Beta Workbook (Model Fits, Initial Map)

Samad Patel

8/6/2018

Abstract

The purpose of this workbook is twofold. First, to create time series forecasts for the monthly max amount of ozone (in parts per million) recorded for 24 stations across the state of New York. Second, to use spatial prediction methods to predict the monthly max amount of ozone across the remaining portion of the state.

This workbook demonstrates the intial steps taken to fit an appropriate time-series model, as well as to create a kriging map for the month of June. Further workbooks (and potentially a Shiny App) will build on the steps here to expand the functionality and user-friendliness.

(Nearly) all the code will be demonstrated here. This will likely not be the case in future workbooks.

Part One: Data Wrangling

1.1: Load the Data

The locations and number of stations in New York that have been recording this pollution data haven't remained consistent over the years. For instance, in 2005, there were fewer locations than in 2018. Therefore, we will only go as far back as 2011, as nearly all the stations that exist in 2018 existed in 2011. Only one new station was created in 2018, so we will ignore that station.

```
# We'll only work with the stations we have in the 2018 file.
eighteen <- read_csv('ozone/o3_2018.csv')
# Find relevant stations
# Station 21 isn't included in prior data, so remove here
stations <- unique(eighteen$AQS_SITE_ID)[-21]
```

Now we want to load all the files contained in the ozone folder. We will merge all the files into one dataframe called pol_all (pollution_all).

```
# Load the file names
files <- list.files(path = 'ozone/', pattern = '*.csv')

# Create function to make loading data easier - this will extract only the relevant stations
myread <- function(x){
   file <- read_csv(paste('ozone/', x, sep = ''))
   file <- file %>% filter(file$AQS_SITE_ID %in% stations)
}

# Apply myread to all files
pol_all <- lapply(files, myread) %>% bind_rows()
```

1.2: Aggregate, Transform

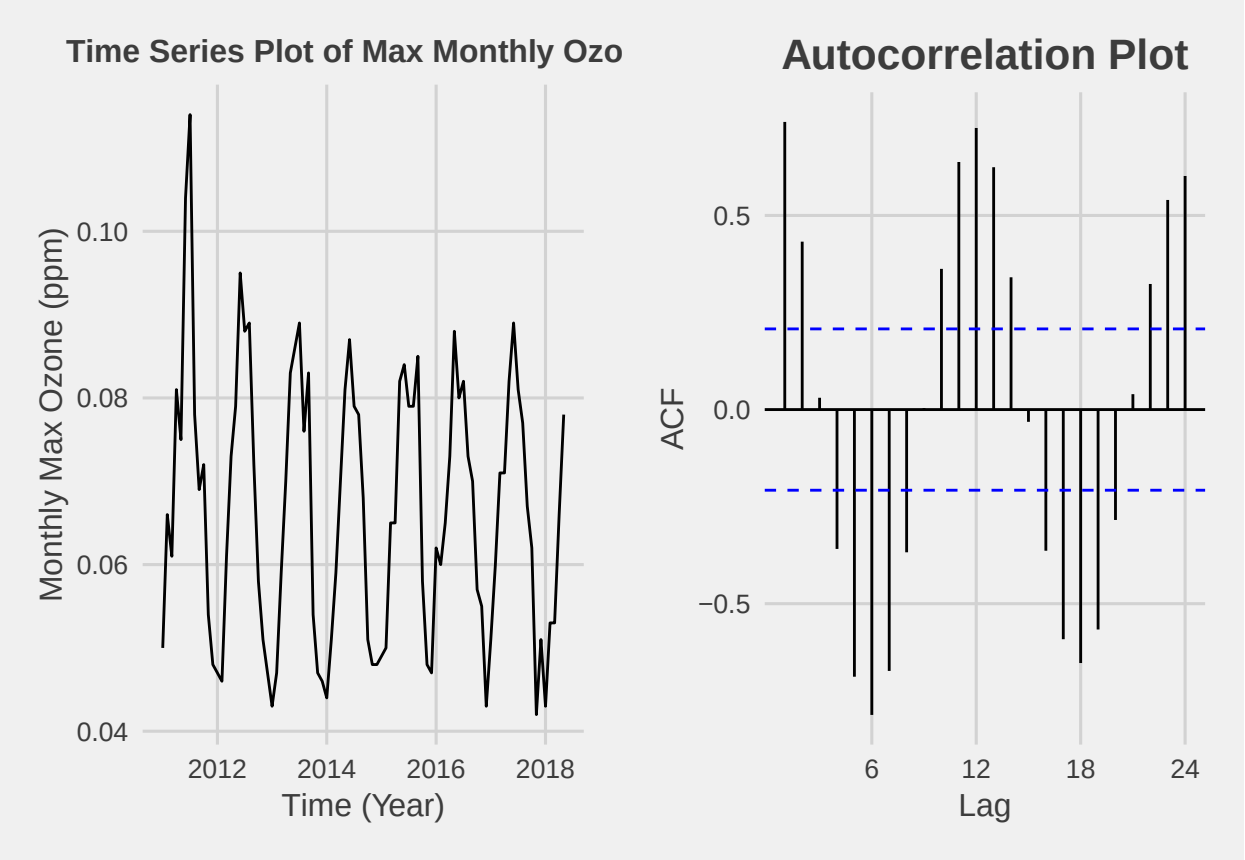
We want to take the monthly maximum values for each station, as that is what we are going to predict. Then we must transform the data into a time-series.

```
pol <- pol_all %>% select(Date, `Daily Max 8-hour Ozone Concentration`)
pol$Date <- as.Date(pol$Date, format = '%m/%d/%Y')
colnames(pol) <- c('date', 'o3')
# Convert time
pol$date <- as.yearmon(pol$date, "%m/%Y")
# Order by ascending date
pol <- pol[order(pol$date), ]
# Groupby day, take averages
pol <- pol %>% group_by(date) %>% summarize('o3' = max(o3))
# Remove date column
pol <- pol[,'o3']
# Make into time-series object
pol_ts <- ts(pol, frequency=12, end = c(2018, 5))
```

