**Time Series Analysis of S&P 500 Index**

**Abstract:**

The S&P 500 price index indicates the performance of 500 large businesses in the United States. It is also a common indicator of how the United States economy is performing overall. The dataset used contains raw daily un-adjusted closing stock prices from 2 January 1986 until 28 June 2018. The objective of this study is to analyze the performance of the S&P 500, to find an appropriate time series model for the S&P, and to forecast based on the fitted time series model. Using this forecast, we can analyze the health of the US Economy and make judgements on investments against a standard S&P 500 index.

**Introduction:**

S&P 500 is a stock market index that measures the stock performance of 500 large companies listed on stock exchanges in the United States, and considered to be one of the benchmark return in financial market as well as a leading economic indicator.

As one of the representatives of stock price indices, S&P 500 is always considered as the benchmark return in the stock market and widely used as market premium in lots of famous asset pricing model like Capital Asset Pricing Model (CAPM). Based on that investors can calculate the required rate of return for an individual asset by multiplying the asset's beta coefficient by the market coefficient, then adding back the risk-free rate. It is an important indicator for both individual or institutional investors to determine their investment strategy. Also, many mutual funds design passive funds exactly following S&P 500 Index to generate return for long term investment.

S&P 500 Index is widely used as a leading economic indicator because the overall performance of the stock market indicates the confidence of future growth. Generally, a rise in stock prices means investors are more confident of future growth. A fall in the stock market means investors are rushing toward traditional safe havens like T-bills. In this sense, S&P 500, or the performance of the stock market can provide us with a preview of the next phase of the business cycle.

In this time series analysis, we analyze the log transformed S&P 500 price and the log return of S&P 500 with various time series models and make forecasts based on the models accuracy. Our analysis shows that a SETAR model performs the best in predicting the cumulative log return of the S&P 500.

**Data:**

We download the data from the website Kaggle. The data consists of daily index prices from 2 January 1986 until 28 June 2018. The data has a general upward trend, with increased volatility over time. Two recessions can be observed in 2004 and 2008. In the past 8 years, the S&P 500 has been growing at a higher rate than in the late 1900s. The first data point was taken from every month because this reflects the trend of the S&P 500 and allows us to compare the data with monthly data points from the same time period that describe the Unemployment Rate, the Consumer Price Index (CPI), the US Dollar Index, the Crude Oil Price and the 10 year Treasury Note. The plot of the monthly data with ACF and PACF is shown in Figure 1.

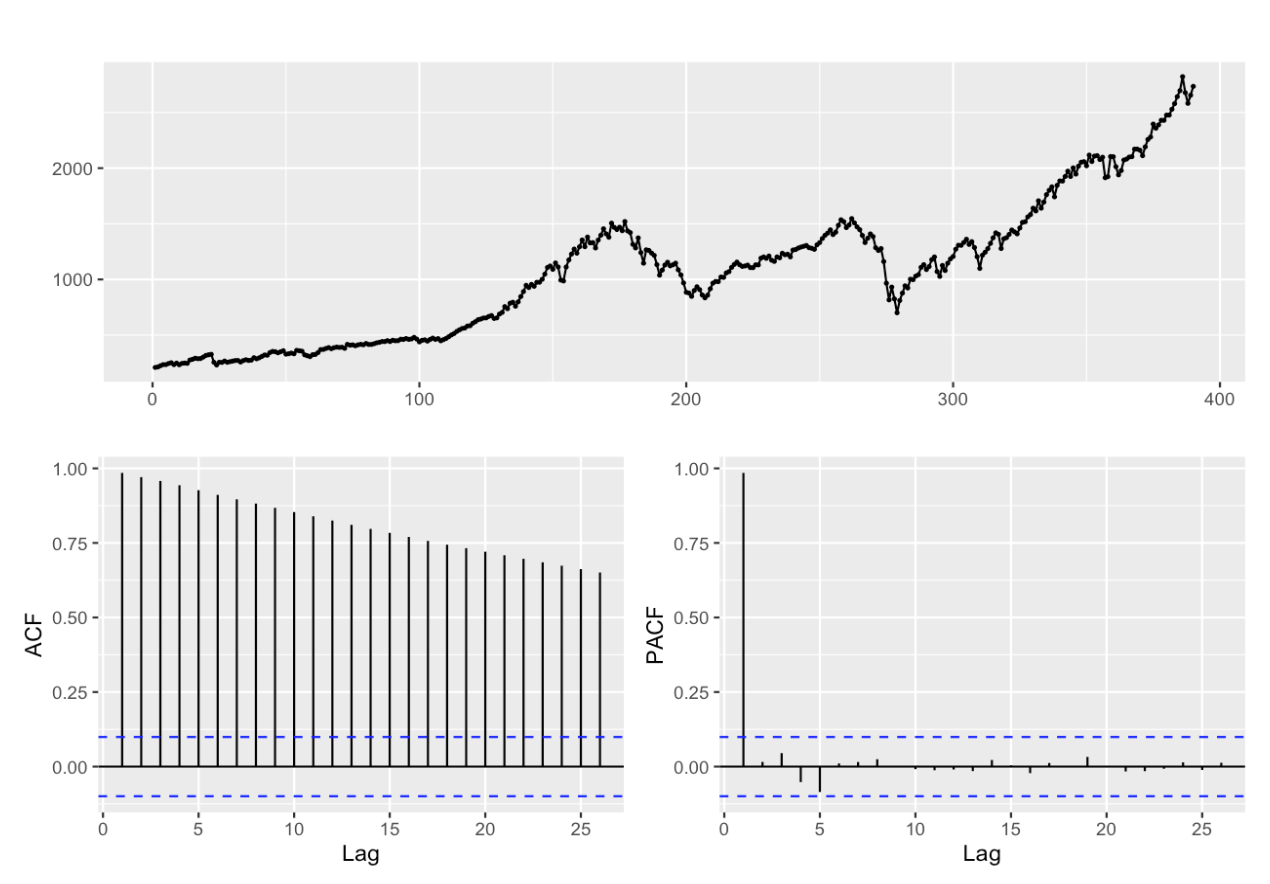


Figure 1: Index price from 1986 to 2018, ACF and PACF

**Preprocessing:**

A non-constant variance can be observed from the index, so a log transformation was computed to stabilize the variance. This has a physical meaning: by taking the difference in log price within 2-timestamps,the result is the cumulative log return within that period. The Box-Cox function returns a lambda of 0.08033, which indicates that a log transformation a justified transformation. Stationarity is necessary for several of the models being explored. To achieve this, the log-transformed data is differenced. This is the log return of the S&P 500 index. The Augmented Dickey Fuller test confirms this with a p-value <.01 , rejecting the null hypothesis of non-stationarity.

