Project (Regular)

Mohammad (s3650497), Siddhant Gehlot (s3620089), Namita Chhibba (s3631442)

15 October 2017

Importing and Reading Bitcoin Cas Price Data

```
bitcoin <- read.csv("C:/Users/user/Desktop/bitcoin_cash_price.csv")
# View(bitcoin)
head(bitcoin)
```

```
##           Date   Open   High    Low  Close      Volume    Market.Cap
## 1 Oct 03, 2017 421.79 421.79 395.74 404.18 1302,47,000 7,022,370,000
## 2 Oct 02, 2017 415.87 430.86 411.84 421.19 2195,90,000 6,921,580,000
## 3 Oct 01, 2017 433.38 436.94 415.15 415.15 1642,90,000 7,207,770,000
## 4 Sep 30, 2017 436.64 445.62 432.53 432.63 1505,65,000 7,258,880,000
## 5 Sep 29, 2017 447.66 447.92 426.99 436.77 1487,25,000 7,441,960,000
## 6 Sep 28, 2017 456.71 465.20 433.50 447.81 3004,21,000 7,592,000,000
```

Converting the data into ascending order

```
bitcoin_asc <- bitcoin[seq(dim(bitcoin)[1],1),]
row.names(bitcoin_asc) <- 1:nrow(bitcoin_asc)
head(bitcoin_asc)
```

```
##           Date   Open   High    Low  Close    Volume Market.Cap
## 1 Jul 23, 2017 555.89 578.97 411.78 413.06    85,013          -
## 2 Jul 24, 2017 412.58 578.89 409.21 440.70  1,90,952          -
## 3 Jul 25, 2017 441.35 541.66 338.09 406.90  5,24,908          -
## 4 Jul 26, 2017 407.08 486.16 321.79 365.82 17,84,640          -
## 5 Jul 27, 2017 417.10 460.97 367.78 385.48  5,33,207          -
## 6 Jul 28, 2017 386.65 465.18 217.06 406.05 12,30,160          -
```

Converting the closing price into time series object considering frequency=7

```
bitcoin_ts <- ts(bitcoin_asc$Close, start= c(2017,4,23), frequency=7)
# View(bitcoin_ts)
head(bitcoin_ts)
```

```
## Time Series:
## Start = c(2017, 4)
## End = c(2018, 2)
## Frequency = 7
## [1] 413.06 440.70 406.90 365.82 385.48 406.05
```

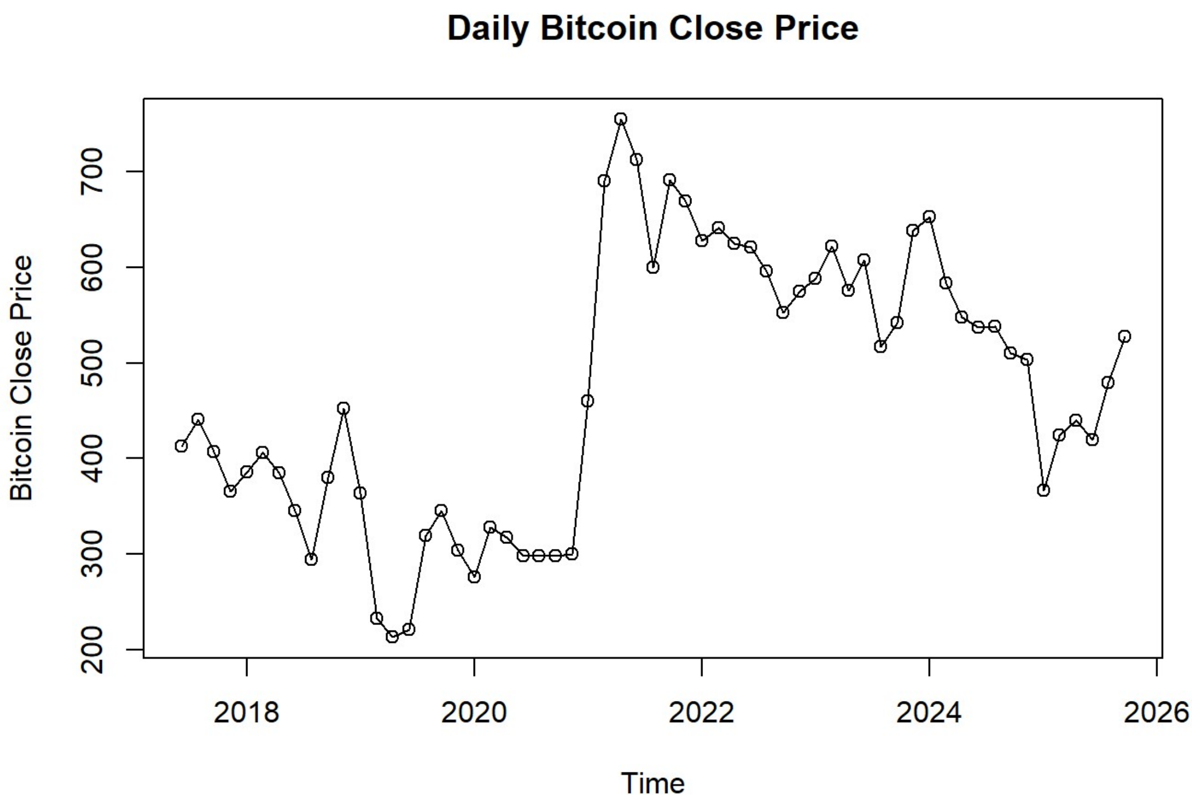
Dividing the time series into two parts. Considering first part for series analysis and model selection and last 2 weeks (14 days) readings for forecasting estimation.

```
bitcoin1= bitcoin_ts[1:(length(bitcoin_ts)-14)]
bitcoin2= bitcoin_ts[(length(bitcoin_ts)-14): length(bitcoin_ts)]
```

```
bitcoin1 <-ts(bitcoin1, start=c(2017,4,23), frequency=7)
bitcoin2 <- ts(bitcoin2, frequency=7)
```

Plotting time series for the bitcoin1

```
par(mfrow=c(1,1))
plot(bitcoin1, type="o",col = "black", ylab='Bitcoin Close Price',
     main = "Daily Bitcoin Close Price")
```

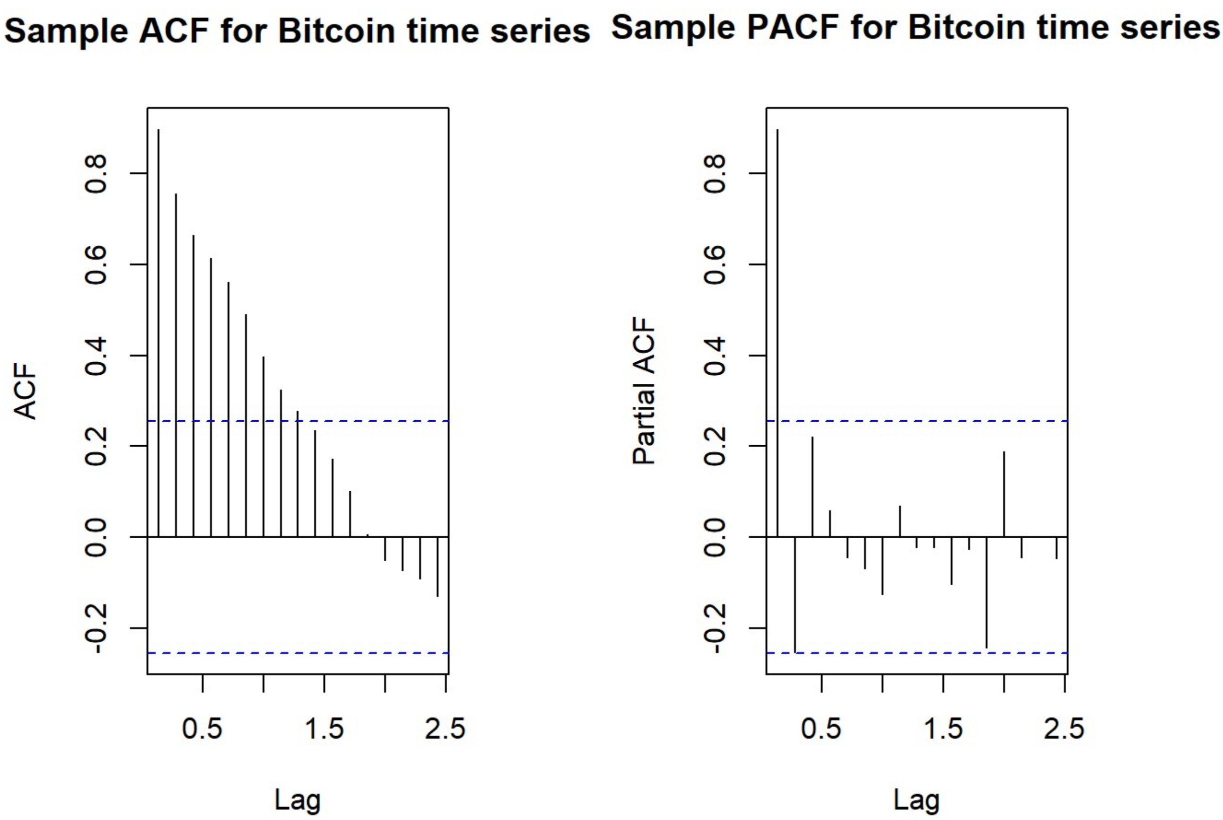


Interpretation

Looking at the time series plot we can say that there is a -probability of trend -presence of huge intervention -hardly any sign of seasonality -moving average -presence of changing variance

Analyzing the ACF and PACF plot for the given time series.

```
par(mfrow=c(1,2), mai=rep(0.9,4))
acf(bitcoin1, main="Sample ACF for Bitcoin time series")
pacf(bitcoin1, main="Sample PACF for Bitcoin time series")
```



Interpretation

Because we have a slowly decreasing pattern in ACF and a very high PACF value at the first lag, we infer that here is a trend in the series which dominates the serial correlation properties of the series. And there is no nign of seasonality.

Model Selection

We will choose the intervention models and state space models to deal with the intervention, trend, changing variance and make the series stationary.

