# Predmatic Project

Shanie Talor

5/12/2022

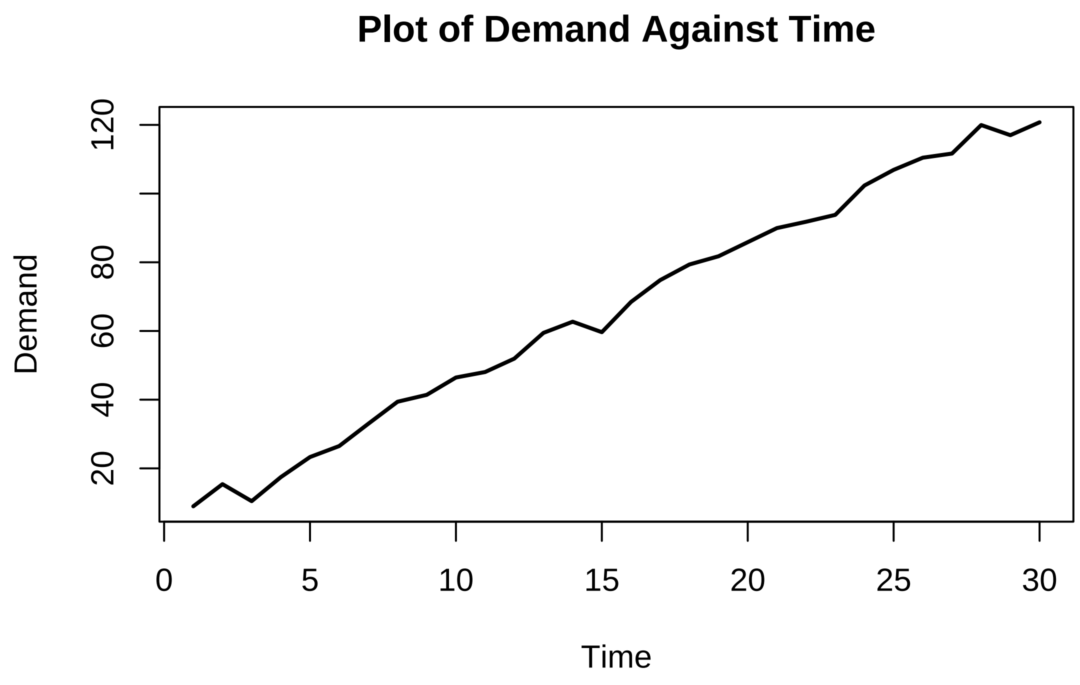
## Table of Contents

1. Create Time Series Variables
2. Picking Our Model
3. Training/Testing Data and Forecasting using Model 1
4. Training/Testing Data and Forecasting using Model 2
5. Conclusion