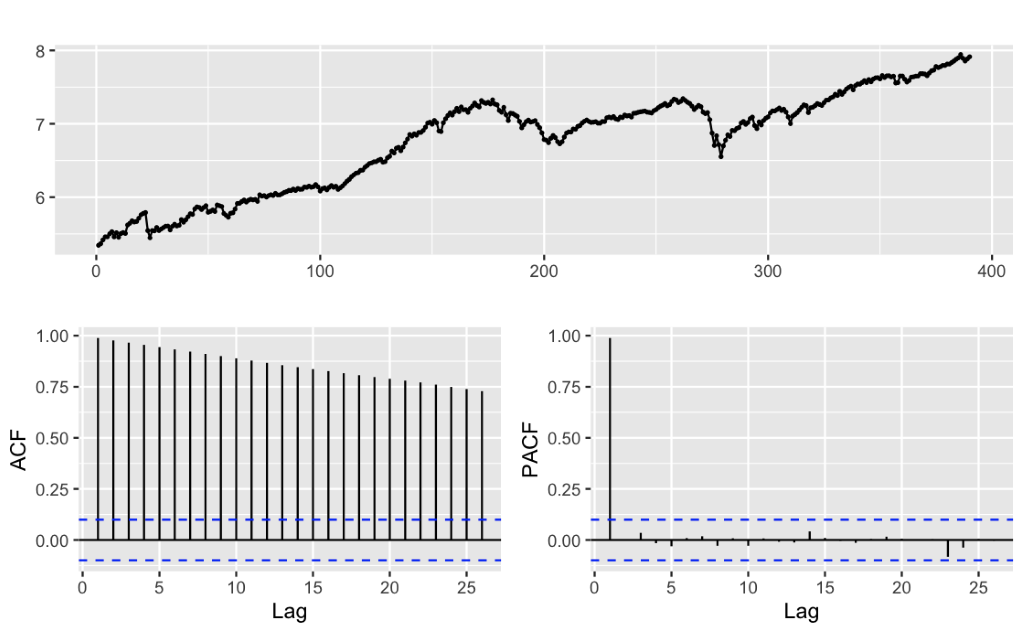
****

Figure 2.1: Plot, ACF, and PACF of log transformation of S&P 500

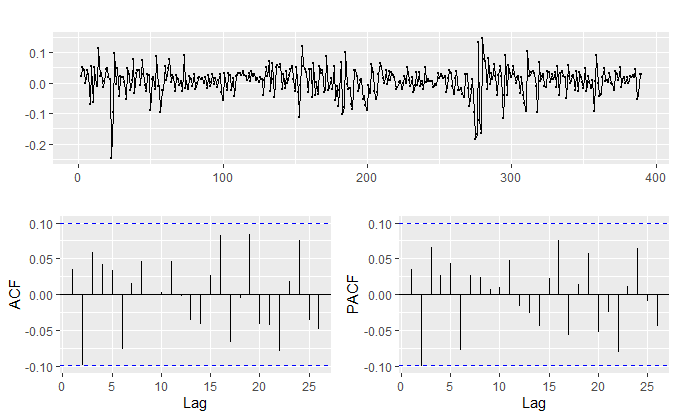
 

Figure 2.2: Plot, ACF, and PACF of log return of S&P 500

**Data split & Out of sample test data:**

We split the data into training and testing data and out of sample data. The training data is composed of 378 points from January 1986 to June 2017. The testing data, 12 points, is from July 2017 to June 2018. Finally, The out of sample data, 18 points, is from July 2018 to December 2019 for out of sample forecast using our best model. Our purpose is to find out if we can predict the cumulative log return of a year given the cumulative data as well as if we can capture the long-term trend of S&P 500 data using our best model.

**Methods:**

1. Time series regression

We first found multiple indexes in the same period that could be related to the S&P 500 index, which are the US unemployment rate, the US dollar index, the CPI, and the Price of Crude Oil and the 10-year treasury notes. US unemployment and CPI reflect whether the economy is booming or is in a recession; US dollar index can affect S&P 500 in terms of foreign investing; being the most important raw material of industry, the crude oil cost is a good barometer of economy. (High crude oil prices generally means a booming economy.) ; the yield of 10-year treasury notes reflects level of risk free rate, which has a counteractive effect on the S&P 500 index. We didn’t choose the GDP, as our data is monthly-based, but the shortest-term data for the GDP is quarterly data. In order to achieve residuals that resemble white noise, we had to do a regression based on the log return of the S&P 500. Then we tried to fit regression models on combinations of these parameters. We chose the model with the smallest AIC and Cross Validation error, which is a linear regression based on the parameters of US dollar index, CPI, and 10-year treasury notes. The model’s residuals are white noise, based on the residuals’ plots (Figure A.1) and the p-value of Ljung-box test(p-value=0.096). The model’s equation is as follows:

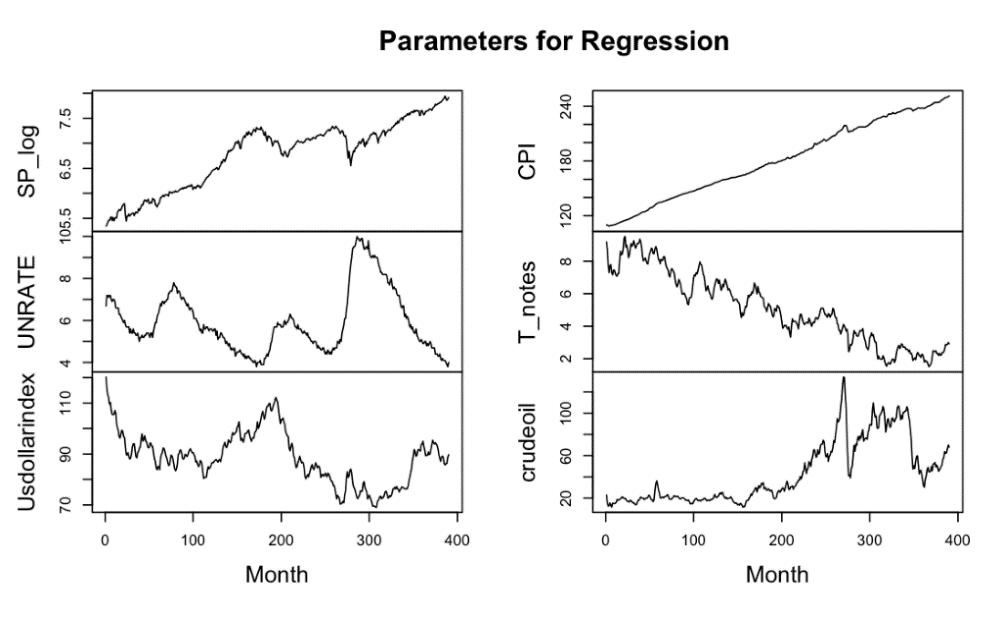
****

Figure 3. Graphs of Parameters from Jan 1986-Jun 2018

However, part of the result deviates from the assumptions we had regarding the relationships of CPI and the US dollar index with the index of the S&P 500. In our assumption these factors are positively-correlated, however, our equation shows a negative correlation. This is likely due to the long-term nature of our data, the relationships between our predictors and the index have changed over time, so they could not produce a reliable long-term relationship, leading to this seemingly unexpected result. Also, it could be the case that there are better predictors that we didn’t take into consideration that have effects on S&P 500 index as well.

1. ETS:

The ets function and a manual search through all ETS models return (A, A, N) as the model with the lowest AICc. Checking the residuals, we can conclude that the residual is white noise based on the p-value for the Ljung-Box test (0.14) and the ACF (Figure A.2).

The equation of ETS is the (A, A, N) with parameters:

1. ARIMA:

The ARIMA model we choose is (1, 1, 1) with drift. We found this result both by using the auto.arima() function and by manually searching through all Arima models with d=1 for p,q=0,1,2,3,4,5. This model has the lowest AICc, and looking at the residuals, we can see both through the ACF (Figure A.3) and p-value of the Ljung-Box test,( p=.6164) that the residuals are uncorrelated and resemble white noise, so we can move forward with this model.

The equation for ARIMA model is:

1. GARCH:

The GARCH (Generalised Autoregressive Conditional Heteroskedasticity) model takes into account the non-constant variance of time series. Because the squared residuals of our ARIMA model show autocorrelation, we should add a GARCH effect to our ARIMA model (Figure A.7). We will use the log return to build an ARMA(1,1) + GARCH(1,1) model. The model residual is white noise (Figure A.4). However, it does not solve the ARCH effect of our residuals squared, so the model is invalid. (Figure A.8)

The equation for our model:

1. Neural network:

For neural network, there is no statistical meaning to the model, so we choose the model based on the lowest AIC. Checking models with different numbers of nodes in the initial and hidden layer lead us to adopt the model nnetTs (8,6). The model residual is white noise (Figure A.5). Because this model relies on randomness, with a different seed we could obtain different results both for the AIC and for the actual prediction. For this reason, the neural network is an unreliable model.

1. Threshold AR (TAR, SETAR):

For financial time series, the recession periods matter a lot. In our data, we have two noticeable downtrends around 2004 and 2008.  In this case, we try to implement a Threshold AR (2) model with a threshold log return of –0.015 to separate recession and normal periods. The model indicates that it has a white noise residual based on the ACF of residual plot (Figure A.6) and a p-value of .543 in the Ljung-Box test.

