TSA Final Project

https://github.com/vivianzzzzz/ZhangXiaGupta_ENV797_TSA_FinalProject

Xiyue Zhang, Haochong Xia, Ayush Gupta

2024-04-26

```r
library(tidyverse)
library(openintro)
library(readxl)
library(lubridate)
library(dplyr)
library(forecast)
library(smooth)
library(zoo)
```

1.Data preprocessing

```r
bitcoin_raw <- read.csv("data/bitcoin.csv",header=TRUE)

bitcoin <-  bitcoin_raw[nrow(bitcoin_raw):1, ] %>%
  mutate(date = as.Date(Date, format="%m/%d/%Y")) %>%
  filter(!is.na('Change')) %>%
  select(date, Price, Change)

bitcoin$Change <- na.locf(bitcoin$Change, na.rm = FALSE)
bitcoin$Change <- gsub("%", "", bitcoin$Change)
bitcoin$Change = as.numeric(as.character(bitcoin$Change))

# Using median to avoid the influence of other potential outliers
median_value <- median(bitcoin$Change[bitcoin$Change != 0], na.rm = TRUE)
bitcoin$Change[bitcoin$Change == 0] <- median_value
```

```r
# Read the dataset
data <- read.csv("data/DCOILBRENTEU.csv")

# Convert DATE column to Date format
data$DATE <- as.Date(data$DATE)

# Create a sequence of dates from the start date to the end date
start_date <- min(data$DATE)
end_date <- max(data$DATE)
dates <- seq(start_date, end_date, by = "day")

# Filter out Fridays
fridays <- filter(data, weekdays(DATE) == "Friday")
```

```r
# Create a dataframe for Saturdays and Sundays
weekend_data <- data.frame(
  DATE = c(fridays$DATE + 1, fridays$DATE + 2),
  DCOILBRENTEU = rep(fridays$DCOILBRENTEU, each = 1)
)

# Combine the original data with the new weekend data
final_data <- bind_rows(data, weekend_data)

# Sort the final dataset by DATE
final_data <- final_data[order(final_data$DATE), ]

# Reset index
final_data <- final_data %>%
  arrange(DATE) %>%
  mutate(index = row_number()) %>%
  select(index, everything())

# View the modified dataset
#final_data

# Fill missing values with zeros
# final_data_filled <- final_data %>%
  # mutate(DCOILBRENTEU = ifelse(is.na(DCOILBRENTEU), 0, DCOILBRENTEU))


# Convert "." to NA in the DCOILBRENTEU column
final_data_filled <- final_data  %>%
  mutate(DCOILBRENTEU = ifelse(DCOILBRENTEU == ".", NA, DCOILBRENTEU))

# Replace NA values with the value right before it
final_data_filled <- final_data_filled %>%
  fill(DCOILBRENTEU)


# Convert string numbers to numerical values
final_data_filled$DCOILBRENTEU <- as.numeric(final_data_filled$DCOILBRENTEU)

# View the modified dataframe
#final_data_filled
```

1.1 Train test splits & Creating ts and msts

```r
# Only use the data after 2017-01-01
bitcoin <- bitcoin %>% filter(date >= ymd("2017-01-01"))

# Create a dummy variable to account for the covid effect. dummy = 1 from march 15 2020 to may 1st 2021
bitcoin <- bitcoin %>% mutate(covid = ifelse(date >= ymd("2020-03-15") & date <= ymd("2021-05-01"), 1, 

# Set the proportion for train(90%), test datasets(10%)
train_prop <- 0.9

# Set the number of observations for train, test datasets
```

```r
n_train <- floor(nrow(bitcoin) * train_prop)
n_val <- nrow(bitcoin) - n_train

# Split the data into train and test datasets
train_bitcoin <- bitcoin %>% filter(date >= ymd("2017-01-01") & date <= ymd("2023-01-01"))
test_bitcoin <- bitcoin %>% filter(date >= ymd("2023-07-18") )

head(train_bitcoin)
```

```
##         date    Price Change covid
## 1 2017-01-01    995.4   3.33     0
## 2 2017-01-02 1,017.00   2.17     0
## 3 2017-01-03 1,033.30   1.60     0
## 4 2017-01-04 1,135.40   9.88     0
## 5 2017-01-05    989.3 -12.86     0
## 6 2017-01-06    886.2 -10.43     0
```

```r
head(test_bitcoin)
```

```
##         date     Price Change covid
## 1 2023-07-18 29,866.80  -0.91     0
## 2 2023-07-19 29,909.70   0.14     0
## 3 2023-07-20 29,801.00  -0.36     0
## 4 2023-07-21 29,903.10   0.34     0
## 5 2023-07-22 29,788.90  -0.38     0
## 6 2023-07-23 30,085.90   1.00     0
```

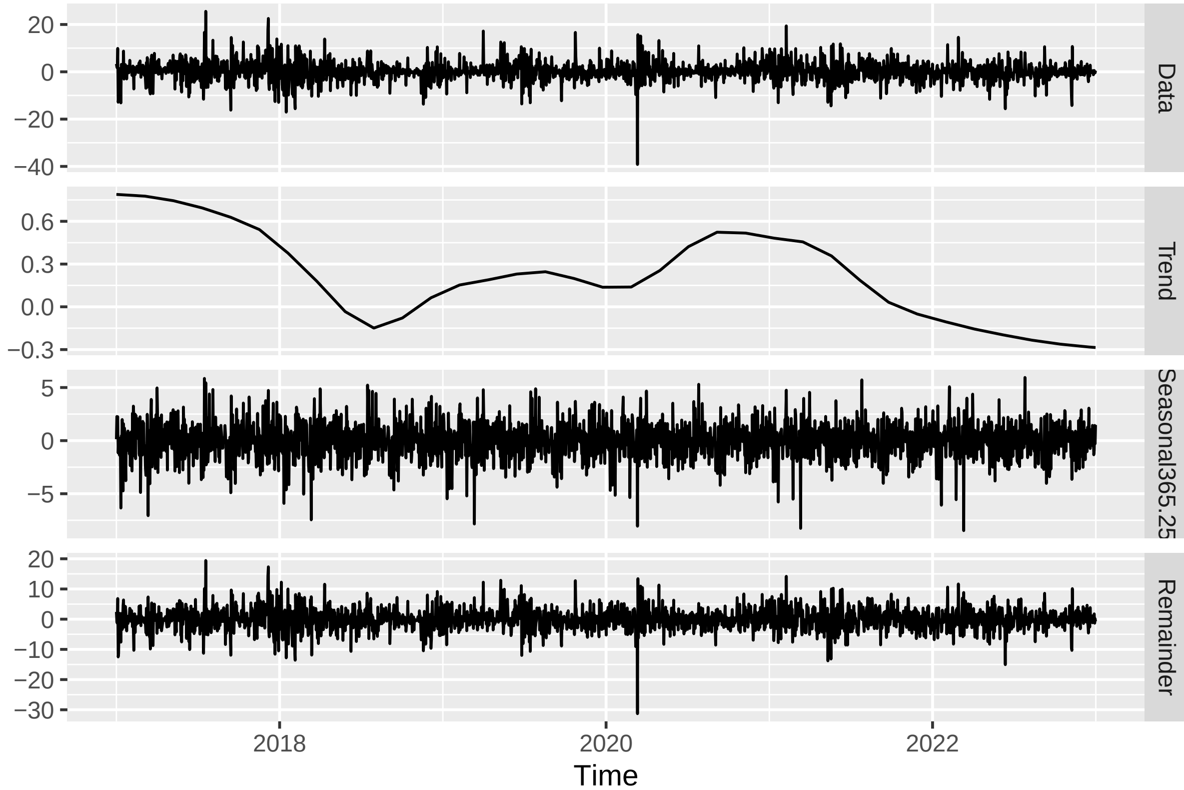
```r
ts_bitcoin <- ts(train_bitcoin$Change, frequency = 365.25, start = c(2017,01,01))
ts_bitcoin_test <- ts(test_bitcoin$Change, frequency = 365.25, start = c(2023,07,18))
msts_bitcoin <- msts(train_bitcoin$Change,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2017,01,01))
#Creating time series with seasonal pattern (quarterly, daily)
msts_bitcoin <- msts(train_bitcoin$Change,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2017,01,01))
msts_bitcoin_test <- msts(test_bitcoin$Change,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2023,07,18))
```

```r
# Set the proportion for train, test datasets
train_prop <- 0.9

# Set the number of observations for train, test datasets
n_train <- floor(nrow(final_data_filled) * train_prop)
n_val <- nrow(final_data_filled) - n_train

# Split the data into train and test datasets
train_oil <- final_data_filled %>% filter(DATE >= ymd("2017-01-01") & DATE <= ymd("2023-01-01"))
test_oil <- final_data_filled %>% filter(DATE >= ymd("2023-07-18") )
#Creating time series
ts_oil <- ts(train_oil$DCOILBRENTEU, frequency = 365.25, start = c(2017,01,01))
ts_oil_test <- ts(test_oil$DCOILBRENTEU, frequency = 365.25, start = c(2023,07,18))
msts_oil <- msts(train_oil$DCOILBRENTEU,
                           seasonal.periods =c( 91.25,365.25),
```

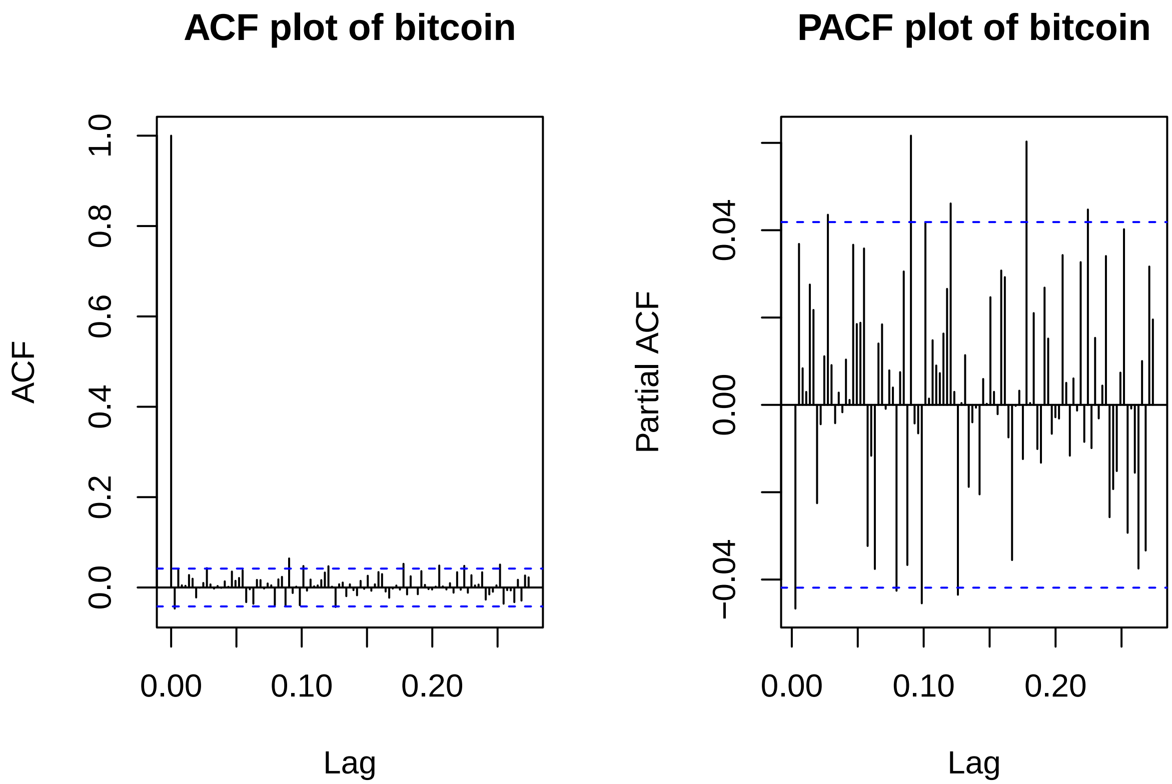
```
                              start=c(2017,01,01))
msts_oil_test <- msts(test_oil$DCOILBRENTEU,
                              seasonal.periods =c( 91.25,365.25),
                              start=c(2023,07,18))
```

2.Plot time series and ACF and PACF

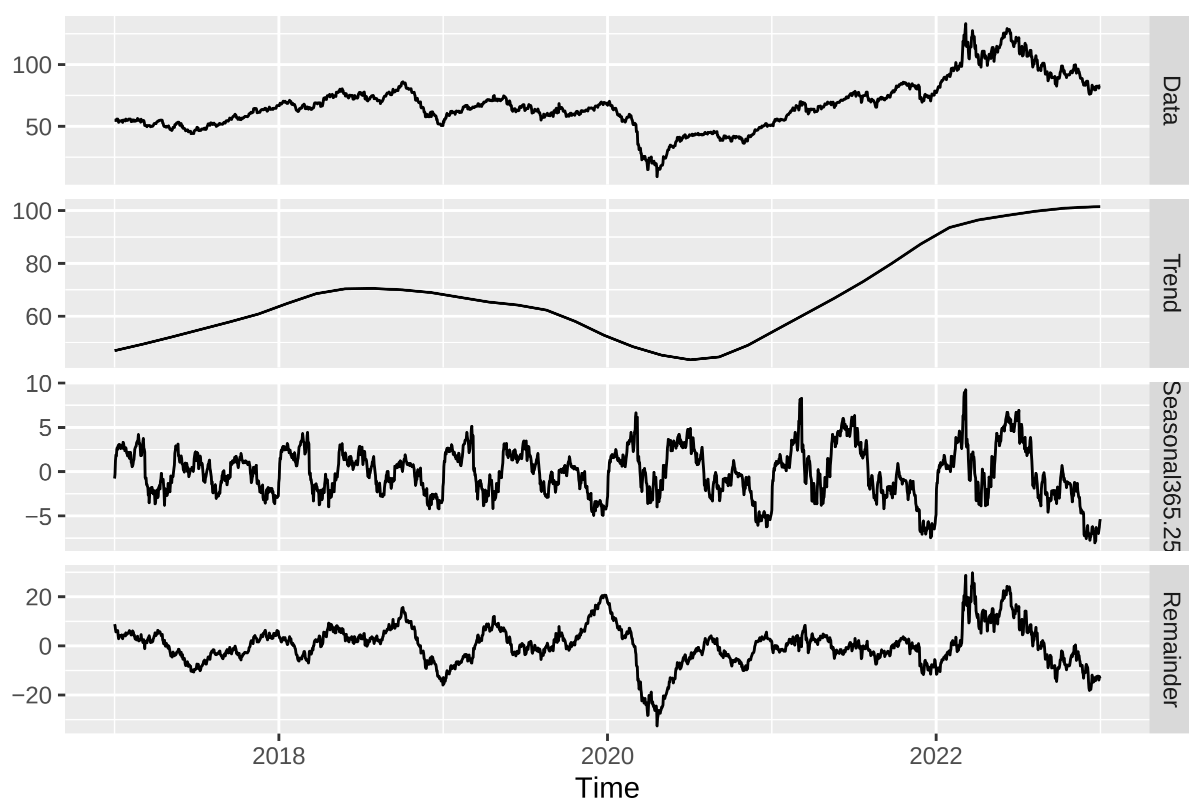
```
#plot the time series
ts_bitcoin %>% mstl() %>%
  autoplot()
```



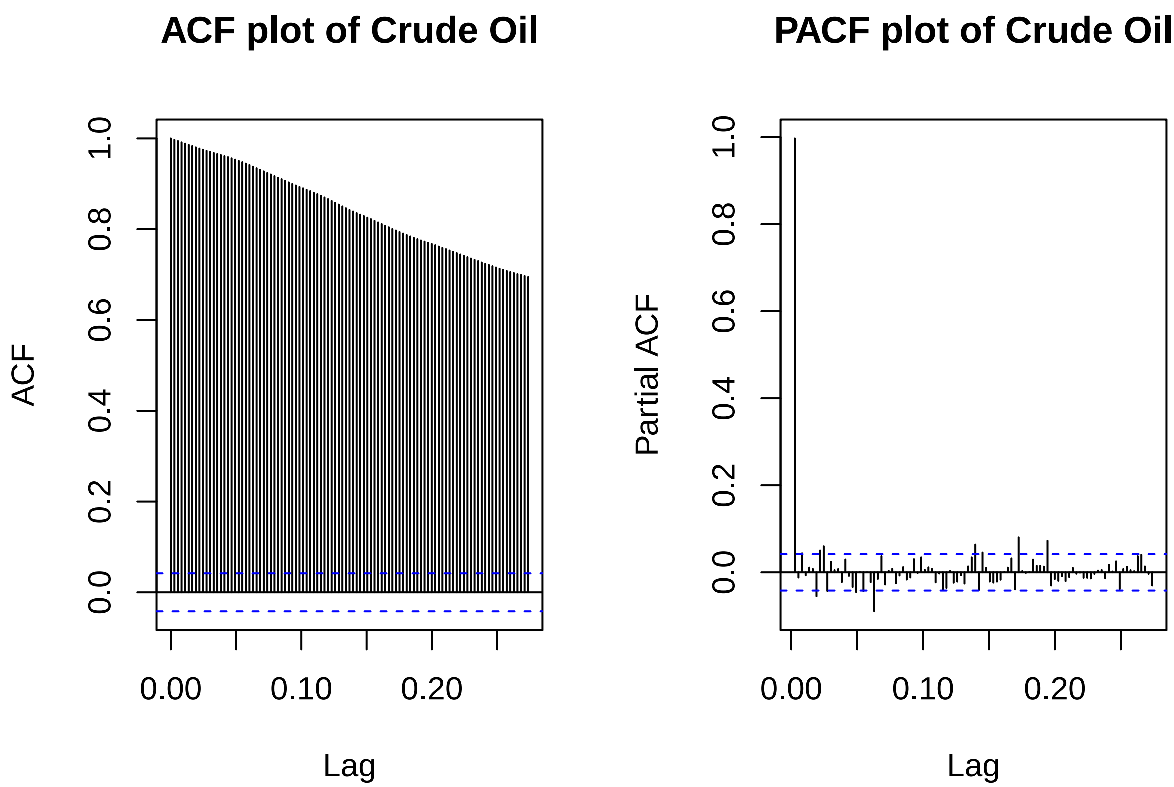

```
# Generate ACF and PACF plots
par(mfrow=c(1,2))
acf(ts_bitcoin,lag.max = 100, main= "ACF plot of bitcoin")
pacf(ts_bitcoin, lag.max = 100, main="PACF plot of bitcoin")
```




```r
#plot the time series
ts_oil %>% mstl() %>%
  autoplot()
```



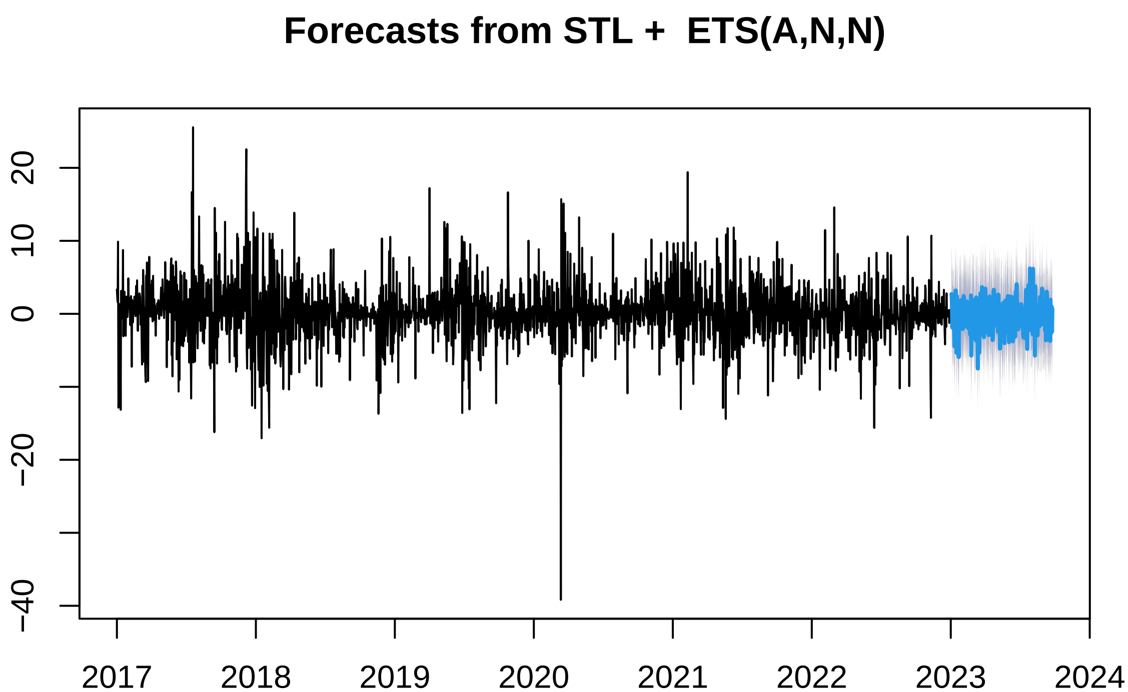
```r
# Generate ACF and PACF plots

par(mfrow=c(1,2))
acf(ts_oil,lag.max = 100, main= "ACF plot of Crude Oil")
pacf(ts_oil, lag.max = 100, main="PACF plot of Crude Oil")
```