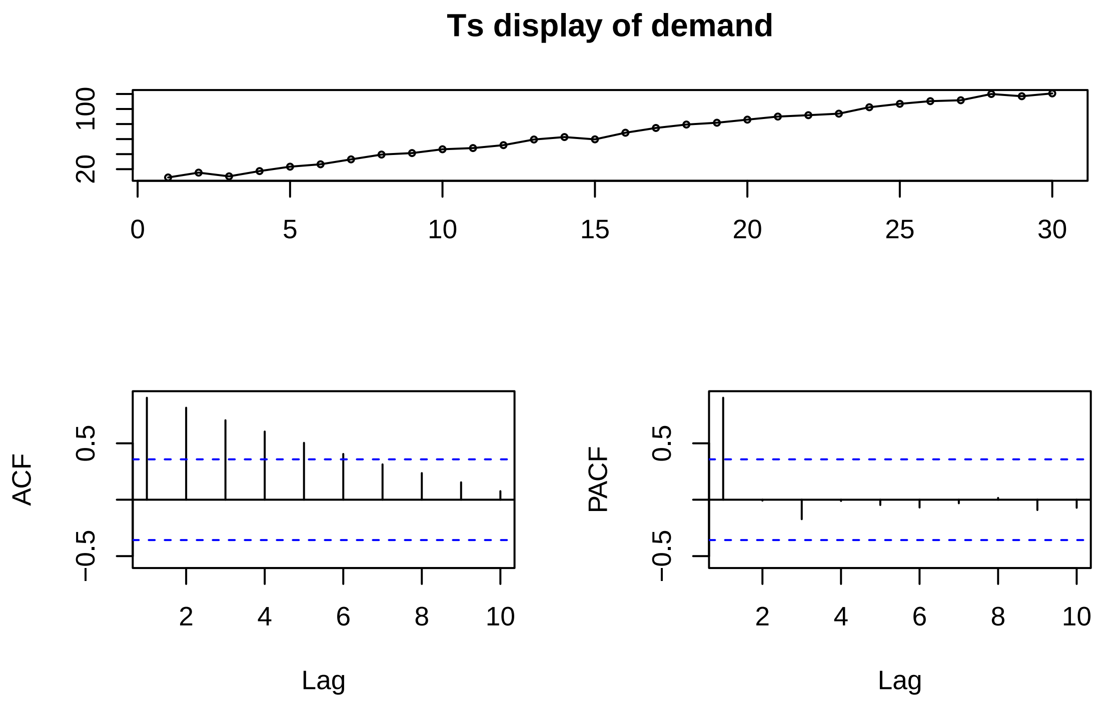
## 1. Create Time Series Variables

1. Make each variable a time series variable.
2. Create a new variable "xreg" containing both fictitious variables "var.1" and "var.2".
3. Plot the adjusted demand variable "demand" against time to check for seasonality.
4. Look at the ACF and PACF .

```r
data.p = read.csv("predmatic.csv")

demand<-ts(as.double(data.p[c(1:30), 2]),start=1)
var.1 = ts(as.double(data.p[c(1:30), 3]), start = 1)
var.2 = ts(as.double(data.p[c(1:30), 4]), start = 1)
xreg = cbind(var.1, var.2)

plot(demand,xlab='Time', ylab="Demand", main = "Plot of Demand Against Time", lwd=2)
```



```
tsdisplay(demand, main = "Ts display of demand")
```

