanced the adjusted R-squared and multiple R-squared. It would be foolish of us not to look at the residuals of this model, because if they are not satisfactory then we cannot consider this model.

```
plot(poly.model)
```







Unfortunately, the assumptions seem to worse than the original linear model we proposed. The normality QQ-plot is now off-center, there are many studentized residuals that are of large value and there are not points whose Cook's distance are approaching a point that suggest they may be influential enough to sway the entire model. We should not use this model.

Nonparametric Learning, K-Nearest Neighbours

We should create a nonparametric approach to the data we have. We are doing this simply to see if our model is close to the unsupervised approach. The main objective to this dataset is to interpret the process at hand. That is, what affects the concentration of SO_2. We will end up using a parametric model of sorts, but we should consider what level of prediction accuracy we are sacrificing for a model that can be properly interpreted. For instance, if this model can greatly outperform the parametric model, then we may be at a disadvantage by trying to interpret the process parametrically.

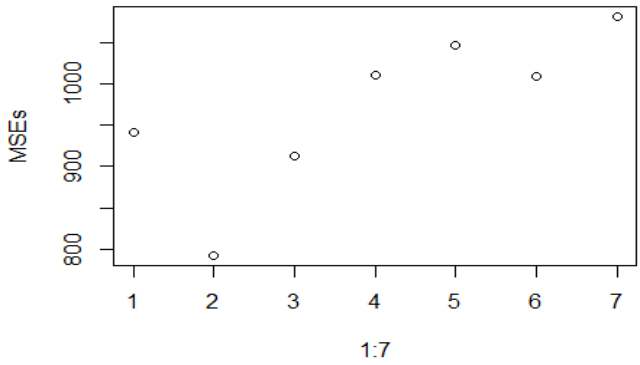
To begin with, we will show the formula for KNN Regression.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i$$

This equation says that for a given value K and prediction point x_0 the predicted response will be the average of the K closest values to y_0. This distance is Euclidean distance (the distance of all the predictors closest), so if there are too many predictors being used, this type of regression can ruin itself through the curse of dimensionality. So, we will only use this with the predictors that we have found to be significant thus far.

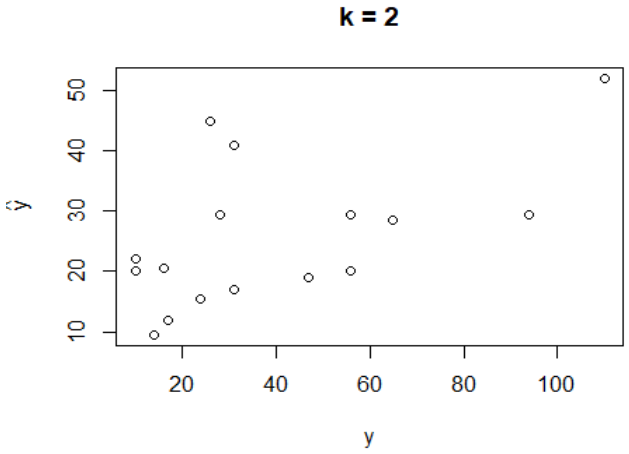
We try the algorithm for multiple values of K.

```
library(FNN)

train.x <- cbind(train.data[,c(3, 5, 6)])
test.x <- cbind(test.data[,c(3, 5, 6)])
train.SO2 <- train.data$SO2

nonp.model1 <- knn.reg(train.x, test.x, train.SO2, k = 1)
nonp.model2 <- knn.reg(train.x, test.x, train.SO2, k = 2)
nonp.model3 <- knn.reg(train.x, test.x, train.SO2, k = 3)
nonp.model4 <- knn.reg(train.x, test.x, train.SO2, k = 4)
nonp.model5 <- knn.reg(train.x, test.x, train.SO2, k = 5)
nonp.model6 <- knn.reg(train.x, test.x, train.SO2, k = 6)
nonp.model7 <- knn.reg(train.x, test.x, train.SO2, k = 7)

MSEs <- c(mean((test.data$SO2 - nonp.model1$pred)^2), mean((test.data$S
O2 - nonp.model2$pred)^2), mean((test.data$SO2 - nonp.model3$pred)^2),
mean((test.data$SO2 - nonp.model4$pred)^2), mean((test.data$SO2 - nonp.
model5$pred)^2), mean((test.data$SO2 - nonp.model6$pred)^2), mean((test
.data$SO2 - nonp.model7$pred)^2))

plot(1:7, MSEs)
```



This suggests our best approach would be to use K = 2.

```
plot(test.data$SO2, nonp.model2$pred, xlab = "y", ylab = expression(hat
(y)), main = "k = 2")
```



This plot should be roughly linear, the closer to linear this is, the better the KNN regression worked. This was not the case.

```
testMSE7 <- MSEs[2]
testMSE7

## [1] 792.4844
```

KNN does not give a better test MSE than our parametric model. This is encouraging news to continue interpretation of the model through parametric approach.

Combination of Best Fits

Out of curiosity, we take a combination of our best attributes we've found and create a model that could not be found otherwise.

```
linear.modelr <- lm(log(SO2) ~ City + Man + poly(Wind, 2), data = train
.data)
summary(linear.modelr)

##
## Call:
## lm(formula = log(SO2) ~ City + Man + poly(Wind, 2), data = train.dat
a)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5878 -0.2085 -0.0088  0.1847  0.5966
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.9928056  0.2109508  14.187 7.47e-11 ***
## CityNE          0.5079691  0.2355032   2.157   0.0456 *
## CitySE         -0.5297235  0.2546163  -2.080   0.0529 .
## CitySW         -0.7710635  0.2930083  -2.632   0.0175 *
## CityW          -0.3102286  0.2666378  -1.163   0.2607
## Man             0.0004520  0.0002462   1.836   0.0838 .
## poly(Wind, 2)1 -1.3623054  0.4453262  -3.059   0.0071 **
## poly(Wind, 2)2 -1.0317314  0.4310790  -2.393   0.0285 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 17 degrees of freedom
## Multiple R-squared:  0.7324, Adjusted R-squared:  0.6222
## F-statistic: 6.646 on 7 and 17 DF,  p-value: 0.0006927

trainMSEr <- mean((train.data$SO2 - exp(predict(linear.modelr)))^2)
trainMSEr

## [1] 51.74831

testMSEr <- mean((test.data$SO2 - exp(predict(linear.modelr, newdata =
test.data)))^2)
testMSEr

## [1] 521.814
```

This model also a gross overfitting of the training data given. It has become obvious to us that linear regression will only provide overfitted models because of our lack of training data.

