Econ 144 Project 3

Shannan Liu, Austin Pham, Zach Wrubel

09 March, 2022

Contents

I. Introduction

In this project, we will forecast the returns of Bitcoin with several models including ARIMA, ETS, Holt-Winters, NNETAR, Prophet, and a forecast combination.

To build these forecasts, we'll also be using data from other cryptocurrencies including Ethereum, Tether, and XRP.

Brief Background on Data:

Bitcoin (BTC): a digital currency that was released on Jan 3, 2009. It is the most well known cryptocurrency in the world, and it popularized the concept of blockchain technology. While some countries have adopted it as legal tender, others such as China, have banned the currency from their country. Although Bitcoin can be used to purchase goods and services, many view it as an investment vehicle because of its meteoric rise in value. Some investors also believe it could replace gold as a safe-haven investment during periods of high inflation or economic uncertainty.

Ethereum (ETH): another digital currency. It was released on July 30, 2015. It is the next most popular cryptocurrency. It introduced smart contract functionality, which Bitcoin does not have.

Tether (USDT): unlike Bitcoin and Ethereum, Tether is a stable coin, meaning its value is pegged to the US dollar at a 1:1 ratio. Thus, its value is more stable than other cryptocurrecies.

XRP: is another popular cryptocurrency created by Ripple and mainly used for the settlement of cross-border remittances.

Source of Data

All of the data was sourced from https://coinmarketcap.com via their web api. We will be analyzing daily observations of the data from December 20, 2015 to March 3, 2022. Company descriptions were adapted from descriptions provided at https://bloomberg.com.

```
# import data
cl <- crypto_list(only_active=TRUE) %>%
  filter(first_historical_data<="2015-12-31",
                last_historical_data>="2022-03-03")
cl_final <- rbind(filter(cl,cl$slug=="bitcoin"),
                  filter(cl,cl$slug=="ethereum"),
                  filter(cl,cl$slug=="xrp"),
                  filter(cl,cl$slug=="tether"))
coins <- crypto_history(coin_list = cl_final,
                        start_date = "20151231", interval="daily")
```

```
## > Scraping historical crypto data
```

```
##
```

```
## > Processing historical crypto data
```

```
##
```

```r
relevant_vars <- c("timestamp","close","volume",
                   "open","high","low","market_cap")

# create separate dfs for each variable
btc <- filter(coins[relevant_vars],coins$slug=="bitcoin")
btc_monthly <- filter(coins[relevant_vars],coins$slug=="bitcoin")
eth <- filter(coins[relevant_vars],coins$slug=="ethereum")
eth_monthly <- filter(coins[relevant_vars],coins$slug=="ethereum")
usdt <- filter(coins[relevant_vars],coins$slug=="tether")
usdt_monthly <- filter(coins[relevant_vars],coins$slug=="tether")
xrp <- filter(coins[relevant_vars],coins$slug=="xrp")
xrp_monthly <- filter(coins[relevant_vars],coins$slug=="xrp")

# subset by monthly
btc_monthly$timestamp <- floor_date(btc_monthly$timestamp, "monthly")
eth_monthly$timestamp <- floor_date(eth_monthly$timestamp, "monthly")
usdt_monthly$timestamp <- floor_date(usdt_monthly$timestamp, "monthly")
xrp_monthly$timestamp <- floor_date(xrp_monthly$timestamp, "monthly")
btc_monthly <- btc_monthly %>%
  group_by(timestamp) %>% summarize(close = mean(close))
eth_monthly <- eth_monthly %>%
  group_by(timestamp) %>% summarize(close = mean(close))
usdt_monthly <- usdt_monthly %>%
  group_by(timestamp) %>% summarize(close = mean(close))
xrp_monthly <- xrp_monthly %>%
  group_by(timestamp) %>% summarize(close = mean(close))

# convert POSIXct to date time
btc$timestamp <- as.Date(btc$timestamp)
eth$timestamp <- as.Date(eth$timestamp)
usdt$timestamp <- as.Date(usdt$timestamp)
xrp$timestamp <- as.Date(xrp$timestamp)
btc_monthly$timestamp <- as.Date(btc_monthly$timestamp)
eth_monthly$timestamp <- as.Date(eth_monthly$timestamp)
usdt_monthly$timestamp <- as.Date(usdt_monthly$timestamp)
xrp_monthly$timestamp <- as.Date(xrp_monthly$timestamp)

# create returns column
btc$returns <- c(NA,diff(log(btc$close)))
eth$returns <- c(NA,diff(log(eth$close)))
usdt$returns <- c(NA,diff(log(usdt$close)))
xrp$returns <- c(NA,diff(log(xrp$close)))
btc_monthly$returns <- c(NA,diff(log(btc_monthly$close)))
eth_monthly$returns <- c(NA,diff(log(eth_monthly$close)))
usdt_monthly$returns <- c(NA,diff(log(usdt_monthly$close)))
xrp_monthly$returns <- c(NA,diff(log(xrp_monthly$close)))

# remove NAs
btc <- na.omit(btc)
eth <- na.omit(eth)
usdt <- na.omit(usdt)
xrp <- na.omit(xrp)
btc_monthly <- na.omit(btc_monthly)
```

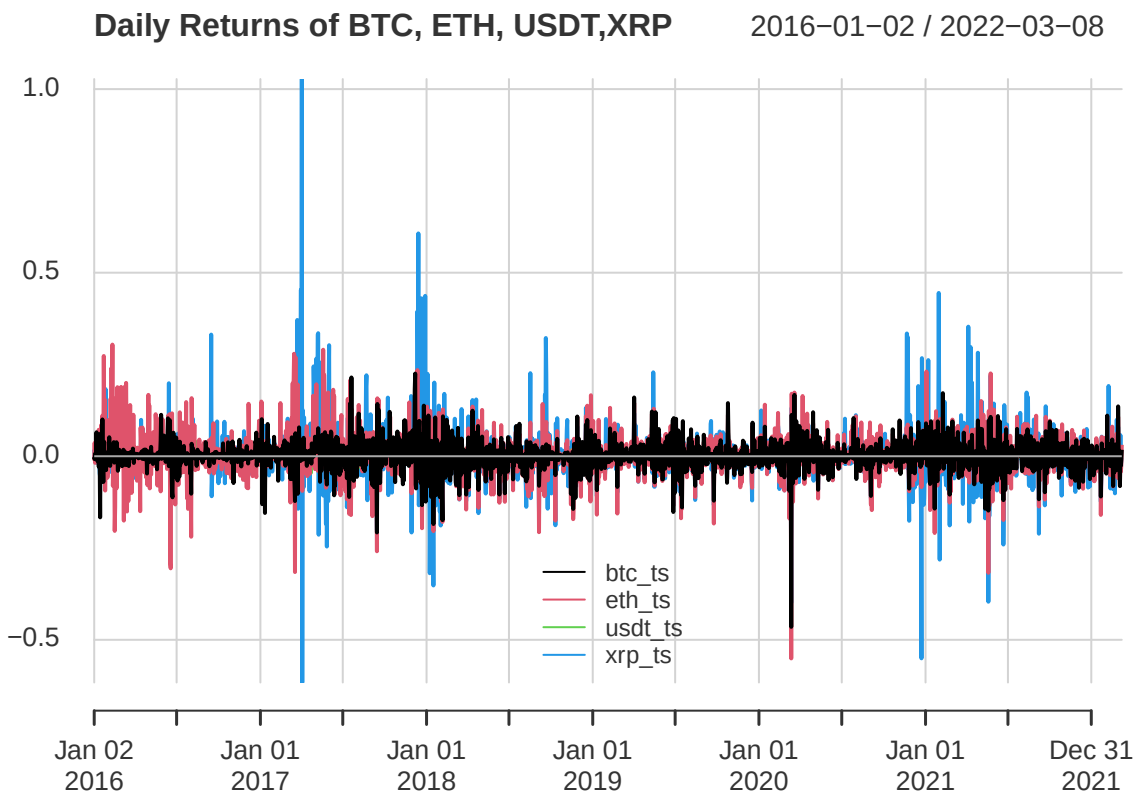
```r
eth_monthly <- na.omit(eth_monthly)
usdt_monthly <- na.omit(usdt_monthly)
xrp_monthly <- na.omit(xrp_monthly)

btc_ts <- ts(btc$returns, frequency = 365, start = c(2016,1))
eth_ts <- ts(eth$returns, frequency = 365, start = c(2016,1))
usdt_ts <- ts(usdt$returns, frequency = 365, start = c(2016,1))
xrp_ts <- ts(xrp$returns, frequency = 365, start = c(2016,1))

btc_ts2 <- ts(btc_monthly$returns, frequency = 12, start = c(2016,1))
eth_ts2 <- ts(eth_monthly$returns, frequency = 12, start = c(2016,1))
usdt_ts2 <- ts(usdt_monthly$returns, frequency = 12, start = c(2016,1))
xrp_ts2 <- ts(xrp_monthly$returns, frequency = 12, start = c(2016,1))
```

II. Results

A) Time Series Plots

From the first graph of all the returns series, we can see that XRP has had the highest volatility. Meanwhile, USDT has the most stable returns.

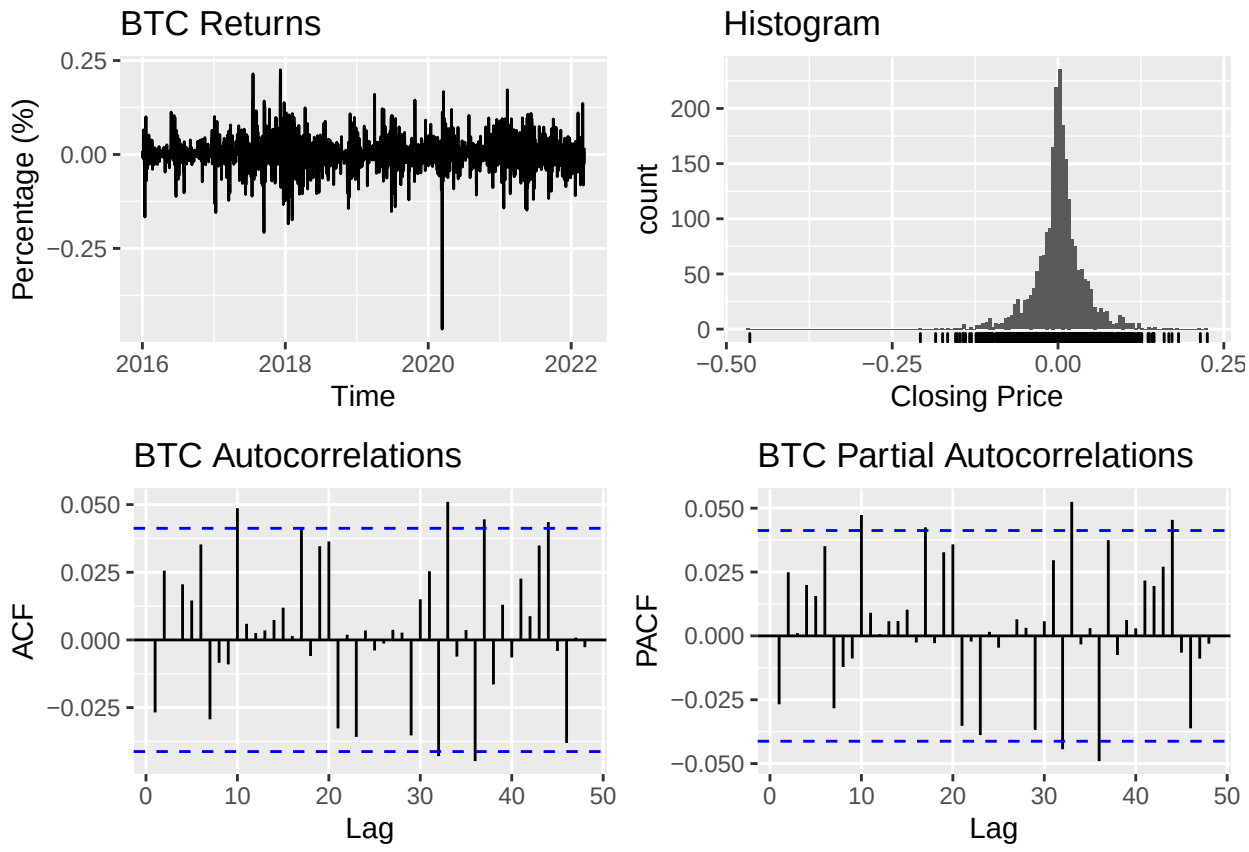
```
# plot returns together together
returns <- data.frame(btc_ts,eth_ts,usdt_ts,xrp_ts)
rownames(returns) <- btc$timestamp
chart.TimeSeries(returns,
      legend.loc="bottom",
      main="Daily Returns of BTC, ETH, USDT,XRP",)
```



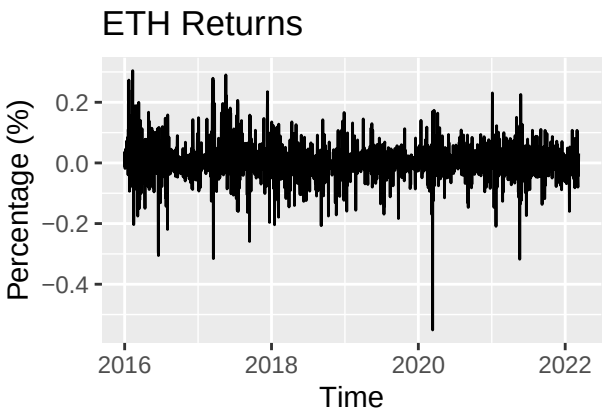
In the individual plots, we observe that the returns data for all the cryptocurrencies look stationary and that they have ACFs and PACFs that resemble white noise. Therefore, they may not be useful as regressors. This also indicates that it may be difficult to model BTC's returns over time.

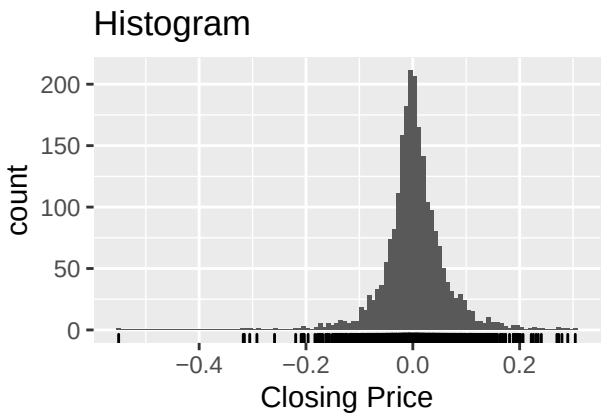
```
library(glue)
tsplot <- function(y,series_name) {
  ax1 = autoplot(y, main = sprintf("%s Returns",series_name),
         ylab = "Percentage (%)",xlab = "Time")
  `Closing Price` = y
  ax2 = gghistogram(`Closing Price`) + ggtitle("Histogram")
  ax3 = ggAcf(y, main = glue('{series_name} Autocorrelations'),
```

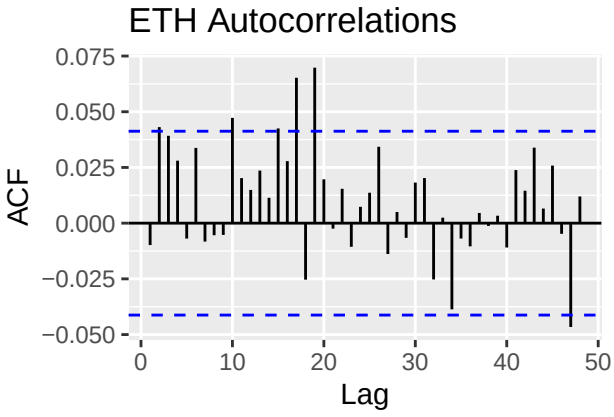
```
              xlab="Lag",lag.max = 48)
  ax4 = ggPacf(y, main = glue('{series_name} Partial Autocorrelations'),
               xlab="Lag",lag.max = 48)
  grid.arrange(ax1,ax2,ax3,ax4,ncol=2,nrow = 2)
}
tsplot(btc_ts,"BTC")
```

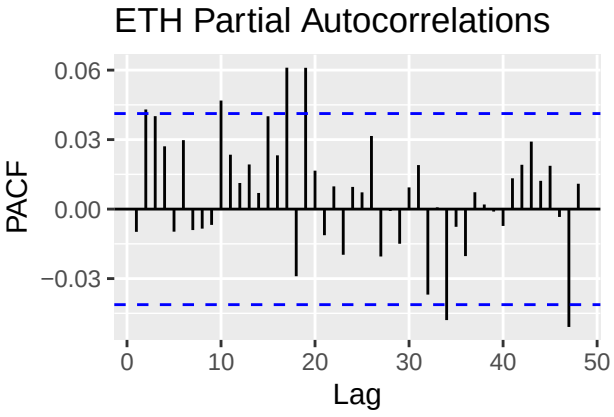



```
tsplot(eth_ts,"ETH")
```

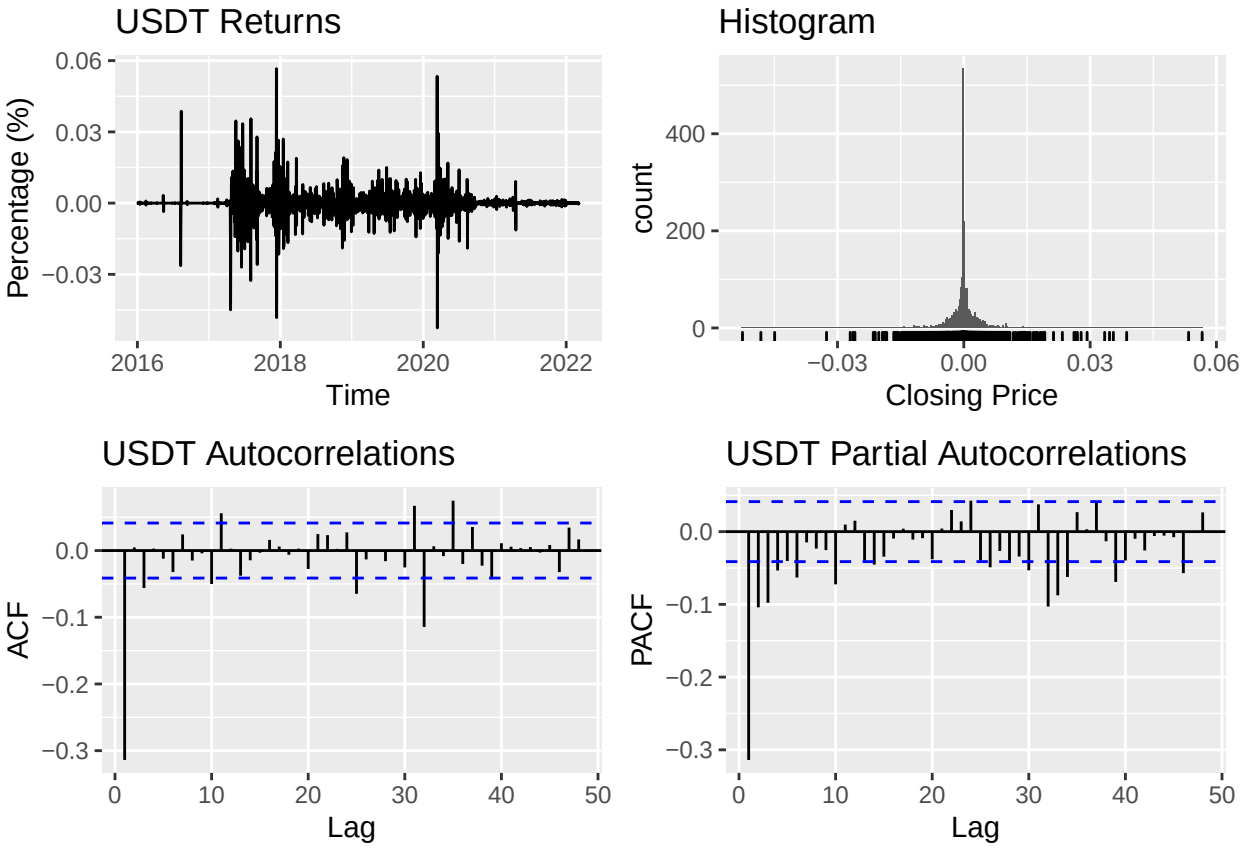




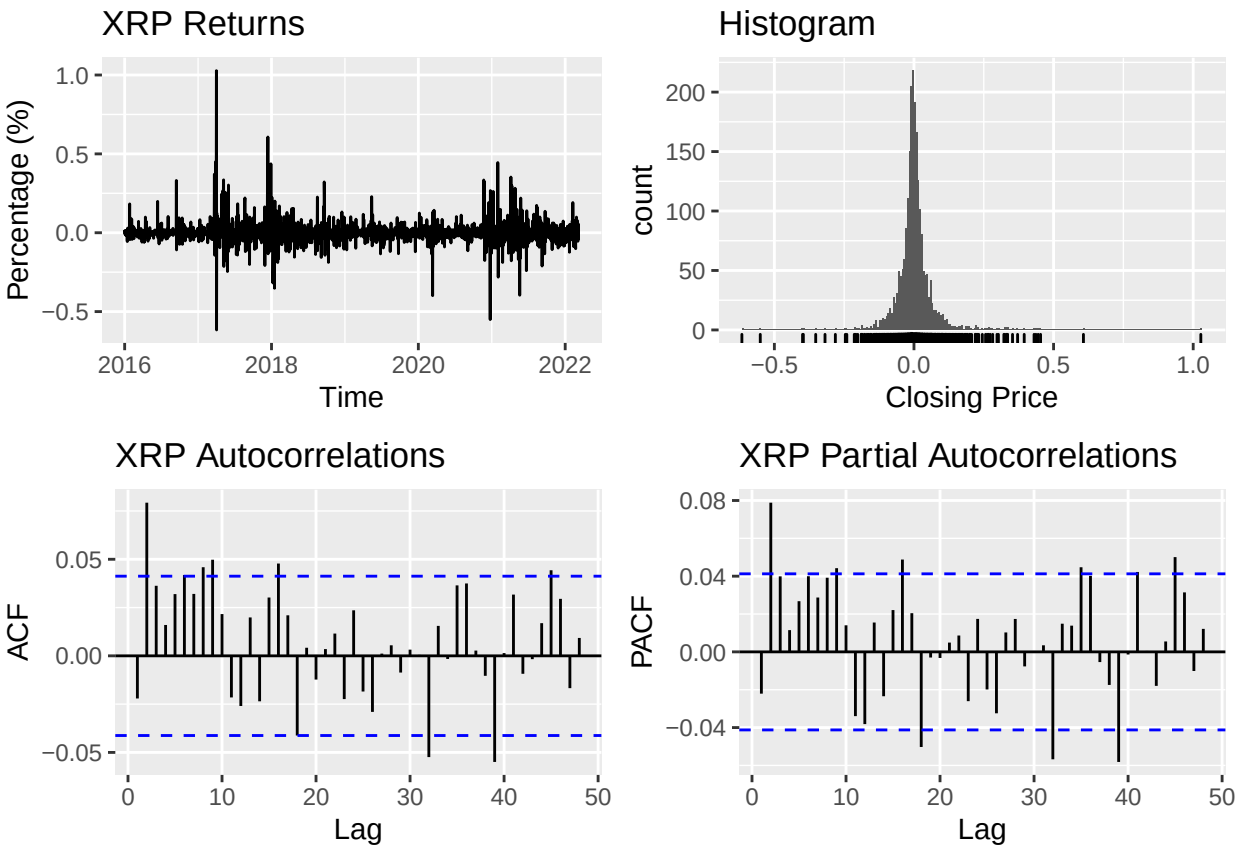





```
tsplot(usdt_ts,"USDT")
```

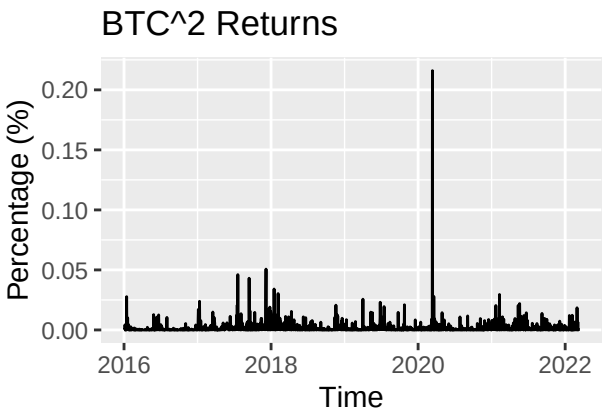


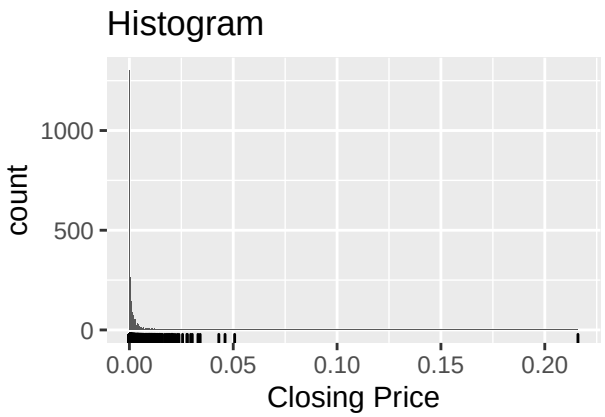

```
tsplot(xrp_ts,"XRP")
```

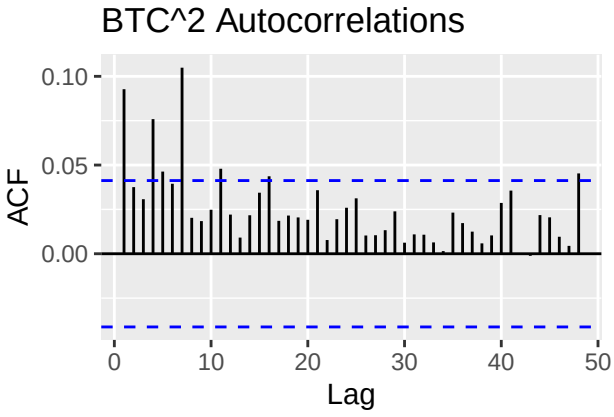


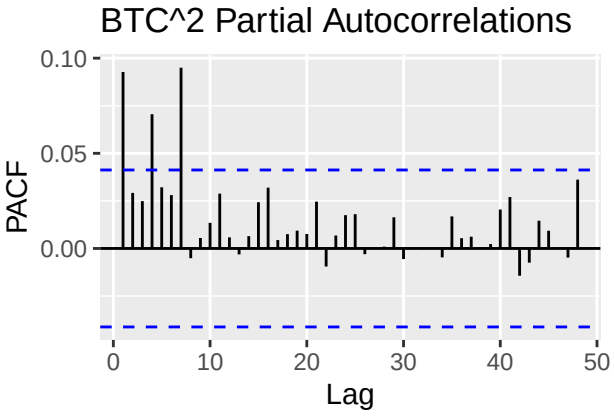
However, non-linear relationships between the lags of BTC's returns do exist. Thus, it may be worthwhile to attempt to model this data.

```
tsplot(btc_ts^2,"BTC^2")
```