Here are some visualizations. The first is simply a plot of our monthly maximums over time. The second is an autocorrelation plot, which demonstrates the amount of correlation that exists between values in prior months.

```
autoplt <- autoplot(pol_ts) +
   theme_fivethirtyeight() +
   theme(plot.title = element_text(hjust = .5, size = 12), axis.title = element_text()) +
   ggtitle('Time Series Plot of Max Monthly Ozone') +
   ylab('Monthly Max Ozone (ppm)') + xlab('Time (Year)')
# Clearly seasonal data
acfplt <- ggAcf(pol_ts) + theme_fivethirtyeight() +
   theme(plot.title = element_text(hjust = .5, size = 16), axis.title = element_text()) +
   ggtitle('Autocorrelation Plot') + ylab('ACF') + xlab('Lag')

grid.arrange(autoplt, acfplt, ncol = 2)
```



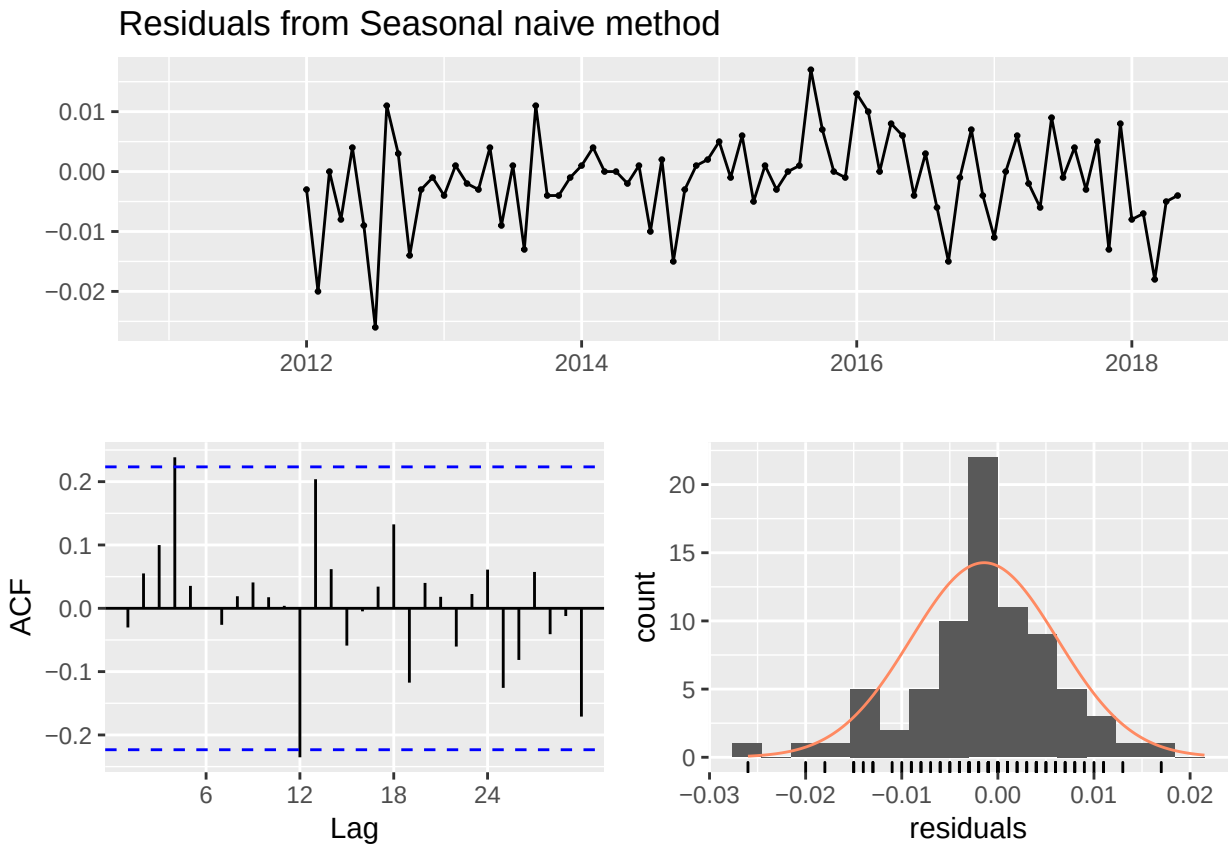
Part Two: Fit Time-Series Models

2.1: Fitting Models

It's clear based on the visualizations above that there is some seasonality to our data. However, there also appears to be a general trend - ozone is decreasing over time. Perhaps climate-change is not a leftist plot to destroy the US - but I'm no expert.