Intervention Analysis Models

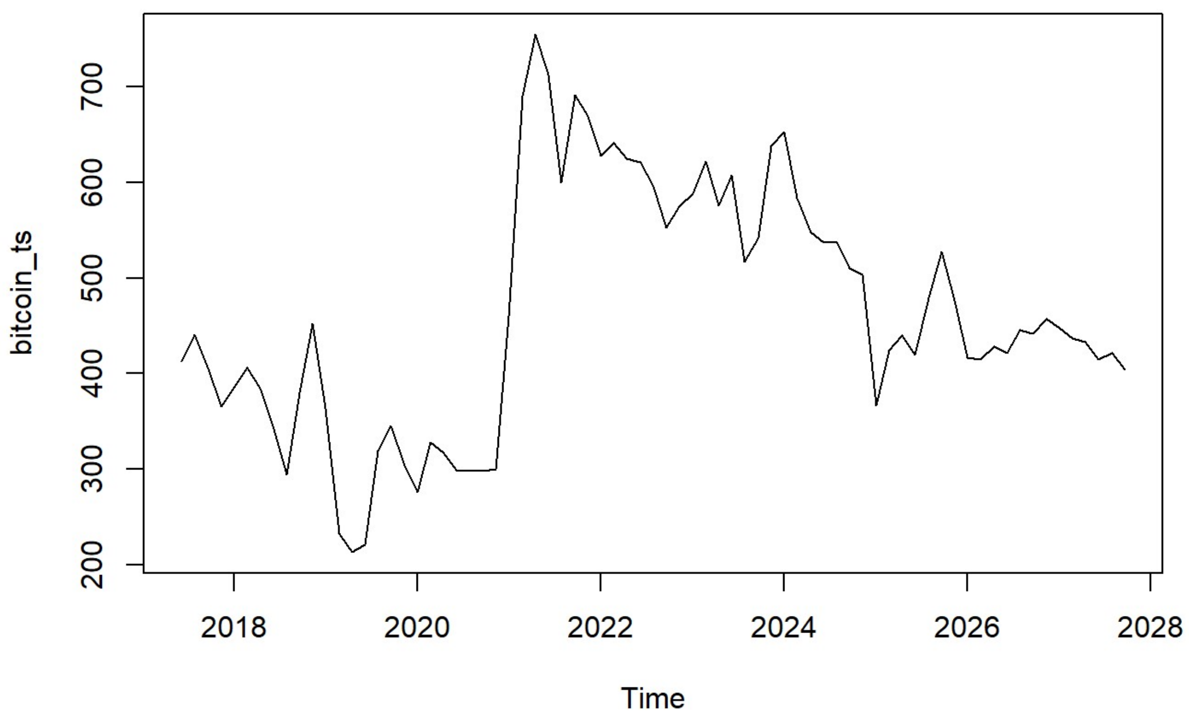
Since we have got an impressionable intervention in the time series. We will check for the time point where the intervention occurs

```
library(tsoutliers)
#library(expsmooth)
library(fma)
```

```
##
## Attaching package: 'fma'
```

```
## The following objects are masked from 'package:MASS':
##
##     cement, housing, petrol
```

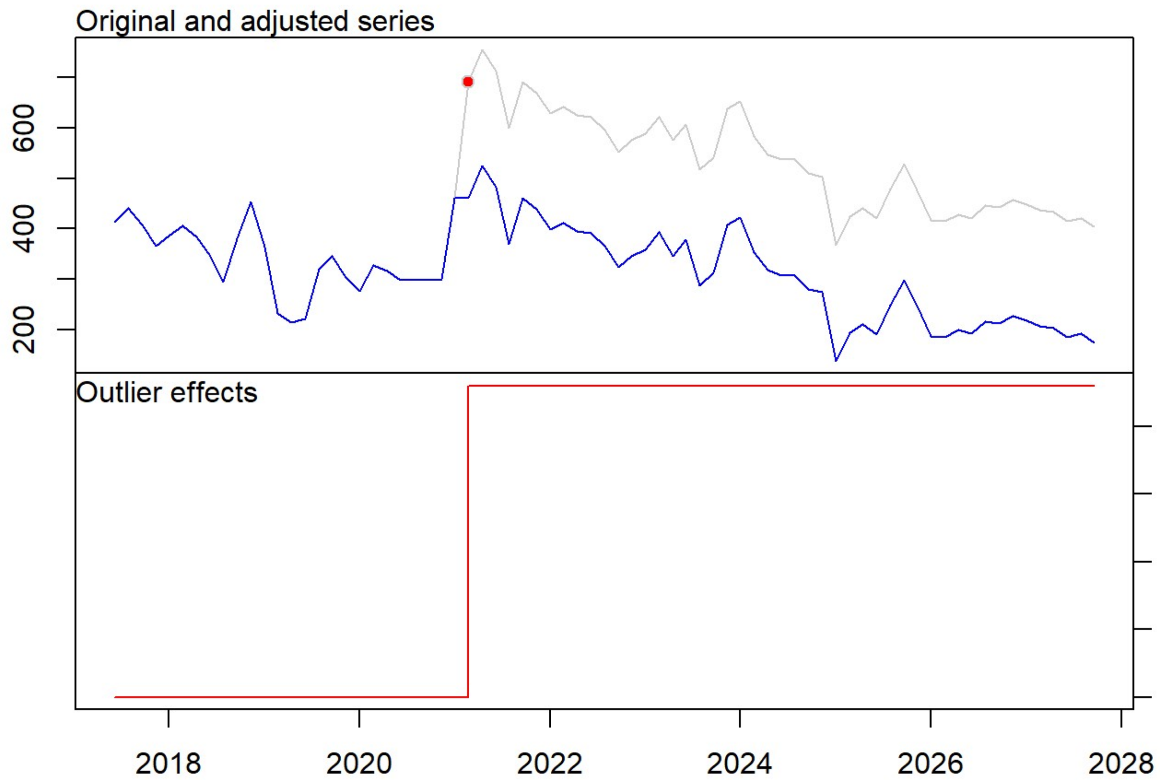
```
#Library(TSA)
plot(bitcoin_ts)
```




```
outlier.bitcoin <- tsoutliers::tso(bitcoin_ts,types = c("AO","LS","TC"),maxit.iloop
=10)
outlier.bitcoin
```

```
## Series: bitcoin_ts
## Regression with ARIMA(0,1,0) errors
##
## Coefficients:
##           LS27
##       229.7100
## s.e.   51.1043
##
## sigma^2 estimated as 2648:  log likelihood=-385.4
## AIC=774.8   AICc=774.98   BIC=779.36
##
## Outliers:
##   type ind    time coefhat tstat
## 1   LS  27 2021:02   229.7 4.495
```

```
plot(outlier.bitcoin)
```

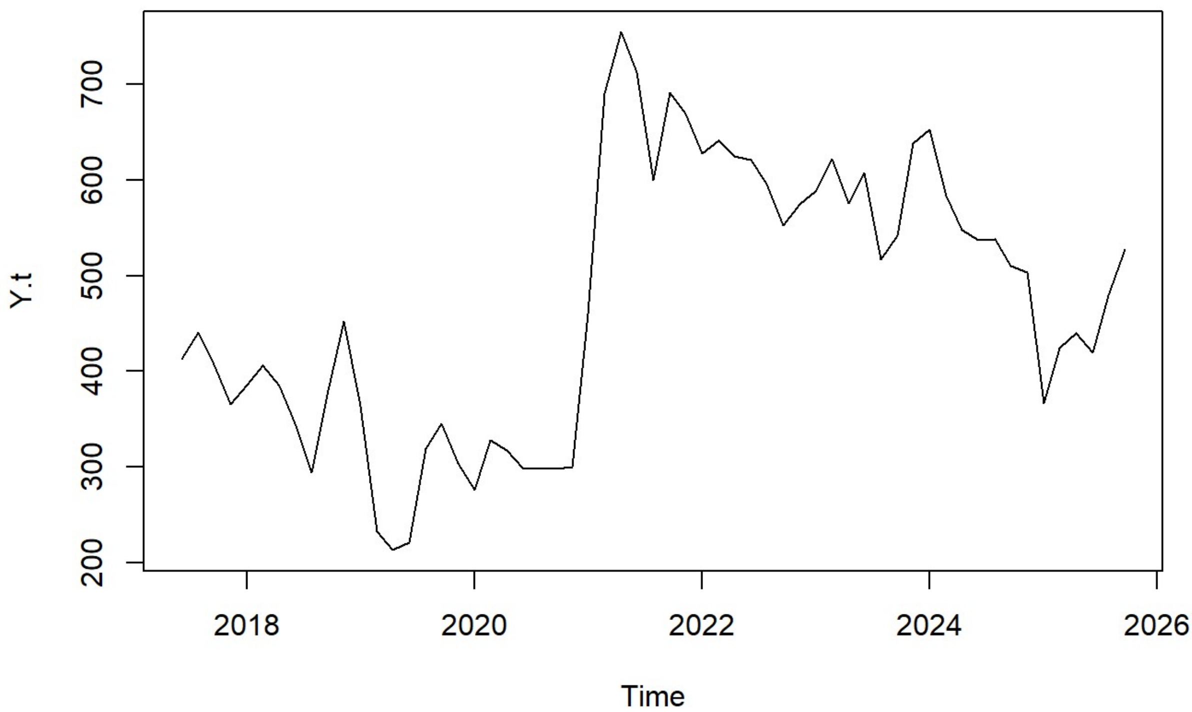


Interpretation As seen the intervention occurs at T=27 and is a step function.Applying dynlm model for intervention T=27.

```
class(bitcoin1)
```

```
## [1] "ts"
```

```
Y.t = bitcoin1
T = 27 # The time point when the intervention occurred
P.t = 1*(seq(bitcoin1) >= T)
P.t.1 = Lag(P.t,+1)
plot(Y.t)
```