The equation for our model:

for

for

**Forecast Analysis and Model Comparison:**

Forecast errors and AIC of the different methods are reported in Table 1:

Table 1. Forecast error comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Linear Regression | ETS | ARIMA | Neural Network | GARCH | **SETAR** |
| AIC | -2330 | -81 | -1254 | -2242 | -1321 | -2242 |
| MSE | 0.00255 | 0.00156 | 0.00165 | 0.00736 | 0.00142 | 0.00120 |

The reason ETS has significantly different AIC from Linear Regression, ARIMA, GARCH, and SETAR is because ETS uses the log price, and the other models require stationarity, so they use the log return. It is difficult to compare AIC from one model to another, but we can see that SETAR outperforms all other models, in terms of MSE. Also we can see in the Figure 4 below that the forecasted values of SETAR more closely align to the log of the index price for the S&P 500 during the test data period (July 2017-June 2018):

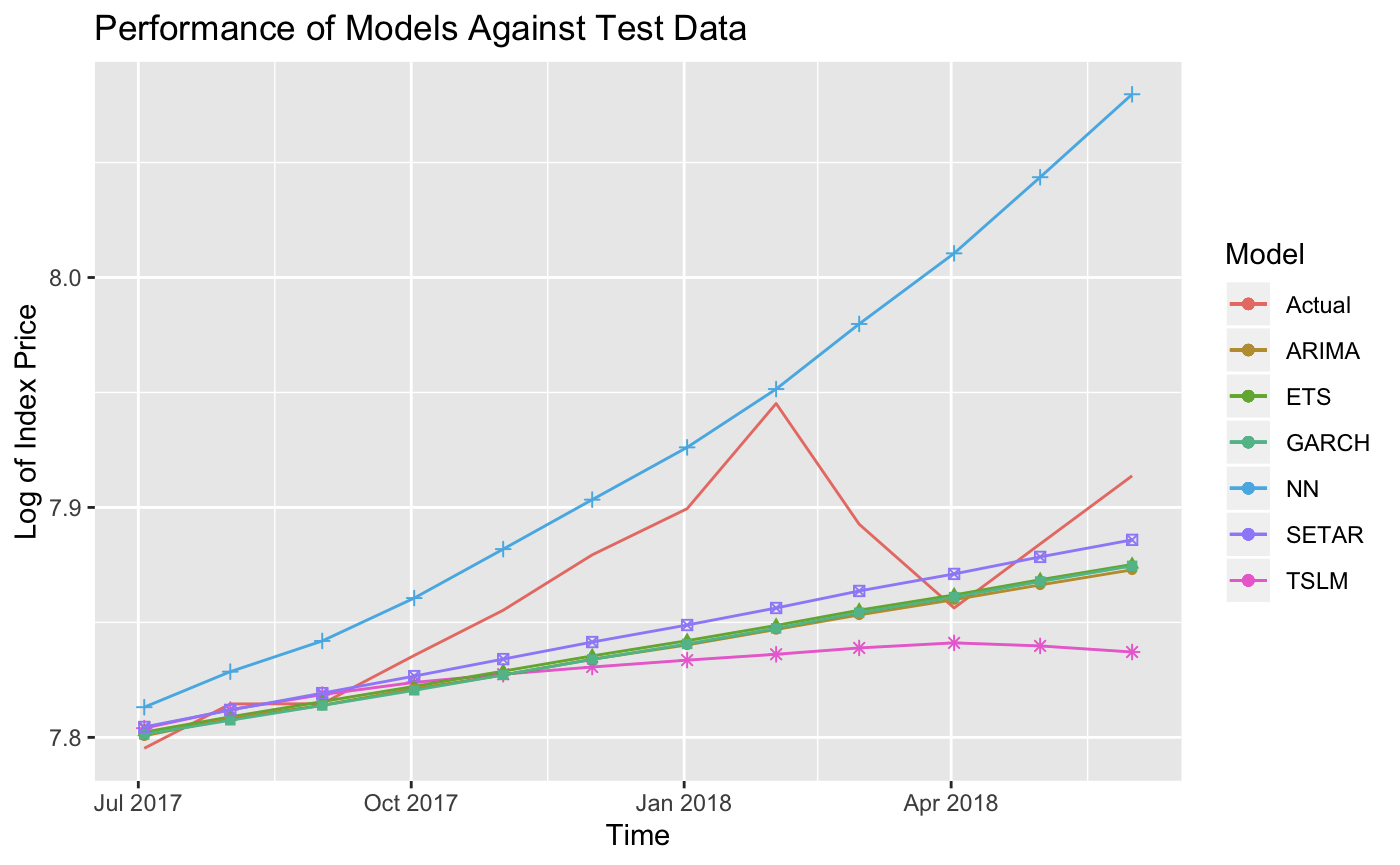
****

Figure 4: Test Data and Model Predictions for Jul 2017 through Jun 2018

**Conclusion:**

We first chose the best model of each method based on AIC, these AIC values are shown in Table 1. We checked each of the residuals for these models and saw that they resembled white noise, and, based on the p-values for the Ljung-Box test, we saw that the residuals were not autocorrelated for any of the selected models. Next, we compared the forecast error based on the MSE from each ideal model to pick the best forecasting model.

Among these, SETAR outperformed all other models. This makes sense, since it can divide the data into two regimes, one which reflects when the market is doing well, and one which reflects when the market is doing poorly.

Table 2. Cumulative Log Return of SETAR and Test data

|  |  |  |
| --- | --- | --- |
| Percent Cumulative Log Return | Jun 2017-Jun 2018 | Jun 2017-Dec 2019 |
| SETAR | 9.02 | 22.35 |
| Test Data | 11.81 | 24.84 |

Using SETAR to make predictions, we can predict the cumulative log return in the time period from June 2017 to June 2018, seen in Table 2 above. Table 2 also shows the out of sample final prediction from June 2017 to December 2019, we can see that SETAR is very consistent in terms of predicting the trend in long term as well. The MSE for the model does increase slightly, from .0012 to .00147, with a majority of this error being due to a sharp dip in performance in the S&P 500 in January 2019. This behavior can be observed in Figure 5. Forecasting for the next Quarter, from December 2019-March 2020, our model predicts a percent cumulative log return of 2.22%.