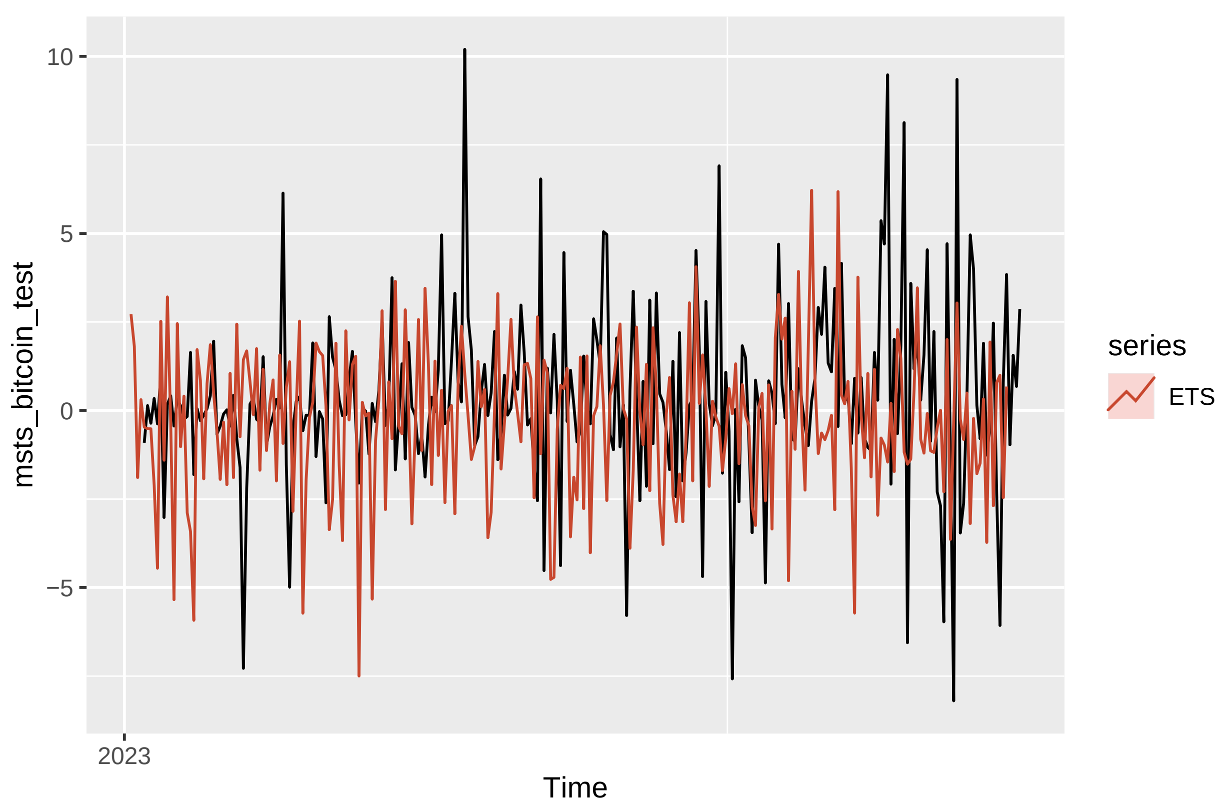
3. Models fit

3.1 STL + ETS model

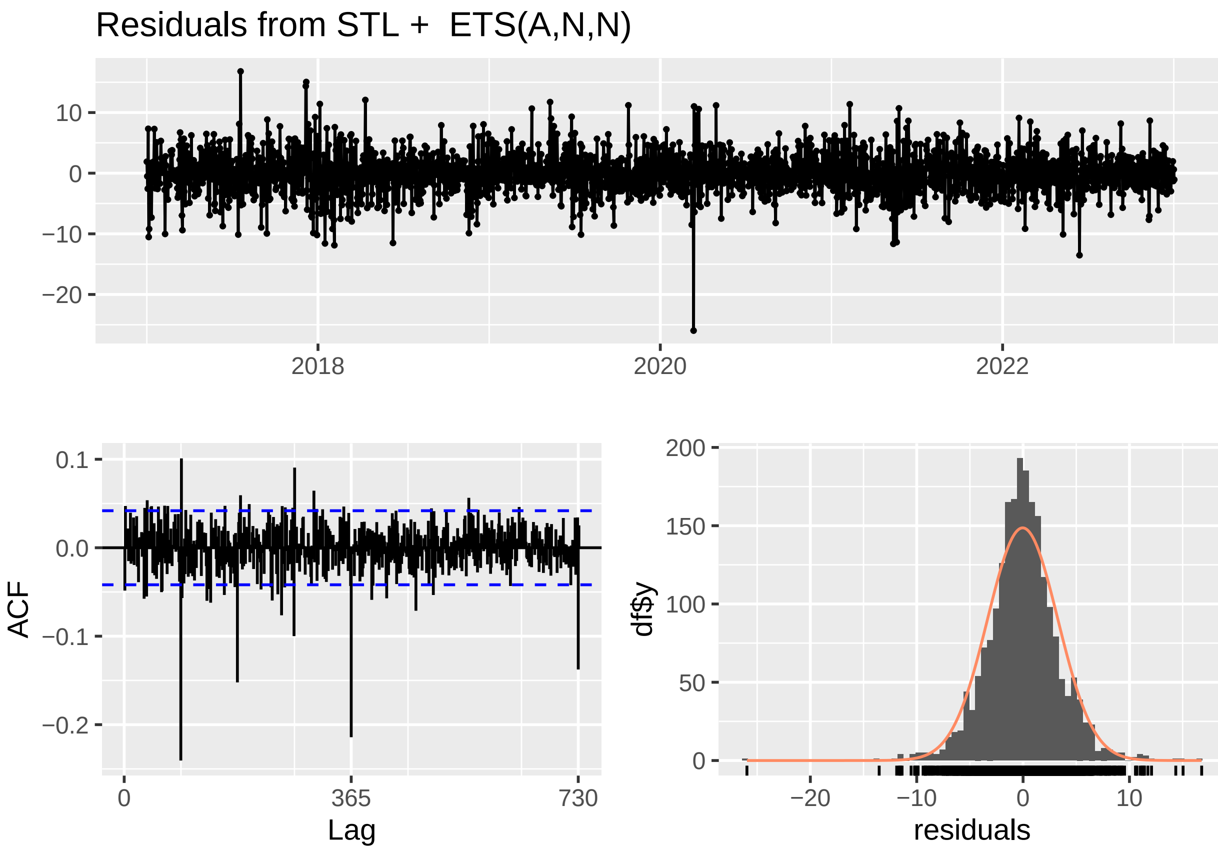
```r
# STL + ETS model
ETS_fit <-  stlf(msts_bitcoin,h=266)

plot(ETS_fit)
```




```r
autoplot(msts_bitcoin_test) + autolayer(ETS_fit, series="ETS",PI=FALSE)
```



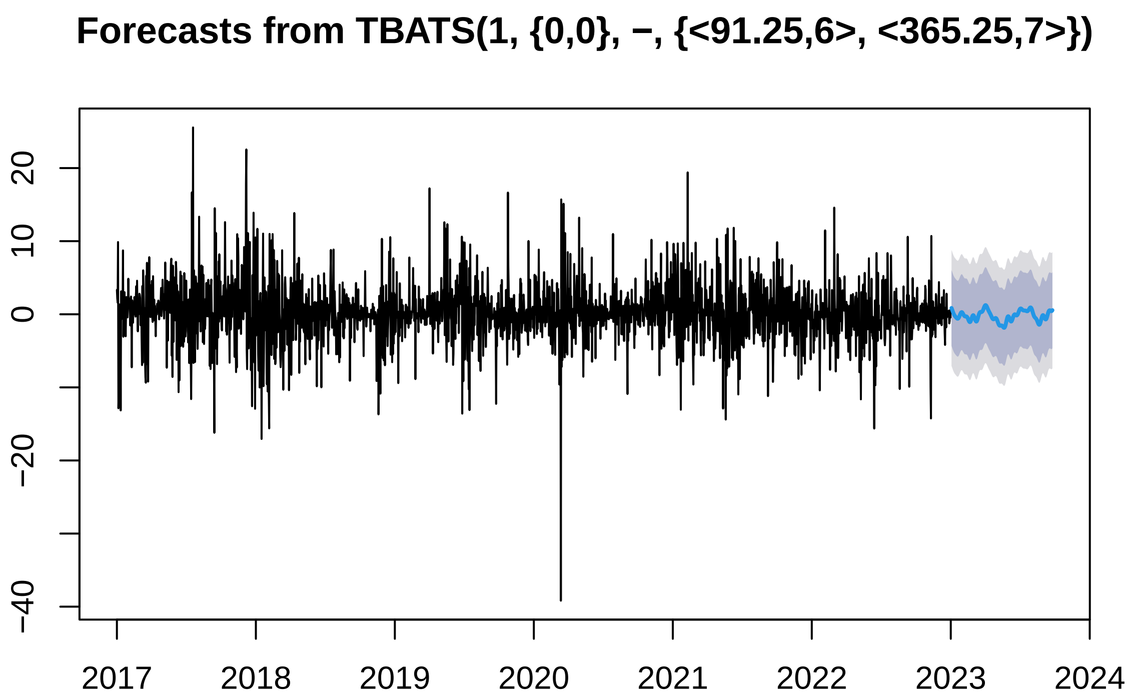

```r
ETS_scores <- accuracy(ETS_fit$mean,msts_bitcoin_test)
#print(ETS_scores)

checkresiduals(ETS_fit)
```



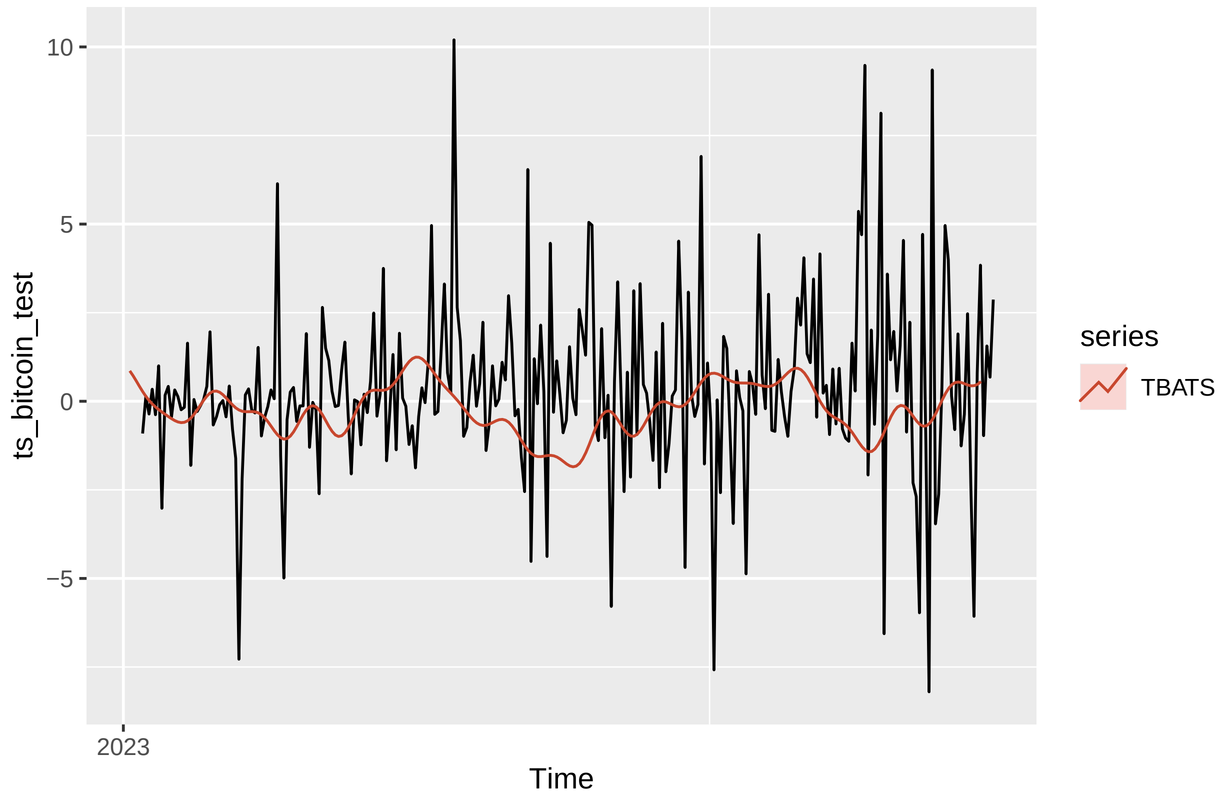

```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,N,N)
## Q* = 1008.2, df = 438, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 438
```

3.2 TBATS model

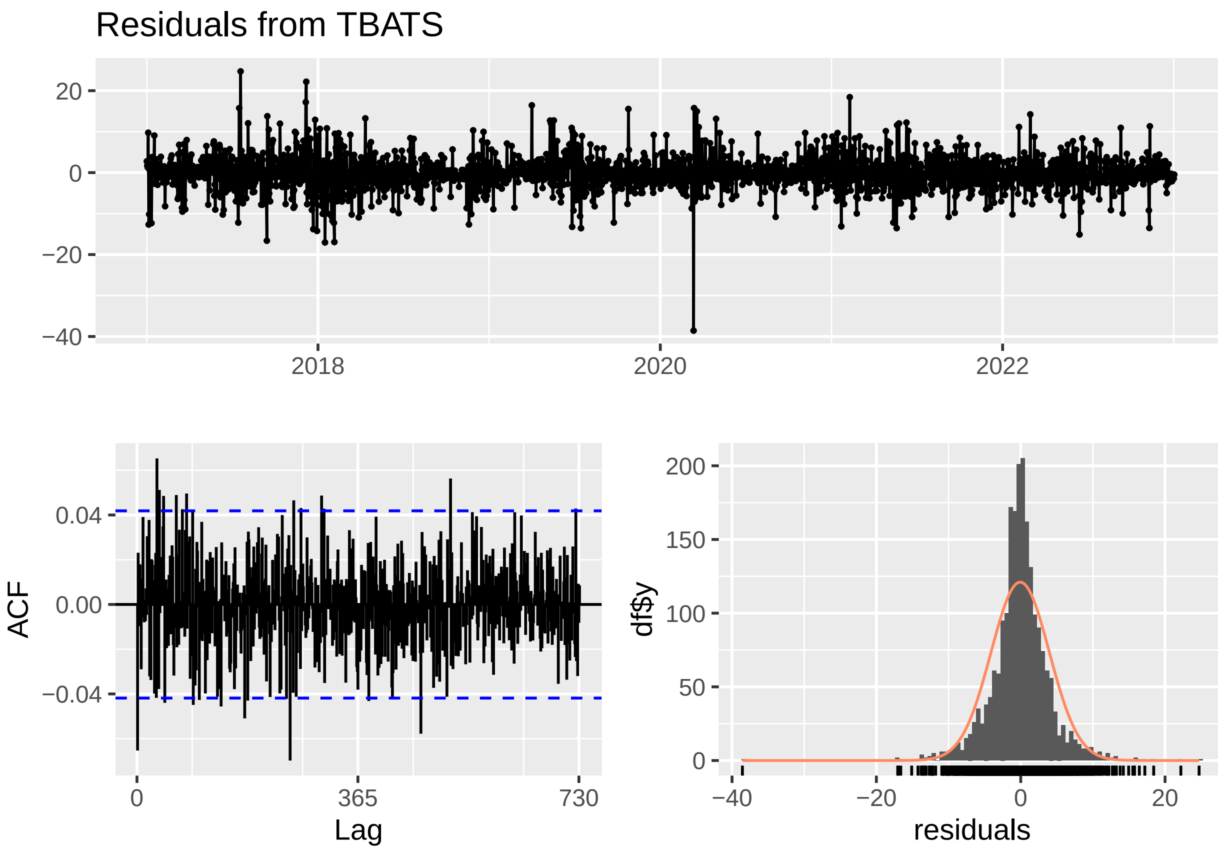
```r
# TBATS model
TBATS_fit <- tbats(msts_bitcoin)
TBATS_forcast <- forecast(TBATS_fit, h=266)
plot(TBATS_forcast)
```




```r
autoplot(ts_bitcoin_test) + autolayer(TBATS_forcast, series="TBATS", PI=FALSE)
```



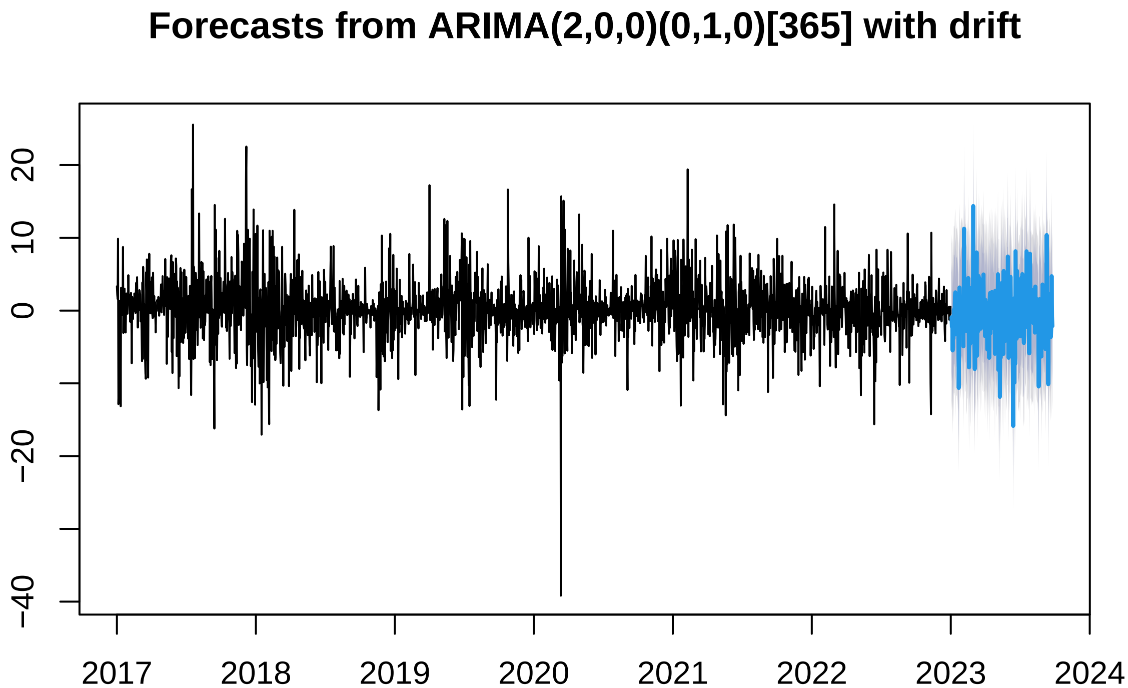

```r
TBATS_scores <- accuracy(TBATS_forcast$mean,msts_bitcoin_test)
#print(TBATS_scores)
checkresiduals(TBATS_fit)
```