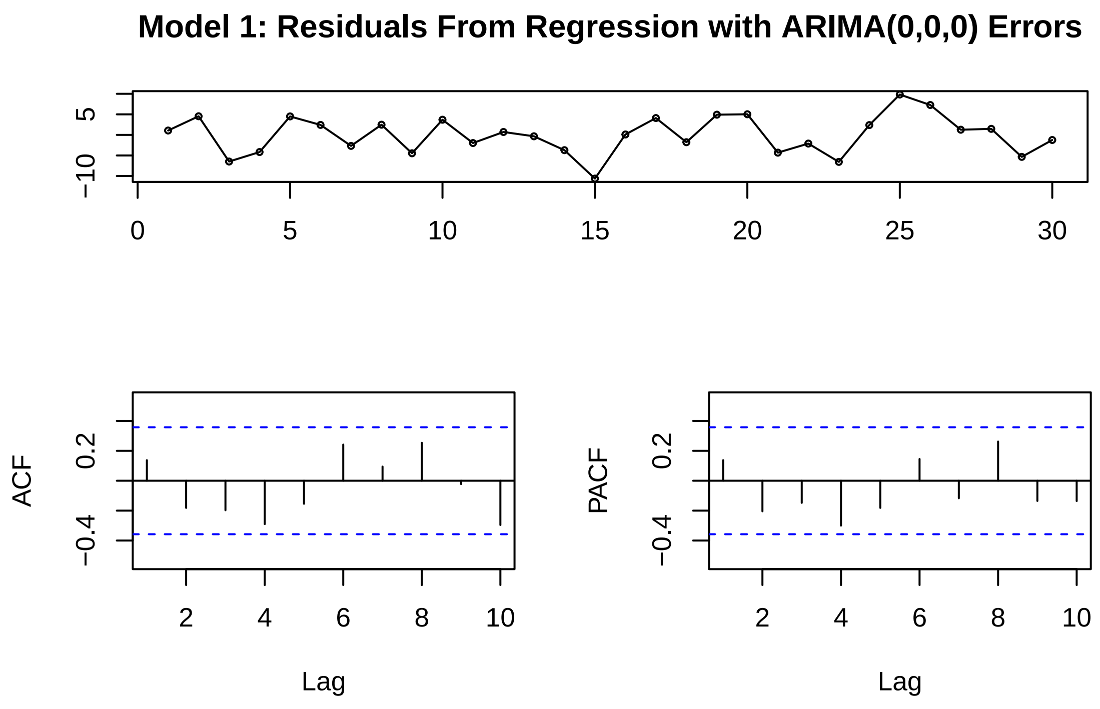


### Results

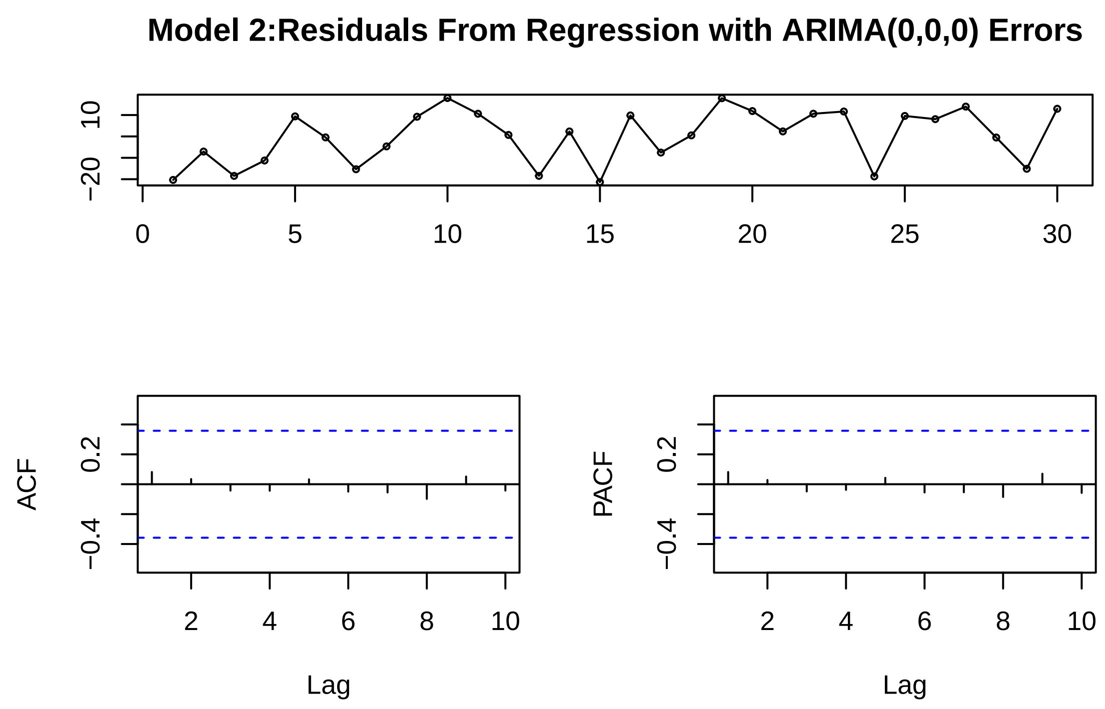
- Based off the graphs,it appears that the data is already adjusted; the variance is stabilized, there are no significant lags in the PACF, and the ACF is gradually decreasing to 0.
  - We can use demand as is.
  - The next step is to determine which fictitious variables to use in our ARIMA model.