We'll throw out a few basic models - a seasonal naive model, an ETS (error, trend, seasonality), and an ARIMA (autoregressive integrated moving average). For the ARIMA model I'll try a Box Cox transformation to see if that improves the model.

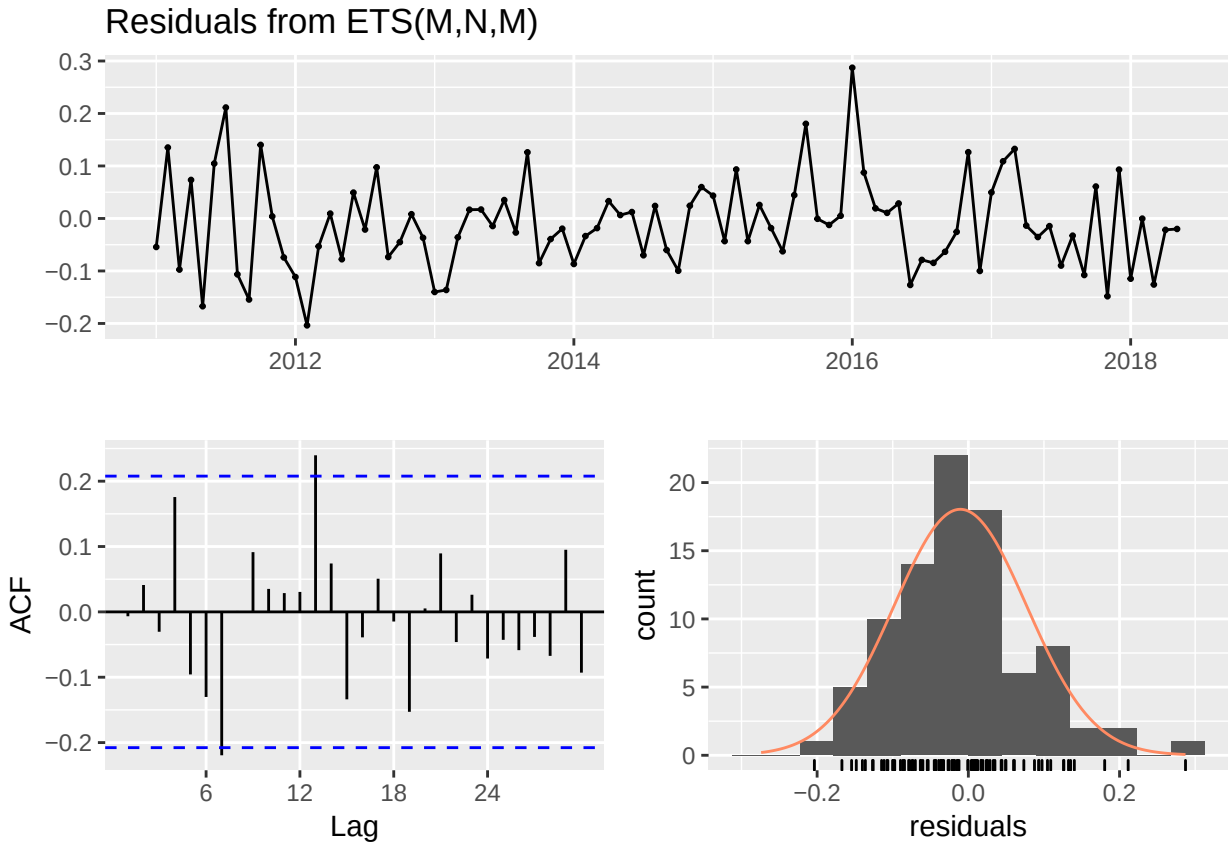
```
fit1 <- snaive(pol_ts, h=7)
checkresiduals(fit1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 16.213, df = 17.8, p-value = 0.5641
##
## Model df: 0.   Total lags used: 17.8
```

Seasonal naive model passes the Ljung-Box test - it's an appropriate fit.