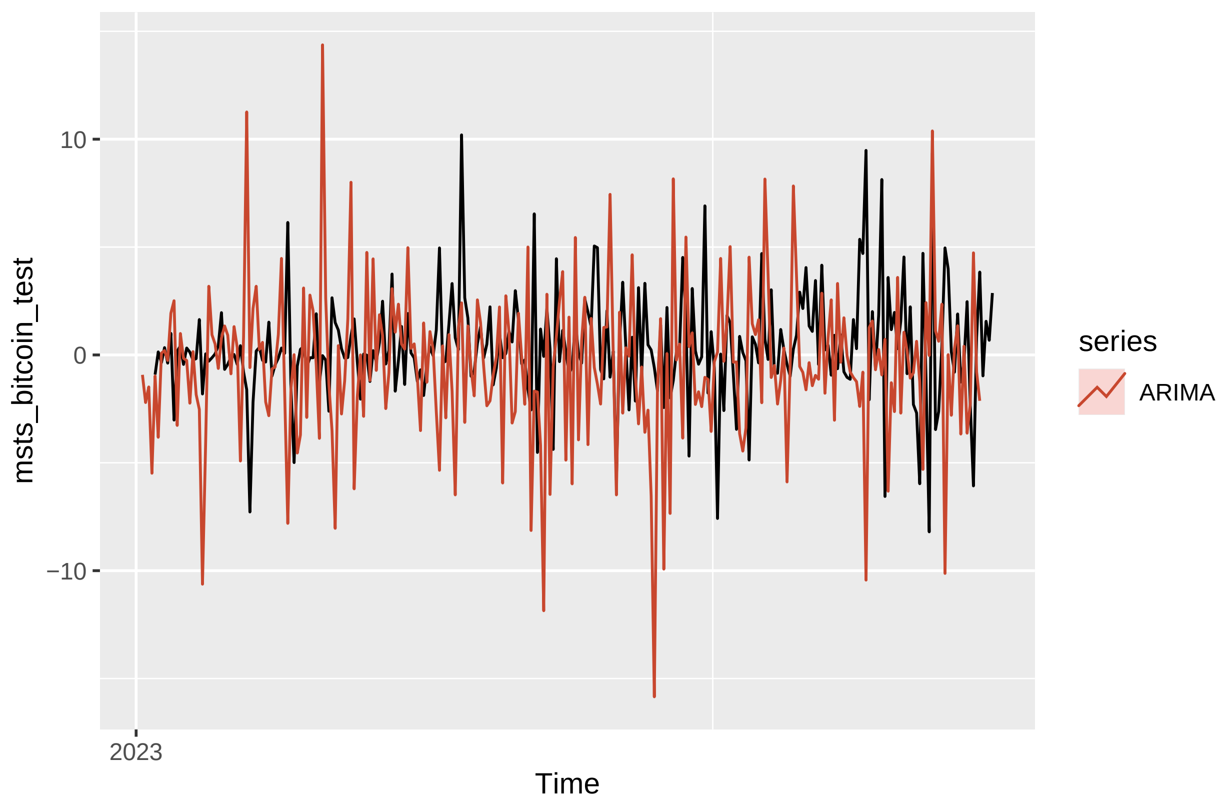

```
##
##  Ljung-Box test
##
## data:  Residuals from TBATS
## Q* = 449.5, df = 438, p-value = 0.3417
##
## Model df: 0.   Total lags used: 438
```

3.3 Arima with seasonality

```
#Arima with seasonality
arima_forecast <- forecast(auto.arima(msts_bitcoin,D=1),h=266)

plot(arima_forecast)
```

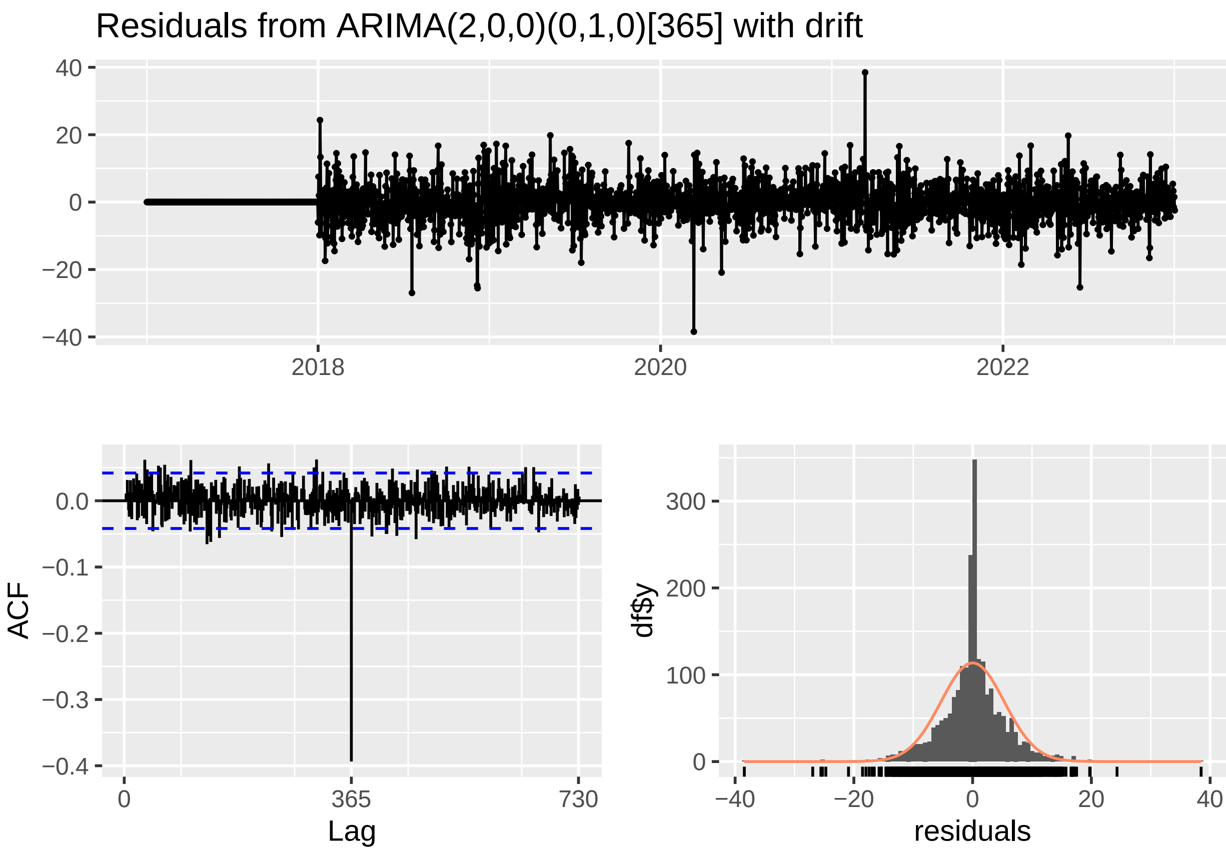



```
autoplot(msts_bitcoin_test) + autolayer(arima_forecast, series="ARIMA", PI=FALSE)
```




```
ARIMA_scores <- accuracy(arima_forecast$mean,msts_bitcoin_test)
#print(ARIMA_scores)
```

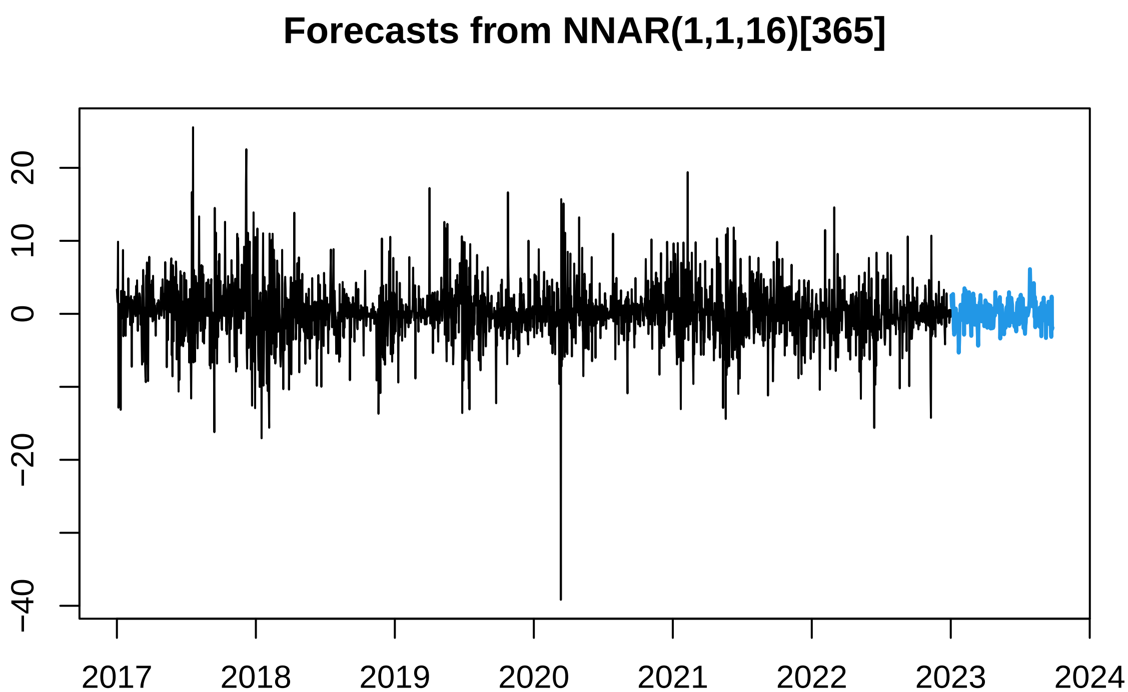
```r
checkresiduals(arima_forecast)
```



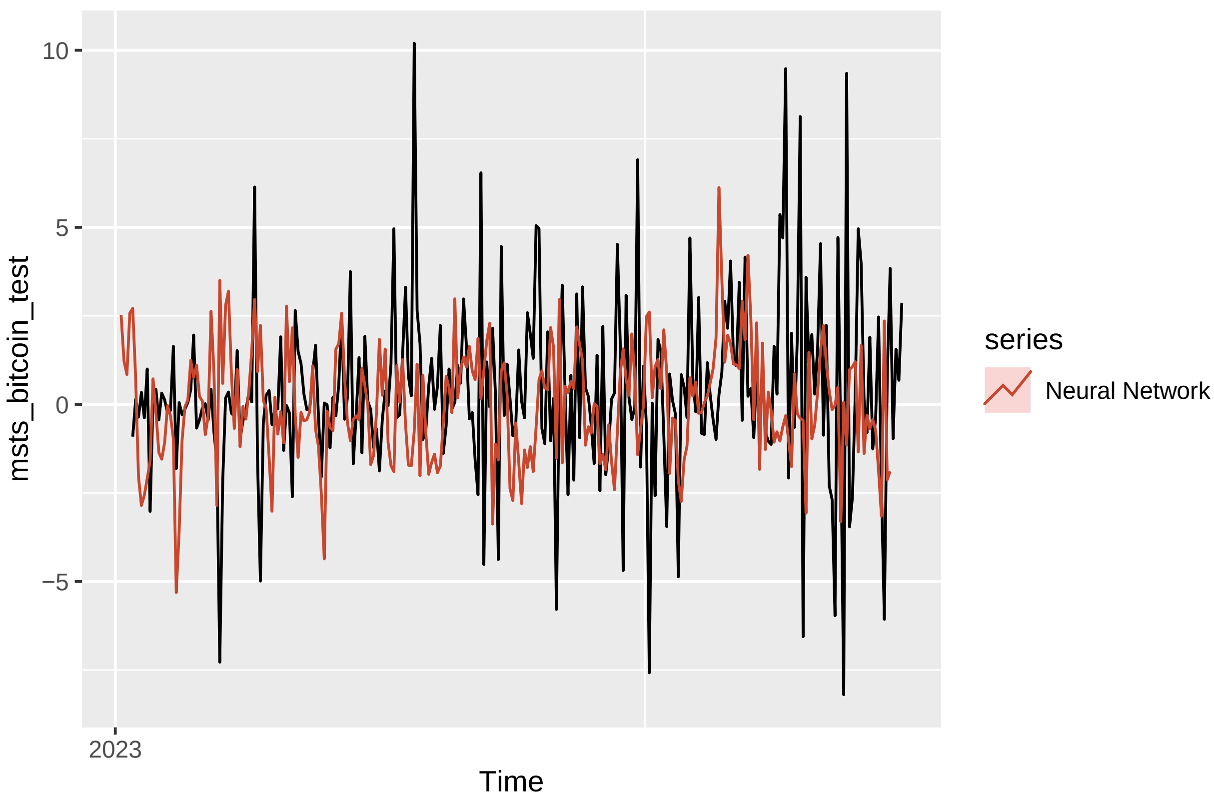

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0)(0,1,0)[365] with drift
## Q* = 995.59, df = 436, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 438
```

3.4 Neural network with fourier

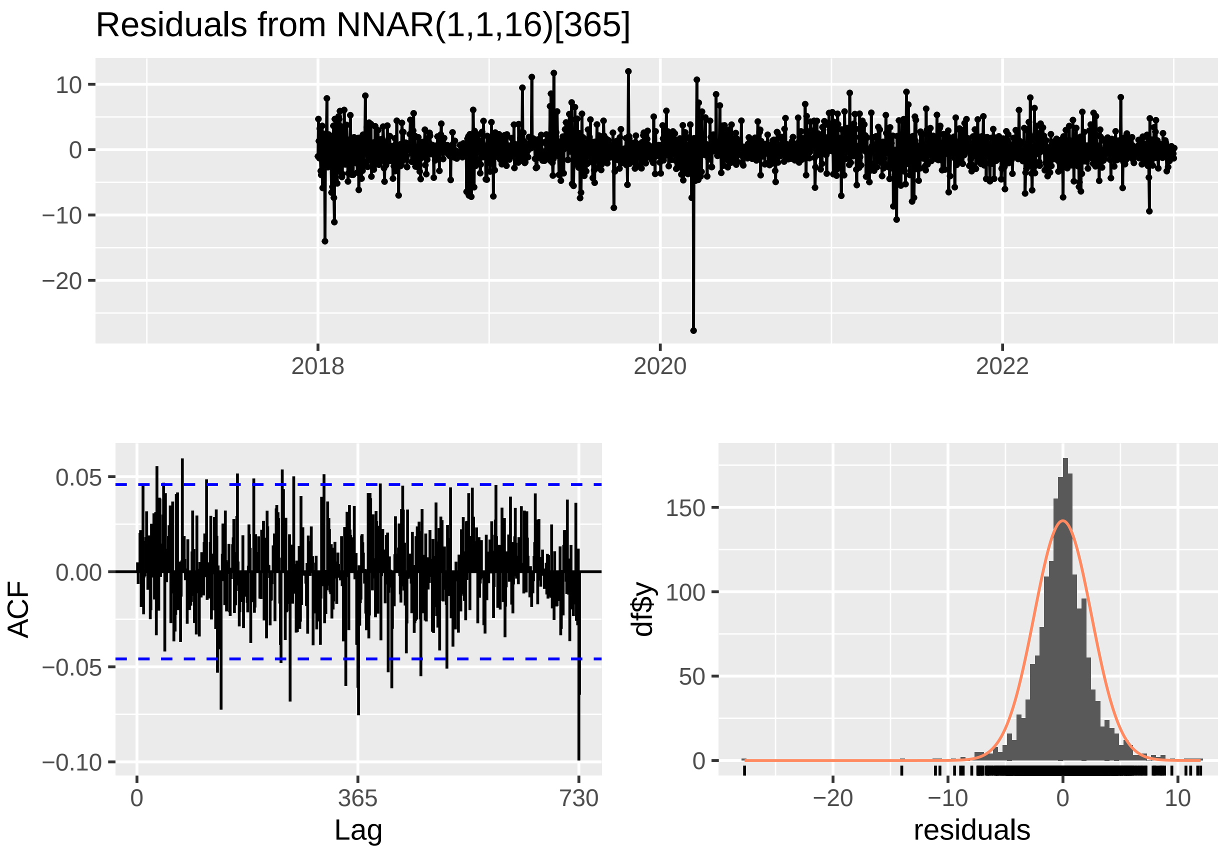
```r
#Neural network with fourier (3,12), p=1,P=1
NN_fit1 <- nnetar(msts_bitcoin,p=1,P=1,xreg=fourier(msts_bitcoin, K=c(3,12)))
NN_for1 <- forecast(NN_fit1,h=266, xreg=fourier(msts_bitcoin, K=c(3,12),h=266))

plot(NN_for1)
```




```r
autoplot(msts_bitcoin_test) +
  autolayer(NN_for1, series="Neural Network",PI=FALSE)
```



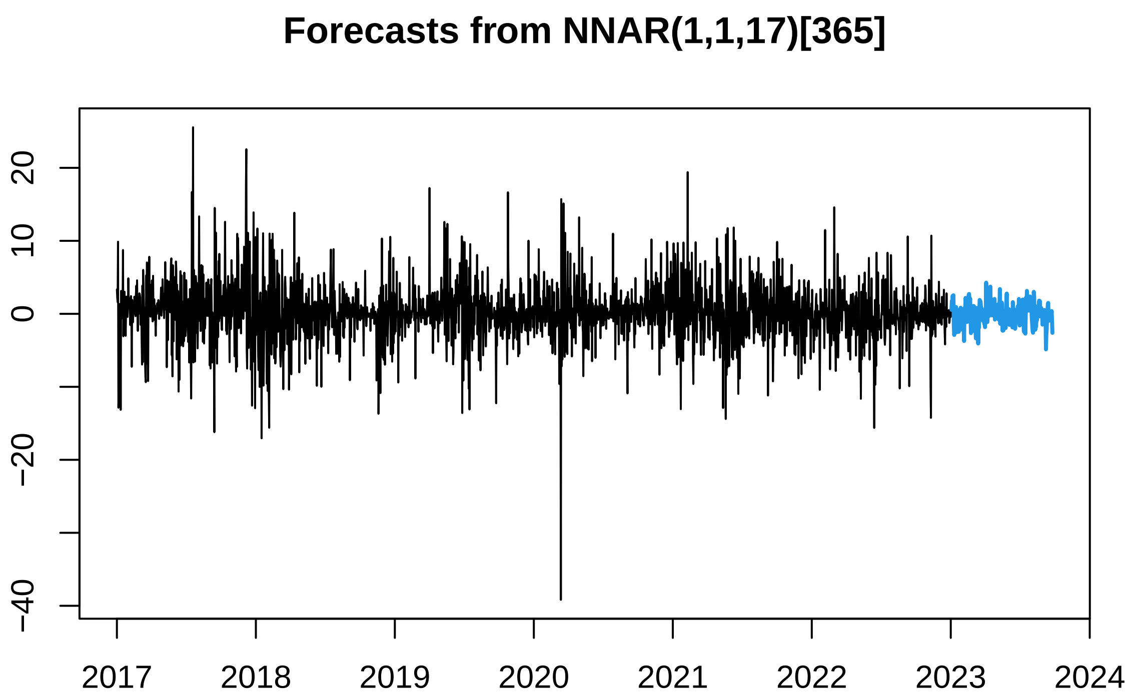

```r
NN_scores1 <- accuracy(NN_for1$mean,msts_bitcoin_test)
#print(NN_scores1)
checkresiduals(NN_fit1)
```



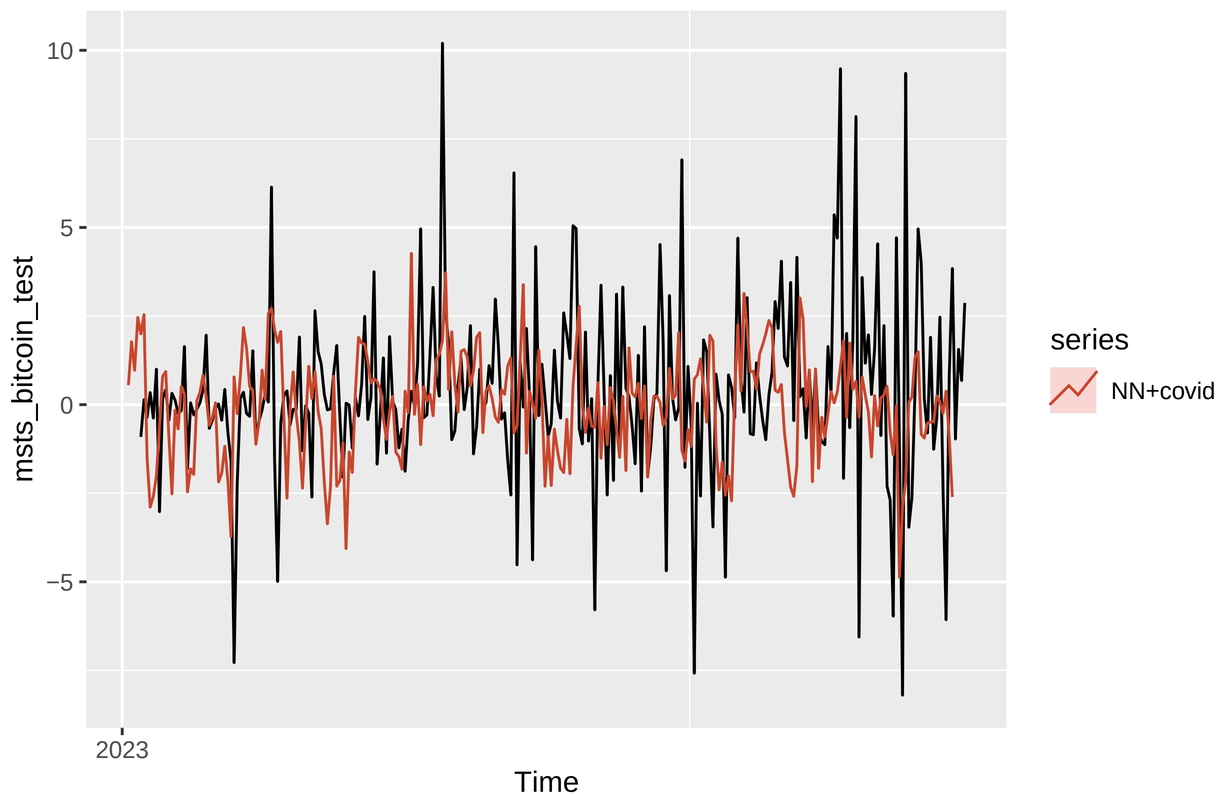