## 2. Picking Our Model

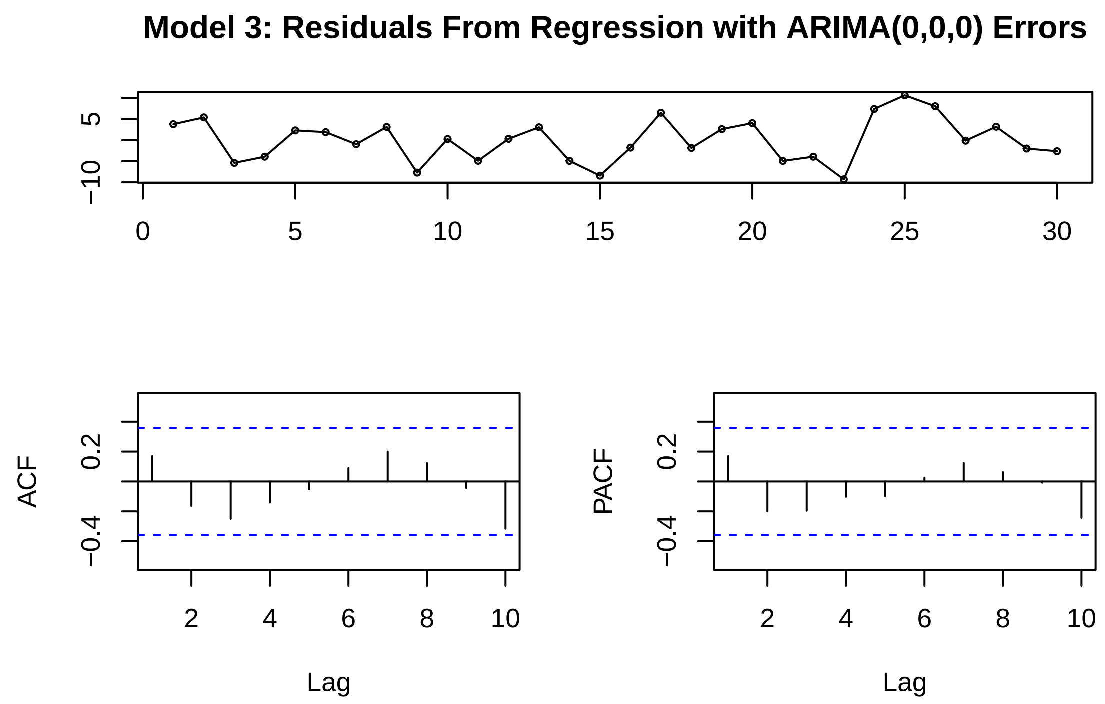
```
#model with demand using both fictitious variables
model1.capture = capture.output(auto.arima(demand, xreg = xreg, trace = TRUE))
model1= auto.arima(demand, xreg = xreg)

tsdisplay(model1$residuals, main = "Model 1: Residuals From Regression with ARIMA(0,0,0) Errors")
```



```
#model with demand using variable 1
model2.capture = capture.output(auto.arima(demand,  xreg = var.1, trace = TRUE))
model2= auto.arima(demand, xreg = var.1)


tsdisplay(model2$residuals, main = "Model 2:Residuals From Regression with ARIMA(0,0,0) Errors")
```



```
#model 4 with demand using variable 2
model3.capture = capture.output(auto.arima(demand,  xreg = var.2, trace = TRUE))
model3= auto.arima(demand, xreg = var.2)
tsdisplay(model3$residuals, main = "Model 3: Residuals From Regression with ARIMA(0,0,0) Errors")
```



```r
#data frame of the different models and their corresponding AIC and RMSE
AIC = c(model1[["aicc"]], model2[["aicc"]], model3[["aicc"]])
RMSE = c(accuracy(model1)[,2],accuracy(model2)[,2], accuracy(model3)[,2])
Type = c("Model 1", "Model 2", "Model 3")

AIC.d.f. = data.frame(Type,AIC, RMSE)
AIC.d.f.
```

```
##      Type      AIC      RMSE
## 1 Model 1 185.0834  4.507688
## 2 Model 2 242.7681 12.327255
## 3 Model 3 188.5022  4.989726
```

```r
#min AIC and RMSE is from model 1
min(AIC)
```

```
## [1] 185.0834
```

```r
min(RMSE)
```

```
## [1] 4.507688
```