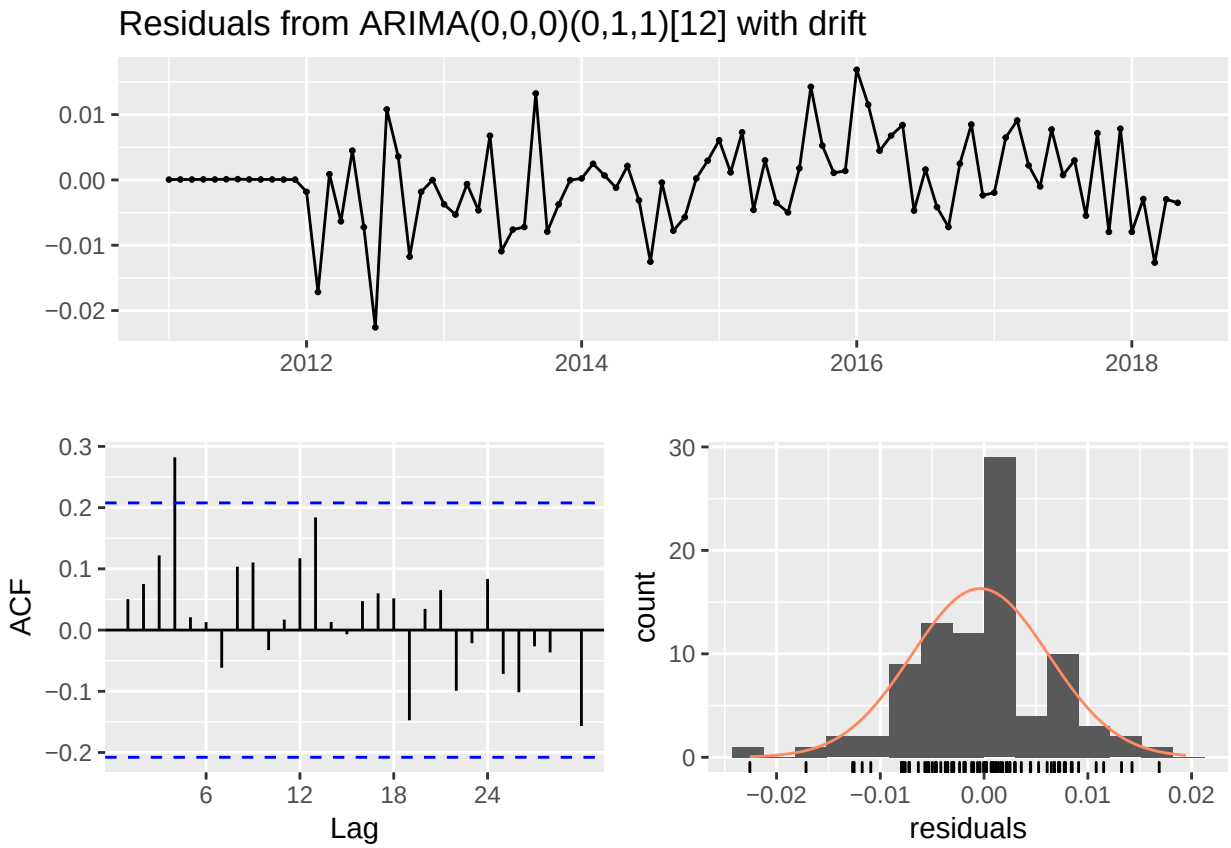
```
fit2 <- ets(pol_ts)
checkresiduals(fit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 20.768, df = 3.8, p-value = 0.0002869
##
## Model df: 14.   Total lags used: 17.8
```

The ETS model fails the Ljung Box test - it's not an appropriate fit.

```
fit3 <- auto.arima(pol_ts)
checkresiduals(fit3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0)(0,1,1)[12] with drift
## Q* = 18.368, df = 15.8, p-value = 0.2906
##
## Model df: 2.   Total lags used: 17.8
```

The ARIMA model passes the Ljung-Box test - it's an appropriate fit.

2.2: Choosing a Model

Now we'll utilize time-series cross-validation to select a model that minimizes mean-squared-error. We'll test between the

```
# Create functions to run through tsCV
fsnaive <- function(x, h){
   snaive(pol_ts, h=h)
}
farima <- function(x, h){
   forecast(auto.arima(x), h=h)
}
farima2 <- function(x, h){
   forecast(auto.arima(x, lambda = BoxCox.lambda(pol_ts)), h=h)
}
# Errors
e1 <- tsCV(pol_ts, fsnaive, h=10)
```

```
e2 <- tsCV(pol_ts, farima, h=10)
e3 <- tsCV(pol_ts, farima2, h=10)

# MSE
paste('MSE of snaive is', round(mean(e1^2, na.rm=TRUE), 5))
```

```
## [1] "MSE of snaive is 0.00048"
```

```
paste('MSE of ARIMA is', round(mean(e2^2, na.rm=TRUE), 5))
```

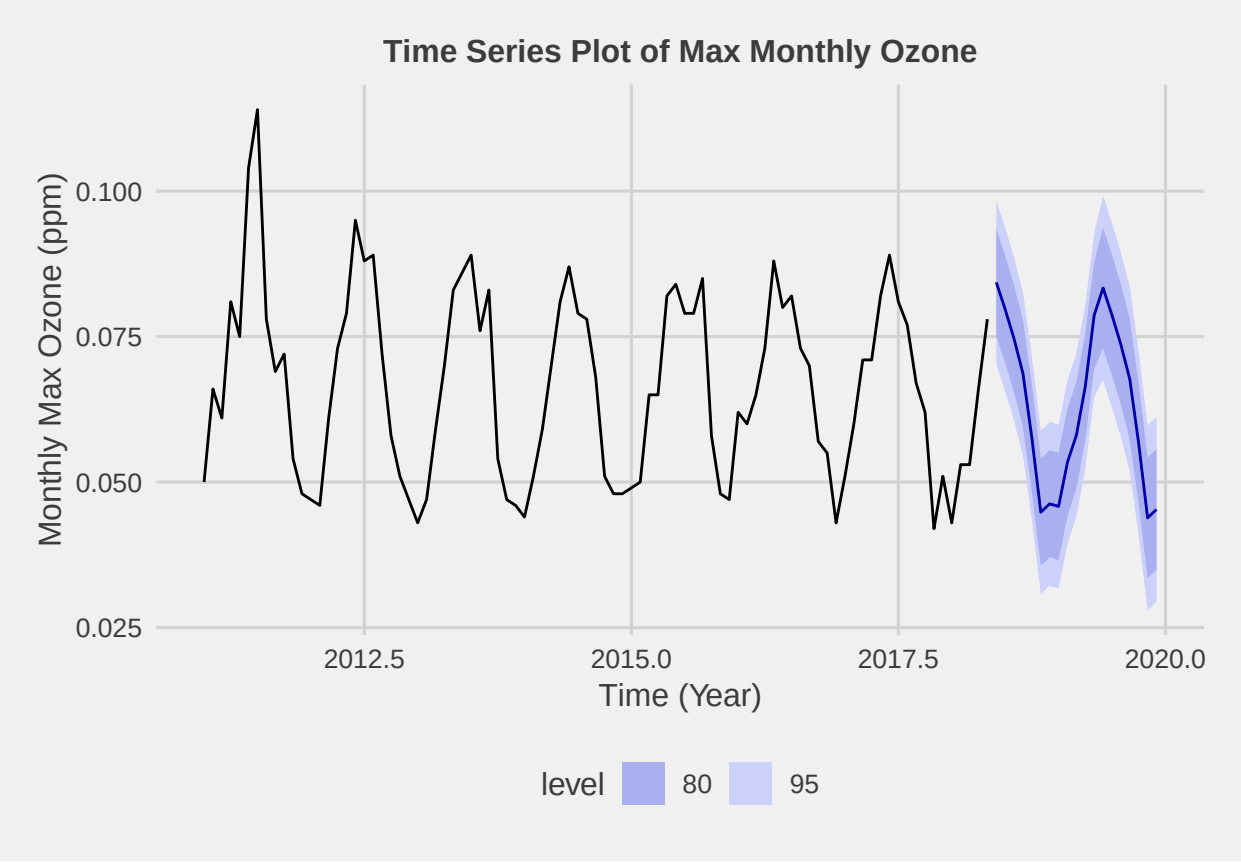
```
## [1] "MSE of ARIMA is 0.00021"
```

```
paste('MSE of transformed ARIMA is', round(mean(e3^2, na.rm=TRUE), 5))
```

```
## [1] "MSE of transformed ARIMA is 0.00054"
```

