

Econ 144 Project 3

Shannan Liu, Austin Pham, Zach Wrubel

10 March, 2022

Contents

I. Introduction	2
Brief Background on Data:	2
Source of Data	2
II. Results	5
A) Time Series Plots	5
B) Correlations	11
C) STL Decomposition	16
D) ARIMA	17
E) ETS, Regression + ETS	25
F) Holt-Winters	33
G) NNETAR	42
H) MAPA	50
I) Prophet	52
J) Train & Test Accuracy	57
K) VAR Models	62
L) Financial Performance & Interpretation	74
III. Conclusion	75
IV. References	75

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 cryptocurrencies.

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
```

```
##
```

```

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)
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))

```

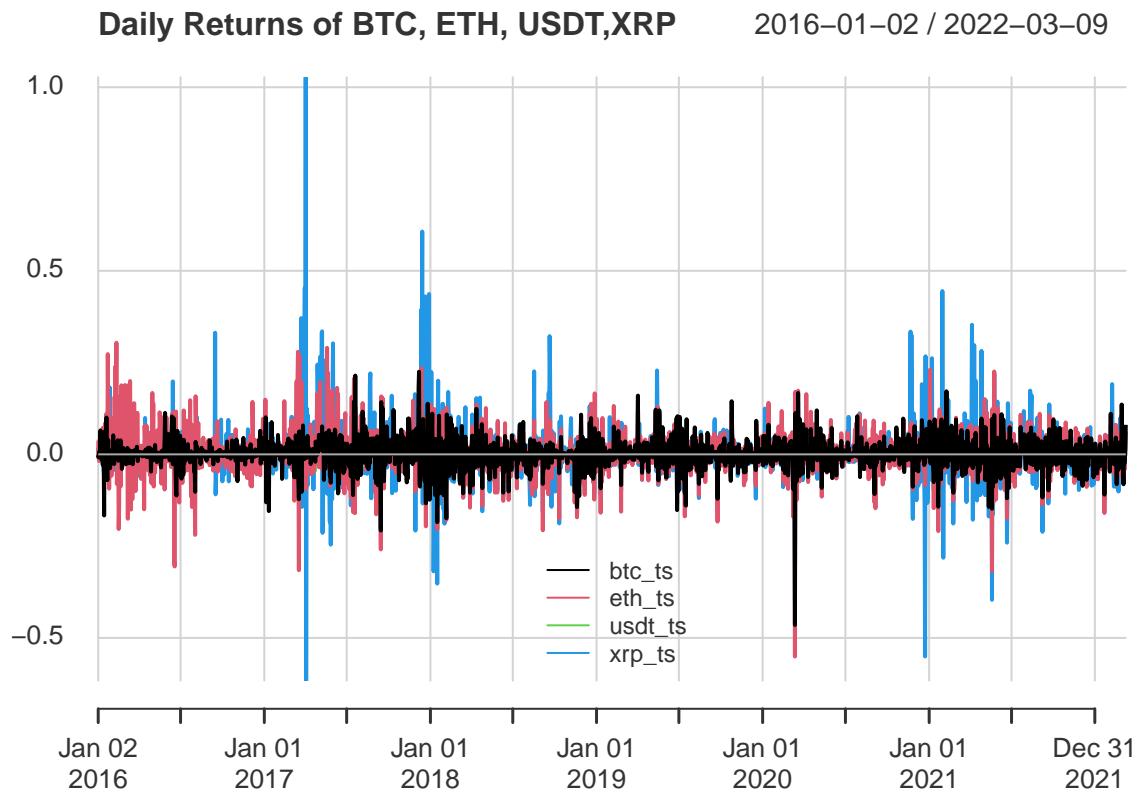
```
usdt_ts <- ts(usdt$returns, frequency = 365, start = c(2016,1))
xrp_ts <- ts(xrp$returns, frequency = 365, start = c(2016,1))
btc_ts2 <- ts(btc_monthly$returns, frequency = 12, start = c(2016,1))
eth_ts2 <- ts(eth_monthly$returns, frequency = 12, start = c(2016,1))
usdt_ts2 <- ts(usdt_monthly$returns, frequency = 12, start = c(2016,1))
xrp_ts2 <- ts(xrp_monthly$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",)
```



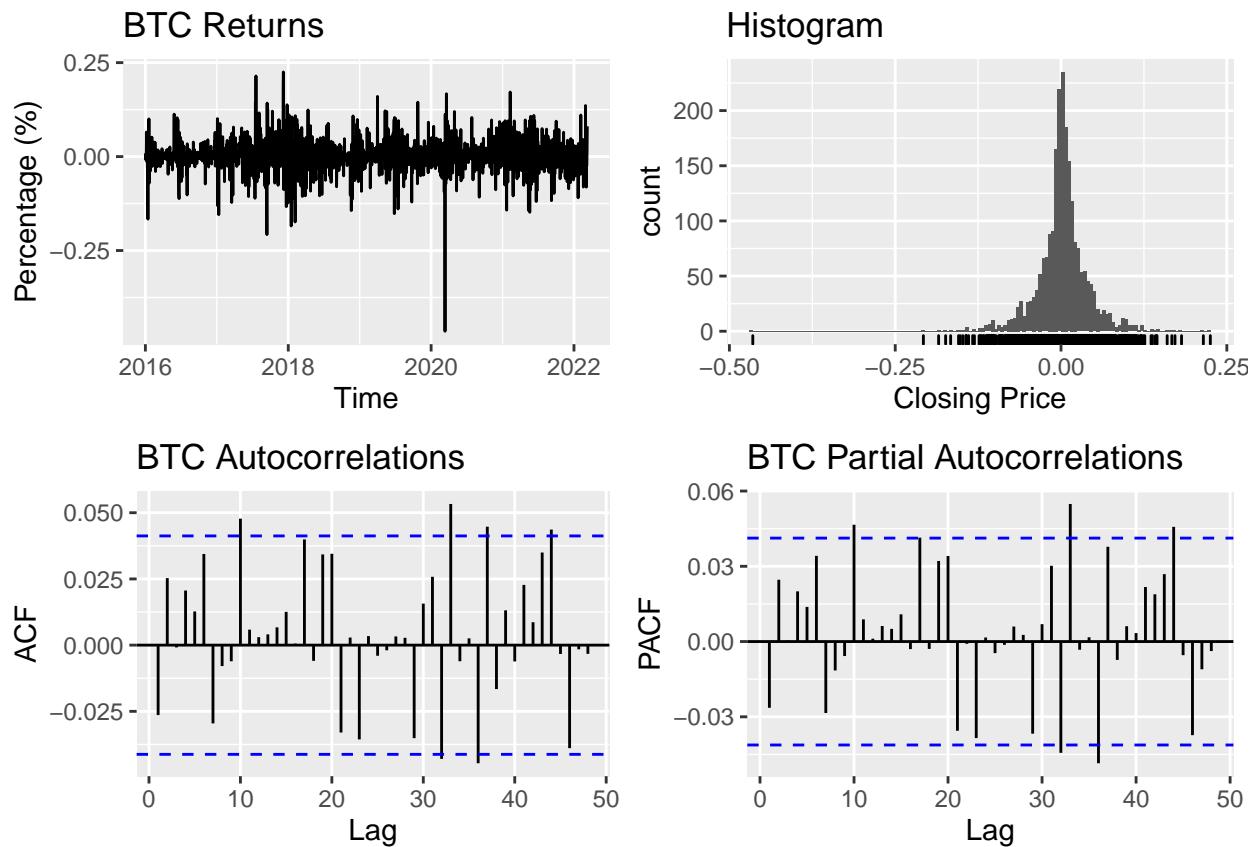
In the individual plots, we observe that the returns data for all the cryptocurrencies look stationary and that they have ACFs and PACFs that resemble white noise. Therefore, they may not be useful as regressors. This also indicates that it may be difficult to model BTC's returns over time.

```
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)
```

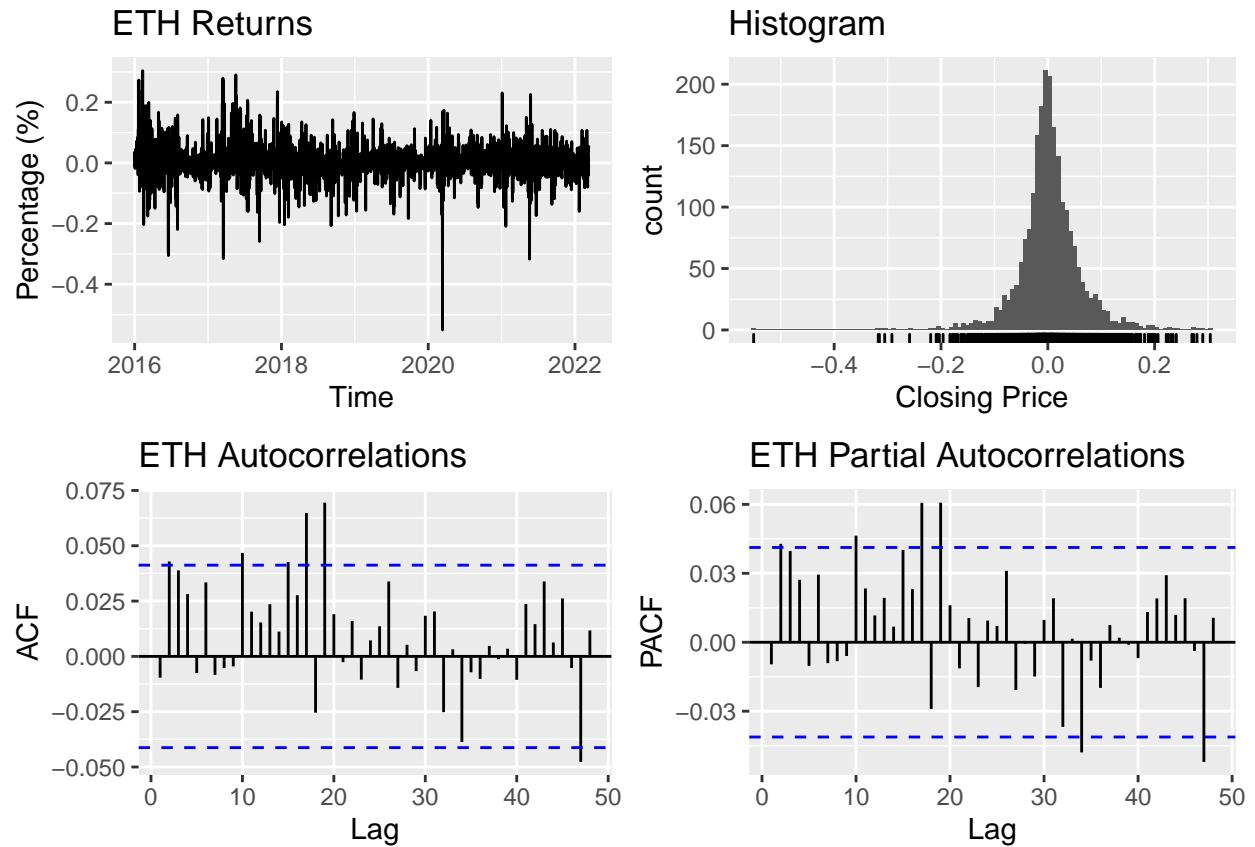
```

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")

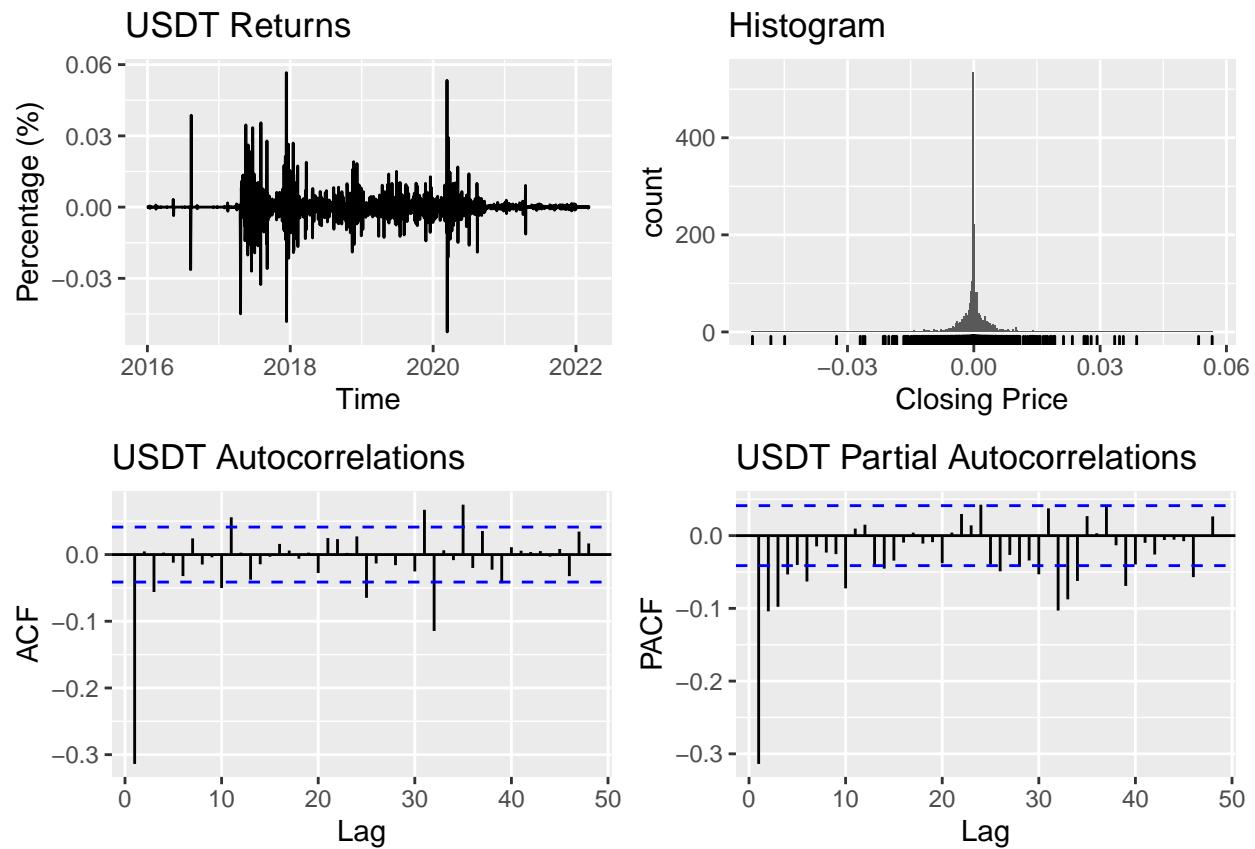
```



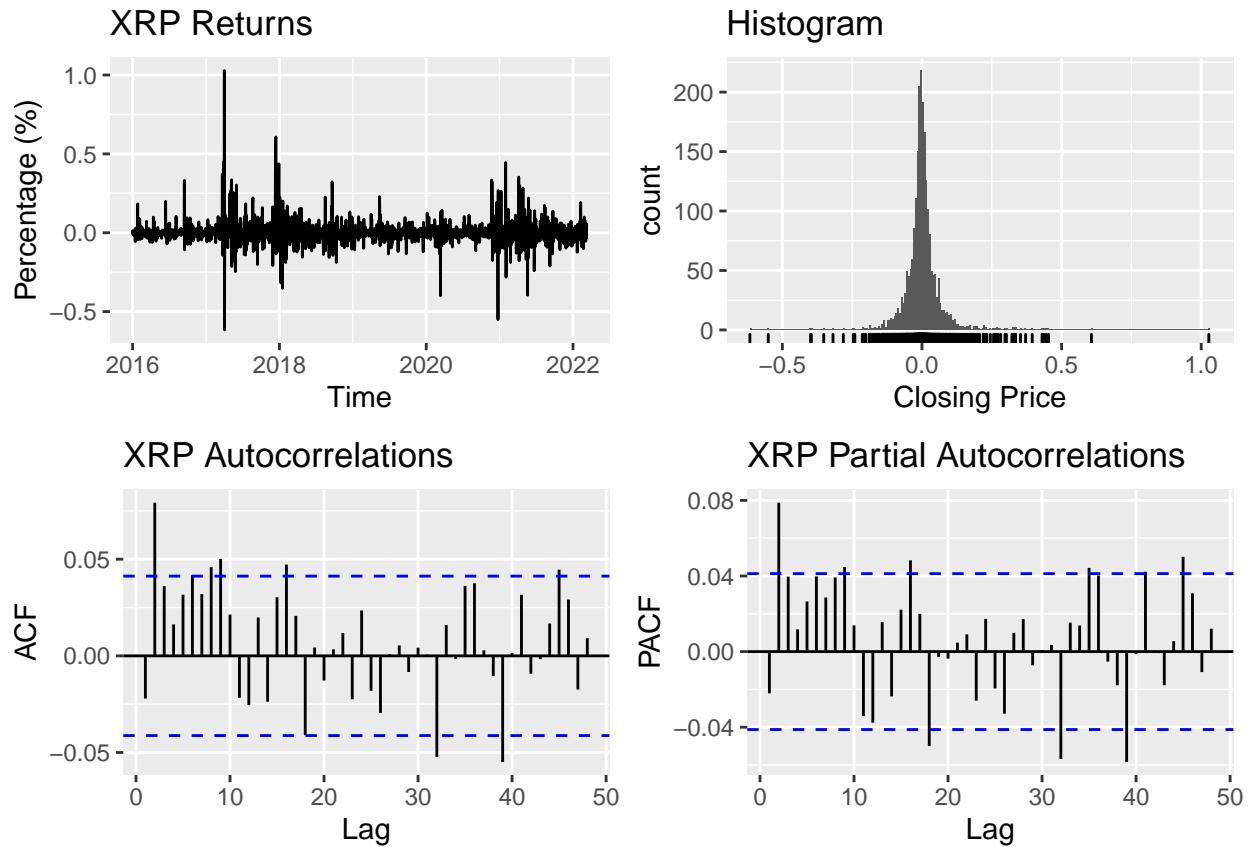
```
tsplot(eth_ts,"ETH")
```



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

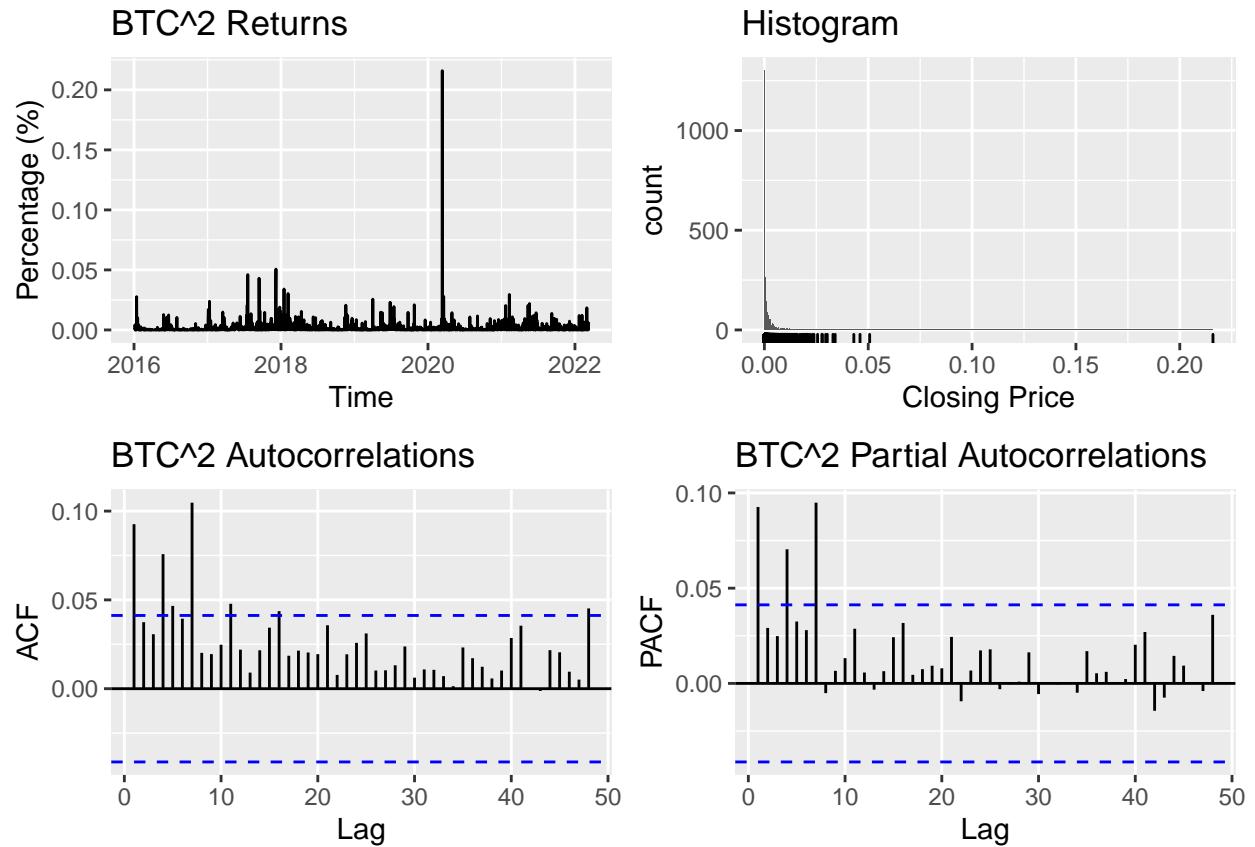