After many different models, we look at a summary of what we have found.

Model	trainMSEs	testMSEs	Second.Train.MSE	Second.Test.MSE
All Variables	57.17618	437.9691	118.1003	475.01631
City, Man, Wind	70.12028	456.5554	166.3250	283.02852
log(y) ~ City, Man, Wind	76.67647	493.0604	279.1319	148.64147
Exhaustive: City, Temp, Pop, Wind	49.50471	583.0676	164.6037	469.32442
Exhaustive: City, Pop, Wind	64.95552	614.8830	199.9921	432.73456
Poly: City, Man, Wind^2	54.92266	473.5411	165.4119	270.60714
KNN, k = 2	NA	792.4844	NA	799.92188
log(y) ~ City, Man, Wind^2t	51.74831	521.8140	258.5611	87.69536

Our findings are unfortunately grim. Using multiple linear regression may not have been our best option for accurate prediction, but it did allow us to find which variables were consistently important to the response.

Spatial Analysis and Kriging Prediction

Before we begin this analysis, we should explain what it is we are assuming and then the type of analysis we will do. Spatial analysis is created from the Theory of Random Functions. This theory allows us to create an abstract and non-deterministic model of the stochastic process we are observing. For us, we are observing the concentrations of SO_2 in various geodesic coordinates in the United States. The possible values of this concentration are explained by the random variable Z(s) where s is the spatial coordinates of the measurement. These spatial coordinates are the longitude and latitude that we showed in the Data Processing section.

```
library(sp)

library(rgdal)

library(raster)

getClass("Spatial")
```

```
## Class "Spatial" [package "sp"]
##
## Slots:
##
## Name:         bbox proj4string
## Class:      matrix         CRS
##
## Known Subclasses:
## Class "SpatialPoints", directly
## Class "SpatialMultiPoints", directly
## Class "SpatialGrid", directly
## Class "SpatialLines", directly
## Class "SpatialPolygons", directly
## Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
## Class "SpatialPixels", by class "SpatialPoints", distance 2
## Class "SpatialMultiPointsDataFrame", by class "SpatialMultiPoints",
distance 2
## Class "SpatialGridDataFrame", by class "SpatialGrid", distance 2
## Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
## Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
## Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distan
ce 2
```

These are specific packages used for spatial data and a list of new classes of objects that are used to define spatial data.

We create a spatial points dataframe using the code below.

```
coordmat <- cbind(Longitude, Latitude)
pts <- SpatialPoints(coordmat)
```

```
llCRS <- CRS("+proj=longlat +datum=WGS84")
pts <- SpatialPoints(coordmat, proj4string = llCRS)
df <- data.frame(id = 1:41,  SO2 = SO2)
spdf <- SpatialPointsDataFrame(pts, data = df)
```

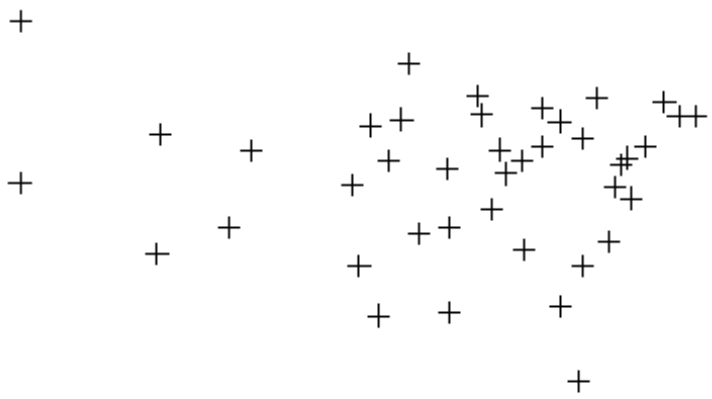
This dataset uses only the coordinates and the SO_2 concentrations measured at those points.

We have changed the dataframe spdf into a spatial data dataframe so that we can perform special procedures. Inside the CRS() function is the setting to treat the data given as Geographic coordinate system.

https://rspatial.org/raster/spatial/index.html

Now we can take a quick look at what it is we have created.

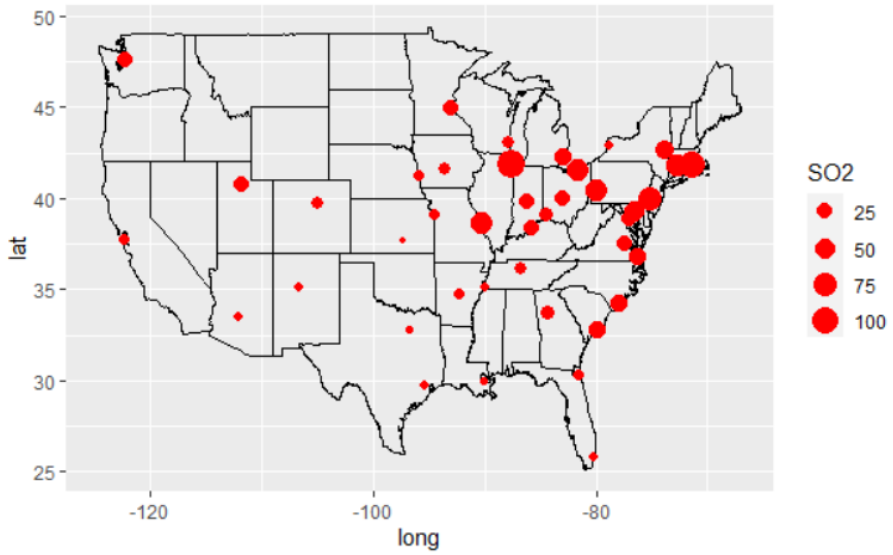
```
plot(spdf)
```



This shows us that we have created spatial data that rough conincides with the shape of the U.S.