Time-Series Cross Validation leads us to conclude that the ARIMA model gives us the best fit. Let's take a look at how the forecasts appear from now until 2021:

```
forecast(fit3, h=19) %>% autoplot() +
   theme_fivethirtyeight() +
   theme(plot.title = element_text(hjust = .5, size = 12), axis.title = element_text()) +
   ggtitle('Time Series Plot of Max Monthly Ozone') +
   ylab('Monthly Max Ozone (ppm)') + xlab('Time (Year)')
```



Part Three: Implementing the Model

3.1: Running the loop.

We will now create the predictions for the month of June. The following loop does all the cleaning and modeling steps above, for each id.

```
# Dataframe to save predictions in.
df <- data.frame('id' = unique(pol_all$AQS_SITE_ID),
                 'pred' = rep(NA, 29))
```

```
for (id in df$id){
   ### Clean data ###
   # Parse ID
   pol <- pol_all %>% filter(AQS_SITE_ID == id) %>%
   select(Date, `Daily Max 8-hour Ozone Concentration`)
   pol$Date <- as.Date(pol$Date, format = '%m/%d/%Y')
   colnames(pol) <- c('date', 'o3')
   # Convert time
   pol$date <- as.yearmon(pol$date, "%m/%Y")
   # Order by ascending date
   pol <- pol[order(pol$date), ]
   # Groupby day, take averages
   pol <- pol %>% group_by(date) %>% summarize('o3' = max(o3))
   # Remove date column
   pol <- pol[,'o3']
   # Make into time-series object
   pol_ts <- ts(pol, frequency=12, end = c(2018, 5))

   ### Fit model ###
   # ARIMA
   fit3 <- auto.arima(pol_ts)
   pred <- forecast(fit3, h=1)[4]$mean[1]
   ### Append to df ###
   df[df$id == id, 'pred'] <- pred
}
```

3.2: Completing the DataFrame

We need the latitude and longitude of each site in order to move on.

```
# Select lat and long for each site
latlon <- select(pol_all, SITE_LONGITUDE, SITE_LATITUDE, AQS_SITE_ID) %>% distinct()
# Change varnames for ease of reading
colnames(latlon) <- c('x', 'y', 'id')
# Left join to df
df_ <- left_join(df, latlon, by = 'id')
```