```
tsplot(xrp_ts, "XRP")
```

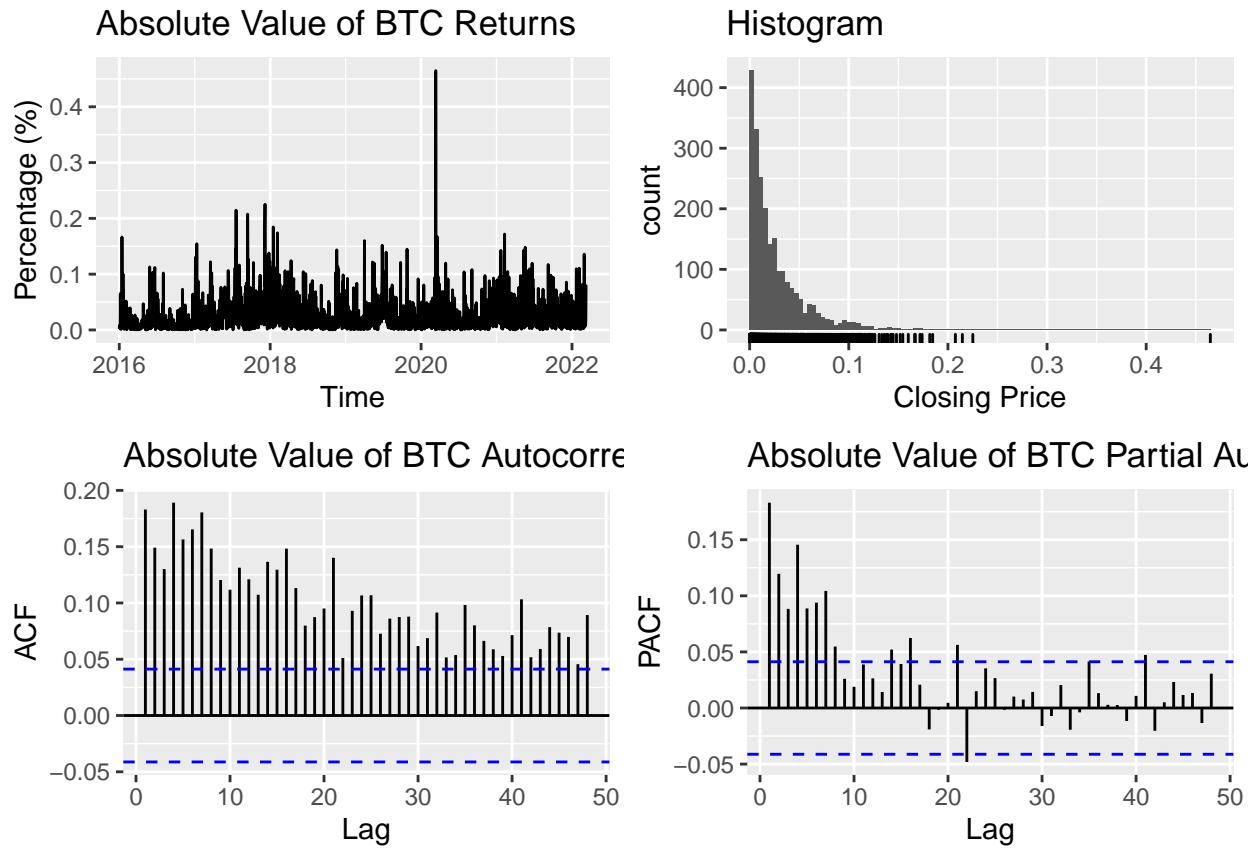


However, non-linear relationships between the lags of BTC's returns do exist. Thus, it may be worthwhile to attempt to model this data.

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



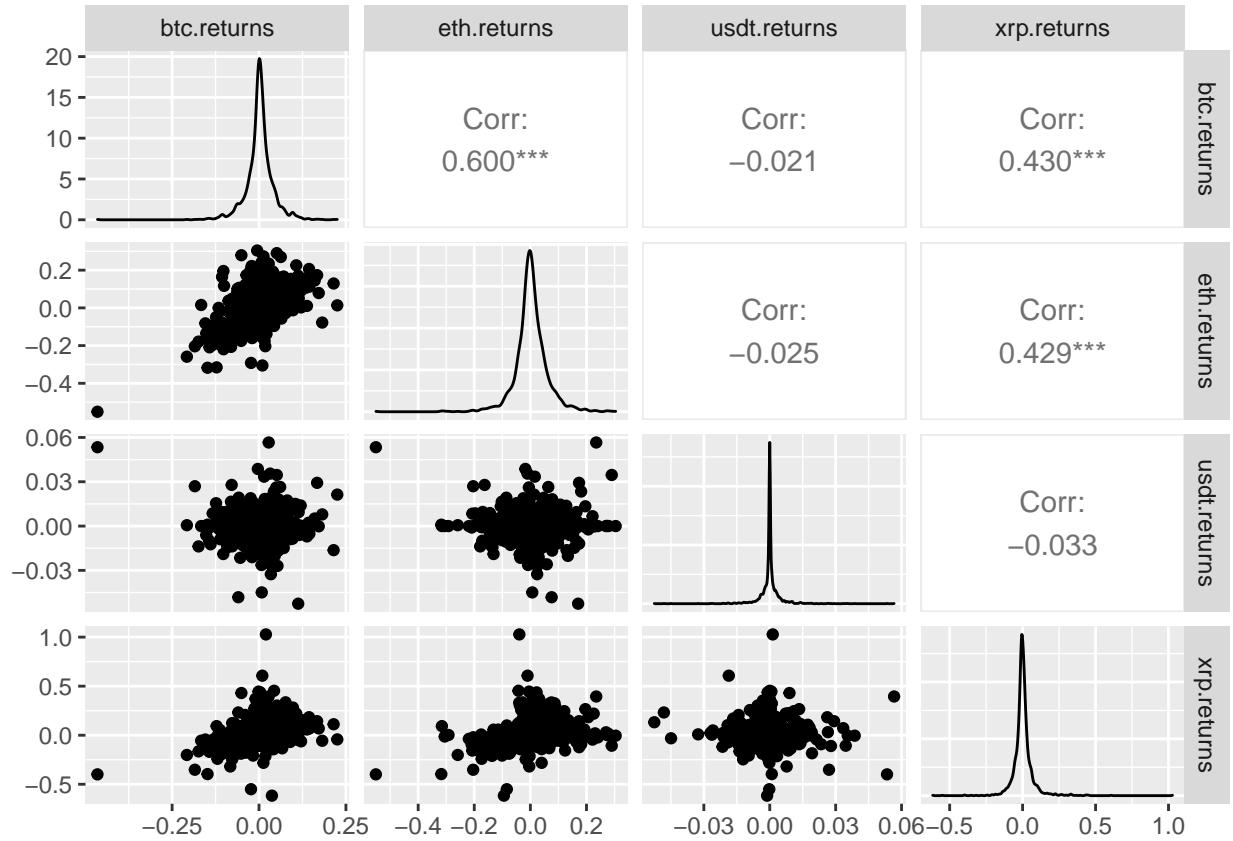
```
tsplot(abs(btc_ts),"Absolute Value of BTC")
```



B) Correlations

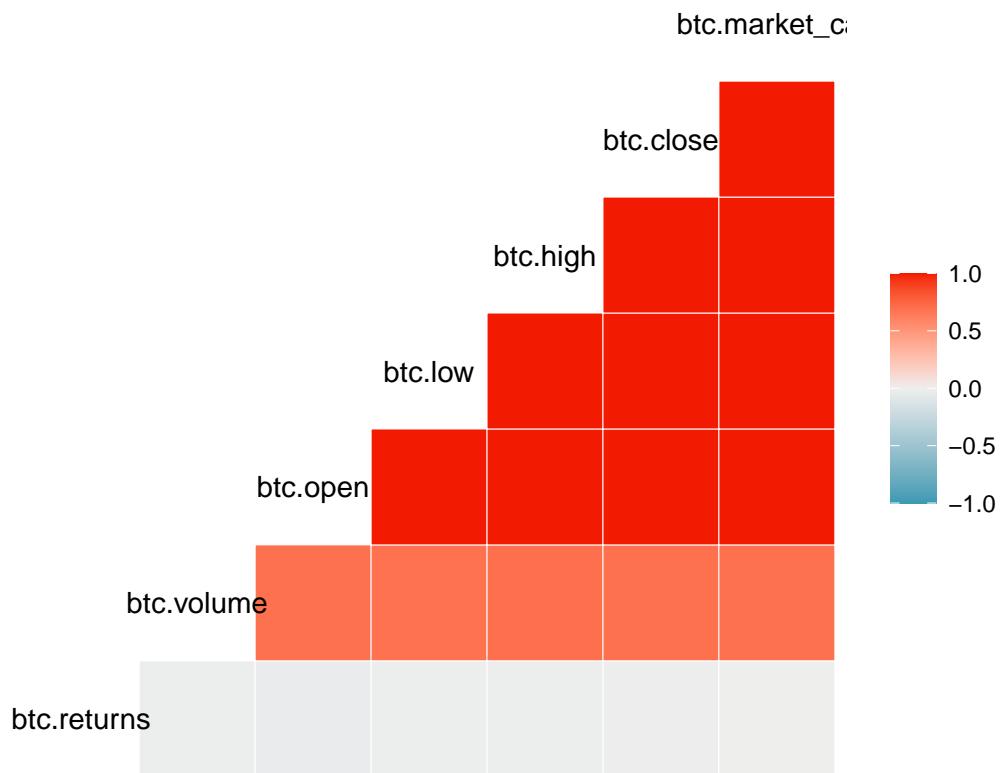
We'll utilize a scatterplot matrix and correlation plot to visualize the relationships between all of the currencies' returns. The scatterplot matrix shows that BTC, ETH, and XRP have moderately positive correlations with each other, suggesting that they may be useful regressors for modelling BTC's returns.

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



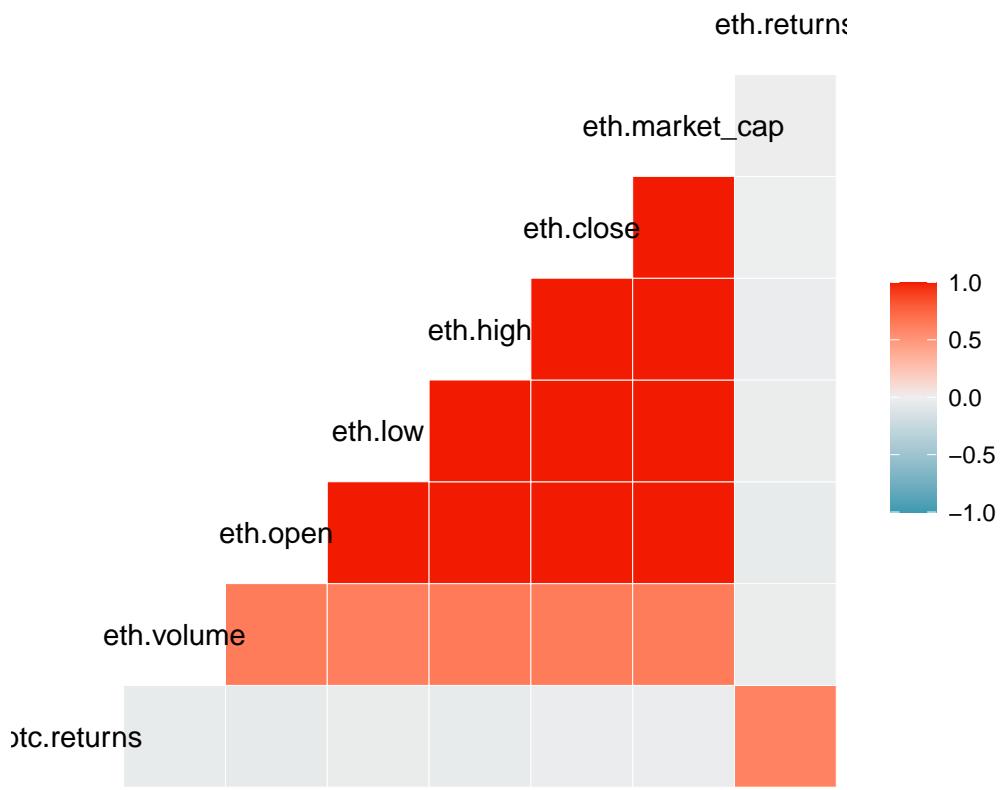
There is very little correlation between BTC's returns and other data about it.

```
# 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)
```



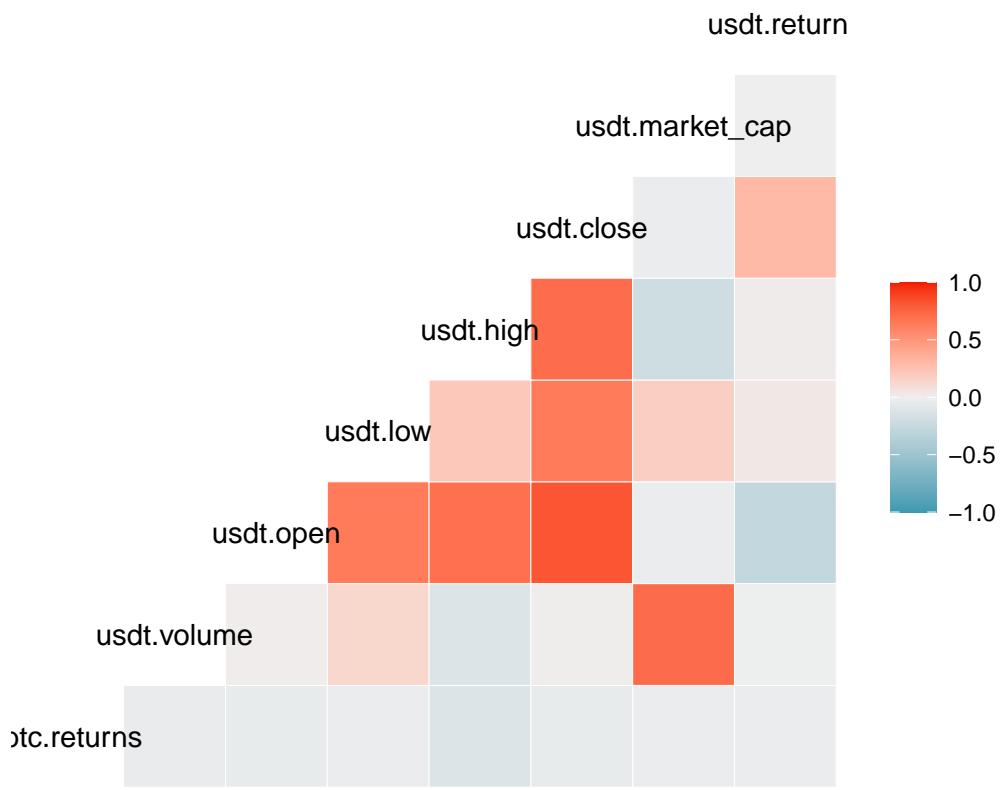
ETH's returns is the only variable that is positively linearly correlated with BTC's returns.

```
# 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)
```



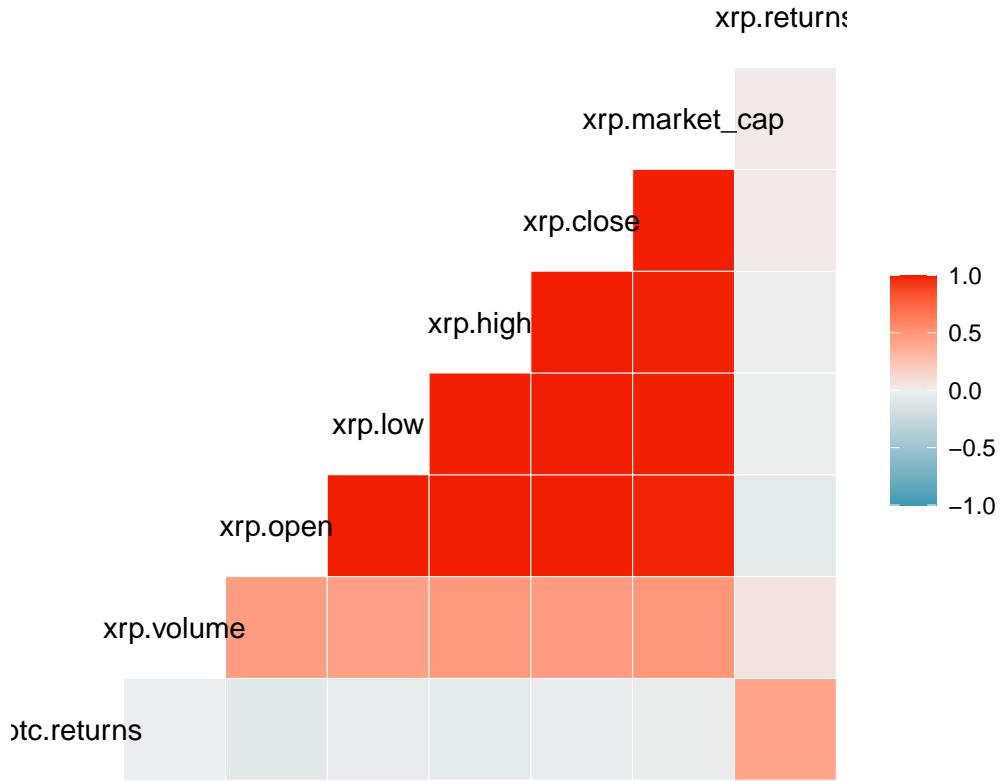
There is no correlation between BTC's returns and USDT's data.

```
# 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)
```

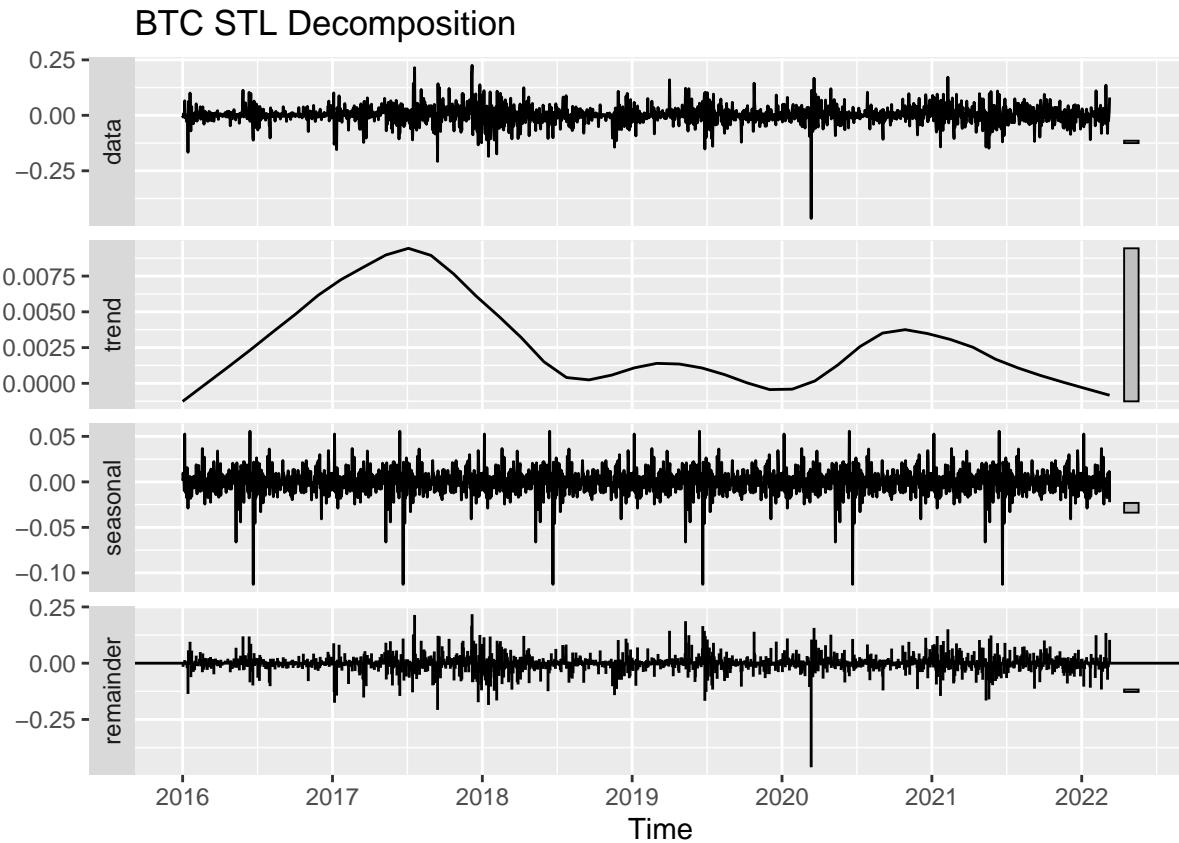


From this, we can conclude that XRP's and ETH's returns are potentially appropriate regressors for forecasting BTC's returns, and we can incorporate them into our models.

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 = 0.001589:  log likelihood = 4074.11
## AIC=-8144.22   AICc=-8144.21   BIC=-8132.77
##
## Training set error measures:
```

```

##               ME         RMSE        MAE      MPE MAPE        MASE        ACF1
## Training set 1.864315e-15 0.03985768 0.02606193 -Inf   Inf 0.6144467 -0.0264022

```

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.

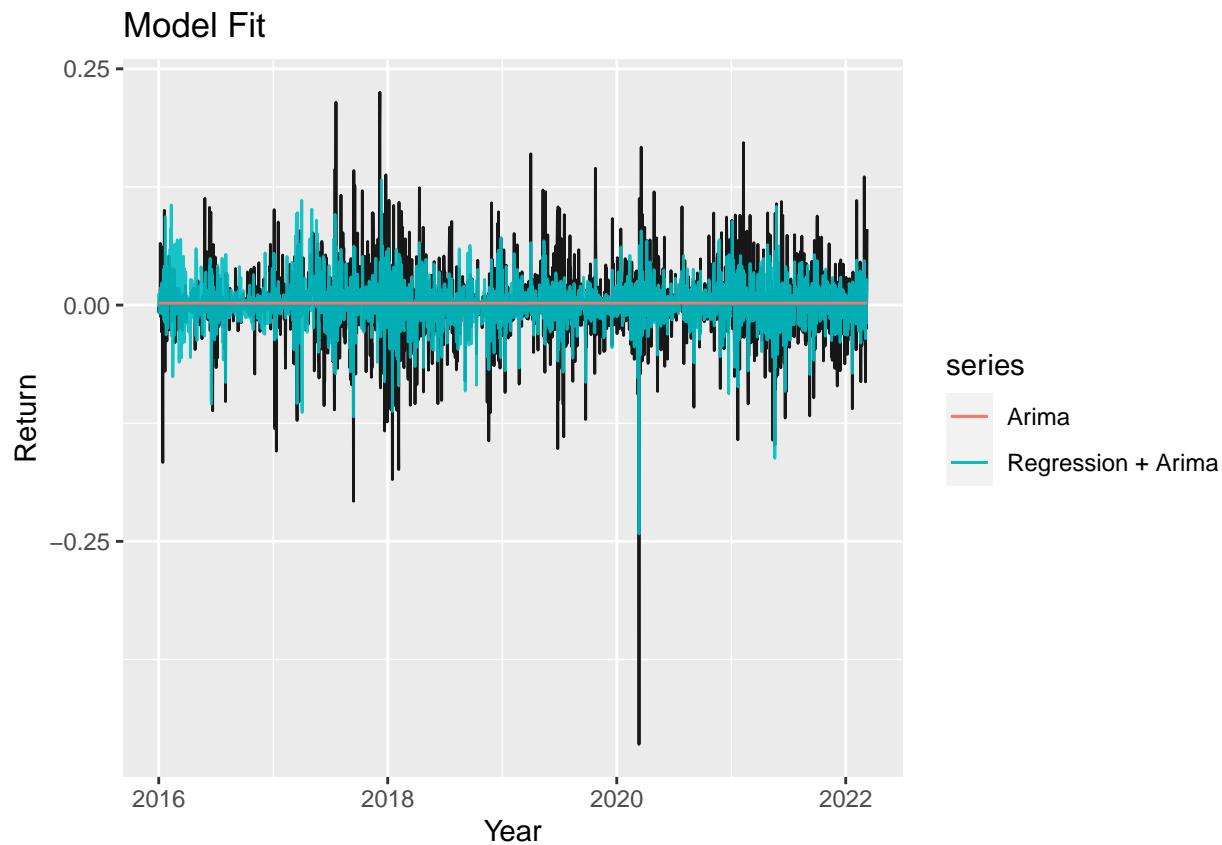
```

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
##            0.0604  0.3532  0.1215
## s.e.  0.0214  0.0124  0.0103
##
## sigma^2 = 0.0009569: log likelihood = 4648.2
## AIC=-9288.39   AICc=-9288.38   BIC=-9265.5
##
## Training set error measures:
##               ME         RMSE        MAE      MPE MAPE        MASE
## Training set 0.0004887037 0.03091313 0.02056979 -Inf   Inf 0.5266715
##                         ACF1
## Training set -0.0008239977

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")

```

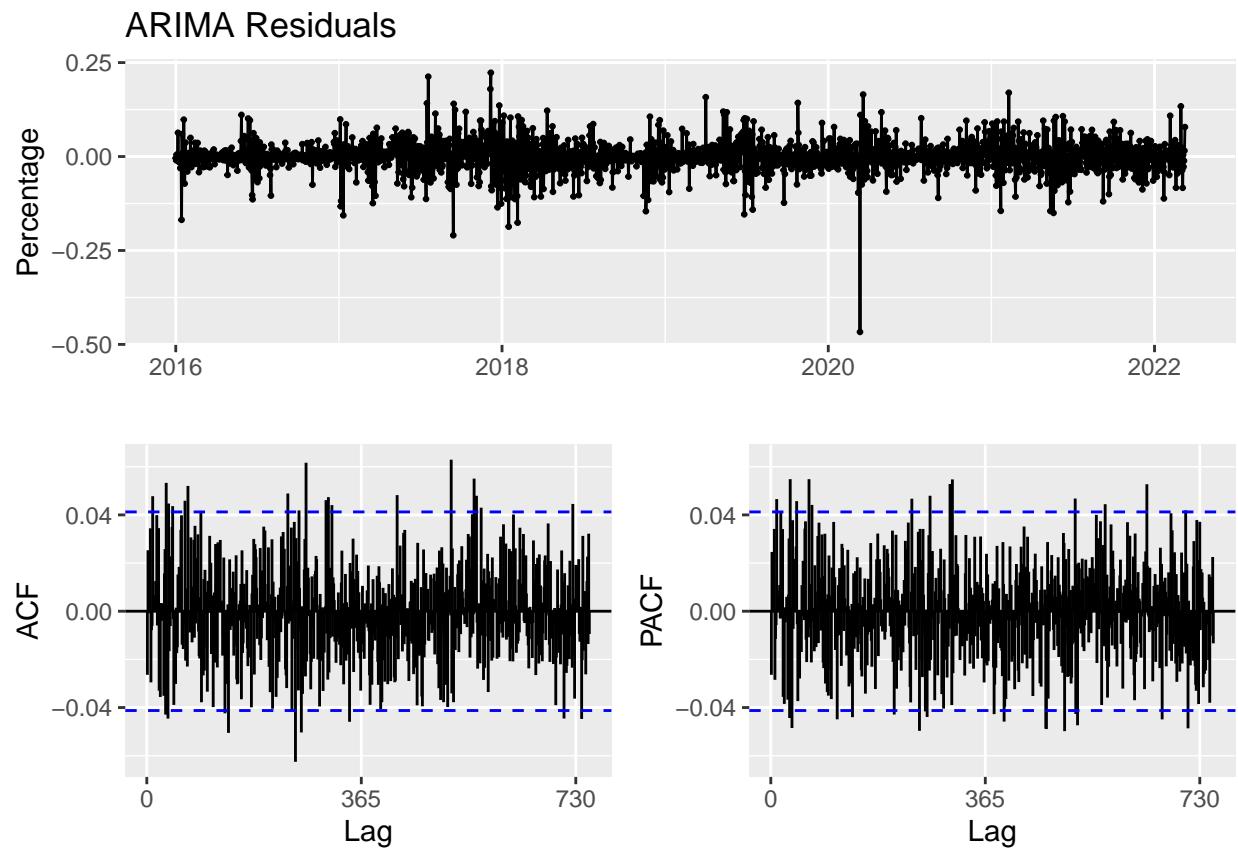


ACF/PACF of Residuals

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.

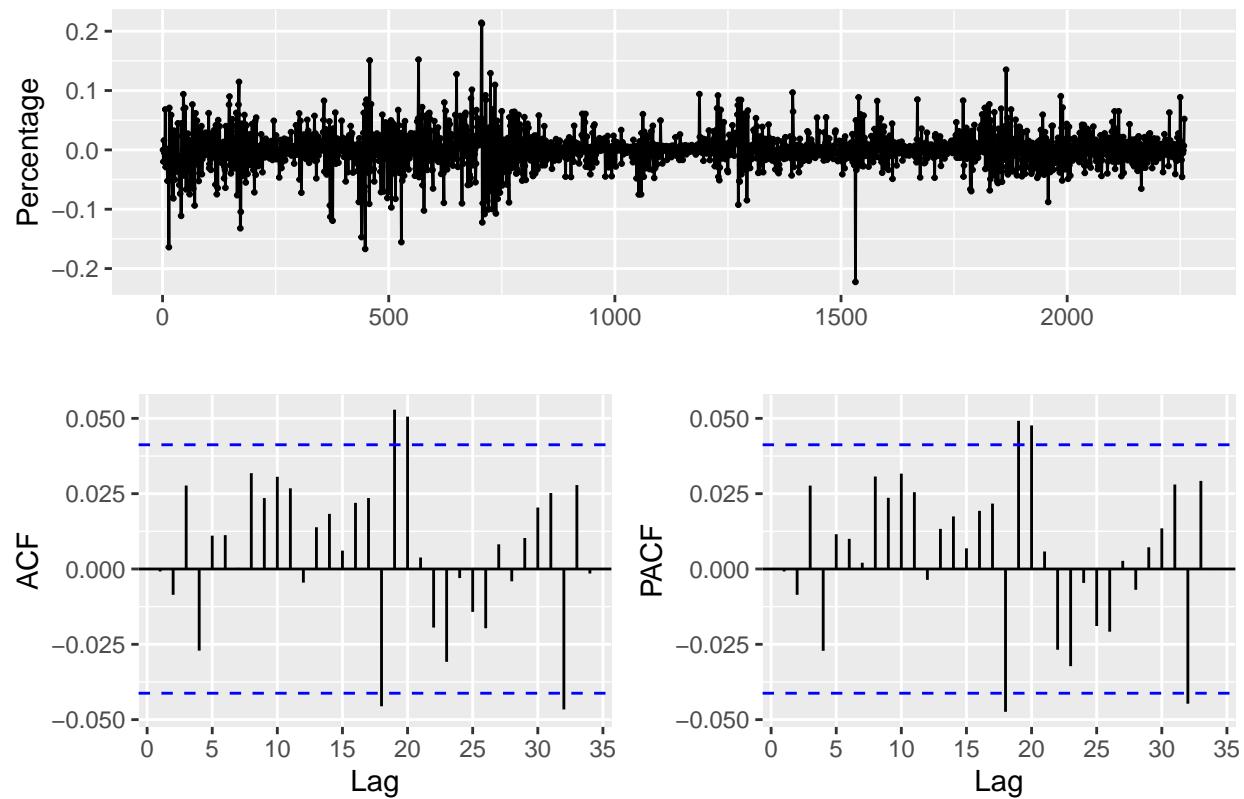
The plot of the residuals appears to retain some structure which can be modeled. However, the magnitude of the autocorrelation spikes seen on the ACF and PACF plots suggest that this structure may not be very significant. Therefore, the regression + ARIMA model fits the data quite well.