```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,16)[365]
## Q* = 450.76, df = 438, p-value = 0.3266
##
## Model df: 0.   Total lags used: 438
```

4. Models fit with external variables as regressor

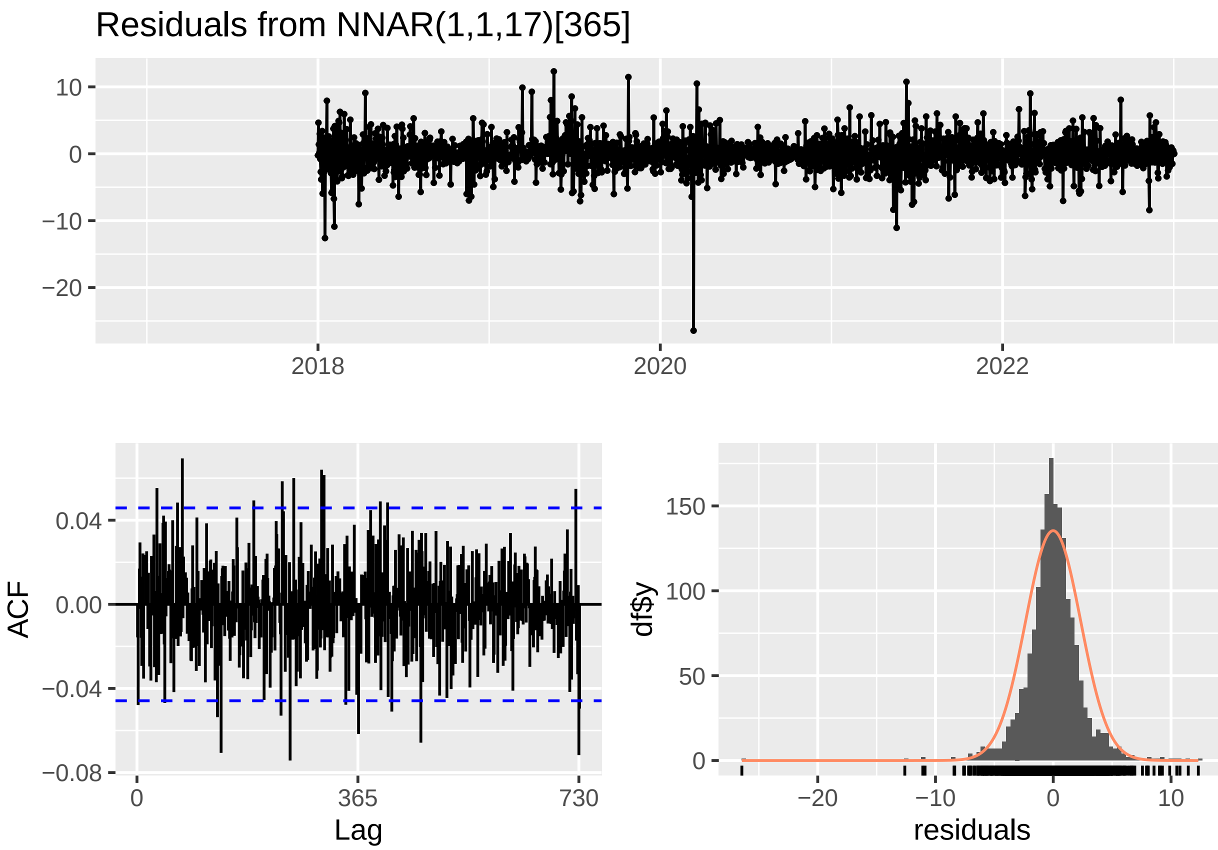
```
# create the covid regressor
# Create a test series with 266 rows, each initialized to 0
msts_covid <- msts(train_bitcoin$covid,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2017,01,01))
covid_regressors <- as.matrix(data.frame(fourier(msts_bitcoin, K=c(3,12),h=nrow(train_bitcoin)), "covid
future_covid_regressors <- as.matrix(data.frame(fourier(msts_bitcoin, K=c(3,12),h=nrow(test_bitcoin)),


# NN+covid
NN_fit3 <- nnetar(msts_bitcoin,p=1,P=1,xreg=covid_regressors)
NN_for3 <- forecast(NN_fit3,h=266, xreg=future_covid_regressors)
plot(NN_for3)
```




```r
autoplot(msts_bitcoin_test) +
  autolayer(NN_for3, series="NN+covid",PI=FALSE)
```




```r
NN_scores3 <- accuracy(NN_for3$mean,msts_bitcoin_test)
#print(NN_scores3)
checkresiduals(NN_fit3)
```

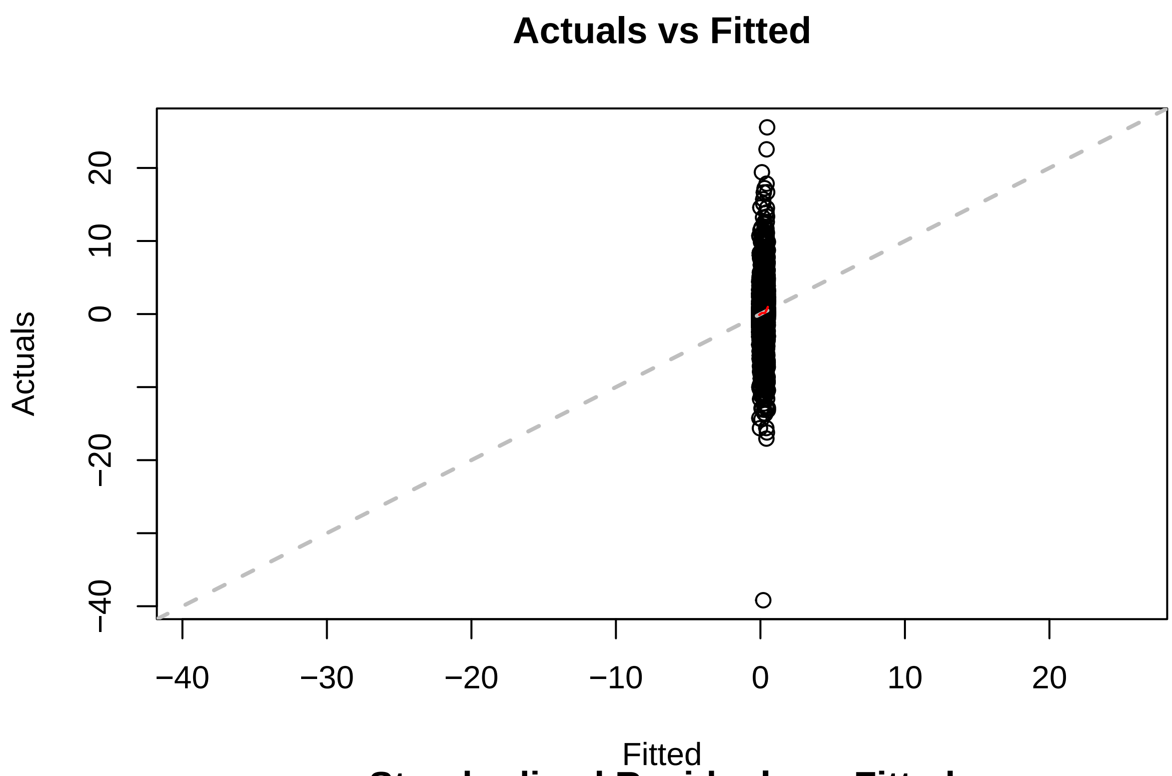


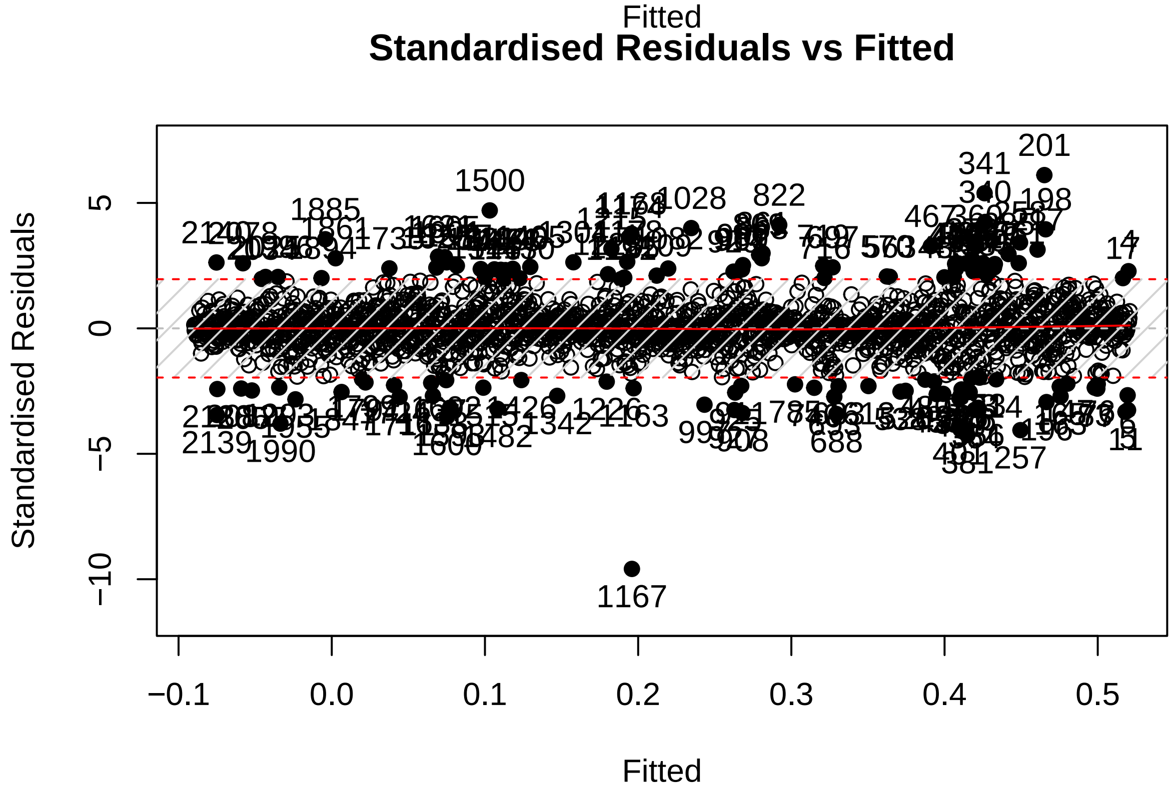

```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,17)[365]
## Q* = 438, df = 438, p-value = 0.491
##
## Model df: 0.   Total lags used: 438
```

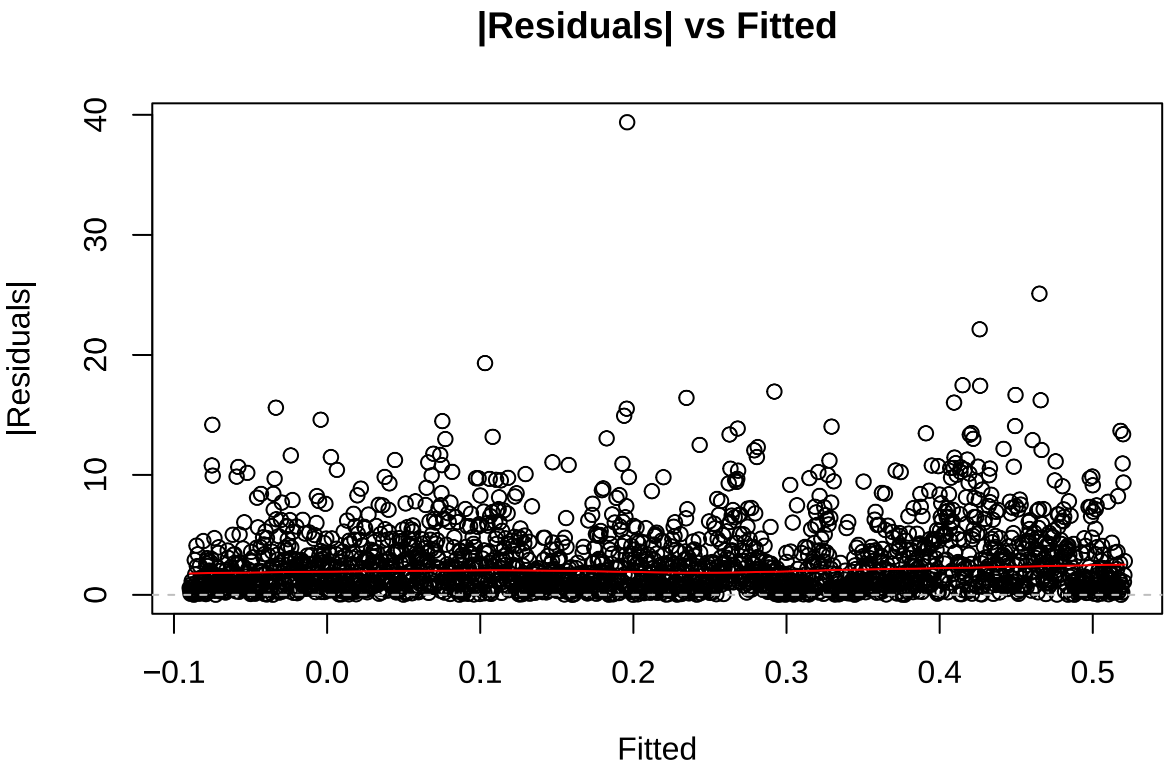
```
#SS Exponential smoothing
SSES_fit1 <- es(msts_bitcoin,model="ZZZ",h=266,holdout=FALSE)
```

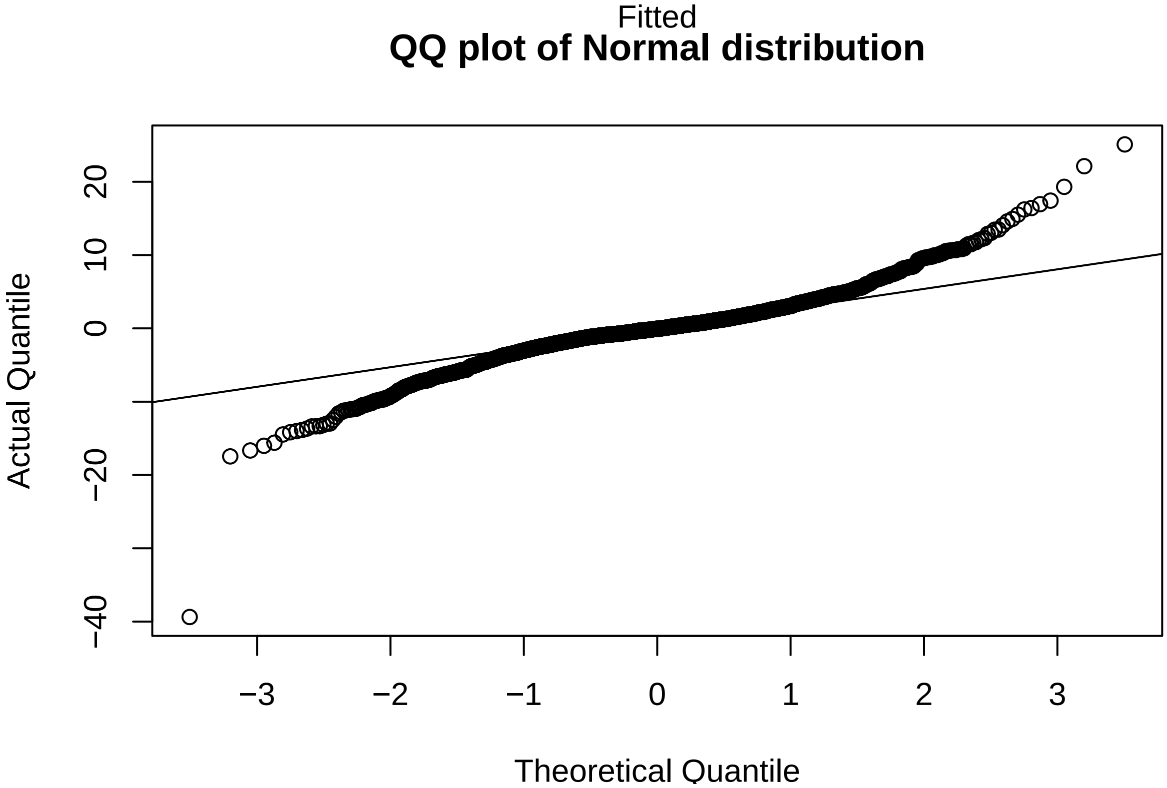
```
## Warning: Only additive models are allowed for your data. Changing the selection
## mechanism.
```

```
plot(SSES_fit1)
```


Actuals vs Fitted
Actuals
Fitted
20
10
0
−20
−40
−40
−30
−20
−10
0
10
20


Standardised Residuals vs Fitted
Standardised Residuals
Fitted
5
0
−5
−10
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
1500
201
341
822
1028
1885
1167
2139
1990
688
257
581

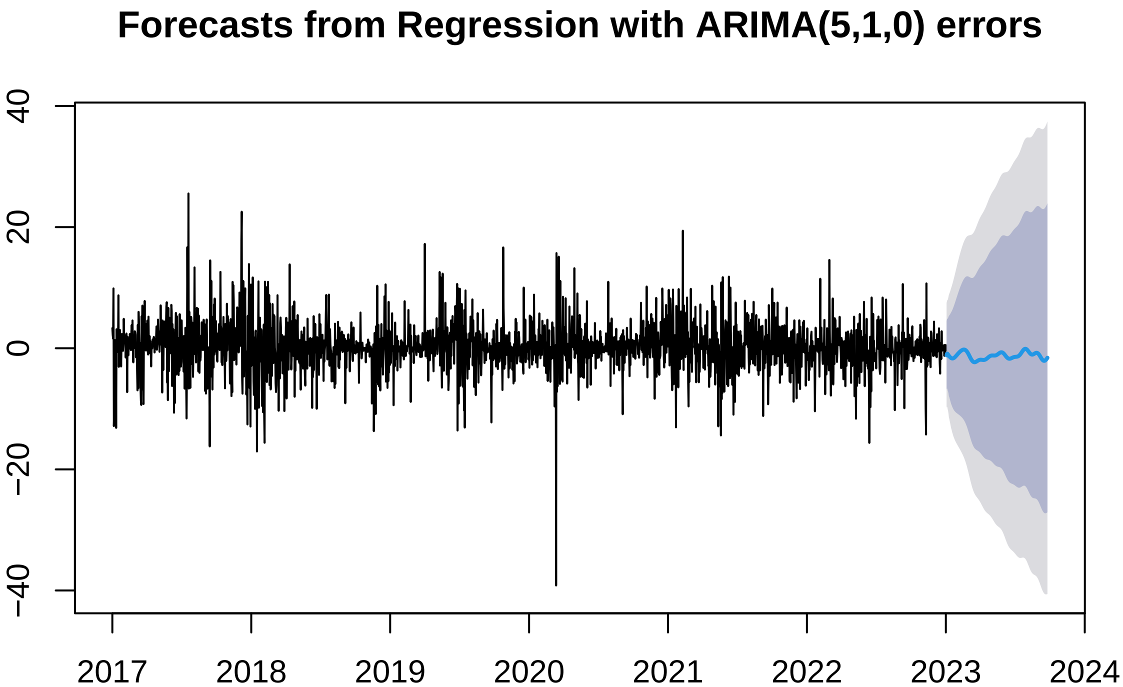




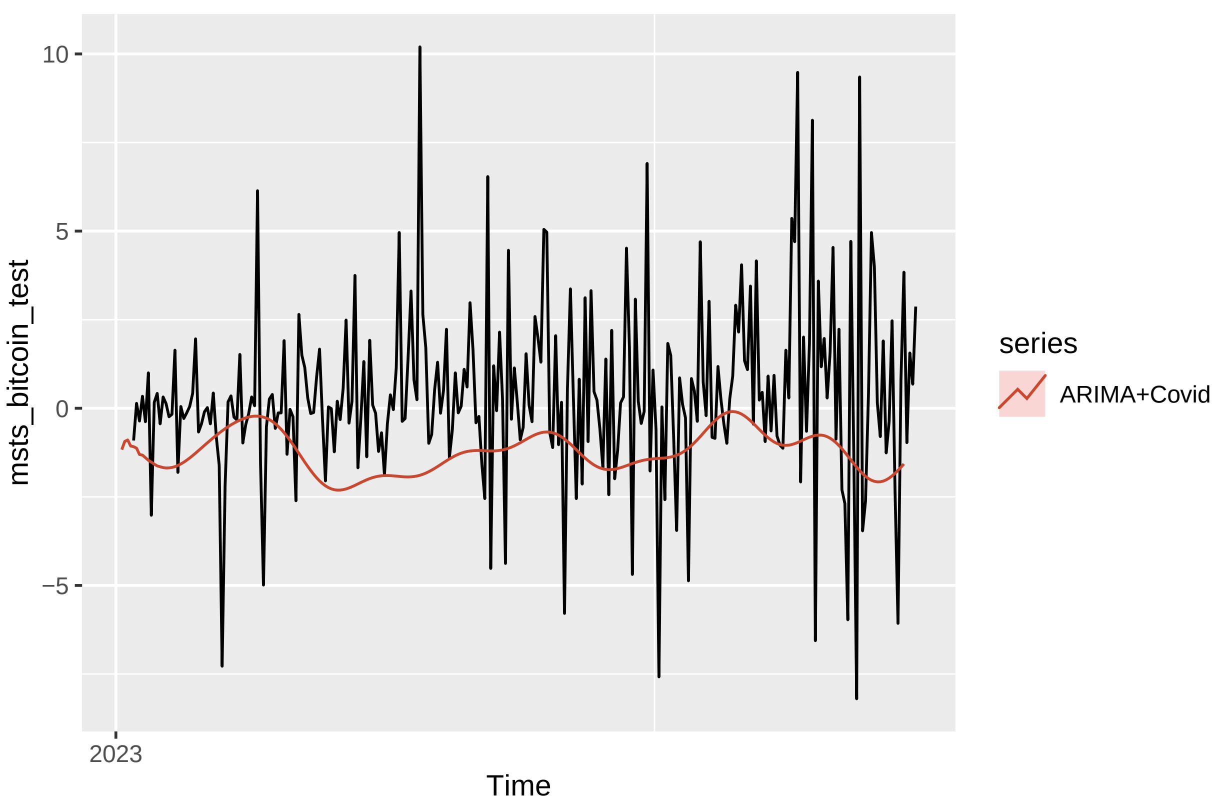