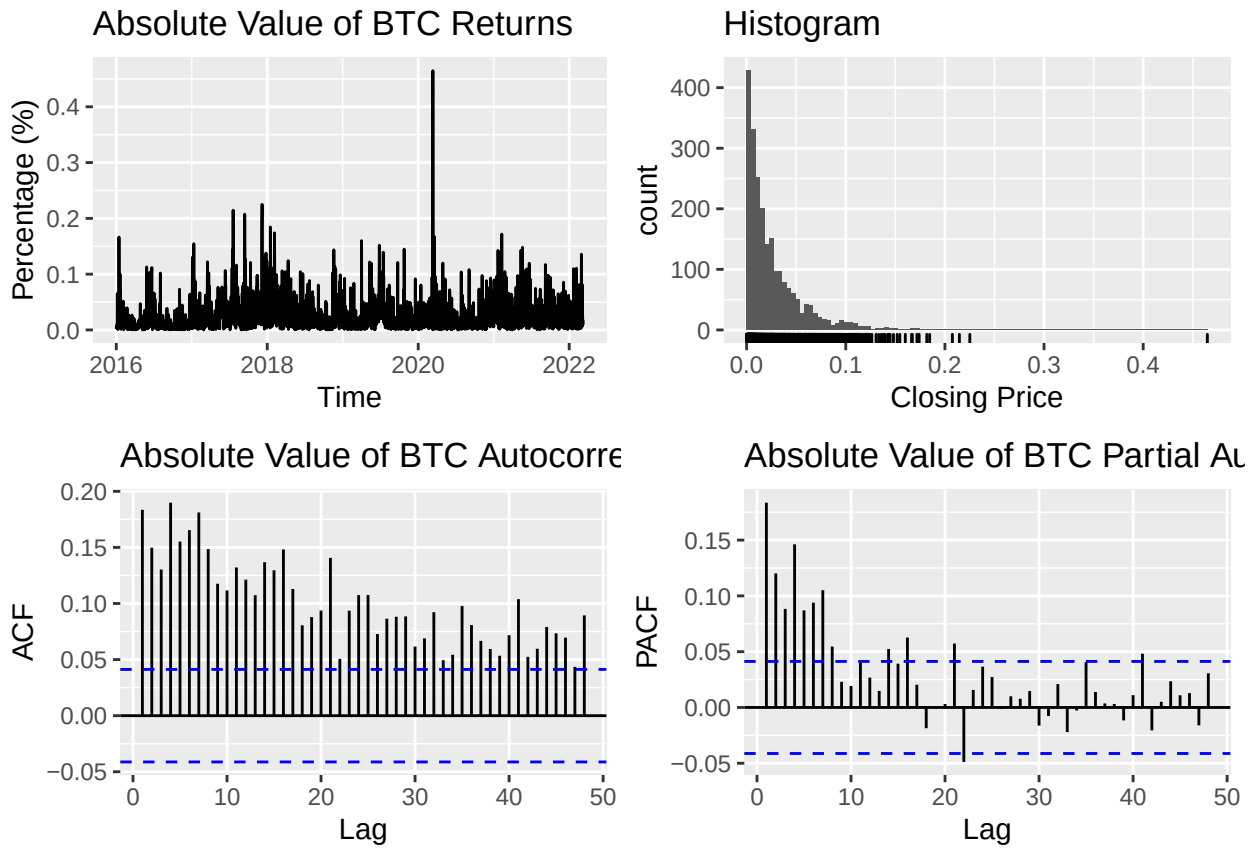








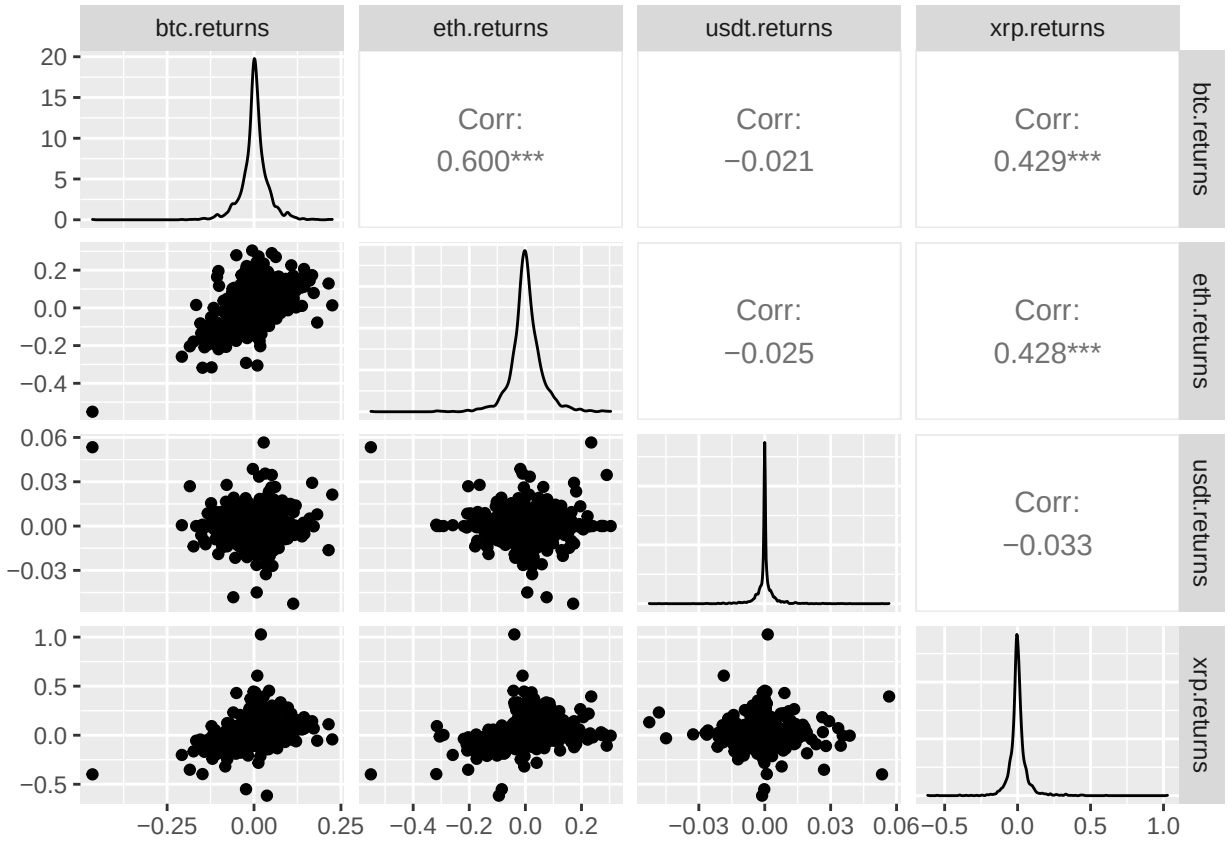

```
tsplot(abs(btc_ts),"Absolute Value of BTC")
```



B) Correlations

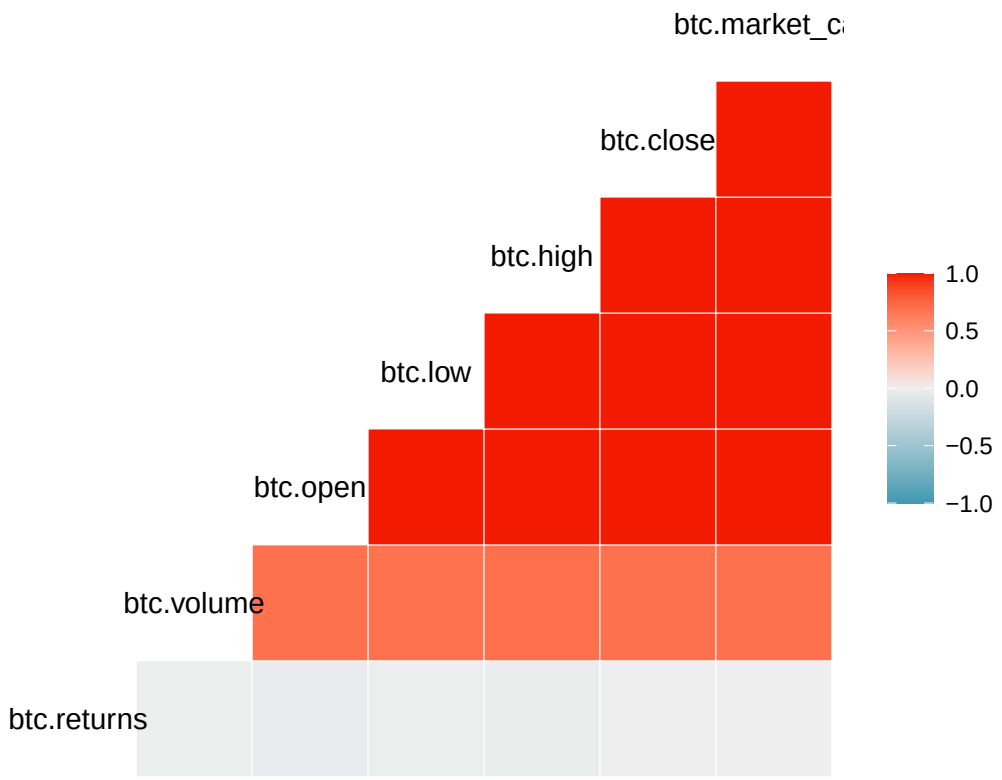
We'll utilize a scatterplot matrix and correlation plot to visualize the relationships between all of the currencies' returns. The scatterplot matrix shows that BTC, ETH, and XRP have moderately positive correlations with each other, suggesting that they may be useful regressors for modelling BTC's returns.

```
ret_data <- data.frame(btc$returns,eth$returns,usdt$returns,xrp$returns)
ggpairs(ret_data)
```



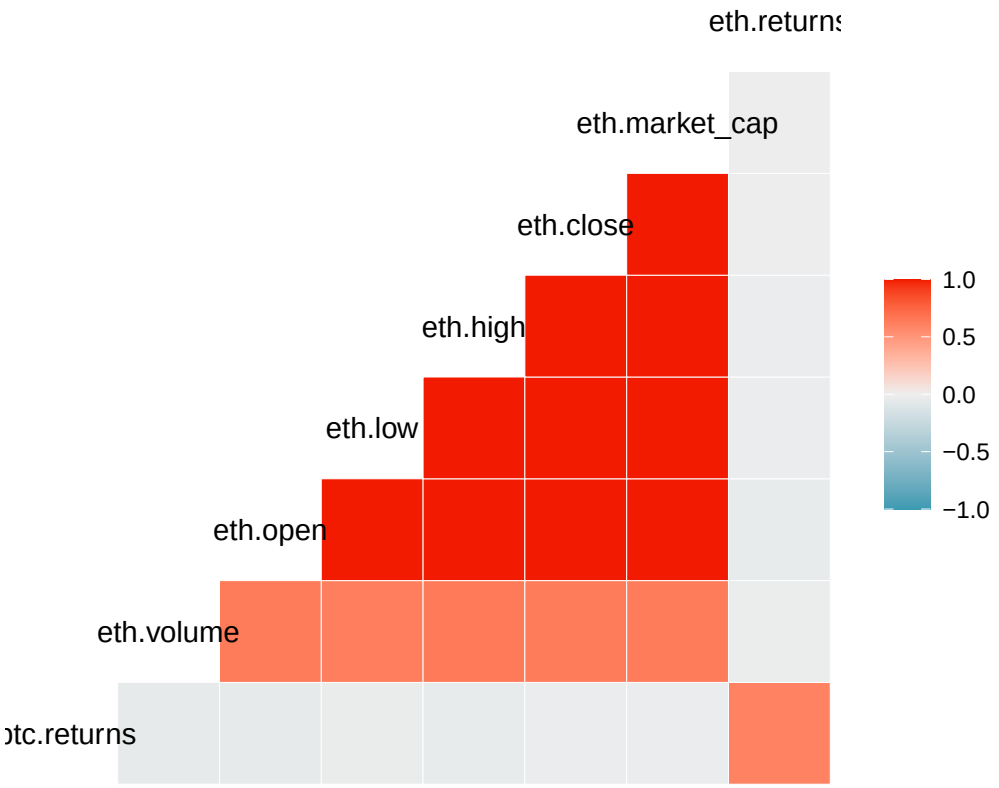
There is very little correlation between BTC's returns and other data about it.

```
# how much do BTC's returns correlate with  its own data
self_data <- data.frame(btc$returns,btc$volume,btc$open,btc$low,
                        btc$high,btc$close,btc$market_cap)
ggcorr(self_data)
```



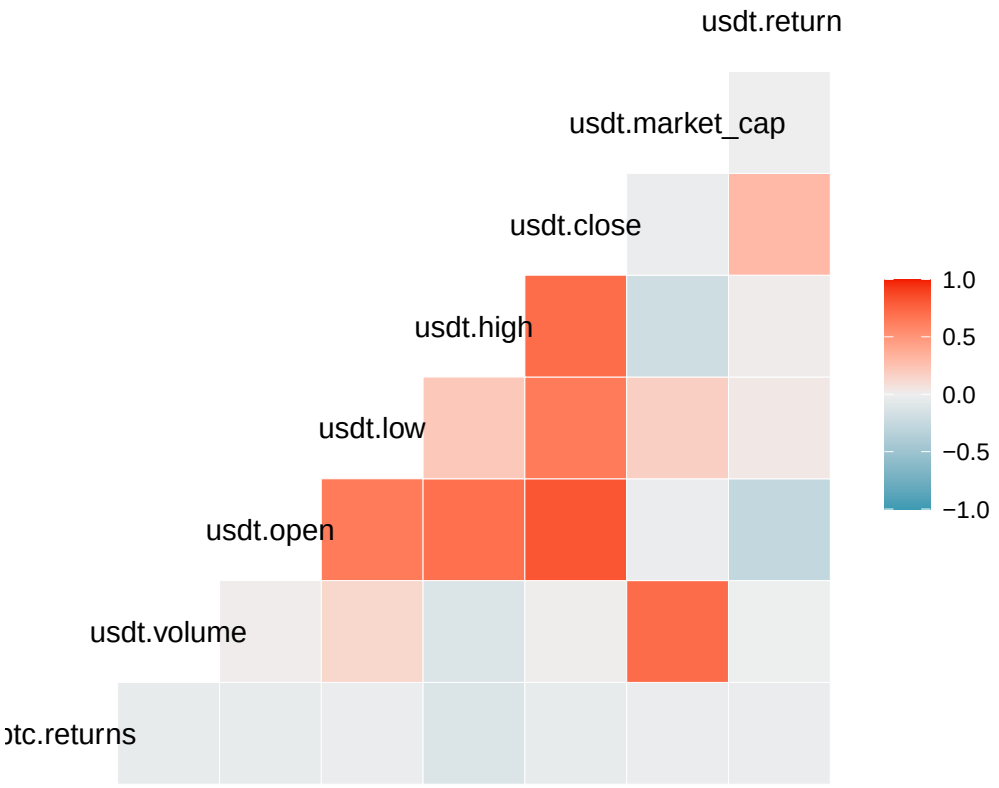
ETH's returns is the only variable that is positively linearly correlated with BTC's returns.

```
# BTC returns and ETH data
eth_data <- data.frame(btc$returns,eth$volume,eth$open,eth$low,eth$high,
                       eth$close,eth$market_cap,eth$returns)
ggcorr(eth_data)
```



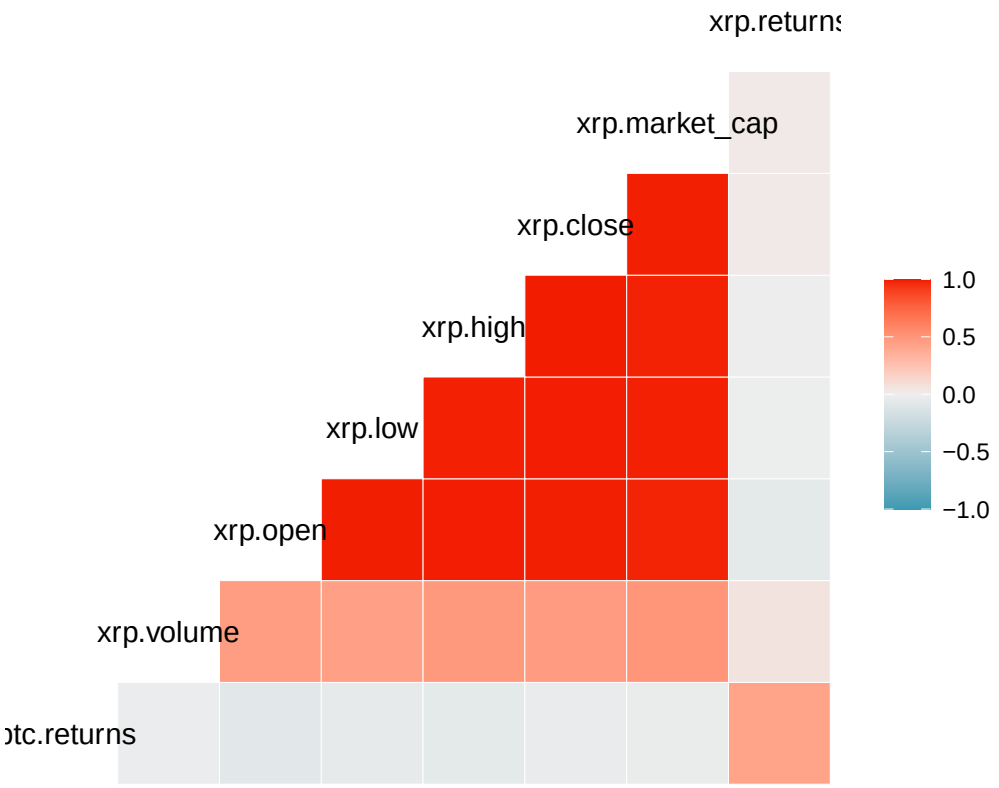
There is no correlation between BTC's returns and USDT's data.

```
# BTC returns and USDT data
usdt_data <- data.frame(btc$returns,usdt$volume,usdt$open,usdt$low,usdt$high,
                        usdt$close,usdt$market_cap,usdt$returns)

ggcorr(usdt_data)
```



XRP's returns is the only variable that is positively linearly correlated with BTC's returns.

```
# BTC returns and ETH data
xrp_data <- data.frame(btc$returns,xrp$volume,xrp$open,xrp$low,xrp$high,
                       xrp$close,xrp$market_cap,xrp$returns)
ggcorr(xrp_data)
```

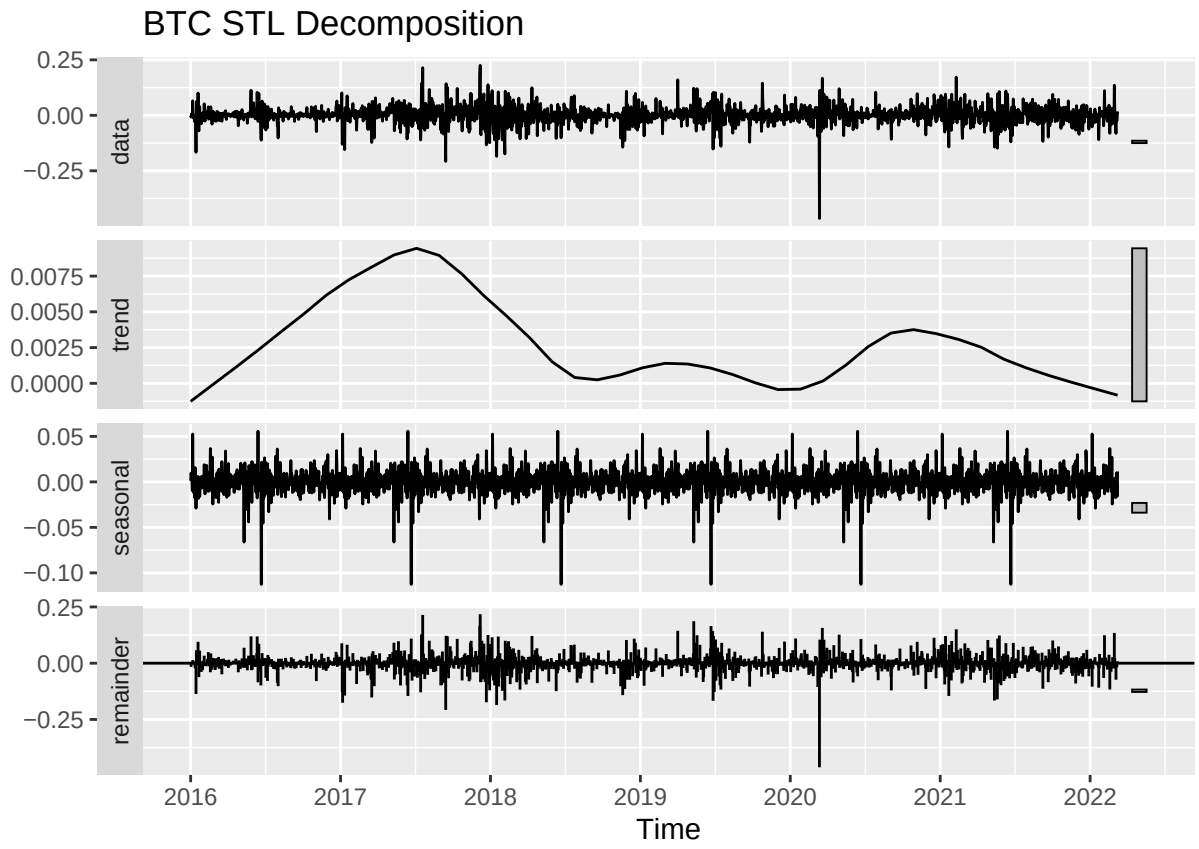


From this, we can conclude that XRP's and ETH's returns are potentially appropriate regressors for forecasting BTC's returns, and we can incorporate them into our models.

C) STL Decomposition

The plot shows a trend, but the trend's magnitude is so small that it can be considered insignificant. There may be a seasonal pattern in the data that can be exploited, as it appears that there are regular periods of higher volatility. There also appears to be some structure in the residuals. To affirm or reject these observations, we can build several models and test their significance.

```
autoplot(stl(btc_ts,s.window = "periodic",robust=TRUE),
     main = "BTC STL Decomposition")
```



D) ARIMA

Fitting the Model

We will fit an ARIMA to BTC's returns in two ways. First, we will use an ARIMA to model the data itself. Second, we will fit a regression to the data and fit the ARIMA to the regression model's errors.

For the pure ARIMA model, we obtain a ARIMA(0,0,0) process with non-zero mean. This suggests that the ARIMA process is not suitable for modeling this data.

```
arima_btc <- auto.arima(btc_ts)
summary(arima_btc)
```

```
## Series: btc_ts
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       2e-03
## s.e.  8e-04
##
## sigma^2 = 0.001587:  log likelihood = 4073.74
## AIC=-8143.49   AICc=-8143.48   BIC=-8132.04
##
## Training set error measures:
```

```
##                         ME      RMSE       MAE  MPE MAPE      MASE        ACF1
## Training set -3.290248e-17 0.0398323 0.0260389 -Inf  Inf 0.6138265 -0.02678913
```

In the second model, we fit a regression to the returns data and obtain a ARIMA (0,0,1) to model our errors. Since the data was already covariance stationary, there was no need for differencing it. Thus, I = 0. The lack of an AR(1) process suggests that the errors do not have persistent autocorrelations. There is also no seasonality or trend captured by the model.

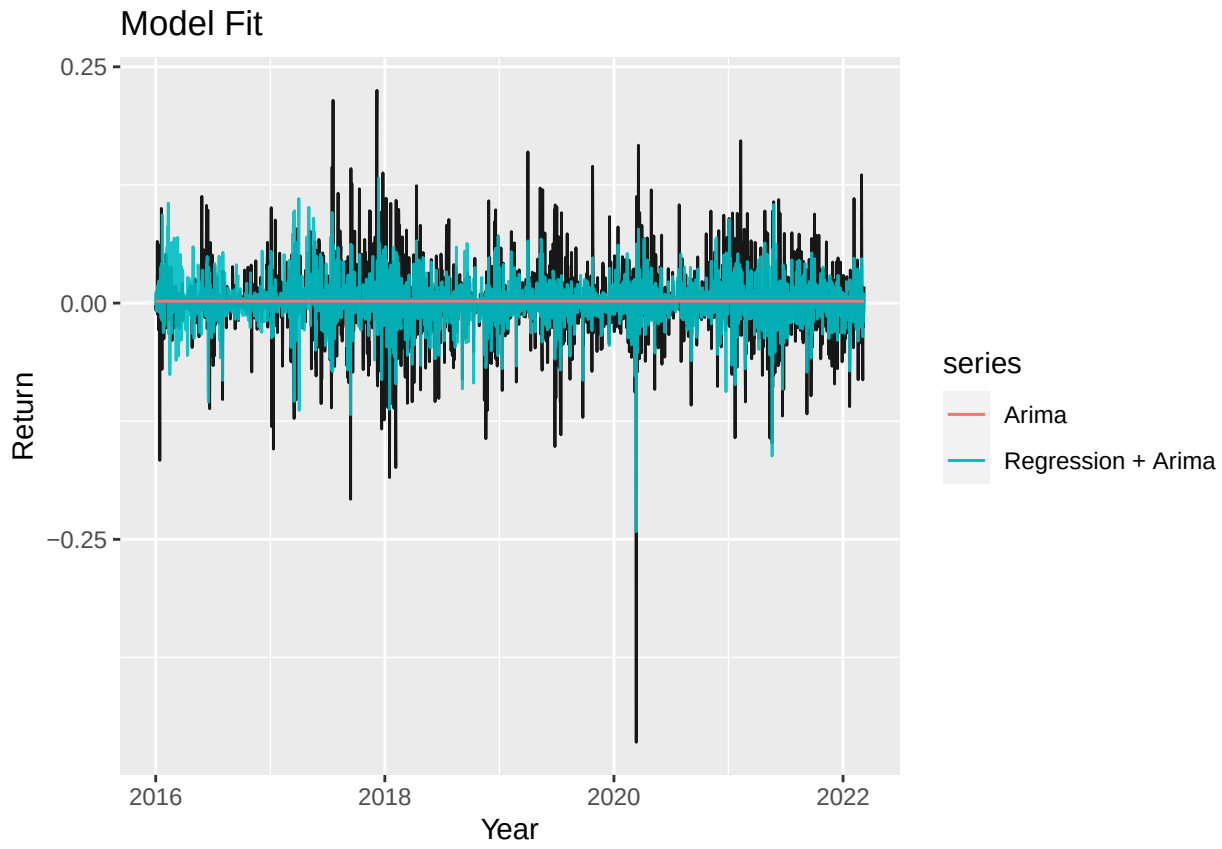
The second model lower AIC, AICc, and BIC. Therefore, it is better than the simpler ARIMA model.

```
btc_tib <- as_tibble(btc)
reg_arima_btc <- auto.arima(btc_tib[,"returns"], xreg=cbind(eth$returns,xrp$returns))
summary(reg_arima_btc)
```

```
## Series: btc_tib[, "returns"]
## Regression with ARIMA(0,0,1) errors
##
## Coefficients:
##          ma1   xreg1   xreg2
##       0.0601  0.3529  0.1213
## s.e.  0.0214  0.0124  0.0103
##
## sigma^2 = 0.0009561:  log likelihood = 4647.06
## AIC=-9286.12   AICc=-9286.1   BIC=-9263.23
##
## Training set error measures:
##                        ME       RMSE        MAE  MPE MAPE      MASE
## Training set 0.0004673229 0.03090052 0.02055618 -Inf  Inf 0.5264655
##                       ACF1
## Training set -0.0008090591
```

INTERPRET

```
arima_fit <- ts(arima_btc$fitted,frequency = 365, start = c(2016,1))
reg_arima_fit <- ts(reg_arima_btc$fitted,frequency = 365, start = c(2016,1))
autoplot(btc_ts, series="BTC Series", col="black", alpha = 0.9) + ggtitle("Model Fit") +
  autolayer(reg_arima_fit, series="Regression + Arima", alpha = 0.9) +
  autolayer(arima_fit, series="Arima") +
  ylab("Return") + xlab("Year")
```

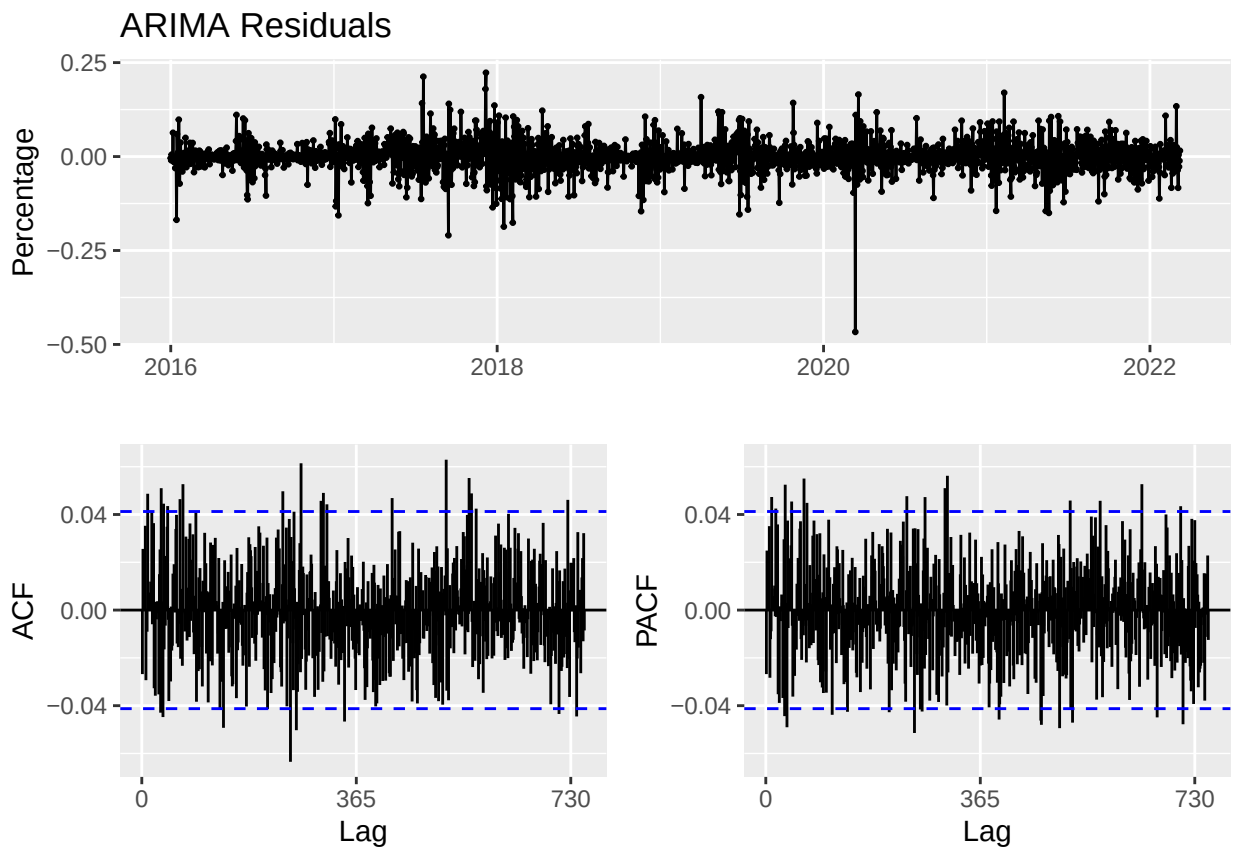


ACF/PACF of Residuals

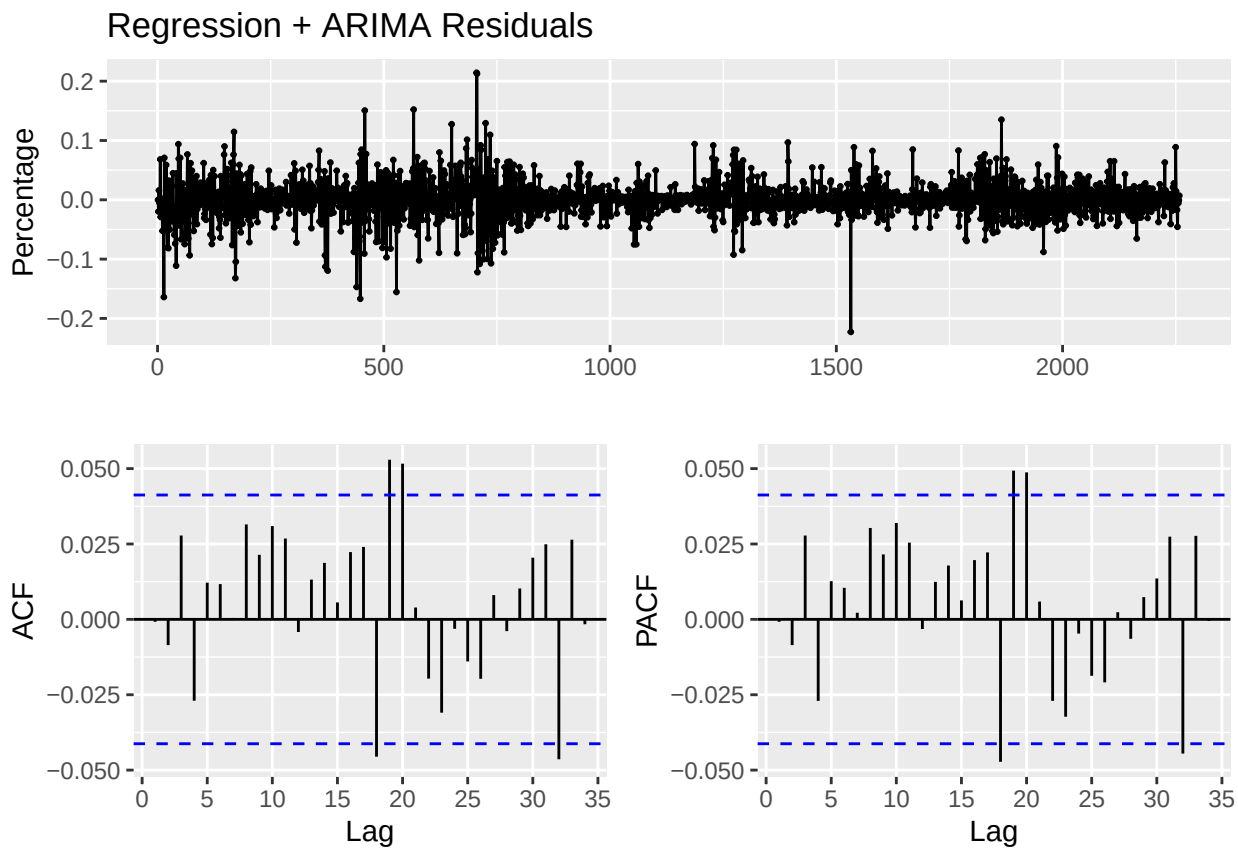
The residuals of the pure ARIMA model do not show very much structure. However, this doesn't mean that the ARIMA is a good model for forecasting the data because it would only predict the mean value of the series at future points in time.