Various intervention models

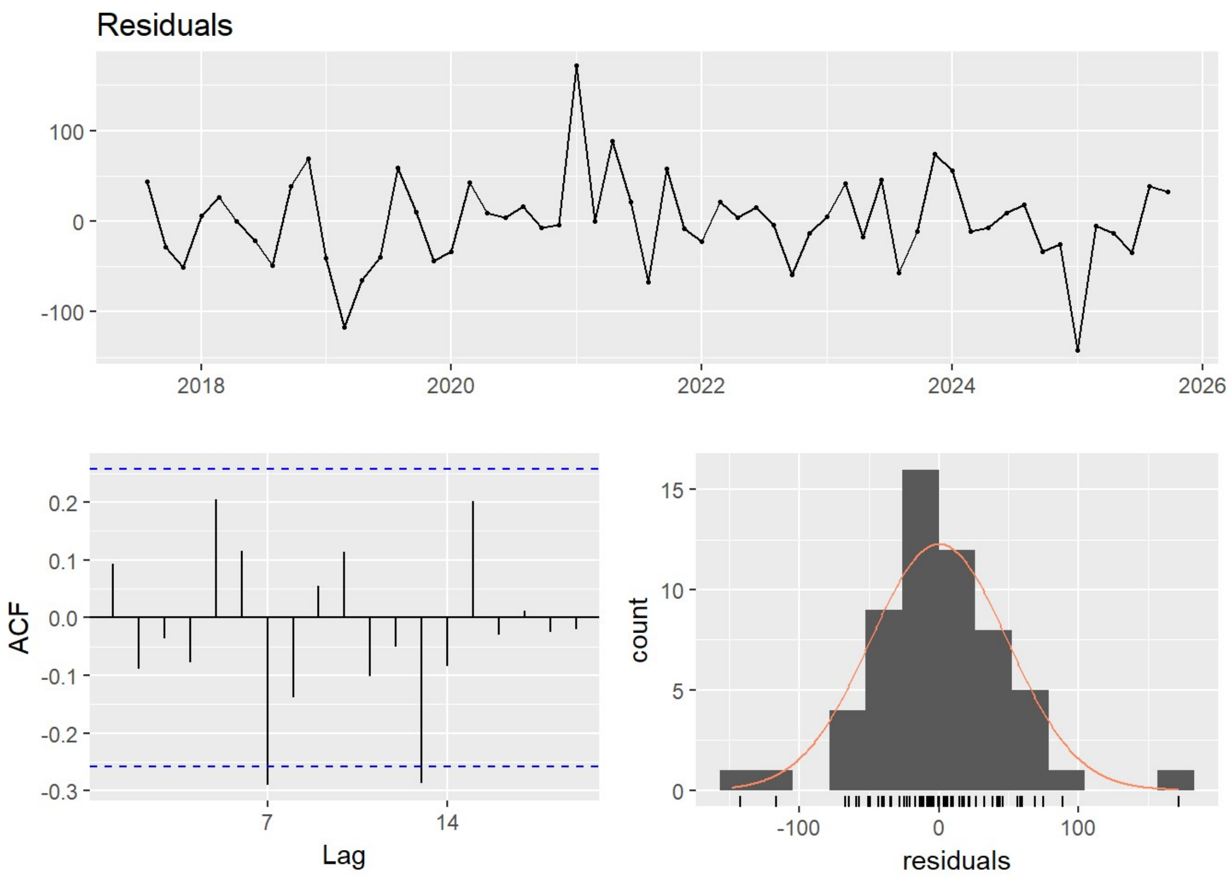
model1

To fit the model, we will use dynlm function of dynlm package. In the model formulation, the argument formula contains trend() to include a trend component and season() to include a seasonal component in the model. The following code chunk implements the dynamic linear regression model for the bitcoin1 series.

```
model1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + trend(Y.t) + season(Y.t))
summary(model1)
```

```
##
## Time series regression with "ts" data:
## Start = 2017(5), End = 2025(6)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t.1 + P.t + trend(Y.t) +
##     season(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -141.934  -27.504   -2.023   25.477  171.687
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    205.2268    74.7027   2.747  0.00849 **
## L(Y.t, k = 1)    0.5123     0.1527   3.355  0.00158 **
## P.t.1         -137.6174    69.6643  -1.975  0.05411 .
## P.t            329.1289    68.9834   4.771 1.82e-05 ***
## trend(Y.t)     -18.8966     9.8690  -1.915  0.06162 .
## season(Y.t)2    -7.1673    28.5188  -0.251  0.80266
## season(Y.t)3    -9.1206    27.5514  -0.331  0.74209
## season(Y.t)4   -13.9716    27.5520  -0.507  0.61446
## season(Y.t)5   -14.6934    26.8946  -0.546  0.58742
## season(Y.t)6    12.3315    26.9687   0.457  0.64960
## season(Y.t)7    13.4511    27.2705   0.493  0.62413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.25 on 47 degrees of freedom
## Multiple R-squared:  0.8812, Adjusted R-squared:  0.8559
## F-statistic: 34.85 on 10 and 47 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model1,test=FALSE)
```




```
bgtest(model1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1
## LM test = 2.6543, df = 1, p-value = 0.1033
```

Interpretation

As seen there is no seasonality, 1st lag of step function is insignificant and trend component is not captured by this model.

From residual check there is some indication of serial correlation but bgtest overrules it. There is slight deviation from normal distribution due to outliers.

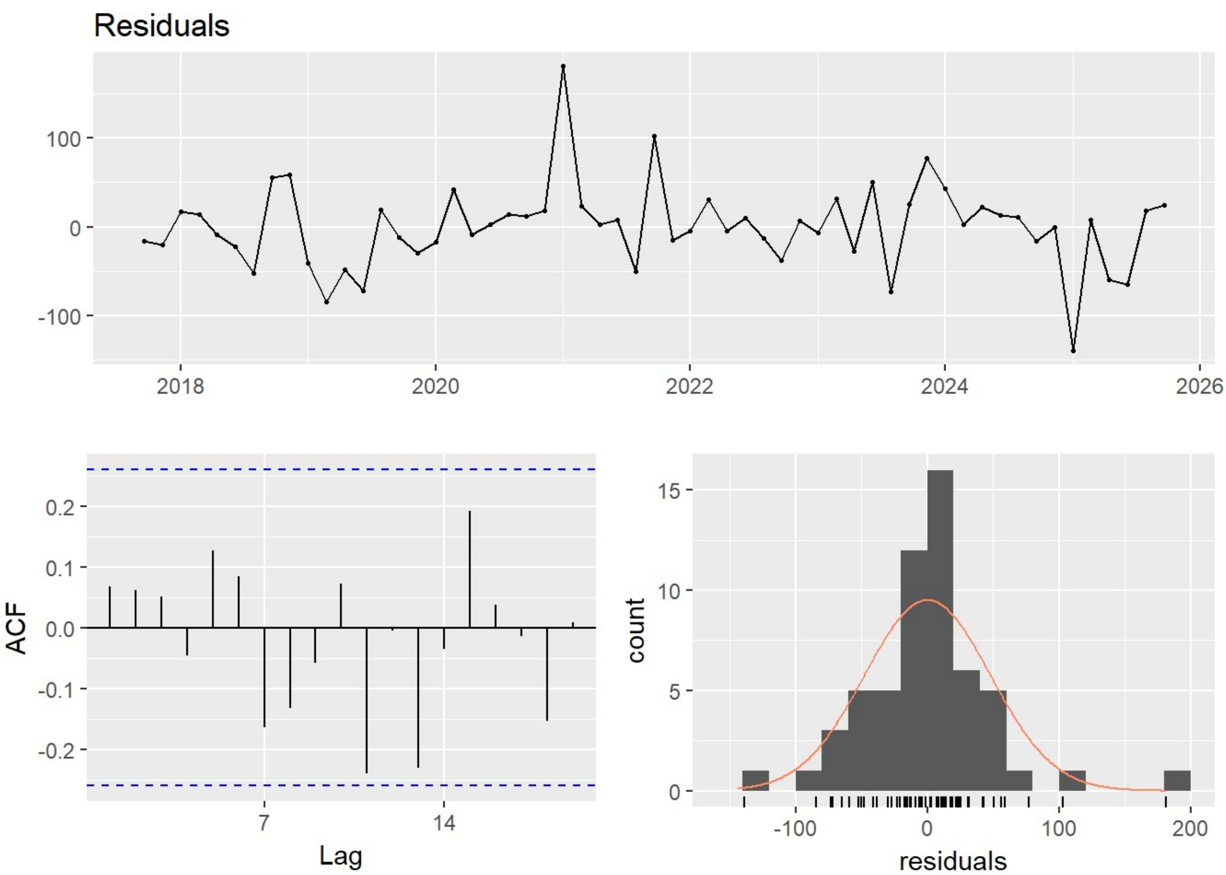
Considering these factors and in attempt to find a better model we go for model2 in which we ignore seasonality, first lag of step function, and add Yt???2 to the model as another predictor variable.

model2

```
model2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t + L(Y.t , k = 2 ) + trend(Y.t))
summary(model2)
```

```
##
## Time series regression with "ts" data:
## Start = 2017(6), End = 2025(6)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t + L(Y.t, k = 2) + trend(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -139.066  -20.486    2.539   18.923  180.991
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   303.1920    52.2488   5.803 3.94e-07 ***
## L(Y.t, k = 1)   0.6445     0.1416   4.552 3.23e-05 ***
## P.t           289.3796    55.5546   5.209 3.30e-06 ***
## L(Y.t, k = 2)  -0.3691     0.1081  -3.413 0.001250 **
## trend(Y.t)    -28.8658     7.4905  -3.854 0.000321 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.65 on 52 degrees of freedom
## Multiple R-squared:  0.8898, Adjusted R-squared:  0.8813
## F-statistic:   105 on 4 and 52 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model2,test=FALSE)
```



```
bgtest(model2)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2
## LM test = 0.64114, df = 1, p-value = 0.4233
```

Interpretation

Trend is now significant. sesonality and 1st lag of step function has been removed.

From residual check: There is no significant serial correlation between different points and distribution is more normal. But there is slight presence of trend in the time series plot which hasn't been captured. So the series is not completely random.

Comparing AIC, BIC, MASE of model1 and model2

```
models.AIC= AIC(model1, model2)
```

```
## Warning in AIC.default(model1, model2): models are not all fitted to the
## same number of observations
```

```
models.BIC= BIC(model1, model2)
```

```
## Warning in BIC.default(model1, model2): models are not all fitted to the
## same number of observations
```

```
sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
  }
}

AIC(model1)
```

```
## [1] 639.649
```

```r
AIC(model2)
```

```
## [1] 613.6851
```

```r
BIC(model1)
```

```
## [1] 664.3743
```

```r
BIC(model2)
```

```
## [1] 625.9434
```

```r
sort.score(models.AIC, "aic")
```

```
##        df      AIC
## model2  6 613.6851
## model1 12 639.6490
```

```r
sort.score(models.BIC, "bic")
```

```
##        df      BIC
## model2  6 625.9434
## model1 12 664.3743
```

```r
accuracy(model1)
```

```
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -3.922389e-15 48.83172 35.47457 -1.465601 8.687562 0.7665844
##                    ACF1
## Training set 0.09312413
```

```r
accuracy(model2)
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 2.990359e-15 47.41796 33.20777 -1.401102 8.158981 0.7141621
##                    ACF1
## Training set 0.06708535
```

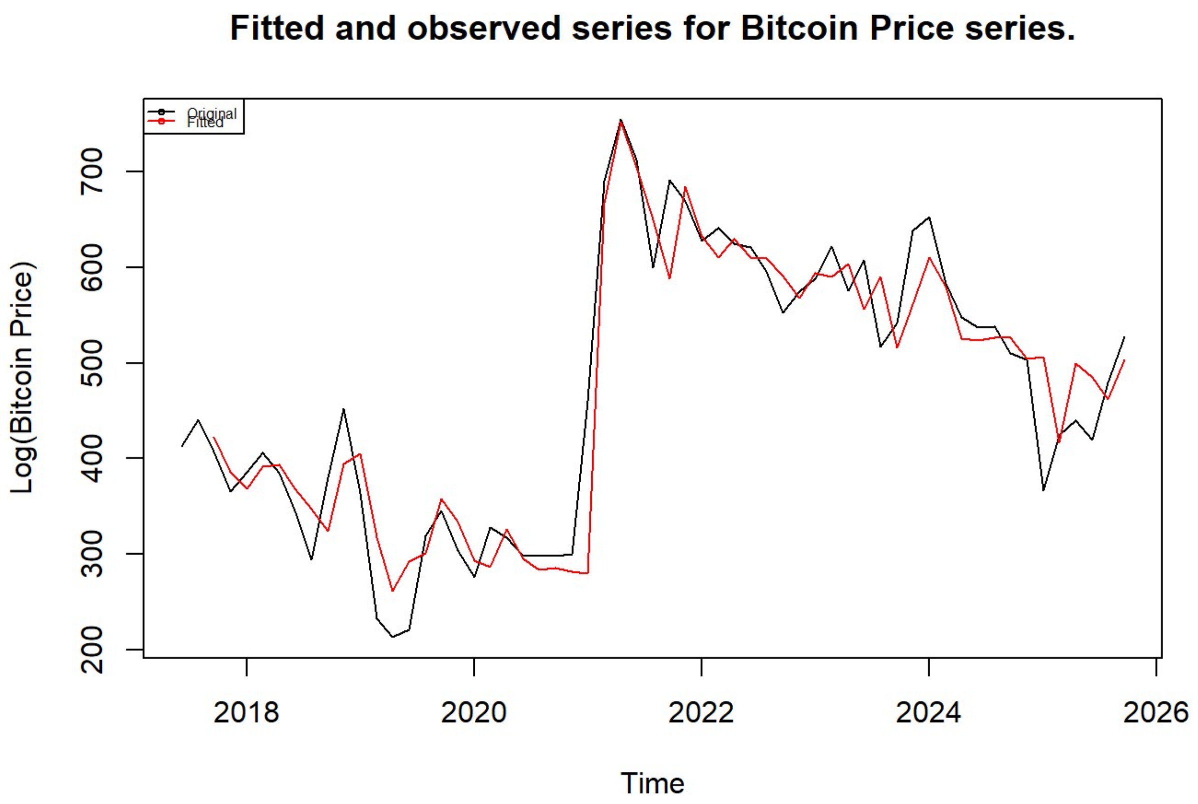
Interpretation

Considering AIC, BIC and MASE model2 has got lower values and hence is more suitable amongst the intervention models.