```
SSES_scores1 <- accuracy(SSES_fit1$forecast,msts_bitcoin_test)
#print(SSES_scores1)
```

```
#Arima+covid as regressor

ARIMA_fit1<-auto.arima(msts_bitcoin,seasonal= FALSE, lambda=1,xreg=covid_regressors)
```

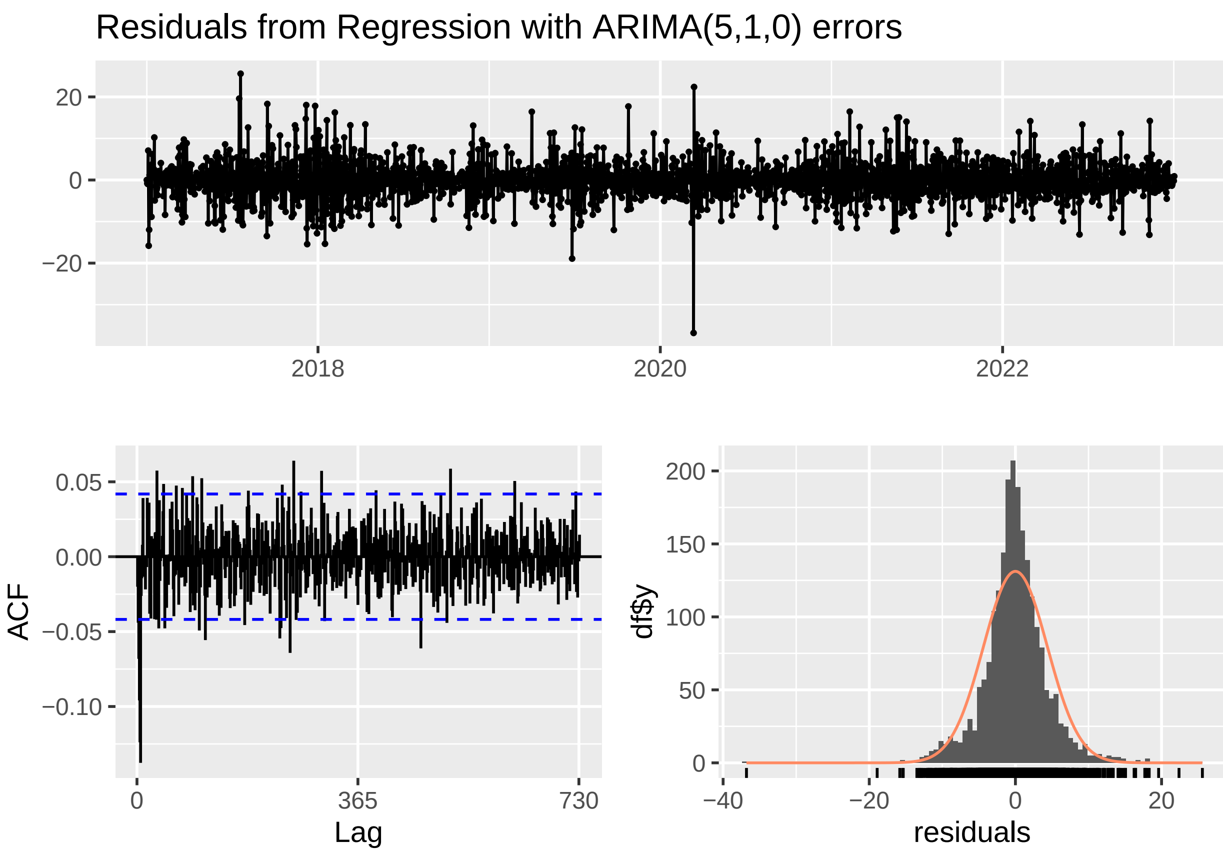
```r
ARIMA_for1<-forecast(ARIMA_fit1,xreg=future_covid_regressors,h=266)

plot(ARIMA_for1)
```




```r
autoplot(msts_bitcoin_test) +
  autolayer(ARIMA_for1, series="ARIMA+Covid",PI=FALSE)
```



```
ARIMA_scores1 <- accuracy(ARIMA_for1$mean,msts_bitcoin_test)
#print(ARIMA_scores1)
checkresiduals(ARIMA_fit1)
```



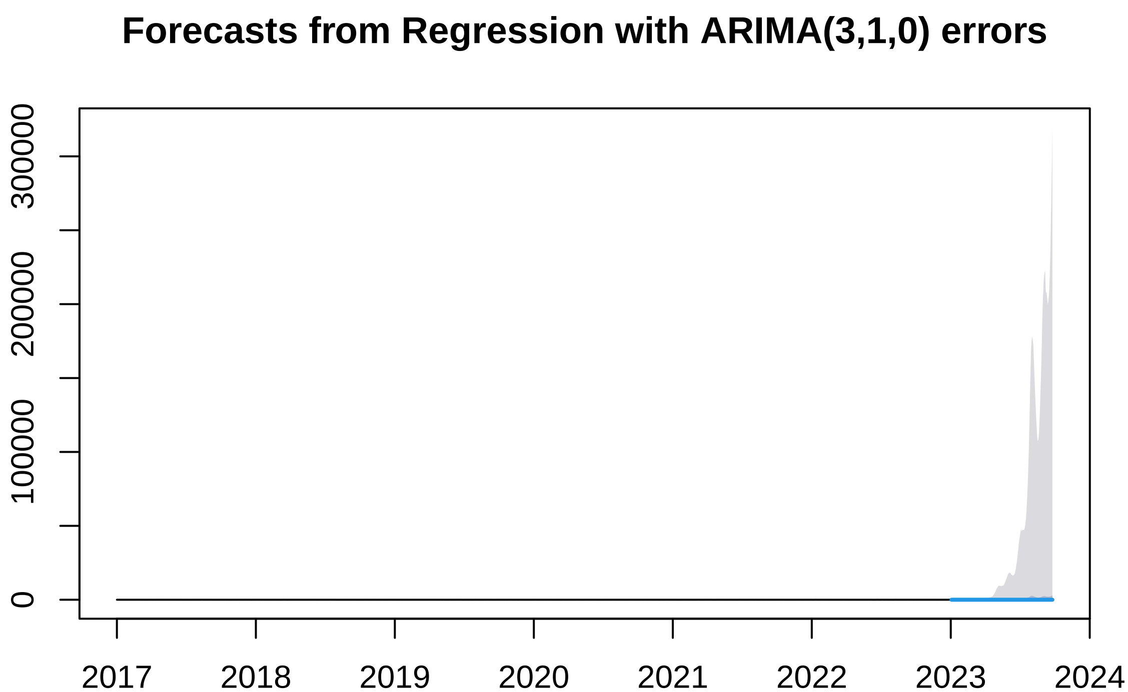

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,0) errors
## Q* = 597.87, df = 433, p-value = 2.343e-07
##
## Model df: 5.   Total lags used: 438
```

```
#Crude oil regressors
oil_regressors<- as.matrix(data.frame(fourier(msts_bitcoin,K=c(3,12)),"oil"=msts_oil))
oil_fc<-forecast(msts_oil, h=266)
oil_regressors_fc<-as.matrix(data.frame(fourier(msts_bitcoin,K=c(3,12), h=266),"oil"= oil_fc$mean))
```

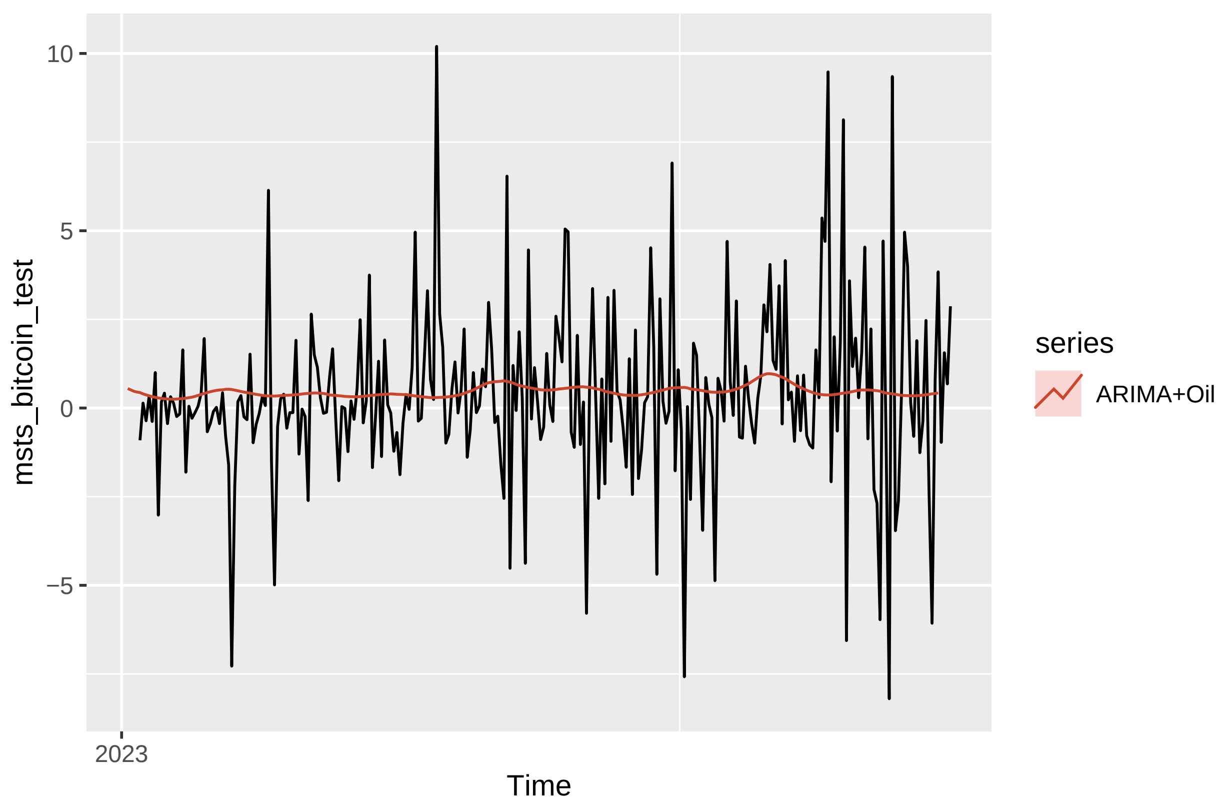
```
#Arima with regressor

arima_fit_r <- auto.arima(msts_bitcoin,seasonal= FALSE, lambda=0,xreg=oil_regressors)
```

```
## Warning in log(x): NaNs produced

## Warning in log(x): NaNs produced
```

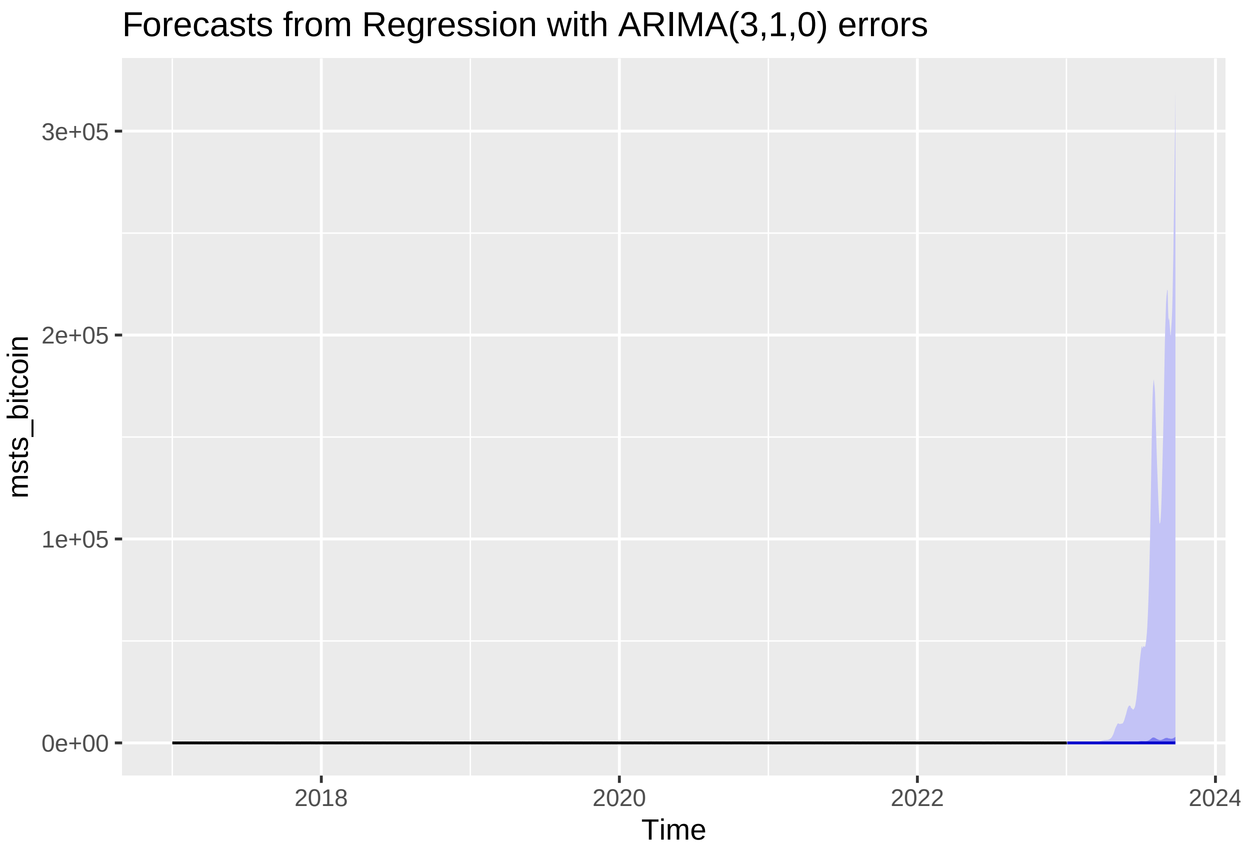
```
arima_forecast_r <- forecast(arima_fit_r, xreg=oil_regressors_fc,h=266)

plot(arima_forecast_r)
```



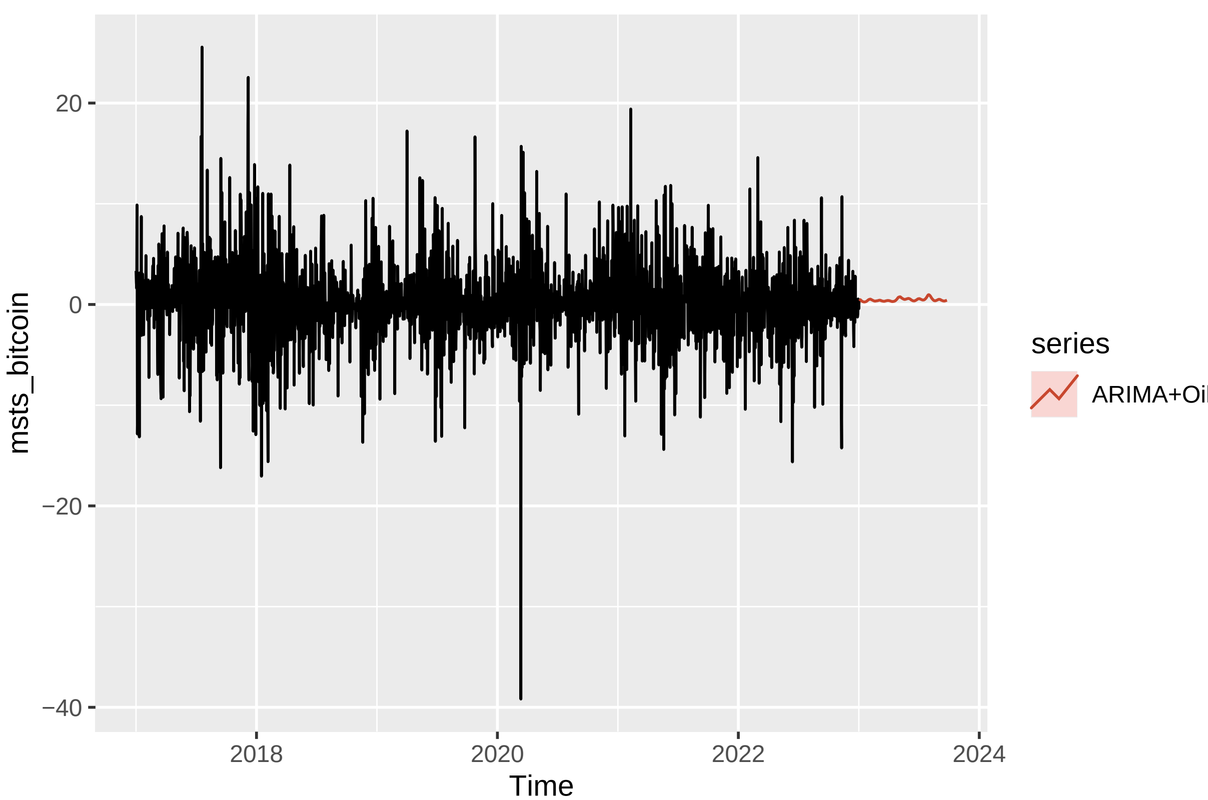

```r
autoplot(msts_bitcoin_test) + autolayer(arima_forecast_r, series="ARIMA+Oil", PI=FALSE)
```