The plot of the residuals appears to retain some structure which can be modeled. However, the magnitude of the autocorrelation spikes seen on the ACF and PACF plots suggest that this structure may not be very significant. Therefore, the regression + ARIMA model fits the data quite well.

```
ggtsdisplay(resid(arima_btc),main = "ARIMA Residuals", ylab = "Percentage")
```

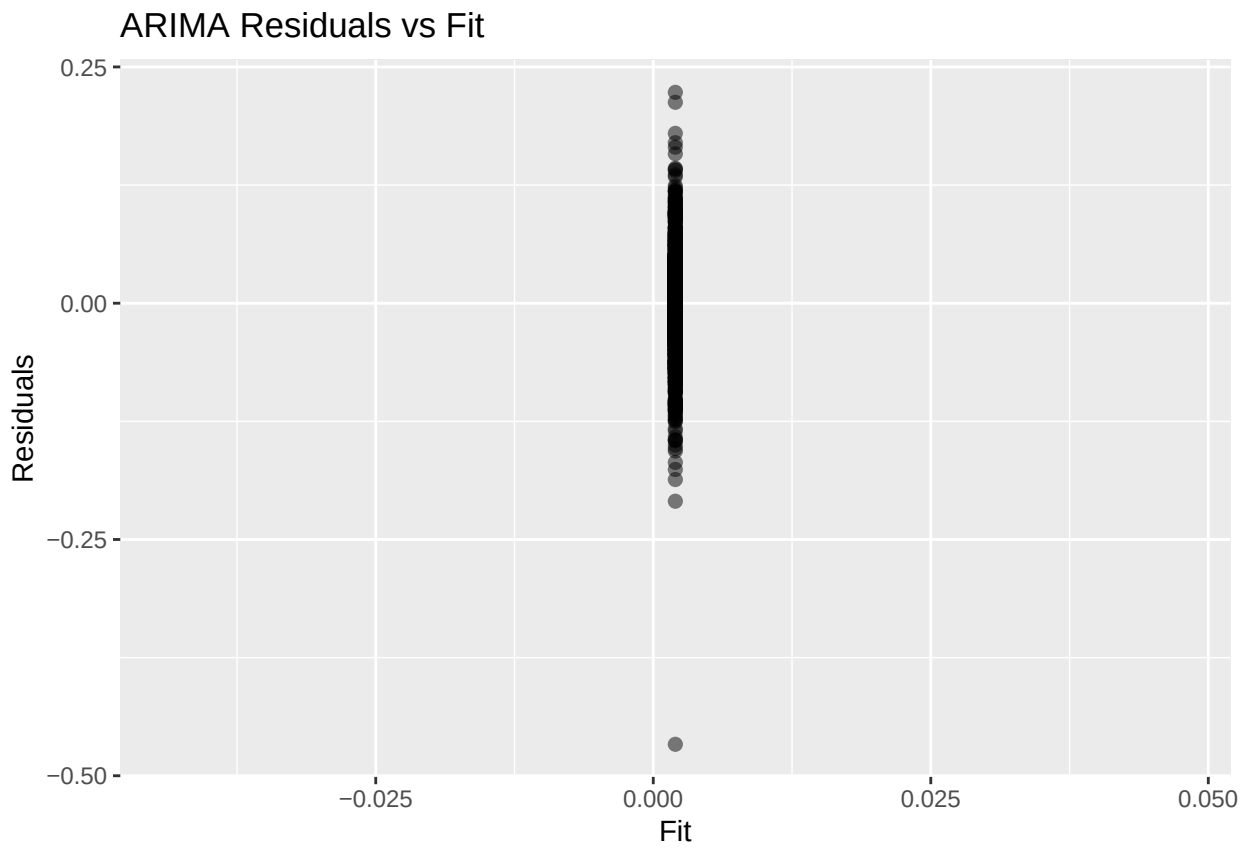



```
ggtsdisplay(resid(reg_arima_btc), main = "Regression + ARIMA Residuals", ylab = "Percentage")
```

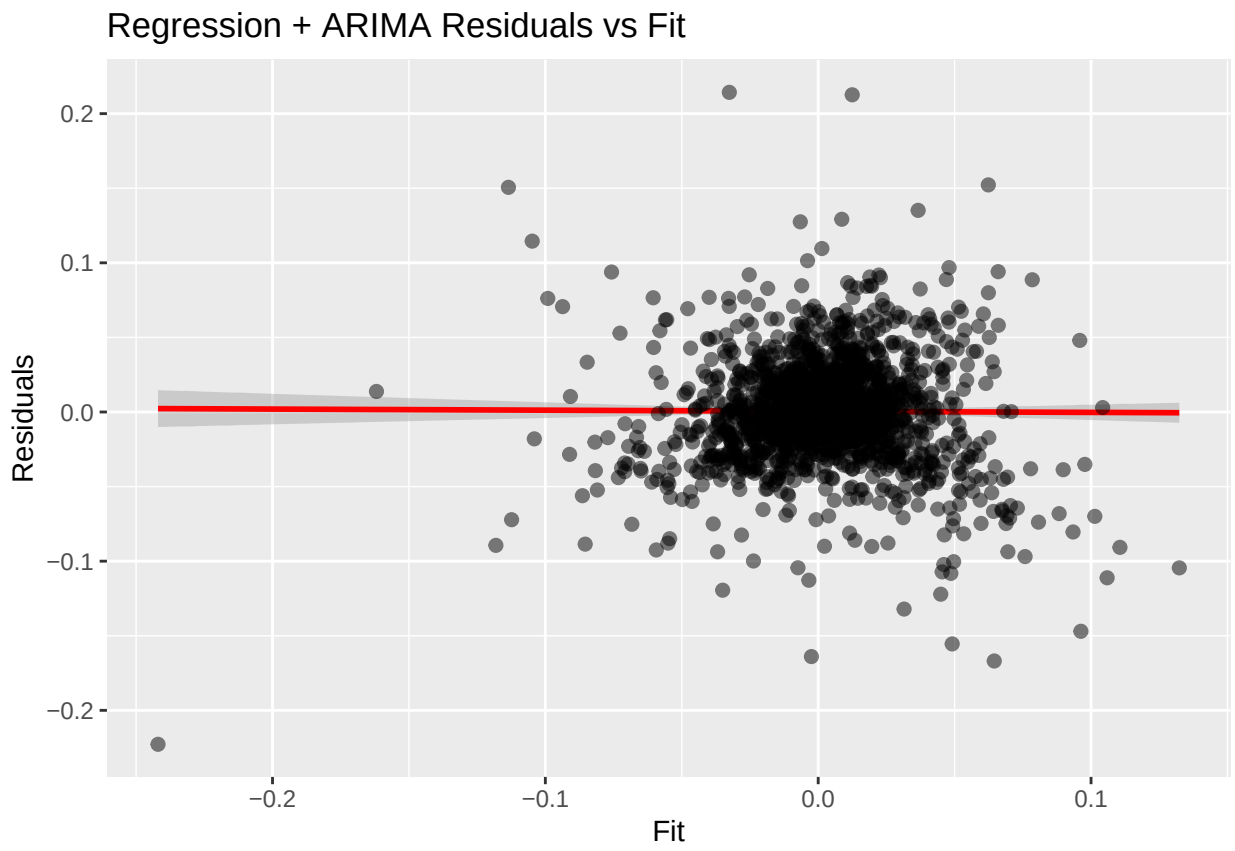


There is not much of a linear relationship between the fit and the residuals, suggesting that our model's fit did pretty well.

```
plot_resid_fit <- function(x,y,title) {
  data = data.frame(x = x, y = y)
  ggplot(data, aes(x =  x, y = y)) +
    geom_smooth(method = "lm", se=T, col = "red",
                formula = y ~ x, show.legend = T) +
    scale_x_continuous() + scale_y_continuous() +
    geom_point(alpha = 0.5,size = 2, col = "black") +
    xlab("Fit") + ylab("Residuals") +
    ggtitle(glue("{title} Residuals vs Fit"))
}
plot_resid_fit(fitted(arima_btc),resid(arima_btc),"ARIMA")
```



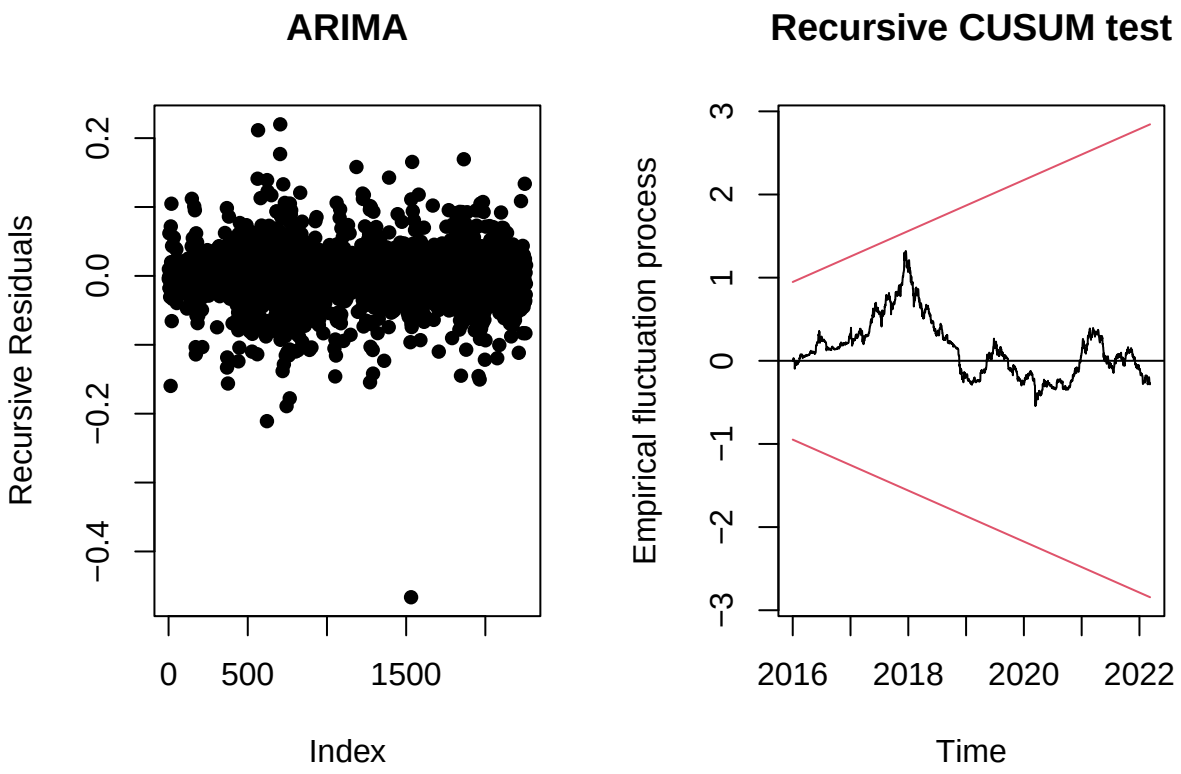

```
plot_resid_fit(fitted(reg_arima_btc),resid(reg_arima_btc),"Regression + ARIMA")
```



Recursive Residuals

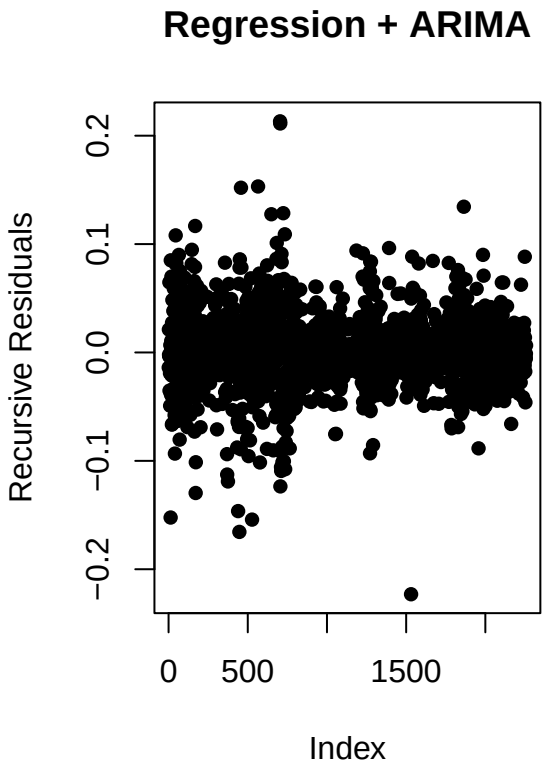
INTERPRET

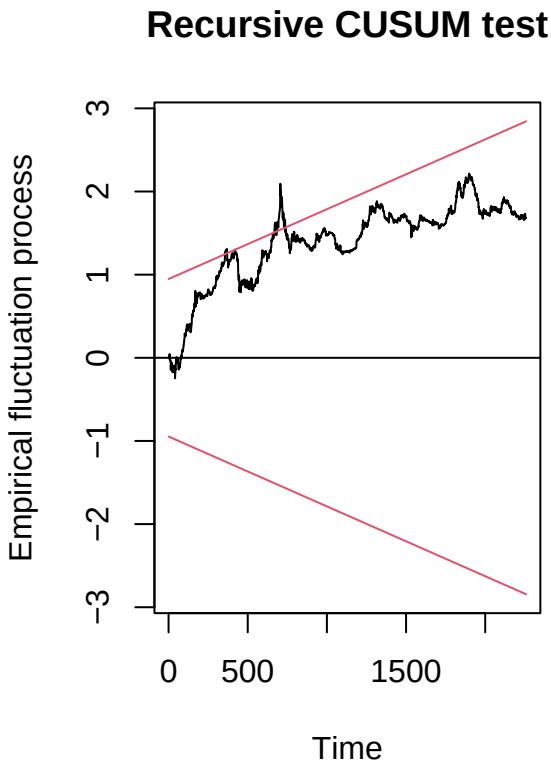
```
par(mfrow=c(1,2))
y = recresid(arima_btc$res ~ 1)
plot(y, pch = 16, main = "ARIMA", ylab = "Recursive Residuals")
plot(efp(arima_btc$residuals~1, type = "Rec-CUSUM"))
```



INTERPRET The recursive residuals are scattered around 0 and do not show any dynamics, reaffirming that the Regression + Arima Fit is a good model.

```
par(mfrow=c(1,2))
y = recresid(reg_arima_btc$res ~ 1)
plot(y, pch = 16, main = "Regression + ARIMA", ylab = "Recursive Residuals")
plot(efp(reg_arima_btc$residuals~1, type = "Rec-CUSUM"))
```





E) ETS

Fitting the Model

INTERPRET

```
ets_btc <- ets(btc_ts2)
summary(ets_btc)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = btc_ts2)
##
##   Smoothing parameters:
##     alpha = 0.2558
##
##   Initial states:
##     l = 0.0485
##
##   sigma:  0.2041
##
##      AIC     AICc      BIC
## 87.28014 87.62300 94.19234
```

```
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.003897378 0.2013235 0.1563278 125.7599 321.8229 0.6128094
##                    ACF1
## Training set 0.2102895
```

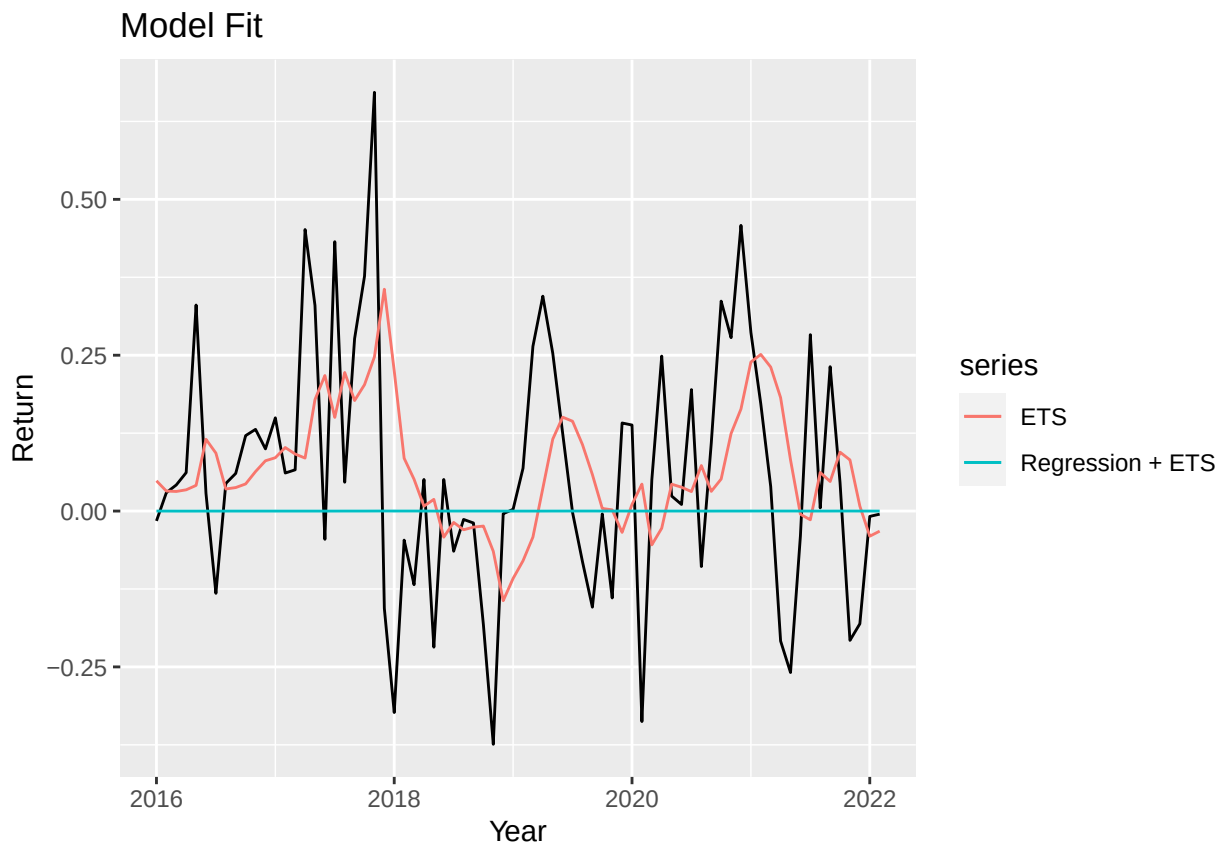
INTERPRET

```
btc_eth_lm <- lm(btc_monthly$returns~eth_monthly$returns)
btc_lm_res_ts <- ts(btc_eth_lm$residuals,freq=12,start=c(2016,1))
reg_ets_btc <- ets(btc_lm_res_ts)
summary(reg_ets_btc)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = btc_lm_res_ts)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0
##
##   sigma:  0.167
##
##      AIC     AICc      BIC
## 57.57386 57.91671 64.48605
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 9.001839e-06 0.1647114 0.1272337 100.0134 100.0134 0.6514489
##                    ACF1
## Training set 0.3391001
```

INTERPRET

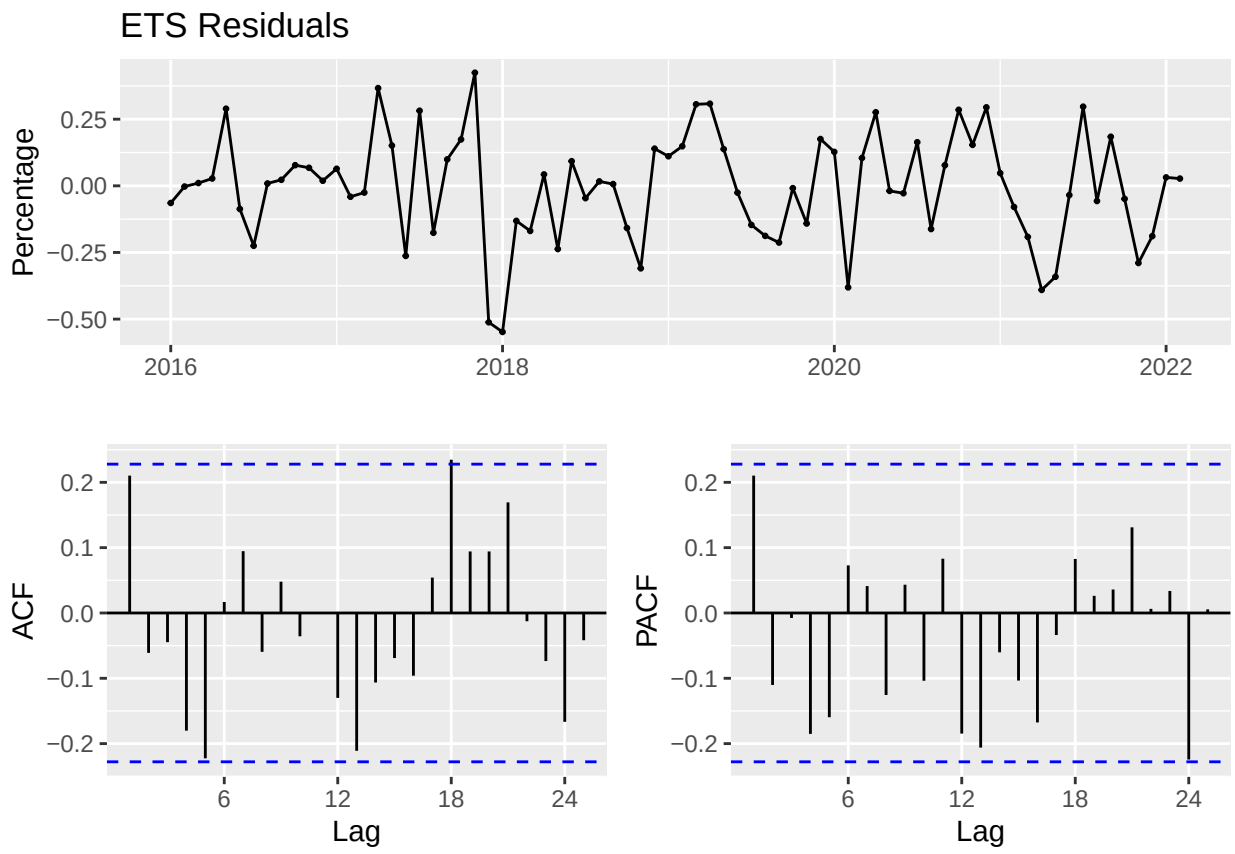
```
autoplot(btc_ts2, series="BTC Series", col="black") + ggtitle("Model Fit") +
  autolayer(ets_btc$fitted, series="ETS") +
  autolayer(reg_ets_btc$fitted, series="Regression + ETS") +
  ylab("Return") + xlab("Year")
```



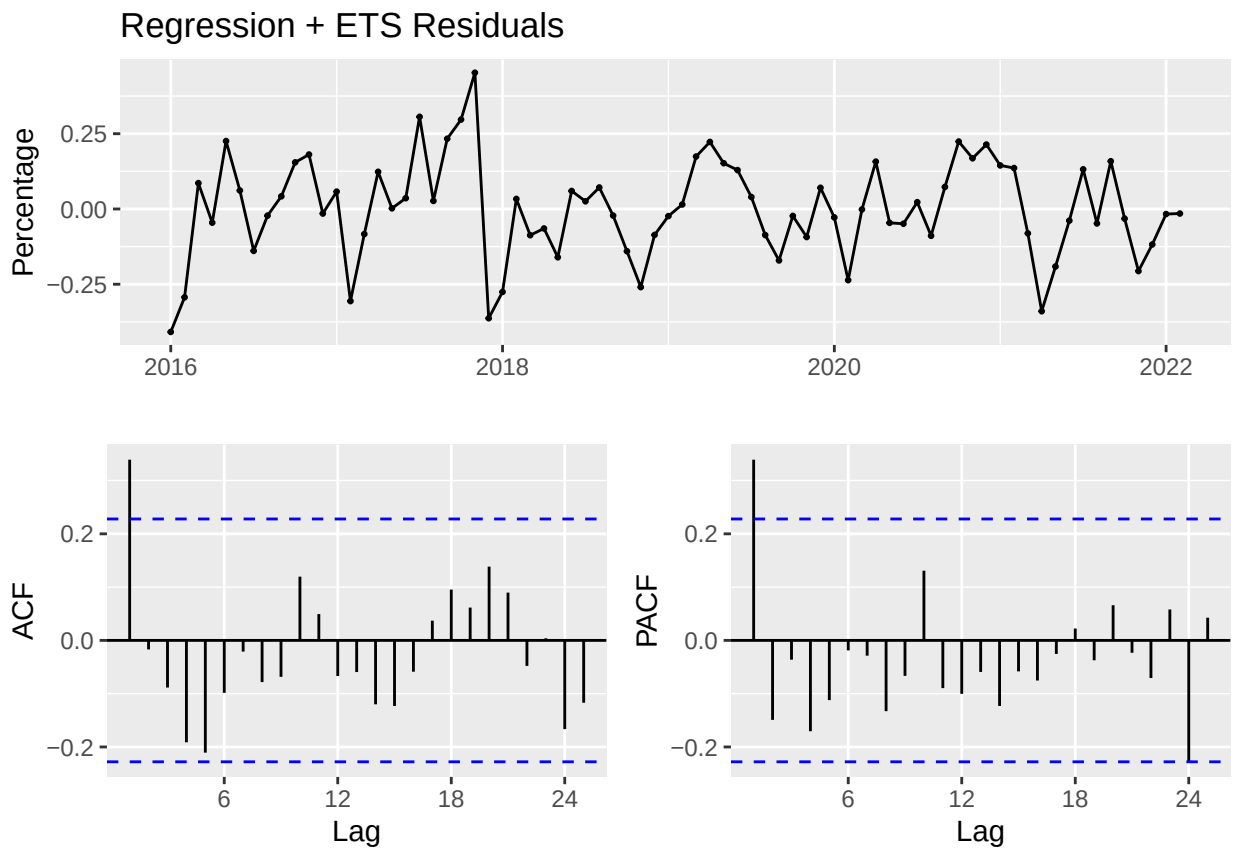
ACF/PACF of Residuals

INTERPRET

```
ggtsdisplay(resid(ets_btc),main = "ETS Residuals", ylab = "Percentage")
```