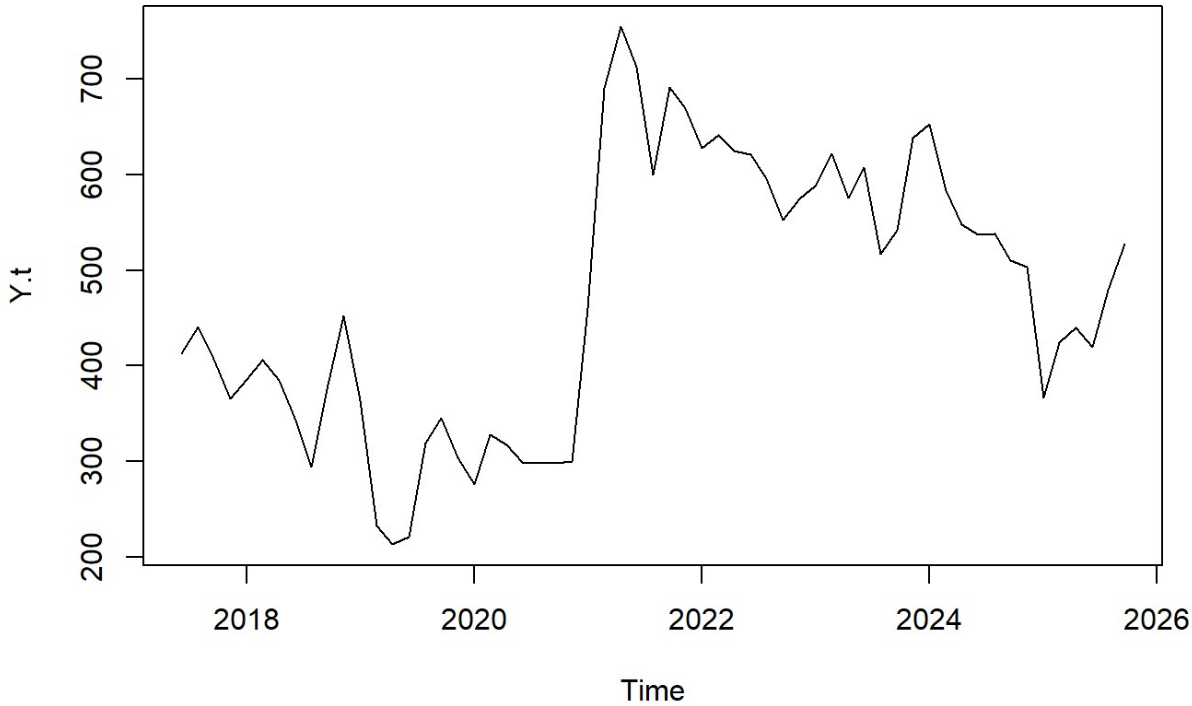
Therefore, the residuals, AIC, BIC and MASE values prove that the model2 for dynlm is the best model.

Observed and fitted values

```
plot(bitcoin1,ylab='Log(Bitcoin Price)',type="l", main="Fitted and observed series
for Bitcoin Price series.")
lines(model2$fitted.values,col="red")
legend("topleft",lty=1, pch = 1, col=c("black","red"), c("Original","Fitted"), cex=
0.5, y.intersp=0.5)
```




```
Y.t = bitcoin1
plot(Y.t)
```




```
q = 14
n = nrow(model2$model)
bitcoin.frc = array(NA , (n + q))
bitcoin.frc[1:n] = Y.t[15:length(Y.t)]
```

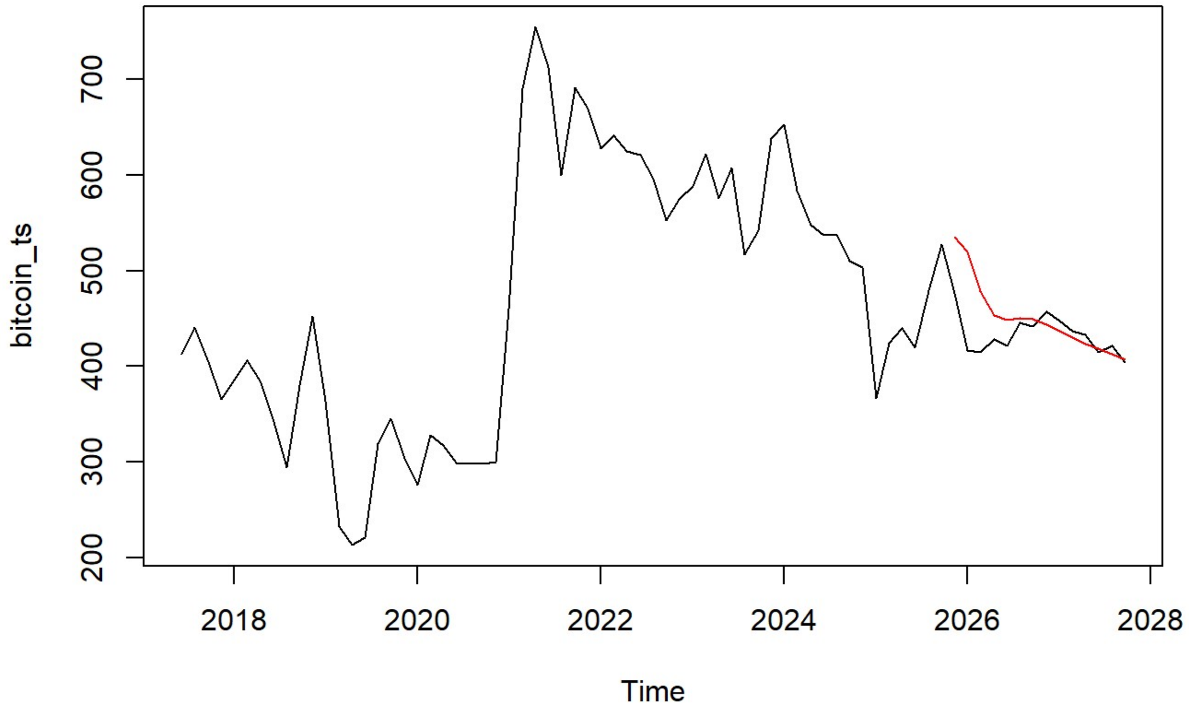
```
## Warning in bitcoin.frc[1:n] = Y.t[15:length(Y.t)]: number of items to
## replace is not a multiple of replacement length
```

```
trend = array(NA,q)
trend.start = model2$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q/7, 1/7)

for (i in 1:q){
  data.new = c(1,bitcoin.frc[n-1+i],P.t[n],bitcoin.frc[n-2+i] ,trend[i])
  bitcoin.frc[n+i] = as.vector(model2$coefficients) %*% data.new
}

par(mfrow=c(1,1))

# plot(Y.t,xlim=c(2018,2027),ylab='Bitcoin Price',xlab='Time',main = "Time series p
lot of Bitcoin Price.")
plot(bitcoin_ts)
lines(ts(bitcoin.frc[(n+1):(n+q)],start=c(2025,7),frequency = 7),col="red")
```



Interpretation

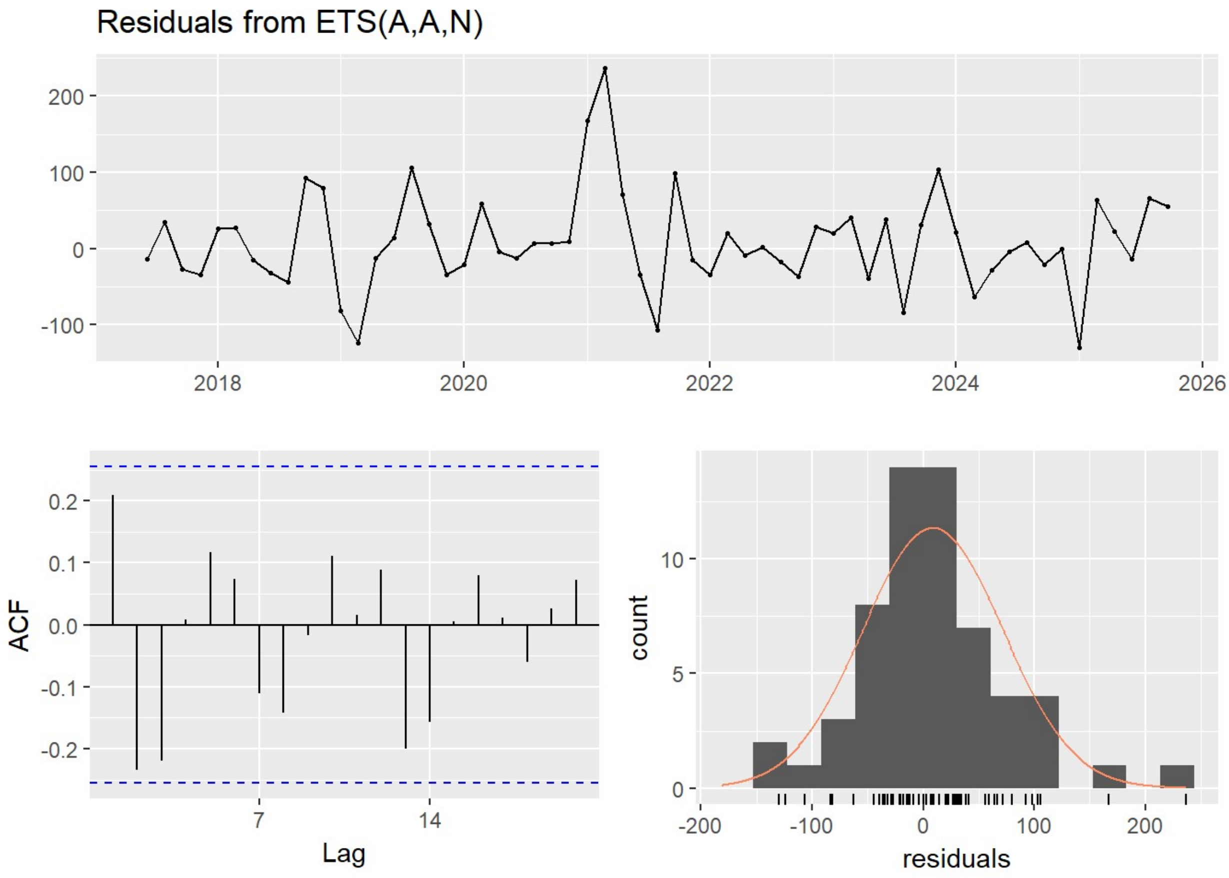
As seen in the above plot the model2 fits very nicely to the original data series. And the forecasts are also in accordance with the estimated values.