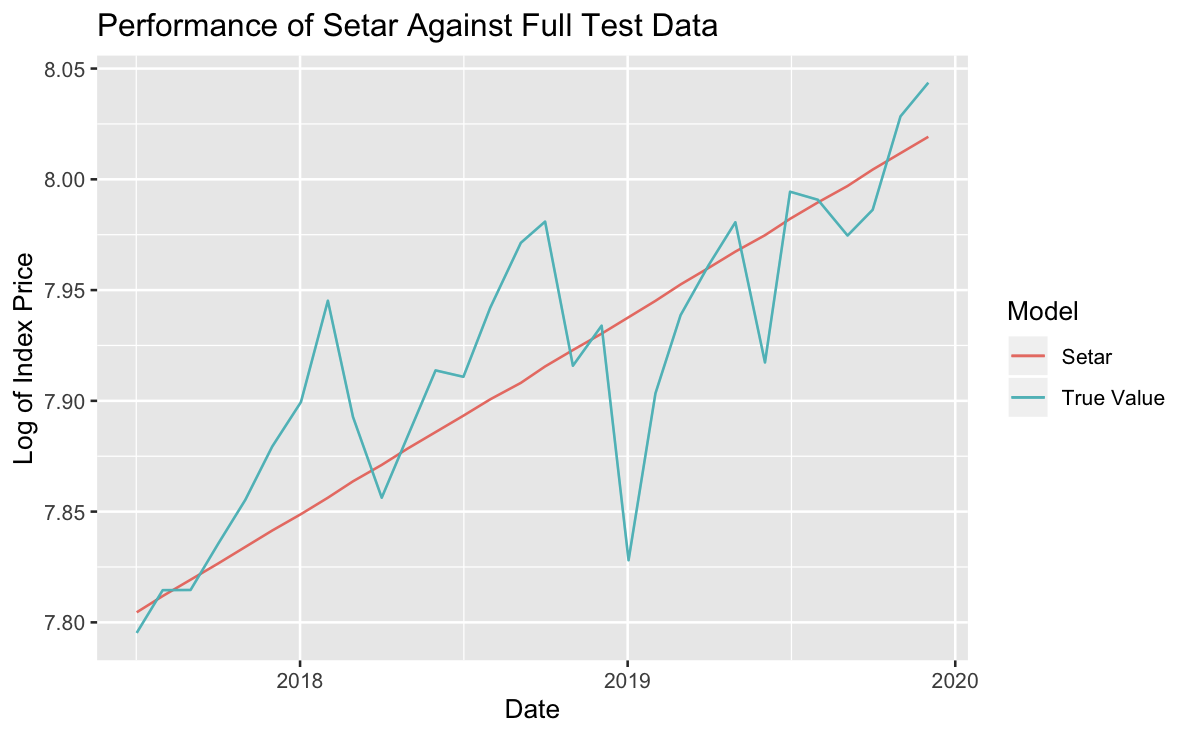


Figure 5. Prediction with SETAR against test data and Out of Sample Test Data

While the true S&P 500 monthly closing index is more volatile in this time period, the general trend is closely matched by the SETAR model. In this regard, we can provide our recommendation: despite claims that a recession is on the horizon, it is still a good time to enter the market for investors who have a long-term investment strategy. However, the potential unpredictable volatility may disappoint some short-term investors who want to make some quick money by doing some simple arbitrage.

For people looking to invest, this analysis shows that the economy in the US is growing at a healthy rate, and that investing in an index that resembles the S&P 500 would result in a good growth of returns based on historical data and our forecasting.

References:

<https://towardsdatascience.com/beating-the-s-p500-using-machine-learning-c5d2f5a19211>

<https://www.quantstart.com/articles/ARIMA-GARCH-Trading-Strategy-on-the-SP500-Stock-Market-Index-Using-R>

<https://www.quantstart.com/articles/Generalised-Autoregressive-Conditional-Heteroskedasticity-GARCH-p-q-Models-for-Time-Series-Analysis>

Appendix

Residuals Figures:

Figure A.1

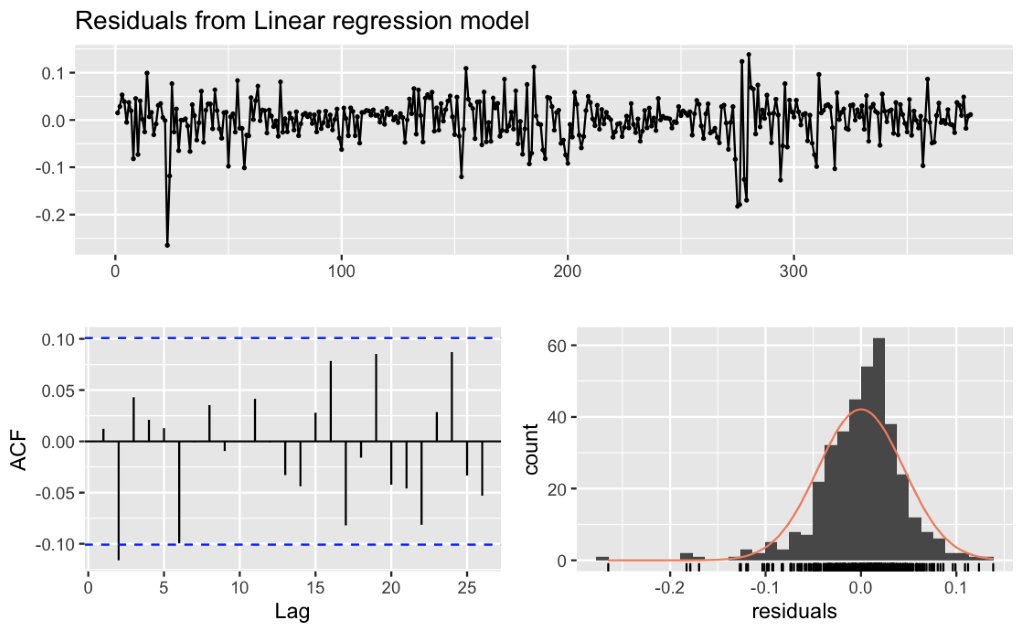
****

Figure A.2

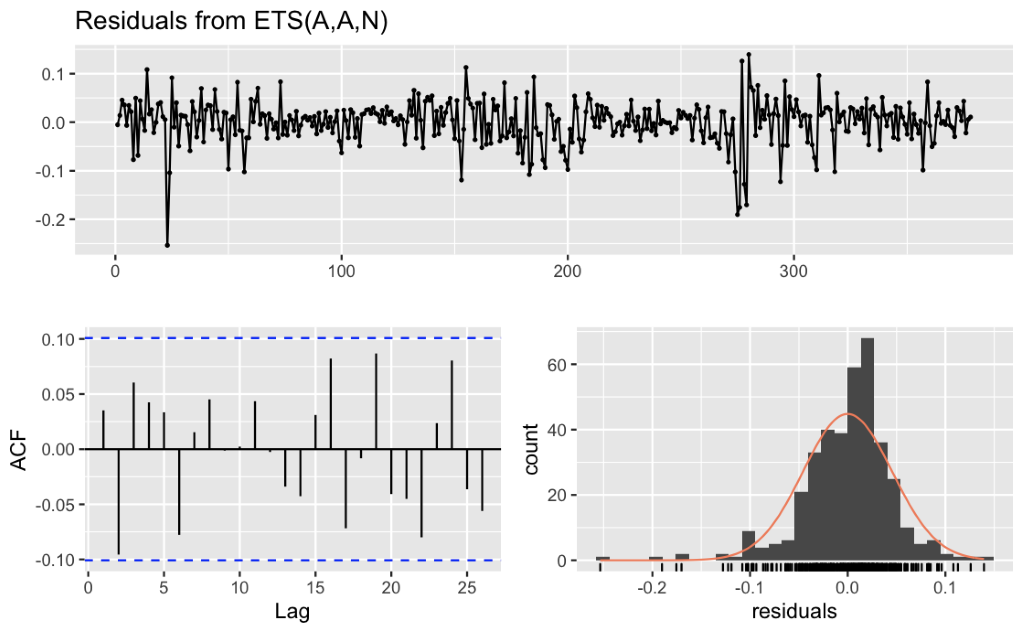
****

Figure A.3

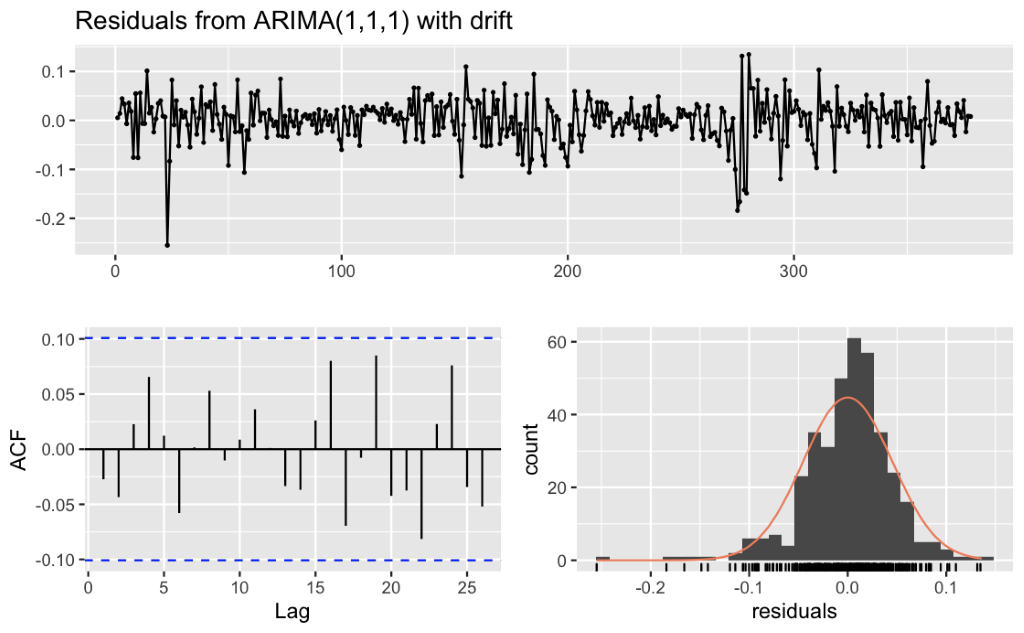
****

Figure A.4 GARCH Residuals

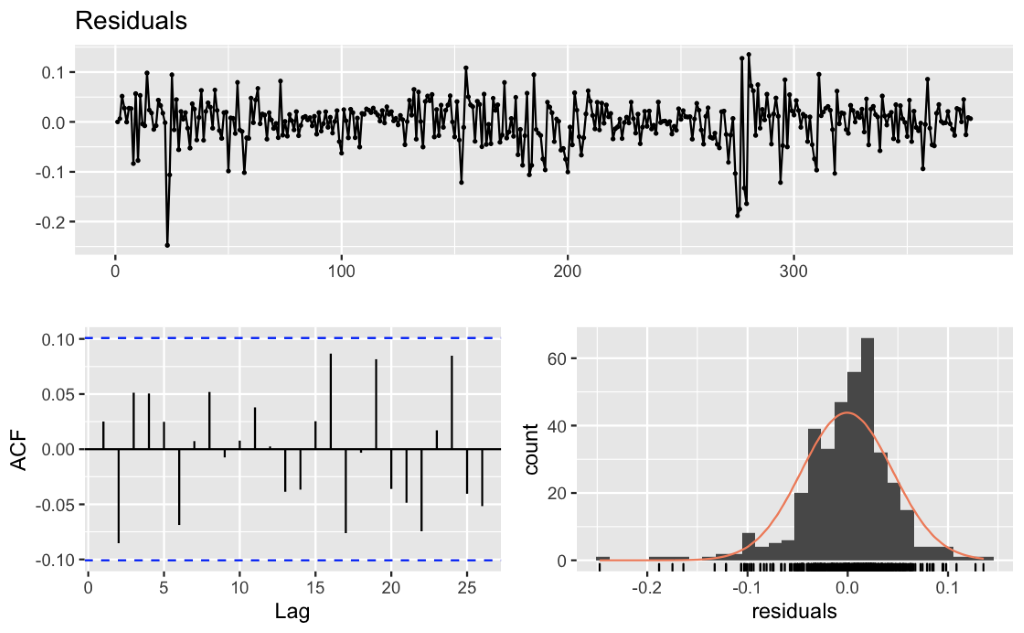
****

Figure A.5 Neural Network Residuals

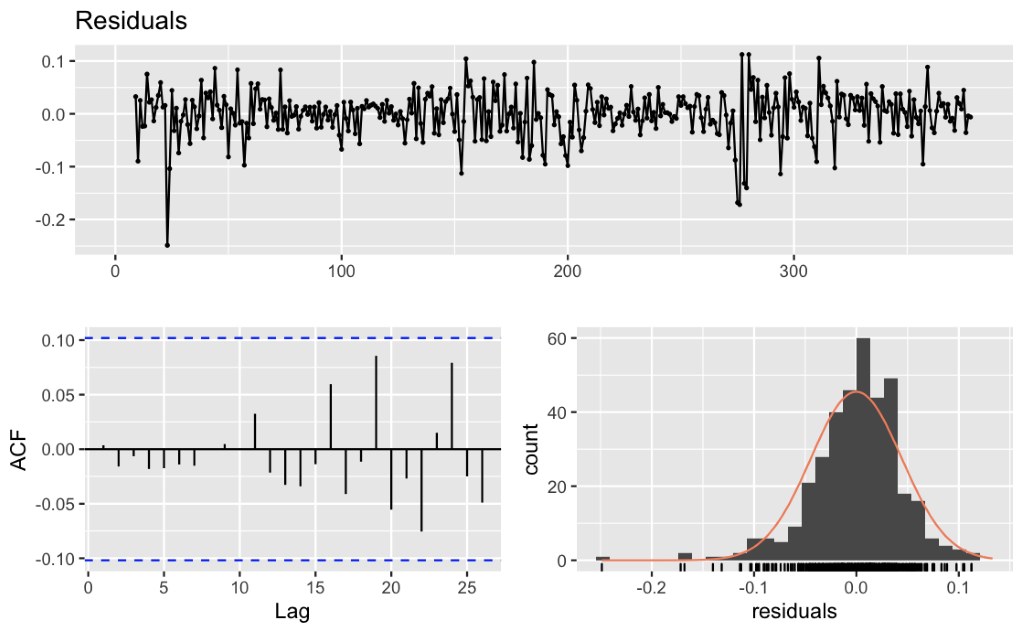
****

Figure A.6 SETAR Residuals

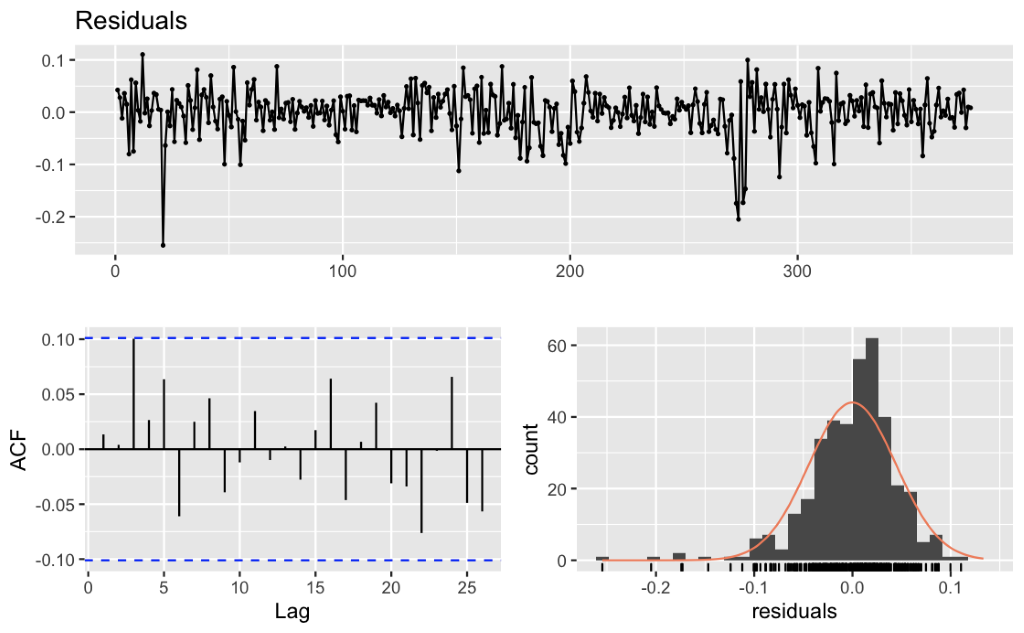
****

Figure A.7 ACF of ARIMA Residuals ^2 showing GARCH effect

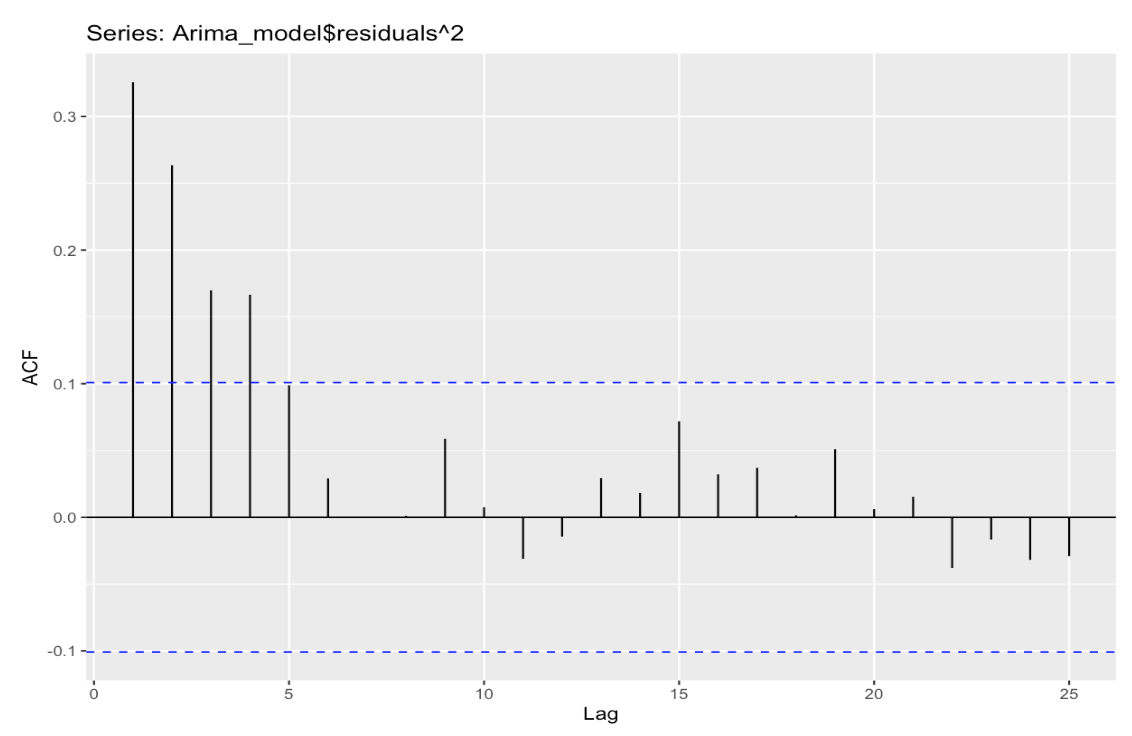
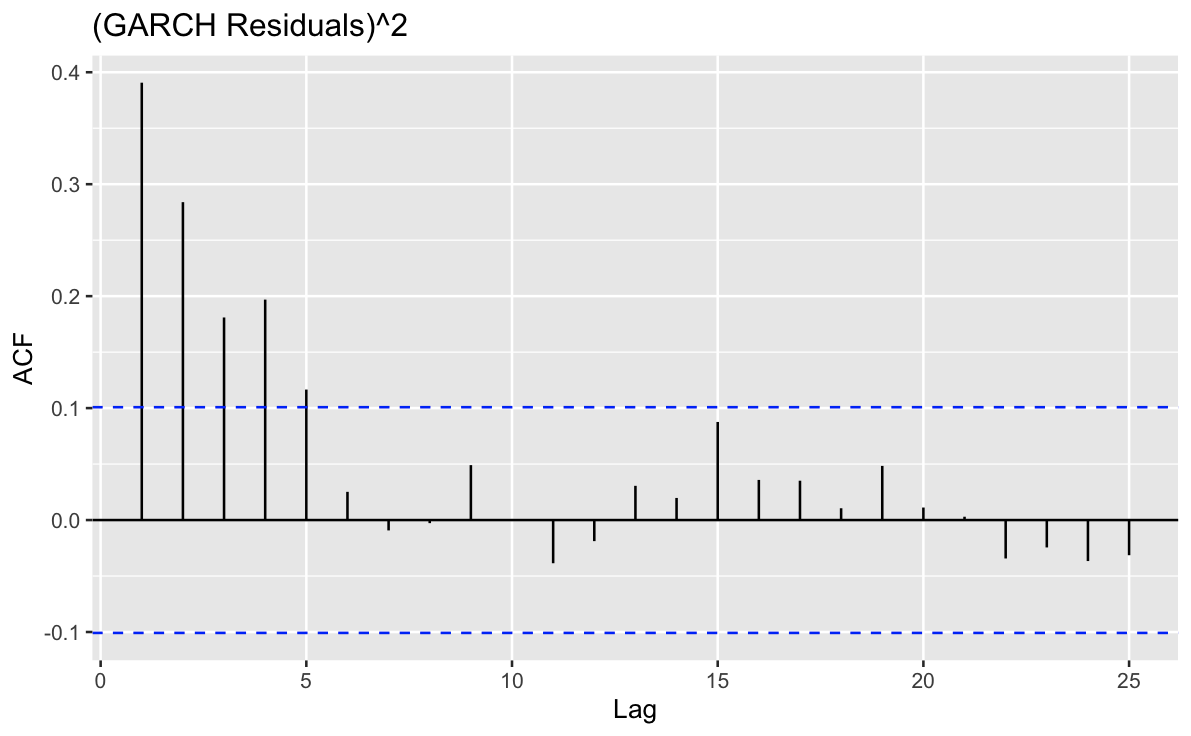


Figure A.8 Residuals squared of GARCH model



---

title: "Time Series Analysis Project"

author: "Steven Dougherty, Bao Khanh Cu, Zherui Liang, Jiyang Huang"

date: "12/8/2019"

output: html\_document

---

### Libraries in use for this project

```{r}

library(fpp2)

library(xts)

library(tseries)

library(fGarch)

library(rugarch)

library(forecast)

library(tsDyn)

this.dir = dirname(rstudioapi::getActiveDocumentContext()$path)

setwd(this.dir)

```

## Setup:

# Import Data set and transform into a time series

```{r}

SPclose<-read.csv("spx.csv",header = TRUE,sep = ",")

SPclose[,1]<-as.Date(SPclose[,1], "%d-%b-%y")

SP<- xts(SPclose[,-1], order.by=SPclose[,1])

autoplot(SP)+xlab("Date")+ylab("Index")+ggtitle("S&P 500 Index")

## Not the cleanest method for getting the first plot point from each month, but is sufficient to account for all non-trading days early in the month.

SPmonthly<-c()

for(i in 1986:2017){

for(j in 1:12){

FirstDay<-FALSE