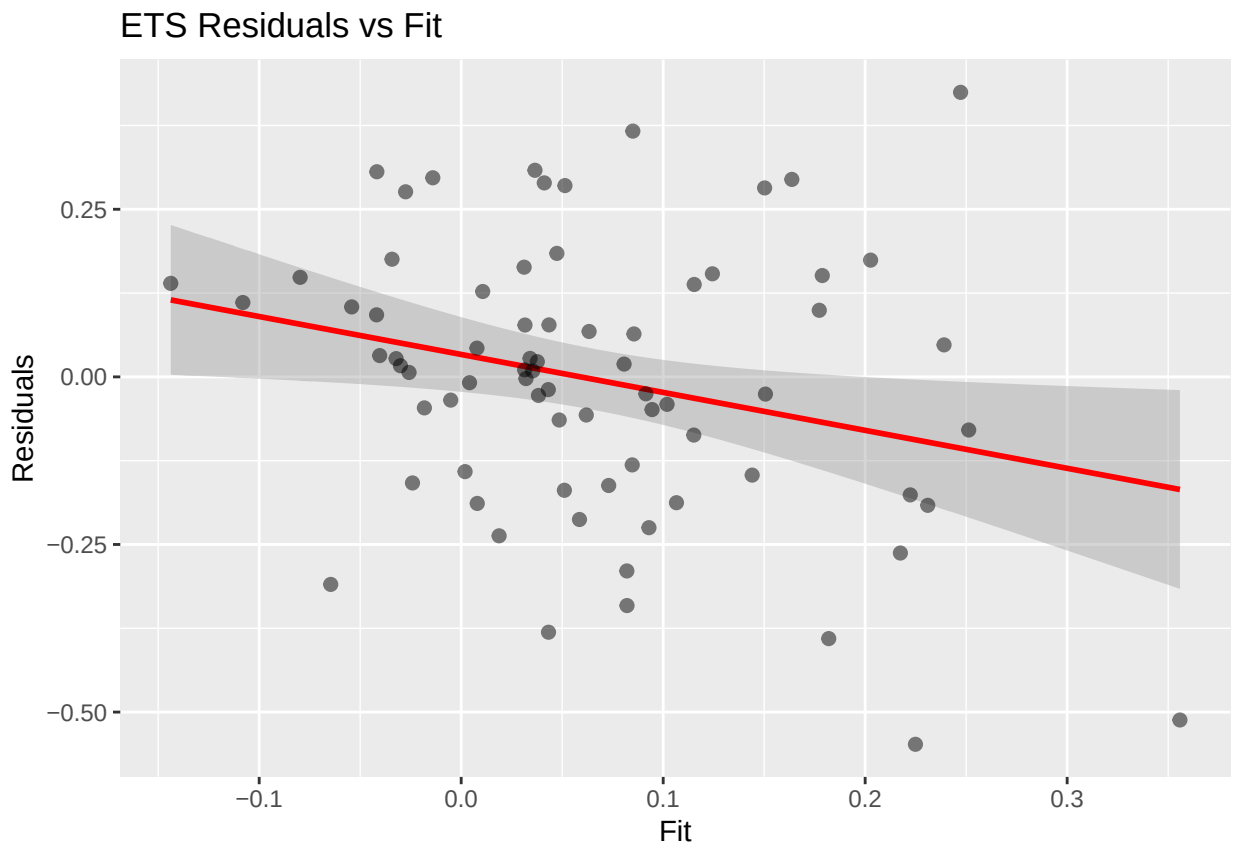



```
ggtsdisplay(resid(reg_ets_btc),main = "Regression + ETS Residuals", ylab = "Percentage")
```

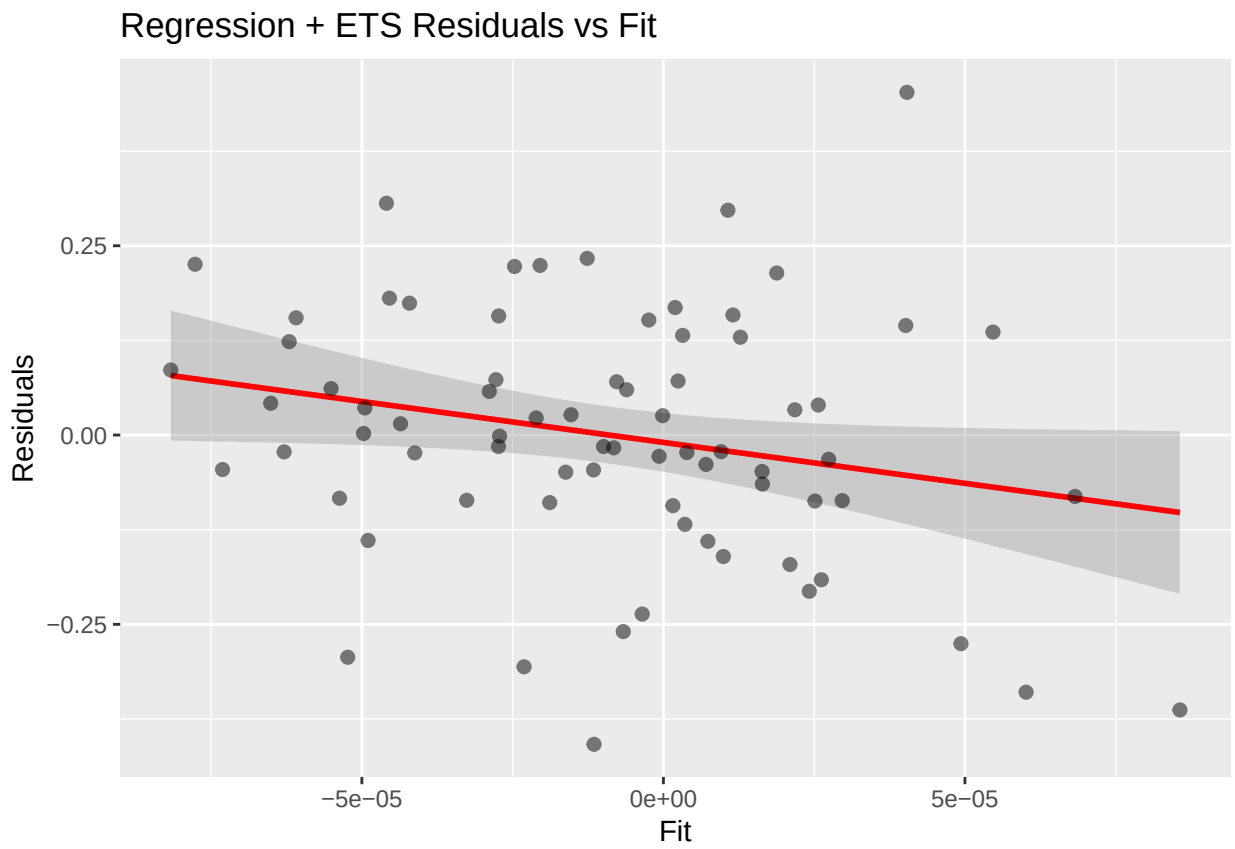


INTERPRET

```
plot_resid_fit(fitted(ets_btc),resid(ets_btc),"ETS")
```



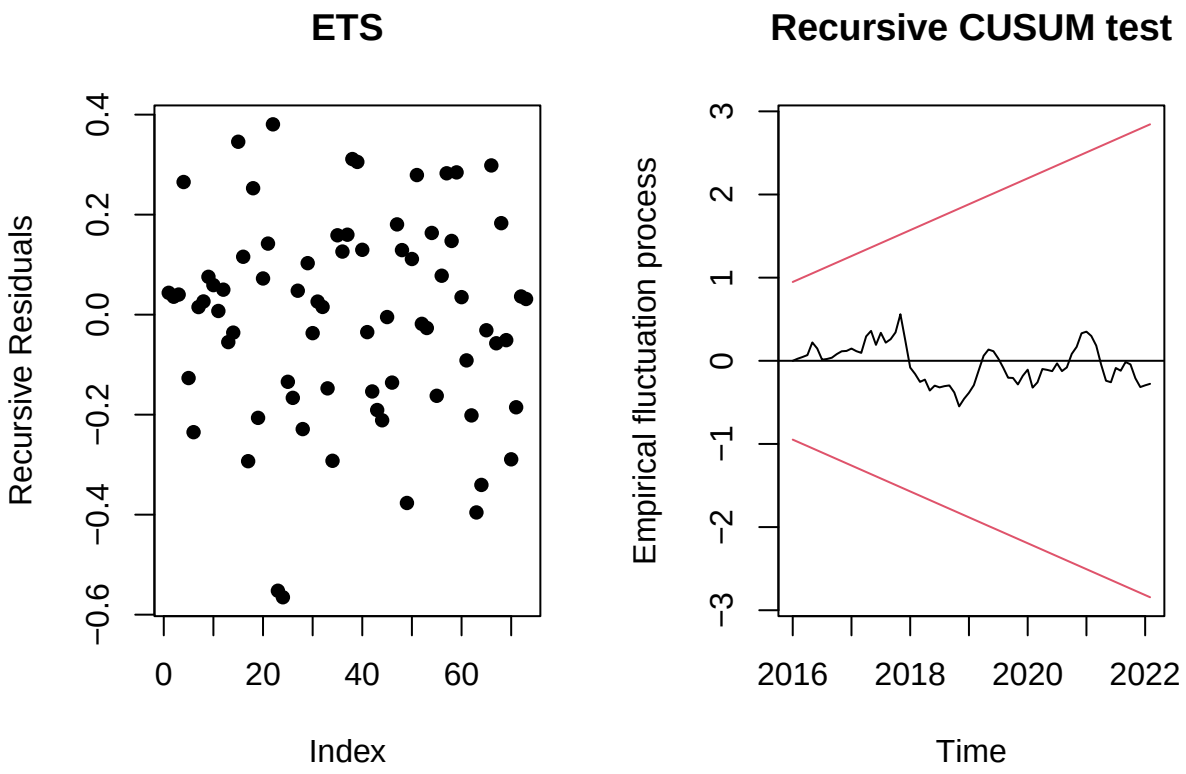

```
plot_resid_fit(fitted(reg_ets_btc),resid(reg_ets_btc),"Regression + ETS")
```



Recursive Residuals

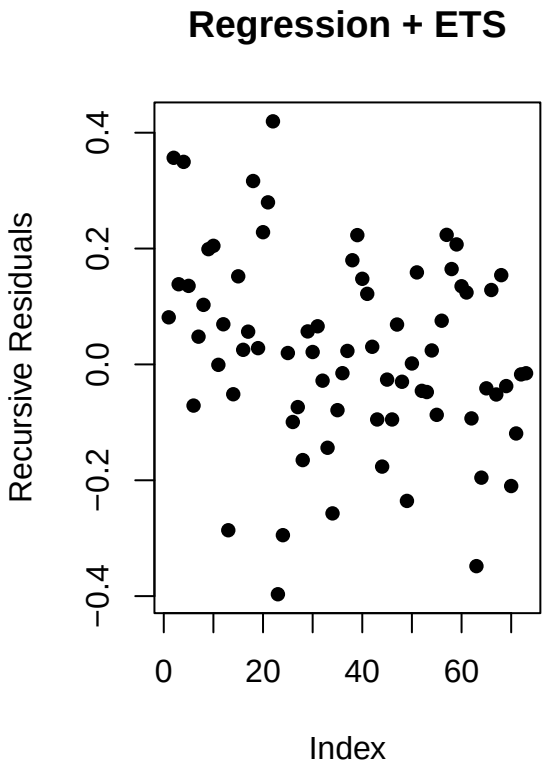
INTERPRET

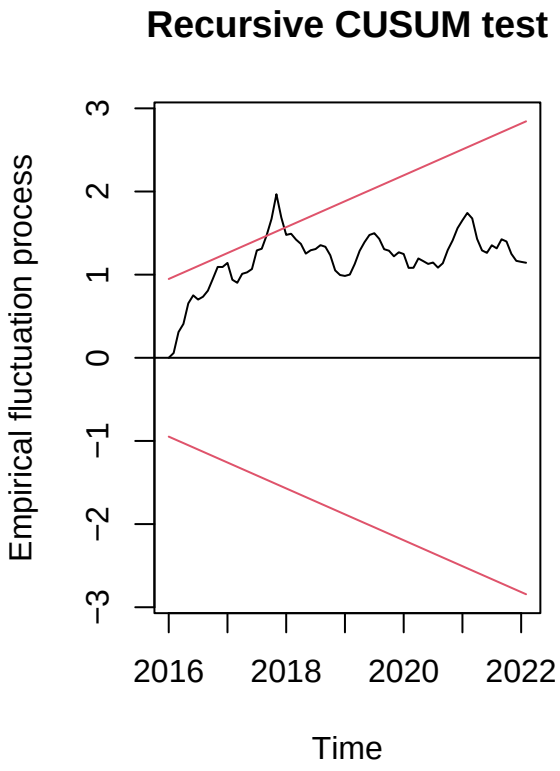
```
par(mfrow=c(1,2))
y = recresid(ets_btc$res ~ 1)
plot(y, pch = 16, main = "ETS", ylab = "Recursive Residuals")
plot(efp(ets_btc$residuals~1, type = "Rec-CUSUM"))
```



INTERPRET

```
par(mfrow=c(1,2))
y = recresid(reg_ets_btc$res ~ 1)
plot(y, pch = 16, main = "Regression + ETS", ylab = "Recursive Residuals")
plot(efp(reg_ets_btc$residuals~1, type = "Rec-CUSUM"))
```





F) Holt-Winters

Fitting the Model

INTERPRET

```
hw_btc <- hw(btc_ts2,seasonal=c("additive"))
summary(hw_btc)
```

```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = btc_ts2, seasonal = c("additive"))
##
##   Smoothing parameters:
##     alpha = 0.0159
##     beta  = 1e-04
##     gamma = 0.0177
##
##   Initial states:
```

```
##     l = 0.0098
##     b = 2e-04
##     s = 0.0423 -0.0053 0.081 -0.0076 -0.0561 0.0895
##            -0.0137 -0.0426 0.088 -0.0133 -0.1036 -0.0586
##
##   sigma:  0.2263
##
##      AIC     AICc      BIC
## 114.5474 125.4760 153.7165
##
## Error measures:
##                       ME      RMSE       MAE      MPE    MAPE      MASE
## Training set 0.01833606 0.2003292 0.1569234 227.1339 272.971 0.6151441
##                    ACF1
## Training set 0.3938461
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## Mar 2022    0.046533666 -0.2434558 0.3365231 -0.3969669 0.4900343
## Apr 2022    0.147779500 -0.1422472 0.4378062 -0.2957781 0.5913371
## May 2022    0.022216321 -0.2678482 0.3122808 -0.4213990 0.4658317
## Jun 2022    0.043221817 -0.2468809 0.3333245 -0.4004520 0.4868956
## Jul 2022    0.146353777 -0.1437876 0.4364952 -0.2973792 0.5900868
## Aug 2022    0.001987288 -0.2881933 0.2921678 -0.4418056 0.4457802
## Sep 2022    0.056027979 -0.2341922 0.3462482 -0.3878255 0.4998815
## Oct 2022    0.139036597 -0.1512237 0.4292969 -0.3048782 0.5829514
## Nov 2022    0.055824323 -0.2344766 0.3461252 -0.3881526 0.4998012
## Dec 2022    0.099309545 -0.1910324 0.3896515 -0.3447301 0.5433492
## Jan 2023    0.007734848 -0.2826486 0.2981183 -0.4363684 0.4518381
## Feb 2023   -0.036907036 -0.3273326 0.2535185 -0.4810746 0.4072605
## Mar 2023    0.050579509 -0.2400214 0.3411805 -0.3938563 0.4950153
## Apr 2023    0.151825343 -0.1388186 0.4424693 -0.2926762 0.5963269
## May 2023    0.026262165 -0.2644253 0.3169496 -0.4183059 0.4708302
## Jun 2023    0.047267660 -0.2434638 0.3379991 -0.3973677 0.4919030
## Jul 2023    0.150399620 -0.1403763 0.4411755 -0.2943038 0.5951030
## Aug 2023    0.006033131 -0.2847878 0.2968540 -0.4387391 0.4508053
## Sep 2023    0.060073822 -0.2307926 0.3509402 -0.3847679 0.5049156
## Oct 2023    0.143082441 -0.1478300 0.4339948 -0.3018297 0.5879946
## Nov 2023    0.059870167 -0.2310887 0.3508291 -0.3851131 0.5048534
## Dec 2023    0.103355389 -0.1876505 0.3943613 -0.3416998 0.5484106
## Jan 2024    0.011780692 -0.2792728 0.3028342 -0.4333472 0.4569086
## Feb 2024   -0.032861193 -0.3239627 0.2582403 -0.4780626 0.4123402
```

INTERPRET

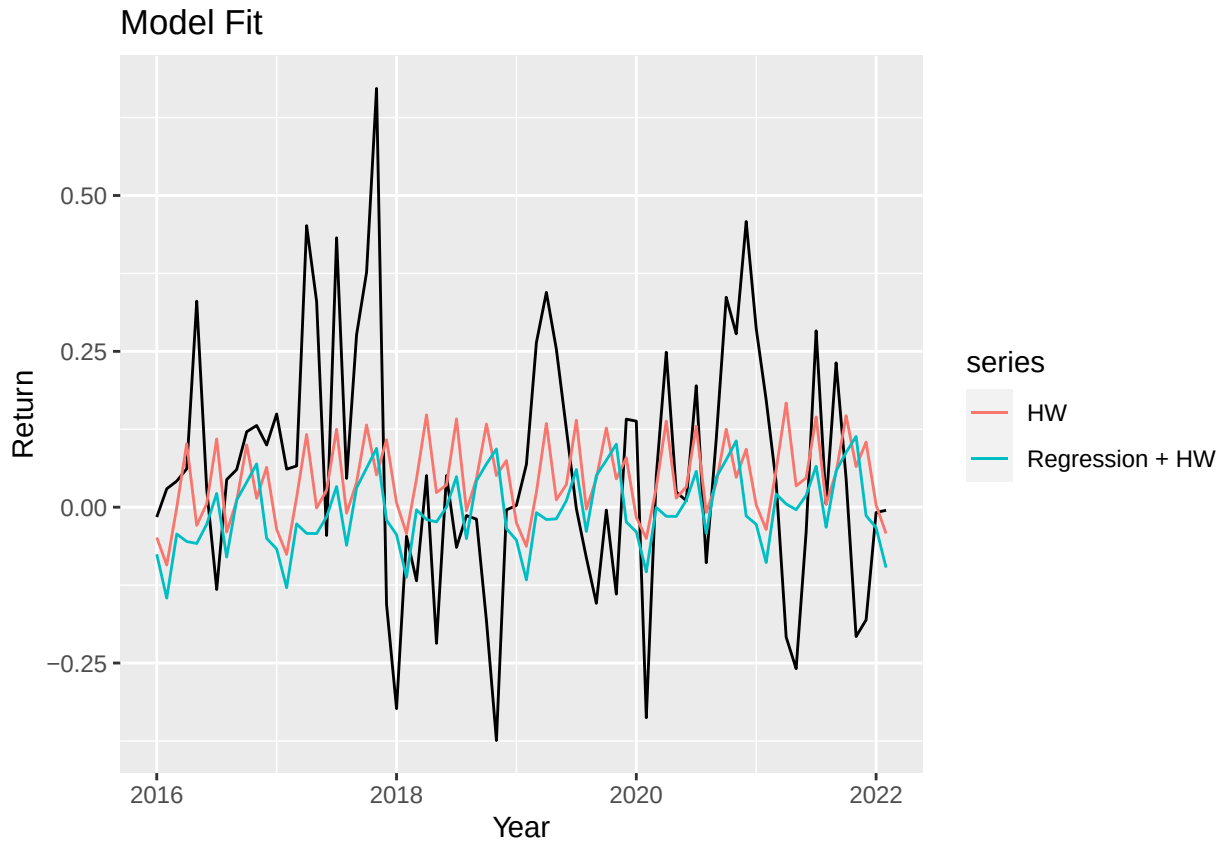
```
reg_hw_btc <- hw(btc_lm_res_ts,seasonal=c("additive"))
summary(reg_ets_btc)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = btc_lm_res_ts)
##
```

```
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0
##
##   sigma:  0.167
##
##      AIC     AICc      BIC
## 57.57386 57.91671 64.48605
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 9.001839e-06 0.1647114 0.1272337 100.0134 100.0134 0.6514489
##                    ACF1
## Training set 0.3391001
```

INTERPRET

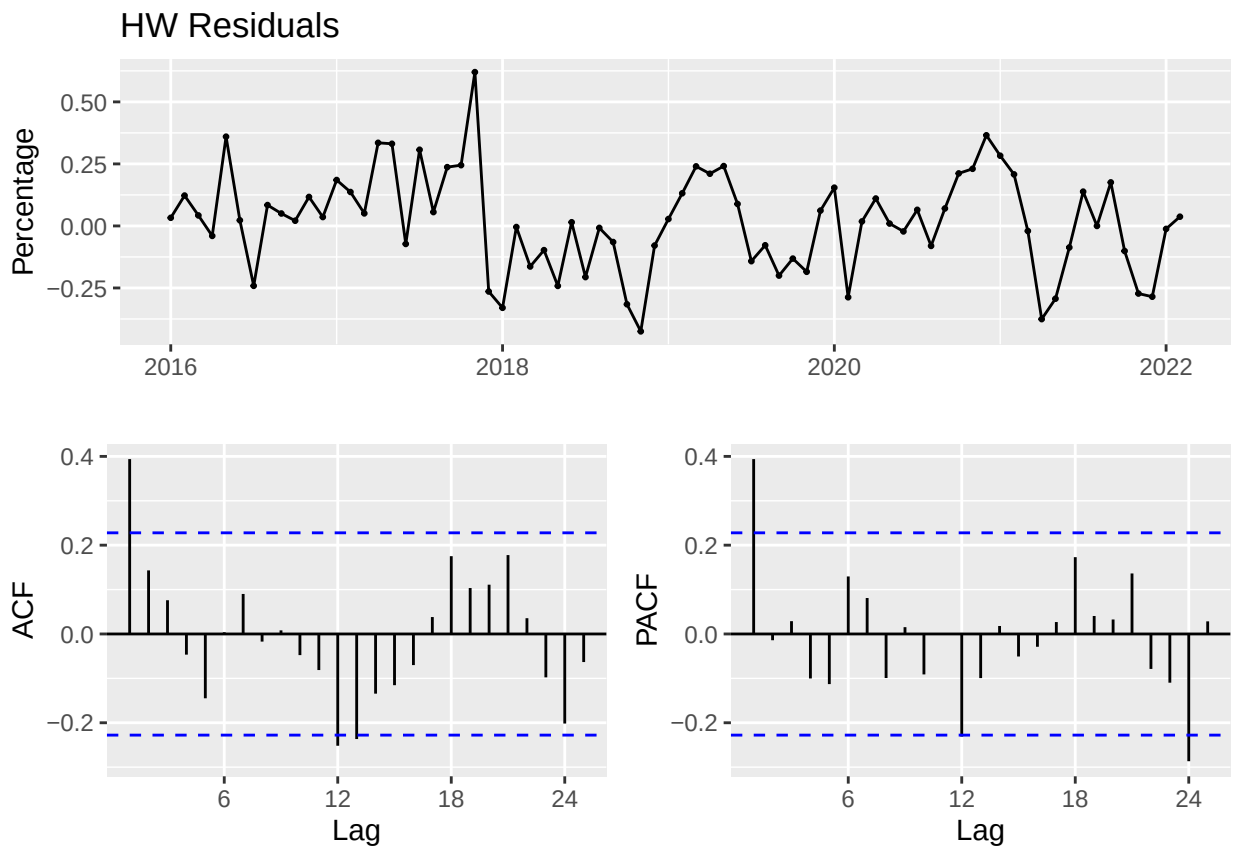
```
autoplot(btc_ts2, series="BTC Series", col="black") + ggtitle("Model Fit") +
  autolayer(hw_btc$fitted, series="HW") +
  autolayer(reg_hw_btc$fitted, series="Regression + HW") +
  ylab("Return") + xlab("Year")
```



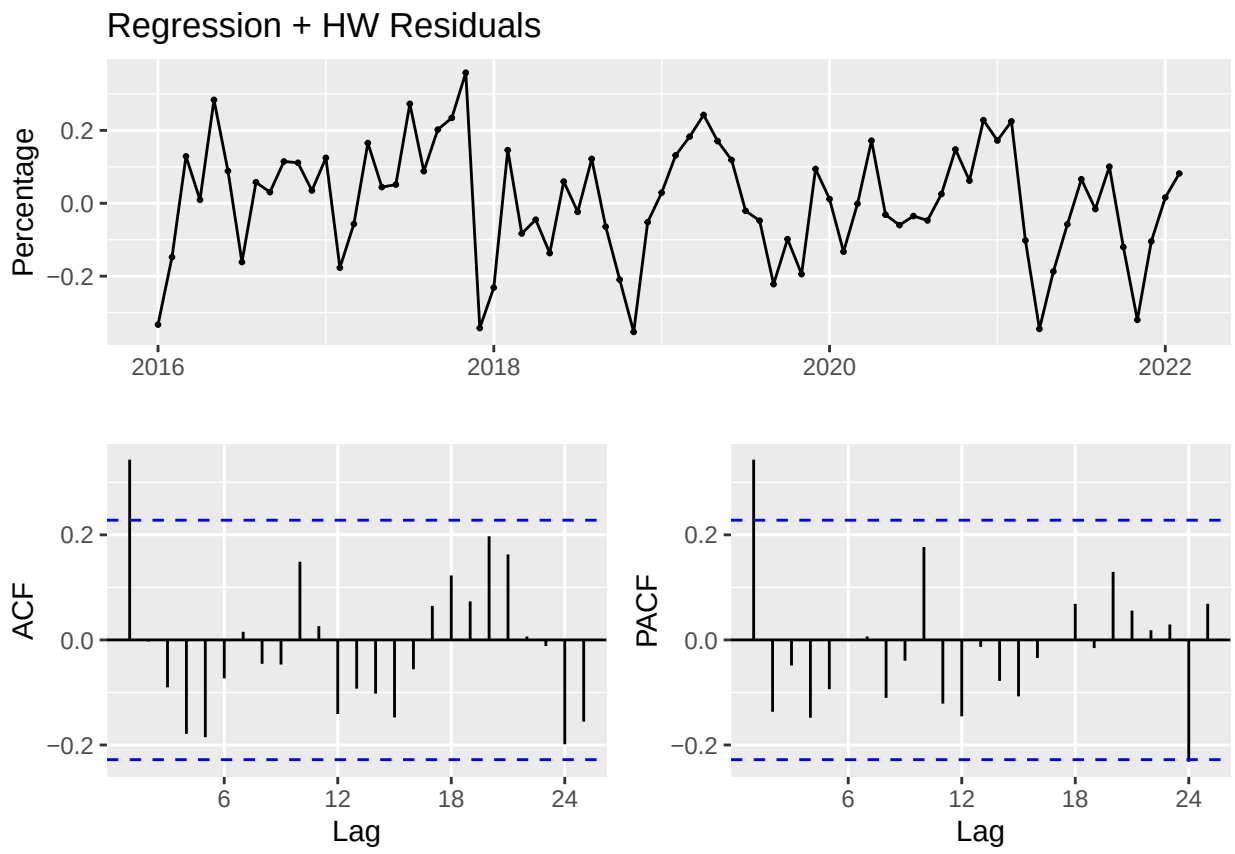
ACF/PACF of Residuals

INTERPRET

```
ggtsdisplay(resid(hw_btc),main = "HW Residuals", ylab = "Percentage")
```

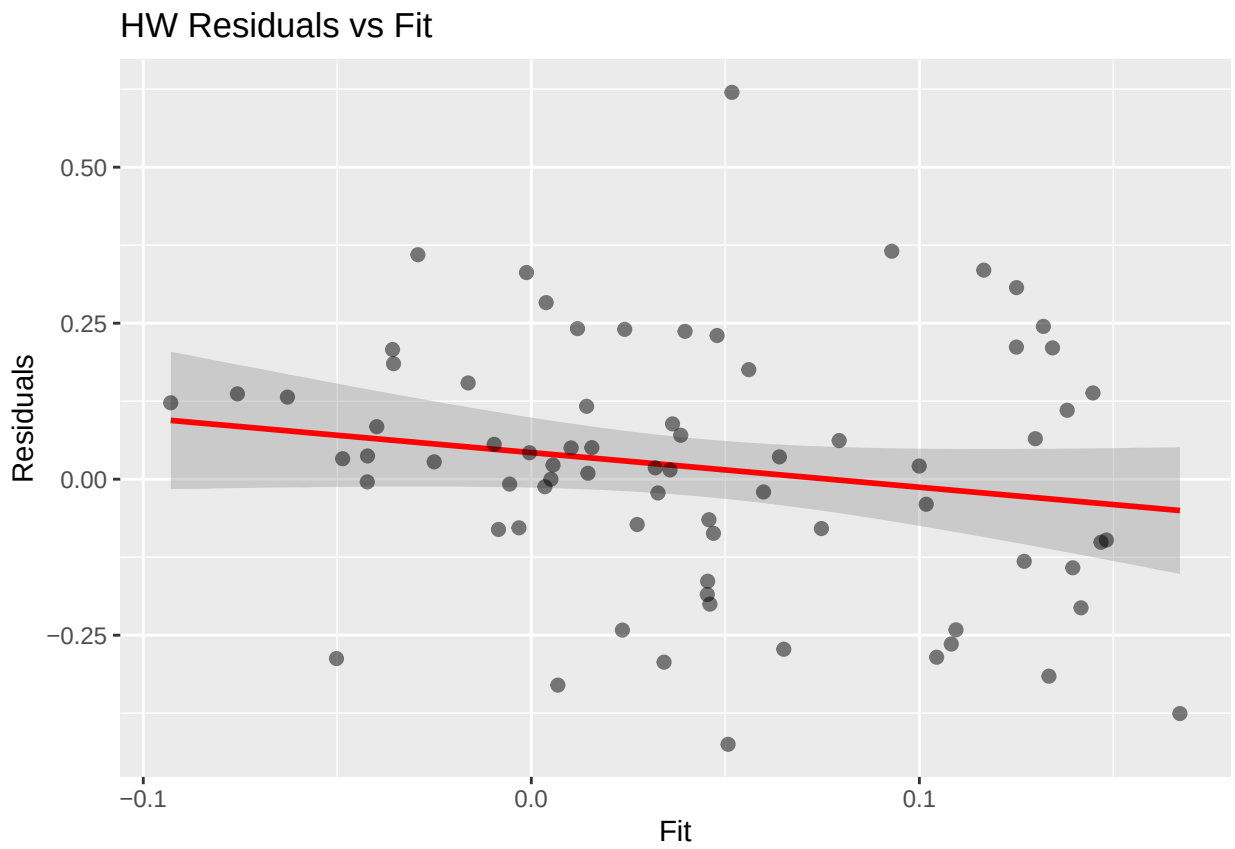



```
ggtsdisplay(resid(reg_hw_btc),main = "Regression + HW Residuals", ylab = "Percentage")
```