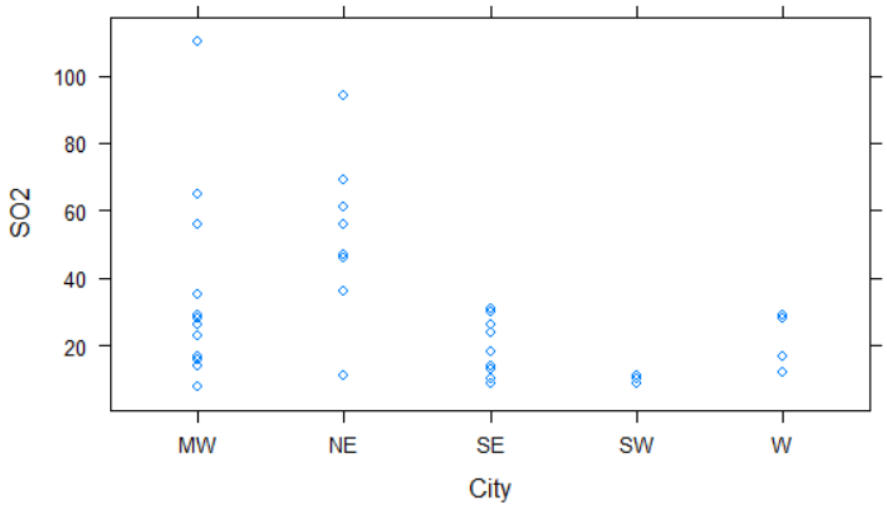
We decide to map these points over an image of the U.S. and create points that reflect the magnitude of the observation made at those points.

```
library(gstat)

library(MAP)

library(ggplot2)
usa <- map_data("state")
ggplot() + geom_path(data = usa, aes(x = long, y = lat, group = group))
+ geom_point(data = air.data, aes(x = Longitude, y = Latitude, size = S
O2), color = "red")
```



We see that we have successfully transformed the data and we now see a trend in the value of SO_2 concentration in the US.

```
xyplot(SO2 ~ City, air.data)
```



This relationship corresponds directly with our plot. The MidWest is not in the west but just west of the North East. The two beside each other are composed of the largest concentrations of SO_2.

Now that we have successfully transformed our data in spatial objects, we can begin to analyze them.

Variograms

We are next going to look for a type of correlation in spatial data. In standard statistics we could use a scatterplot to determine what variables are correlated. But in spatial data we must use what is called a variogram. But before we do this, we should outline some underlying theory that describes what we are seeing. First, we assume the process (SO_2 concentrations) is modeled by a random function Z(s) that has two components. This model implies that the process is stationary. We consider a process to be stationary if its variance is constant independent of spatial location.

$$Z(s) = m + e(s)$$

$$E[Z(s)] = m, \;\; Residual = e(s)$$

A process is intrinsically stationary if the equation below is true.

$$E[Z(s) - Z(s+h)] = 0$$

Where h is our lag, which is any spatial difference that is within the set of possible values that Z(s) can take on. This means that the semivariance of the process is reduced to the γ below. This is our variogram equation.

$$Var[Z(s) - Z(s+h)] = E[(Z(s) - Z(s+h))^2] = 2\gamma(h)$$

$$\Longrightarrow \gamma(h) = \frac{1}{2}E(Z(s) - Z(s+h))^2$$

Before we plot the variogram, we first consider transforming our SO_2 variable.

We decide to log the SO_2 variable because of large skewness.

```
library(psych)

describe(SO2)

##    vars  n  mean    sd median trimmed   mad min max range skew kurto
sis   se
## X1    1 41 30.05 23.47     26      26 17.79   8 110   102 1.58     2
.26 3.67
```

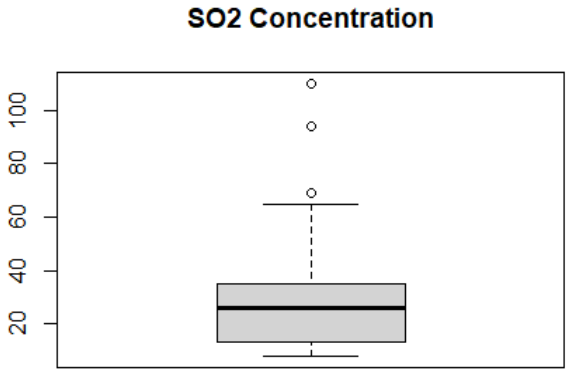
```
describe(log(SO2))
```

```
##    vars  n mean  sd median trimmed  mad  min max range skew kurtosis
se
## X1    1 41 3.15 0.7   3.26    3.12 0.92 2.08 4.7  2.62 0.31    -0.92
0.11
```

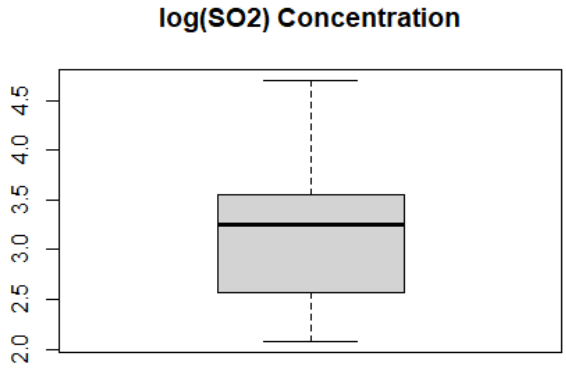
A skewness greater than 1 or less than -1 will greatly affect the variogram in a manner that makes its' interpretations wrong.

Variograms are also highly susceptible to outliers, so we also check the outliers because no matter how small the sample is, it would be a mistake to model a variogram with outliers.

```
boxplot(SO2, main = "SO2 Concentration")
```



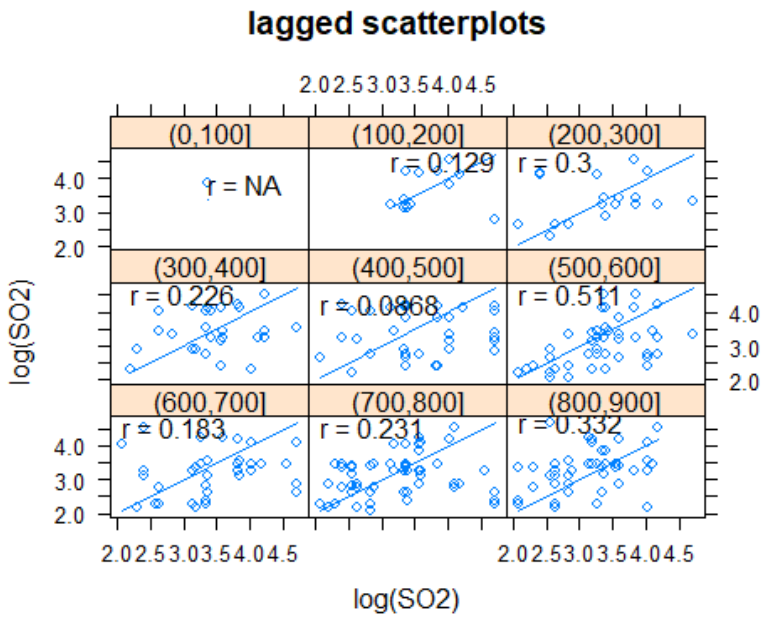

```
boxplot(log(SO2), main = "log(SO2) Concentration")
```