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



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

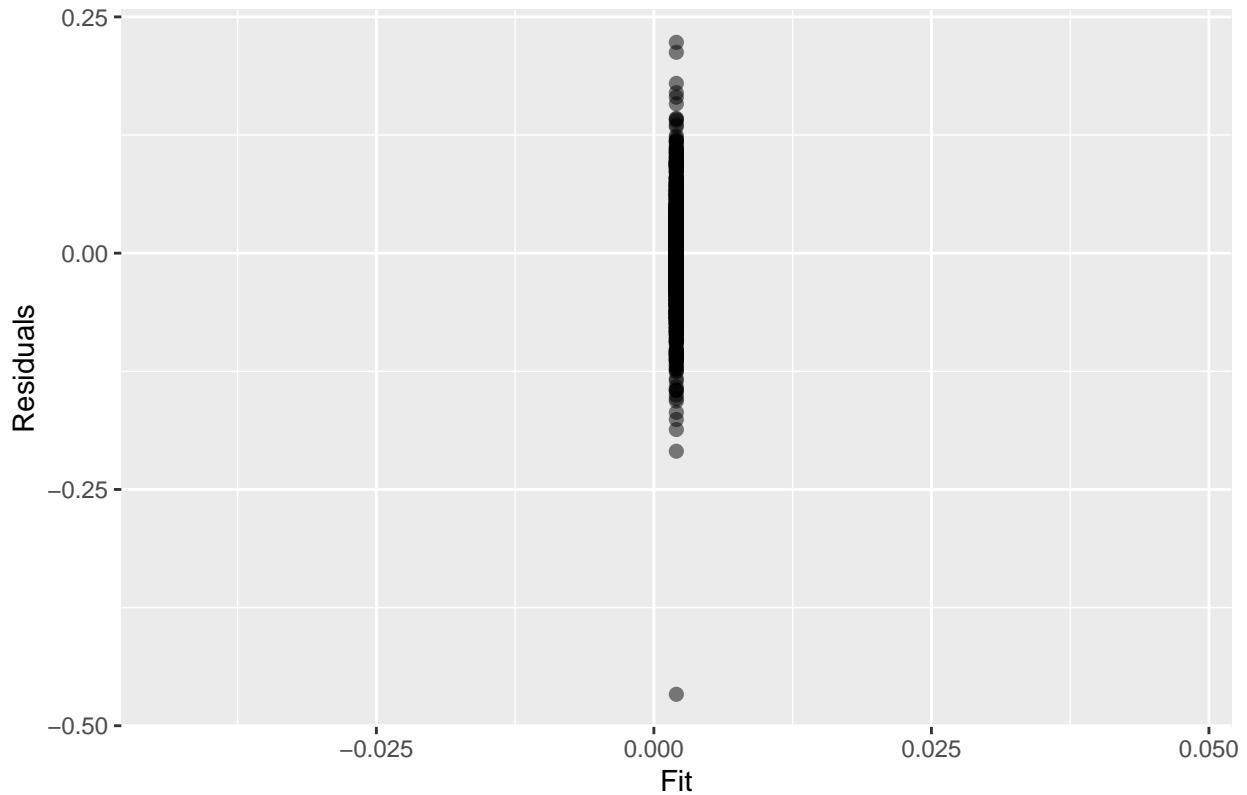
Regression + ARIMA Residuals



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

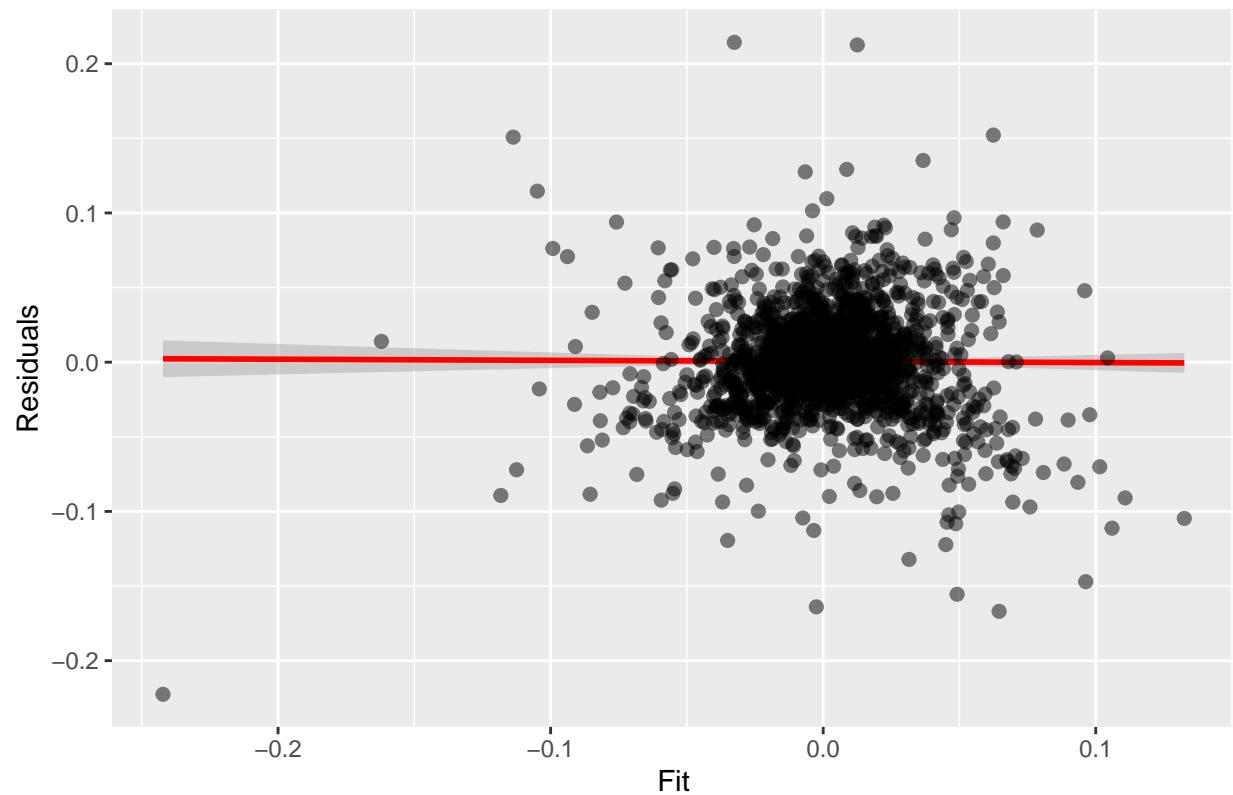
```
plot_resid_fit <- function(x,y,title) {
  data = data.frame(x = x, y = y)
  ggplot(data, aes(x = x, y = y)) +
    geom_smooth(method = "lm", se=T, col = "red",
               formula = y ~ x, show.legend = T) +
    scale_x_continuous() + scale_y_continuous() +
    geom_point(alpha = 0.5, size = 2,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

ARIMA Residuals vs Fit



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

Regression + ARIMA Residuals vs Fit

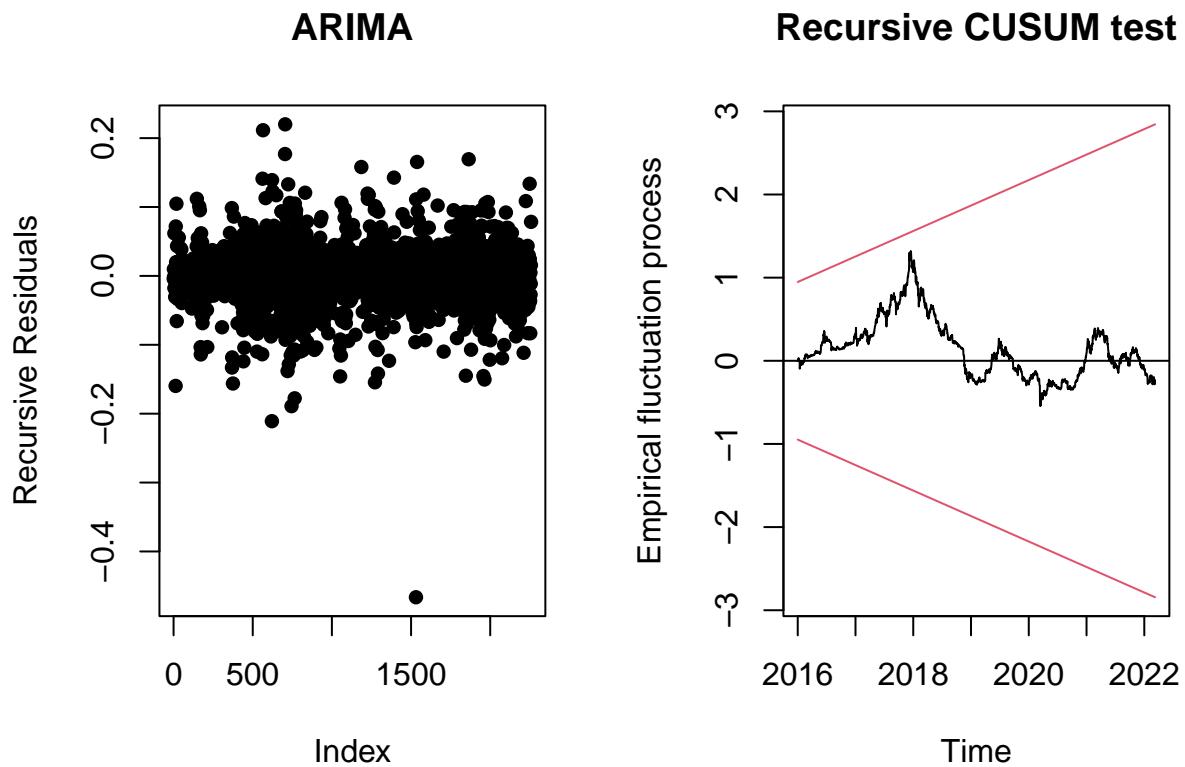


There is no linear relationship between the fit and the residuals. Suggesting our model did quite well.

Recursive Residuals

According to our recursive CUSUM test, the fluctuation process remains well within the confidence bands meaning that there is no structural break in our ARIMA model.

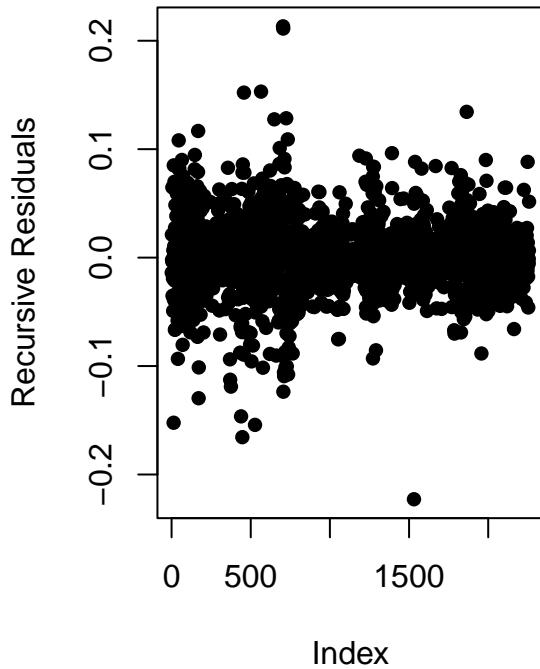
```
par(mfrow=c(1,2))
y = recresid(arima_btc$res ~ 1)
plot(y, pch = 16, main = "ARIMA", ylab = "Recursive Residuals")
plot(efp(arima_btc$residuals^1, type = "Rec-CUSUM"))
```



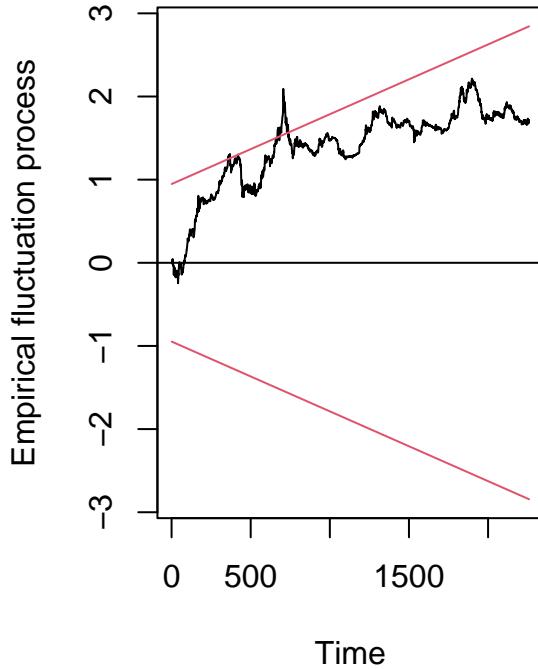
The recursive residuals are scattered around 0 and do not show any dynamics, reaffirming that the Regression + ARIMA Fit is a good model. However, the process does protrude the confidence bands, suggesting that there may be structural breaks in the model.

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

Regression + ARIMA



Recursive CUSUM test



E) ETS, Regression + ETS

We will utilise monthly data instead of daily data to build the ETS models otherwise they don't work.

Fitting the Model

The ETS model suggested by R is an additive one. The RMSE of this model is larger than the Regression + ARIMA model developed above. This suggests its performance may not be as good.

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

```
## ETS(A,N,N)
##
## Call:
##   ets(y = btc_ts2)
##
##   Smoothing parameters:
##     alpha = 0.255
##
##   Initial states:
##     l = 0.0487
##
```

```

##   sigma:  0.2041
##
##      AIC      AICc      BIC
## 87.28572 87.62858 94.19792
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.003857056 0.2013311 0.1563642 94.9544 352.1959 0.6131027
##                  ACF1
## Training set 0.2109265

```

Now we fit a Regression + ETS model to the data. It has a lower RMSE than the vanilla ETS model. This suggests that the relationship between BTC and ETH is important in modelling the return dynamics of BTC. The ETS component of this model is also an additive smoothing model.

```

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
## 57.57030 57.91315 64.48249
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 7.506215e-06 0.1647074 0.1271874 100.0144 100.0144 0.6513799
##                  ACF1
## Training set 0.3390848

```

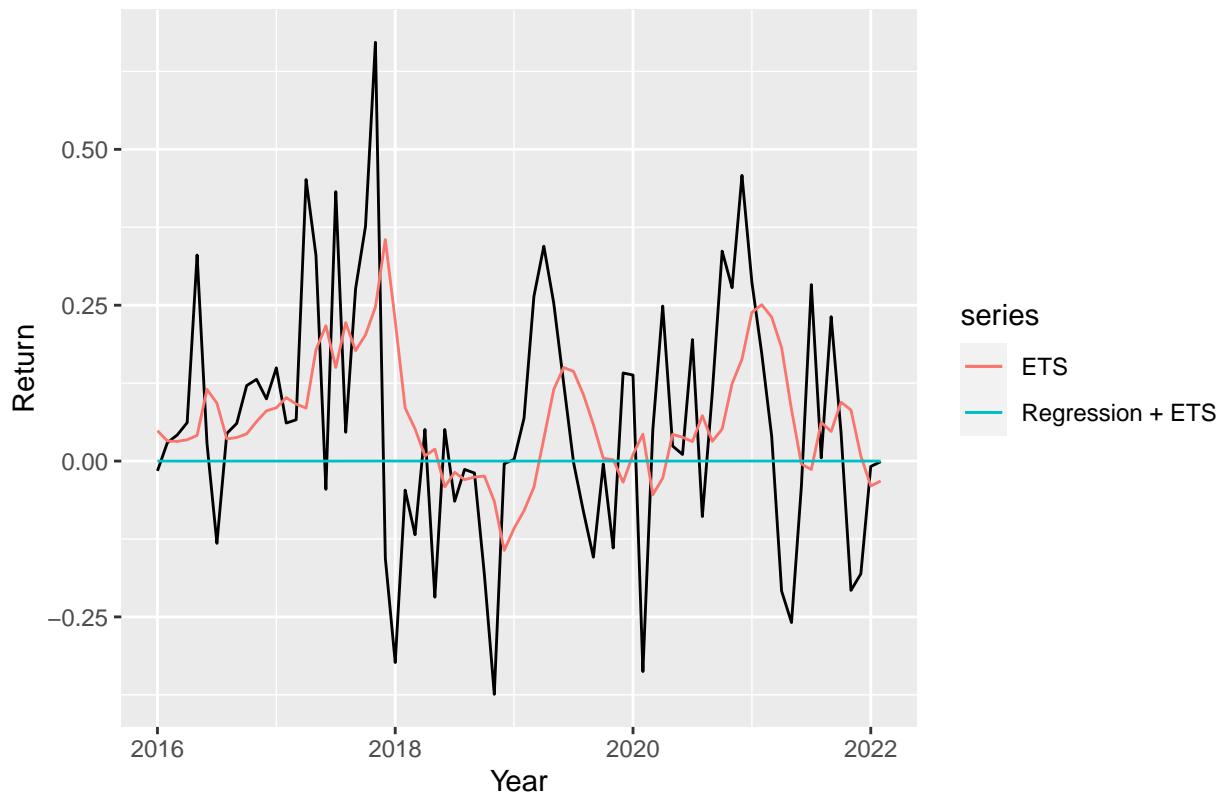
Looking at the actual fit of the data, we can see that although the Regression + ETS model looks better analytically, the plain ETS model provides much more information than the Regression + ETS model. Whereas the ETS model attempts to map to the data, the Regression + ETS model is simply a straight line.