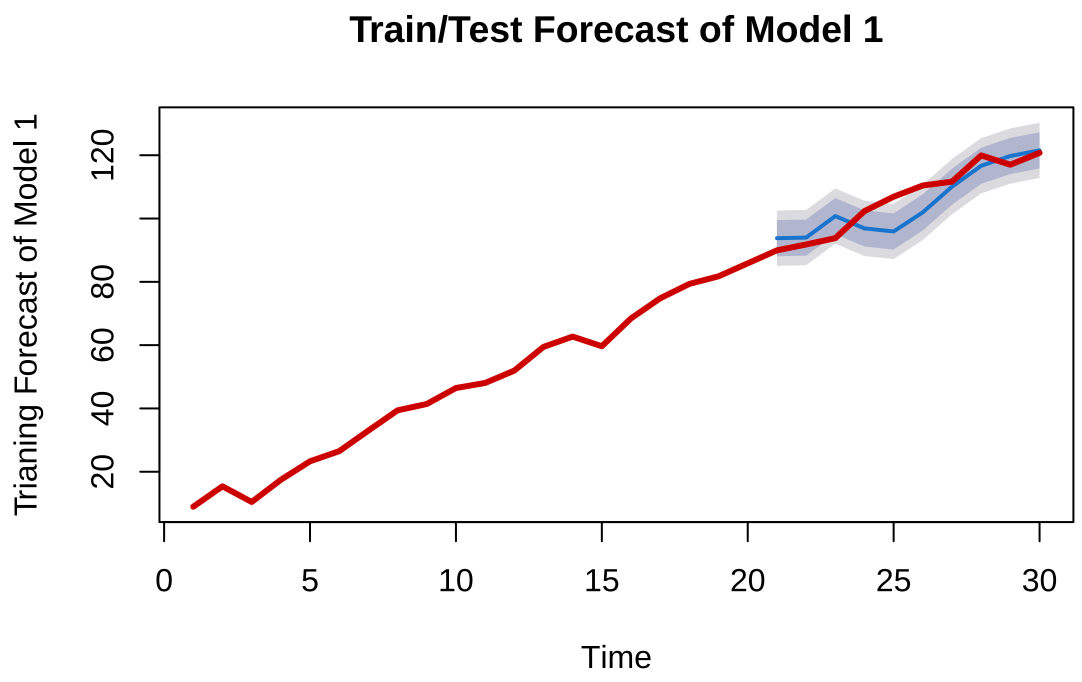
### Results

- Model 1-3 all return:" Best model: Regression with ARIMA(0,0,0) errors "
  - This confirms that our data using the variables are consistent with white noise.
- Our next step is to determine which model is the best fit using AIC and RMSE.
  - Looking at the AIC/RMSE table, we can see that model 1 has the lowest AIC and RMSE.
  - Looking at the tsdisplay of model 1, we can further confirm that model 1 is consistent with white noise.
- Lets train and test our data using known values to see how model 1 performs.
  - Lets also look at model 3 since the AIC and RMSE was so similar to model 1.