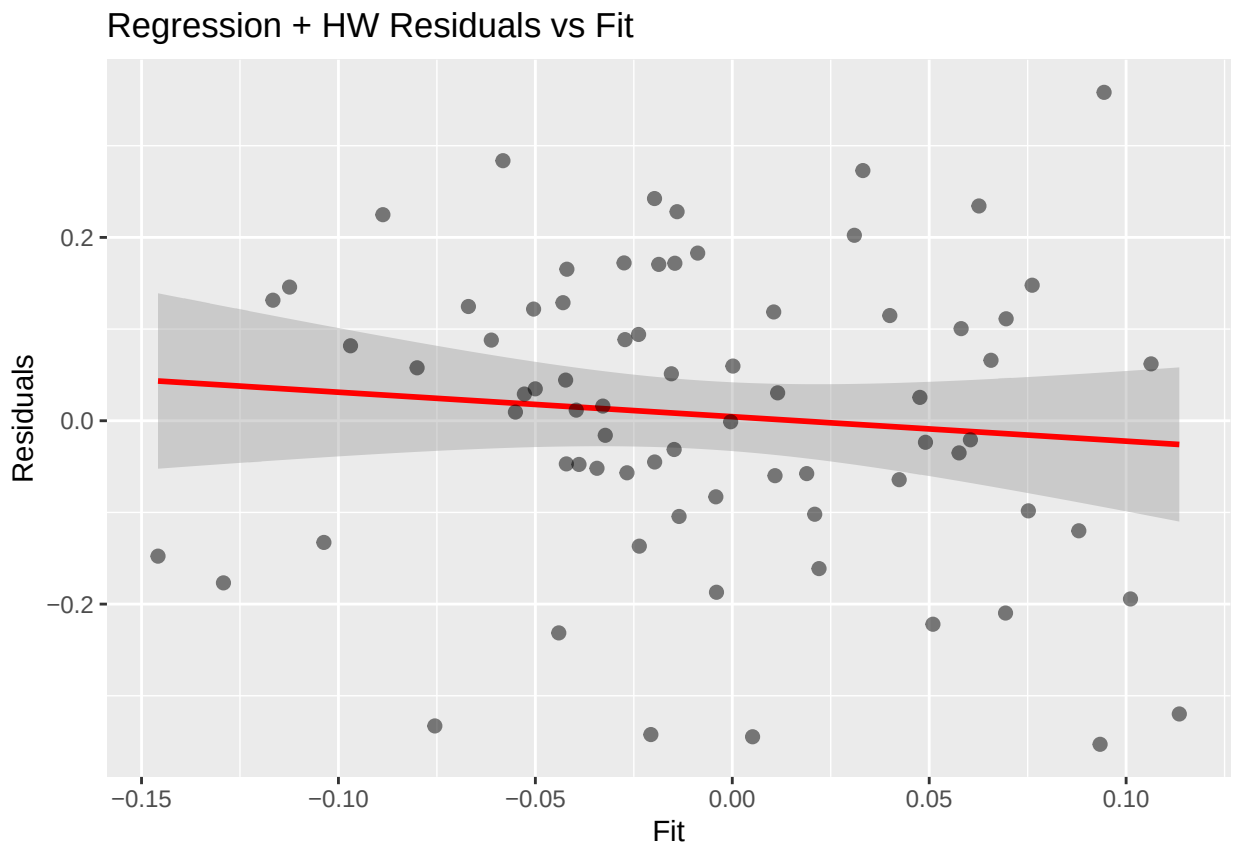


INTERPRET

```
plot_resid_fit(fitted(hw_btc),resid(hw_btc),"HW")
```



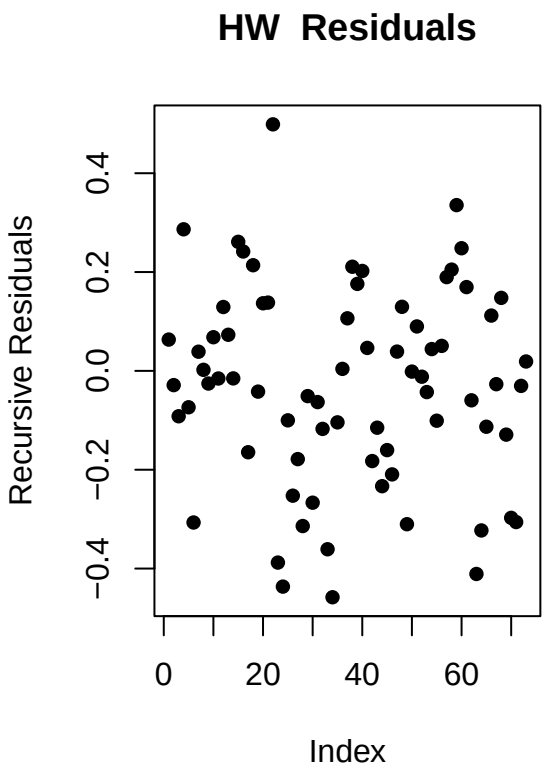

```
plot_resid_fit(fitted(reg_hw_btc),resid(reg_hw_btc),"Regression + HW")
```

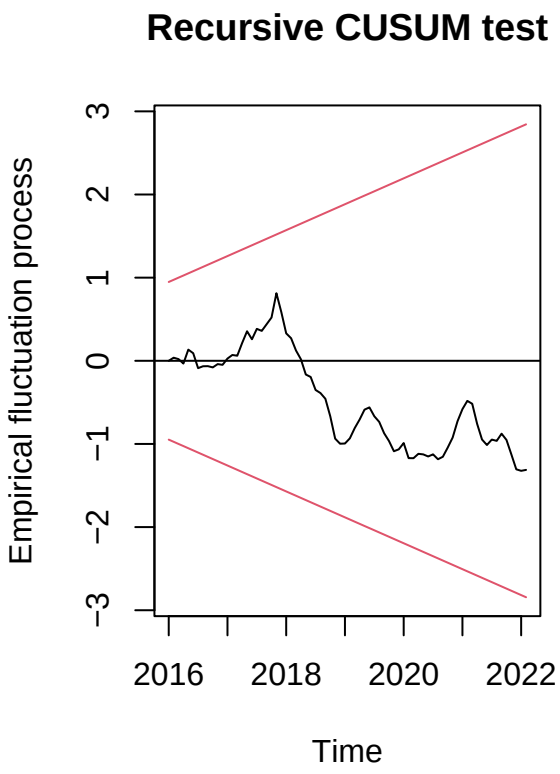


Recursive Residuals

INTERPRET

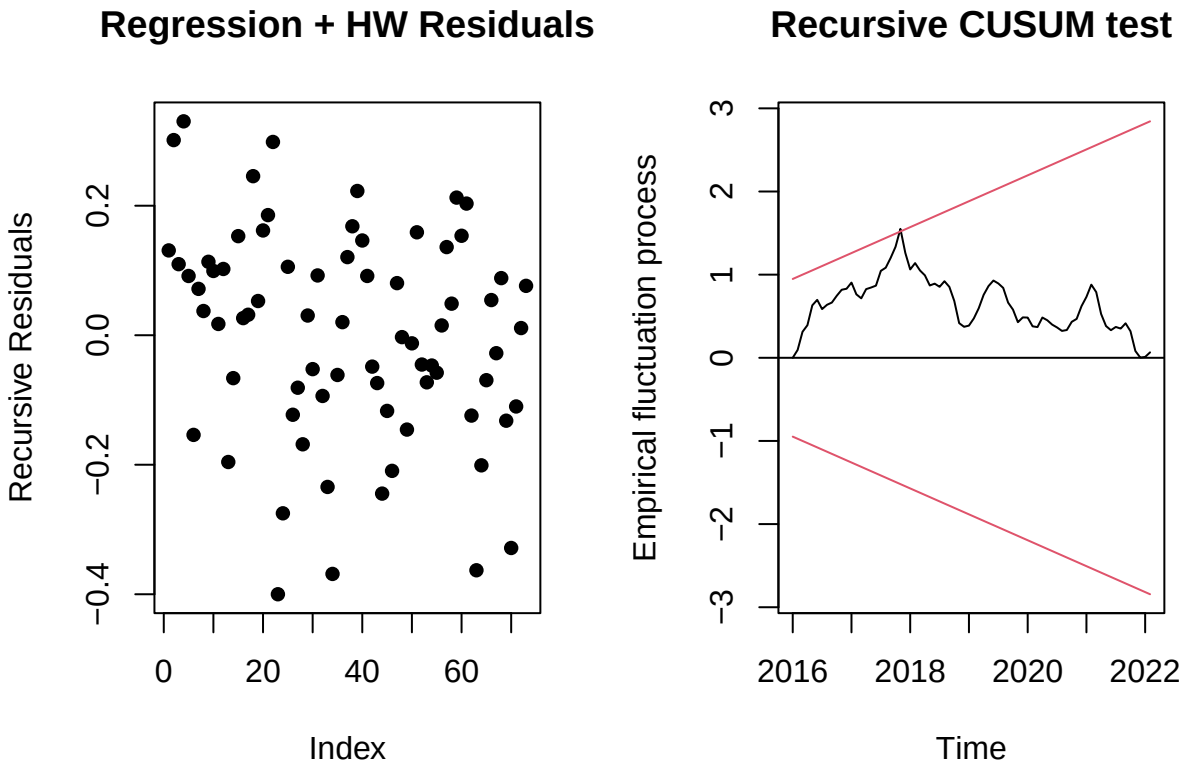
```
par(mfrow=c(1,2))
y = recresid(hw_btc$res ~ 1)
plot(y, pch = 16, main = "HW  Residuals", ylab = "Recursive Residuals")
plot(efp(hw_btc$residuals~1, type = "Rec-CUSUM"))
```





INTERPRET

```
par(mfrow=c(1,2))
y = recresid(reg_hw_btc$res ~ 1)
plot(y, pch = 16, main = "Regression + HW Residuals", ylab = "Recursive Residuals")
plot(efp(reg_hw_btc$residuals~1, type = "Rec-CUSUM"))
```

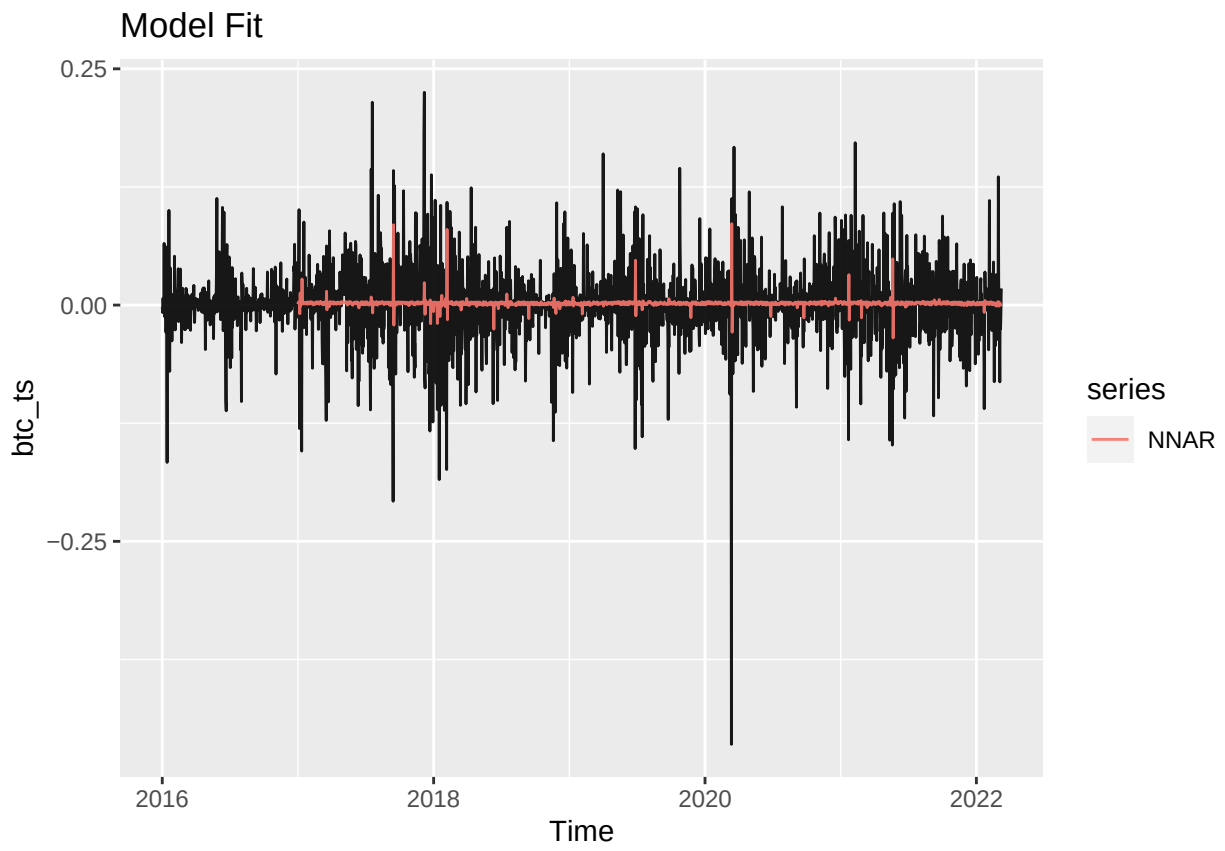


G) NNETAR

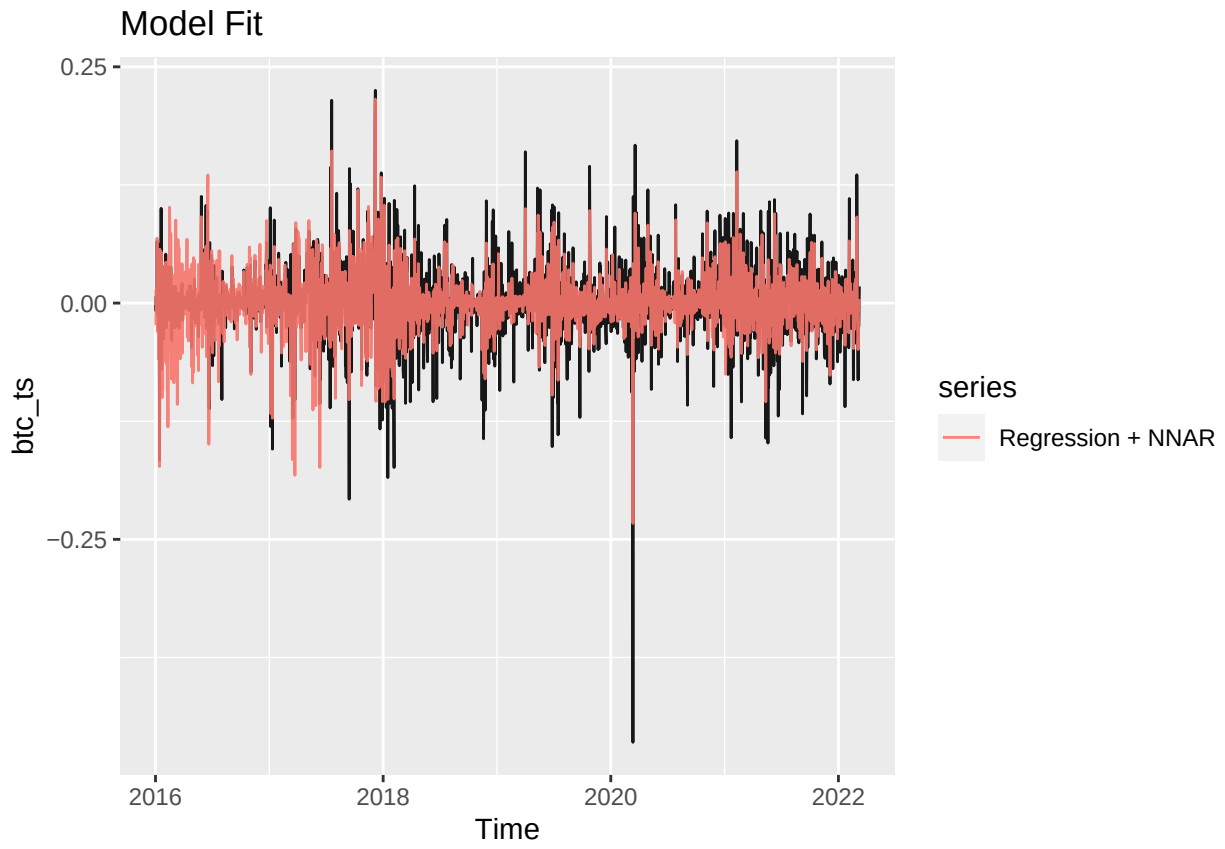
Fitting the Model

We will fit a NNETAR model to BTC Returns, as well as a NNETAR model consisting of a linear model of BTC Returns regressed on ETH Returns and a NNETAR model of the residuals. Judging from the plot, the Linear Model + NNETAR Model appears to outperform the simple NNETAR model. We will now look at the residuals from each in order to choose a model.

```
nnar_btc <- nnetar(btc_ts)
btc_eth_lm <- lm(returns$btc~returns$eth_ts)
reg_nnar_btc <- nnetar(btc_eth_lm$residuals, xreg=cbind(returns$btc_ts,returns$eth_ts))

nnar_fit <- ts(nnar_btc$fitted,frequency = 365, start = c(2016,1))
reg_nnar_fit <- ts(reg_nnar_btc$fitted,frequency = 365, start = c(2016,1))

autoplot(btc_ts, series="BTC Returns", color="black", alpha = 0.9) +
  autolayer(nnar_fit, series="NNAR", alpha = 0.9) + ggtitle("Model Fit")
```



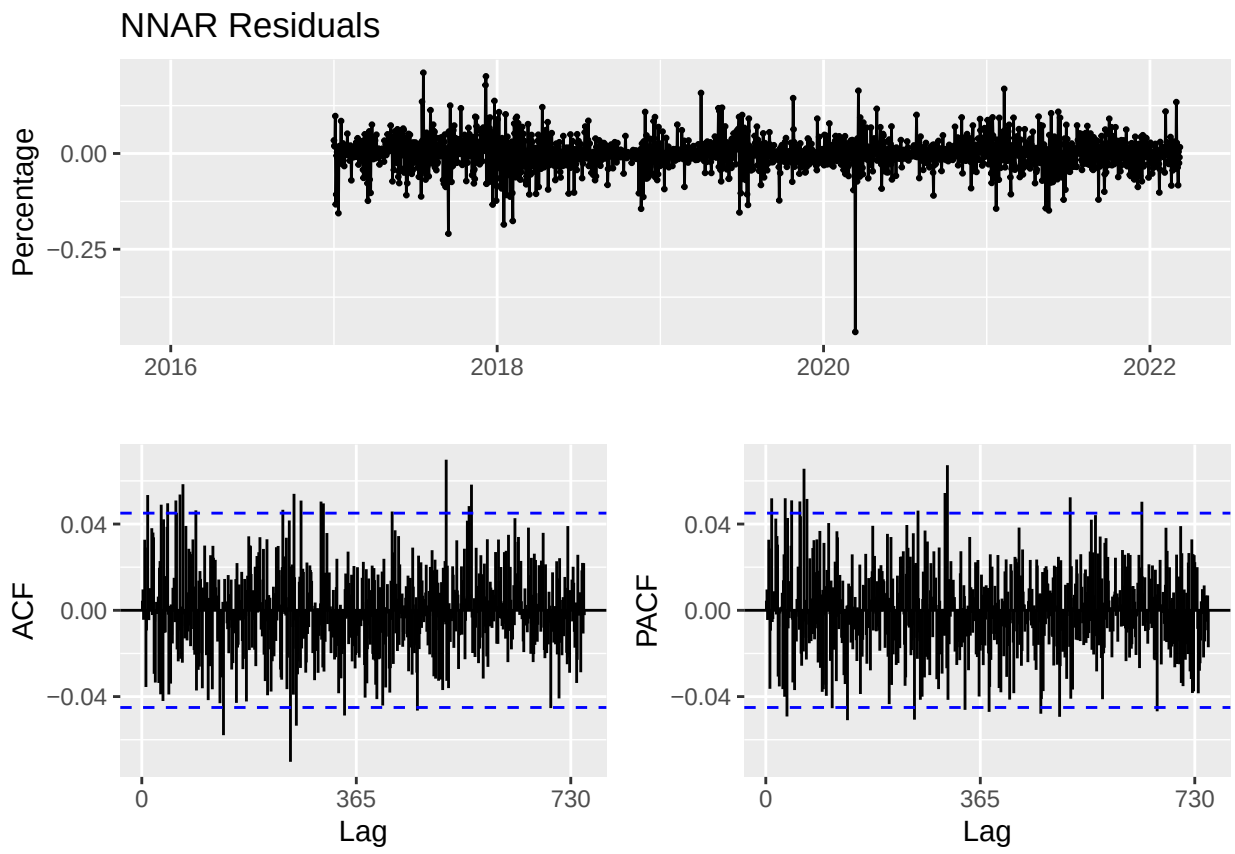

```
autoplot(btc_ts, series="BTC Returns", color="black", alpha = 0.9) +
  autolayer(reg_nnar_fit, series="Regression + NNAR", alpha = 0.9) +
  ggtitle("Model Fit")
```



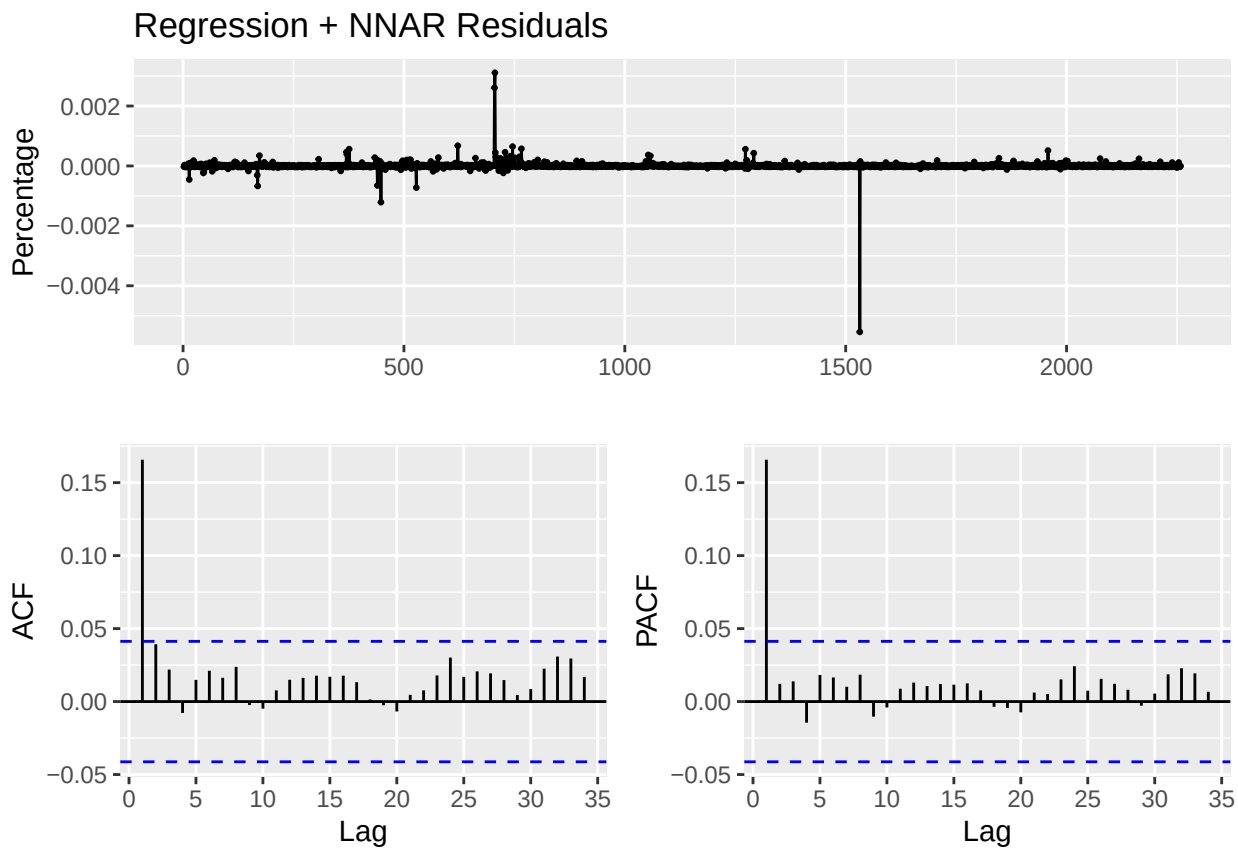
ACF/PACF of Residuals

The residuals for both models appear to be white noise and normally distributed, but the magnitude of the lags for the Linear Model + NNETAR model are significantly lower and display less dynamics. Therefore, we will create our forecasts with the Linear Model + NNETAR Model.

```
ggtsdisplay(resid(nnar_btc),main = "NNAR Residuals", ylab = "Percentage")
```

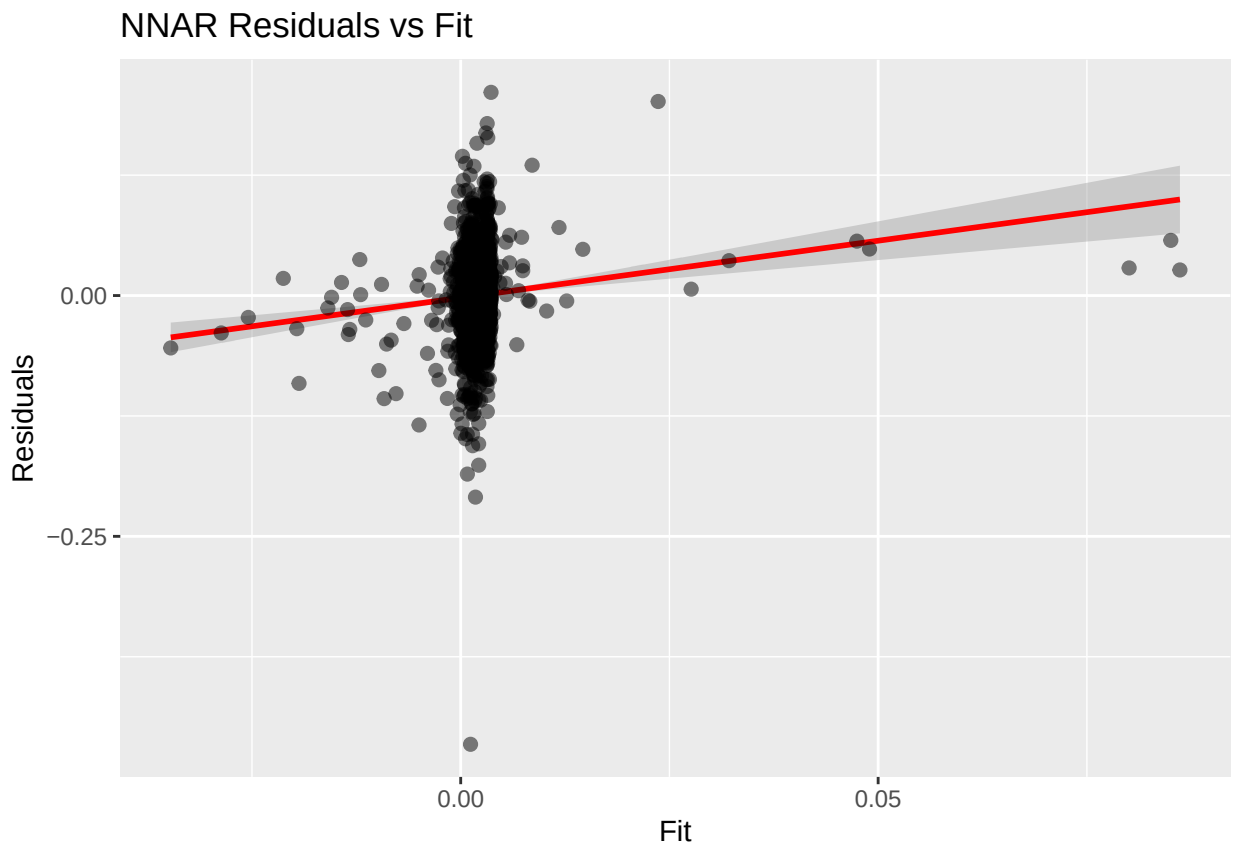



```
ggtsdisplay(resid(reg_nnar_btc),main = "Regression + NNAR Residuals", ylab = "Percentage")
```