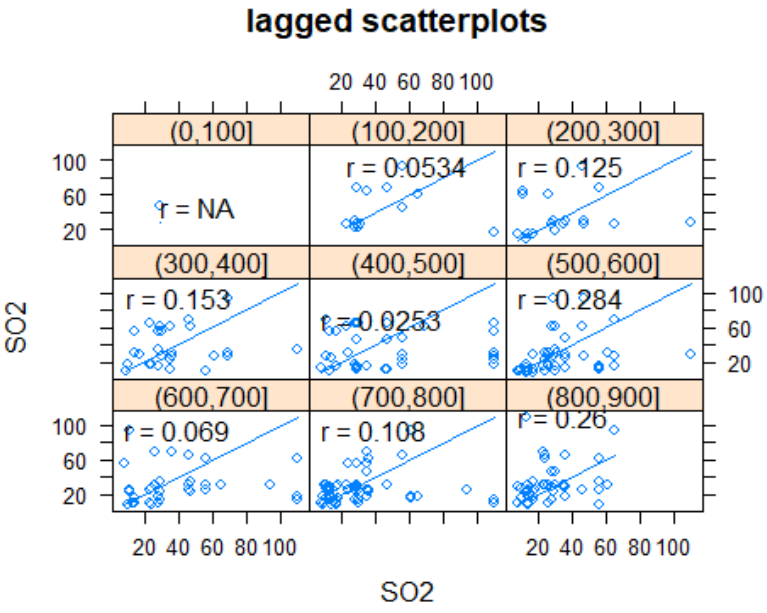
We see that after a log transformation, our data no longer has any outliers, this is also an encouraging reason for us to continue using the log transformation.

Below we look at scatter plots of spatial data that are grouped by lags, given at the top of each graph. We see that if we do not log() our response then our data will not be evenly spread out.

```
hscat(log(SO2) ~ 1, spdf, (0:9)*100)
```

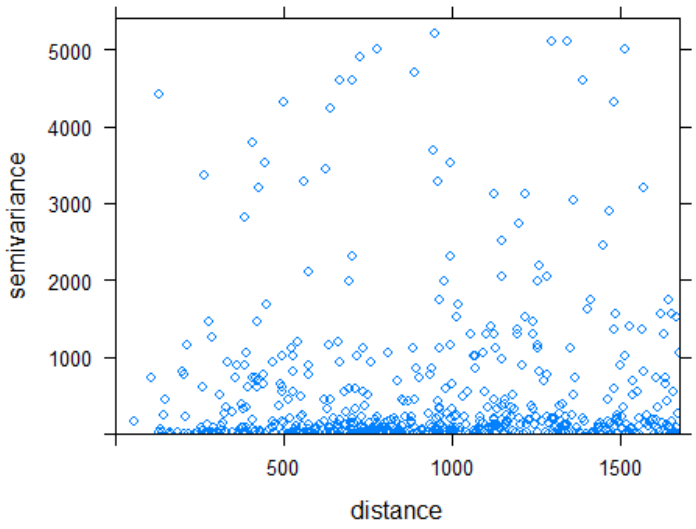



```
hscat(SO2 ~ 1, spdf, (0:9)*100)
```

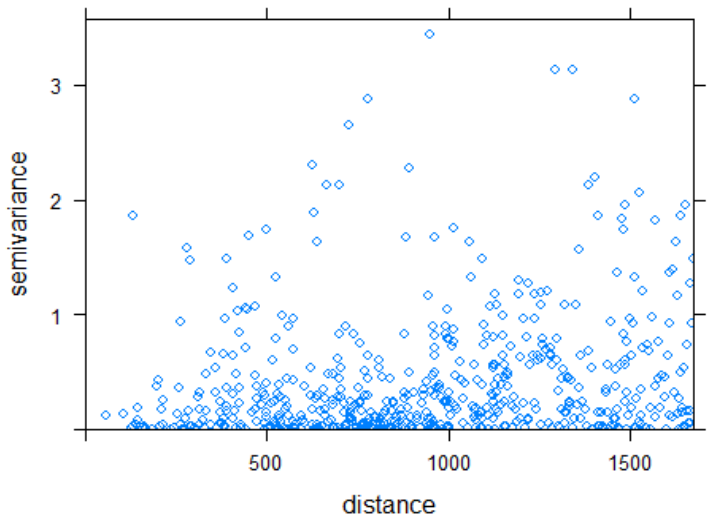


The variogram beneath plots all possible $[Z(s_i) - Z(s_j)]^2$ values against the distance, h, given from the data given.

```
library(kableExtra)

plot(variogram(SO2 ~ 1, spdf, cloud = TRUE))
```




```
plot(variogram(log(SO2) ~ 1, spdf, cloud = TRUE))
```



These values are created by averaging a large amount of $[Z(s_i) - Z(s_j)]^2$ values of varying sizes. Then calculating the average distance of this sum of differences to be

plotted in the cloud above. We will take a look at some of these values in the table below.

```
v <- variogram(log(SO2) ~ 1, spdf, cloud = TRUE)
df <- data.frame(np = v[1:10,1], AvgDist = v[1:10,2], Est = v[1:10, 3])
kable(df, "simple")
```

np	AvgDist	Est
2	1053.1257	0.0166206
3	942.4259	0.1407832
65539	1254.3448	0.0359828
131075	1529.0405	0.0606588
65541	1317.8009	0.5187420
262149	955.2874	0.0976081
65542	1437.1045	0.3218799
262150	485.6810	0.2165188
327686	525.4921	0.0233762
65543	1111.5439	0.0027460

The table produced shows the number of comparisons $(Z(s) - Z(s+h))^2$, np, that were made to estimate $\hat{\gamma}(h)$, Est, where the average h was AvgDist. We see that there is an incredible amount of comparisons made for every distance, except the first two semivariance estimates. This occurs because the data is a) small and b) is irregularly spaced.