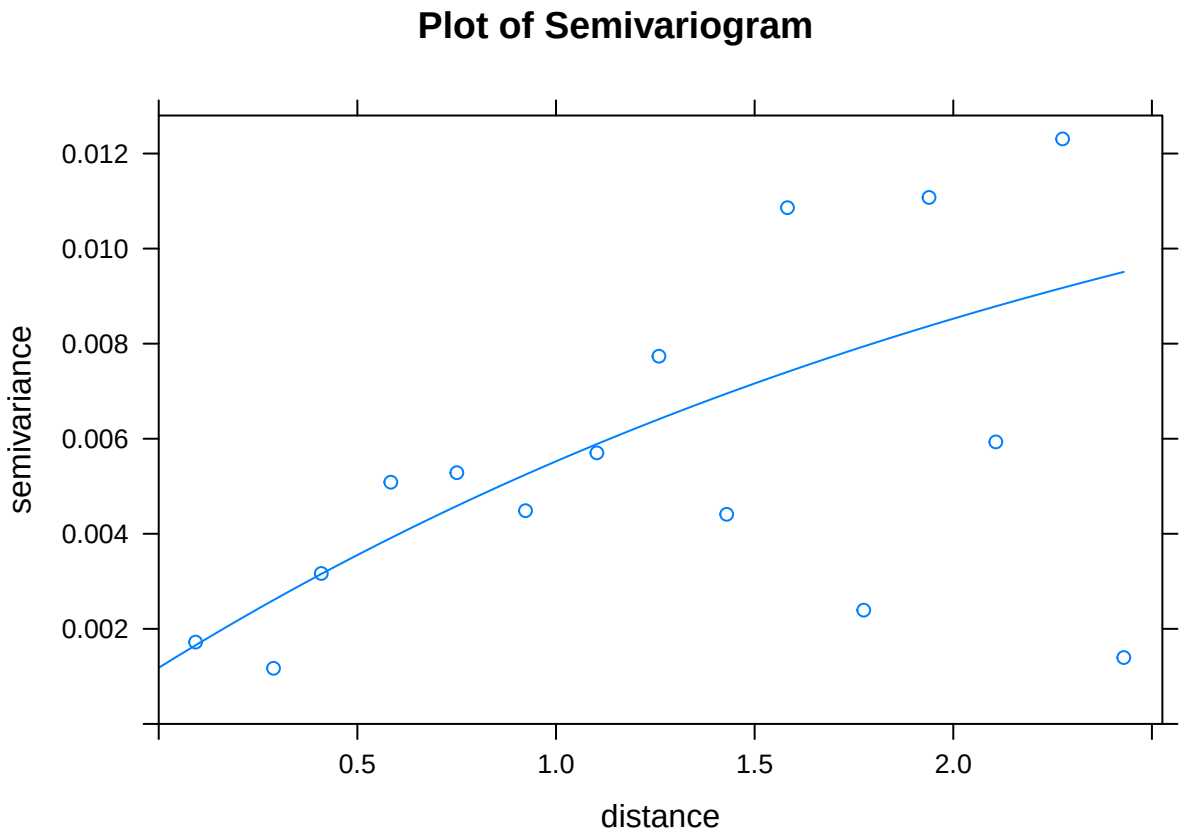
Part Four: Kriging

4.1: Theoretical Variograms

We'll now utilize the kriging method. It's important to check the spatial nature of our data. I will not go in depth at all about the theoretical background of kriging, I will just assume the reader's knowledge.

First we need to fit the theoretical variogram to our data.

```
library(gstat)
# Set seed for consistency
set.seed(60471531)
# Log transformation improves fit
g <- gstat(id='ozone', formula = log(pred)~1, locations = ~x+y, data = df_)
q <- variogram(g)
v.fit <- fit.variogram(q, vgm('Mat'))
plot(q, v.fit, main = 'Plot of Semivariogram')
```



It's clear that no theoretical semivariogram can fit especially well. This indicates that kriging actually won't be an especially effective method for values beyond a certain distance. This demonstrates the weakness in our data - we don't have as many stations as kriging requires. Generally a few hundred observations would make kriging more valid, but because they are well spread out across New York, areas that are far from stations won't be as effectively predicted.

We will now create a grid space for our predictions. This will be the space across which we predict.

```
x.range <- range(df_$x)
y.range <- range(df_$y)
grd <- expand.grid(x=seq(x.range[1], x.range[2], by = .03),
```

```
                    y=seq(y.range[1], y.range[2], by = .03))
```

4.2: Fitting Models

Universal Kriging

```
# UNIVERSAL
u <- gstat(id='o3', formula = log(pred)~x+y, locations=~x+y, data=df_)
v <- variogram(u)
vfit <- fit.variogram(v, vgm('Mat'))
# Predictions
pr_uk <- krige(log(pred)~x+y, data=df_, locations = ~x+y, model = vfit, newdata = grd)
```

```
## [using universal kriging]
```

```
# Cross Validation
cv_uk<- krige.cv(log(pred)~x+y,data=df_, locations=~x+y, model=vfit,nfold=nrow(df_))
# Back transform the PRESS
paste('The PRESS for Universal Krigging is', sum(exp(cv_uk$residual)^2))
```

```
## [1] "The PRESS for Universal Krigging is 29.2732846537697"
```

Ordinary Kriging

```
g <- gstat(id='ozone', formula = log(pred)~1, locations = ~x+y, data = df_)
q <- variogram(g)
v.fit <- fit.variogram(q, vgm('Mat'))
# Log transformation reduces press, improves fit
pr_ok <- krige(id= 'ozone', log(pred)~1, locations=~x+y, model=v.fit,
               data=df_, newdata=grd)
```

```
## [using ordinary kriging]
```

```
cv_ok <- krige.cv(log(pred)~1,data=df_, locations=~x+y, model=v.fit, nfold=nrow(df_))
paste('The PRESS for Universal Krigging is', sum(exp(cv_ok$residual)^2))
```

```
## [1] "The PRESS for Universal Krigging is 29.2553033993847"
```

Ordinary Krigging appears to be slightly better, so we'll go with that.

A note of improvement: these pollutants are highly correlated with other pollutants. Both the ARIMA and kriging models could likely be improved by incororating another (or more than one) chemical that is measured. Unfortunately, downloading that data from the EPA is quite tedious, so I didn't care to do it, but the implementation once the data is collected would not be very difficult.