```

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")

```

Model Fit

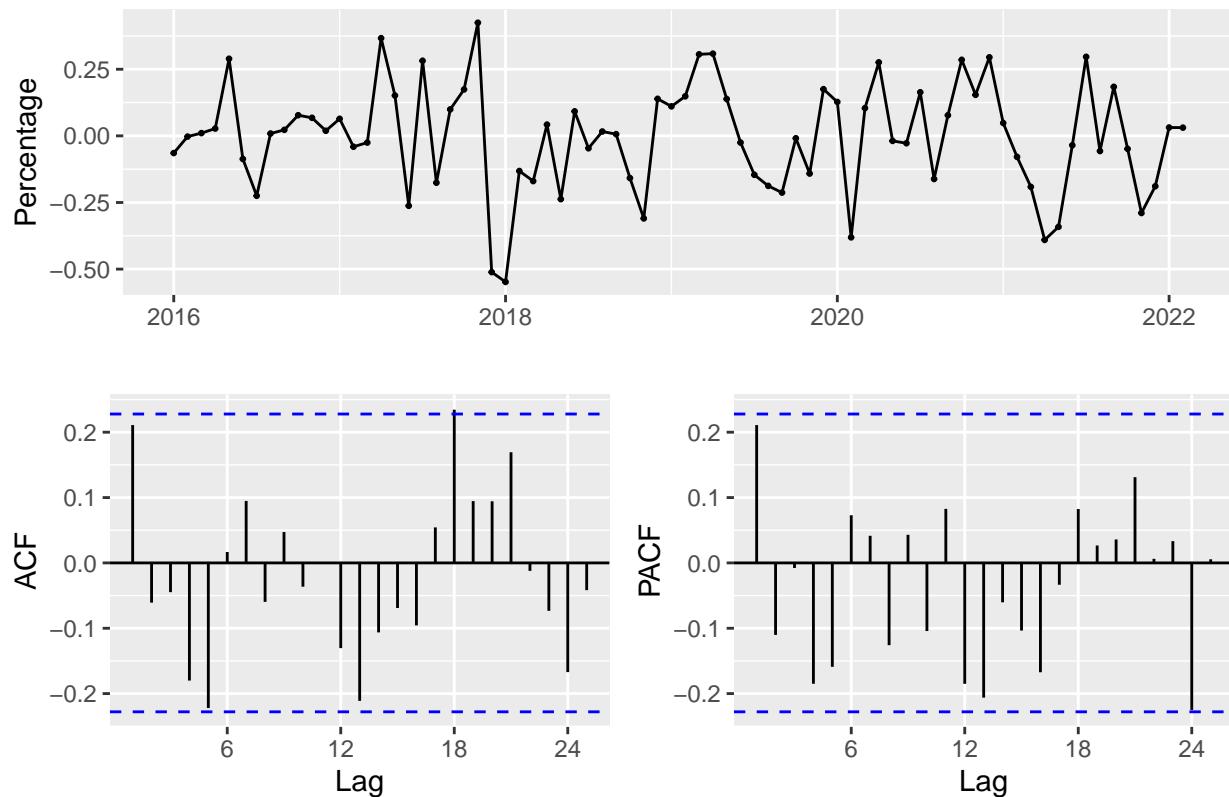


ACF/PACF of Residuals

The ACF and PACF of residuals for both models show little structure. Although it would be a stretch to say that the models fit well to the data, the residual ACF and PACF plots at least suggest that the ETS models are reasonable fits.

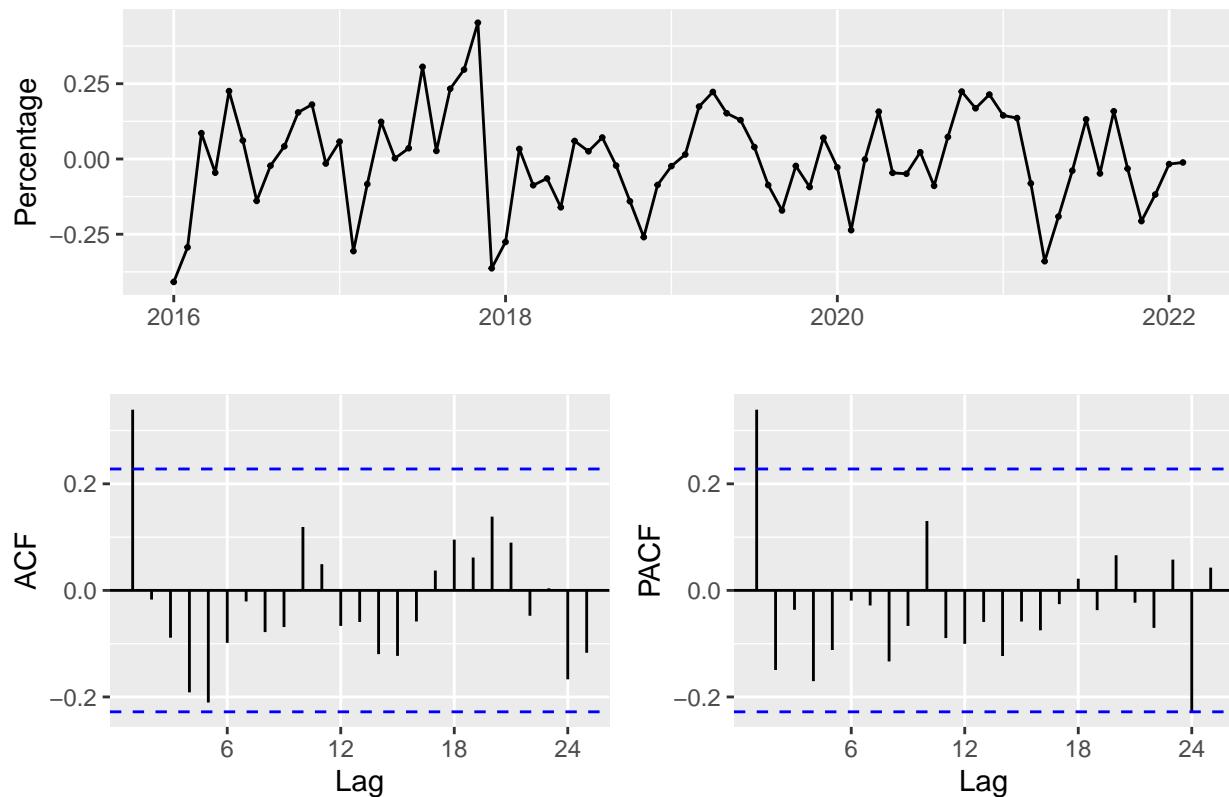
```
ggtstdisplay(resid(ets_btc),main = "ETS Residuals", ylab = "Percentage")
```

ETS Residuals



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

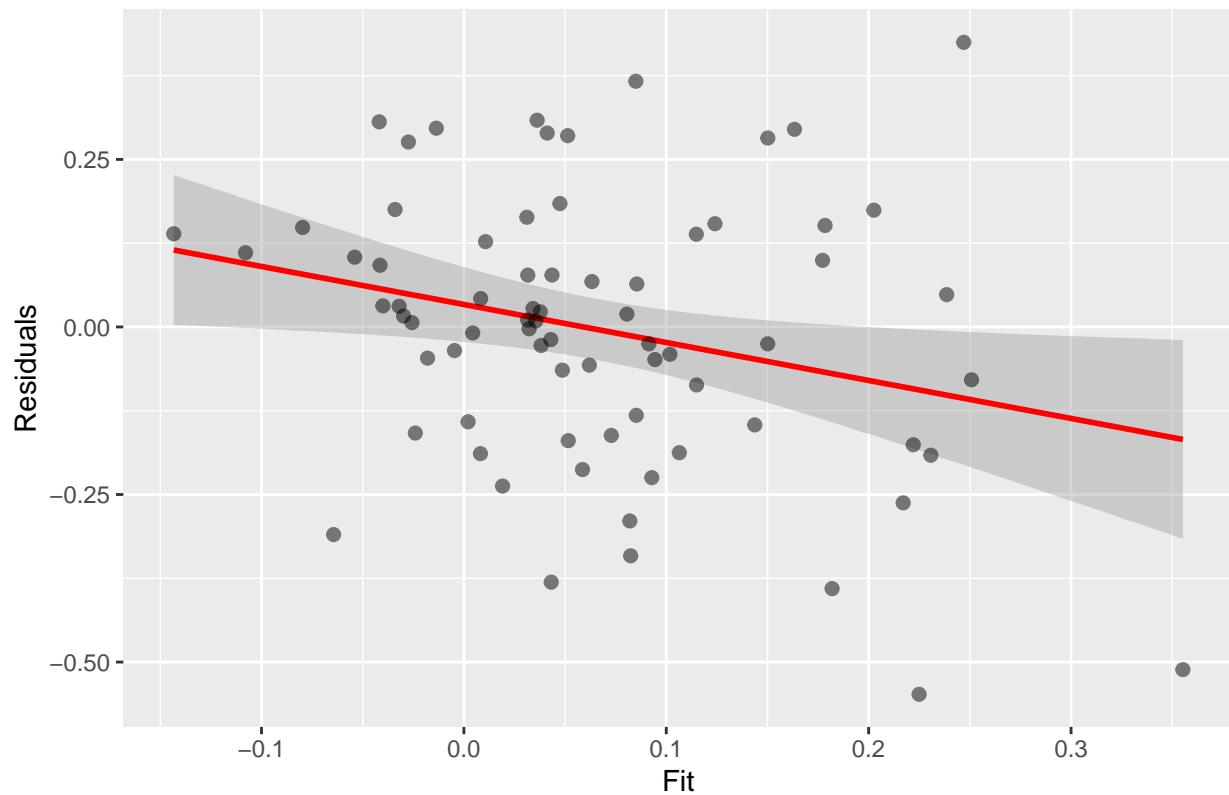
Regression + ETS Residuals



The Residuals vs Fitted plots for both ETS models show that there is a slight negative linear correlation between the fit and the data. Therefore, there are still dynamics that can be captured, which the ETS models have not caught on to.

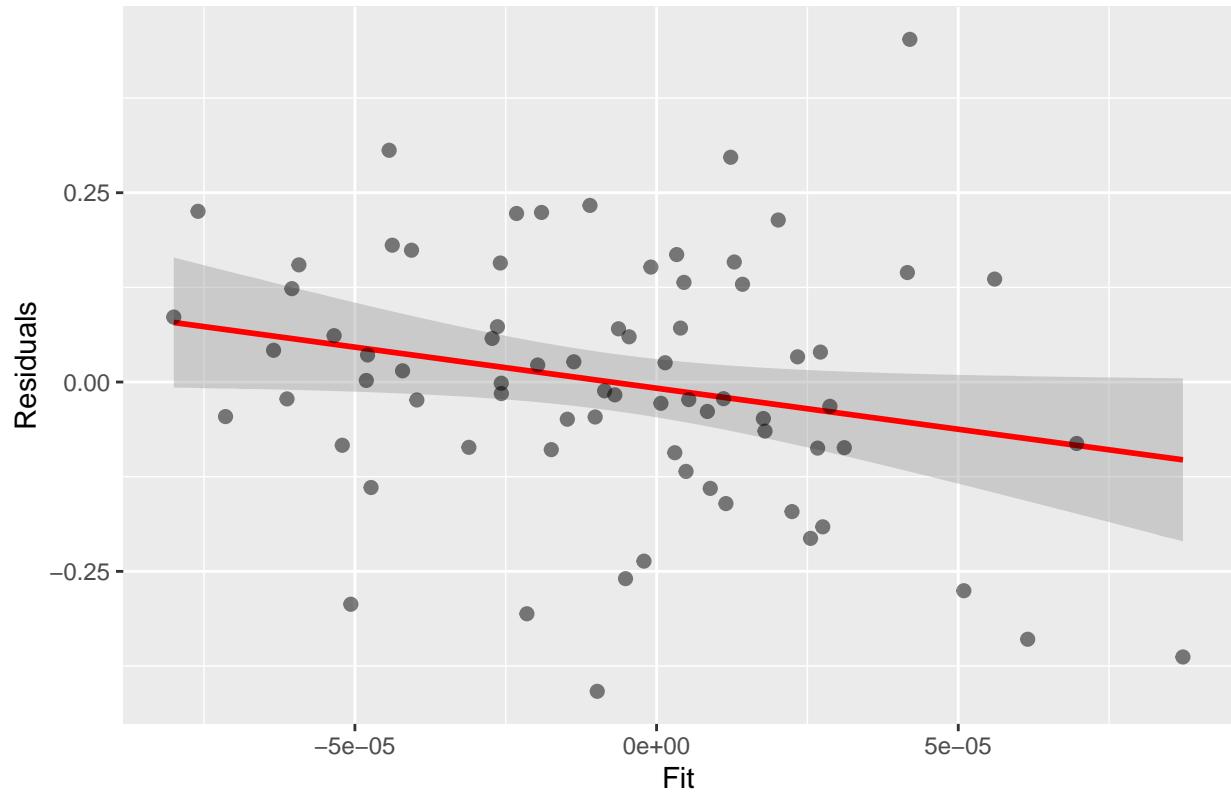
```
plot_resid_fit(fitted(ets_btc), resid(ets_btc), "ETS")
```

ETS Residuals vs Fit



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

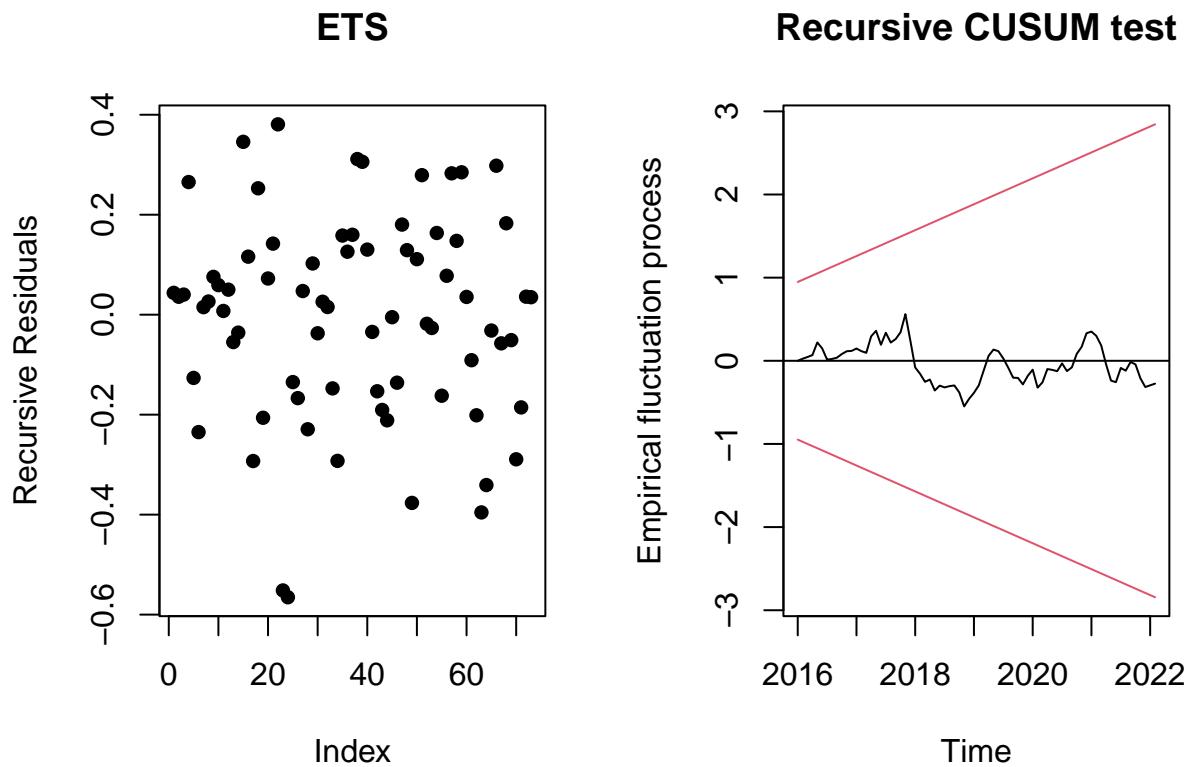
Regression + ETS Residuals vs Fit



Recursive Residuals

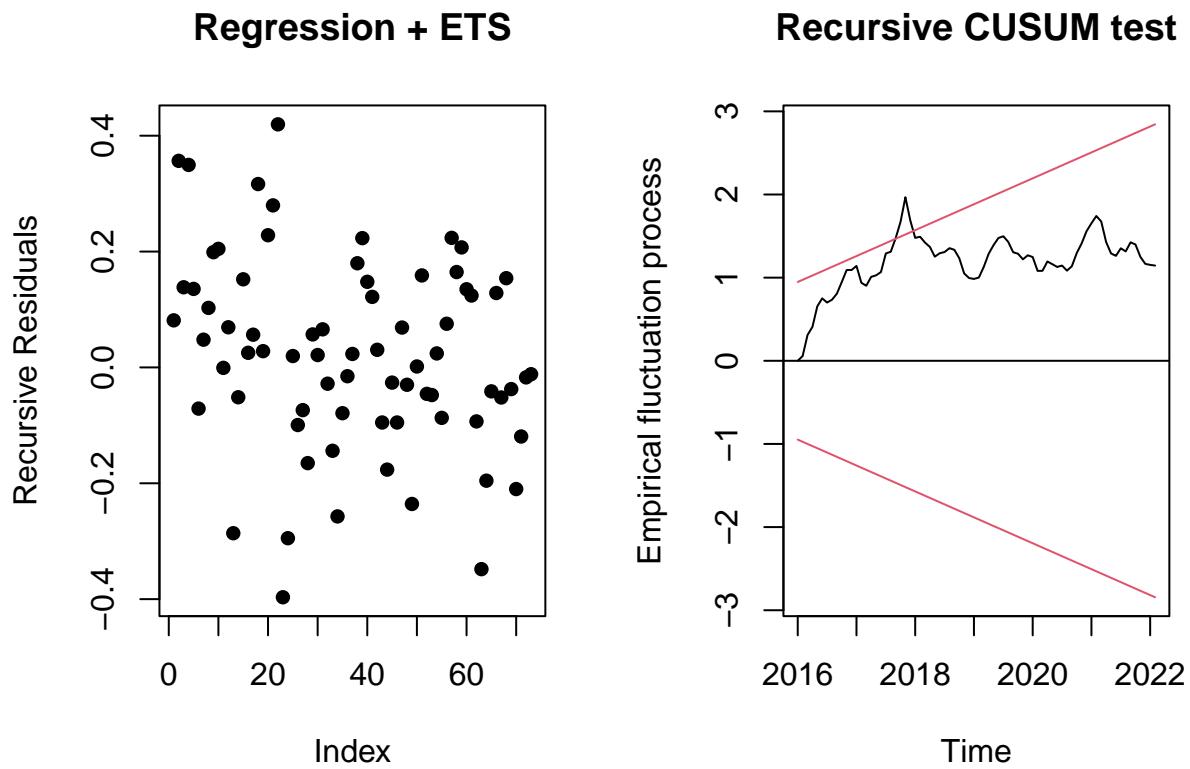
According to our recursive CUSUM test, the fluctuation process remains well within the confidence bands meaning that there is no structural break in our ETS model.

```
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"))
```



The recursive residuals are scattered around 0 and do not show any dynamics, reaffirming that the Regression + ARIMA Fit is a good model. However, the process does protrude the confidence bands, suggesting that there may be structural breaks in the model.

```
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

We will utilise monthly data instead of daily data to build the Holt-Winters models otherwise they don't work.

Fitting the Model

The Holt-Winters model with additive seasonality has a slightly lower RMSE than the pure ETS model above, so it may be a better predictive model. Utilising seasonality can help us capture any seasonal variations in the monthly returns.

```
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 = 0.0091
##   beta  = 1e-04
##   gamma = 0.0042
##
## Initial states:
##   l = 0.0047
##   b = 5e-04
##   s = 0.0182 0.0205 0.098 -0.0216 -0.0449 0.049
##           -0.0177 -0.0178 0.0893 -0.0164 -0.1003 -0.0562
##
## sigma: 0.2251
##
##          AIC      AICc      BIC
## 113.7918 124.7204 152.9609
##
## Error measures:
##          ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.02046556 0.1993091 0.1565569 168.5933 281.99 0.6138586 0.3990505
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## Mar 2022 0.05304361 -0.2354691 0.3415563 -0.3881985 0.4942858
## Apr 2022 0.15928472 -0.1292401 0.4478095 -0.2819759 0.6005453
## May 2022 0.05349127 -0.2350459 0.3420284 -0.3877883 0.4947708
## Jun 2022 0.05283631 -0.2357135 0.3413861 -0.3884625 0.4941351
## Jul 2022 0.12094101 -0.1676217 0.4095037 -0.3203775 0.5622596
## Aug 2022 0.02675945 -0.2618164 0.3153353 -0.4145792 0.4680981
## Sep 2022 0.05252283 -0.2360665 0.3411121 -0.3888364 0.4938821
## Oct 2022 0.17059691 -0.1180061 0.4592000 -0.2707834 0.6119772
## Nov 2022 0.09423547 -0.1943816 0.3828525 -0.3471662 0.5356372
## Dec 2022 0.09272259 -0.1959088 0.3813540 -0.3487010 0.5341462
## Jan 2023 0.02046466 -0.2681813 0.3091106 -0.4209812 0.4619106
## Feb 2023 -0.02295119 -0.3116120 0.2657097 -0.4644199 0.4185175
## Mar 2023 0.06101893 -0.2276719 0.3497098 -0.3804956 0.5025335
## Apr 2023 0.16726004 -0.1214463 0.4559664 -0.2742782 0.6087983
## May 2023 0.06146659 -0.2272555 0.3501887 -0.3800958 0.5030290
## Jun 2023 0.06081163 -0.2279266 0.3495498 -0.3807753 0.5023986
## Jul 2023 0.12891633 -0.1598383 0.4176709 -0.3126957 0.5705284
## Aug 2023 0.03473477 -0.2540365 0.3235061 -0.4069028 0.4763724
## Sep 2023 0.06049816 -0.2282902 0.3492865 -0.3811655 0.5021618
## Oct 2023 0.17857223 -0.1102334 0.4673779 -0.2631179 0.6202624
## Nov 2023 0.10221079 -0.1866125 0.3910341 -0.3395063 0.5439279
## Dec 2023 0.10069791 -0.1881434 0.3895392 -0.3410467 0.5424425
## Jan 2024 0.02843998 -0.2604196 0.3172996 -0.4133326 0.4702126
## Feb 2024 -0.01497587 -0.3038541 0.2739023 -0.4567770 0.4268252

```

The Regression + Holt-Winters model with additive seasonality has a slightly lower RMSE than the pure Holt-Winters model above, so it may be a better. However, it may also give us a flat set of fitted values like the Regression + ETS models. Next, we will graphically evaluate its performance against the pure Holt-Winters model.

```

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
## 57.57030 57.91315 64.48249
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 7.506215e-06 0.1647074 0.1271874 100.0144 100.0144 0.6513799
##          ACF1
## Training set 0.3390848

```

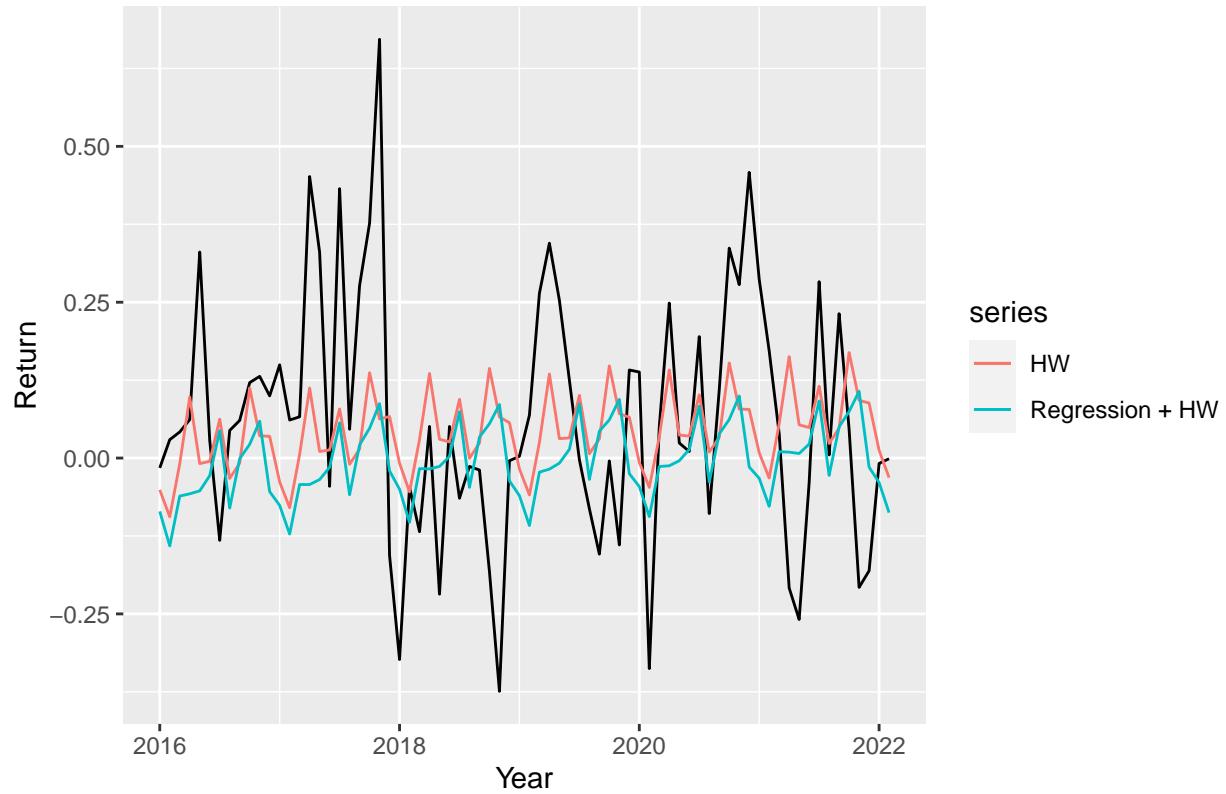
Unlike the Regression + ETS, the Regression + Holt-Winters model shows fluctuating predictions. Since it has a lower RMSE and MAPE than the pure Holt-Winters model, we prefer it.

```

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")

```

Model Fit

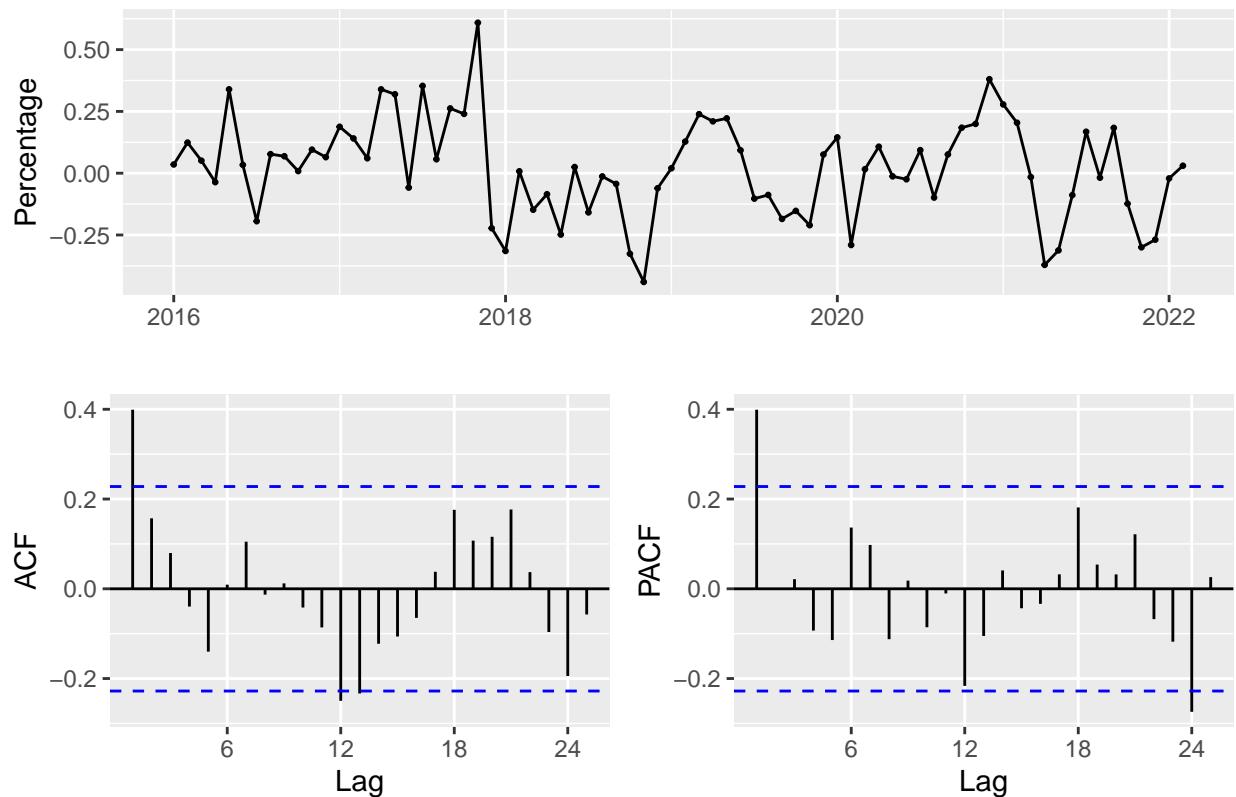


ACF/PACF of Residuals

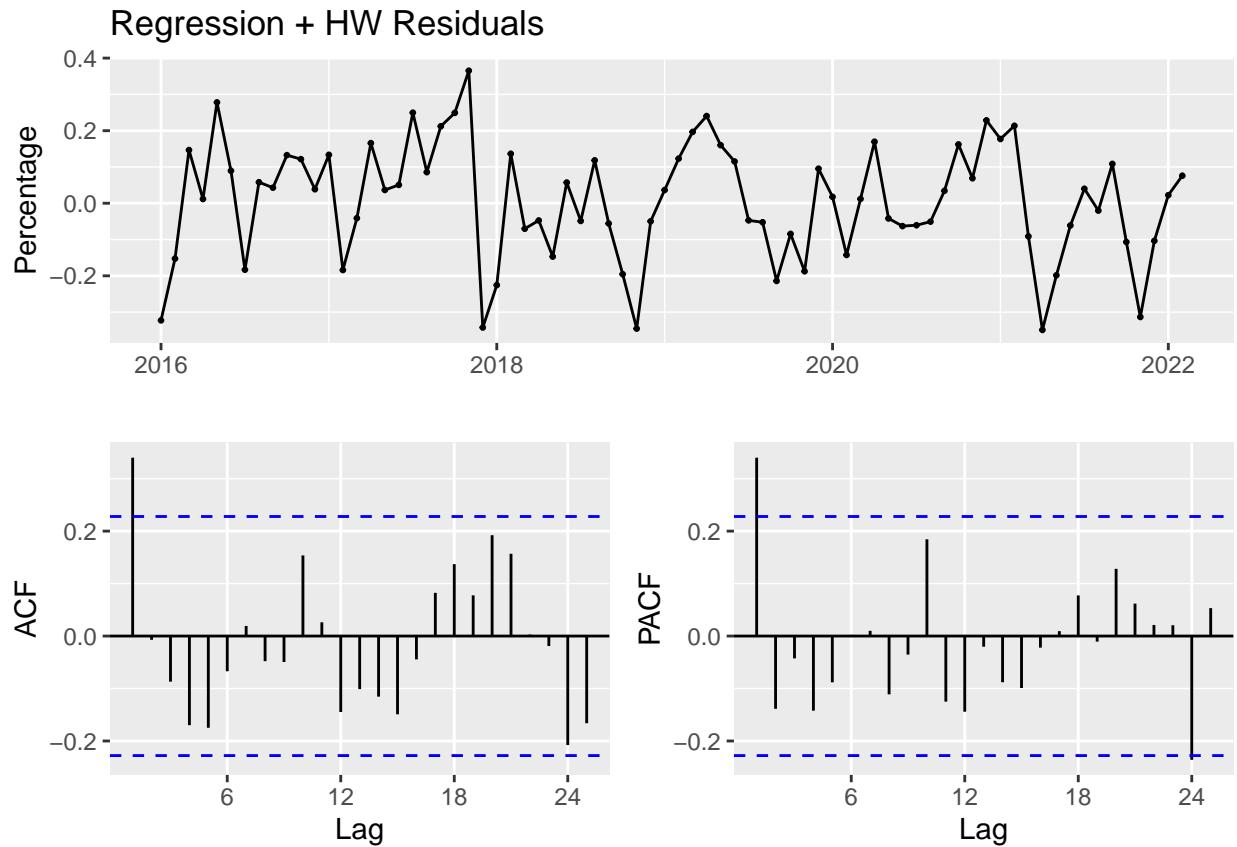
The ACF/PACF plots of the Holt-Winters models show little structure, suggesting the models did well in capturing BTC's monthly returns dynamics.