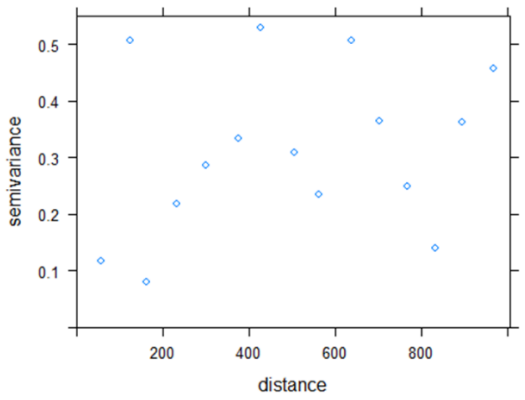
The drastic difference in log transformed data leads us to use the log(SO_2). There is no possible way we could have used semivariance estimates of 1000 to 5000.

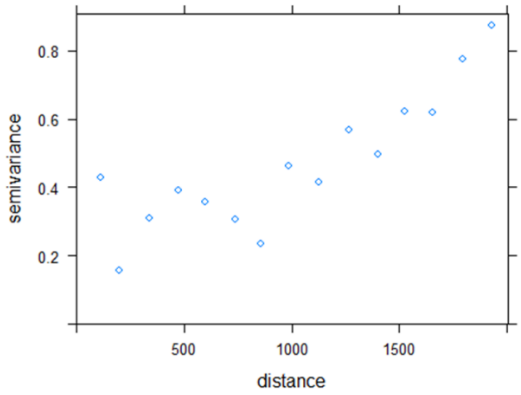
Variogram Fit

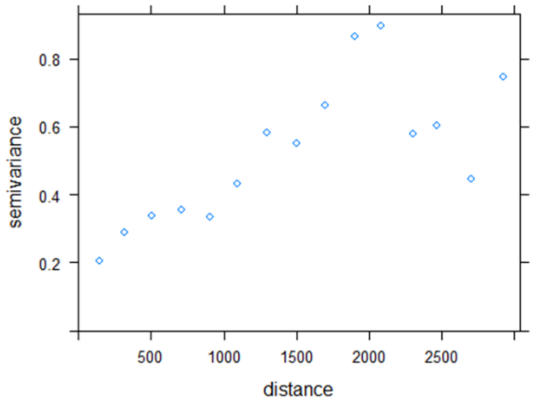
What we are going to try to do now is model the semivariances. ie: give $\gamma(h)$ a parametric equation. The key to variogram modeling is to capture the spatial correlation that initially takes place at a certain lag bewteen observations. What we are implying in that sentence is that all observations are correlated to the ones that surround it up to a certain dsitance, anywhere past that distance the observations become independent. We should be able to find a parametric model that corresponds to that initial spatial correlation.

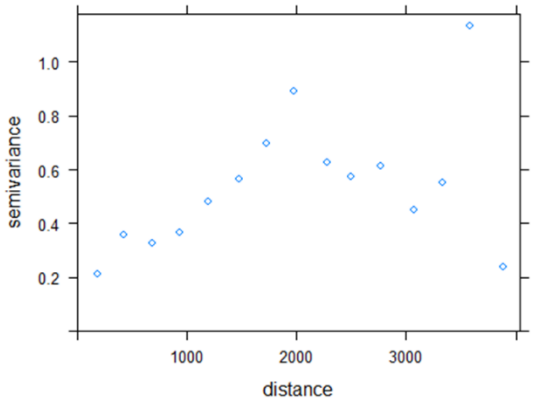
To begin with, we need to consider what cutoff would be most beneficial to the variogram. A look at the Average Distances considered in the previously shown table means that generally anywhere above 5000 would likely be distances too large to look at.

```r
exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 1000)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 2000)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 3000)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 4000)
plot(exp.var1)
```





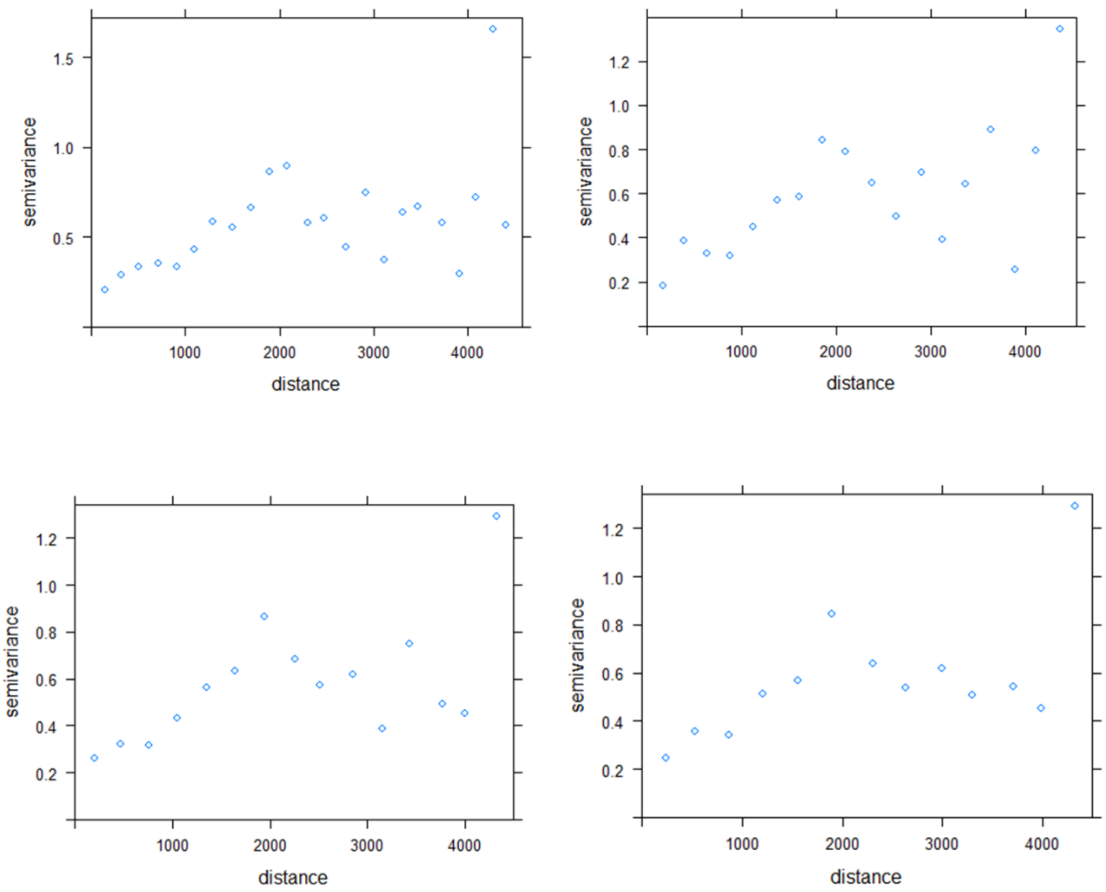




We see that cutoffs of 2000 and 3000 give reasonable patterns, but we need a cutoff that goes well past 2000 or 3000.Prior knowledge of what typical variograms look like lead us to believe that a cutoff of 2000 leaves out an essential spatially dependent pattern that we need to model. Below are bins for a cutoff of 4500.