State Space Models

```
fit.AAN = ets(bitcoin1, model="AAN")
summary(fit.AAN)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = bitcoin1, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##
##   Initial states:
##     l = 433.8527
##     b = -6.9815
##
##   sigma:  63.4071
##
##      AIC     AICc      BIC
## 740.2246 741.3567 750.6123
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 8.552056 63.40706 45.49098 0.8896856 10.26763 0.3935579
##                   ACF1
## Training set 0.2095565
```

```
checkresiduals(fit.AAN)
```

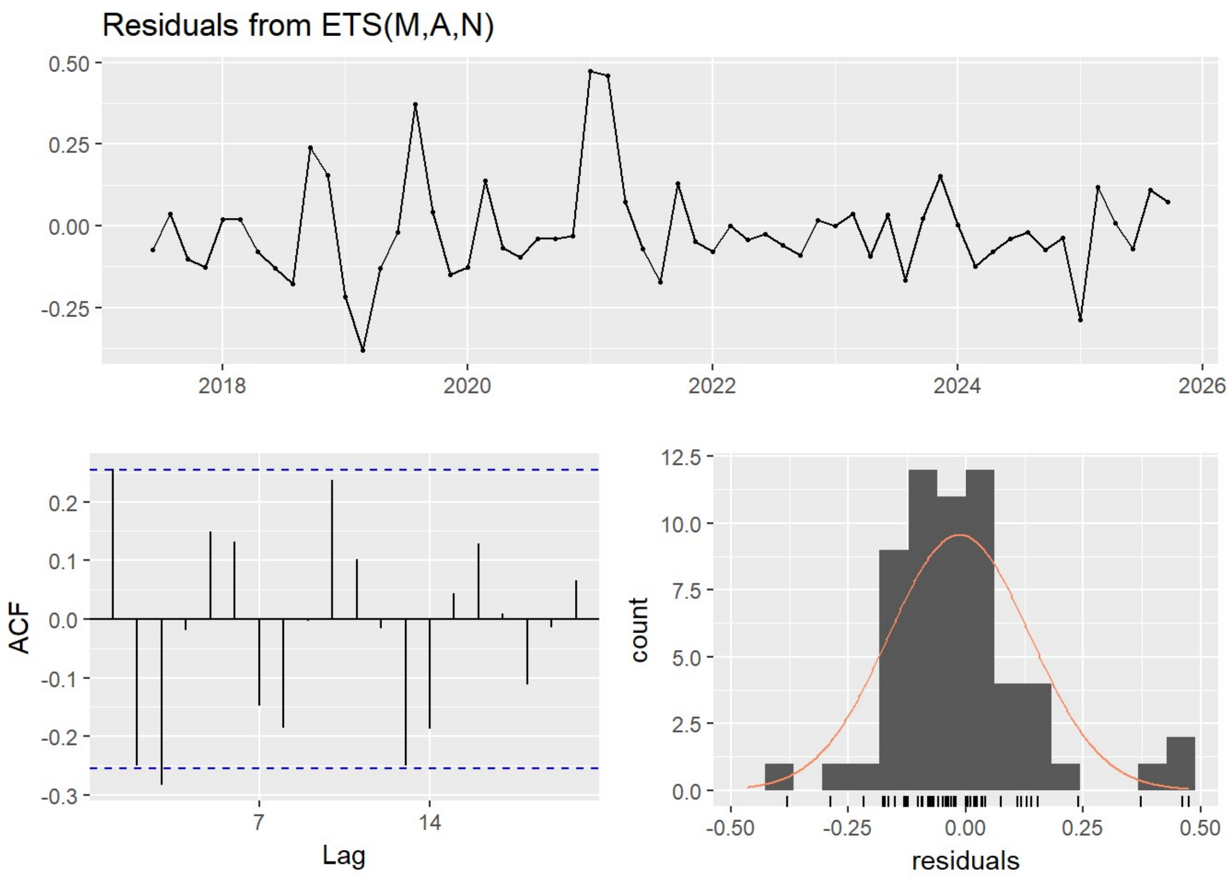


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 19.529, df = 10, p-value = 0.03403
##
## Model df: 4.   Total lags used: 14
```

```
fit.MAN = ets(bitcoin1, model="MAN")
summary(fit.MAN)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = bitcoin1, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##
##   Initial states:
##     l = 433.7501
##     b = 12.2142
##
##   sigma:  0.1498
##
##      AIC     AICc      BIC
## 750.2291 751.3612 760.6168
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE     MAPE      MASE
## Training set -10.58137 63.71289 47.4354 -3.60329 11.00602 0.4103798
##                   ACF1
## Training set 0.209562
```

```
checkresiduals(fit.MAN)
```

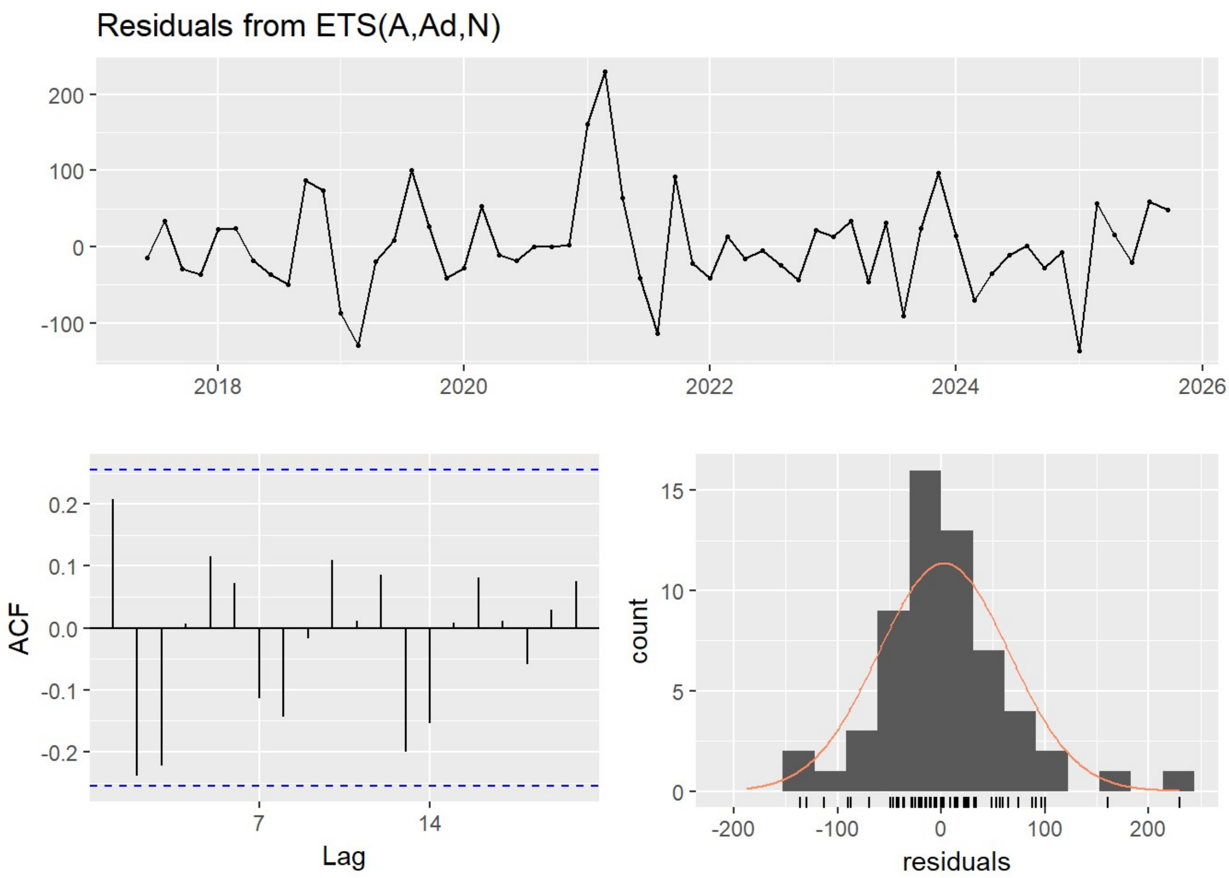



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 32.431, df = 10, p-value = 0.0003393
##
## Model df: 4.   Total lags used: 14
```

```
fit.AAdN = ets(bitcoin1, model="AAN", damped=TRUE)
summary(fit.AAdN)
```

```
## ETS(A,Ad,N)
##
## Call:
##  ets(y = bitcoin1, model = "AAN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##     phi   = 0.8646
##
##   Initial states:
##     l = 433.8802
##     b = -7.882
##
##   sigma:  62.777
##
##      AIC     AICc      BIC
## 741.0462 742.6616 753.5114
##
## Training set error measures:
##                    ME   RMSE     MAE        MPE     MAPE      MASE
## Training set 2.445443 62.777 45.3686 -0.5139576 10.28234 0.3924992
##                  ACF1
## Training set 0.207801
```

```
checkresiduals(fit.AAdN)
```

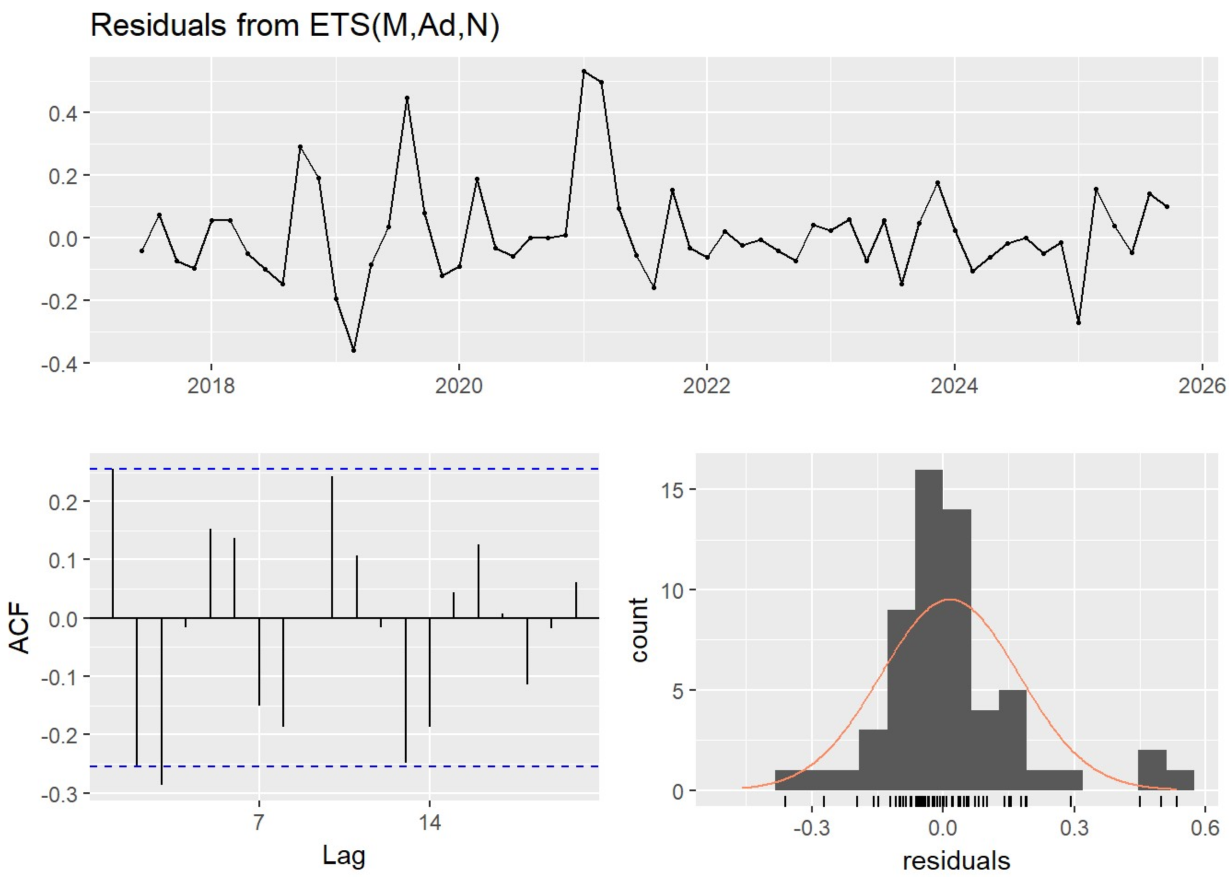


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,N)
## Q* = 19.567, df = 9, p-value = 0.02078
##
## Model df: 5.   Total lags used: 14
```