## 3. Training/Testing Data and Forecasting using Model 1

```r
#2/3 of the observations of demand
demand_train = subset( demand, start = 1, end = 20)

#2/3 of observations of xreg: var 1 and var 2
model1_train <- subset(xreg, start = 1, end = 20)

# the remaining 1/3 observations of xreg
model1_test<- subset(xreg, start = 21, end = 30)

#create a fitted model using the trained demand and trained variables
model1_train.arima = auto.arima(demand_train,  seasonal=FALSE, xreg = model1_train)
```

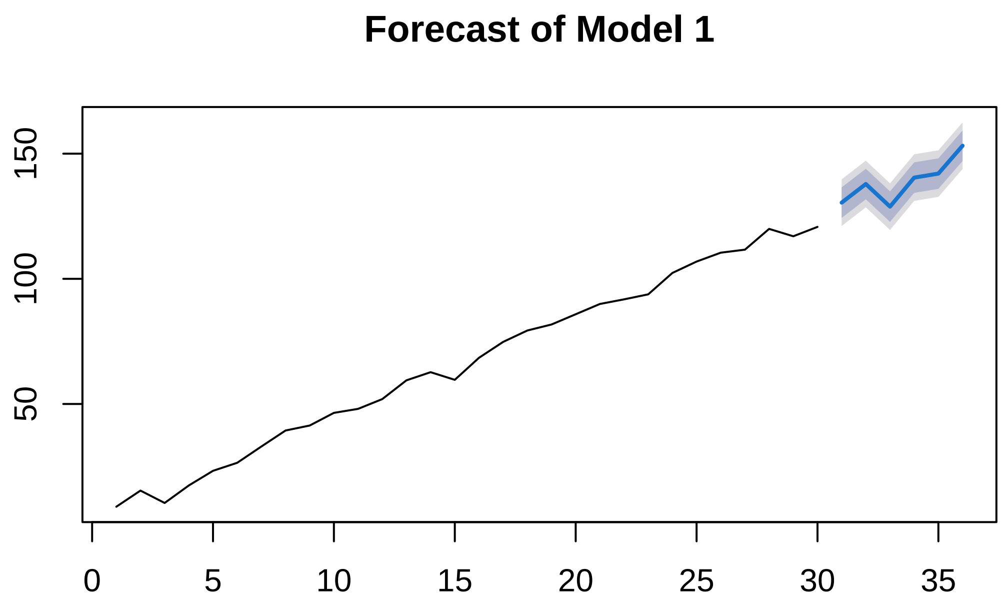
```r
#forecast demand using the test variables to compare with the remaining demand values
model1.train.forecast = forecast(model1_train.arima, xreg = model1_test)

#plot the forecasted demand values against the known demand values
plot(model1.train.forecast,ylab="Trianing Forecast of Model 1", xlab="Time", main = "Train/Test Forecas
lines(demand,col="red3",lwd=3)
```



```r
#extend variable 1 and 2 to include the forecasted values
var.1.test = ts(as.double(data.p[c(1:36), 3]), start = 1)
var.2.test = ts(as.double(data.p[c(1:36), 4]), start = 1)
xregtest = cbind(var.1.test,var.2.test)



xreg.predict <- subset(xregtest, start = 31, end = 36)
model1.forecast = forecast(model1, xreg = xreg.predict)
model1.forecast$mean
```

```
## Time Series:
## Start = 31
## End = 36
## Frequency = 1
## [1] 130.4345 137.8751 128.8350 140.4333 142.0343 153.1766
```

```r
plot(model1.forecast, main = "Forecast of Model 1")
```