These bins are the interval for which semivariance estimations are made. So all the different lags that went into one of the estimations were all in a certain lag interval, this is what we are looking at.

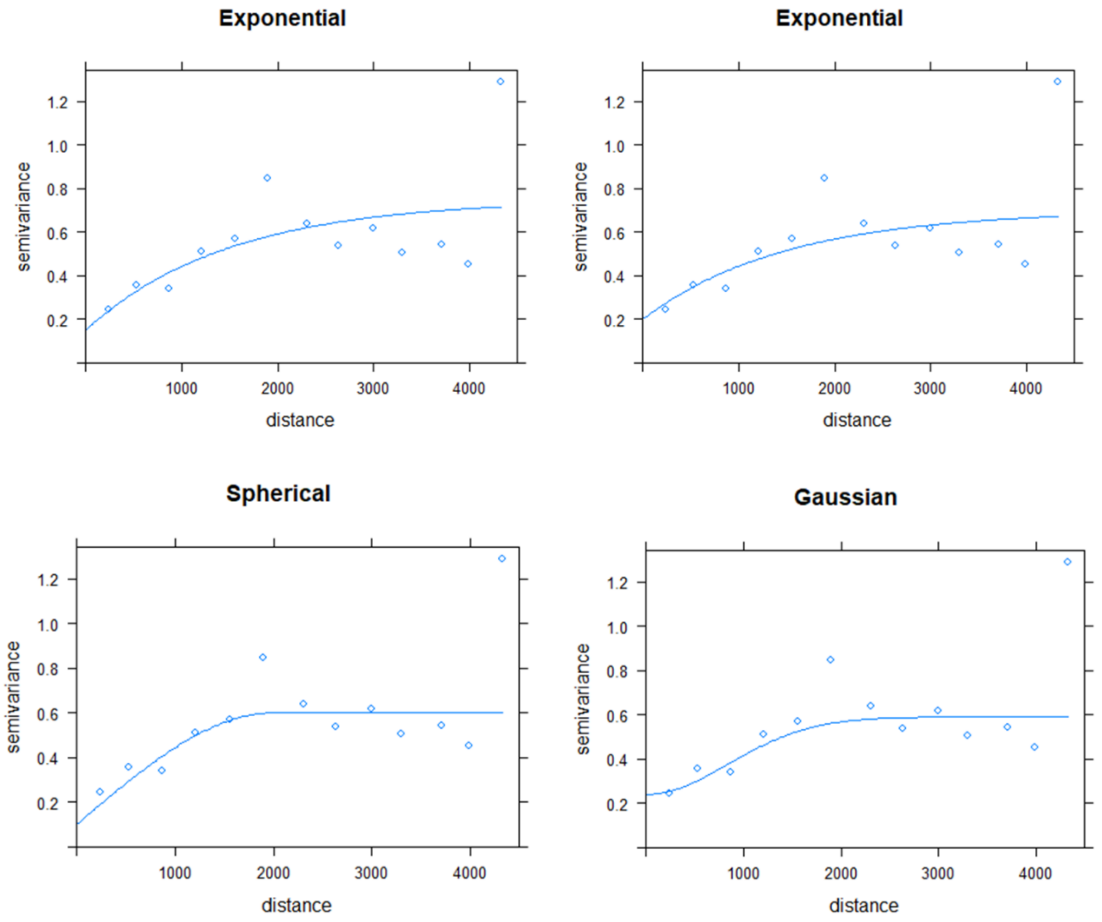
```
exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 4500, width = 200)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 4500, width = 250)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 4500, width = 300)
plot(exp.var1)

exp.var1 <- variogram(log(SO2) ~ 1, spdf, cutoff = 4500, width = 350)
plot(exp.var1)
```



We see that a bin width of 350 is sufficient for creating an experimental variogram that we can use to model.

```
v.experi <- variogram(log(SO2) ~ 1, spdf, cutoff = 4500, width = 350)
```

```
v.fita <- vgm(0.6, "Exp", 1500, 0.15)
v.fitb <- vgm(0.5, "Exp", 1500, 0.2)
v.fitc <- vgm(0.5, "Sph", 2000, 0.1)
v.fitd <- vgm(0.35, "Gau", 1200, 0.24)

plot(v.experi, v.fita, main = "Exponential")

plot(v.experi, v.fitb, main = "Exponential")

plot(v.experi, v.fitc, main = "Spherical")

plot(v.experi, v.fitd, main = "Gaussian")
```



All four of these models seem to fit the variogram very well. We will have to decide this by the Sum of Squares Error that is produced by fitting these models.

```
fit1 <- fit.variogram(v.experi, v.fita)
fit2 <- fit.variogram(v.experi, v.fitb)
fit3 <- fit.variogram(v.experi, v.fitc)
fit4 <- fit.variogram(v.experi, v.fitd)

attr(fit1, "SSErr")

## [1] 3.307024e-06

attr(fit2, "SSErr")

## [1] 3.307024e-06

attr(fit3, "SSErr")

## [1] 2.974091e-06

attr(fit4, "SSErr")

## [1] 3.150223e-06
```

The best fit is the Spherical model and close behind is the Gaussian fit. We will using these models to predict our SO_2 concentrations.