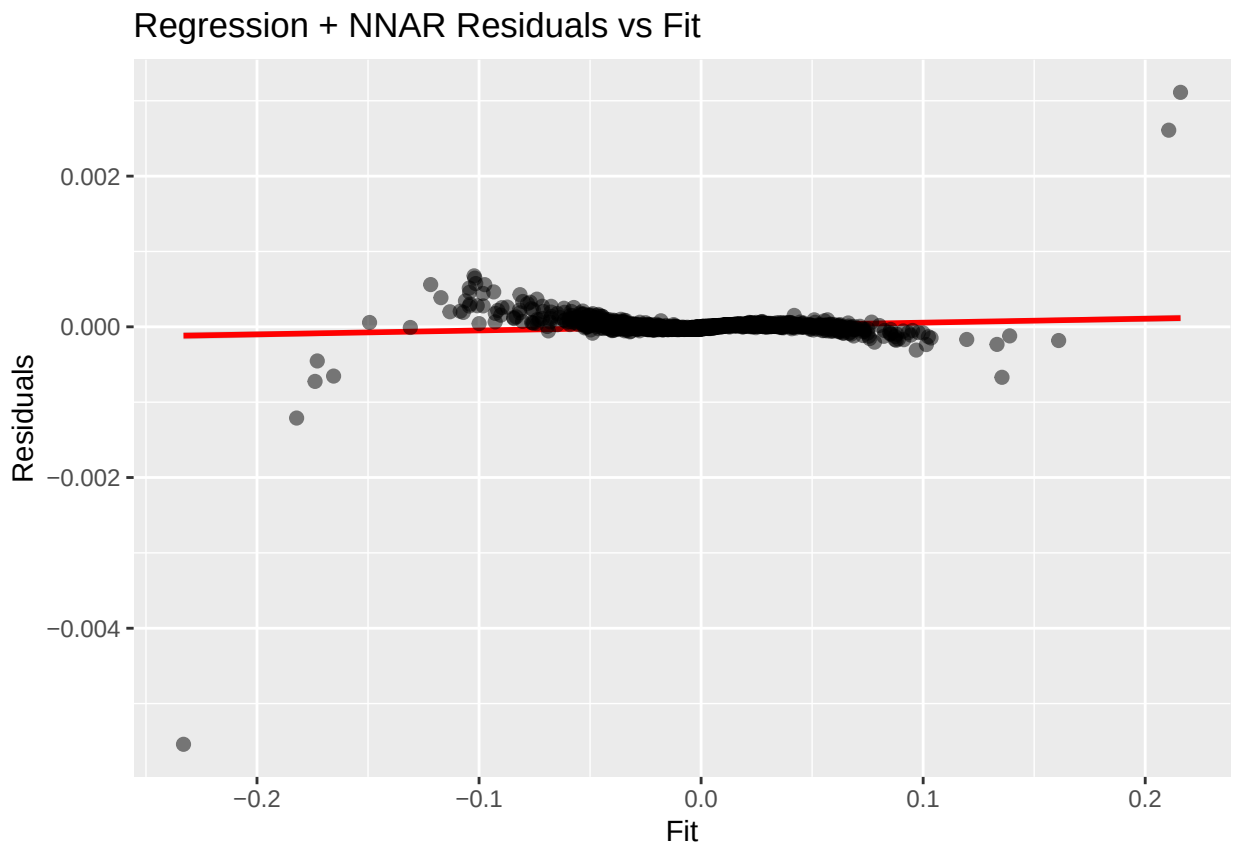


INTERPRET

```
plot_resid_fit(fitted(nnar_btc),resid(nnar_btc),"NNAR")
```




```
plot_resid_fit(fitted(reg_nnar_btc),resid(reg_nnar_btc),"Regression + NNAR")
```

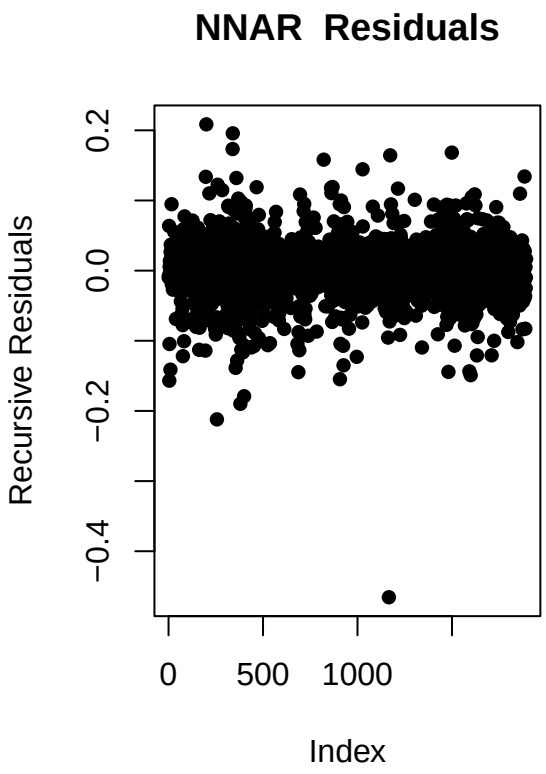


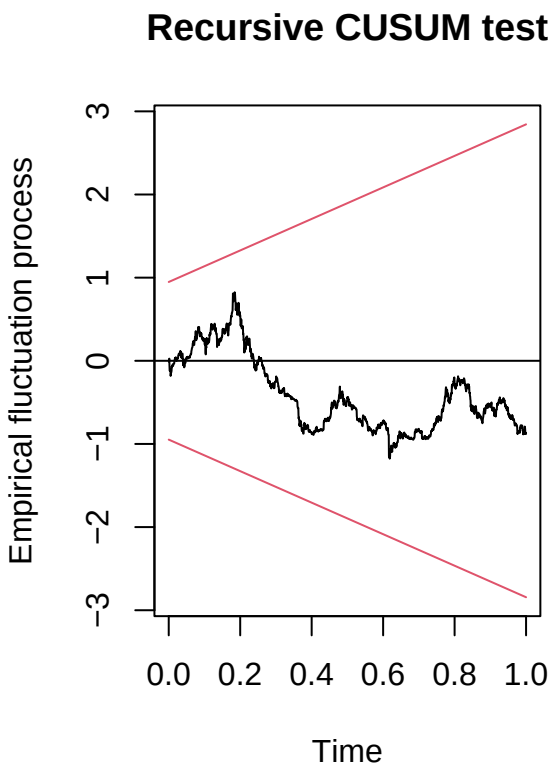
Recursive Residuals

INTERPRET

The recursive residuals from the Regression + NNETAR model reaffirm that this model is a good fit. Thus, we will continue and forecast with this fit.

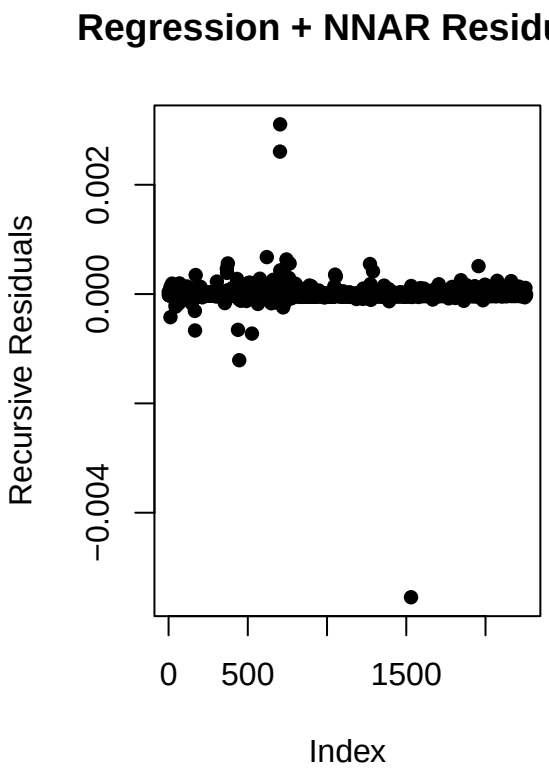
```
par(mfrow=c(1,2))
y = recresid(nnar_btc$res ~ 1)
plot(y, pch = 16, main = "NNAR  Residuals", ylab = "Recursive Residuals")
plot(efp(nnar_btc$residuals~1, type = "Rec-CUSUM"))
```

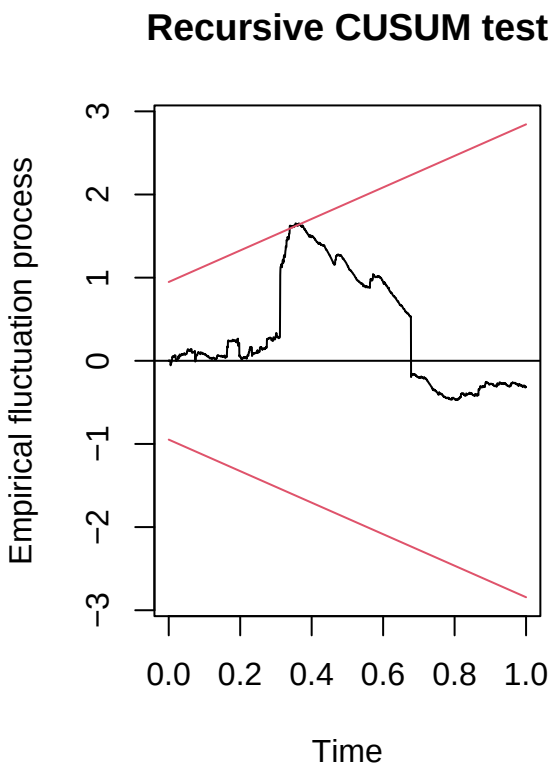




INTERPRET

```
par(mfrow=c(1,2))
y = recresid(reg_nnar_btc$res ~ 1)
plot(y, pch = 16, main = "Regression + NNAR Residuals", ylab = "Recursive Residuals")
plot(efp(reg_nnar_btc$residuals~1, type = "Rec-CUSUM"))
```



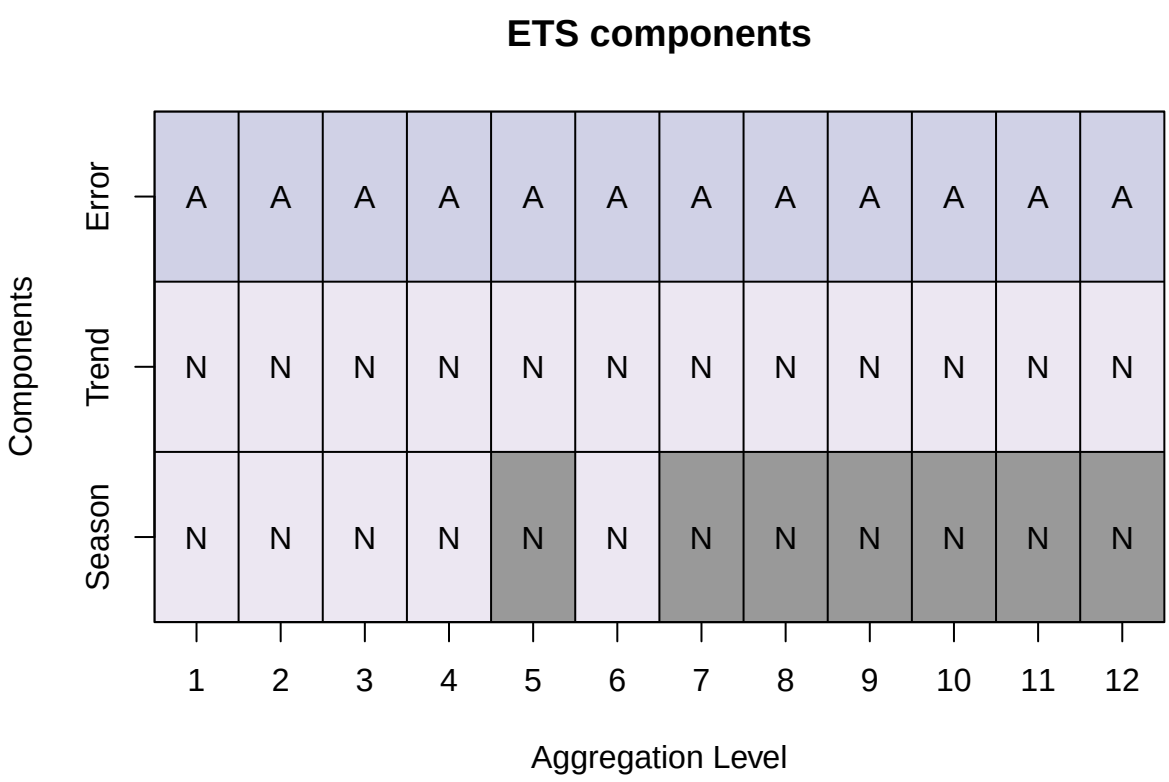


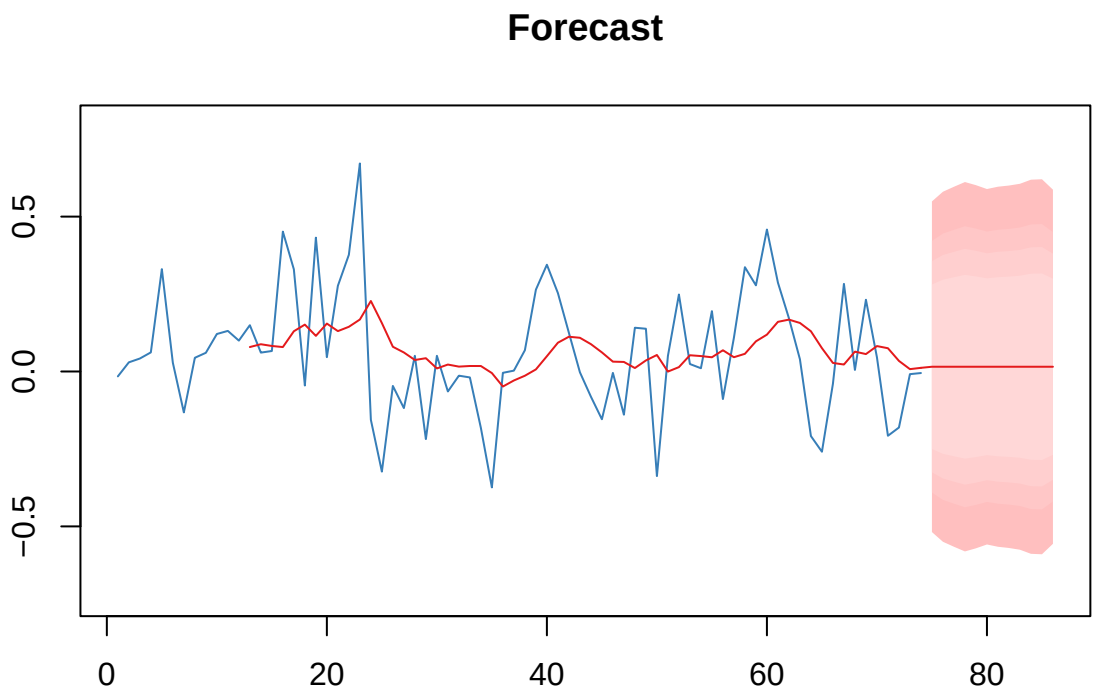
H) MAPA

```
mapa_fit <- mapaest(btc_ts2,outplot=2, paral=2,type="es")
```

```
## Running with 12 cores
```

```
mapa_for <- mapa(btc_ts2,conf.lvl=c(0.8,0.9,0.95,0.99),outplot=1)
```

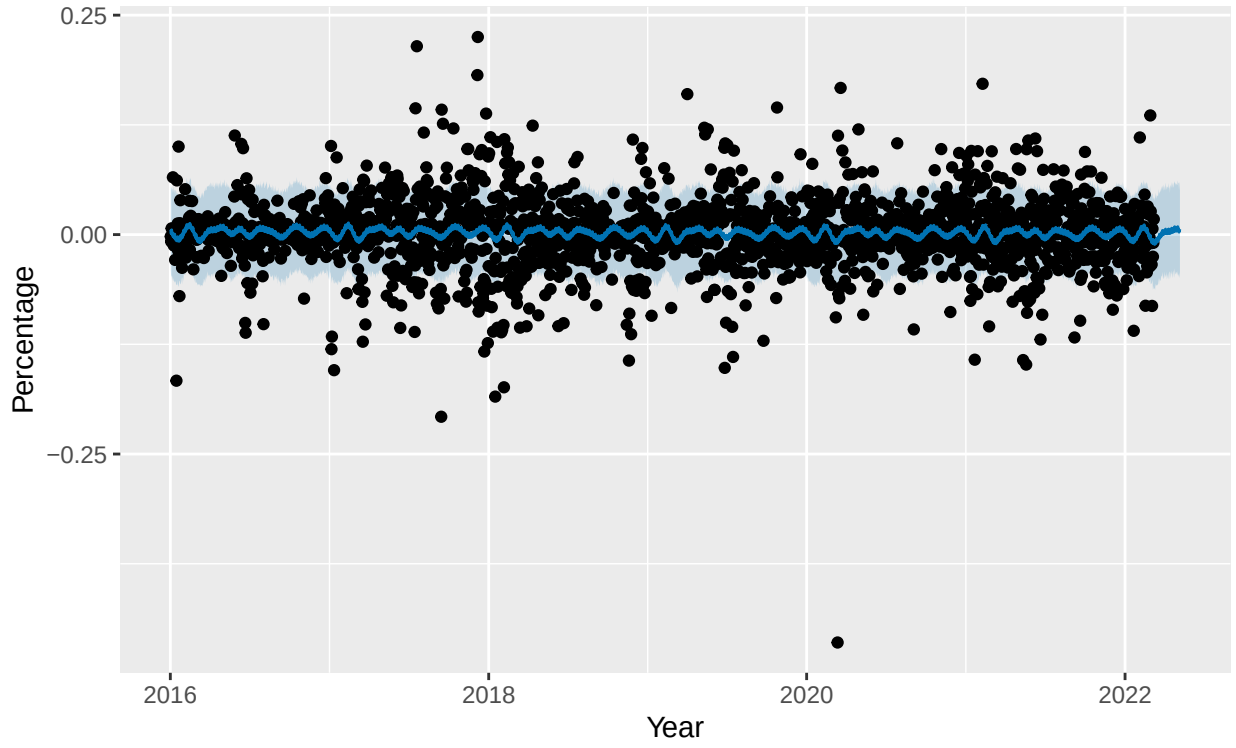

ETS components
Components
Error
Trend
Season
A A A A A A A A A A A A
N N N N N N N N N N N N
N N N N N N N N N N N N
1 2 3 4 5 6 7 8 9 10 11 12
Aggregation Level



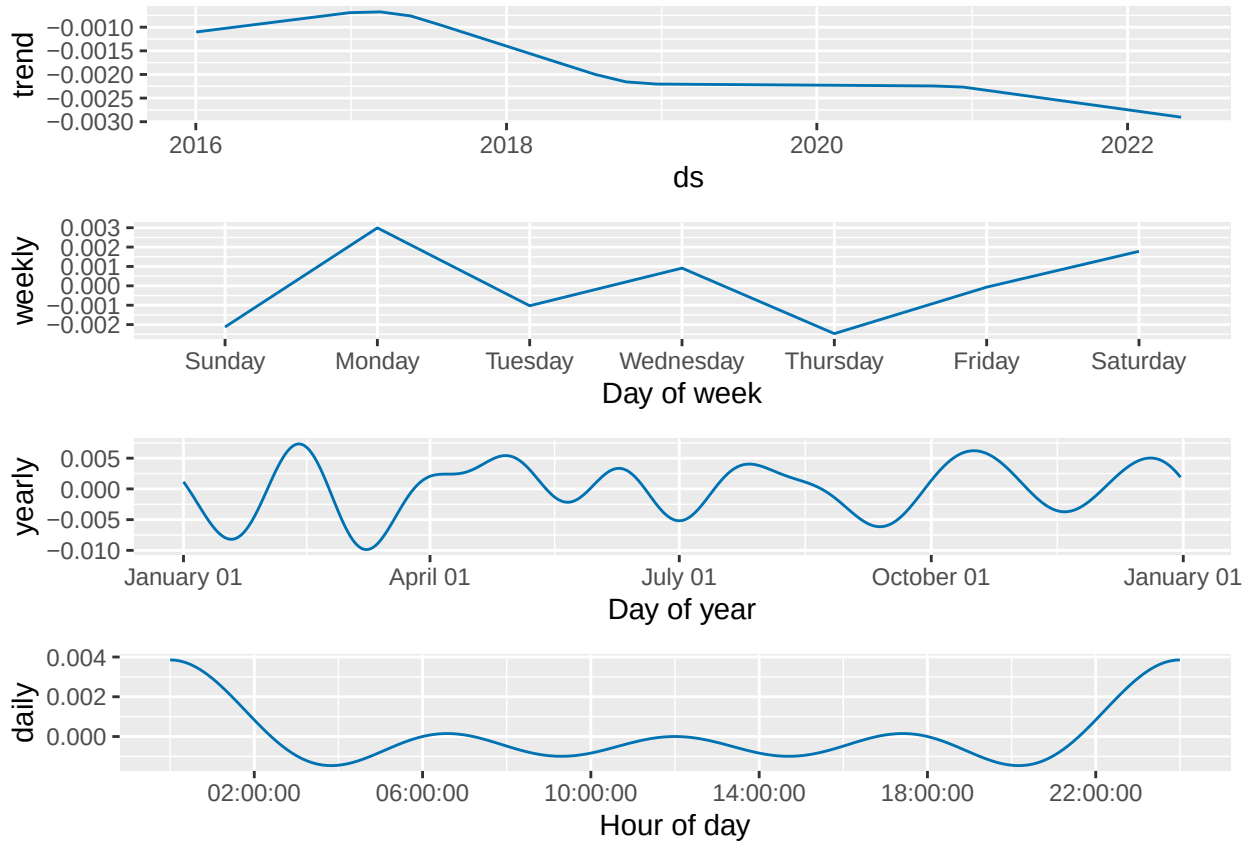
I) Prophet

Fitting the Model

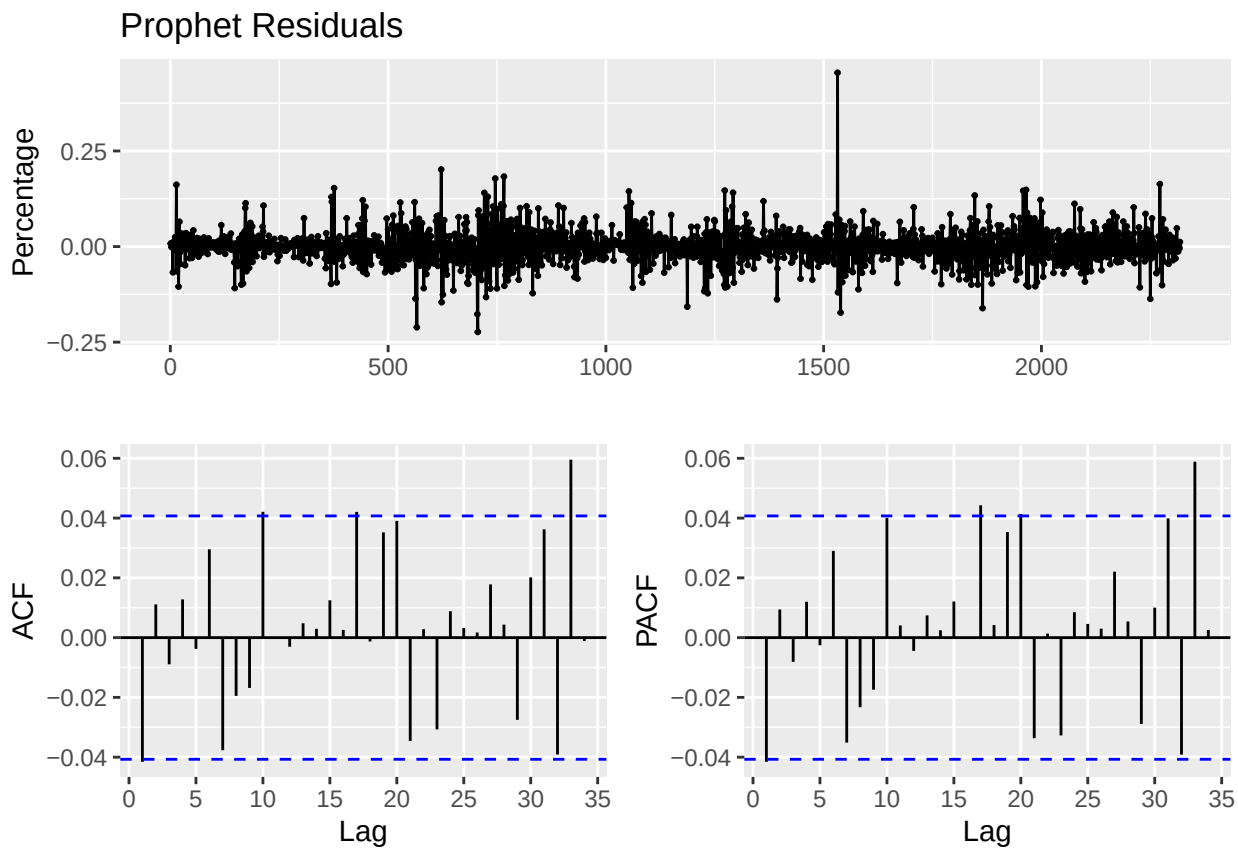
```
df_r_btc <- data.frame(ds = btc$timestamp, y = as.numeric(btc$returns))
preds_btc <- prophet::prophet(df_r_btc,daily.seasonality=TRUE)
future_btc <- prophet::make_future_dataframe(preds_btc, periods = 60)
forecast_btc <- predict(preds_btc, future_btc)
plot(preds_btc, forecast_btc, main="BTC Returns",
     ylab = "Percentage", xlab = "Year")
```



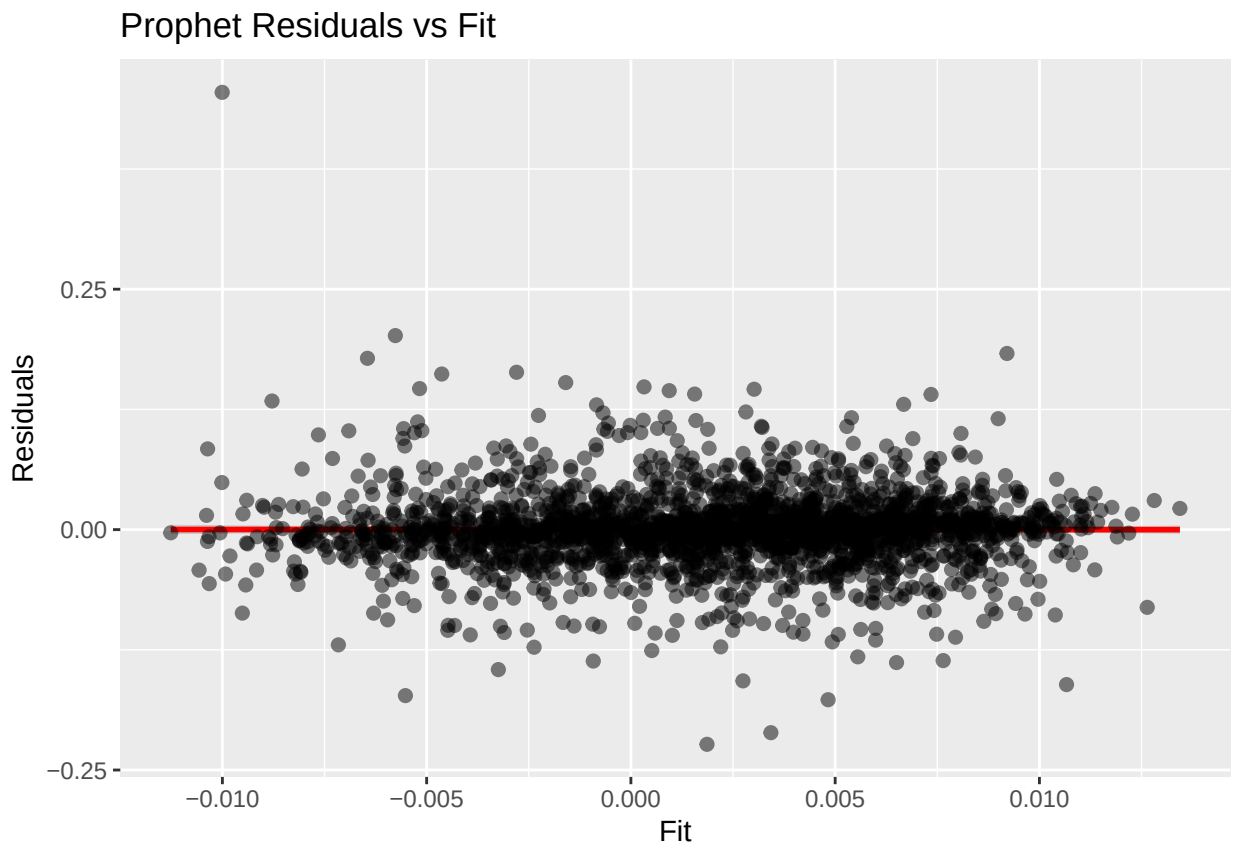

```
prophet_plot_components(preds_btc, forecast_btc)
```



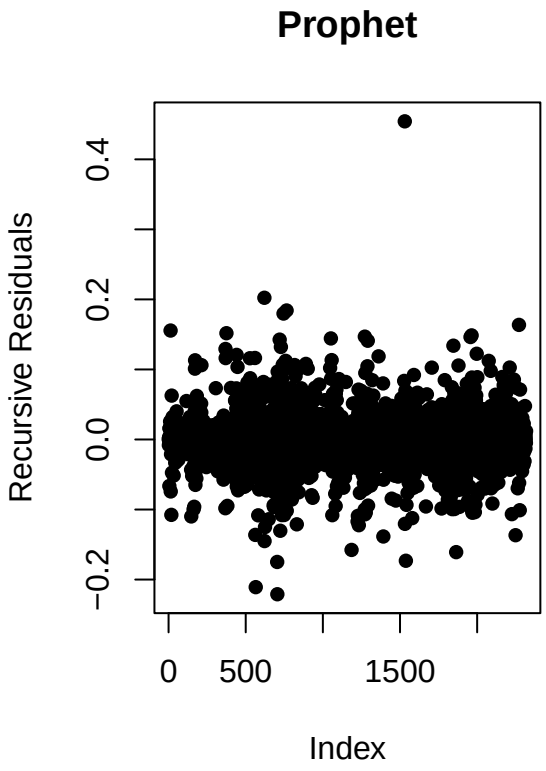

```
res <- forecast_btc$yhat - df_r_btc$y
ggtsdisplay(res, main = "Prophet Residuals", ylab = "Percentage")
```