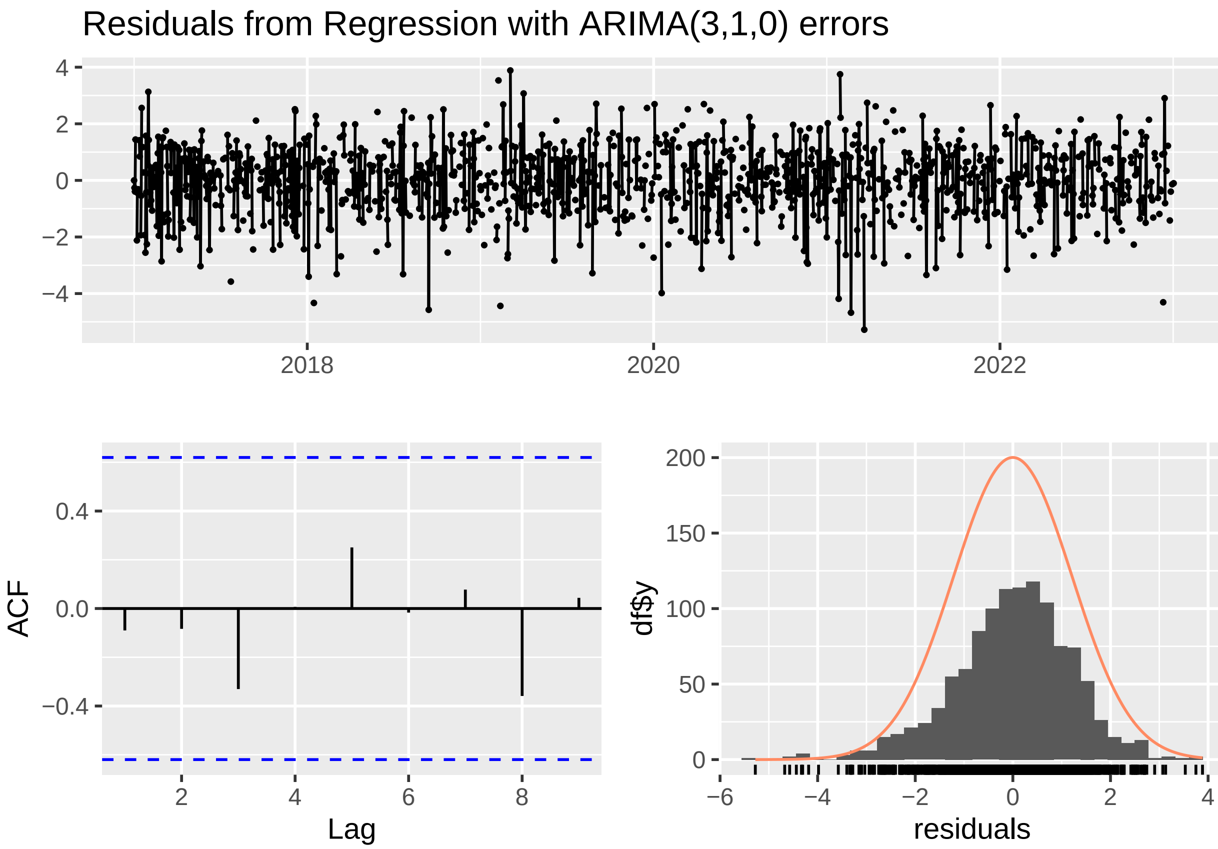

```r
autoplot(arima_forecast_r)
```




```r
autoplot(msts_bitcoin) +
  autolayer(arima_forecast_r, series="ARIMA+Oil",PI=FALSE)
```

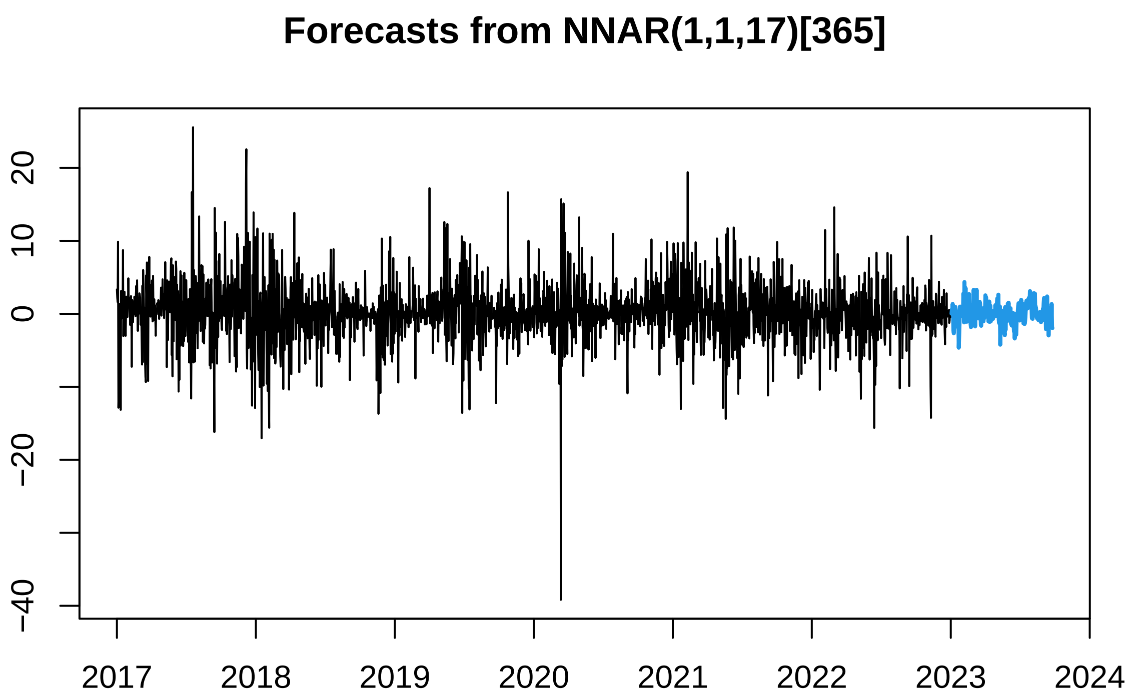


```
ARIMA_scores_oil <- accuracy(arima_forecast_r$mean,msts_bitcoin_test)
#print(ARIMA_scores_oil)

# Use checkresiduals to plot and assess residuals
checkresiduals(arima_forecast_r)
```

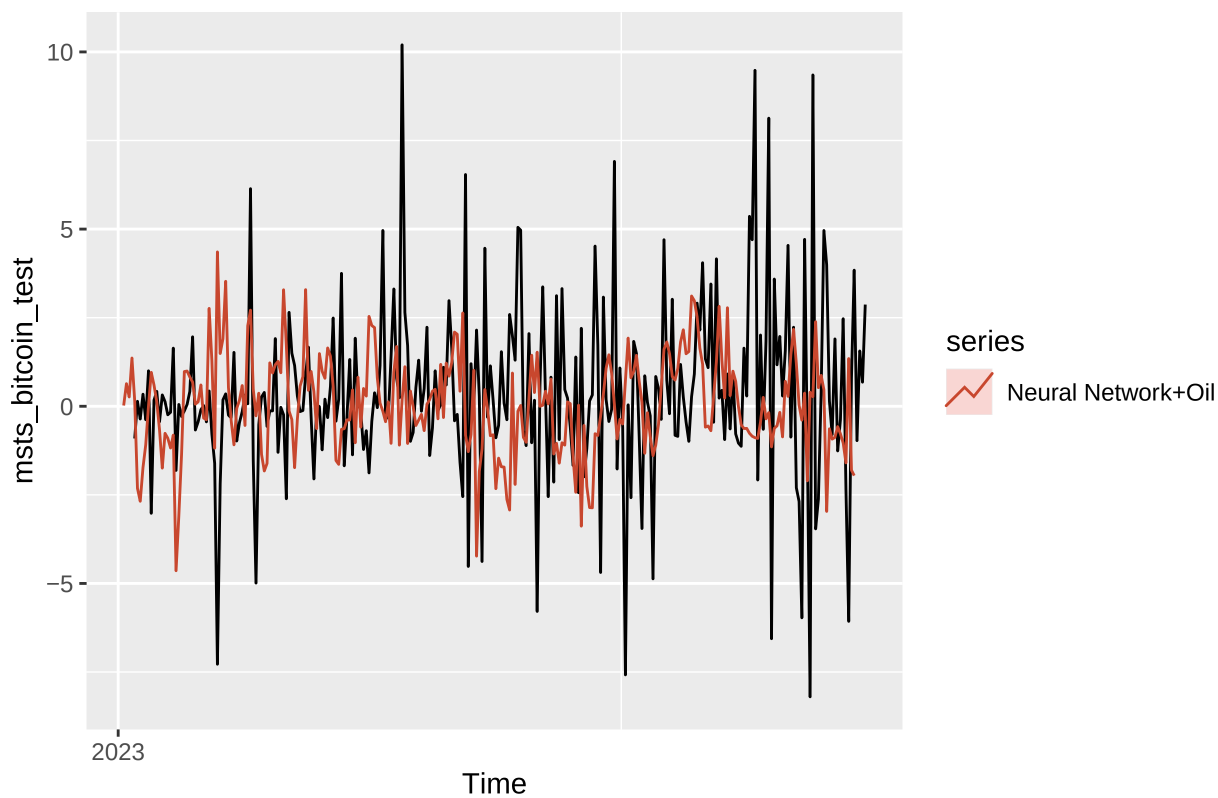



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,1,0) errors
## Q* = 1507.3, df = 435, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 438
```

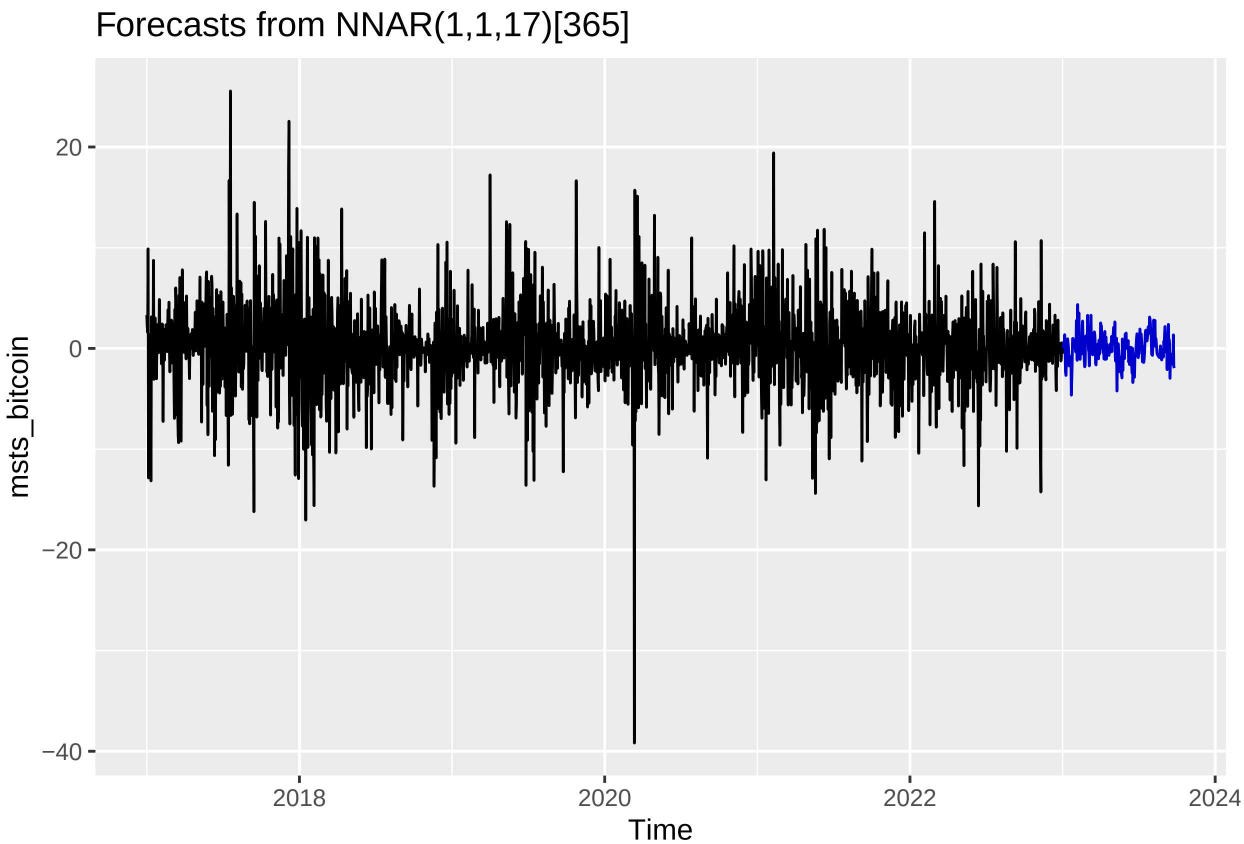
```
NN_fit_o <- nnetar(msts_bitcoin,p=1,P=1,xreg=oil_regressors)
NN_fc_o <- forecast(NN_fit_o,h=266, xreg=oil_regressors_fc)

plot(NN_fc_o)
```



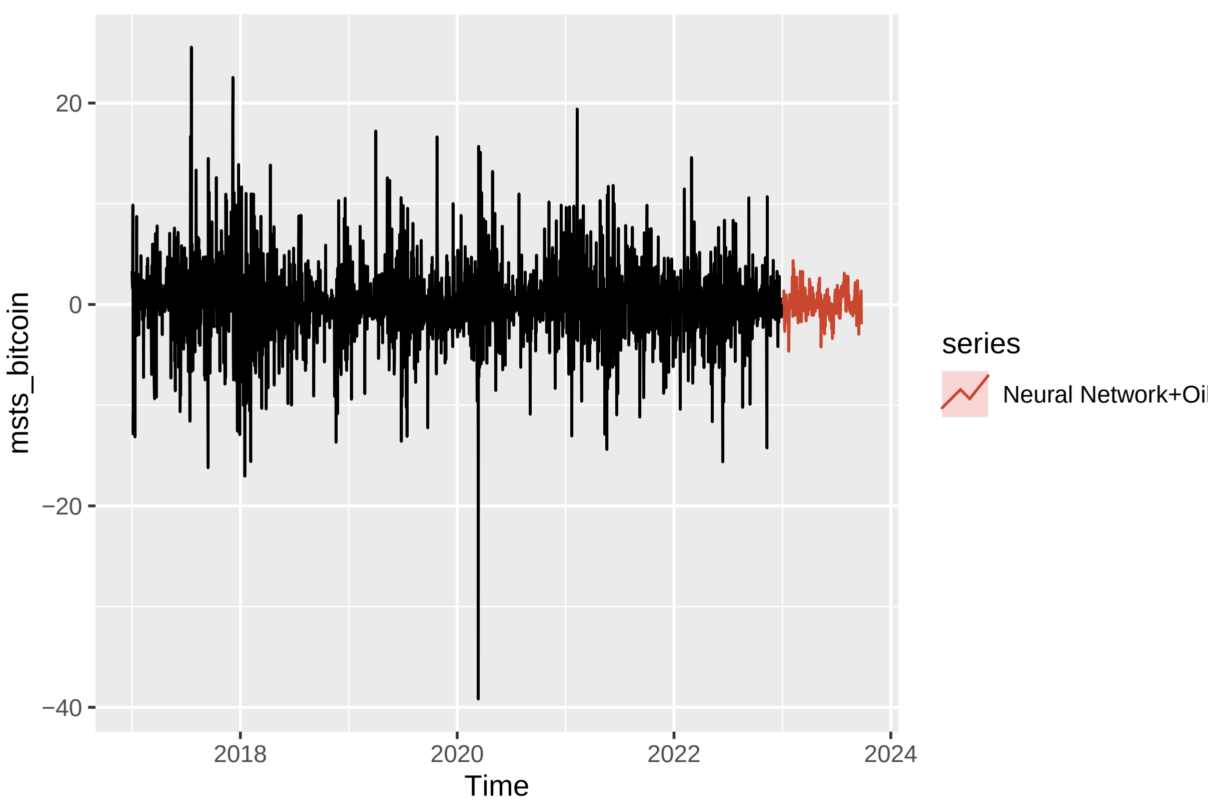

```r
autoplot(msts_bitcoin_test) + autolayer(NN_fc_o, series="Neural Network+Oil", PI=FALSE)
```



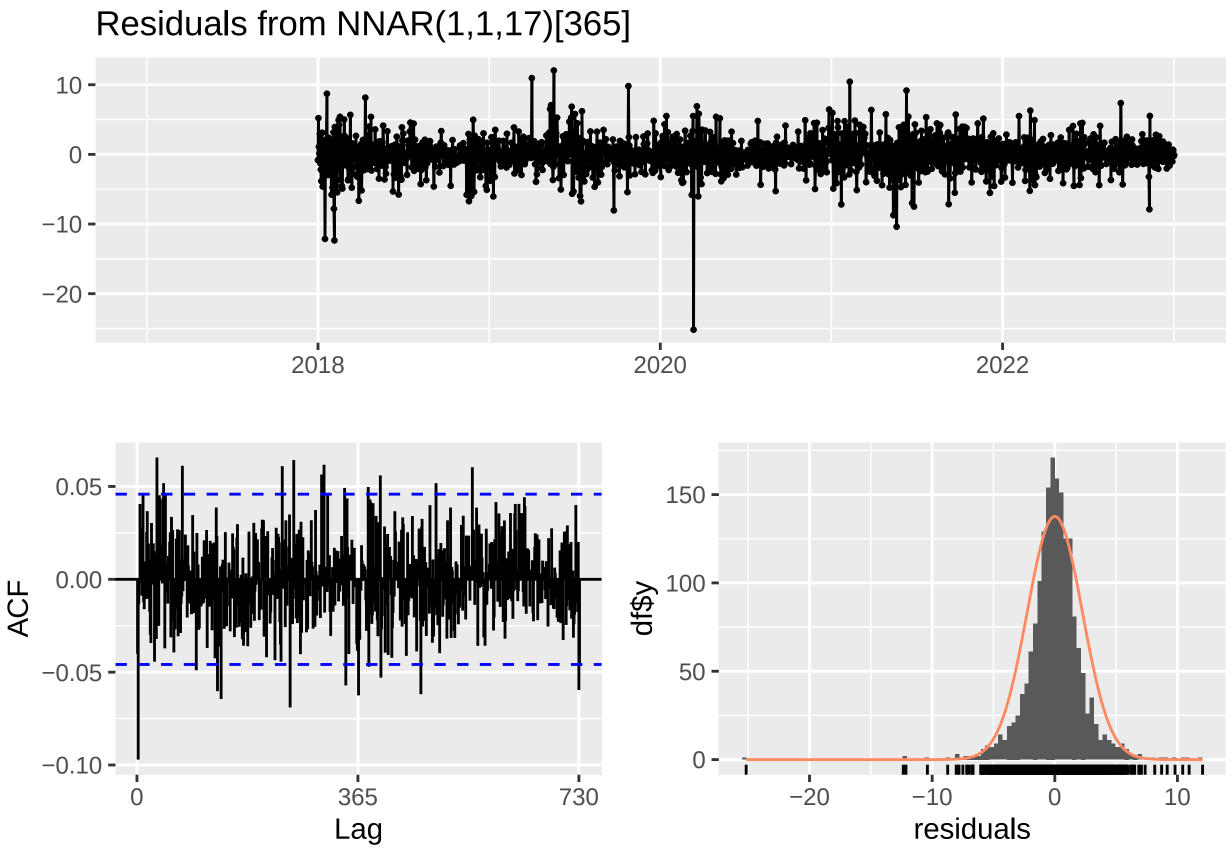

```r
autoplot(NN_fc_o)
```




```r
autoplot(msts_bitcoin) +
  autolayer(NN_fc_o, series="Neural Network+Oil",PI=FALSE)
```



```r
NN_scores_o <- accuracy(NN_fc_o$mean,msts_bitcoin_test)
#print(NN_scores_o)

# Use checkresiduals to plot and assess residuals
checkresiduals(NN_fc_o)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,17)[365]
## Q* = 490.47, df = 438, p-value = 0.04192
##
## Model df: 0.   Total lags used: 438
```

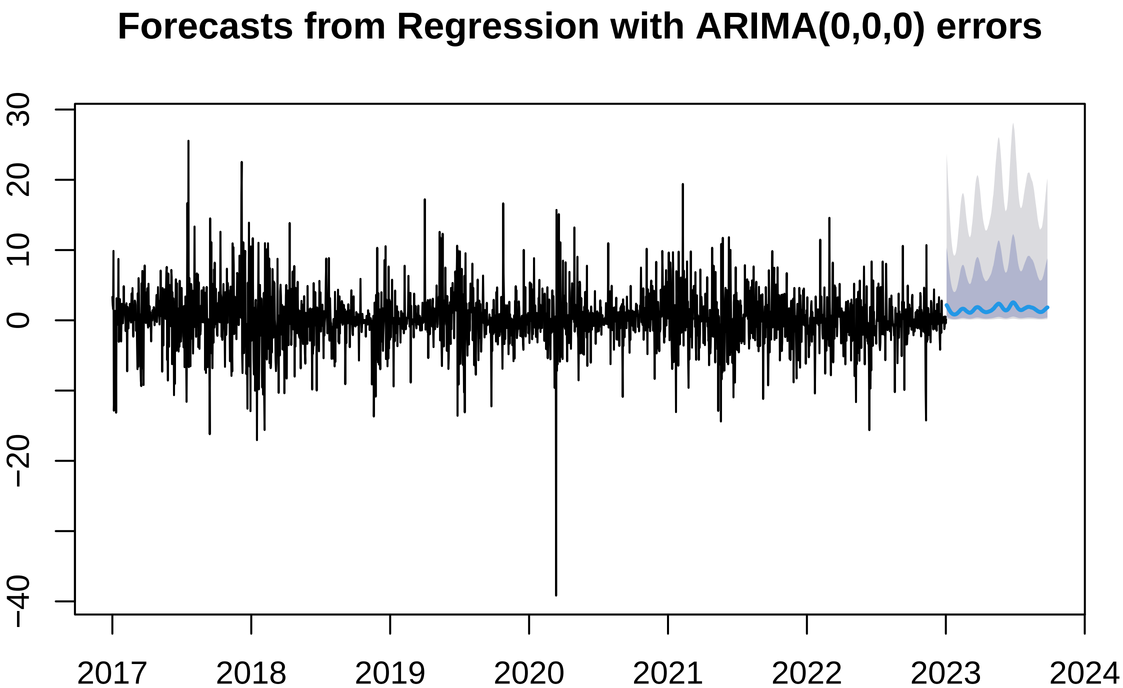
```r
# Combine two regressors
covid_oil_regressor <- as.matrix(data.frame(fourier(msts_bitcoin, K=c(3,12), h=nrow(train_bitcoin)), "c
covid_oil_regressor_fc <- as.matrix(data.frame(fourier(msts_bitcoin, K=c(3,12),h=nrow(test_bitcoin)), "
```