Below we have tables that show the parameters estimated. Because we have not added any extra parameters by using the Spherical models, we can trust the best SSE method. Otherwise, we would have used AIC to account for the number of parameters per model.

```
kable(fit3, "simple")
```

model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.1835652	0.000	0.0	0	0	0	1	1
Sph	0.4917201	2684.134	0.5	0	0	0	1	1

```
kable(fit4, "simple")
```

model	psill	range	kappa	ang1	ang2	ang3	anis1	anis2
Nug	0.2412378	0.000	0.0	0	0	0	1	1
Gau	0.4356466	1257.821	0.5	0	0	0	1	1

Here are the tables of the parameters to show that Spherical and Gaussian models have the same number of parameters. Now we fit these to be used for prediction.

```
fitted.fit1 <- fit.variogram(v.experi, vgm(psill = 0.4917201, model = "
Sph", range = 2684.134, nugget =     0.1835652, kappa = 0.5))
fitted.fit2 <- fit.variogram(v.experi, vgm(psill = 0.4356466, model = "
Gau", range = 1257.821, nugget = 0.2412378, kappa = 0.5))
```

Kriging

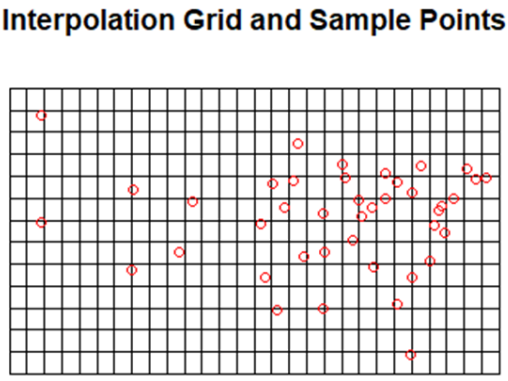
An accurate and short definition of kriging is given in the quote below.

“Kriging predicts values at unvisited sites from sparse sample data based on a stochastic model of continuous spatial variation. It does so by taking into account knowledge of the spatial variation as represented in the variogram or covariance function.”*

Basic Steps in Geostatistics: The Variogram and Kriging, Springer*

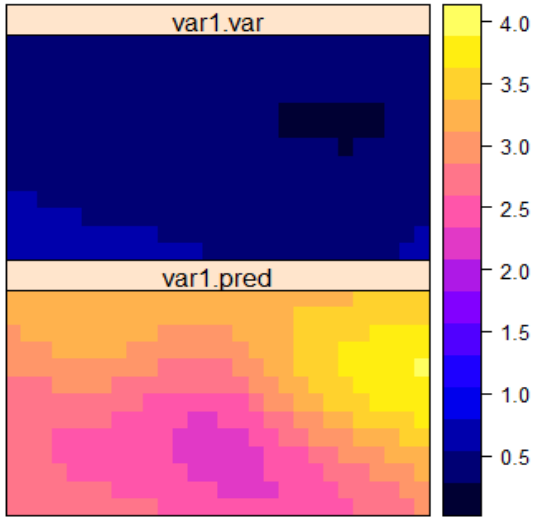
This is the reason for why we have gone into detail on the variogram. But now we need to create a grid for the Spatial dataframe that we have.

```
library(geoR)

Cols <- seq(from = -125, to = -70, by = 2)
Rows <- seq(from = 25, to = 50, by = 2)
air.grid <- expand.grid(x = Cols, y = Rows)
air.grid <- SpatialPoints(air.grid, proj4string = llCRS)
gridded(air.grid) <- TRUE
plot(air.grid)
points(spdf, col = "red")
title("Interpolation Grid and Sample Points")
```



Now that we have a prediction spatial dataset, we can begin the kriging process to create predictions that can be verified through cross-validation.

```
ordinary.krig <- krige(log(SO2) ~ 1, spdf, air.grid, model = fitted.fit
2)

## Warning in proj4string(d$data): CRS object has comment, which is los
t in output

## Warning in proj4string(newdata): CRS object has comment, which is lo
st in output

## [using ordinary kriging]

## Warning in proj4string(newdata): CRS object has comment, which is lo
st in output

spplot(ordinary.krig)
```



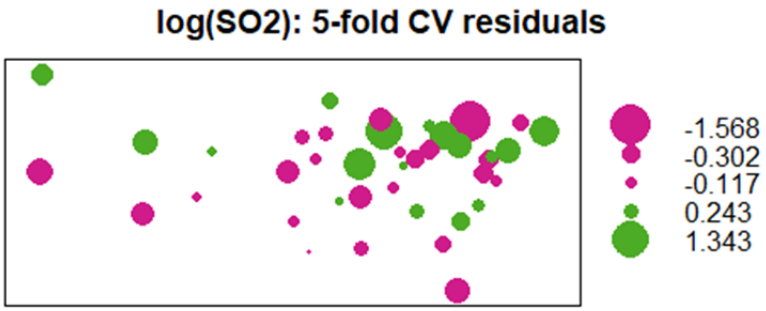
First off, I have left in the Warnings because I could not find the reason as to why they occur, or what they even mean. We get meaningful results so we decided to ignore this Warning. I did manage to trace back the only difference in my Rcode from other examples. The only difference lies in the CRS() code that I use to define the spatial objects. I use the same code as others but I receive a “no_defs” portion even though I should get an “ellps=” portion and a “towgs84=” portion. But even after writing these parts in, they are excluded for some unknown reason.

These Warning messages occur for all krige() and cv.krige() functions, but we will only show it for one. We will continue with our analysis now.

The second map shown above is the spatial grid filled in with estimates of log(SO_2), then there is the variance of those predictions in the first map. We see a pattern that shows larger values in the North Eastern region and even all along the Eastern region. The South West has the lowest and North West is larger than the SW. This is in accordance

with our findings. We also see that the variance of the predictions is not drastic, meaning that we have reasonable results.

```
cv.krig <- krige.cv(log(SO2) ~ 1, spdf, fitted.fit1, nmax = 40, nfold=5
)
```

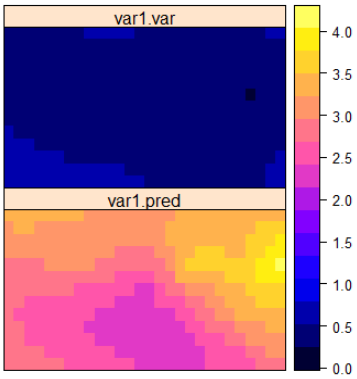



```
mean(cv.krig$residual^2)
```

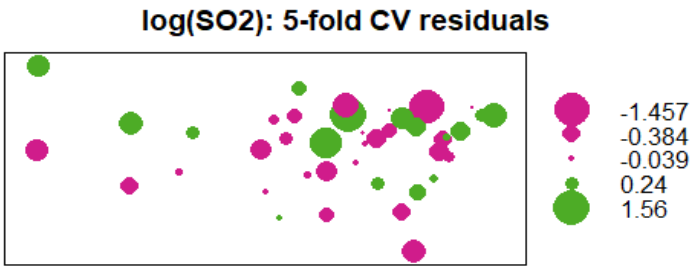
```
## [1] 0.2768118
```

This cross-validation value is phenomenal. No where close to what we got using the Linear Regression. Lets try the second best variogram model.

```
ord.krig <- krige(log(SO2) ~ 1, spdf, air.grid, model = fitted.fit1)

spplot(ord.krig)
```



```
cv.krig1 <- krige.cv(log(SO2) ~ 1, spdf,fitted.fit2, nmax = 40, nfold=5
)

bubble(cv.krig1, "residual", main = "log(SO2): 5-fold CV residuals")
```




```
mean(cv.krig1$residual^2)

## [1] 0.2889524
```

We find similar results in the second model, which means that both models are very good fits. Either way, spatial analysis of this data is clearly the most effective way to analyze the data.