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

HW Residuals



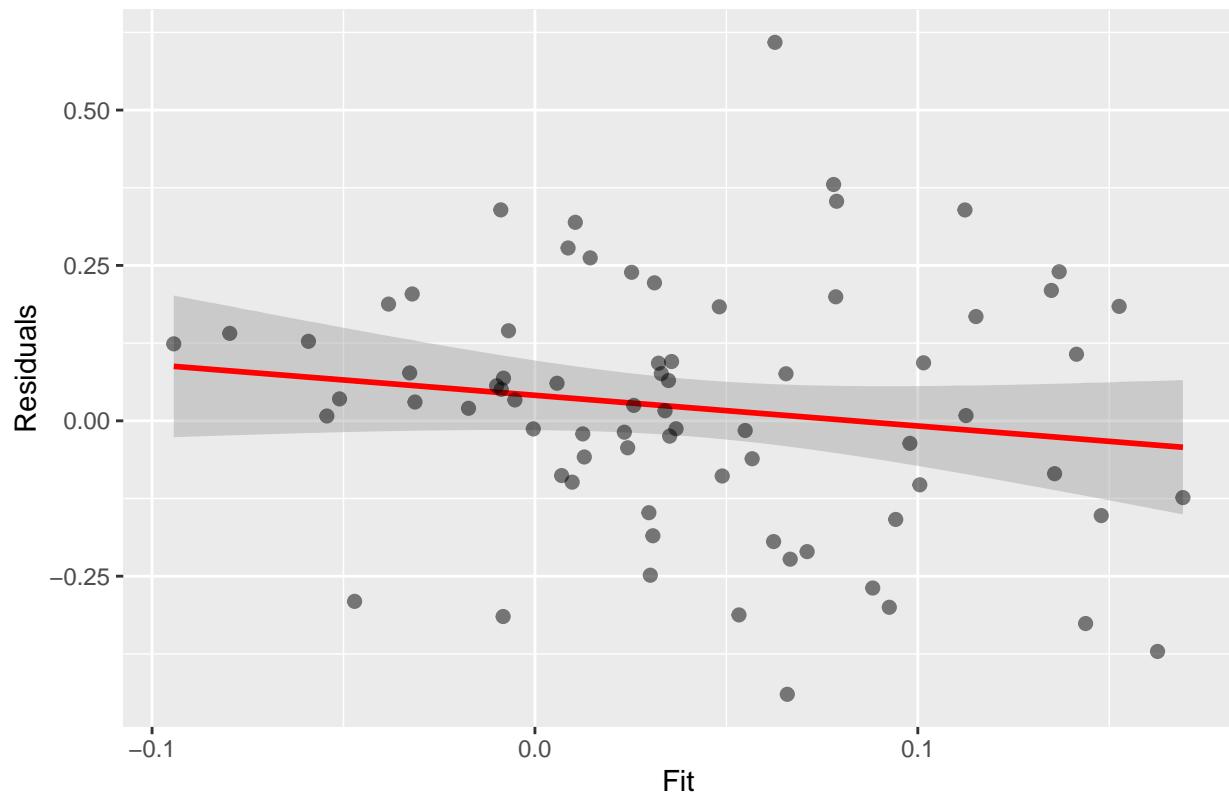
```
ggtstdisplay(resid(reg_hw_btc), main = "Regression + HW Residuals", ylab = "Percentage")
```



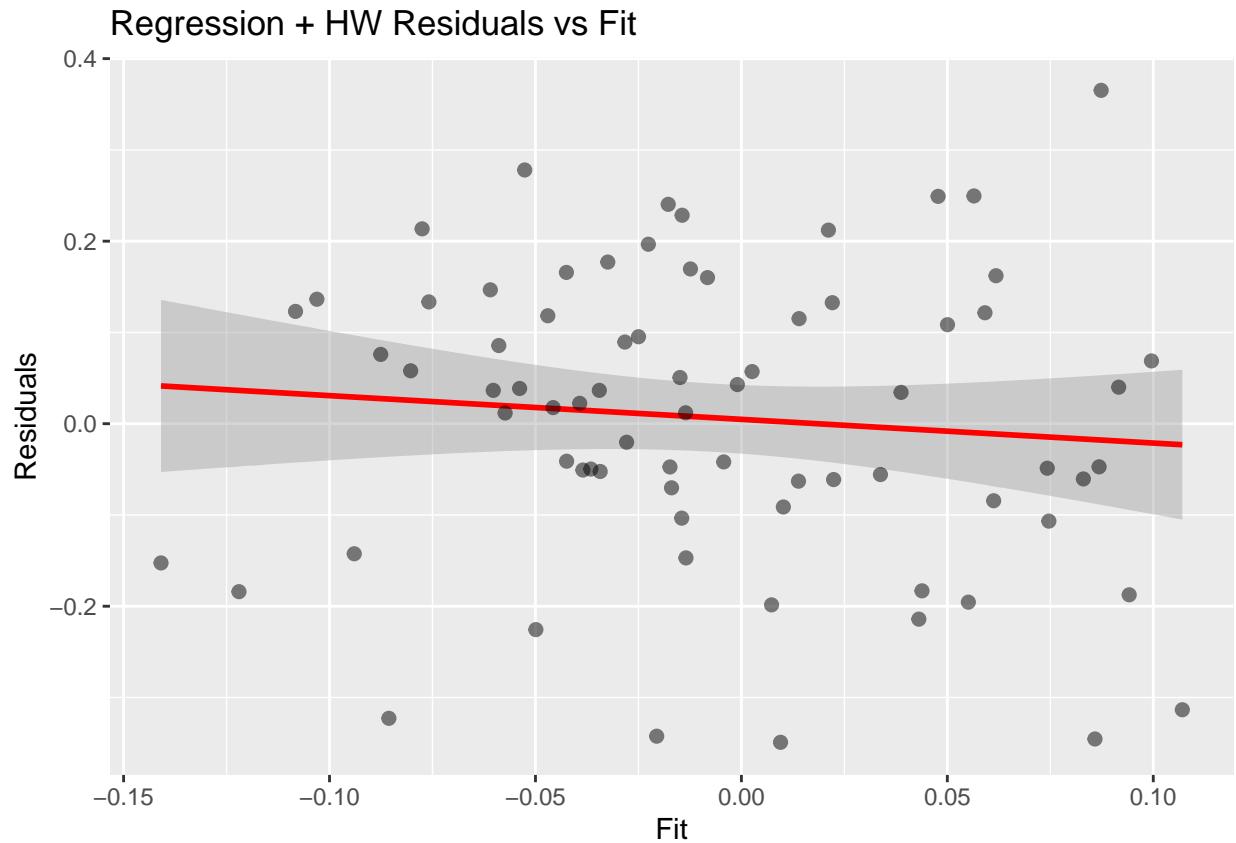
There is a slight negative correlation between the residuals and fitted values, suggesting that both models did not capture some of the return series' dynamics.

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

HW Residuals vs Fit



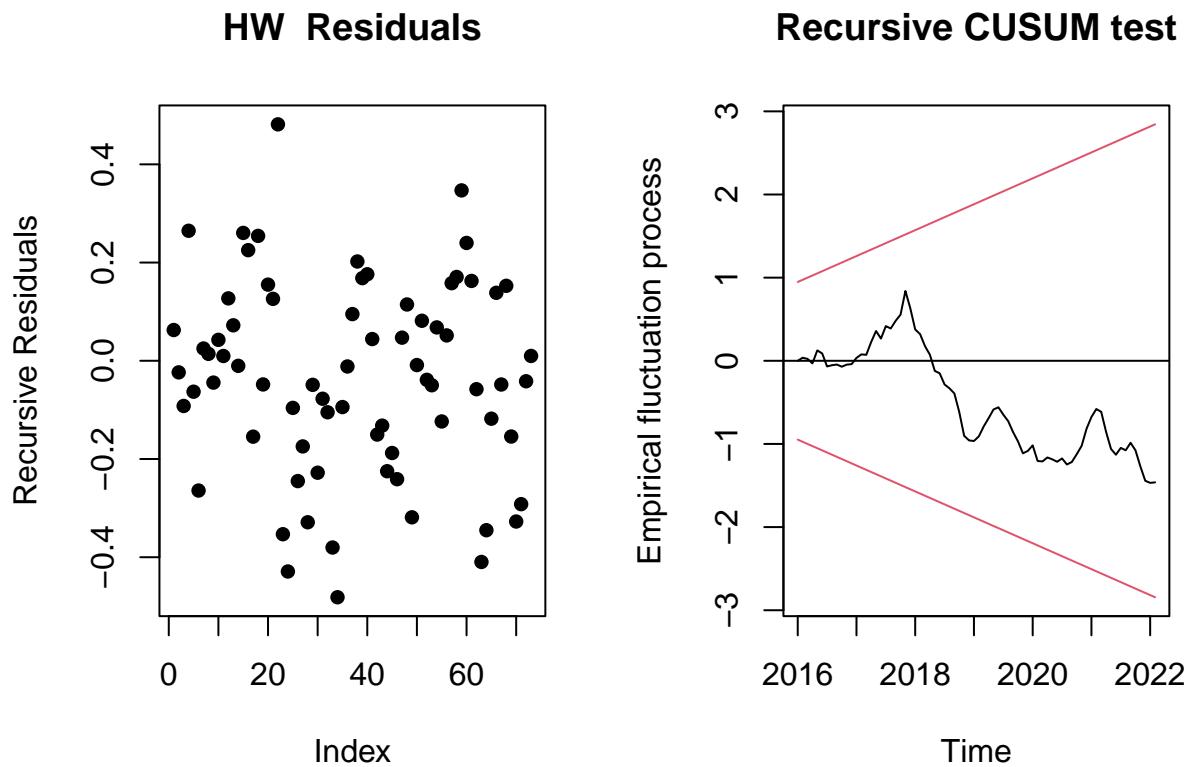
```
plot_resid_fit(fitted(reg_hw_btc), resid(reg_hw_btc), "Regression + HW")
```



Recursive Residuals

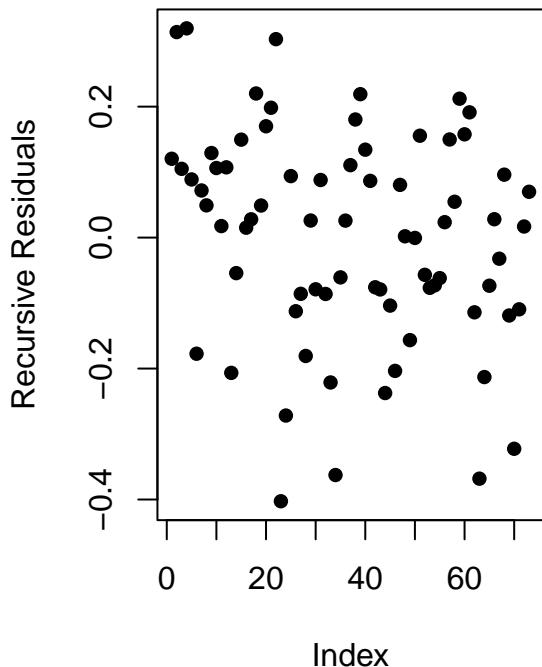
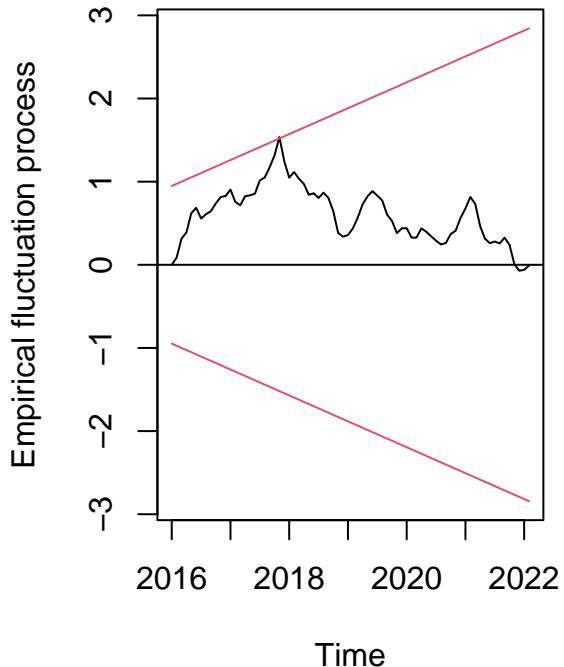
According to our recursive CUSUM test, the fluctuation process remains well within the confidence bands meaning that there is no structural break in our Holt-Winters model.

```
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"))
```



According to our recursive CUSUM test, the fluctuation process remains well within the confidence bands meaning that there is no structural break in our Regression + Holt-Winters model.

```
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"))
```

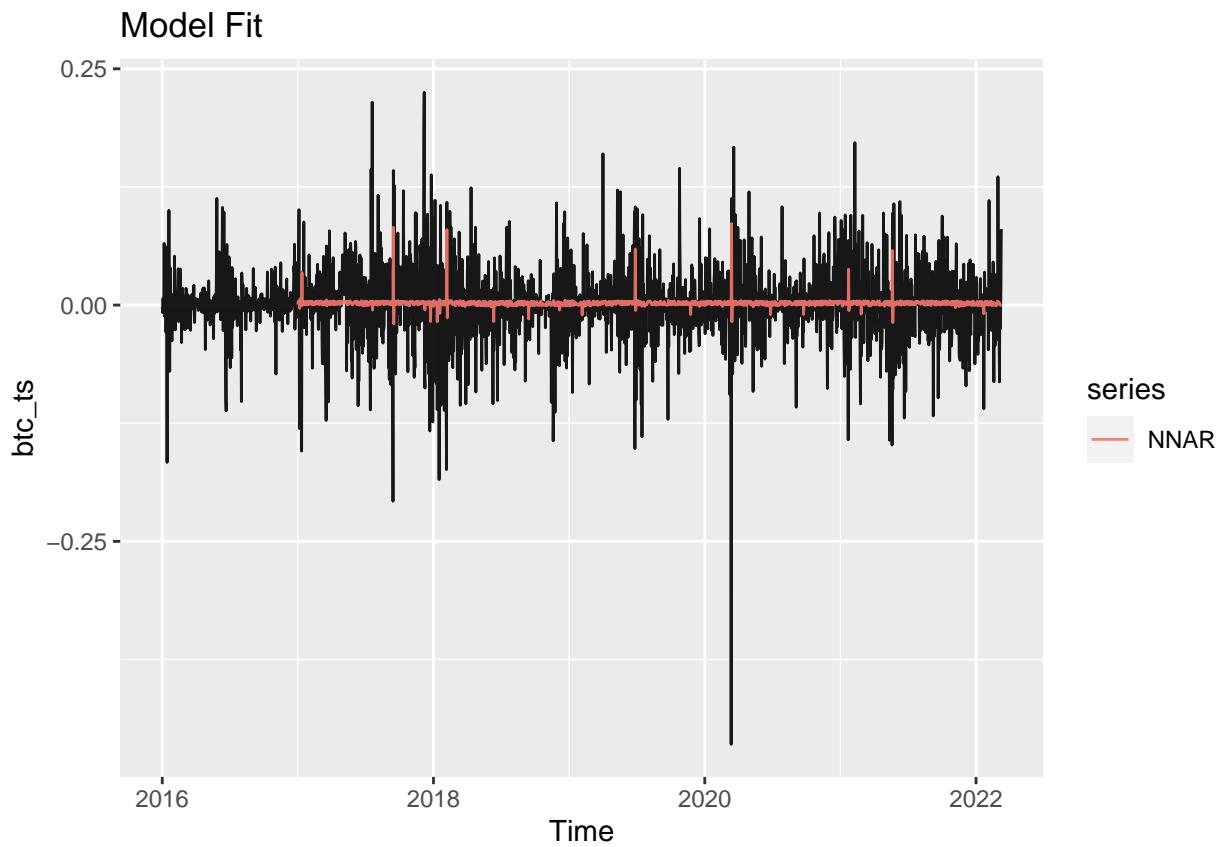
Regression + HW Residuals**Recursive CUSUM test**