```
fit.MAdN = ets(bitcoin1, model="MAN", damped=TRUE)
summary(fit.MAdN)
```

```
## ETS(M,Ad,N)
##
## Call:
##  ets(y = bitcoin1, model = "MAN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##     phi   = 0.8
##
##   Initial states:
##     l = 434.1614
##     b = -3.4517
##
##   sigma:  0.1573
##
##      AIC     AICc      BIC
## 754.5860 756.2014 767.0512
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.822221 62.81807 45.48157 -0.6852466 10.32153 0.3934765
##                   ACF1
## Training set 0.2088827
```

```
checkresiduals(fit.MAdN)
```



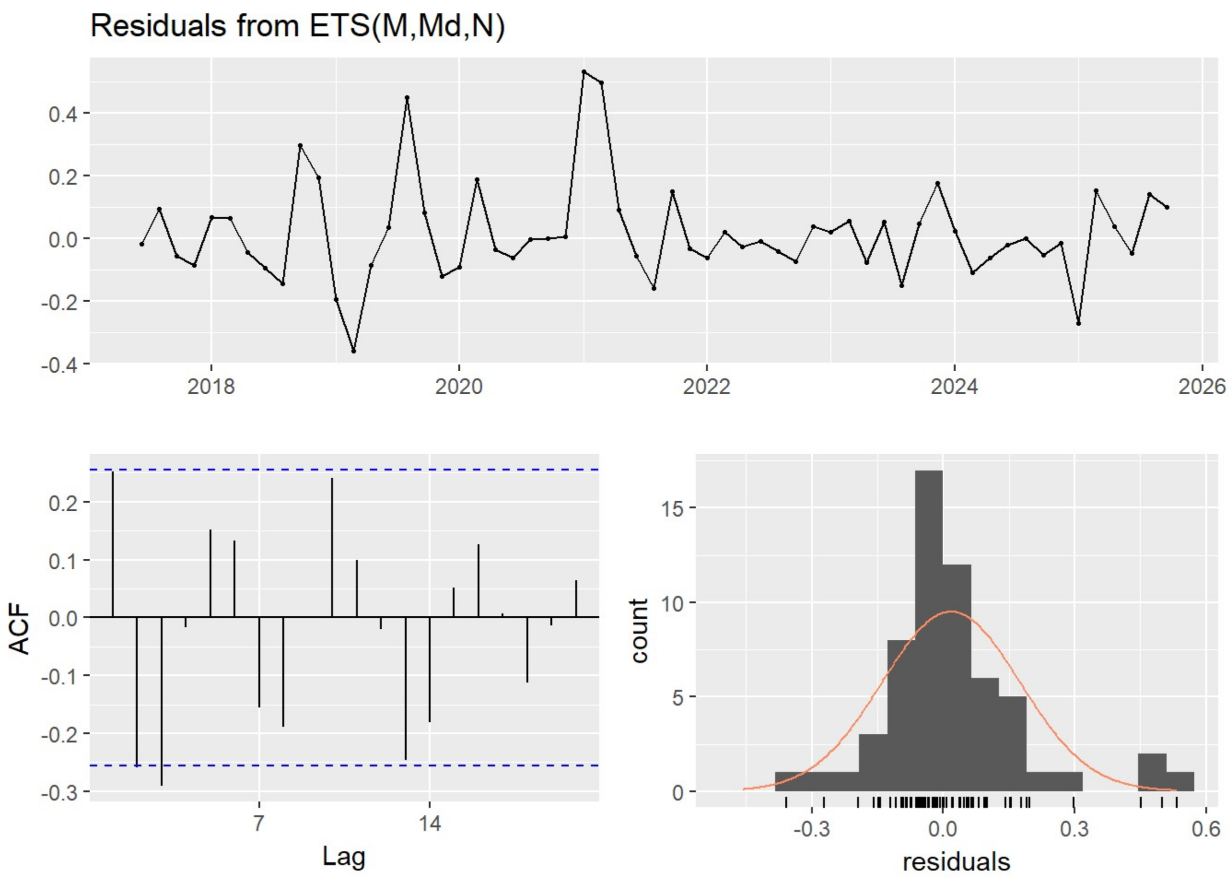

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,N)
## Q* = 33.031, df = 9, p-value = 0.0001319
##
## Model df: 5.   Total lags used: 14
```

```
#fit.AMN = ets(bitcoin1, model="AMN", damped=TRUE)
# Forbidden
```

```
fit.MMN = ets(bitcoin1, model="MMN", damped=TRUE)
summary(fit.MMN)
```

```
## ETS(M,Md,N)
##
## Call:
##  ets(y = bitcoin1, model = "MMN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##     phi   = 0.8118
##
##   Initial states:
##     l = 433.9336
##     b = 0.96
##
##   sigma:  0.1574
##
##      AIC     AICc      BIC
## 754.3753 755.9907 766.8405
##
## Training set error measures:
##                    ME     RMSE     MAE        MPE    MAPE      MASE
## Training set 2.764756 62.72971 45.2172 -0.4416929 10.2434 0.3911893
##                  ACF1
## Training set 0.207668
```

```
checkresiduals(fit.MMN)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Md,N)
## Q* = 33.017, df = 9, p-value = 0.0001327
##
## Model df: 5.   Total lags used: 14
```

Interpretation

Considering residual checks of all the state space models we can say that though all the models have little to negligible correlation, the distribution is near normal for all and time series plots are random but model AAN seems to be the best in terms of all the state space model in terms of no serial correlation, randomness of the series and the normal disrtribution.

Considering AIC, BIC and MASE of state space models

AIC

```
AIC(fit.AAN)
```

```
## [1] 740.2246
```

```
AIC(fit.MAN)
```

```
## [1] 750.2291
```

```
AIC(fit.AAdN)
```

```
## [1] 741.0462
```

```
AIC(fit.MAdN)
```

```
## [1] 754.586
```

```
AIC(fit.MMN)
```

```
## [1] 754.3753
```

BIC

```
BIC(fit.AAN)
```

```
## [1] 750.6123
```

```
BIC(fit.MAN)
```

```
## [1] 760.6168
```

```
BIC(fit.AAdN)
```

```
## [1] 753.5114
```

```
BIC(fit.MAdN)
```

```
## [1] 767.0512
```

```
BIC(fit.MMN)
```

```
## [1] 766.8405
```

MASE

```
accuracy(fit.AAN)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 8.552056 63.40706 45.49098 0.8896856 10.26763 0.3935579
##                   ACF1
## Training set 0.2095565
```

```
accuracy(fit.MAN)
```

```
##                     ME     RMSE     MAE      MPE     MAPE      MASE
## Training set -10.58137 63.71289 47.4354 -3.60329 11.00602 0.4103798
##                  ACF1
## Training set 0.209562
```

```
accuracy(fit.AAdN)
```

```
##                     ME   RMSE     MAE        MPE     MAPE      MASE
## Training set 2.445443 62.777 45.3686 -0.5139576 10.28234 0.3924992
##                   ACF1
## Training set 0.207801
```

```
accuracy(fit.MAdN)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.822221 62.81807 45.48157 -0.6852466 10.32153 0.3934765
##                    ACF1
## Training set 0.2088827
```

```
accuracy(fit.MMN)
```

```
##                     ME     RMSE     MAE        MPE    MAPE      MASE
## Training set 2.764756 62.72971 45.2172 -0.4416929 10.2434 0.3911893
##                   ACF1
## Training set 0.207668
```

Interpretation

Considering AIC, BIC and MASE AAN seems to be the best model amongst all the considered state space models.

Fitting and Forecasting using state space models by plotting for AAN.

```
frc.AAN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.AAN
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2025.857       520.9435 439.6841 602.2029 396.66792 645.2190
## 2026.000       514.0125 399.0943 628.9307 338.26033 689.7647
## 2026.143       507.0816 366.3314 647.8317 291.82279 722.3403
## 2026.286       500.1506 337.6196 662.6817 251.58081 748.7204
## 2026.429       493.2196 311.4962 674.9431 215.29765 771.1416
## 2026.571       486.2887 287.2113 685.3660 181.82617 790.7512
## 2026.714       479.3577 264.3194 694.3961 150.48494 808.2305
## 2026.857       472.4268 242.5300 702.3235 120.83000 824.0235
## 2027.000       465.4958 221.6416 709.3500  92.55298 838.4386
## 2027.143       458.5648 201.5074 715.6223  65.42935 851.7003
## 2027.286       451.6339 182.0165 721.2513  39.28964 863.9781
## 2027.429       444.7029 163.0828 726.3230  14.00218 875.4036
## 2027.571       437.7720 144.6380 730.9059 -10.53773 886.0816
## 2027.714       430.8410 126.6265 735.0555 -34.41498 896.0970
```

```
bitcoin2
```

```
## Time Series:
## Start = c(1, 1) 
## End = c(3, 1) 
## Frequency = 7 
##  [1] 527.88 476.05 416.26 415.09 428.50 421.03 445.80 441.83 457.31 447.81
## [11] 436.77 432.63 415.15 421.19 404.18
```

```
frc.MAN = forecast(fit.MAN, h =14) # Produce forecasts for MAN model
frc.MAN
```

```
##          Point Forecast    Lo 80     Hi 80      Lo 95     Hi 95
## 2025.857       540.0147 436.3207  643.7086 381.428443  698.6009
## 2026.000       552.1530 403.0371  701.2689 324.099874  780.2061
## 2026.143       564.2913 378.6065  749.9762 280.310846  848.2718
## 2026.286       576.4296 358.4607  794.3986 243.074877  909.7844
## 2026.429       588.5680 340.8602  836.2757 209.731655  967.4042
## 2026.571       600.7063 324.9269  876.4856 178.938179 1022.4744
## 2026.714       612.8446 310.1469  915.5423 149.908435 1075.7808
## 2026.857       624.9829 296.1899  953.7759 122.137450 1127.8284
## 2027.000       637.1212 282.8297  991.4128  95.279085 1178.9634
## 2027.143       649.2596 269.9034 1028.6157  69.084422 1229.4347
## 2027.286       661.3979 257.2897 1065.5061  43.367684 1279.4281
## 2027.429       673.5362 244.8950 1102.1774  17.986043 1329.0863
## 2027.571       685.6745 232.6459 1138.7032  -7.173026 1378.5221
## 2027.714       697.8128 220.4831 1175.1426 -32.200015 1427.8257
```

```
frc.AAdN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.AAdN
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2025.857       520.9435 439.6841 602.2029 396.66792 645.2190
## 2026.000       514.0125 399.0943 628.9307 338.26033 689.7647
## 2026.143       507.0816 366.3314 647.8317 291.82279 722.3403
## 2026.286       500.1506 337.6196 662.6817 251.58081 748.7204
## 2026.429       493.2196 311.4962 674.9431 215.29765 771.1416
## 2026.571       486.2887 287.2113 685.3660 181.82617 790.7512
## 2026.714       479.3577 264.3194 694.3961 150.48494 808.2305
## 2026.857       472.4268 242.5300 702.3235 120.83000 824.0235
## 2027.000       465.4958 221.6416 709.3500  92.55298 838.4386
## 2027.143       458.5648 201.5074 715.6223  65.42935 851.7003
## 2027.286       451.6339 182.0165 721.2513  39.28964 863.9781
## 2027.429       444.7029 163.0828 726.3230  14.00218 875.4036
## 2027.571       437.7720 144.6380 730.9059 -10.53773 886.0816
## 2027.714       430.8410 126.6265 735.0555 -34.41498 896.0970
```

```
frc.MAdN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.MAdN
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2025.857       520.9435 439.6841 602.2029 396.66792 645.2190
## 2026.000       514.0125 399.0943 628.9307 338.26033 689.7647
## 2026.143       507.0816 366.3314 647.8317 291.82279 722.3403
## 2026.286       500.1506 337.6196 662.6817 251.58081 748.7204
## 2026.429       493.2196 311.4962 674.9431 215.29765 771.1416
## 2026.571       486.2887 287.2113 685.3660 181.82617 790.7512
## 2026.714       479.3577 264.3194 694.3961 150.48494 808.2305
## 2026.857       472.4268 242.5300 702.3235 120.83000 824.0235
## 2027.000       465.4958 221.6416 709.3500  92.55298 838.4386
## 2027.143       458.5648 201.5074 715.6223  65.42935 851.7003
## 2027.286       451.6339 182.0165 721.2513  39.28964 863.9781
## 2027.429       444.7029 163.0828 726.3230  14.00218 875.4036
## 2027.571       437.7720 144.6380 730.9059 -10.53773 886.0816
## 2027.714       430.8410 126.6265 735.0555 -34.41498 896.0970
```

```
frc.MMN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.MMN
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2025.857       520.9435 439.6841 602.2029 396.66792 645.2190
## 2026.000       514.0125 399.0943 628.9307 338.26033 689.7647
## 2026.143       507.0816 366.3314 647.8317 291.82279 722.3403
## 2026.286       500.1506 337.6196 662.6817 251.58081 748.7204
## 2026.429       493.2196 311.4962 674.9431 215.29765 771.1416
## 2026.571       486.2887 287.2113 685.3660 181.82617 790.7512
## 2026.714       479.3577 264.3194 694.3961 150.48494 808.2305
## 2026.857       472.4268 242.5300 702.3235 120.83000 824.0235
## 2027.000       465.4958 221.6416 709.3500  92.55298 838.4386
## 2027.143       458.5648 201.5074 715.6223  65.42935 851.7003
## 2027.286       451.6339 182.0165 721.2513  39.28964 863.9781
## 2027.429       444.7029 163.0828 726.3230  14.00218 875.4036
## 2027.571       437.7720 144.6380 730.9059 -10.53773 886.0816
## 2027.714       430.8410 126.6265 735.0555 -34.41498 896.0970
```

```
plot(frc.AAN, ylab="Bitcoin Close Price",plot.conf=FALSE, type="l", fcol="red", xla
b="Year", main="Fitting and forecasts for state space models")
```

```
## Warning in plot.window(xlim, ylim, log, ...): "plot.conf" is not a
## graphical parameter
```