Leave One Out Cross Validation (LOOCV)

Due to the nature of our data and the small sample size. It is clear that train/test split is not a reasonable approach. This is because our data is so small that randomly splitting the data into a test/train set can lead to dramatically different results.

Here we look at a cross validation approach to get passed our problem with train and test splits. We use LOOCV as our data is very small, making this possible. LOOCV is essentially K-fold CV where K equals the sample size.

This block of code will determine the LOOCV residuals for the models that include a transformation of the response.

```
resid.cv.log = resid.cv.mix = numeric(41) #creates empty vectors
for(i in 1:41){
  # Remove the ith obs
  train = air.data1[-i,]
  # Fit models using training_data
  cv_log = lm(log(SO2) ~ City + Man + Wind, data = train)
  cv_mix  = lm(log(SO2) ~ City + Man + poly(Wind, 2), data = train)
  # Prediction for the ith obs and obtain the residual
  resid.cv.log[i] = air.data1$SO2[i]- exp(predict(cv_log, newdata=air.d
ata1[i,]))
  resid.cv.mix[i] = air.data1$SO2[i] - exp(predict(cv_mix, newdata=air.
data1[i,]))
}
```

Here we determine the LOOCV scores for the various models.

```
library(boot)
loocv.fit = glm(SO2 ~ ., data = air.data1)
LOOCV = cv.glm(air.data1, loocv.fit, K = 41)
LOOCV$delta[1]

## [1] 257.8051

loocv.fit1 = glm(SO2 ~ City + Man + Wind, data = air.data1)
LOOCV1 = cv.glm(air.data1, loocv.fit1, K = 41)
LOOCV1$delta[1]

## [1] 252.7577

loocv.log = mean(resid.cv.log^2); loocv.log

## [1] 285.1106

loocv.fit2 = glm(SO2 ~ City + Temp + Pop + Wind, data = air.data1)
LOOCV2 = cv.glm(air.data1, loocv.fit2, K = 41)
LOOCV2$delta[1]

## [1] 353.7139
```

```
loocv.fit3 = glm(SO2 ~ City + Pop + Wind, data = air.data1)
LOOCV3 = cv.glm(air.data1, loocv.fit3, K = 41)
LOOCV3$delta[1]

## [1] 379.8318

loocv.fit4 = glm(SO2 ~ City + Man + poly(Wind,2), data = air.data1)
LOOCV4 = cv.glm(air.data1, loocv.fit4, K = 41)
LOOCV4$delta[1]

## [1] 270.4166

loocv.mix = mean(resid.cv.mix^2); loocv.mix

## [1] 251.9151
```

Based in LOOCV the last model and the model using only City, Man and Wind as predictors appear to be the best.

```
Model <- c("All Variables", "City, Man, Wind", "log(y) ~ City, Man, Win
d", "Exh Selection: City, Temp, Pop, Wind", "Exh Selection: City, Pop,
Wind", "Poly: City, Man, Wind^2", "log(y) ~ City, Man, Wind^2")
LOOCVs <- c(LOOCV$delta[1], LOOCV1$delta[1], loocv.log,  LOOCV2$delta[1
],  LOOCV3$delta[1],  LOOCV4$delta[1], loocv.mix)

summarize <- data.frame(Model, "LOOCV MSE" = LOOCVs)
kable(summarize, "simple")
```

Model	LOOCV.MSE
All Variables	257.8051
City, Man, Wind	252.7577
log(y) ~ City, Man, Wind	285.1106
Exh Selection: City, Temp, Pop, Wind	353.7139
Exh Selection: City, Pop, Wind	379.8318
Poly: City, Man, Wind^2	270.4166
log(y) ~ City, Man, Wind^2	251.9151

Statistical Findings and Conclusions

From our investigation we can conclude that the most important features with respect to the response are City, Man, Wind, and $Wind^2$. Various other predictors were significant at one point or another, but these three predictors were consistently the most important to the Concentration of Sulphur Dioxide.

It truly is hard to make a concrete statement on which models performed better than each other. In our model summaries we've shown the results from two pairs of training and test datasets and the results were worrisome as were drastically inconsistent.

As we have highlighted repeatedly, our sample size was our single largest problem and made analysis very difficult. This resulted in either large overfitting of the training data or a lack of data to test on.

What we learned in this investigation is that problematic data leads to problematic results, so we should be wary of what we 'conclude' because it was only based off 41 observations.

From our investigation we can conclude that the most important features are City, Man, Wind. The coefficient estimates for wind were always negative, indicating that an increase in wind speed will result in decrease SO_2 concentration. Upon further research the reason why wind is an important predictor is attributed to the fact that higher wind speeds can disperse the SO_2 particles more, effectively lowering SO_2 concentration. Man was an intuitively important factor, who would have thought that more large manufacturers would directly correlate with an increase in Sulphur Dioxide concentration. Lastly, the City factor was consistently important, this may be due to the level of urbanization in different areas of the USA and also likely speaks volumes about the practices of Sulphur Dioxide emission control per state.

In our spatial analysis we find that the concentration of SO_2 is a spatially dependent process. Our LOOCV results are extremely low with residuals centered around zero. The ordinary kriging processes used means that we only used spatial coordinates and did not include any variables. We did this to determine if the response variable is spatially dependent.

References

Websites

https://www.epa.gov/so2-pollution

https://rspatial.org/raster/spatial/index.html

Textbooks

Basic Steps in Geostatistics The Variogram and Kriging by Margaret A. Oliver, Richard Webster

Applied Spatial Data Analysis with R by Roger S. Bivand, Edzer J. Pebesma, Virgilio Gómez-Rubio

Regression: Linear Models in Statistics (Springer Undergraduate Mathematics Series)

An Introduction to Statistical Learning with Applications in R by Gareth JamesDaniela WittenTrevor HastieRobert Tibshirani