G) NNETAR

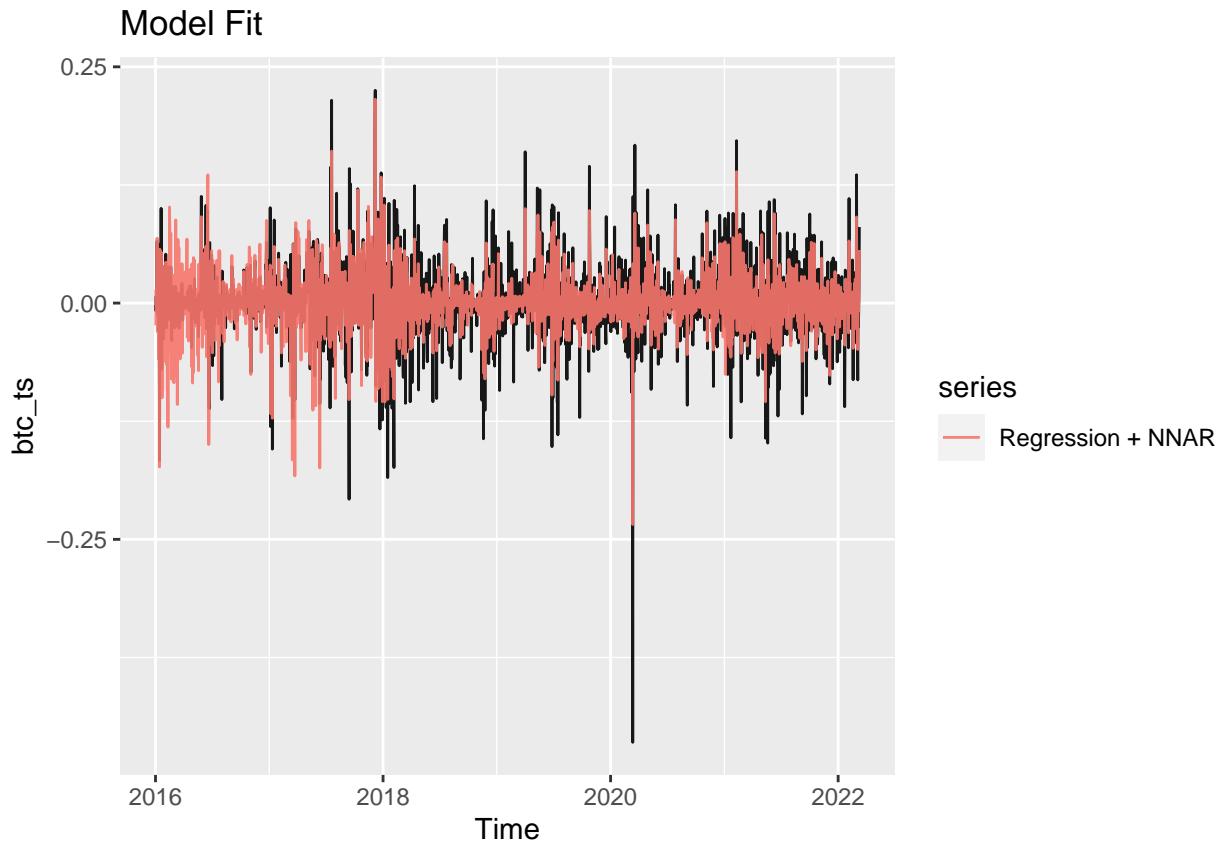
Fitting the Model

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")
```



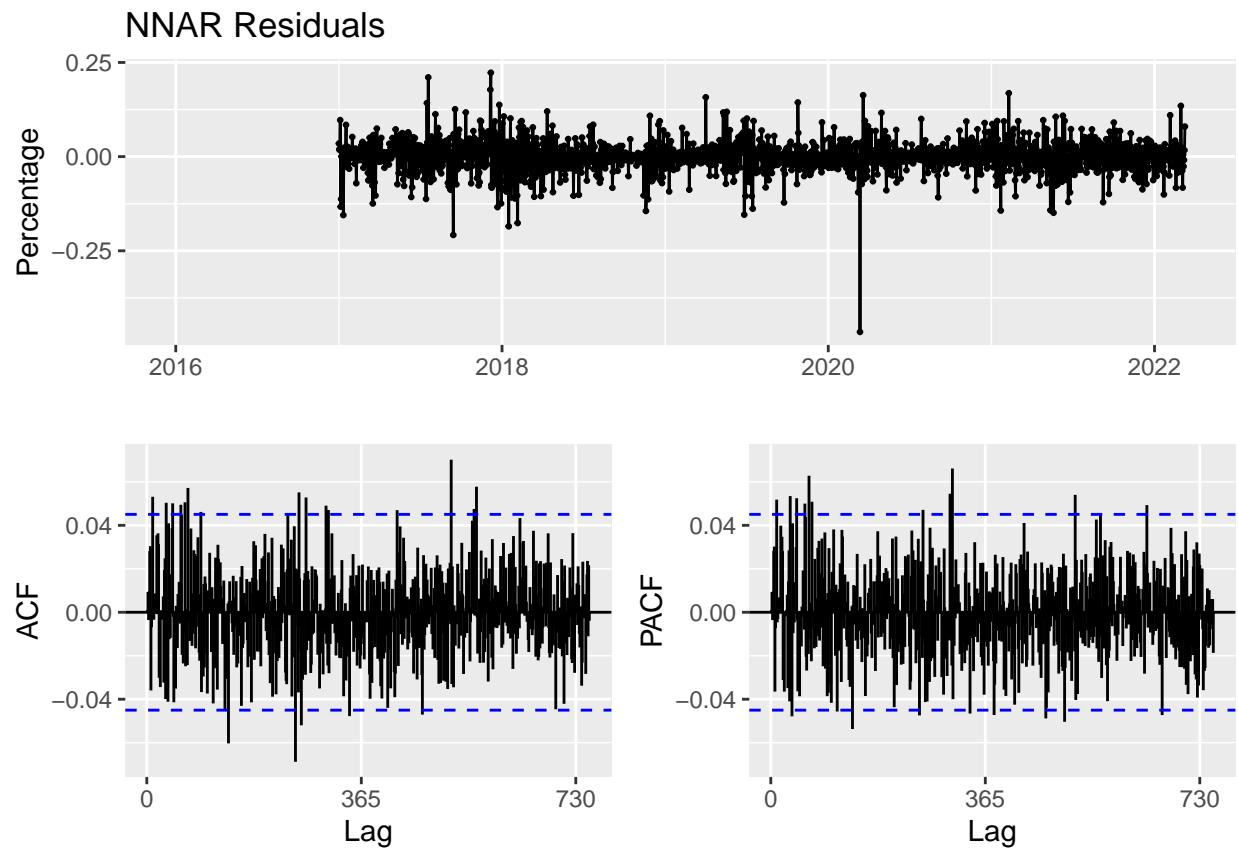
```
autoplot(btc_ts, series="BTC Returns", color="black", alpha = 0.9) +
  autolayer(reg_nnar_fit, series="Regression + NNAR", alpha = 0.9) +
  ggtitle("Model Fit")
```



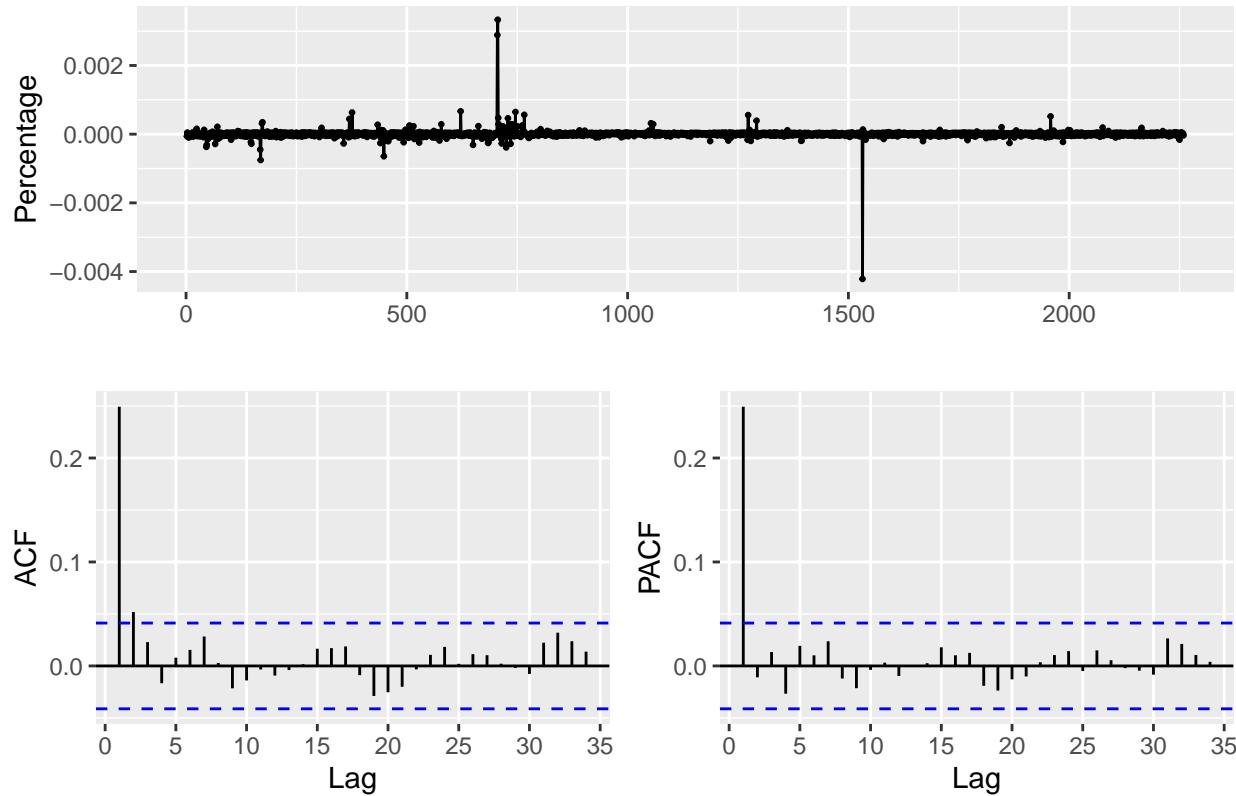
ACF/PACF of Residuals

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.

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



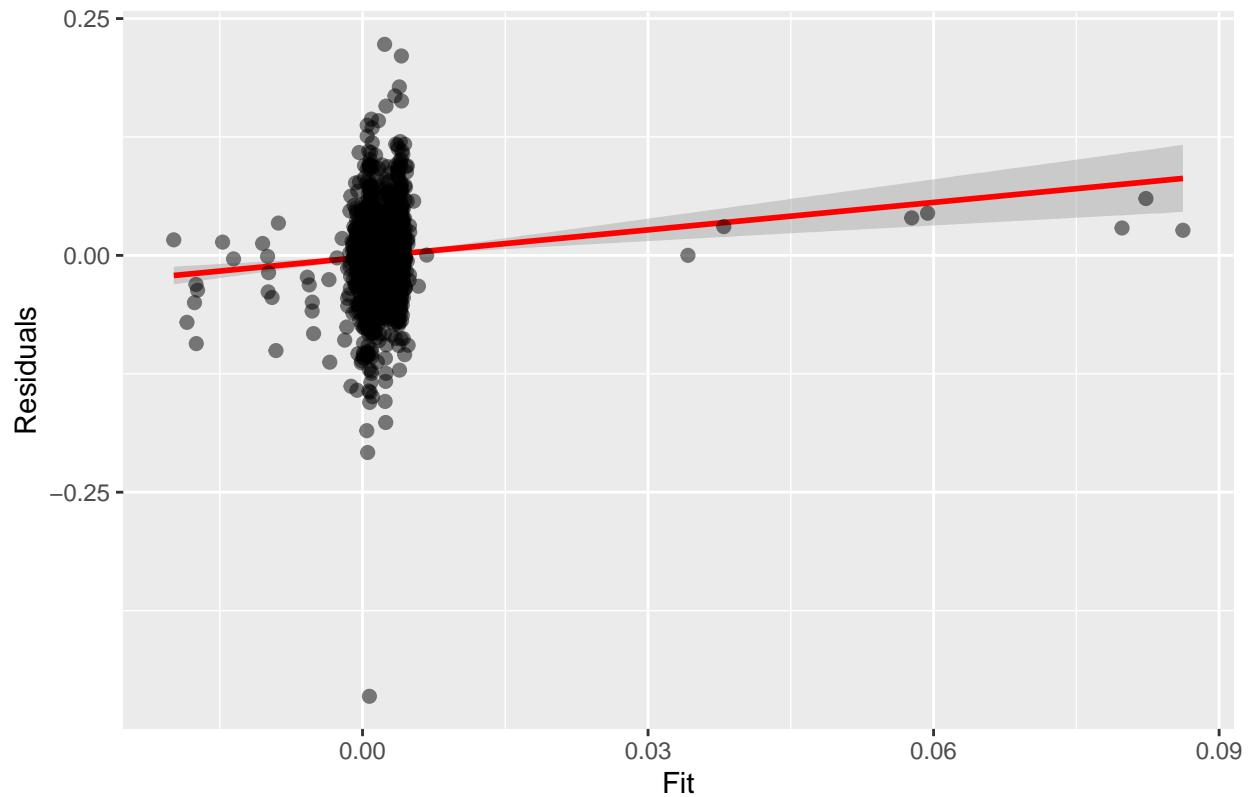
Regression + NNAR Residuals



The NNETAR's Residuals vs Fit plot shows that there is a clear linear correlation between the residuals and the fit. Thus, it is not a strong model. On the other hand, the Linear Regression + NNETAR Residuals vs Fit plot shows practically no relationship between the fitted values and residuals, affirming that it is a good model.

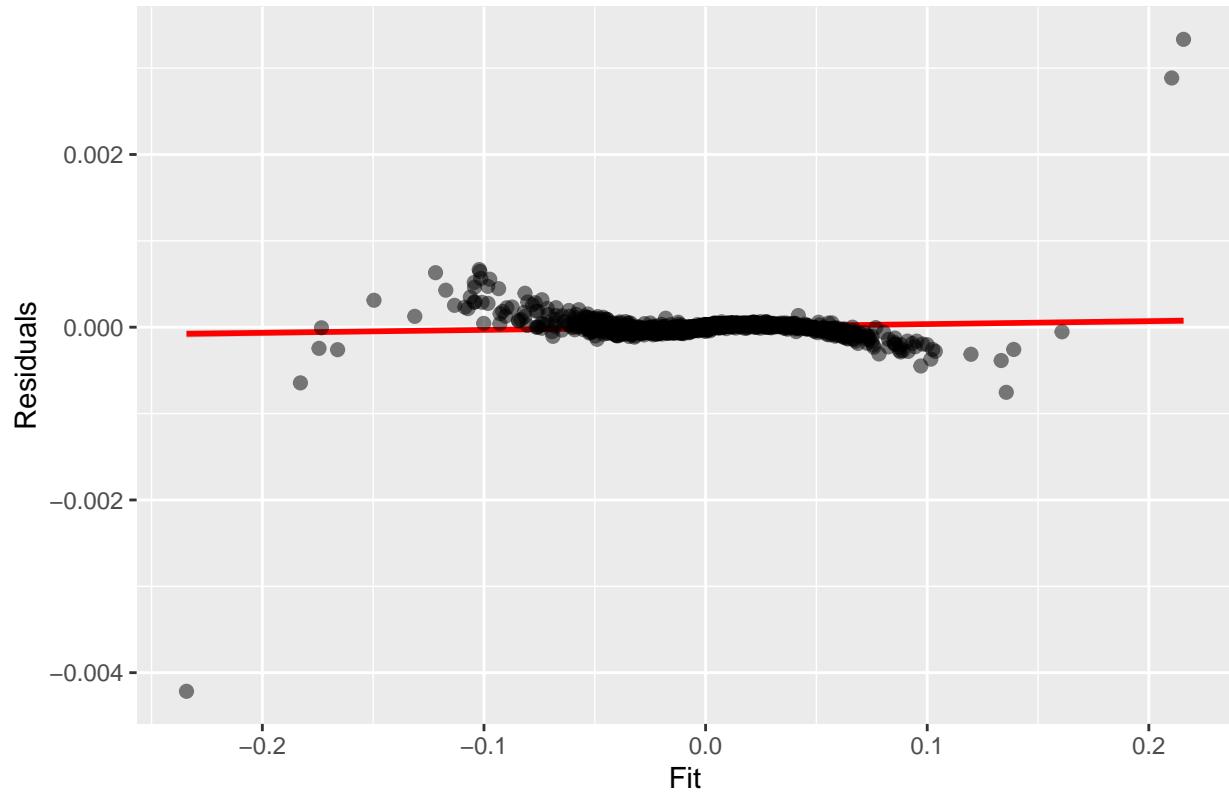
```
plot_resid_fit(fitted(nnar_btc), resid(nnar_btc), "NNAR")
```

NNAR Residuals vs Fit



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

Regression + NNAR Residuals vs Fit

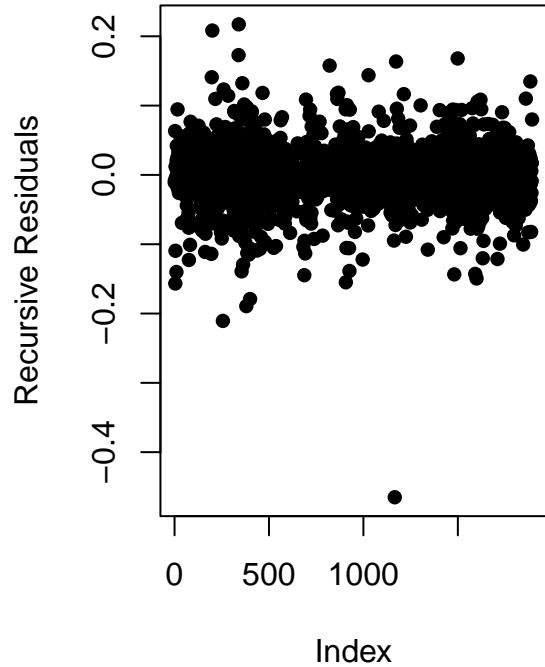


Recursive Residuals

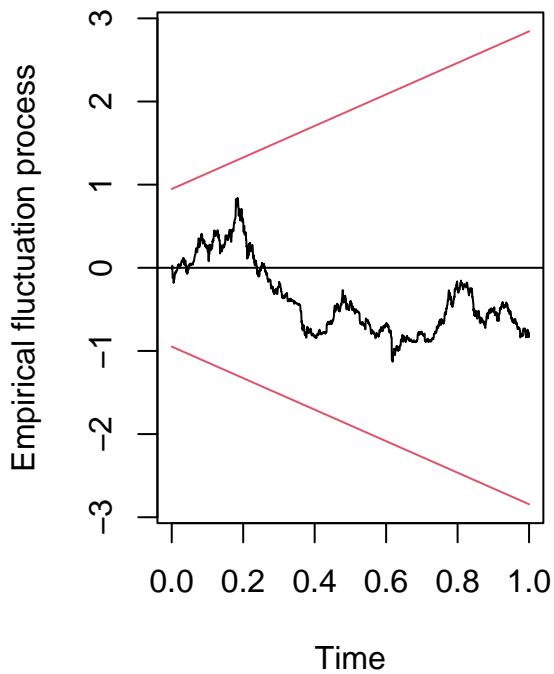
The recursive residuals from the Regression + NNETAR model reaffirm that this model is a good fit. Thus, we will continue and forecast with this fit.

```
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"))
```

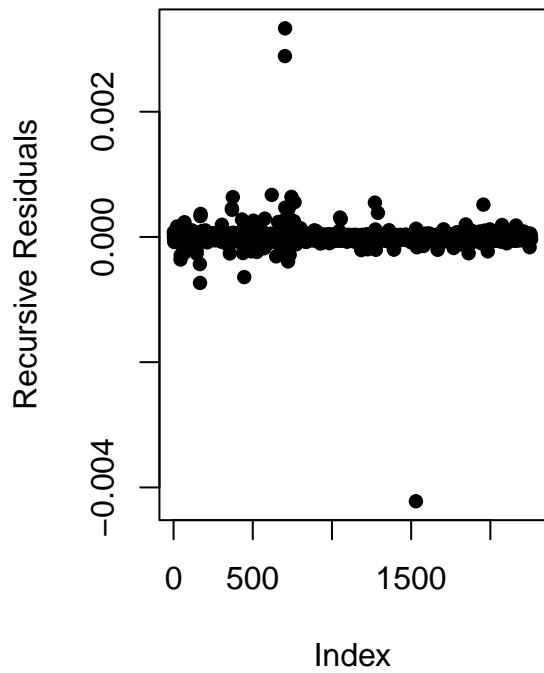
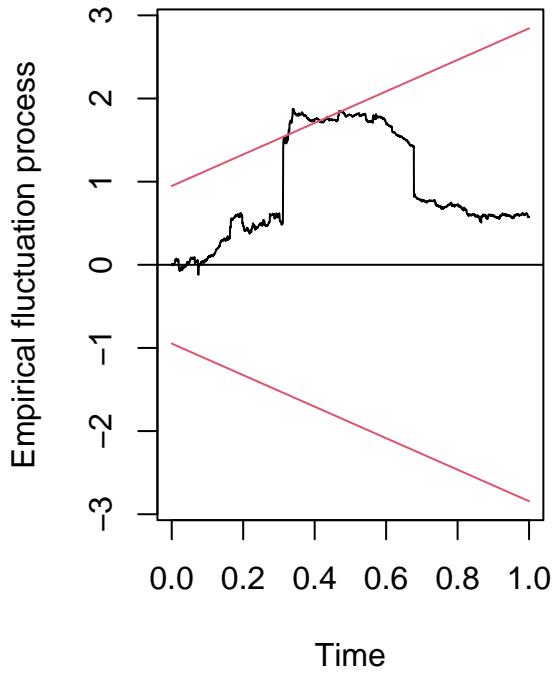
NNAR Residuals



Recursive CUSUM test



```
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"))
```

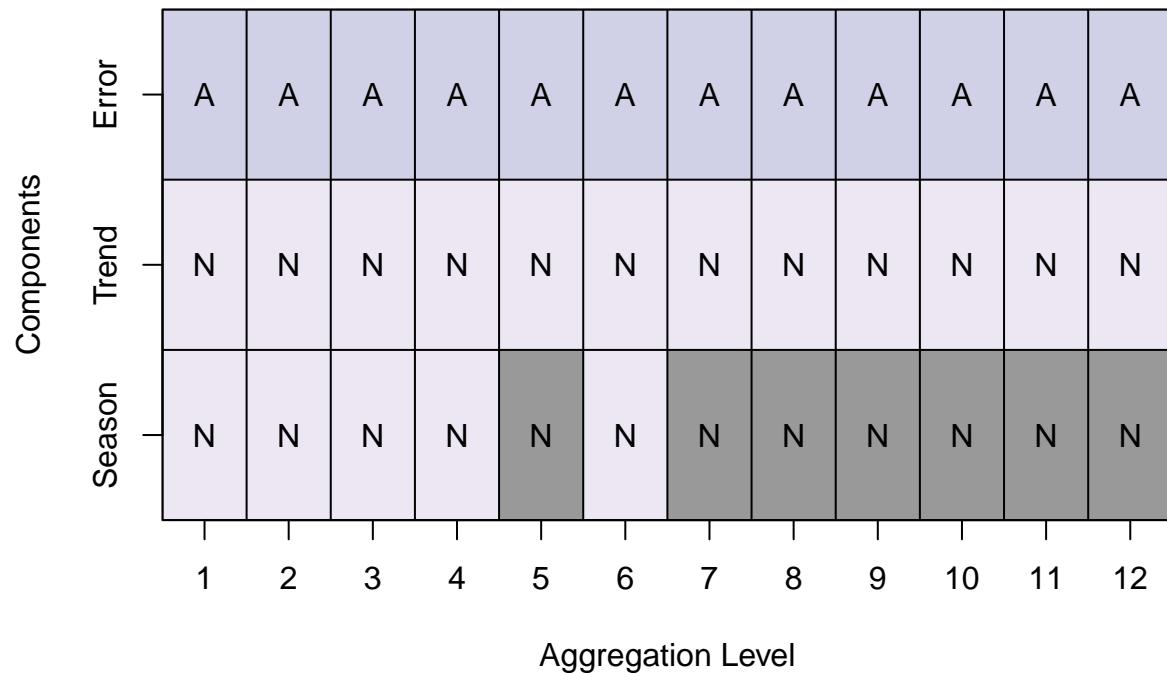
Regression + NNAR Residuals**Recursive CUSUM test**

H) MAPA

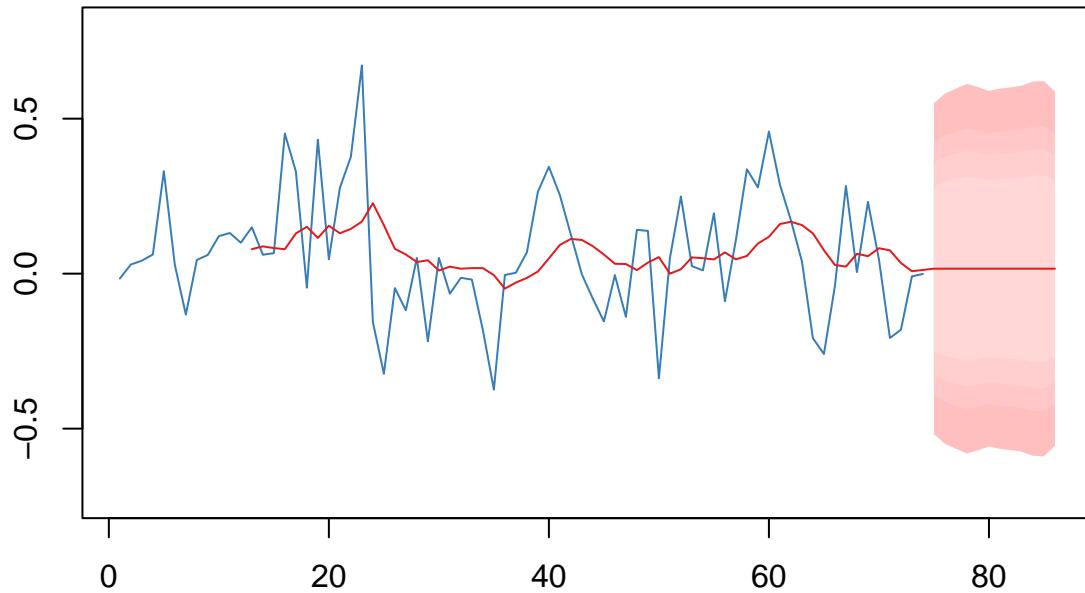
The MAPA model does not fit to the data very well

```
mapa_fit <- mapaest(btc_ts2,outplot=2, paral=2,type="es")  
  
## Running with 12 cores  
  
mapa_for <- mapa(btc_ts2,conf.level=c(0.8,0.9,0.95,0.99),outplot=1)
```

ETS components



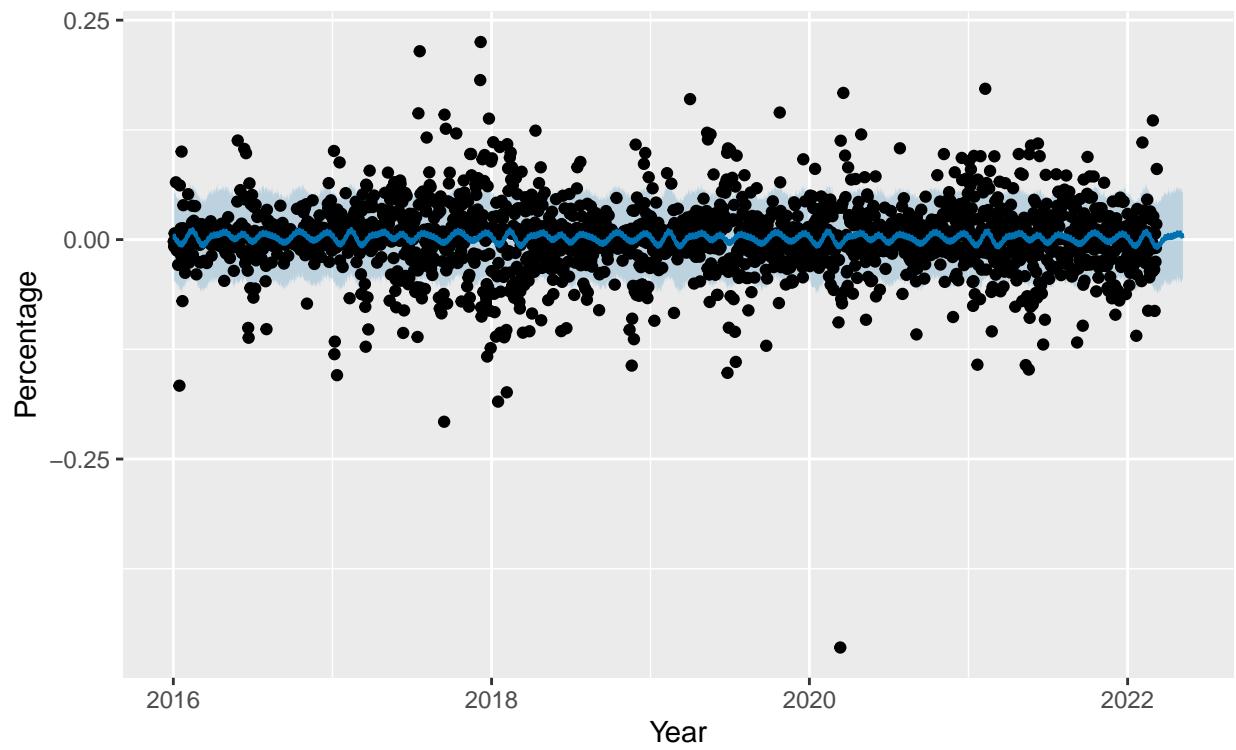
Forecast



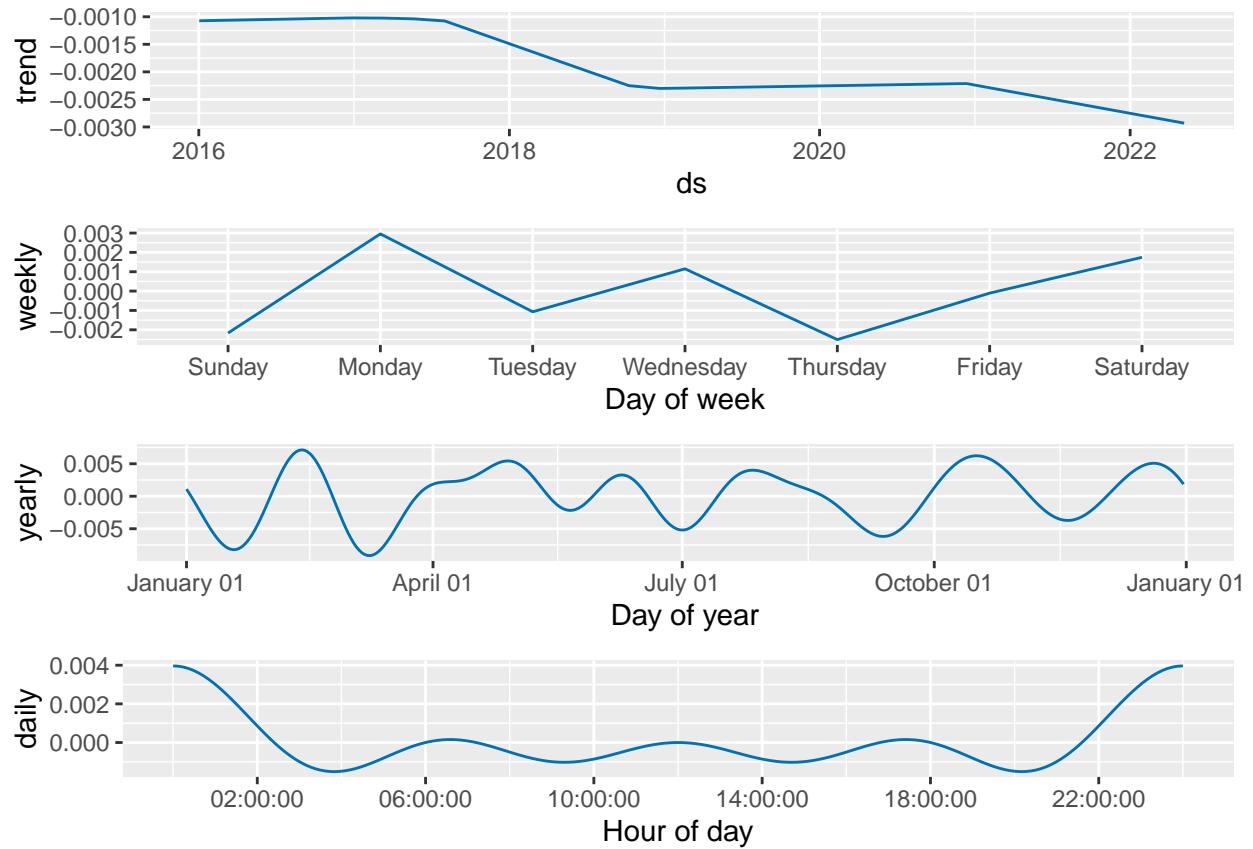
I) Prophet

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")
```

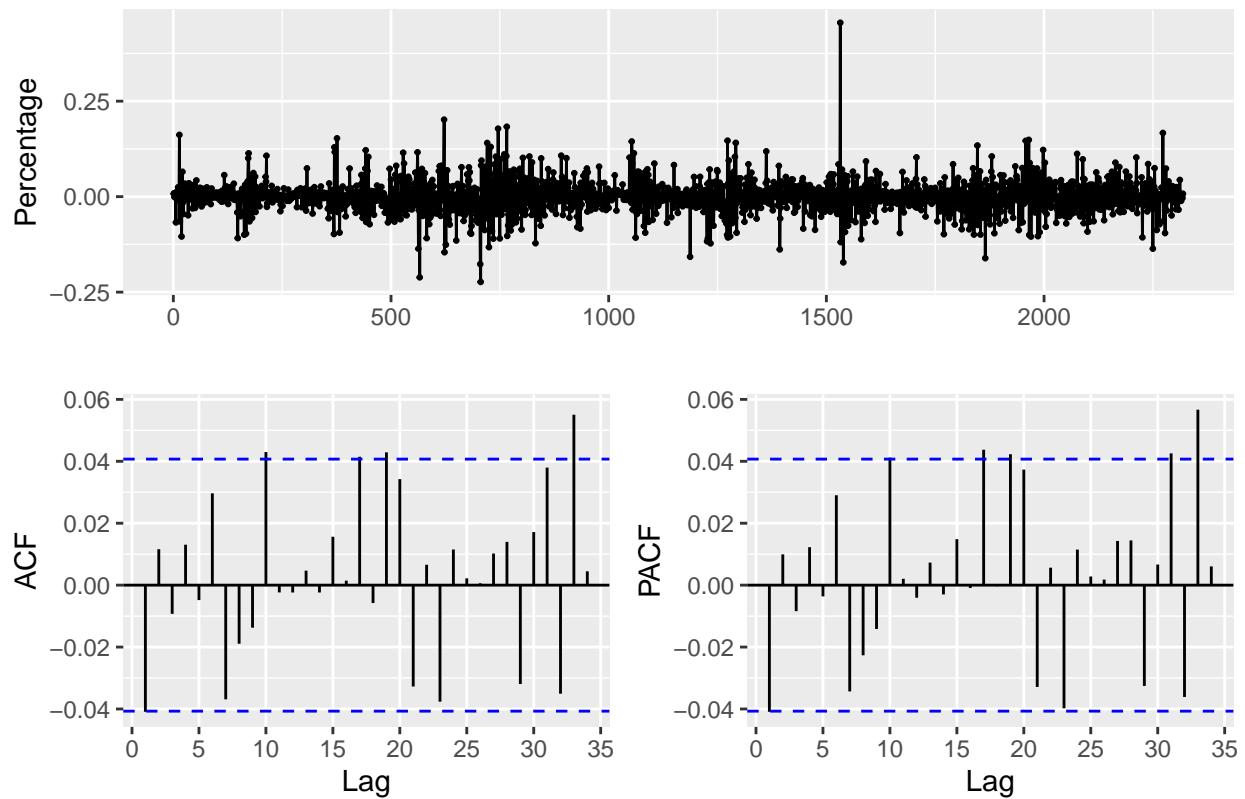


```
prophet_plot_components(preds_btc, forecast_btc)
```