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-01",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-02",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-03",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-04",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-05",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

SPmonthly<-c(SPmonthly,minSPmonth)

}

}

i=2018

for(j in 1:6){

FirstDay<-FALSE

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-01",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-02",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-03",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-04",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

if(FirstDay==FALSE){

SPmonthlymin<-paste(as.character(i),"-",as.character(j),"-05",sep = "")

minSPmonth<-which(index(SP)==SPmonthlymin)

if(any(index(SP)==SPmonthlymin)){FirstDay<-TRUE}

}

SPmonthly<-c(SPmonthly,minSPmonth)

}

SPmonthly<-SP[SPmonthly,]

```

## Visualization of data:

# Plotting the ACF, PACF, and plot of our timeseries

```{r}

autoplot(SPmonthly)+ylab("S&P 500 Index Price")+xlab("Date")

ggAcf(SPmonthly)+ylab("S&P 500 Index Price")

ggPacf(SPmonthly)+ylab("S&P 500 Index Price")

SP\_log<-log(SPmonthly)

autoplot(SP\_log)+ylab("S&P 500 Log Index Price")+xlab("Date")

ggAcf(SP\_log)+ylab("S&P 500 Index Price")

ggPacf(SP\_log)+ylab("S&P 500 Index Price")

SP\_logrt<-diff(log(SPmonthly))

SP\_logrt[1]<-0

adf.test(SP\_logrt)

autoplot(SP\_logrt)+ylab("S&P 500 Log Return")+xlab("Date")

ggAcf(SP\_logrt)+ylab("S&P 500 Log Return")

ggPacf(SP\_logrt)+ylab("S&P 500 Log Return")