```r
#Arima with two regressor

arima_fit_co <- auto.arima(msts_bitcoin,seasonal= FALSE, lambda=0,xreg=covid_oil_regressor)
```

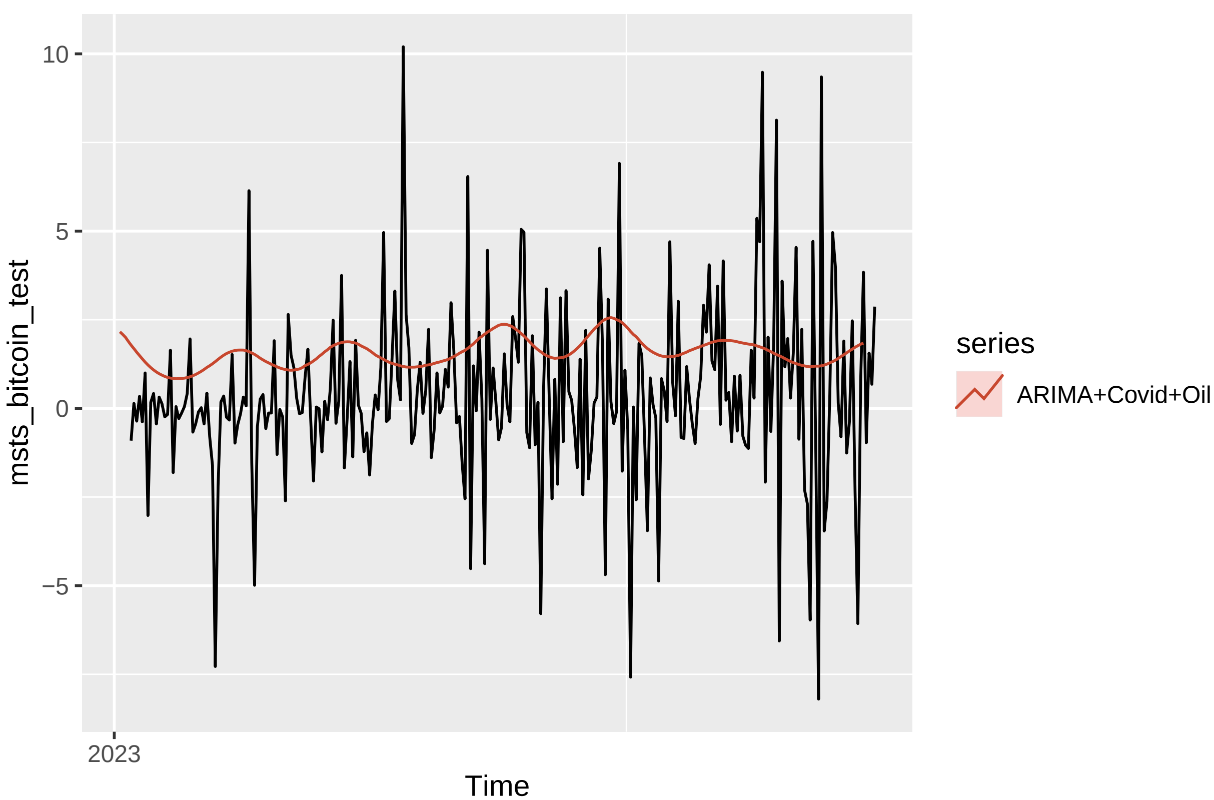
```
## Warning in log(x): NaNs produced

## Warning in log(x): NaNs produced
```

```r
arima_forecast_co <- forecast(arima_fit_co, xreg=covid_oil_regressor_fc,h=266)
```

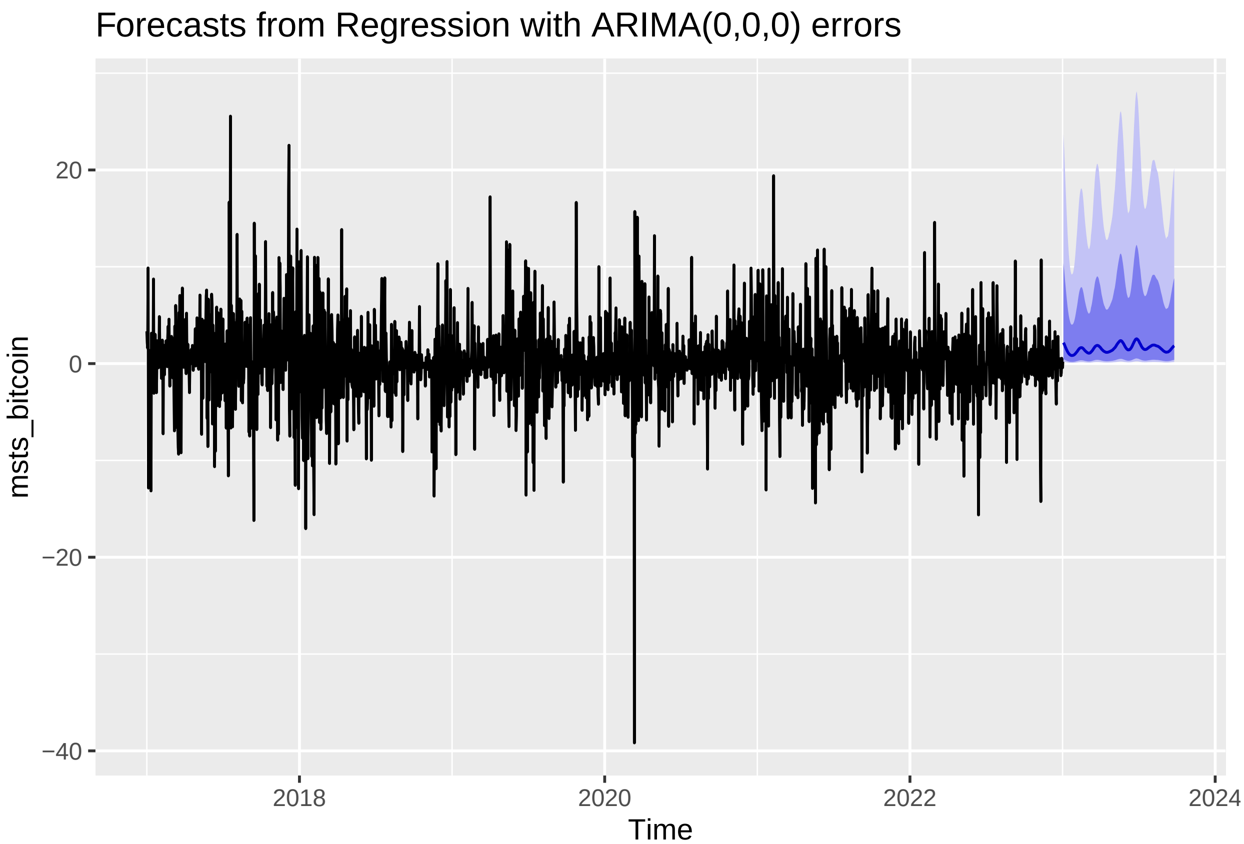
```r
plot(arima_forecast_co)
```



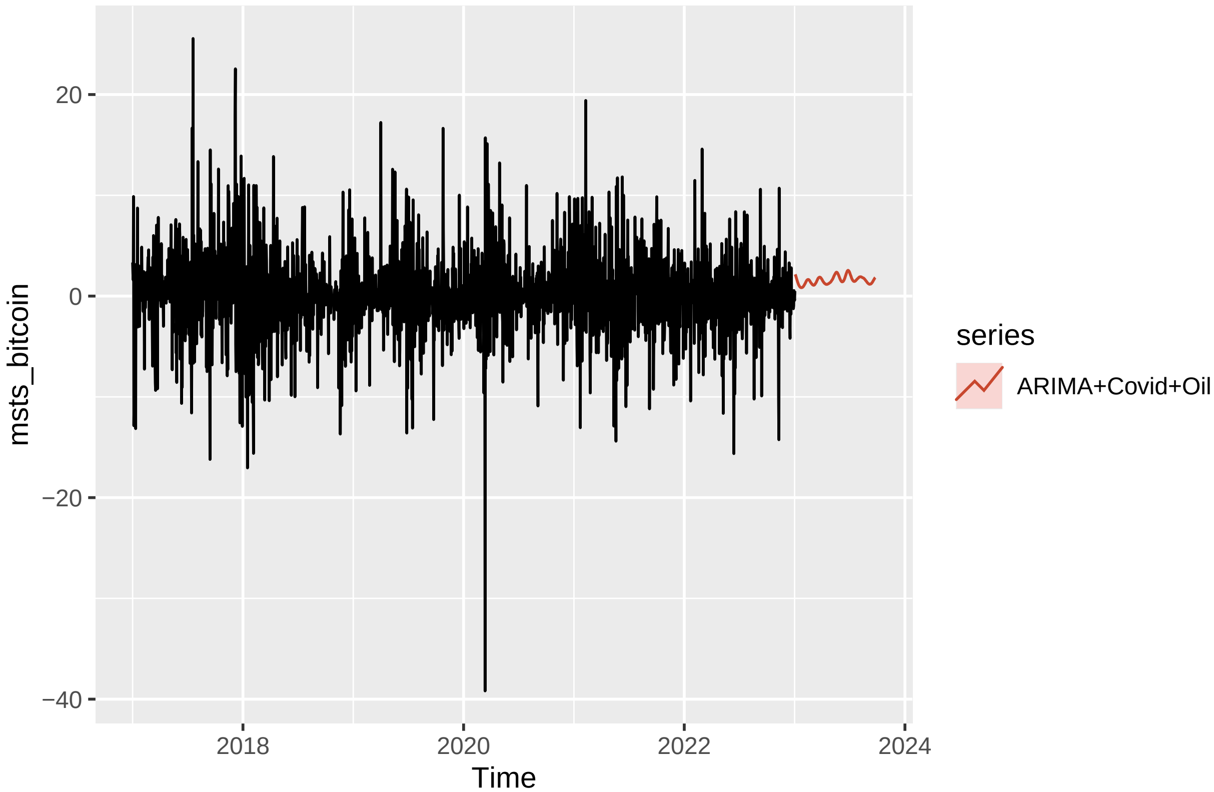

```r
autoplot(msts_bitcoin_test) + autolayer(arima_forecast_co, series="ARIMA+Covid+Oil", PI=FALSE)
```



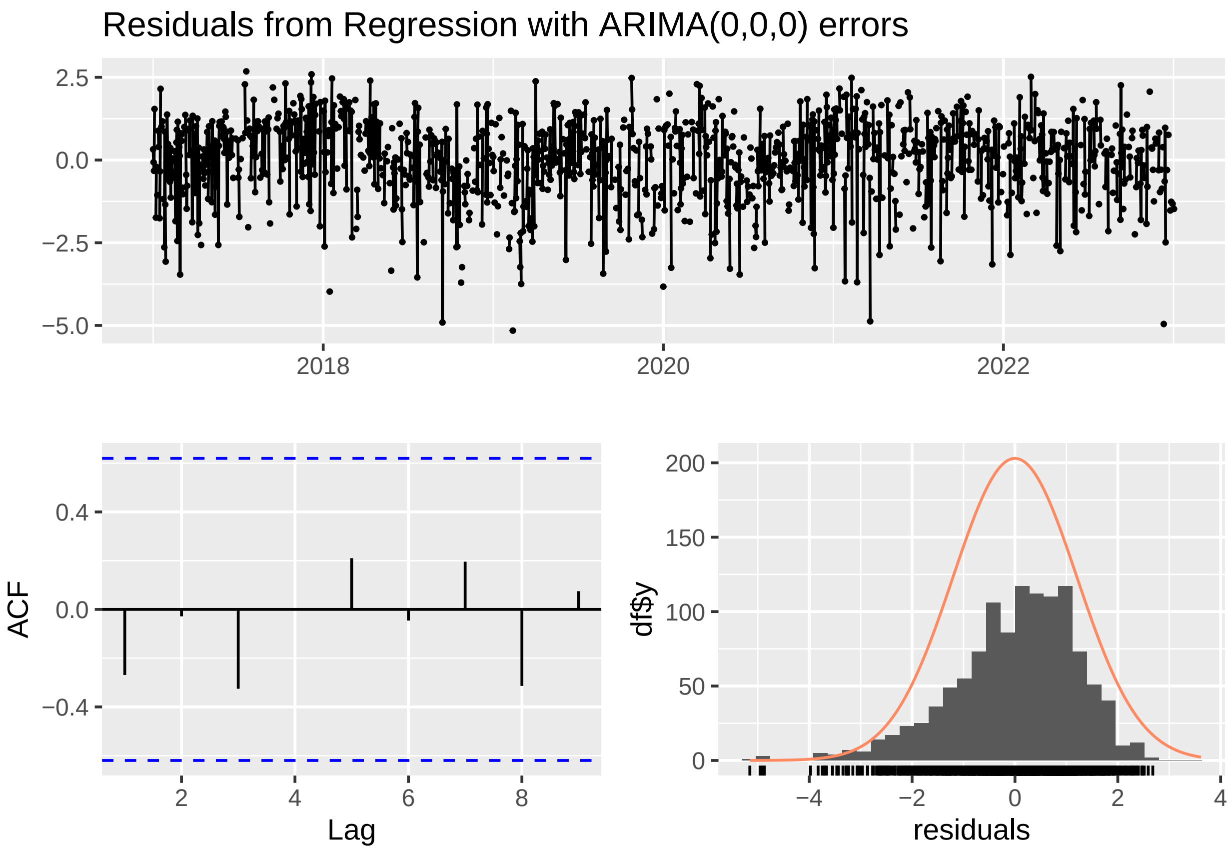
```r
autoplot(arima_forecast_co)
```




```r
autoplot(msts_bitcoin) +
  autolayer(arima_forecast_co, series="ARIMA+Covid+Oil",PI=FALSE)
```

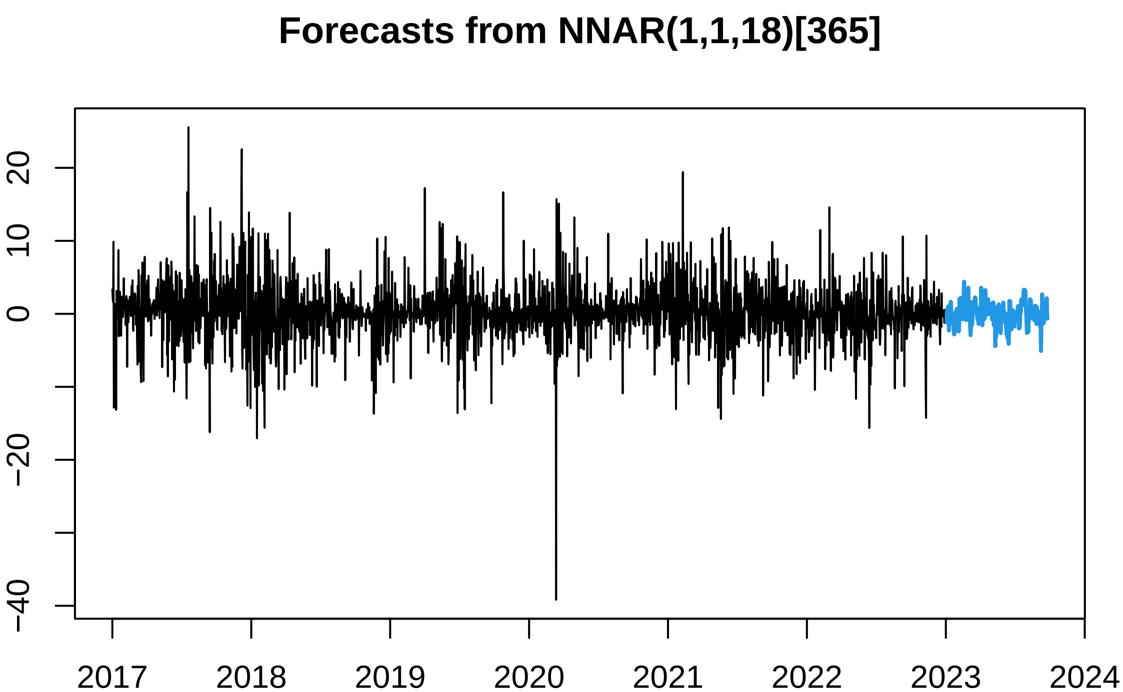



```
ARIMA_scores_co <- accuracy(arima_forecast_co$mean,msts_bitcoin_test)
#print(ARIMA_scores_co)
# Use checkresiduals to plot and assess residuals
checkresiduals(arima_forecast_co)
```

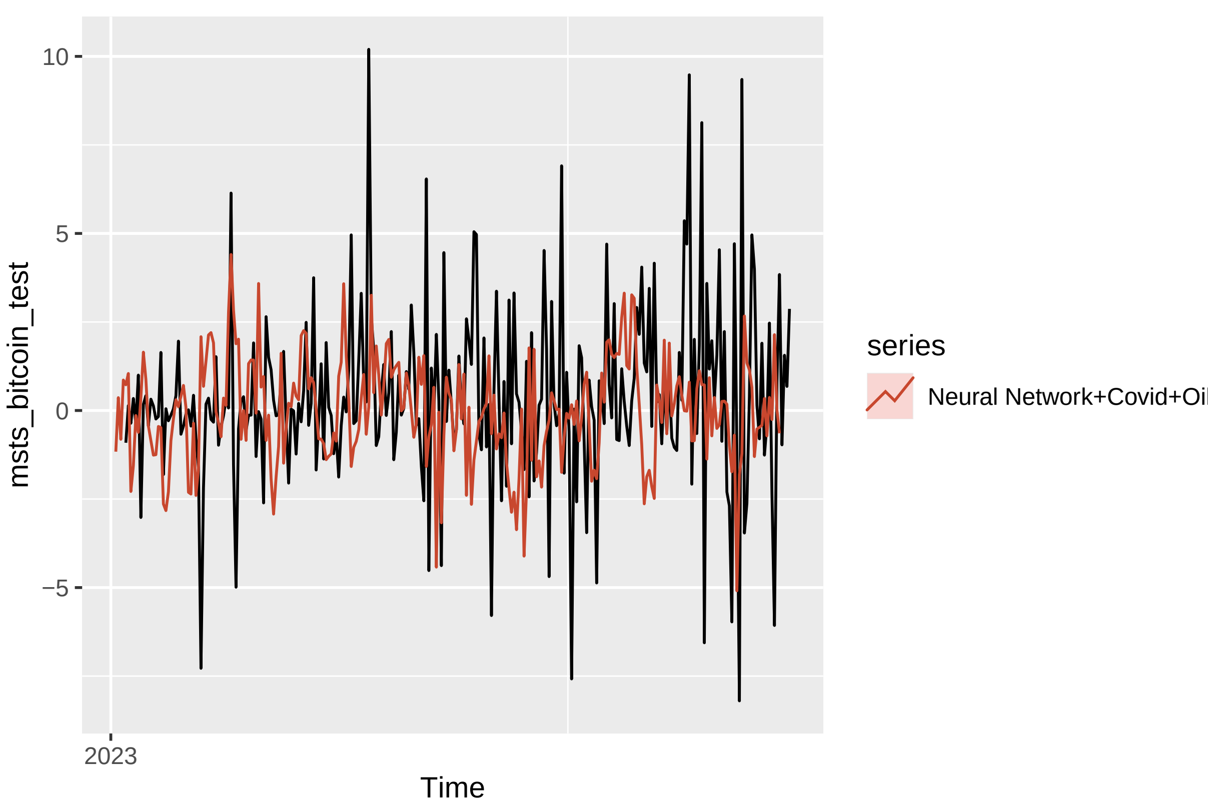



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,0) errors
## Q* = 3678.2, df = 438, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 438
```