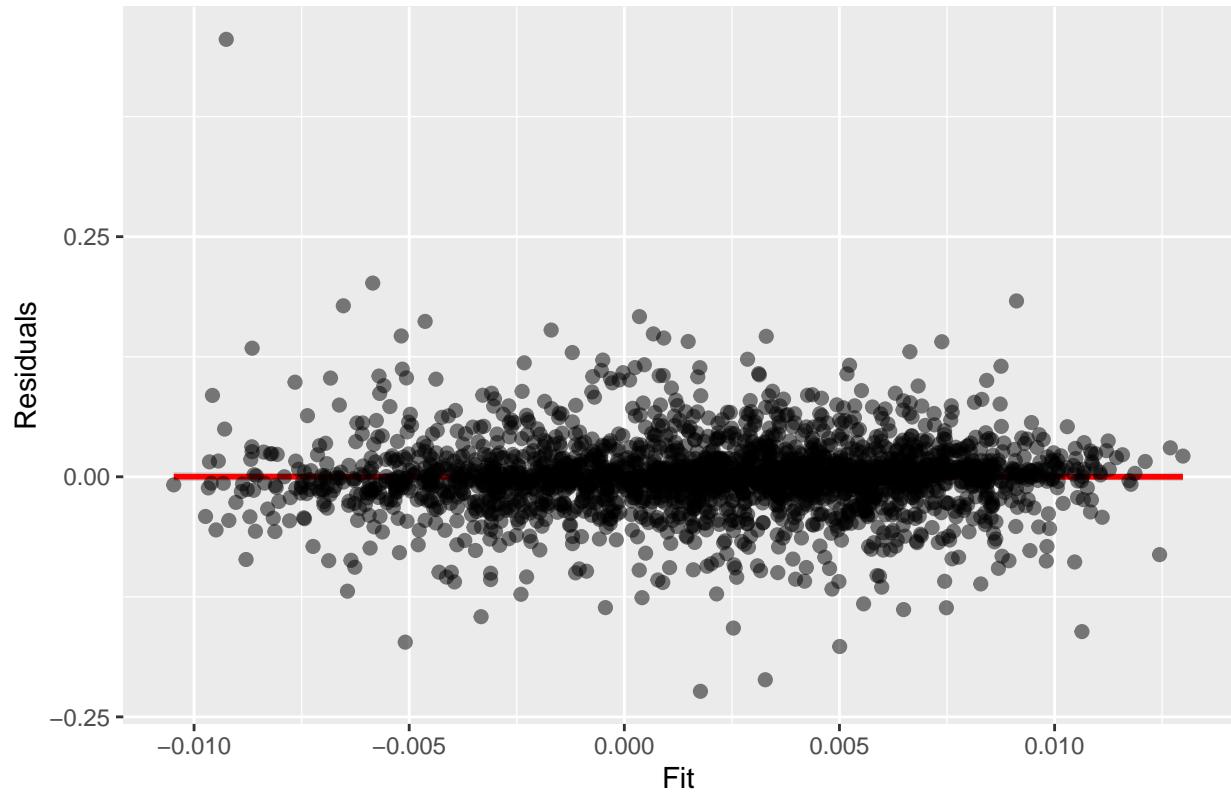
```
res <- forecast_btc$yhat - df_r_btc$y
ggttsdisplay(res, main = "Prophet Residuals", ylab = "Percentage")
```

Prophet Residuals

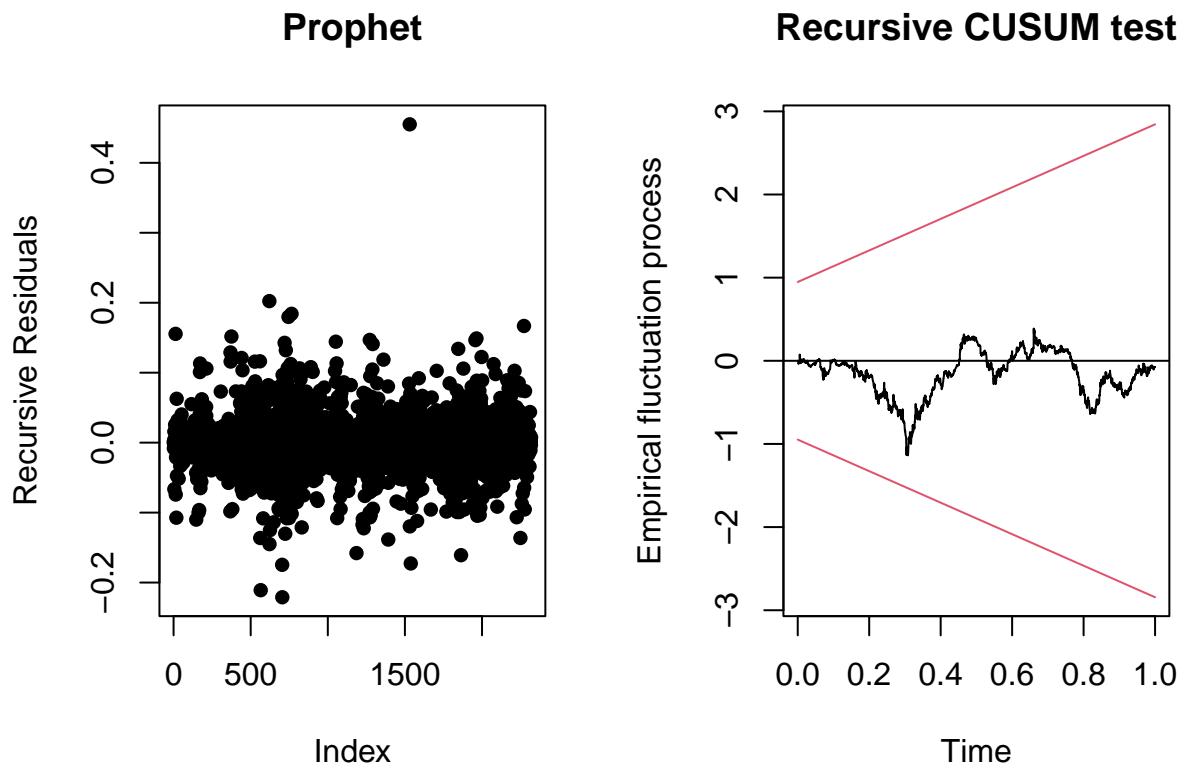


```
plot_resid_fit(forecast_btc$yhat,res,"Prophet")
```

Prophet Residuals vs Fit



```
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 predict the entire year of 2021 for both frequencies. With the high frequency models we will do a 365-step forecast while with the lower frequency 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 < "2022-01-01")
```

```

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

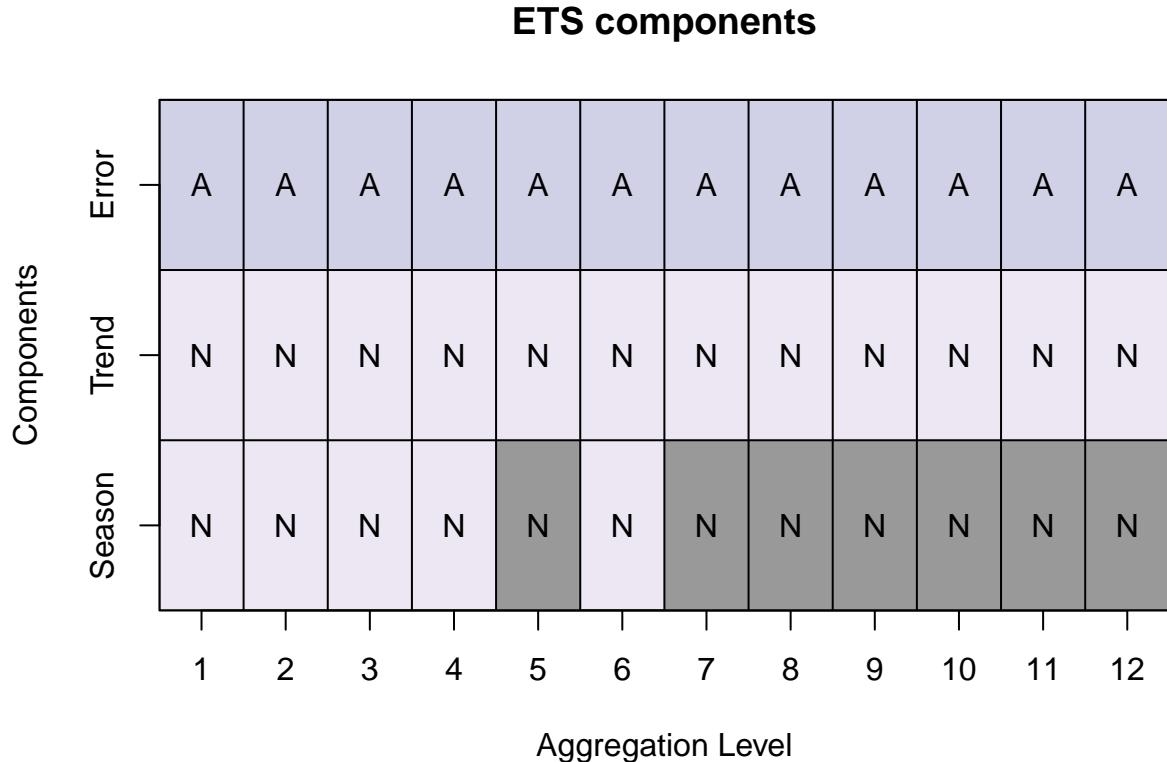
For NNETAR, we get a MAPE of 12.89427.

```
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)

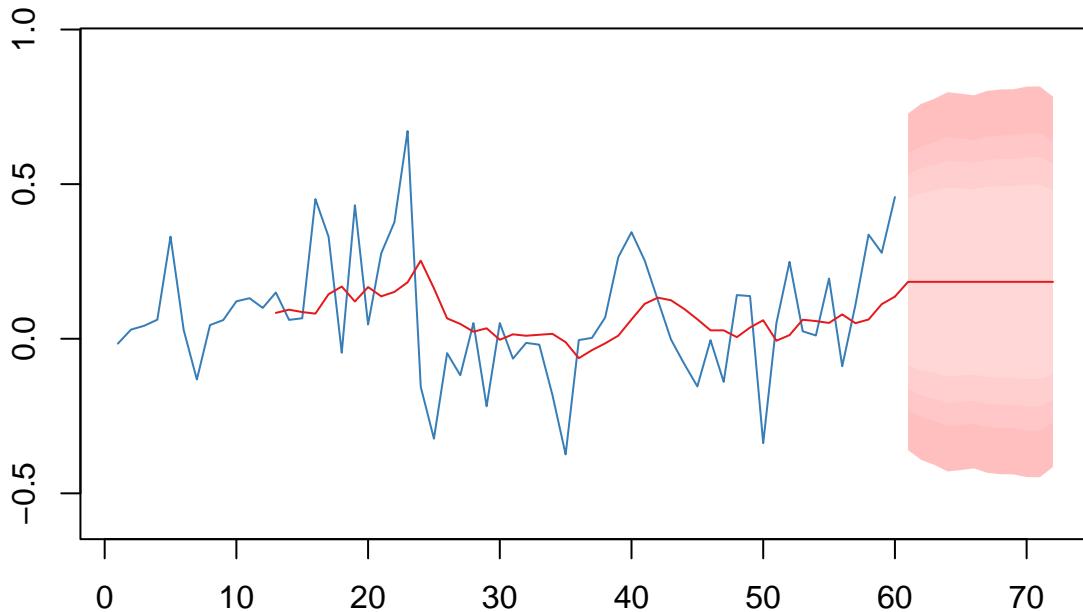
## [1] 20.17489
```

For MAPA, we get a MAPE of 1.149696.

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



Forecast



```
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
```

For the High Frequency models, we get a combined forecast that yields a MAPE of 18.57247. For the Low Frequency models, we get a combined forecast that yields a MAPE of 1.342106.

```
HIGH_COMBO <- (ARIMA + PRO + NNAR) / 3
MAPE(HIGH_COMBO, test)
```

```
## [1] 15.74662
```

```
LOW_COMBO <- (MAPA + HW + ETS) / 3
MAPE(LOW_COMBO, test2)
```

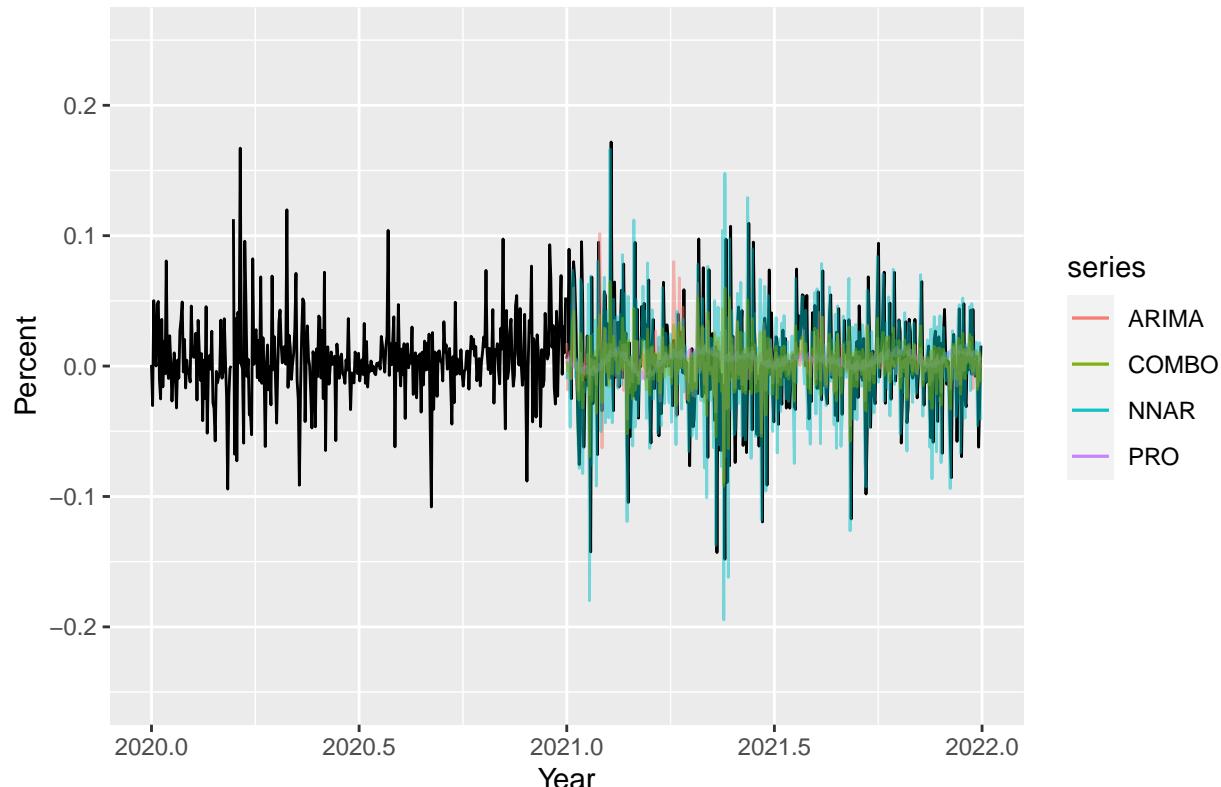
```
## [1] 1.342106
```

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")
```

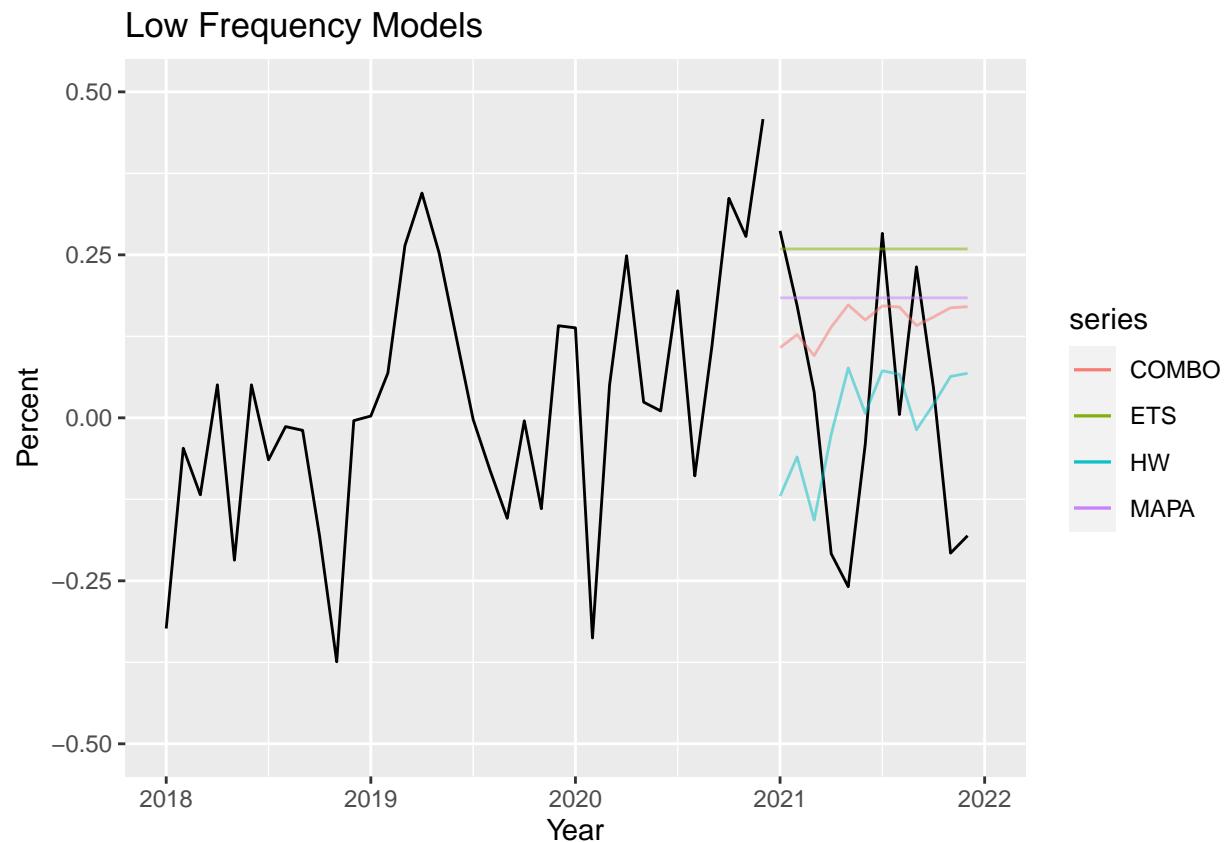
```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

High Frequency Models



```
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.
```



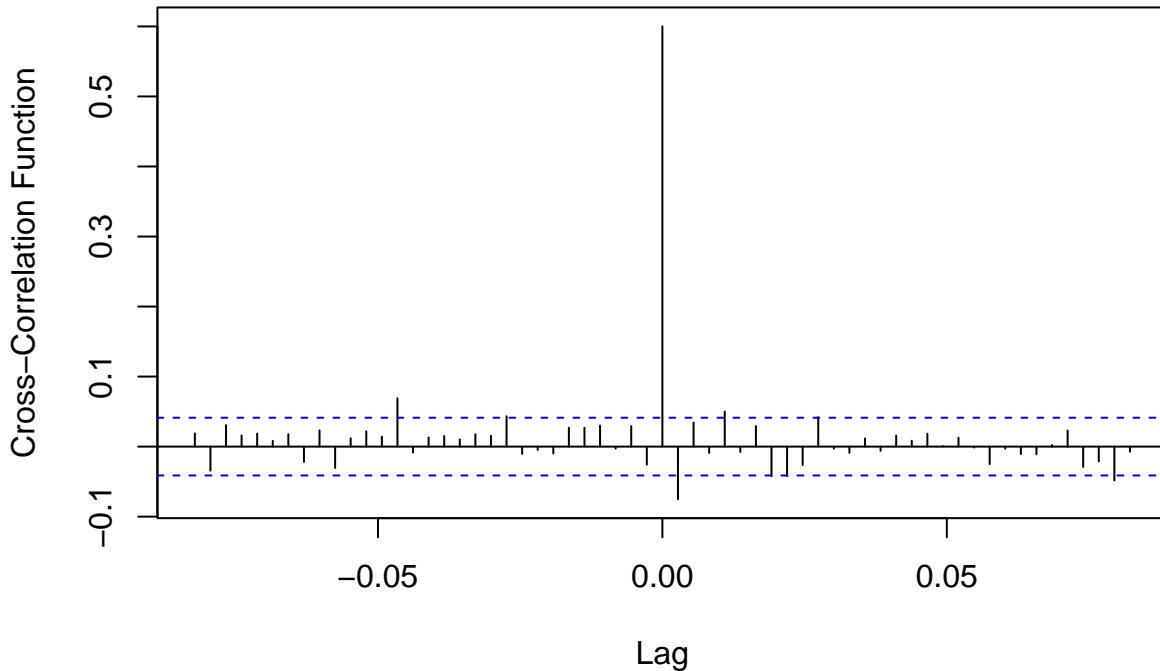
K) VAR Models

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")
```

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) -1.258437e+01 -1.258271e+01 -1.258419e+01 -1.258499e+01 -1.258315e+01
## HQ(n)  -1.257880e+01 -1.257343e+01 -1.257120e+01 -1.256829e+01 -1.256273e+01
## SC(n)  -1.256912e+01 -1.255729e+01 -1.254859e+01 -1.253922e+01 -1.252721e+01
## FPE(n) 3.425131e-06 3.430808e-06 3.425752e-06 3.423008e-06 3.429322e-06
##           6          7          8          9         10
## AIC(n) -1.258125e+01 -1.258000e+01 -1.257970e+01 -1.257708e+01 -1.257661e+01
## HQ(n)  -1.255712e+01 -1.255216e+01 -1.254815e+01 -1.254182e+01 -1.253763e+01
## SC(n)  -1.251514e+01 -1.250372e+01 -1.249326e+01 -1.248046e+01 -1.246982e+01
## FPE(n) 3.435852e-06 3.440142e-06 3.441159e-06 3.450194e-06 3.451812e-06
```

From VARselect, we will set the order 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: 2255
## Log Likelihood: 7810.428
## Roots of the characteristic polynomial:
## 0.5093 0.5093 0.4798 0.4545 0.4265 0.4265 0.4137 0.4137
## 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.l4
##
##           Estimate Std. Error t value Pr(>|t|)    
## btc_ts.l1  0.0363450  0.0264253   1.375 0.169149    
## eth_ts.l1 -0.0680790  0.0181049  -3.760 0.000174 ***  
## btc_ts.l2  0.0010955  0.0264313   0.041 0.966942    
## eth_ts.l2  0.0230545  0.0181469   1.270 0.204060    
## btc_ts.l3  0.0152129  0.0264252   0.576 0.564878    
## eth_ts.l3 -0.0099934  0.0181476  -0.551 0.581912    
## btc_ts.l4 -0.0201437  0.0263214  -0.765 0.444175    
## eth_ts.l4  0.0452583  0.0181551   2.493 0.012743 *   
## const      0.0019989  0.0008438   2.369 0.017926 *  
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03976 on 2246 degrees of freedom
## Multiple R-Squared: 0.01069, Adjusted R-squared: 0.00717 
## F-statistic: 3.035 on 8 and 2246 DF,  p-value: 0.002137
##
##
## 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.l4
##
##           Estimate Std. Error t value Pr(>|t|)    
## btc_ts.l1 -0.040445  0.038586  -1.048 0.29466    
## eth_ts.l1  0.005822  0.026436   0.220 0.82570    
## btc_ts.l2  0.017989  0.038594   0.466 0.64119    
## eth_ts.l2  0.031996  0.026498   1.208 0.22736    
## btc_ts.l3 -0.056108  0.038585  -1.454 0.14605    
## eth_ts.l3  0.064636  0.026499   2.439 0.01480 *   
## btc_ts.l4  0.034767  0.038434   0.905 0.36577    
## eth_ts.l4  0.009306  0.026510   0.351 0.72560    
## const      0.003227  0.001232   2.619 0.00888 ** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##

```

```

## Residual standard error: 0.05805 on 2246 degrees of freedom
## Multiple R-Squared: 0.00599, Adjusted R-squared: 0.002449
## F-statistic: 1.692 on 8 and 2246 DF, p-value: 0.09541
##
##
##
## Covariance matrix of residuals:
##      btc_ts   eth_ts
## btc_ts 0.001581 0.001392
## eth_ts 0.001392 0.003370
##
## Correlation matrix of residuals:
##      btc_ts eth_ts
## btc_ts 1.0000 0.6032
## eth_ts 0.6032 1.0000

```

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.

```
vmmod_btc_eth$varresult$btc_ts
```