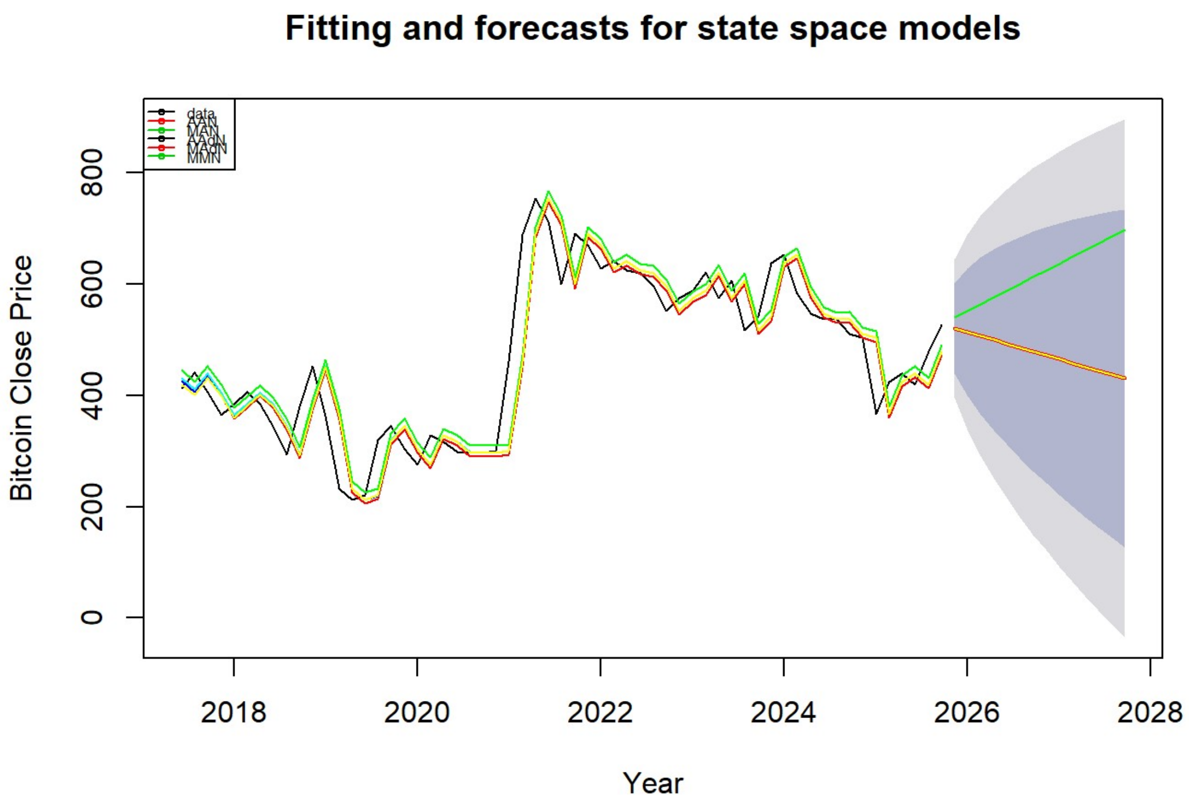
```
## Warning in title(main = main, xlab = xlab, ylab = ylab, ...): "plot.conf"
## is not a graphical parameter
```

```
## Warning in axis(1, ...): "plot.conf" is not a graphical parameter
```

```
## Warning in axis(2, ...): "plot.conf" is not a graphical parameter
```

```
## Warning in box(...): "plot.conf" is not a graphical parameter
```

```
lines(fitted(fit.AAN), col="red", lty=1)
lines(fitted(fit.MAN), col="green", lty=1)
lines(fitted(fit.AAdN), col="blue", lty=1)
lines(fitted(fit.MAdN), col="cyan", lty=1)
lines(fitted(fit.MMN), col="yellow", lty=1)
lines(frc.MAN$mean,col="green", type="l")
lines(frc.AAdN$mean,col="blue", type="l")
lines(frc.MAdN$mean,col="cyan", type="l")
lines(frc.MMN$mean,col="yellow", type="l")
legend("topleft",lty=1, pch=1, col=1:3, c("data","AAN", "MAN", "AAdN", "MAdN", "MM
N"), cex=0.5, y.intersp=0.5)
```



Interpretation

AAN is a better fitted model and has better AIC, BIC and MASE value amongst the state space models.

```
accuracy(ts(bitcoin.frc[(n+1):(n+q)], start=c(2025,7), frequency=7), bitcoin_ts[60:
73])
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set -17.78751 38.11337 24.74514 -4.167824 5.744349
```

```
accuracy(model2)
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 2.990359e-15 47.41796 33.20777 -1.401102 8.158981 0.7141621
##                    ACF1
## Training set 0.06708535
```

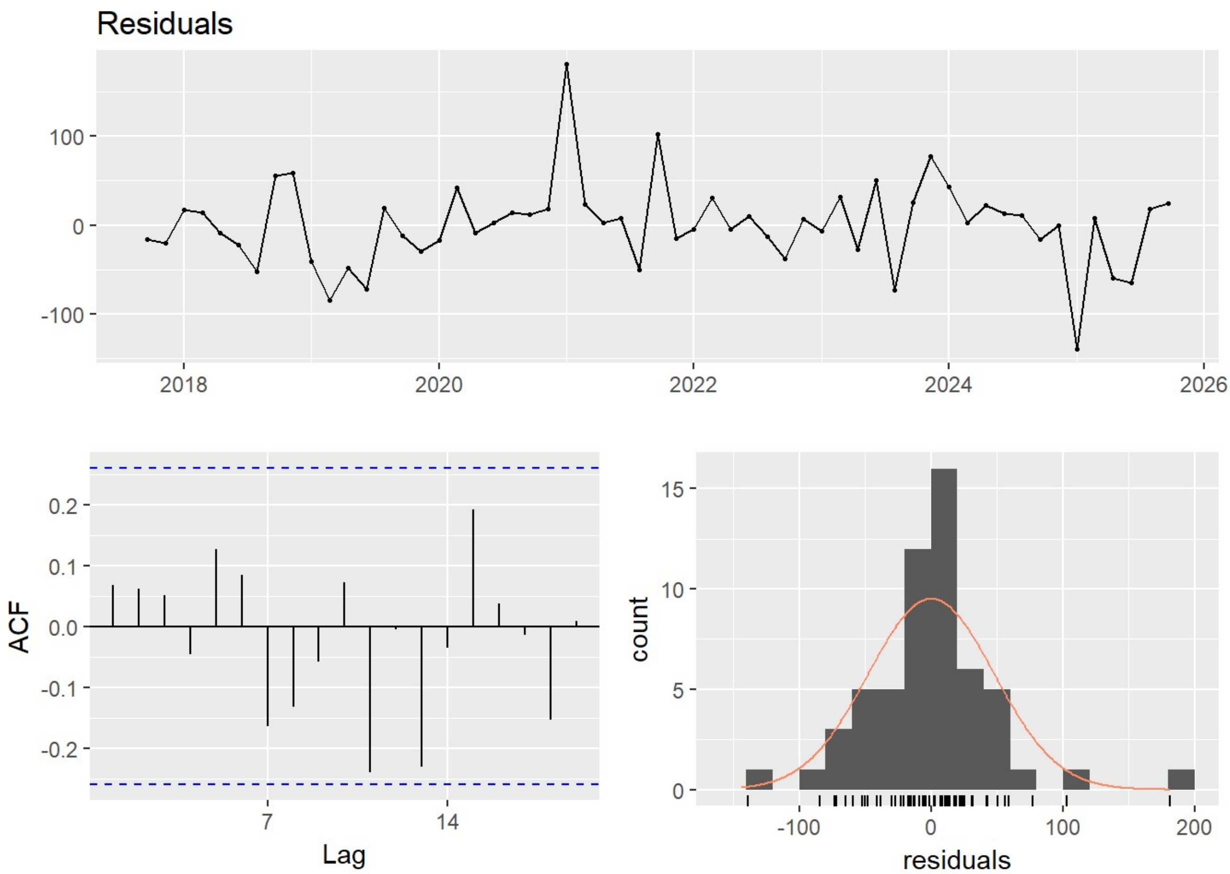
```
accuracy(fit.AAN)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 8.552056 63.40706 45.49098 0.8896856 10.26763 0.3935579
##                   ACF1
## Training set 0.2095565
```