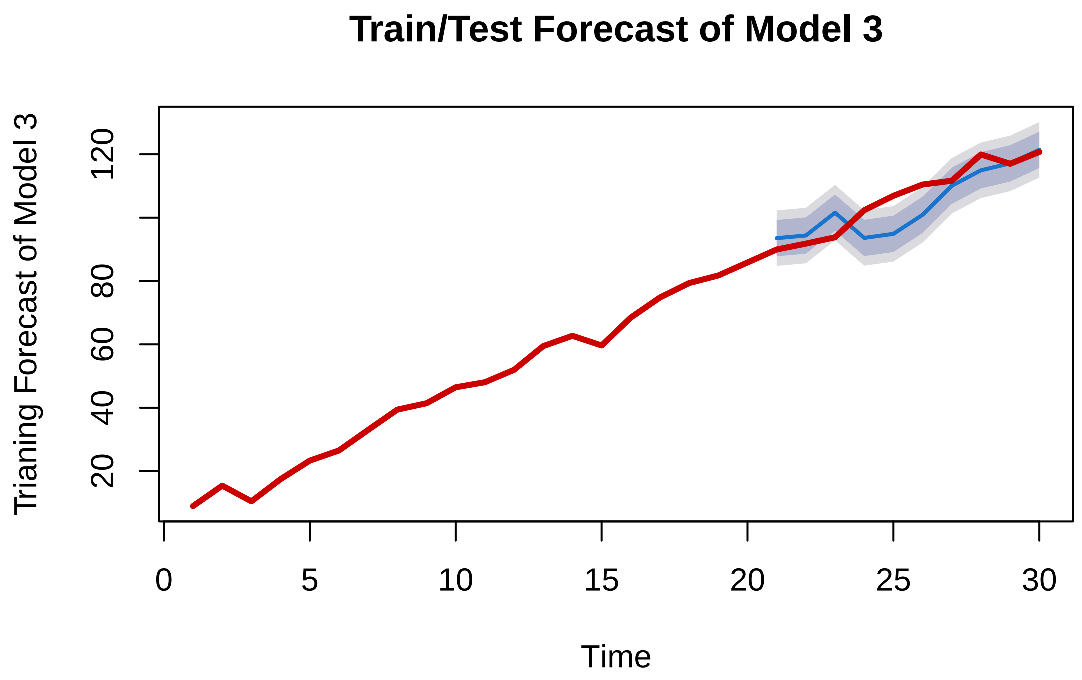


### Results

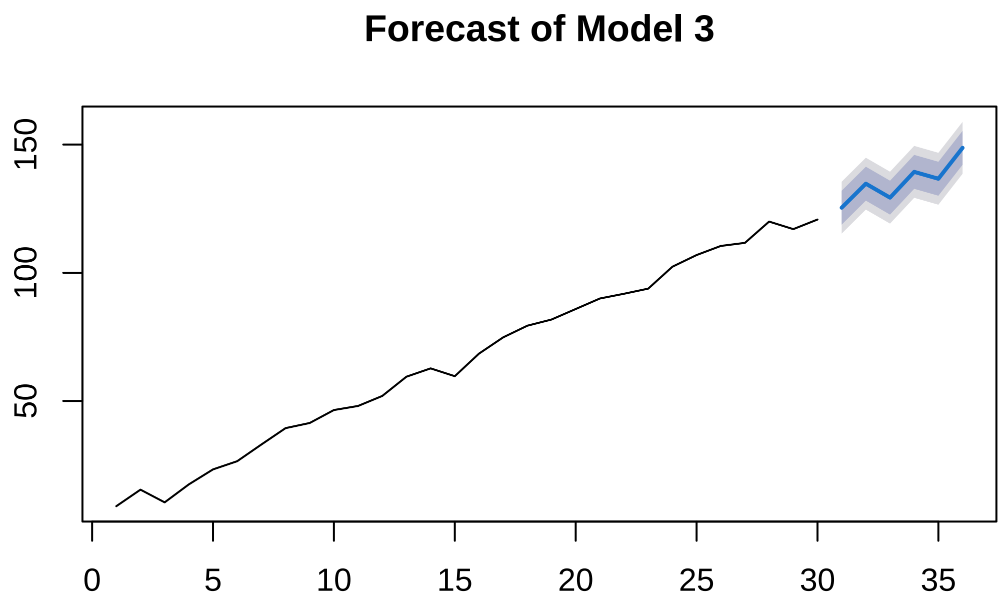
- Model 1 appears to preform relatively well using the training/testing data.
- The forecast of the next 6 months seems to follow the trend of previous data as well.