Part Five: Creating Pollution Maps

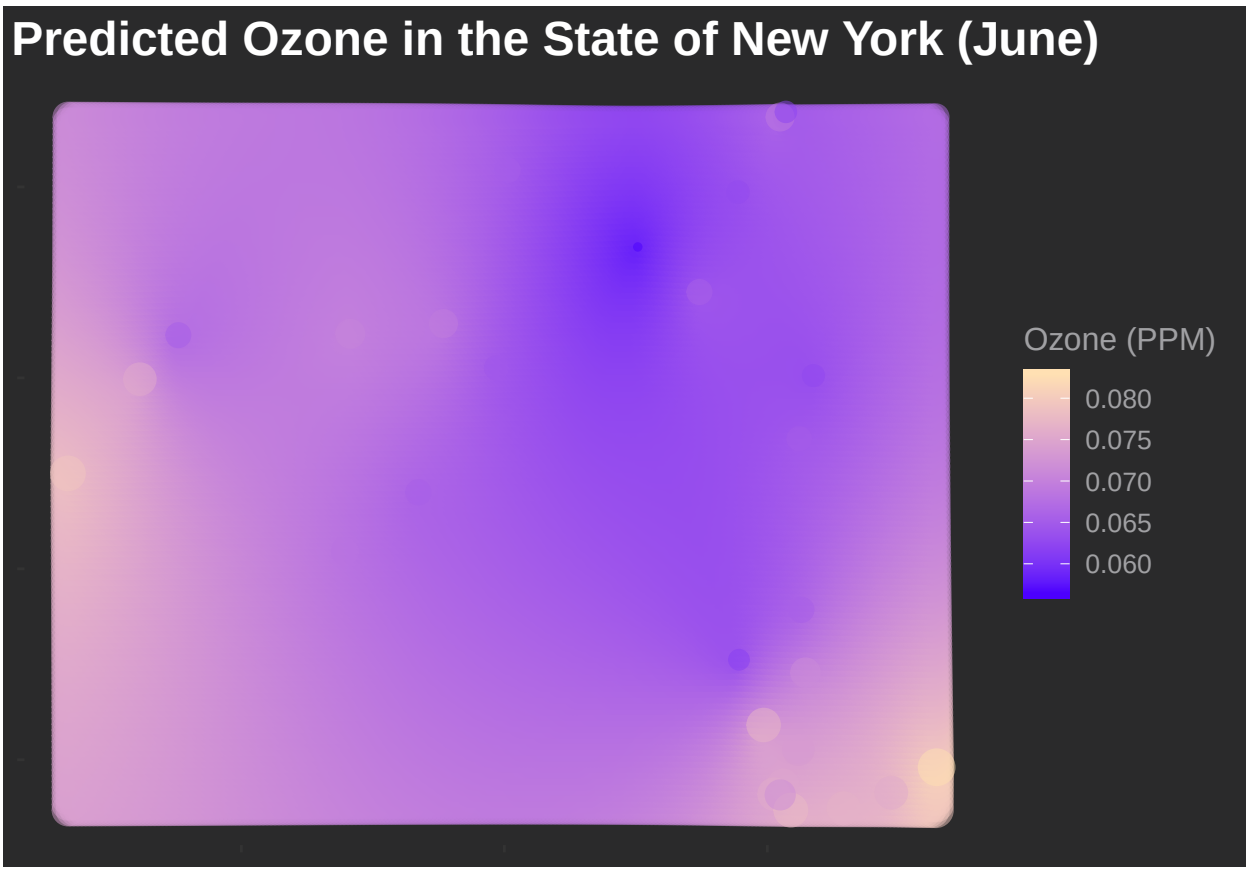
5.1: Cleaning Data

Using the ordinary kriging predicion object (pr_ok), we can save our predictions for every lat and long to a dataframe. This will be the basis of our map.

```
mapdf <- pr_ok[,c('ozone.pred', 'x', 'y')]
mapdf$ozone.pred <- exp(mapdf$ozone.pred)
colnames(mapdf)[1] <- 'pred'
mapdf <- rbind(mapdf, df_[,2:4])
```

But there is a problem here. If we plot this data, we will literally have a grid. It won't be clear that the graph represents NY. Observe:

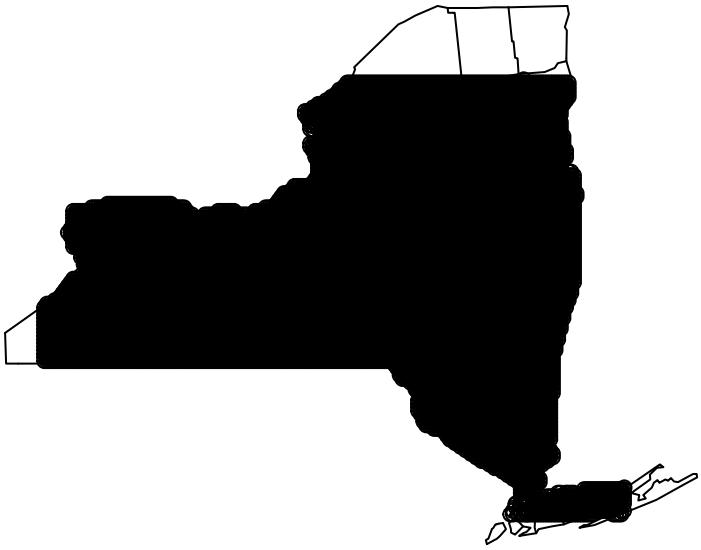
```
ggplot(mapdf) + geom_point(aes(x = x, y = y, size = pred,
                               color = pred), alpha = .5) +
   theme_hc(bgcolor = 'darkunica') +
   ggtitle('Max Predicted Ozone in the State of New York (June)') +
   theme(legend.position = 'right', legend.direction = 'vertical',
         plot.title = element_text(hjust = .5, size = 18, face = 'bold'),
         axis.text = element_blank(),
         axis.title.x = element_blank(), axis.title.y = element_blank(),
         panel.grid.major.y = element_blank()) +
   scale_color_gradient(low = "#4C00FFFF", high = "#FFE0B3FF", name = 'Ozone (PPM)') +
   scale_size_continuous(guide = FALSE)
```