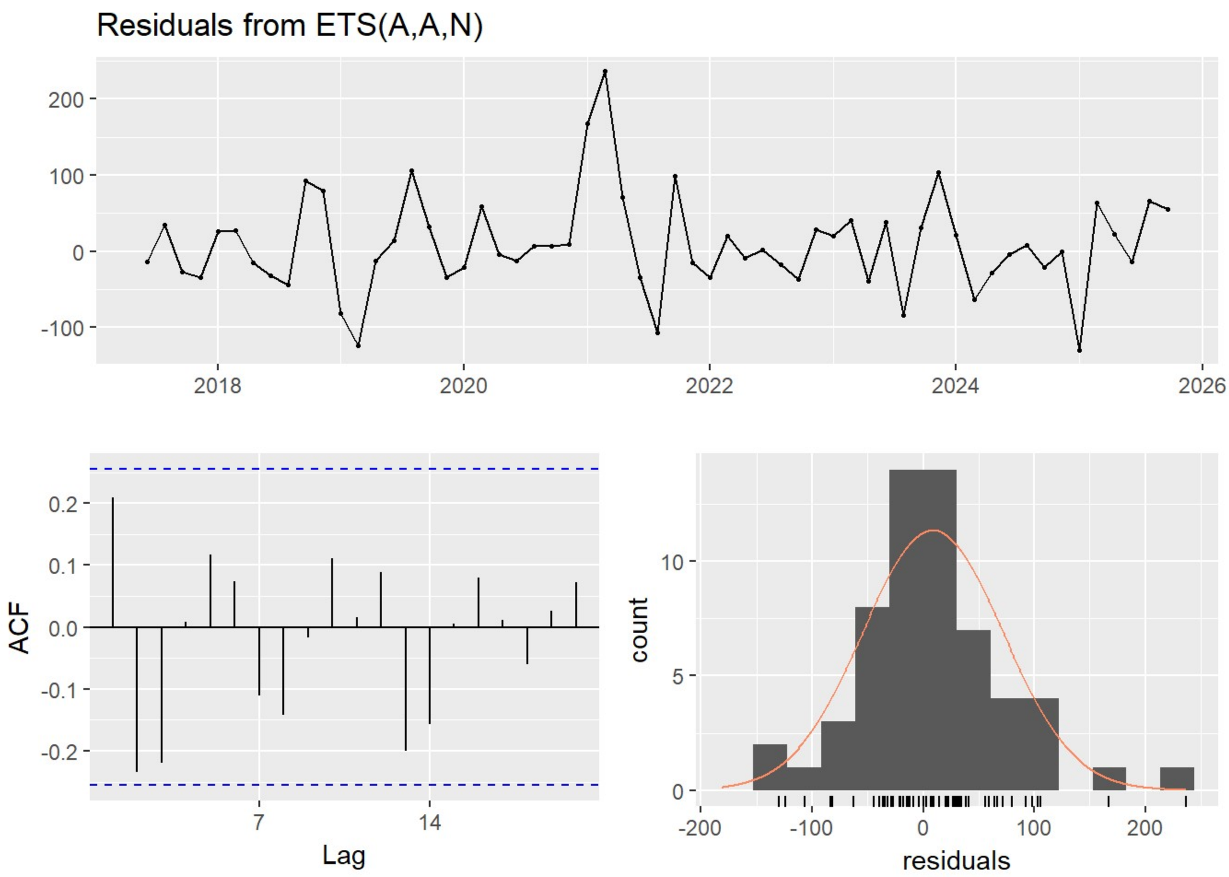
Interpretation

Considering MASE to compare the suitability of the models amongst best Intervention model and best state space model we find that MASE value is quite low for AAN model in comparison to the model2 intervention model. Considering MAPE values, percentage error between the forecasted and the estimated values is lower for Intervention model.

```
checkresiduals(model2, test=FALSE)
```




```
checkresiduals(fit.AAN, test=FALSE)
```



Interpretation

Considering the residuals it’s clearly seen that AAN model leads to more random series and residual plot is more nomally distributed for it. Hence, we choose AAN state space model as a better fit and hence choose it for forecasting.

Summary:

We had the data related to bitcoin cash price from Jan 23, 2017 to Oct, 03, 2017. Data was in reverse order so we changed it into ascending order irst and then converted it into time series. After that we divided the resultant series into two parts. 1:59 for analysis and 60:73 for estimating forecasts and named them as bitcoin1 and bitcoin2 respectively.

Working on bitcoin1, we first plotted the time series plot for it which showed that there is presence of trend, intervention, changing variance, moving average but no seasonality. And ACF and PACF plot further ensured our findings.

Since there was lucid intervention and trend the models selected were dynamic models (intervention models) and the state space models.

For model identification, various models were run for intervention analysis and state space models separartely. And considering the residual checks, AIC, BIC, MASE, the best model amongst intervention analysis models was model2 i.e. the model considering Yt-2 and trend with no seasonality and the intervention at T=27. And the best model amongst the state space models was AAN model i.e. model with additive trend, additive errors and no seasonality.

On fitting the models AAN gave the much better fit in comparison to the model2. Though MAPE values were lower for intervention model but since AAN model outperforms model2 in all other aspects and the little more percentage error is acceptable so we go along with AAN state space model as our bst fit.

Appendix

```r
# Installing  Libraries
#install.packages("dynlm")
#install.packages("lme4", repos="http://cran.rstudio.com/", dependencies=TRUE)
library(dynlm)
library(ggplot2)
library(AER)
#install.packages("swirl", repos="http://cran.rstudio.com/", dependencies=TRUE)
#install.packages("Hmisc")
#install.packages("checkmate", repos="http://cran.rstudio.com/", dependencies=TRUE)
library(Hmisc)
#install.packages("curl", repos="http://cran.rstudio.com/", dependencies=TRUE)
#install.packages("forecast")
library(forecast)
library(dLagM)
library(expsmooth)
library(TSA)
library(x12)
library(car)
library(ltsa)
library(readr)
library(bestglm)
library(FitAR)
library(xts)
library(seasonal)
bitcoin <- read.csv("C:/Users/user/Desktop/bitcoin_cash_price.csv")

head(bitcoin)

bitcoin_asc <- bitcoin[seq(dim(bitcoin)[1],1),]
row.names(bitcoin_asc) <- 1:nrow(bitcoin_asc)
head(bitcoin_asc)



bitcoin_ts <- ts(bitcoin_asc$Close, start= c(2017,4,23), frequency=7)
head(bitcoin_ts)


bitcoin1= bitcoin_ts[1:(length(bitcoin_ts)-14)]
bitcoin2= bitcoin_ts[(length(bitcoin_ts)-14): length(bitcoin_ts)]


bitcoin1 <-ts(bitcoin1, start=c(2017,4,23), frequency=7)
bitcoin2 <- ts(bitcoin2, frequency=7)



par(mfrow=c(1,1))
plot(bitcoin1, type="o",col = "black", ylab='Bitcoin Close Price',
     main = "Daily Bitcoin Close Price")
```

```
par(mfrow=c(1,2), mai=rep(0.9,4))
acf(bitcoin1, main="Sample ACF for Bitcoin time series")
pacf(bitcoin1, main="Sample PACF for Bitcoin time series")

library(tsoutliers)
library(fma)

plot(bitcoin_ts)
outlier.bitcoin <- tsoutliers::tso(bitcoin_ts,types = c("AO","LS","TC"),maxit.iloop
=10)
outlier.bitcoin
plot(outlier.bitcoin)

class(bitcoin1)
Y.t = bitcoin1
T = 27 # The time point when the intervention occurred
P.t = 1*(seq(bitcoin1) >= T)
P.t.1 = Lag(P.t,+1)
plot(Y.t)

model1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + trend(Y.t) + season(Y.t))
summary(model1)
checkresiduals(model1,test=FALSE)
bgtest(model1)


model2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t + L(Y.t , k = 2 ) + trend(Y.t))
summary(model2)
checkresiduals(model2,test=FALSE)
bgtest(model2)

models.AIC= AIC(model1, model2)
models.BIC= BIC(model1, model2)

sort.score <- function(x, score = c("bic", "aic")){
if (score == "aic"){
x[with(x, order(AIC)),]
} else if (score == "bic") {
x[with(x, order(BIC)),]
} else {
warning('score = "x" only accepts valid arguments ("aic","bic")')
}
}

AIC(model1)
AIC(model2)
BIC(model1)
BIC(model2)
sort.score(models.AIC, "aic")
sort.score(models.BIC, "bic")
accuracy(model1)
accuracy(model2)
```

```
plot(bitcoin1,ylab='Log(Bitcoin Price)',type="l", main="Fitted and observed series
for Bitcoin Price series.")
lines(model2$fitted.values,col="red")
legend("topleft",lty=1, pch = 1, col=c("black","red"), c("Original","Fitted"), cex=
0.5, y.intersp=0.5)

Y.t = bitcoin1
plot(Y.t)
q = 14
n = nrow(model2$model)
bitcoin.frc = array(NA , (n + q))
bitcoin.frc[1:n] = Y.t[15:length(Y.t)]

trend = array(NA,q)
trend.start = model2$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q/7, 1/7)

for (i in 1:q){
data.new = c(1,bitcoin.frc[n-1+i],P.t[n],bitcoin.frc[n-2+i] ,trend[i])
bitcoin.frc[n+i] = as.vector(model2$coefficients) %*% data.new
}

par(mfrow=c(1,1))


plot(bitcoin_ts)
lines(ts(bitcoin.frc[(n+1):(n+q)],start=c(2025,7),frequency = 7),col="red")


fit.AAN = ets(bitcoin1, model="AAN")
summary(fit.AAN)
checkresiduals(fit.AAN)


fit.MAN = ets(bitcoin1, model="MAN")
summary(fit.MAN)
checkresiduals(fit.MAN)


fit.AAdN = ets(bitcoin1, model="AAN", damped=TRUE)
summary(fit.AAdN)
checkresiduals(fit.AAdN)


fit.MAdN = ets(bitcoin1, model="MAN", damped=TRUE)
summary(fit.MAdN)
checkresiduals(fit.MAdN)


fit.MMN = ets(bitcoin1, model="MMN", damped=TRUE)
```

```
summary(fit.MMN)
checkresiduals(fit.MMN)


AIC(fit.AAN)
AIC(fit.MAN)
AIC(fit.AAdN)
AIC(fit.MAdN)
AIC(fit.MMN)


BIC(fit.AAN)
BIC(fit.MAN)
BIC(fit.AAdN)
BIC(fit.MAdN)
BIC(fit.MMN)


accuracy(fit.AAN)
accuracy(fit.MAN)
accuracy(fit.AAdN)
accuracy(fit.MAdN)
accuracy(fit.MMN)


frc.AAN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.AAN
bitcoin2
frc.MAN = forecast(fit.MAN, h =14) # Produce forecasts for MAN model
frc.MAN
frc.AAdN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.AAdN
frc.MAdN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.MAdN
frc.MMN = forecast(fit.AAN, h =14) # Produce forecasts for AAN model
frc.MMN
plot(frc.AAN, ylab="Bitcoin Close Price",plot.conf=FALSE, type="l", fcol="red", xla
b="Year", main="Fitting and forecasts for state space models")
lines(fitted(fit.AAN), col="red", lty=1)
lines(fitted(fit.MAN), col="green", lty=1)
lines(fitted(fit.AAdN), col="blue", lty=1)
lines(fitted(fit.MAdN), col="cyan", lty=1)
lines(fitted(fit.MMN), col="yellow", lty=1)
lines(frc.MAN$mean,col="green", type="l")
lines(frc.AAdN$mean,col="blue", type="l")
lines(frc.MAdN$mean,col="cyan", type="l")
lines(frc.MMN$mean,col="yellow", type="l")
legend("topleft",lty=1, pch=1, col=1:3, c("data","AAN", "MAN", "AAdN", "MAdN", "MM
N"), cex=0.5, y.intersp=0.5)


accuracy(ts(bitcoin.frc[(n+1):(n+q)], start=c(2025,7), frequency=7), bitcoin_ts[60:
```

```
73])
accuracy(model2)
accuracy(fit.AAN)

checkresiduals(model2, test=FALSE)

checkresiduals(fit.AAN, test=FALSE)
```