## 4. Training/Testing Data and Forecasting using Model 2

```
#2/3 of observations of xreg: var 1 and var 2
model3_train <- subset(var.2, start = 1, end = 20)

# the remaining 1/3 observations of xreg
model3_test<- subset(var.2, start = 21, end = 30)

#create a fitted model using the trained demand and trained variables
model3_train.arima = auto.arima(demand_train,  seasonal=FALSE, xreg = model3_train)

#forecast demand using the test variables to compare with the remaining demand values
model3.train.forecast = forecast(model3_train.arima, xreg = model3_test)

#plot the forecasted demand values against the known demand values
plot(model3.train.forecast,ylab="Trianing Forecast of Model 3", xlab="Time", main = "Train/Test Forecas
lines(demand,col="red3",lwd=3)
```



```
model3.reg.predict <- subset(var.2.test, start = 31, end = 36)
model3.forecast = forecast(model3, xreg = model3.reg.predict)

plot(model3.forecast, main = "Forecast of Model 3")
```

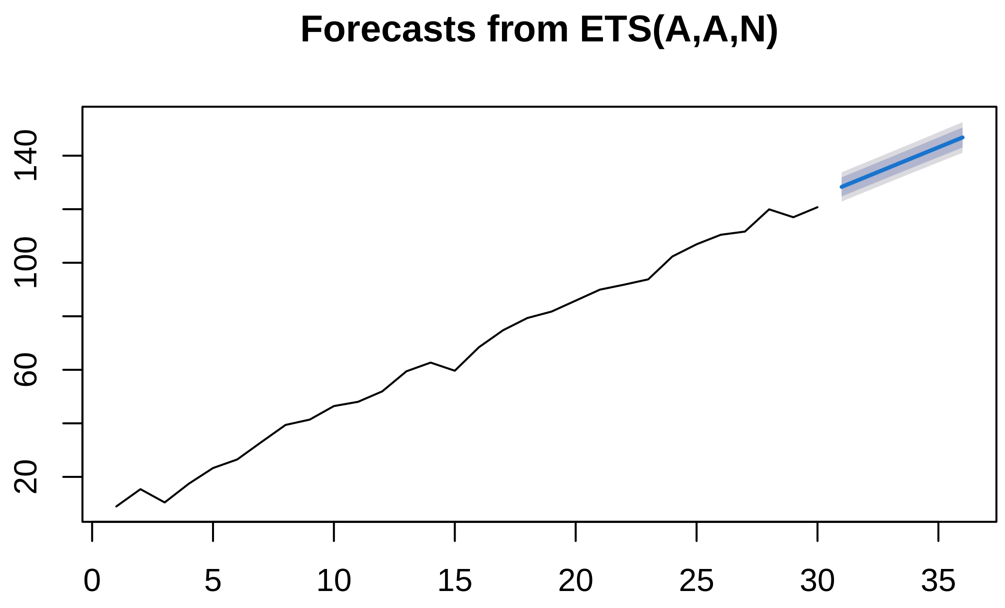


### Results

- It appears model 3 preforms very similarly when using the training and testing data, however model 1 seems to perform a little better. This does not necessarily imply a better overall model, however we will continue using Model 1 as our best fit.

## 5. Conclusion

```
plot(forecast(demand, h = 6))
```



```
model1.forecast$mean
```

```
## Time Series:
## Start = 31
## End = 36
## Frequency = 1
## [1] 130.4345 137.8751 128.8350 140.4333 142.0343 153.1766
```

### Result

- Looking at an ETS forecast can give us an idea of what we want our 6 predicted values to look like.
- Forecasting our model 1 using AUTO ARIMA, we see that the predicted plot has some dynamics compared to the linear ETS forecast.
- Based on the previous tests can safely conclude that predicted values for the next six months are: **130.4345, 137.8751, 128.8350, 140.4333, 142.0343, 153.1766**