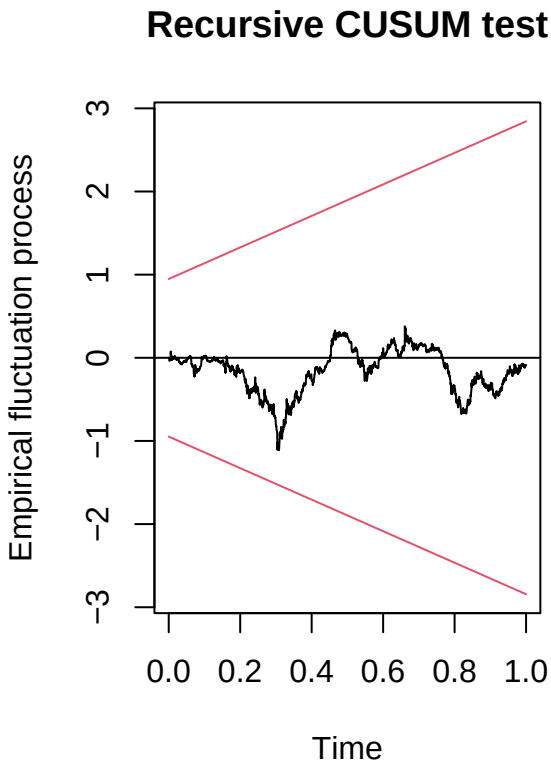



```
plot_resid_fit(forecast_btc$yhat,res,"Prophet")
```




```
par(mfrow=c(1,2))
y = recresid(res ~ 1)
plot(y, pch = 16, main = "Prophet", ylab = "Recursive Residuals")
plot(efp(res~1, type = "Rec-CUSUM"))
```





J) Train & Test Accuracy

To test both the high frequency (arima/nnetar/prophet) and lower frequency models, we use training sets. We will predic the entire year of 2021 for both frequencies. With the high frequency models we will do a 365-step forecast while with the lower freuquency model we will do a 12-step forecast. We will use MAPE to compare our model accuracy.

```
train <- window(btc_ts, end=c(2020,365))
train_eth <- window(eth_ts, end=c(2020,365))
train_xrp <- window(xrp_ts, end=c(2020,365))
test <- window(btc_ts, start=c(2021,1), end=c(2021,365))

train2 <- window(btc_ts2, end=c(2020,12))
train_eth2 <- window(eth_ts2, end=c(2020,12))
train_xrp2 <- window(xrp_ts2, end=c(2020,12))
test2 <- window(btc_ts2, start=c(2021,1), end=c(2021,12))

btc_train <- as_tibble(btc) %>% filter(timestamp < "2021-01-01")
eth_train <- eth %>% filter(timestamp < "2021-01-01")
xrp_train <- xrp %>% filter(timestamp < "2021-01-01")
btc_test <- as_tibble(btc) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-01-01")
eth_test <- as_tibble(eth) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-01-01")
xrp_test <- as_tibble(xrp) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-01-01")

btc_train2 <- as_tibble(btc_monthly) %>% filter(timestamp < "2021-01-01")
```

```
eth_train2 <- eth_monthly %>% filter(timestamp < "2021-01-01")
xrp_train2 <- xrp_monthly %>% filter(timestamp < "2021-01-01")
btc_test2 <- as_tibble(btc_monthly) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-
eth_test2 <- as_tibble(eth_monthly) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-
xrp_test2 <- as_tibble(xrp_monthly) %>% filter(timestamp >= "2021-01-01") %>% filter(timestamp < "2022-

eth_train$returns <- c(NA,diff(log(eth_train$close)))
eth_test$returns <- c(NA,diff(log(eth_test$close)))
eth_train <- na.omit(eth_train)
eth_test <- na.omit(eth_test)

eth_train2$returns <- c(NA,diff(log(eth_train2$close)))
eth_test2$returns <- c(NA,diff(log(eth_test2$close)))
xrp_train2$returns <- c(NA,diff(log(xrp_train2$close)))
xrp_test2$returns <- c(NA,diff(log(xrp_test2$close)))

btc_train2 <- na.omit(btc_train2)
eth_train2 <- na.omit(eth_train2)
xrp_train2 <- na.omit(xrp_train2)

btc_test2 <- na.omit(btc_test2)
eth_test2 <- na.omit(eth_test2)
xrp_test2 <- na.omit(xrp_test2)

btc_train2 <- btc_train2[2:nrow(btc_train2),]

eth_monthly$returns <- c(NA,diff(log(eth_monthly$close)))
usdt_monthly$returns <- c(NA,diff(log(usdt_monthly$close)))
xrp_monthly$returns <- c(NA,diff(log(xrp_monthly$close)))

h <- length(test)
h2 <- length(test2)
```

For ARIMA, we get a MAPE of 11.59763.

```
ARIMA_mod <- Arima(btc_train[,"returns"], order=c(0,0,1), xreg=cbind(eth_train$returns,xrp_train$returns
ARIMA_for <- predict(ARIMA_mod, newxreg=cbind(eth_test$returns,xrp_test$returns))
ARIMA <- ts(ARIMA_for$pred, start=c(2021,1), freq=365)
MAPE(ARIMA, btc_test$returns)
```

```
## [1] 11.59763
```

For ETS, we get a MAPE of 0.9792409.

```
ETS_mod <- ets(train2)
ETS_for <- predict(ETS_mod, h=h2)
ETS <- ts(ETS_for$mean, start=c(2021,1), freq=12)
MAPE(ETS, test2)
```

```
## [1] 0.9792409
```

For HW, we get a MAPE of 4.456509.

```
btc_eth_lm <- lm(btc_train2$returns~eth_train2$returns)
btc_lm_res_ts <- ts(btc_eth_lm$residuals,freq=12,start=c(2016,2))
HW_mod <- hw(btc_lm_res_ts,seasonal=c("additive"))
HW_for <- predict(HW_mod, h=h2)
HW <- ts(HW_for$mean, start=c(2021,1), freq=12)
MAPE(HW, test2)
```

```
## [1] 4.456509
```

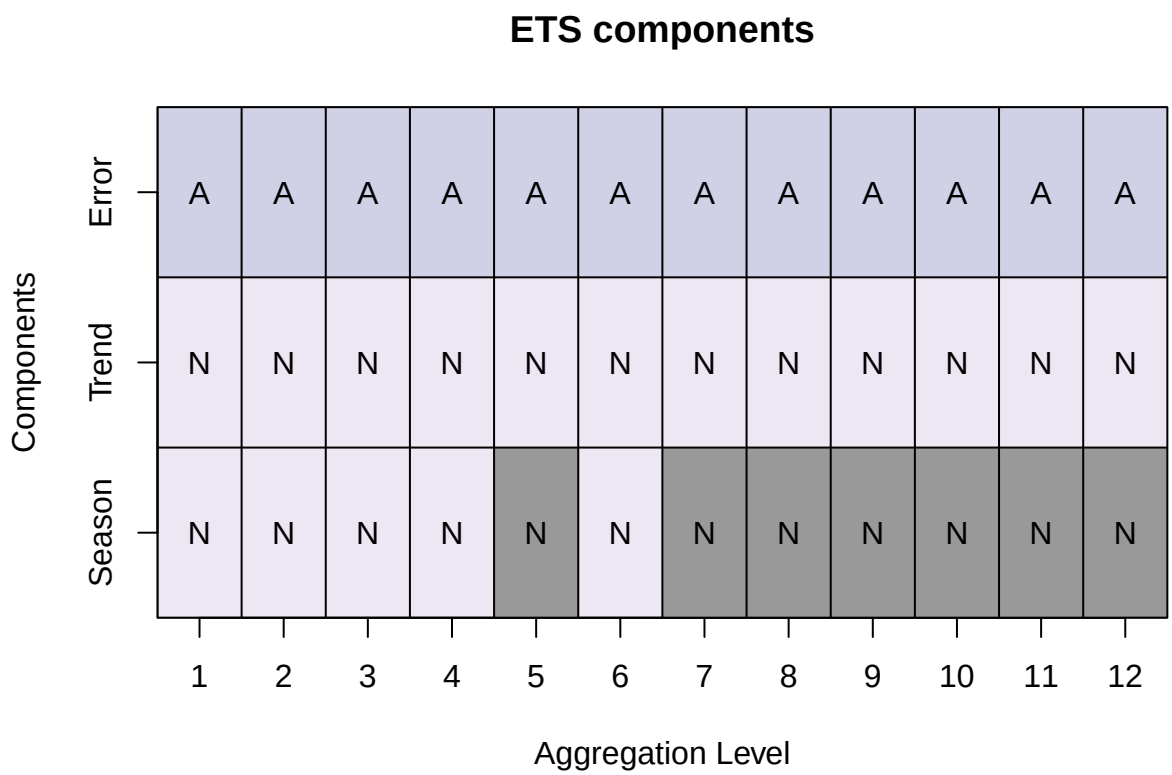
For NNETAR, we get a MAPE of 12.89427.

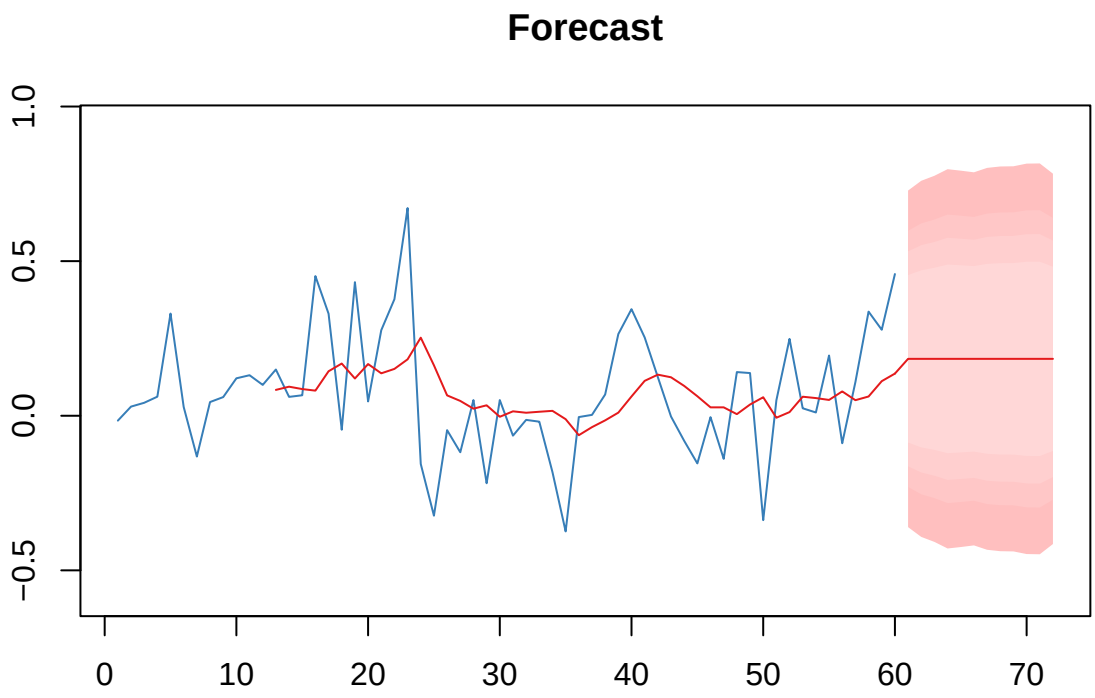
```
btc_train <- btc_train[2:nrow(btc_train),]
btc_eth_lm <- lm(btc_train$returns~eth_train$returns)
NNAR_mod <- nnetar(btc_eth_lm$residuals, xreg=cbind(btc_train$returns,eth_train$returns))
NNAR_for <- predict(NNAR_mod, xreg=cbind(btc_test$returns,eth_test$returns))
NNAR <- ts(NNAR_for$mean, start=c(2021,1), freq=365)
MAPE(NNAR, test)
```

```
## [1] 11.56675
```

For MAPA, we get a MAPE of 1.149696.

```
MAPA_for <- mapa(train2,conf.lvl=c(0.8,0.9,0.95,0.99),outplot=1)
```






```
MAPA <- ts(MAPA_for$outfor, start=c(2021,1), frequency=12)
MAPE(MAPA, test2)
```

```
## [1] 1.149696
```

For Prophet, we get a MAPE of 39.98843.

```
df_r_btc <- data.frame(ds = btc_train$timestamp, y = as.numeric(btc_train$returns))
preds_btc <- prophet::prophet(df_r_btc,daily.seasonality=TRUE)
future_btc <- prophet::make_future_dataframe(preds_btc, periods = 365)
pred_btc <- predict(preds_btc, future_btc)
PRO_for <- pred_btc[1826:nrow(pred_btc),]
PRO <- ts(PRO_for$yhat, start=c(2021,1), freq=365)
MAPE(PRO, test)
```

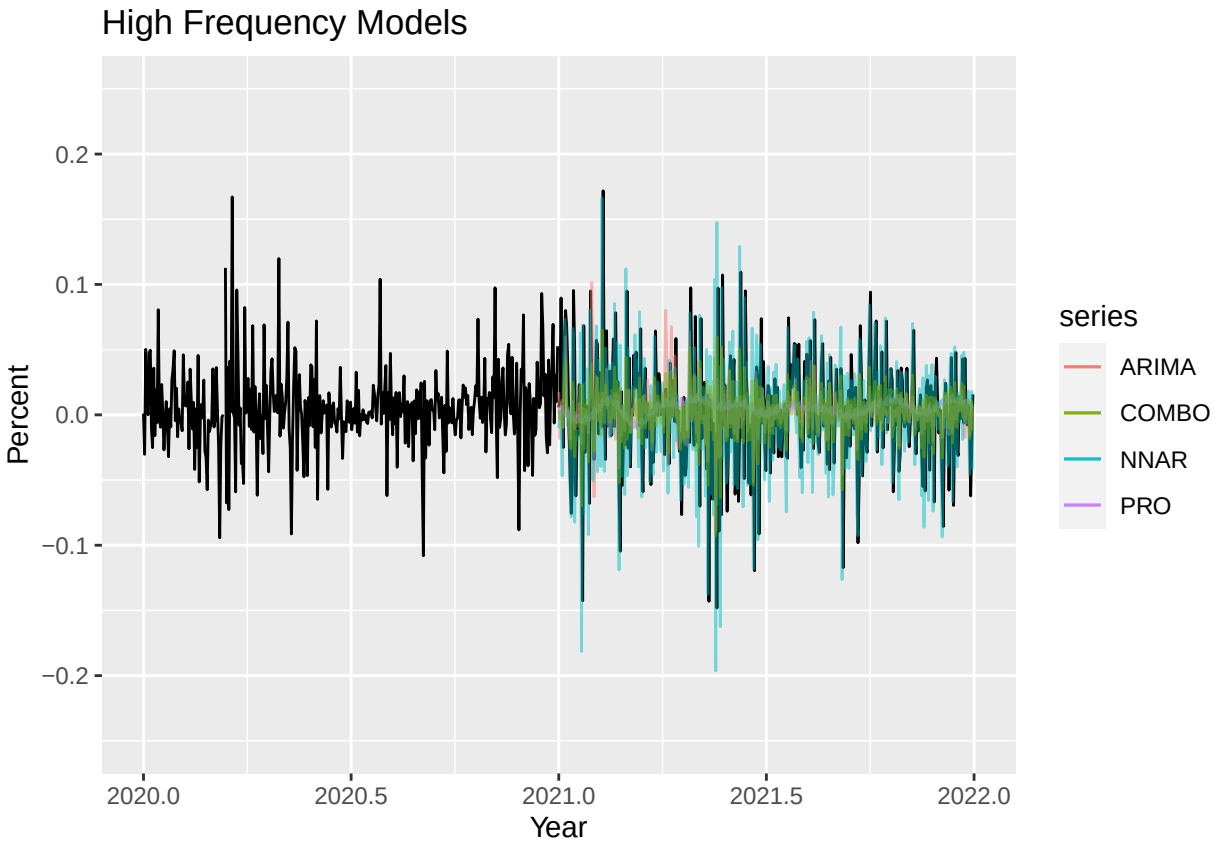
```
## [1] 39.98843
```

```
HIGH_COMBO <- (ARIMA + PRO + NNAR) / 3
LOW_COMBO <- (MAPA + HW + ETS) / 3
```

Let's plot the forecasts for both high and lower frequency models.

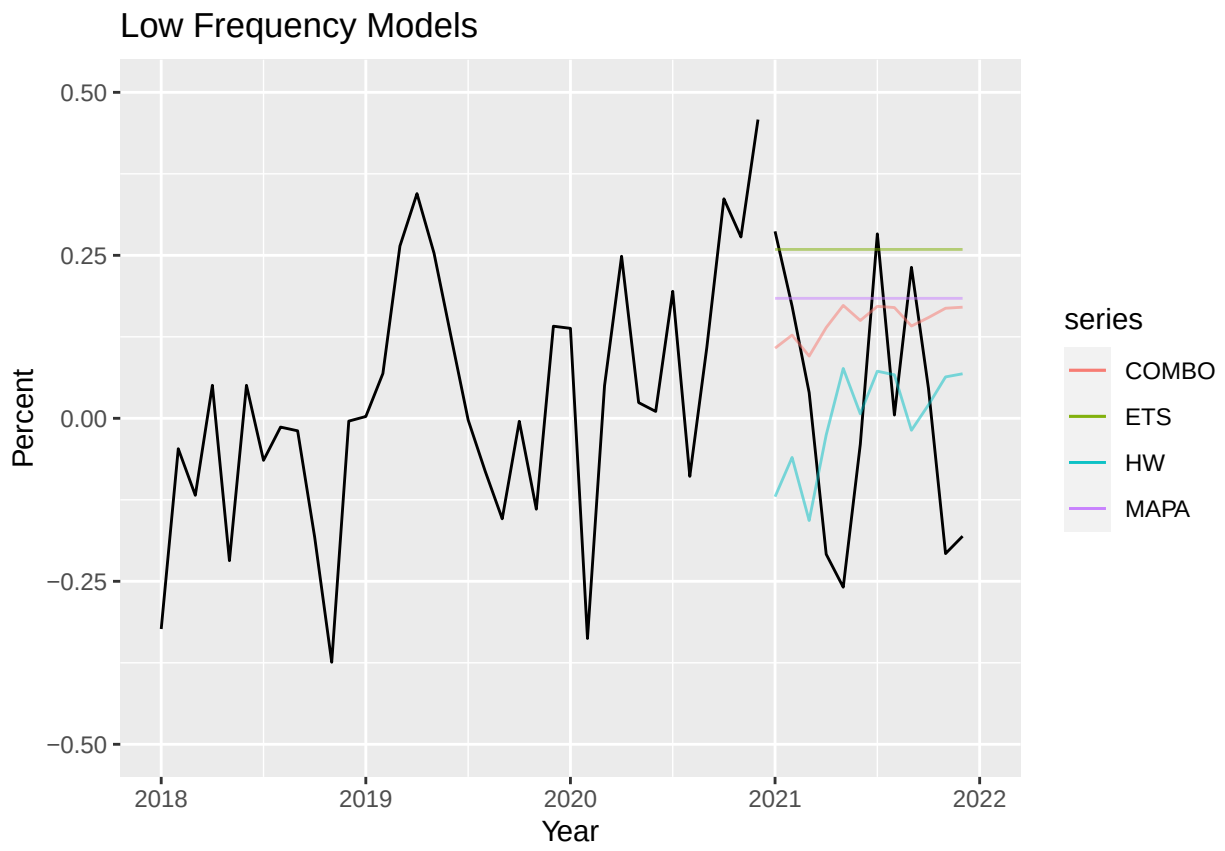
```
autoplot(train) + autolayer(test, series="Actual", color="Black") +
  autolayer(ARIMA, series="ARIMA", alpha=0.5) +
  autolayer(PRO, series="PRO", alpha=0.5) +
  autolayer(NNAR, series="NNAR", alpha=0.5) +
  autolayer(HIGH_COMBO, series="COMBO", alpha=0.5) +
  xlim(c(2020,2022)) + ylim(c(-0.25,0.25)) +
  ylab("Percent") + xlab("Year") + ggtitle("High Frequency Models")
```

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




```
autoplot(train2) + autolayer(test2, series="Actual", color="Black") +
  autolayer(MAPA, series="MAPA", alpha=0.5) +
  autolayer(HW, series="HW", alpha=0.5) +
  autolayer(ETS, series="ETS", alpha=0.5) +
  autolayer(LOW_COMBO, series="COMBO", alpha=0.5) +
  xlim(c(2018,2022)) + ylim(c(-0.5,0.5)) +
  ylab("Percent") + xlab("Year") + ggtitle("Low Frequency Models")
```

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

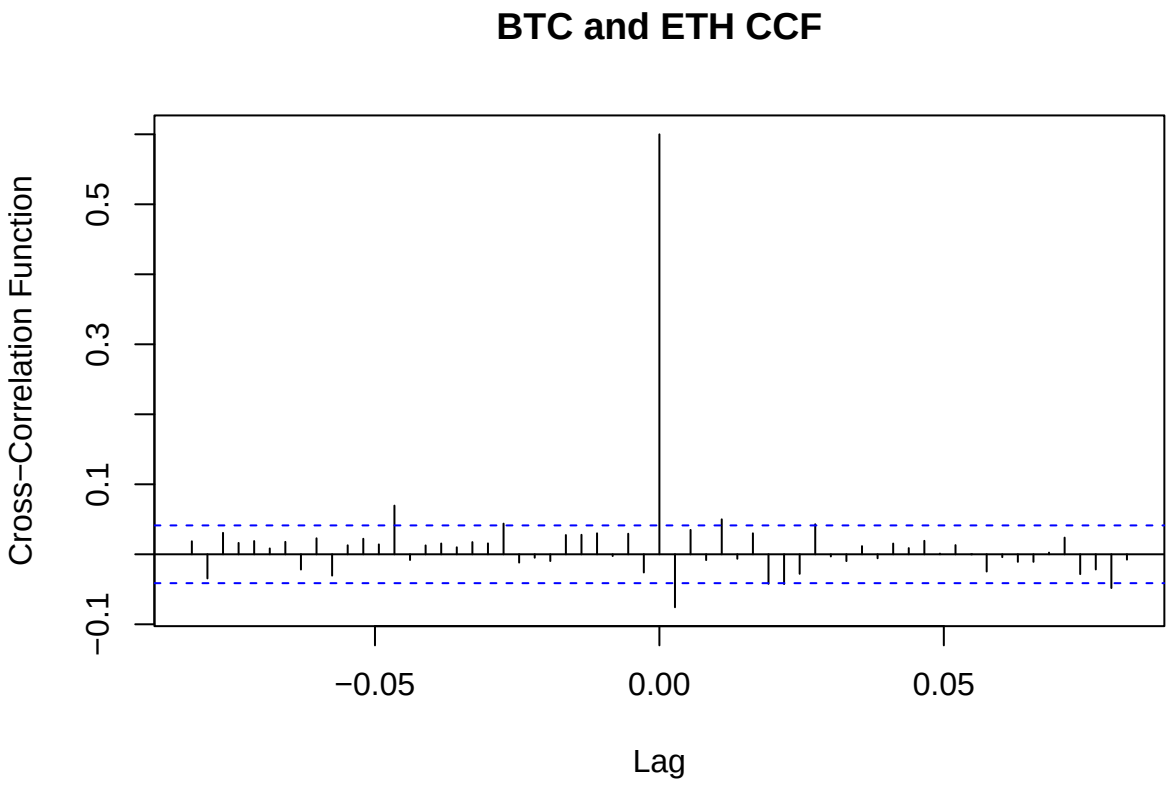


K) VAR Models

We will now create VAR models to see if we can outperform our Regression + NNETAR Model. We will create two VAR models: one between BTC and ETH, and one between BTC and XRP. First we will start with a BTC and ETH VAR Model.

BTC & ETH

```
ccf(btc_ts, eth_ts, ylab = "Cross-Correlation Function", main = "BTC and ETH CCF")
```




```
y_btc_eth = cbind(btc_ts, eth_ts)
y_tot_btc_eth = data.frame(y_btc_eth)
VARselect(y_tot_btc_eth, lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      1      1      4
##
## $criteria
##                   1             2             3             4             5
## AIC(n) -1.258532e+01 -1.258369e+01 -1.258515e+01 -1.258593e+01 -1.258413e+01
## HQ(n)  -1.257975e+01 -1.257440e+01 -1.257215e+01 -1.256922e+01 -1.256371e+01
## SC(n)  -1.257006e+01 -1.255825e+01 -1.254954e+01 -1.254015e+01 -1.252818e+01
## FPE(n)  3.421891e-06  3.427474e-06  3.422470e-06  3.419792e-06  3.425942e-06
##                   6             7             8             9            10
## AIC(n) -1.258230e+01 -1.258104e+01 -1.258075e+01 -1.257818e+01 -1.257780e+01
## HQ(n)  -1.255816e+01 -1.255319e+01 -1.254918e+01 -1.254290e+01 -1.253880e+01
## SC(n)  -1.251617e+01 -1.250473e+01 -1.249427e+01 -1.248152e+01 -1.247097e+01
## FPE(n)  3.432234e-06  3.436564e-06  3.437562e-06  3.446417e-06  3.447722e-06
```

From VARselect, we will set the oder p=4 for our VAR model.

```
vmod_btc_eth=VAR(y_btc_eth,p=4)
summary(vmod_btc_eth)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: btc_ts, eth_ts
## Deterministic variables: const
## Sample size: 2254
## Log Likelihood: 7808.03
## Roots of the characteristic polynomial:
## 0.5092 0.5092 0.4801 0.4536 0.4265 0.4265 0.4133 0.4133
## Call:
## VAR(y = y_btc_eth, p = 4)
##
##
## Estimation results for equation btc_ts:
## =======================================
## btc_ts = btc_ts.l1 + eth_ts.l1 + btc_ts.l2 + eth_ts.l2 + btc_ts.l3 + eth_ts.l3 + btc_ts.l4 + eth_ts.l
##
##             Estimate Std. Error t value Pr(>|t|)
## btc_ts.l1  0.0362162  0.0264076   1.371 0.170377
## eth_ts.l1 -0.0683279  0.0180932  -3.776 0.000163 ***
## btc_ts.l2  0.0011765  0.0264135   0.045 0.964477
## eth_ts.l2  0.0233009  0.0181350   1.285 0.198976
## btc_ts.l3  0.0154947  0.0264078   0.587 0.557431
## eth_ts.l3 -0.0096118  0.0181364  -0.530 0.596182
## btc_ts.l4 -0.0201061  0.0263036  -0.764 0.444717
## eth_ts.l4  0.0451053  0.0181430   2.486 0.012987 *
## const      0.0019622  0.0008435   2.326 0.020087 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03973 on 2245 degrees of freedom
## Multiple R-Squared: 0.01077, Adjusted R-squared: 0.007247
## F-statistic: 3.056 on 8 and 2245 DF,  p-value: 0.002004
##
##
## Estimation results for equation eth_ts:
## =======================================
## eth_ts = btc_ts.l1 + eth_ts.l1 + btc_ts.l2 + eth_ts.l2 + btc_ts.l3 + eth_ts.l3 + btc_ts.l4 + eth_ts.l
##
##            Estimate Std. Error t value Pr(>|t|)
## btc_ts.l1 -0.040537   0.038586  -1.051  0.29357
## eth_ts.l1  0.005645   0.026437   0.214  0.83093
## btc_ts.l2  0.018046   0.038595   0.468  0.64013
## eth_ts.l2  0.032172   0.026498   1.214  0.22484
## btc_ts.l3 -0.055908   0.038586  -1.449  0.14751
## eth_ts.l3  0.064908   0.026500   2.449  0.01439 *
## btc_ts.l4  0.034794   0.038434   0.905  0.36541
## eth_ts.l4  0.009197   0.026510   0.347  0.72869
## const      0.003201   0.001232   2.597  0.00946 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

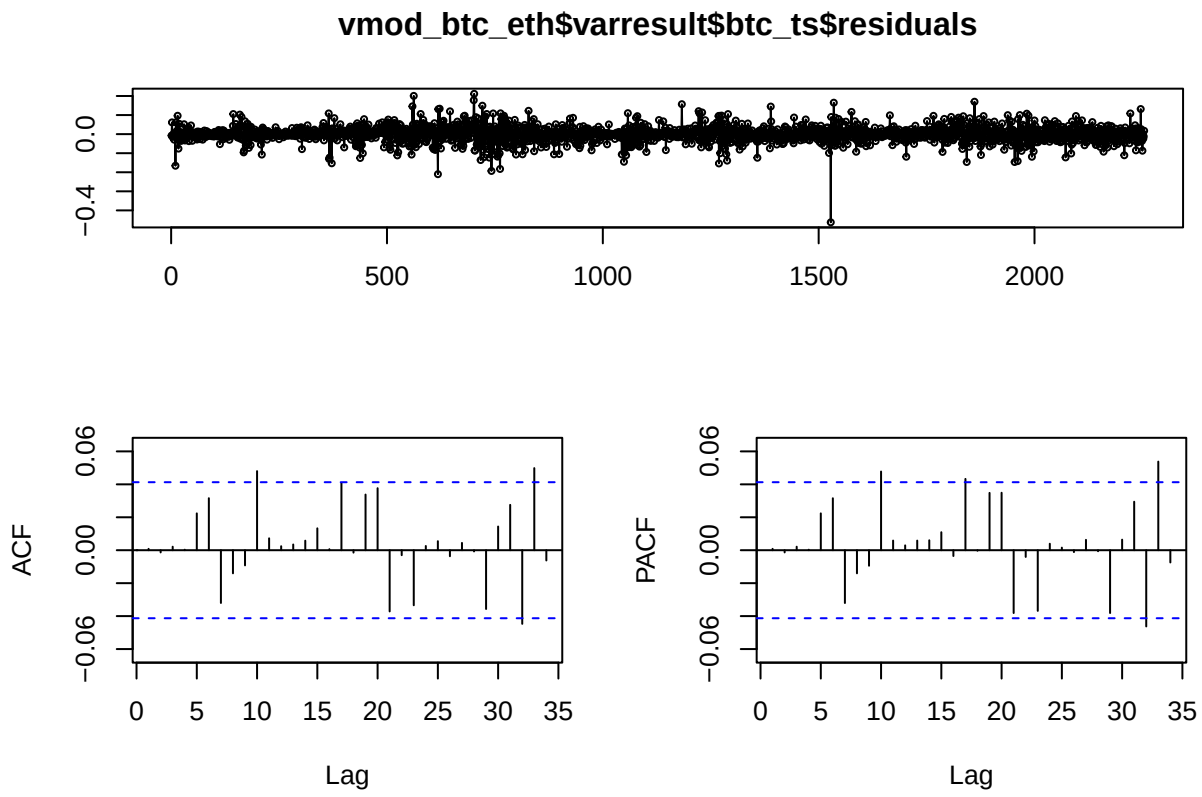
```
## Residual standard error: 0.05805 on 2245 degrees of freedom
## Multiple R-Squared: 0.006031,    Adjusted R-squared: 0.002489
## F-statistic: 1.703 on 8 and 2245 DF,  p-value: 0.09286
##
##
##
## Covariance matrix of residuals:
##          btc_ts   eth_ts
## btc_ts 0.001579 0.001391
## eth_ts 0.001391 0.003370
##
## Correlation matrix of residuals:
##        btc_ts eth_ts
## btc_ts  1.000  0.603
## eth_ts  0.603  1.000
```

The residuals appear to be mostly scattered around 0 with very little dynamics. Thus, this VAR model appears to be a good fit so far.