```

## Transformation:

# This is plotted along with its associated ACF and PACF below. The Augmented Dickey Fuller test returns a p-value of .01, indicating that this. To make the data stationary, which we require for some of our methods, we will look at the log return.

```{r}

SP\_log<-as.ts(SP\_log)

adf.test(SP\_log)

SP\_logrt<-as.ts(SP\_logrt)

# This data is not stationary, but a single difference will show stationarity

adf.test(SP\_logrt)

```

##Splitting Data

```{r}

SP\_train\_log<-window(SP\_log, end = 378)

SP\_test\_log<-window(SP\_log, start = 379)

```

## Time Series Linear Regression (ON LOG RETURN)

```{r}

#data frame for tslm (log return)

#creating a time series that is indexed with the same dates as SPmonthly

SP\_log\_ret<-xts(rep(0,390),order.by =index(SPmonthly))

SP\_log\_ret[1]<-0

S<-SPmonthly%>%log()%>%diff()

SP\_log\_ret[2:390]<-na.omit(S)

## Importing our Predictors

X<-read.csv('predictors.csv') #predictors

X1<- xts(X[,-1], order.by=index(SPmonthly))

train\_log\_ret<-window(SP\_log\_ret,end=c("2017-06-01"))

test\_log\_ret<-window(SP\_log\_ret, start=c("2017-07-03"))

SP\_log\_ret\_train<-ts(data.frame(SP\_INDEX=train\_log\_ret,X[1:378,]))

SP\_log\_ret\_test<-ts(data.frame(SP\_INDEX=test\_log\_ret,X[379:390,]))

SP\_log\_params<-ts(data.frame(SP\_log=SP\_log,X))

plot(SP\_log\_params[,c(1,3,4,5,6,7)],main="Parameters for Regression", xlab="Month" )

```

```{r}

#model comparison

tslm(SP\_INDEX ~ UNRATE+Usdollarindex+CPI+trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ UNRATE+Usdollarindex+crudeoil+trend,data =SP\_log\_ret\_train )%>%CV()# best

tslm(SP\_INDEX ~ UNRATE+T\_notes +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ CPI+T\_notes +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ Usdollarindex+CPI+T\_notes +trend,data =SP\_log\_ret\_train )%>%CV()#best for logr

tslm(SP\_INDEX ~ Usdollarindex+CPI +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ Usdollarindex+T\_notes +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ UNRATE+T\_notes +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ UNRATE+CPI +crudeoil +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(SP\_INDEX ~ UNRATE+CPI+T\_notes +trend,data =SP\_log\_ret\_train )%>%CV()

tslm(formula = SP\_INDEX ~ UNRATE + Usdollarindex + CPI + T\_notes +

crudeoil + trend, data = SP\_log\_ret\_train)%>%CV() # full model

```

## Checking Residuals for best model

```{r}

fit1<-tslm(SP\_INDEX ~ Usdollarindex+CPI+T\_notes +trend,data =SP\_log\_ret\_train )

summary(fit1)

checkresiduals(fit1)

CVfit1<-CV(fit1)

plot(fit1$fitted.values,SP\_log\_ret\_train[,1],type='p',pch=16,cex=.2,main='Unemployment Rate + USD Index + T-note + CPI + trend') + abline(a=0,b=1,lwd=2,col='red')

```

```{r}

#forecast error of log return

newdata<-data.frame(DATE=SP\_log\_ret\_test[,2],UNRATE=SP\_log\_ret\_test[,3],Usdollarindex=SP\_log\_ret\_test[,4],CPI=SP\_log\_ret\_test[,5],T\_notes=SP\_log\_ret\_test[,6],crudeoil=SP\_log\_ret\_test[,7])

fit1\_forecast<-forecast(fit1, newdata = newdata)

```

```{r}

#Back transformation

tslmForecast<-rep(0,length(fit1\_forecast$mean))

final\_value<- SP\_log[378]

tslmForecast[1] <- final\_value+fit1\_forecast$mean[1]

for (i in 2:length(fit1\_forecast$mean)) {

tslmForecast[i] <- fit1\_forecast$mean[i]+ tslmForecast[i-1]

}

#forecast error of transformed data

tslmForecast<-as.ts(ts(tslmForecast,start = 379, end = 390))

```

# Application of ETS, using both ets function and manually with a for loop to find lowest AICc

```{r}

ETS\_AICc\_matrix<-matrix(10^8,nrow=2,ncol=5)

ETS\_AICc\_matrix[1,1]<-ets(SP\_train\_log,model = "ANN")$aicc

ETS\_AICc\_matrix[1,2]<-ets(SP\_train\_log,model = "AAN",damped = FALSE)$aicc

ETS\_AICc\_matrix[1,4]<-ets(SP\_train\_log,model = "AAN",damped = TRUE)$aicc

ETS\_AICc\_matrix[2,1]<-ets(SP\_train\_log,model = "MNN")$aicc

ETS\_AICc\_matrix[2,2]<-ets(SP\_train\_log,model = "MAN",damped = FALSE)$aicc

ETS\_AICc\_matrix[2,3]<-ets(SP\_train\_log,model = "MMN",damped = FALSE)$aicc

ETS\_AICc\_matrix[2,4]<-ets(SP\_train\_log,model = "MAN",damped = TRUE)$aicc

ETS\_AICc\_matrix[2,5]<-ets(SP\_train\_log,model = "MMN",damped = TRUE)$aicc

min(ETS\_AICc\_matrix)

ETS\_AICc\_matrix

ETS\_AICc\_matrix[1,2]

##AAN same choice manually as

ETS\_model<-ets(SP\_train\_log)

ETS\_model

```

## Application of ARIMA, doing a manual gridsearch for the optimal ARIMA(p,d,q)

```{r}

d<-ndiffs(SP\_train\_log)

ARIMA\_AICc\_matrix<-matrix(0,nrow=6,ncol=6)

for(i in 1:6){

for(j in 1:6){

ARIMA\_AICc\_matrix[i,j]<-Arima(SP\_train\_log, order = c(i-1,d,j-1),include.drift = TRUE)$aicc

}

}

ARIMA\_AICc\_matrix

min(ARIMA\_AICc\_matrix)

Arima\_model<-auto.arima(SP\_train\_log, stepwise = FALSE, approximation = FALSE)

Arima\_model

Arima\_model$aicc

## ARIMA(1,1,1) is the best model (with draft)

Arima(SP\_train\_log, order = c(1,1,1),include.drift = TRUE)$aicc

```

##Check residulas of models with lowest AICc

# ARIMA

```{r}

ETS\_model<-ets(SP\_train\_log) ##AAN

Arima\_model<-auto.arima(SP\_train\_log, stepwise = FALSE, approximation = FALSE) ##(1,1,1) with drift

### Checking Residuals

checkresiduals(Arima\_model)

checkresiduals(ETS\_model)

```

### GARCH Model

```{r}

##ARMA(1,1)+GARCH(1,1)

ggAcf((Arima\_model$residuals)^2)+ggtitle("(ARIMA residuals)^2")

fitgarch<-garchFit(~arma(1,1)+garch(1,1),train\_log\_ret,trace=F)

summary(fitgarch)

garch.predict<- predict(fitgarch,12)

garch.predict$meanForecast

## Residuals of GARCH

checkresiduals(fitgarch@residuals)

ggAcf((fitgarch@residuals)^2)+ggtitle("(GARCH Residuals)^2")

Box.test((fitgarch@residuals)^2, type = "Ljung")

## Backtransforming GARCH

final\_value<- SP\_train\_log[length(SP\_train\_log)]

garch.predict$meanForecast[1] <- final\_value+garch.predict$meanForecast[1]

for (i in 2:12) {

garch.predict$meanForecast[i] <- garch.predict$meanForecast[i]+ garch.predict$meanForecast[i-1]

}

```

### Neural Network alternative

```{r}

set.seed(1)

nnet<- nnetTs(SP\_train\_log, m=8, size=6)

AIC(nnet)

nnet.predict<-predict(nnet,n.ahead=12)

checkresiduals(nnet)

Box.test(nnet$residuals,lag=10,type = "Ljung")