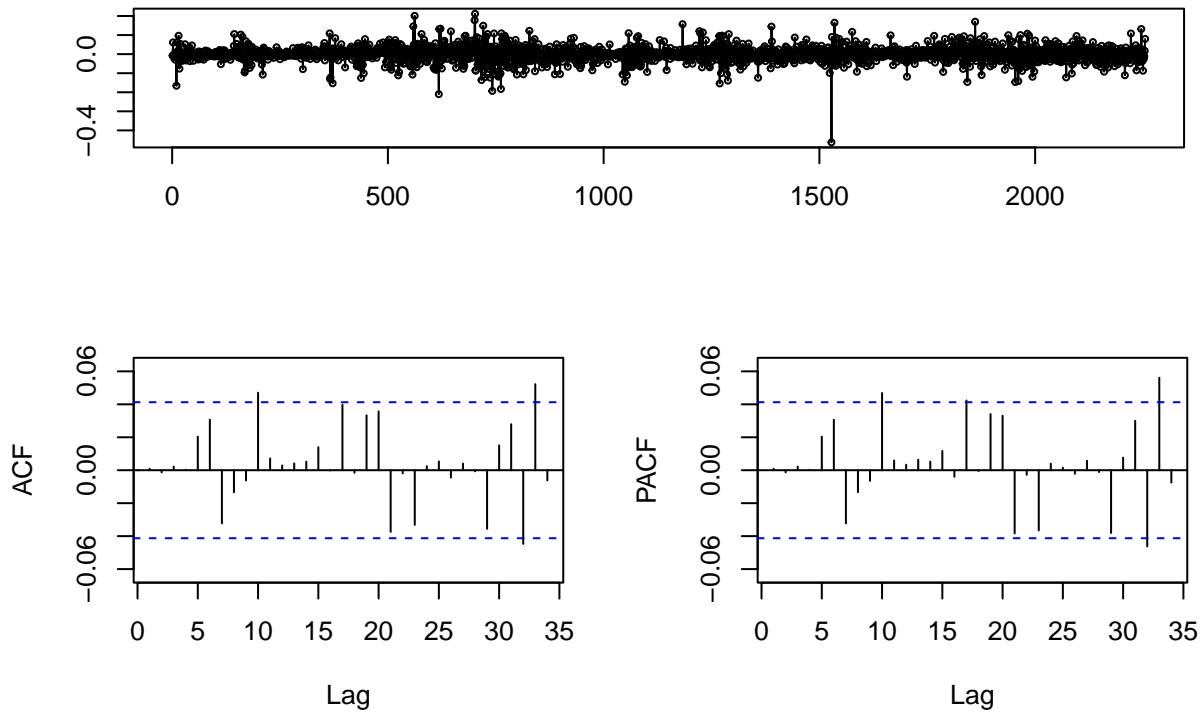
```

##
## 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
## 0.036345 -0.068079  0.001096  0.023054  0.015213 -0.009993 -0.020144
## eth_ts.l4      const
## 0.045258  0.001999

```

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

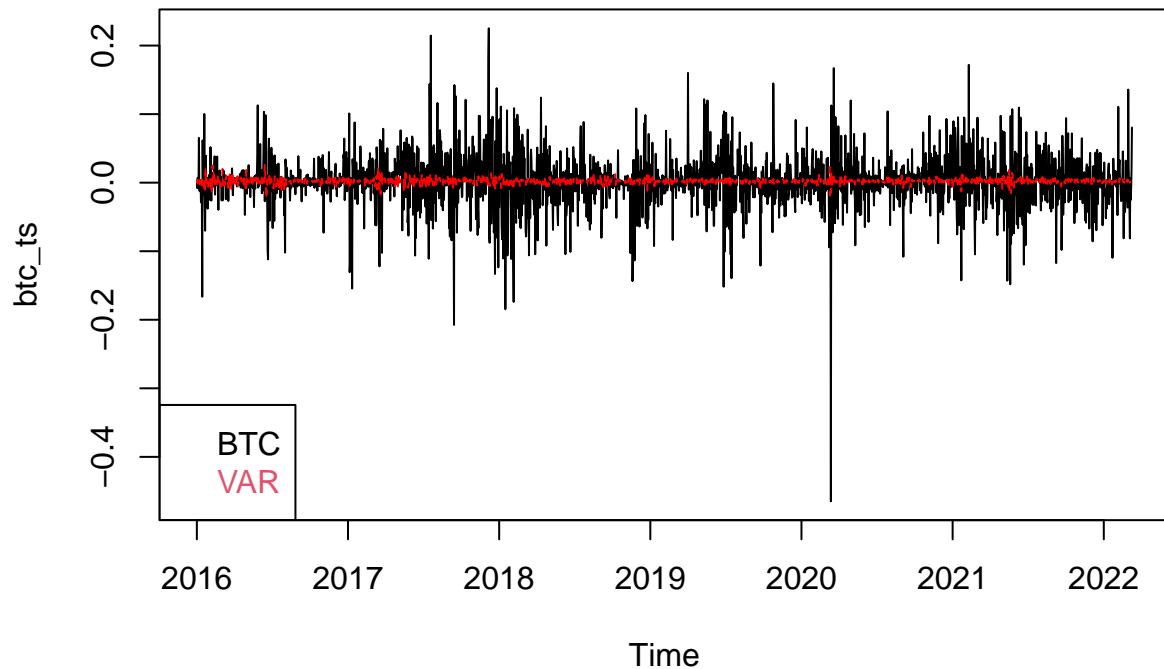
vmod_btc_eth\$varresult\$btc_ts\$residuals



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")
```

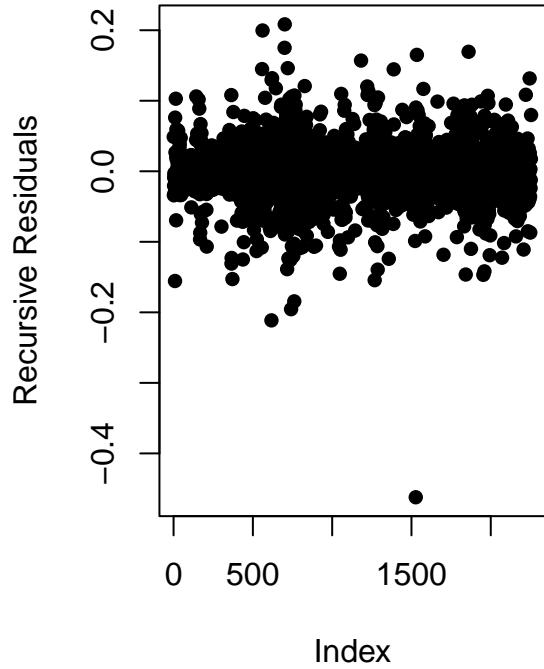
BTC Returns and Fitted Values from BTC,ETH VAR Model



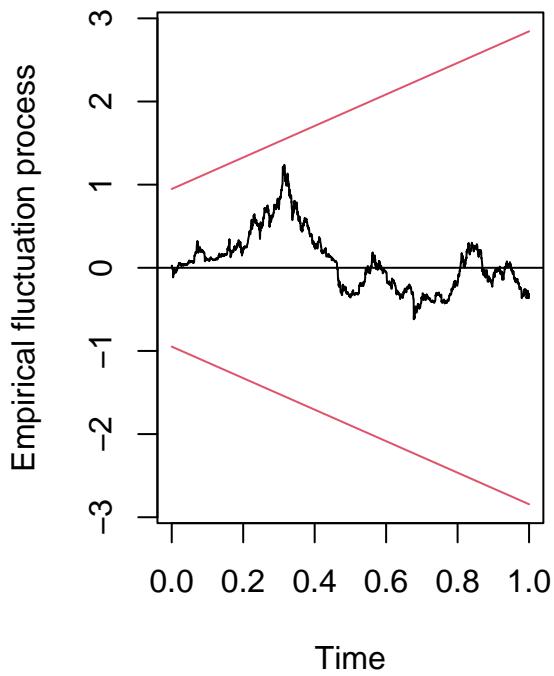
According to our recursive CUSUM test, the fluctuation process remains well within the confidence bands meaning that there is no structural break in our Holt-Winters model.

```
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"))
```

HW Residuals



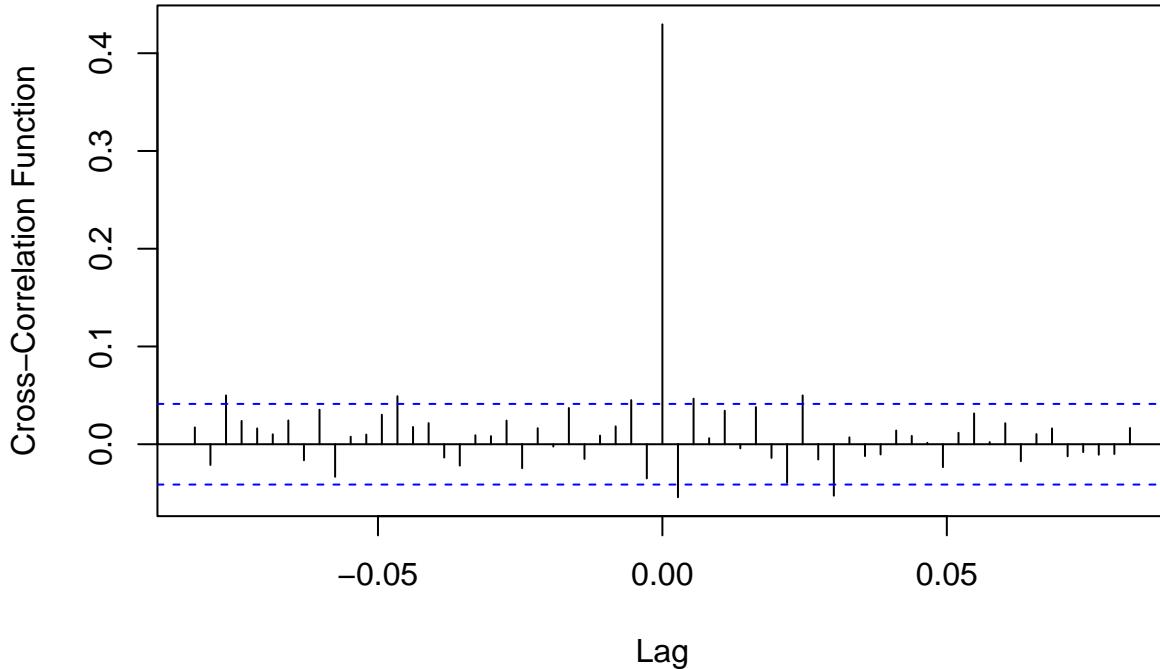
Recursive CUSUM test



BTC & XRP

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

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) -1.196424e+01 -1.196703e+01 -1.196522e+01 -1.196281e+01 -1.196254e+01
## HQ(n)  -1.195867e+01 -1.195775e+01 -1.195223e+01 -1.194611e+01 -1.194212e+01
## SC(n)  -1.194899e+01 -1.194161e+01 -1.192962e+01 -1.191705e+01 -1.190660e+01
## FPE(n) 6.367892e-06 6.350142e-06 6.361667e-06 6.376999e-06 6.378748e-06
##           6           7           8           9          10
## AIC(n) -1.196149e+01 -1.196025e+01 -1.196285e+01 -1.196499e+01 -1.196558e+01
## HQ(n)  -1.193736e+01 -1.193241e+01 -1.193130e+01 -1.192973e+01 -1.192660e+01
## SC(n)  -1.189539e+01 -1.188397e+01 -1.187640e+01 -1.186837e+01 -1.185879e+01
## FPE(n) 6.385431e-06 6.393360e-06 6.376772e-06 6.363120e-06 6.359384e-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: 2257
## Log Likelihood: 7115.981
## Roots of the characteristic polynomial:
## 0.299 0.2688 0.05293 0.05293
## 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 -0.0014427  0.0233217 -0.062   0.9507
## xrp_ts.l1 -0.0299595  0.0132964 -2.253   0.0243 *
## btc_ts.l2  0.0048731  0.0233033  0.209   0.8344
## xrp_ts.l2  0.0247247  0.0133067  1.858   0.0633 .
## const      0.0020338  0.0008404  2.420   0.0156 * 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03982 on 2252 degrees of freedom
## Multiple R-Squared: 0.005024,    Adjusted R-squared: 0.003256
## F-statistic: 2.843 on 4 and 2252 DF,  p-value: 0.02293
##
##
## 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 -0.047485  0.040794 -1.164   0.24454
## xrp_ts.l1 -0.008404  0.023258 -0.361   0.71789
## btc_ts.l2  0.023276  0.040762  0.571   0.56805
## xrp_ts.l2  0.071899  0.023276  3.089   0.00203 ** 
## const      0.002061  0.001470  1.402   0.16098
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.06965 on 2252 degrees of freedom
## Multiple R-Squared: 0.007429,    Adjusted R-squared: 0.005666
## F-statistic: 4.214 on 4 and 2252 DF,  p-value: 0.002115
##
##
## Covariance matrix of residuals:
##          btc_ts   xrp_ts

```

```

## btc_ts 0.001586 0.001185
## xrp_ts 0.001185 0.004851
##
## Correlation matrix of residuals:
##      btc_ts xrp_ts
## btc_ts 1.0000 0.4273
## xrp_ts 0.4273 1.0000

```

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
```

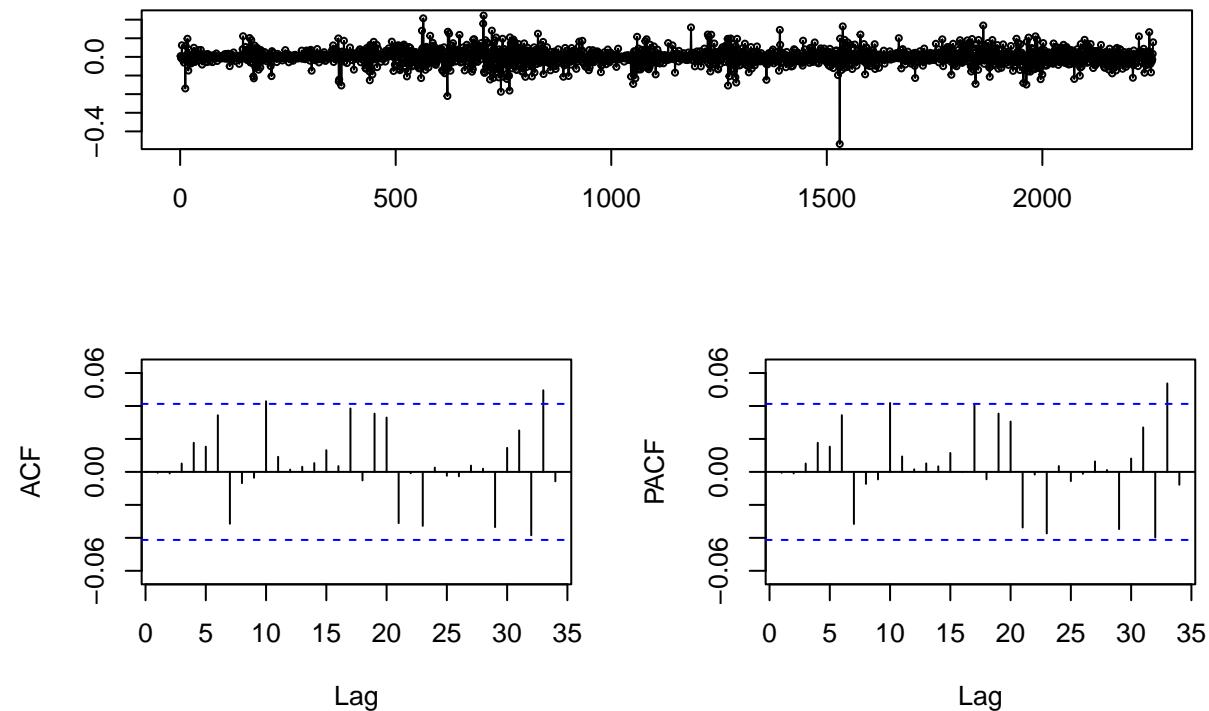
```

##
## Call:
## lm(formula = y ~ -1 + ., data = datamat)
##
## Coefficients:
## btc_ts.l1  xrp_ts.l1  btc_ts.l2  xrp_ts.l2      const
## -0.001443 -0.029959   0.004873   0.024725   0.002034

```

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

vmod_btc_xrp\$varresult\$btc_ts\$residuals

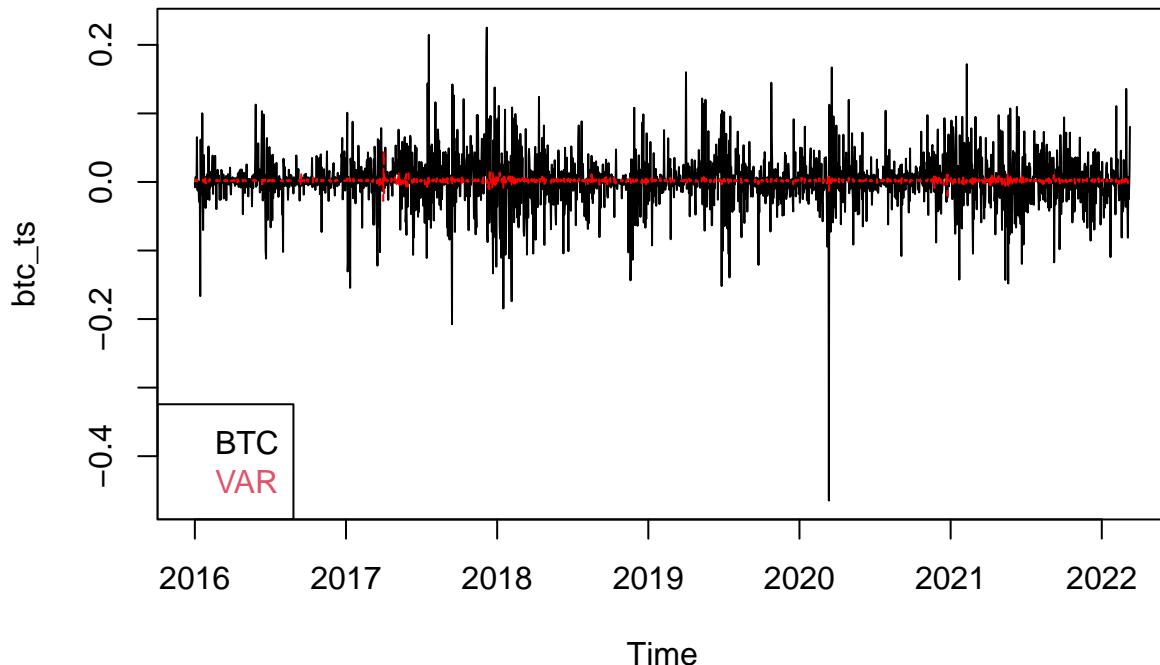


As with the previous VAR model, the VAR model using ETH returns does not seem to capture any of the volatility in the BTC returns series. We will now train and test each VAR model (the first one using ETH

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 outperform them.

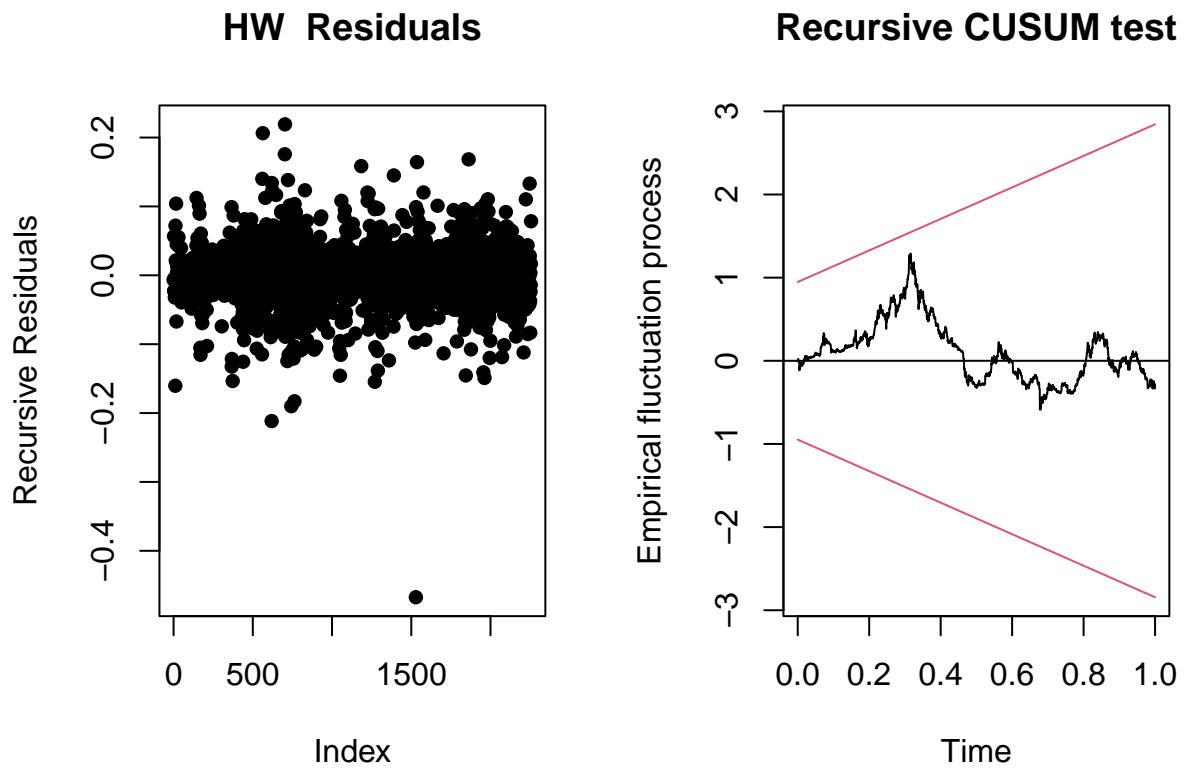
```
vmod_xrp_ts = ts(vmod_btc_xrp$varresult$btc_ts$fitted.values, start=c(2016,1), freq=365)
plot(btc_ts,col="black",lwd=1,lty=1,main="BTC Returns and Fitted Values from BTC,XRP VAR Model")
lines(vmod_xrp_ts,col="red",lty=2)
legend("bottomleft",legend=c("BTC","VAR"),text.col=1:4,bty = "o")
```

BTC Returns and Fitted Values from BTC,XRP VAR Model



The CUSUM test for recursive residuals shows that our model has no structural breaks.

```
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])

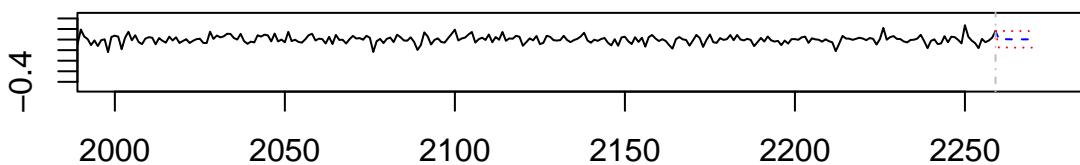
## [1] 147.8384

Forecast 12 steps ahead using ETH VAR Model

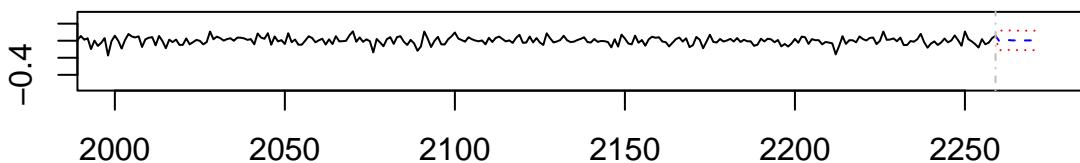
vmod.pred <- predict(object=vmod_btc_eth, n.ahead=12)
plot(vmod.pred, xlim=c(2000,2275))

```

Forecast of series btc_ts



Forecast of series eth_ts



L) Financial Performance & Interpretation

Given the data and interpretations above, we will attempt to evaluate the financial performance of the models predicting the daily returns of BTC.

First, we evaluate the performance of the Regression + ARIMA model:

The hit ratio is really good, and the returns of the Regression + ARIMA model are staggering. However, this brings into question whether or not there was leakage in the process of making predictions.

Now we move onto calculating the financial returns of our Regression + NNETAR model.

The hit ratio is surprisingly good, and the returns of the Regression + NNETAR model are fantastic, but questionable. Is this another sign of data leakage? Or did our model perform very well?

The hit ratio of the Prophet model is significantly lower than the other daily predictive models. However, this is given that we didn't incorporate ETH as a regression and only used a vanilla Prophet model.

Nonetheless, this model achieved a hit accuracy of 55%, just slightly better than a coin flip. Although the returns aren't as good as the other models, they are quite good in any case.

III. Conclusion

In the project, we predicted the returns for Bitcoin using a variety of models including ARIMA, ETS, Holt-Winters, VAR, NNETAR, MAPA, and more. For some of the models such as ARIMA and ETS, we also devised a regression, regressing the returns of BTC on the returns of ETH, which was identified as a significant variable because of its strong correlation with BTC's returns.

Due to technical limitations, different models had to be constructed based on different time frequencies. For instance, our ETS and Holt-Winters models were constructed from data on monthly returns, while the ARIMA model was developed based on the daily returns of BTC.

Training and testing sets were used to evaluate and measure the forecast performance of the models, and overall, the best performing model according to its MAPE was the Regression + ARIMA for modelling daily returns and the Regression + ETS for modelling monthly returns.

In addition to measuring the accuracy of our models in this way, we also measured the financial performance of the models. In particular, we measured the returns of the models used to forecast daily returns, and this measurement affirmed that the Regression + ARIMA model was the best performing model as well.

In the future, to measure financial performance, we could also incorporate ways to subtract transaction costs. In addition to that, we may also attempt to add other regressors to our models to improve their accuracy.

IV. References

- <https://otexts.com/fpp3/>
- <https://robjhyndman.com/>
- https://facebook.github.io/prophet/docs/quick_start.html
- <http://www.econ.ucla.edu/rrojas/>
- <https://www.statology.org/aggregate-daily-data-in-r/>