```
vmod_btc_eth$varresult$btc_ts
```

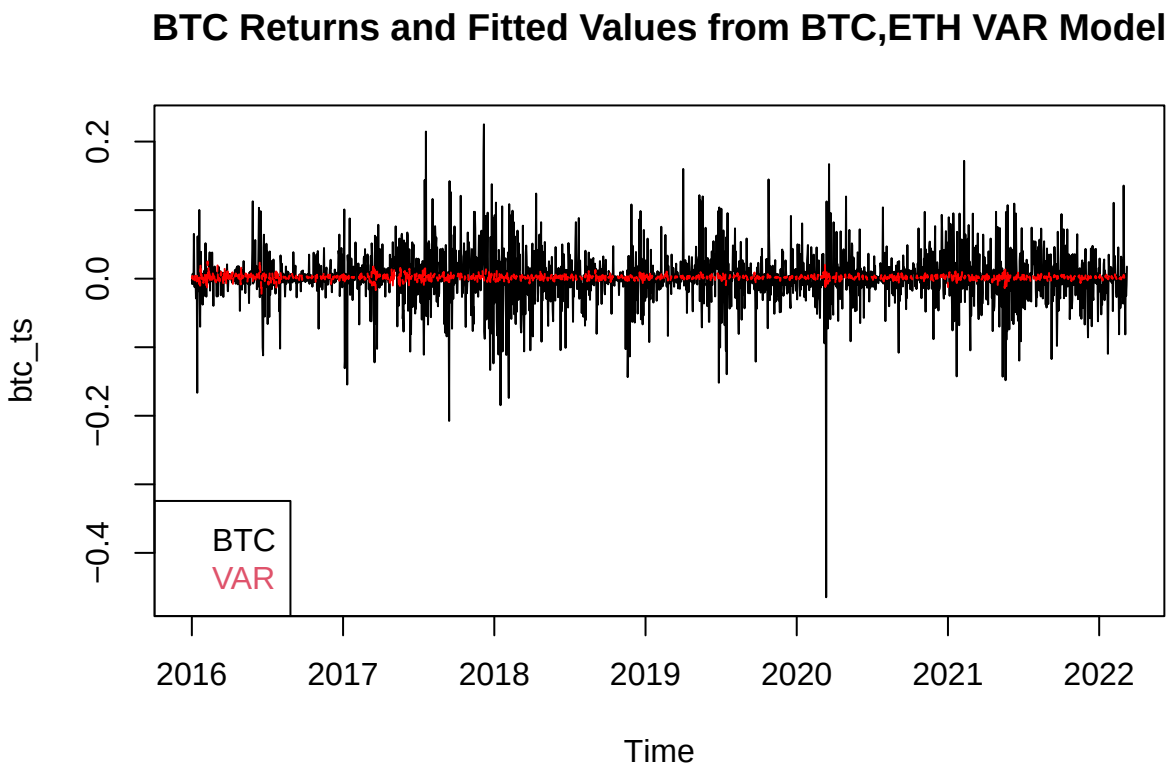
```
##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Coefficients:
## btc_ts.l1  eth_ts.l1  btc_ts.l2  eth_ts.l2  btc_ts.l3  eth_ts.l3  btc_ts.l4
##  0.036216  -0.068328   0.001176   0.023301   0.015495  -0.009612  -0.020106
## eth_ts.l4      const
##  0.045105   0.001962
```

```
tsdisplay(vmod_btc_eth$varresult$btc_ts$residuals)
```



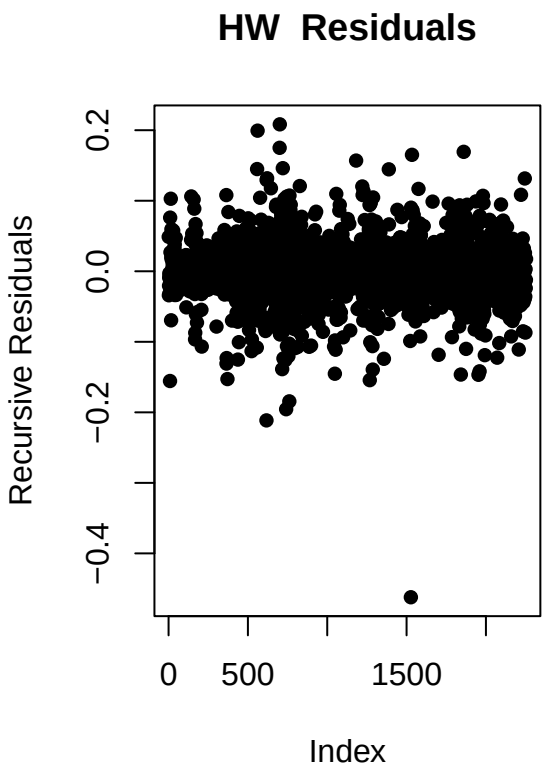
While the residuals appeared to resemble white noise, it is apparent that our VAR model using ETH returns does not capture a large portion of the volatility in BTC returns. We will now try to create another VAR model using XRP returns to forecast BTC returns.

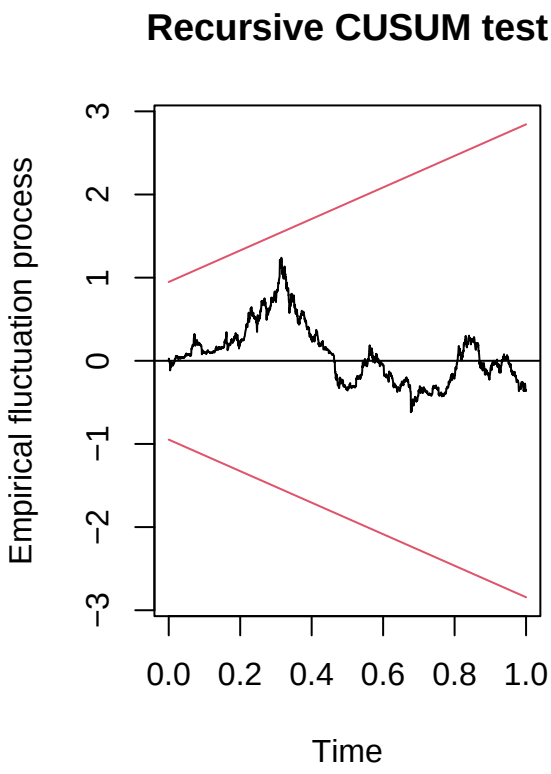
```
vmod_eth_ts = ts(vmod_btc_eth$varresult$btc_ts$fitted.values,start=c(2016,1),freq=365)
plot(btc_ts,col="black",lwd=1,lty=1,main="BTC Returns and Fitted Values from BTC,ETH VAR Model")
lines(vmod_eth_ts,col="red",lty=2)
legend("bottomleft",legend=c("BTC","VAR"),text.col=1:4,bty = "o")
```



Interpretation

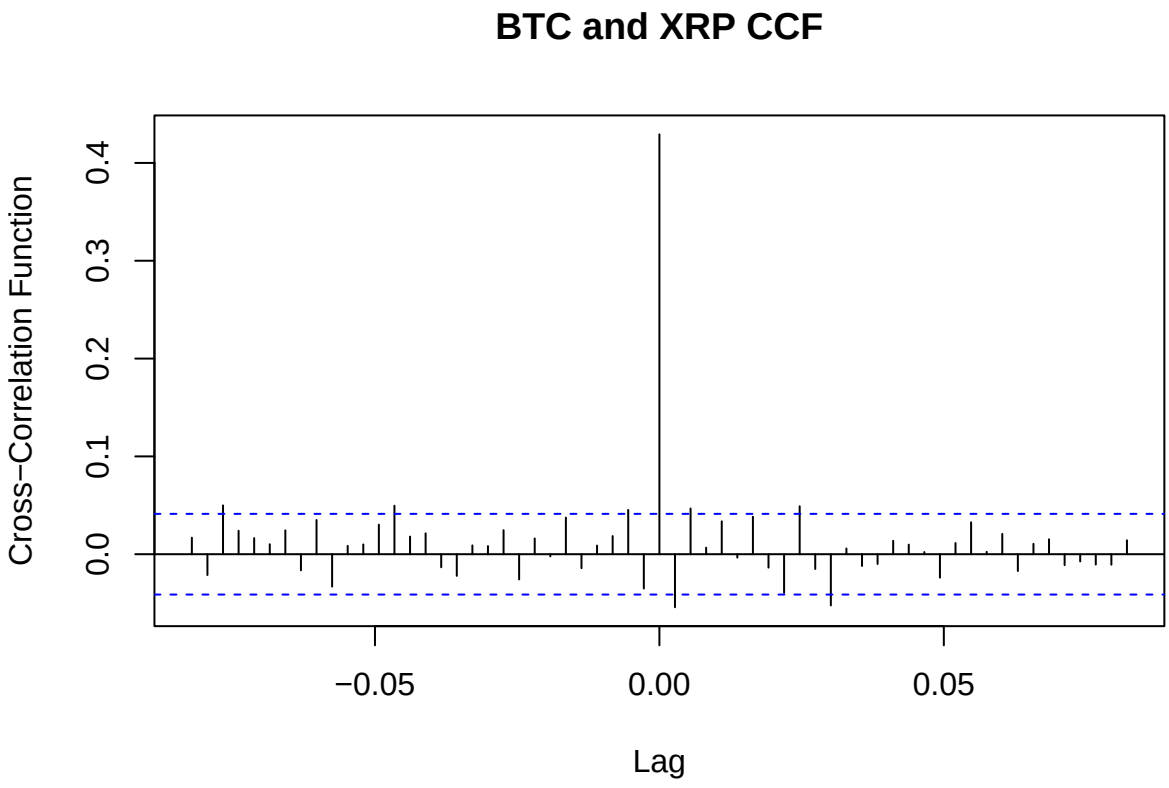
```
par(mfrow=c(1,2))
y = recresid(vmod_btc_eth$varresult$btc_ts$residuals ~ 1)
plot(y, pch = 16, main = "HW  Residuals", ylab = "Recursive Residuals")
plot(efp(vmod_btc_eth$varresult$btc_ts$residuals~1, type = "Rec-CUSUM"))
```





BTC & XRP

```
ccf(btc_ts,xrp_ts,ylab="Cross-Correlation Function", main = "BTC and XRP CCF")
```




```
y_btc_xrp=cbind(btc_ts,xrp_ts)
y_tot_btc_xrp=data.frame(y_btc_xrp)
VARselect(y_btc_xrp, lag.max=10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      1      1      2
##
## $criteria
##                   1             2             3             4             5
## AIC(n) -1.196507e+01 -1.196787e+01 -1.196607e+01 -1.196362e+01 -1.196340e+01
## HQ(n)  -1.195950e+01 -1.195859e+01 -1.195307e+01 -1.194691e+01 -1.194297e+01
## SC(n)  -1.194981e+01 -1.194244e+01 -1.193046e+01 -1.191784e+01 -1.190744e+01
## FPE(n)  6.362620e-06  6.344822e-06  6.356266e-06  6.371847e-06  6.373273e-06
##                   6             7             8             9            10
## AIC(n) -1.196241e+01 -1.196116e+01 -1.196376e+01 -1.196599e+01 -1.196665e+01
## HQ(n)  -1.193827e+01 -1.193330e+01 -1.193219e+01 -1.193071e+01 -1.192766e+01
## SC(n)  -1.189628e+01 -1.188485e+01 -1.187728e+01 -1.186934e+01 -1.185983e+01
## FPE(n)  6.379560e-06  6.387571e-06  6.370992e-06  6.356787e-06  6.352563e-06
```

From VARselect we will choose order p=2 for our VAR model.

```
vmod_btc_xrp=VAR(y_btc_xrp,p=2)
summary(vmod_btc_xrp)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: btc_ts, xrp_ts
## Deterministic variables: const
## Sample size: 2256
## Log Likelihood: 7113.779
## Roots of the characteristic polynomial:
## 0.2992 0.2688 0.05111 0.05111
## Call:
## VAR(y = y_btc_xrp, p = 2)
##
##
## Estimation results for equation btc_ts:
## =======================================
## btc_ts = btc_ts.l1 + xrp_ts.l1 + btc_ts.l2 + xrp_ts.l2 + const
##
##            Estimate Std. Error t value Pr(>|t|)
## btc_ts.l1 -0.001896   0.023308  -0.081   0.9352
## xrp_ts.l1 -0.029806   0.013288  -2.243   0.0250 *
## btc_ts.l2  0.005146   0.023289   0.221   0.8251
## xrp_ts.l2  0.024689   0.013298   1.857   0.0635 .
## const      0.001999   0.000840   2.380   0.0174 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03979 on 2251 degrees of freedom
## Multiple R-Squared: 0.005035,    Adjusted R-squared: 0.003267
## F-statistic: 2.848 on 4 and 2251 DF,  p-value: 0.02273
##
##
## Estimation results for equation xrp_ts:
## =======================================
## xrp_ts = btc_ts.l1 + xrp_ts.l1 + btc_ts.l2 + xrp_ts.l2 + const
##
##            Estimate Std. Error t value Pr(>|t|)
## btc_ts.l1 -0.047840   0.040798  -1.173  0.24108
## xrp_ts.l1 -0.008284   0.023260  -0.356  0.72177
## btc_ts.l2  0.023489   0.040765   0.576  0.56453
## xrp_ts.l2  0.071871   0.023277   3.088  0.00204 **
## const      0.002034   0.001470   1.383  0.16667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.06965 on 2251 degrees of freedom
## Multiple R-Squared: 0.007446,    Adjusted R-squared: 0.005683
## F-statistic: 4.222 on 4 and 2251 DF,  p-value: 0.002084
##
##
##
## Covariance matrix of residuals:
##          btc_ts   xrp_ts
```

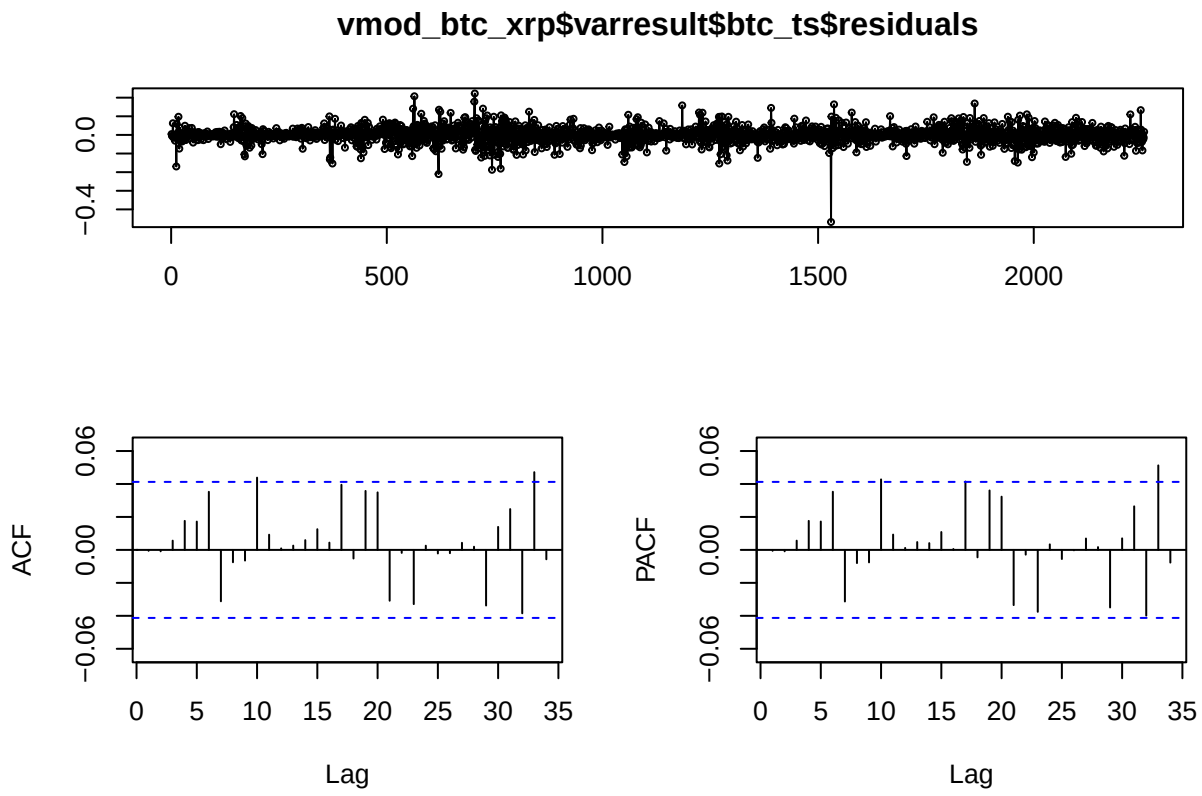
```
## btc_ts 0.001583 0.001184
## xrp_ts 0.001184 0.004852
##
## Correlation matrix of residuals:
##        btc_ts xrp_ts
## btc_ts  1.000  0.427
## xrp_ts  0.427  1.000
```

These residuals also appear to resemble white noise with very little dynamics that do not appear to be significant. Once again we will move forward with this model since the residuals suggest this VAR model is a good fit.

```
vmod_btc_xrp$varresult$btc_ts
```

```
##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Coefficients:
## btc_ts.l1  xrp_ts.l1  btc_ts.l2  xrp_ts.l2      const
## -0.001896  -0.029806   0.005146   0.024689   0.001999
```

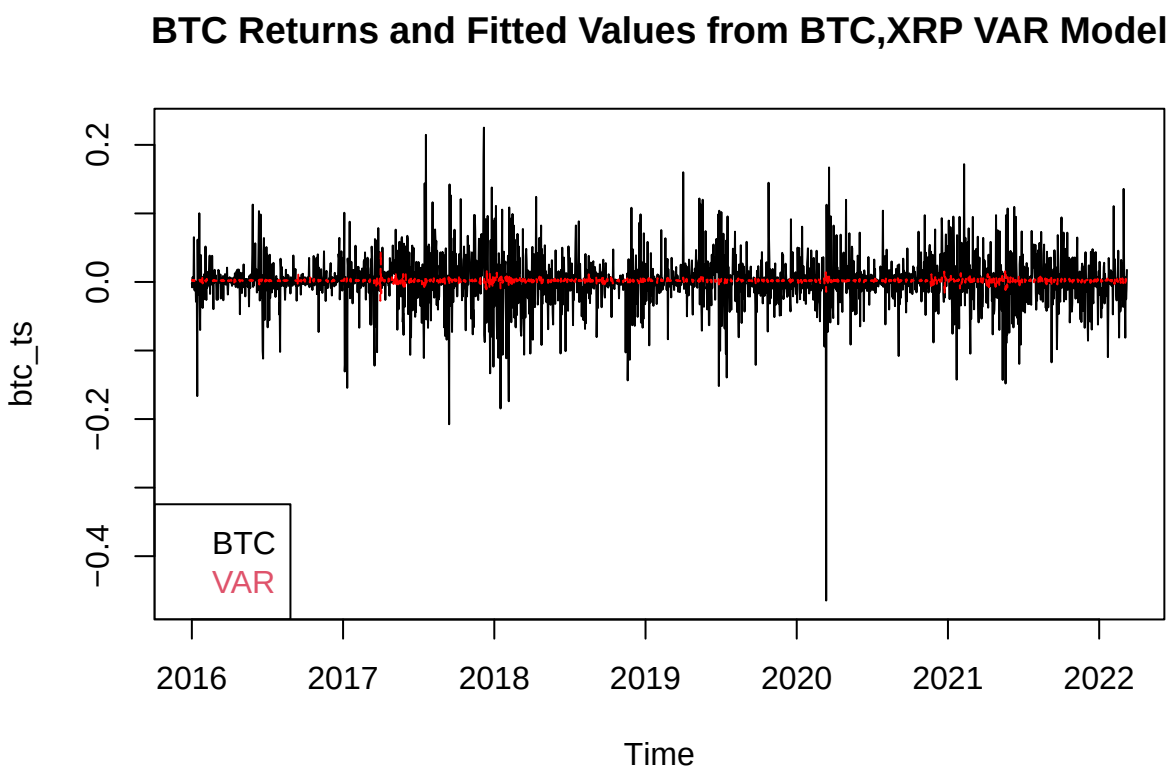
```
tsdisplay(vmod_btc_xrp$varresult$btc_ts$residuals)
```



As with the orevious VAR model, the VAR model using ETH returns do not seem to capture any of the volatility in the BTC returns series. We will now train and test each VAR model (the first one using ETH

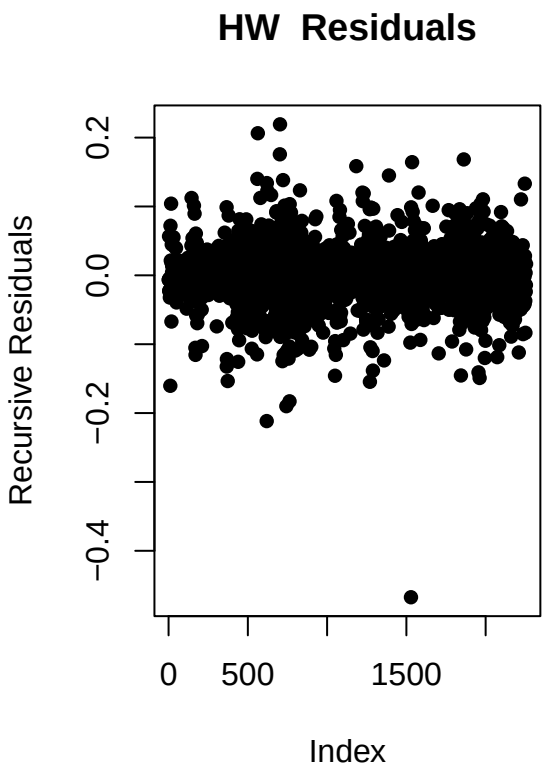
returns and the second using XRP returns) in order to measure the MAPE to decide which VAR model performs better in predicting BTC returns. We will then compare this chosen model to our previous models to assess if we can outperfrom them.

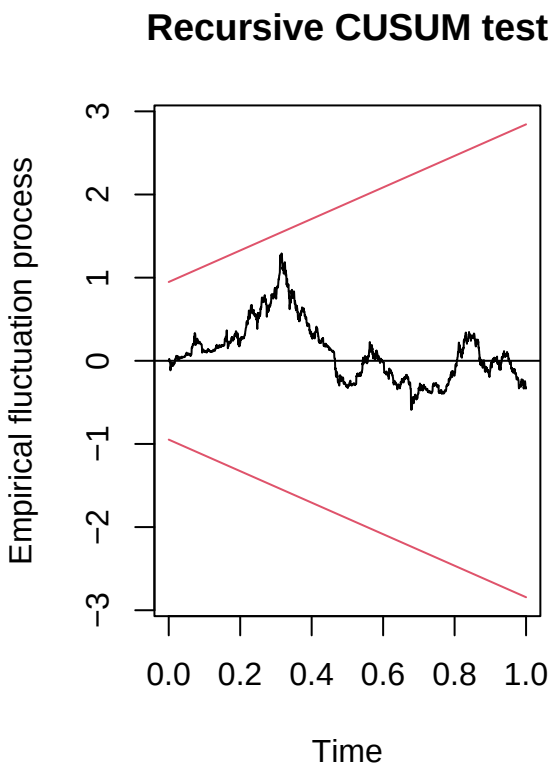
```
vmod_xrp_ts = ts(vmod_btc_xrp$varresult$btc_ts$fitted.values, start=c(2016,1),freq=365)
plot(btc_ts,col="black",lwd=1,lty=1,main="BTC Returns and Fitted Values from BTC,XRP VAR Model")
lines(vmod_xrp_ts,col="red",lty=2)
legend("bottomleft",legend=c("BTC","VAR"),text.col=1:4,bty = "o")
```



Interpretation

```
par(mfrow=c(1,2))
y = recresid(vmod_btc_xrp$varresult$btc_ts$residuals ~ 1)
plot(y, pch = 16, main = "HW  Residuals", ylab = "Recursive Residuals")
plot(efp(vmod_btc_xrp$varresult$btc_ts$residuals~1, type = "Rec-CUSUM"))
```





Train & Test Accuracy

Our VAR model using ETH returns to predict BTC returns has a significantly lower MAPE, so we will forecast with this model.

```
#Train and test ETH VAR Model
btc.train <- window(btc_ts, end=c(2021,1))
eth.train <- window(eth_ts, end=c(2021,1))
y.eth.train=cbind(btc.train,eth.train)
y_eth_train=data.frame(y.eth.train)
vmod.eth.train=VAR(y_eth_train,p=4)
vmod.eth.train.pred=predict(object=vmod.eth.train, n.ahead=12)

#Train and test XRP VAR Model
xrp.train <- window(xrp_ts, end=c(2021,1))
y.xrp.train=cbind(btc.train,xrp.train)
y_xrp_train=data.frame(y.xrp.train)
vmod.xrp.train=VAR(y_xrp_train,p=2)
vmod.xrp.train.pred=predict(object=vmod.xrp.train, n.ahead=12)

#Compare MAPEs
MAPE(vmod.eth.train.pred$model$varresult$btc.train$fitted.values, y_eth_train$btc.train[1:1822])
```

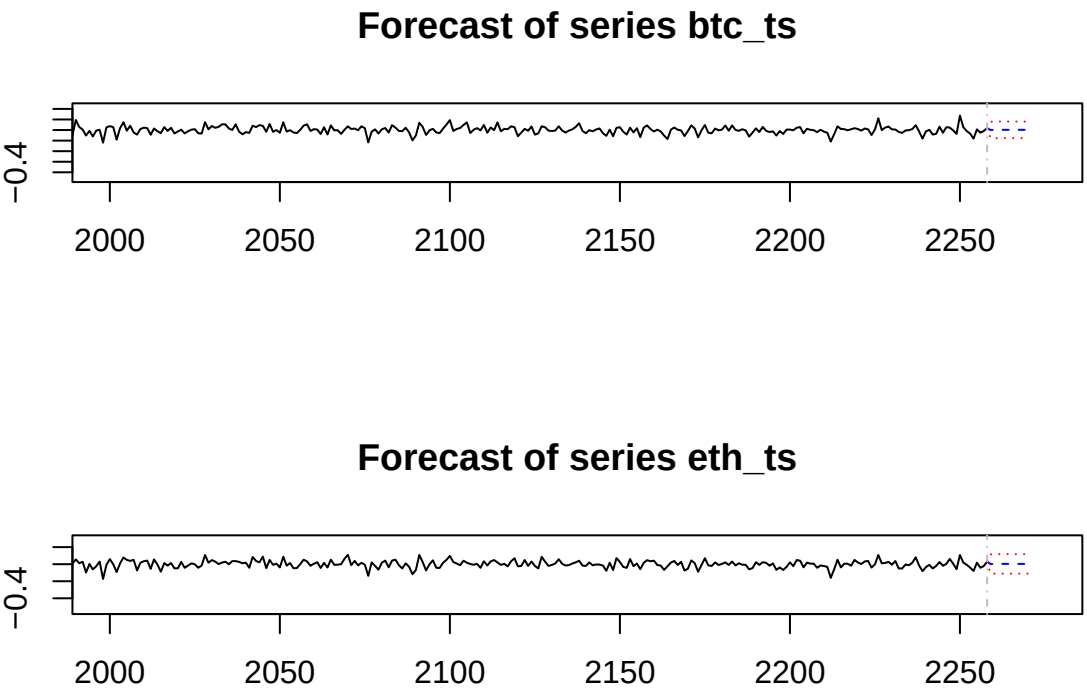
```
## [1] 35.50285
```

```
MAPE(vmod.xrp.train.pred$model$varresult$btc.train$fitted.values, y_xrp_train$btc.train[1:1824])
```

```
## [1] 147.8384
```

Forecast 12 steps ahead using ETH VAR Model

```
vmod.pred <- predict(object=vmod_btc_eth, n.ahead=12)
plot(vmod.pred, xlim=c(2000,2275))
```



L) Financial Performance & Interpretation

III. Conclusion

Make sure to explain why we did not use HW / ETS because they were not built for this high of frequency data and crashed our Rstudio so we supplemented with THIEF / MAPAS / TBATS.

IV. References

https://facebook.github.io/prophet/docs/quick_start.html