```

###SETAR model

```{r}

fit.setar<-setar(train\_log\_ret,m=2,thDelay = 1)

fit.setar

pred.setar <- predict(fit.setar,n.ahead=12)

pred.setar[1]<-pred.setar[1]+SP\_train\_log[378]

for (i in 2:12){

pred.setar[i]<-pred.setar[i]+pred.setar[i-1]

}

pred.setar

checkresiduals(fit.setar$residuals)

Box.test(fit.setar$residuals,lag=10,type='Ljung')

```

## Checking AIC

```{r}

# TSLM

AIC\_TSLM<-CVfit1[3]

AIC\_TSLM

#ETS

AIC\_ETS<-ETS\_model$aic

AIC\_ETS

#ARIMA

AIC\_Arima<-Arima\_model$aic

AIC\_Arima

#GARCH

AIC\_Garch<-fitgarch@fit$ics[1]\*length(SP\_train\_log)

AIC\_Garch

# #Neural

AIC\_NN<-AIC(nnet)

AIC\_NN

## SETAR

AIC\_SETAR<-AIC(fit.setar)

AIC\_SETAR

```

### Checking Fit For All Models

```{r}

# TSLM

# forecast::accuracy(tslmForecast,SP\_test\_log) %>% round(4)

MSE\_TSLM<-(1/12)\*sum((tslmForecast-SP\_test\_log)^2)

MSE\_TSLM

#ETS

# ETS\_model %>% forecast(h=12) %>%forecast::accuracy(SP\_test\_log) %>% round(4)

fETS<-forecast(ETS\_model,h=12)

MSE\_ETS<-(1/12)\*sum((fETS$mean-SP\_test\_log)^2)

MSE\_ETS

#ARIMA

# Arima\_model %>% forecast(h=12) %>% forecast::accuracy(SP\_test\_log) %>% round(4)

fArima<-forecast(Arima\_model,h=12)

MSE\_Arima<-(1/12)\*sum((fArima$mean-SP\_test\_log)^2)

MSE\_Arima

#GARCH

# forecast::accuracy(ts(SP\_test\_log, start=379, end=390),plot.garch)%>%round(4)

MSE\_Garch<-(1/12)\*sum((plot.garch-SP\_test\_log)^2)

MSE\_Garch

# #Neural

# forecast::accuracy(neural.forecast, SP\_test\_log)%>% round(4)

MSE\_NN<-(1/12)\*sum((nnet.predict-SP\_test\_log)^2)

MSE\_NN

## SETAR

MSE\_SETAR<-(1/12)\*sum((pred.setar-SP\_test\_log)^2)

MSE\_SETAR

```

## SETAR has the lowest test errors of the three models presented here.

### Graph of Test Data with all models

### Graph of Test Data with all models

```{r}

dat.predict<-data.frame(Time<-index(SPmonthly[379:390]),actual<-SP\_test\_log,TSLM<-tslmForecast,ETS<-fETS$mean,ARIMA<-fArima$mean,NN<-nnet.predict,GARCH<-garch.predict$meanForecast,SETAR<-pred.setar)

ggplot(dat.predict, mapping=aes(x=Time))+geom\_line(mapping=aes(y=actual,col="Actual"))+geom\_line(mapping=aes(y=TSLM,col="TSLM"))+geom\_point(mapping=aes(y=TSLM,col="TSLM",shape="TSLM"))+geom\_line(mapping=aes(y=ETS,col="ETS"))+geom\_point(mapping=aes(y=ETS,col="ETS",shape="ETS"))+geom\_line(mapping=aes(y=ARIMA,col="ARIMA"))+geom\_point(mapping=aes(y=ARIMA,col="ARIMA",shape="ARIMA"))+geom\_line(mapping=aes(y=NN,col="NN"))+geom\_point(mapping=aes(y=NN,col="NN",shape="NN"))+geom\_line(mapping=aes(y=GARCH,col="GARCH"))+geom\_point(mapping=aes(y=GARCH,col="GARCH",shape="GARCH"))+geom\_line(mapping=aes(y=SETAR,col="SETAR"))+geom\_point(mapping=aes(y=SETAR,col="SETAR",shape="SETAR"))+ylab("Log of Index Price")+ggtitle("Performance of Models Against Test Data")+scale\_color\_discrete(name="Model")+guides(shape=F)

```

## Modeling Cummulative Log Return with SETAR

```{r}

setar\_cummulative\_return\_pred<-pred.setar[12]-SP\_train\_log[length(SP\_train\_log)]

setar\_cummulative\_return\_pred

SP\_test\_log[12]-SP\_train\_log[length(SP\_train\_log)]

```

## Forecasting to Dec 2019 with SETAR

```{r}

pred.further.setar <- predict(fit.setar,n.ahead=30)

pred.further.setar[1]<-pred.further.setar[1]+SP\_train\_log[378]

for (i in 2:30){

pred.further.setar[i]<-pred.further.setar[i]+pred.further.setar[i-1]

}

## SETAR Cummulative Log Return

pred.further.setar[30]-SP\_train\_log[length(SP\_train\_log)]

```

## Forecasting further 1 Quarter

```{r}

future.forecast <- predict(fit.setar,n.ahead=33)

future.forecast[1]<-future.forecast[1]+SP\_train\_log[378]

for (i in 2:33){

future.forecast[i]<-future.forecast[i]+future.forecast[i-1]

}

## Percent Cumulative Log Return for Dec 2019- Mar 2020

(future.forecast[33]-future.forecast[30])\*100

```

##Out of Sample Forecast Data

```{r}

# Out of Sample Test Data for July 2018-December 2019

Future\_Test\_Data<-rev(c(3113.87,3066.91,2940.25,2906.27,2953.56,2964.33,2744.45,2923.73,2867.19,2803.69,2706.53,2510.03,2790.37,2740.37,2924.59,2896.72,2813.36,2726.71))

Future\_Test\_Dates<-as.Date(c("2018-07-02","2018-08-01","2018-09-04","2018-10-01","2018-11-01","2018-12-03","2019-01-02","2019-02-01","2019-03-01","2019-04-01","2019-05-01","2019-06-03","2019-07-01","2019-08-01","2019-09-03","2019-10-01","2019-11-01","2019-12-02"))

SP\_future<-xts(Future\_Test\_Data, order.by = Future\_Test\_Dates)

SP500xts<-as.xts(c(SPmonthly,SP\_future))

SP500ts<-as.ts(c(SPmonthly,SP\_future))

SPtrain500<-window(SP500xts,end="2017-06-06")

SPtest500<-window(SP500xts,start="2017-06-06",end="2018-06-06")

SPfuture500<-window(SP500xts,start="2018-06-06")

## SP500 Cummulative Log Return June 2017-December 2019

log(SP\_future[18])-log(SPtest[1])

```

```{r}

## Plot of full SP 500 data

autoplot(cbind(SPtrain500,SPtest500,SPfuture500), facets = NULL)+ggtitle("S&P 500 Index")+ylab("Closing Price") + theme(legend.position = "none")+xlab("Date")

SPfulltest<-log(SP500ts[379:408])

### MSE of longer term SETAR

MSE\_SETAR\_further<-(1/30)\*sum((pred.further.setar-SPfulltest)^2)

MSE\_SETAR\_further

SP500dat<-data.frame(x=index(SP500xts[379:408]),y<-pred.further.setar,z<-log(SP500ts[379:408]))

ggplot(SP500dat, aes(x=x))+geom\_line(mapping=aes(y=y,col="Setar"))+ geom\_line(mapping=aes(y=z,col="True Value"))+ylab("Log of Index Price")+ggtitle("Performance of Setar Against Full Test Data") + scale\_color\_discrete(name="Model")+xlab("Date")

## SETAR Forecast Through Next Quarter

autoplot(log(window(SP500ts,start=379)))+autolayer(future.forecast)+theme(legend.position = "none") + ylab("Log of Index Price") + ggtitle("SETAR Forecast Until March 2020")

```