We need to remove points that are not in New York. The following code will accomplish that:

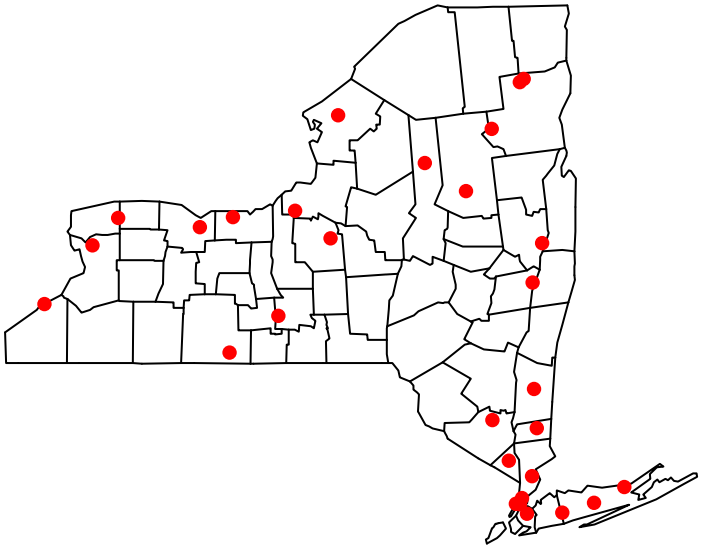
```
### USING THE MAPS LIBRARY ###
# Vector that determines which state each point in the grid is in
in_what_state <- map.where(database="state", mapdf$x, mapdf$y)
# Subset only those that are in new york - call it in_ny
in_ny <- which(in_what_state == 'new york:main'|in_what_state == 'new york:long island'|
```

```
                  in_what_state == 'new york:manhattan')
# Subset those values from mapdf
mapdf <- mapdf[in_ny, ]
# Confirm that our map is correct
map('county', 'new york'); points(mapdf$x, mapdf$y)
```



The above plot confirms that all our points lie in New York. Interestingly enough, no pollution monitoring stationns exist in the top of the state. Here are the stations only:

```
map('county', 'new york')
points(df_$x, df_$y, pch = 16, col = 'red')
```



We ought to fix the grid so that it's predicting all values in New York. The map function contains the entire range of NY, so we can use that for our grid and re-run our kriging code to predict over the new grid.

```
map('county', 'new york', plot=FALSE)$range
```

```
## [1] -79.76718 -71.87756  40.48520  45.01157
```

```
# New grid
x.range <- map('county', 'new york', plot=FALSE)$range[1:2]
y.range <- map('county', 'new york', plot=FALSE)$range[3:4]
grd <- expand.grid(x=seq(x.range[1], x.range[2], by = .03),
                   y=seq(y.range[1], y.range[2], by = .03))

# Kriging
g <- gstat(id='ozone', formula = log(pred)~1, locations = ~x+y, data = df_)
q <- variogram(g)
v.fit <- fit.variogram(q, vgm('Mat'))
# Log transformation reduces press, improves fit
pr_ok <- krige(id= 'ozone', log(pred)~1, locations=~x+y, model=v.fit,
               data=df_, newdata=grd)
```

```
## [using ordinary kriging]
```

```
# Create initial mapdf
mapdf <- pr_ok[,c('ozone.pred', 'x', 'y')]
mapdf$ozone.pred <- exp(mapdf$ozone.pred)
colnames(mapdf)[1] <- 'pred'
mapdf <- rbind(mapdf, df_[,2:4])

# Select only points in NY
# Vector that determines which state each point in the grid is in
in_what_state <- map.where(database="state", mapdf$x, mapdf$y)
# Subset only those that are in new york - call it in_ny
in_ny <- which(in_what_state == 'new york:main'|in_what_state == 'new york:long island'|
                 in_what_state == 'new york:manhattan')
# Subset those values from mapdf
mapdf <- mapdf[in_ny, ]
# Confirm that our map is correct
map('county', 'new york'); points(mapdf$x, mapdf$y)
```

5.2: Final Map

```
ggplot(mapdf) + geom_point(aes(x = x, y = y, size = pred,
                               color = pred), alpha = .5) +
   theme_hc(bgcolor = 'darkunica') +
   ggtitle('Max Predicted Ozone in the State of New York (June)') +
   theme(legend.position = 'right', legend.direction = 'vertical',
         plot.title = element_text(hjust = .5, size = 18, face = 'bold'),
         axis.text = element_blank(),
         axis.title.x = element_blank(), axis.title.y = element_blank(),
         panel.grid.major.y = element_blank()) +
   scale_color_gradient(low = "#4C00FFFF", high = "#FFE0B3FF", name = 'Ozone (PPM)') +
   scale_size_continuous(guide = FALSE)
```