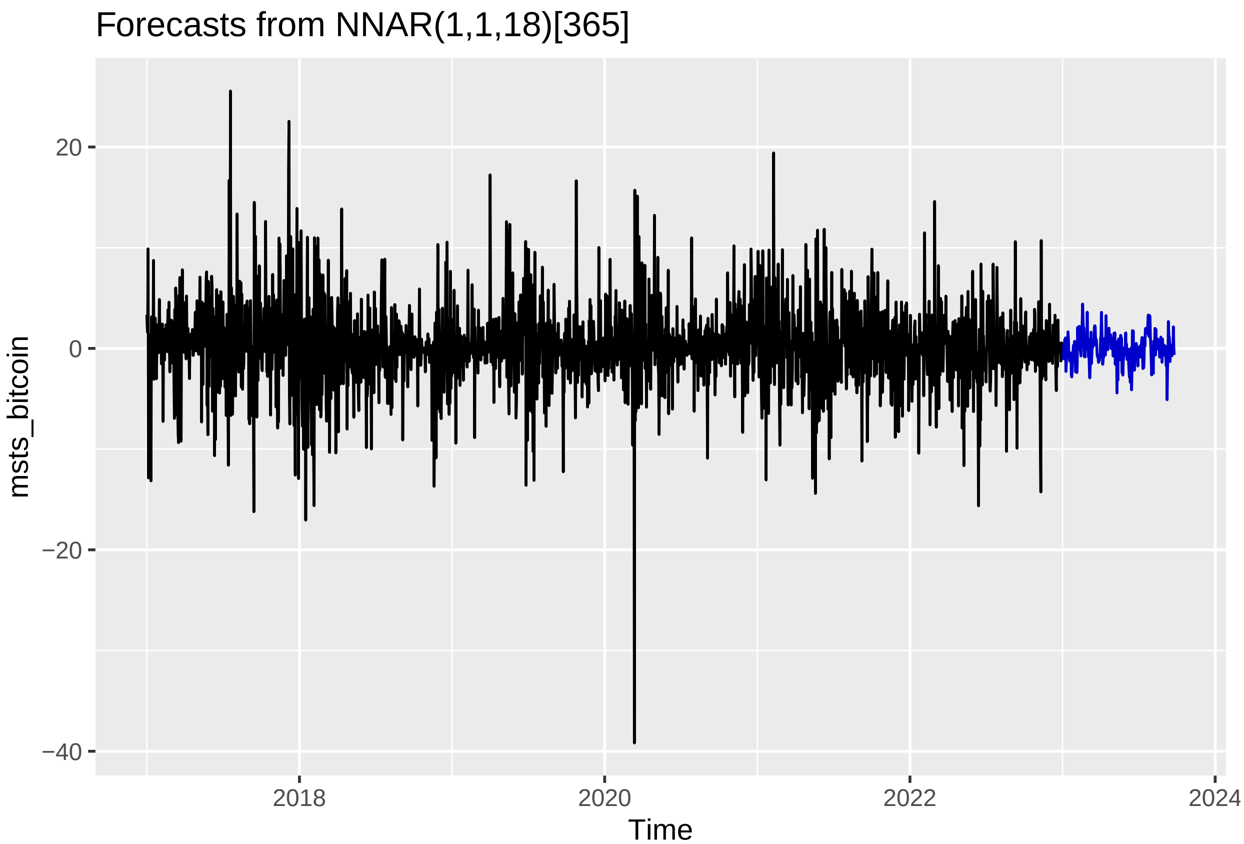
```r
# NN+covid+oil
NN_fit_co <- nnetar(msts_bitcoin,p=1,P=1,xreg=covid_oil_regressor)
NN_for_co <- forecast(NN_fit_co,h=266, xreg=covid_oil_regressor_fc)

plot(NN_for_co)
```



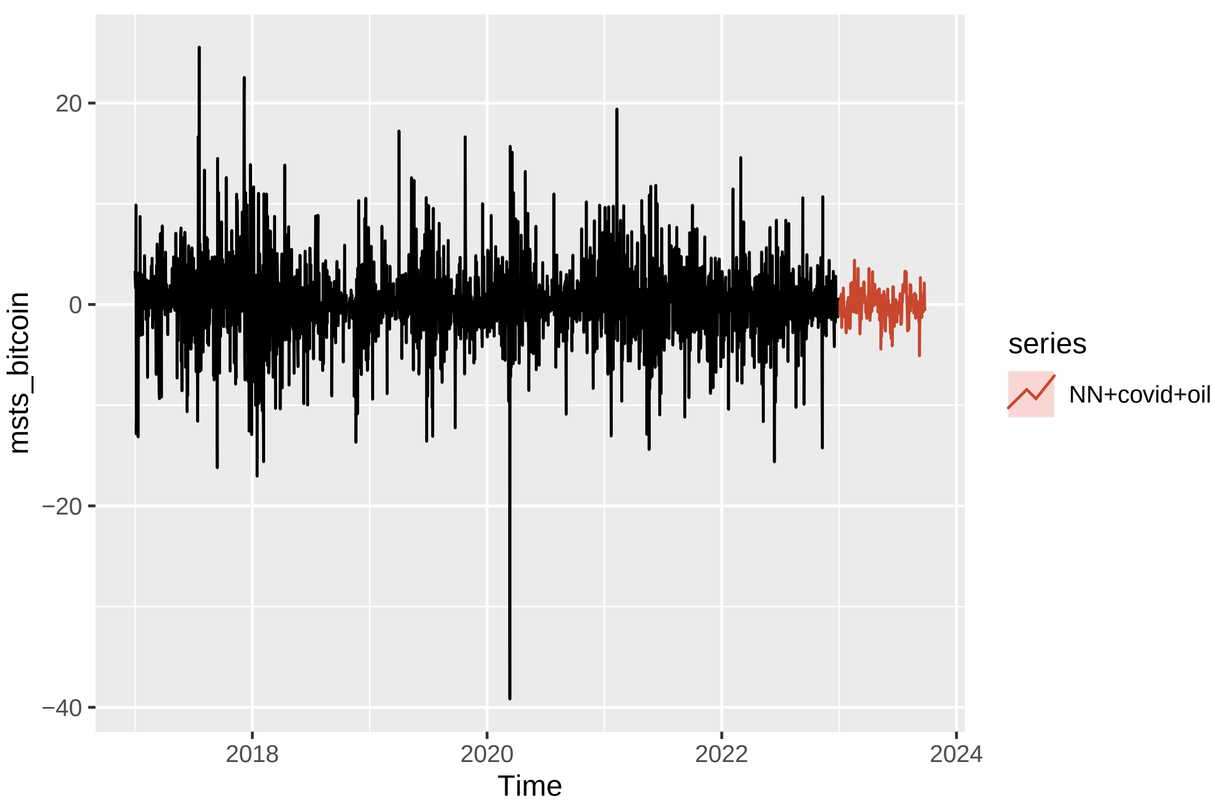

```r
autoplot(msts_bitcoin_test) + autolayer(NN_for_co, series="Neural Network+Covid+Oil", PI=FALSE)
```



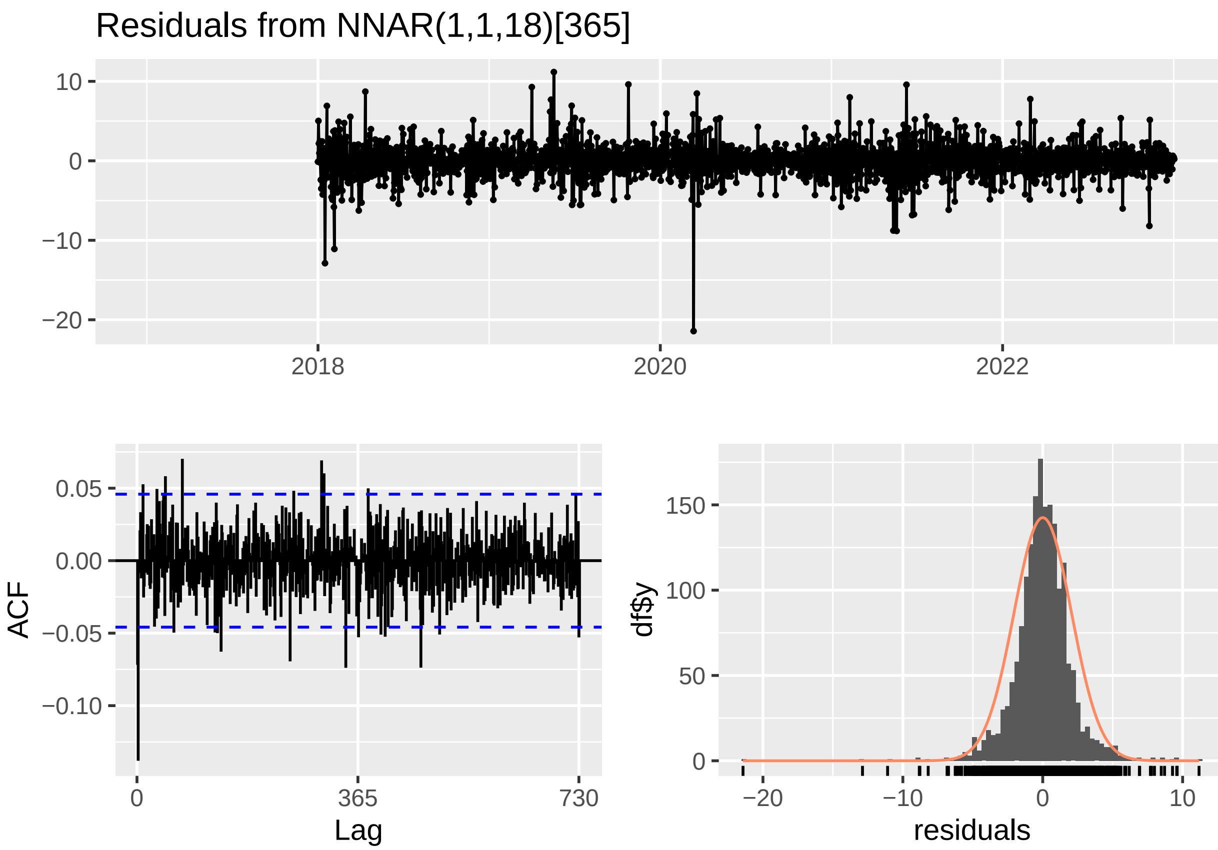

```r
autoplot(NN_for_co)
```




```r
autoplot(msts_bitcoin) +
  autolayer(NN_for_co, series="NN+covid+oil",PI=FALSE)
```



```r
NN_scores_co <- accuracy(NN_for_co$mean,msts_bitcoin_test)
#print(NN_scores_co)
# Use checkresiduals to plot and assess residuals
checkresiduals(NN_for_co)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,18)[365]
## Q* = 483.98, df = 438, p-value = 0.06367
##
## Model df: 0.   Total lags used: 438
```

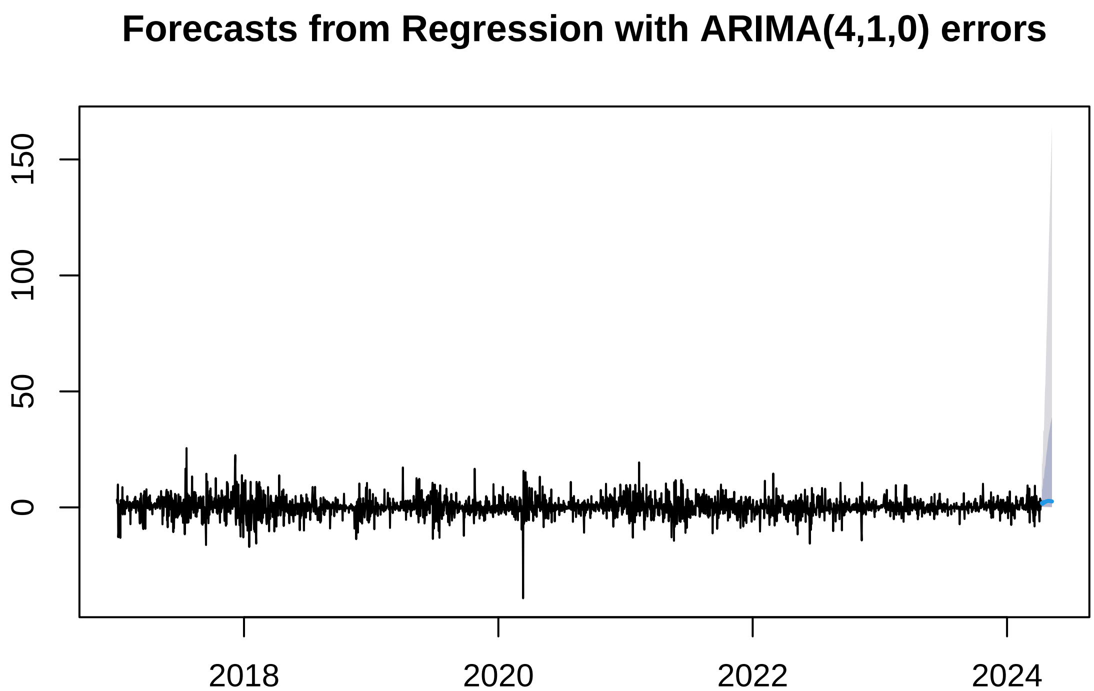
5. Forecast

```r
ts_oil_all <- final_data_filled %>% filter(DATE >= ymd("2017-01-01"))
combined_msts <- msts(ts_oil_all$DCOILBRENTEU,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2017,01,01))
ts_bit_all <- bitcoin %>% filter(date >= ymd("2017-01-01"))
combined_msts_bit <- msts(ts_bit_all$Change,
                           seasonal.periods =c( 91.25,365.25),
                           start=c(2017,01,01))
oil_regressors_pd<- as.matrix(data.frame(fourier(combined_msts_bit,K=c(3,12)),"oil"=combined_msts))
oil_fc_pd<-forecast(combined_msts, h=30)
oil_regressors_fc_pd<-as.matrix(data.frame(fourier(combined_msts_bit,K=c(3,12), h=30),"oil"= oil_fc_pd$
```

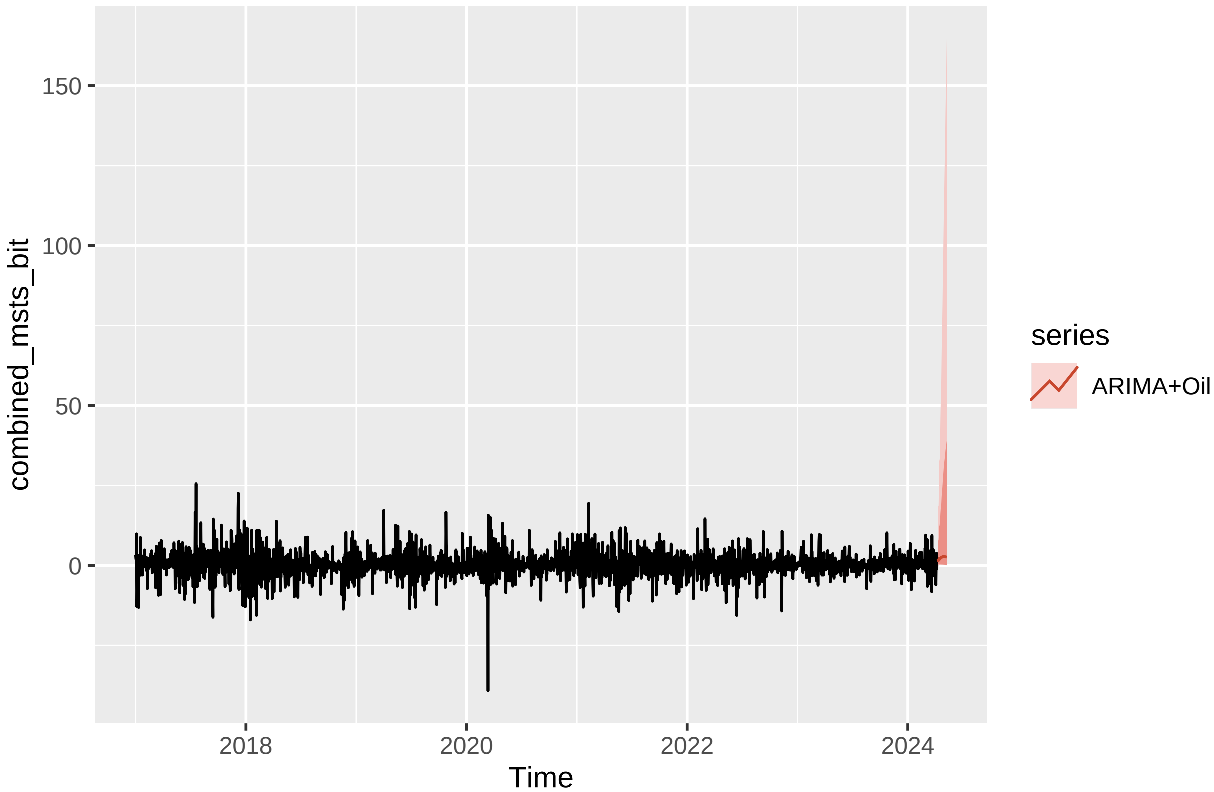
```r
arima_fit_r_pd <- auto.arima(combined_msts_bit,seasonal= FALSE, lambda=0,xreg=oil_regressors_pd)
```

```
## Warning in log(x): NaNs produced

## Warning in log(x): NaNs produced
```

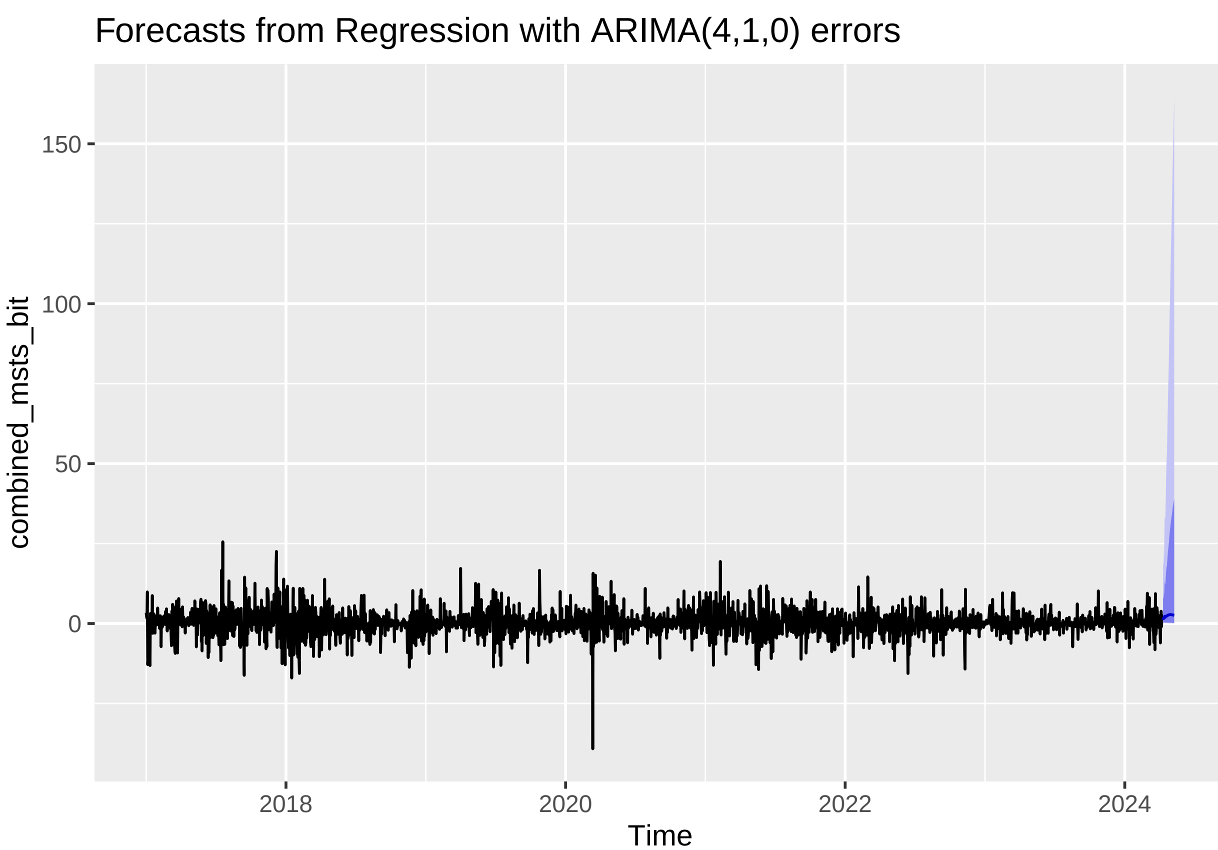
```r
arima_forecast_r_pd <- forecast(arima_fit_r_pd, xreg=oil_regressors_fc_pd,h=30)

plot(arima_forecast_r_pd)
```



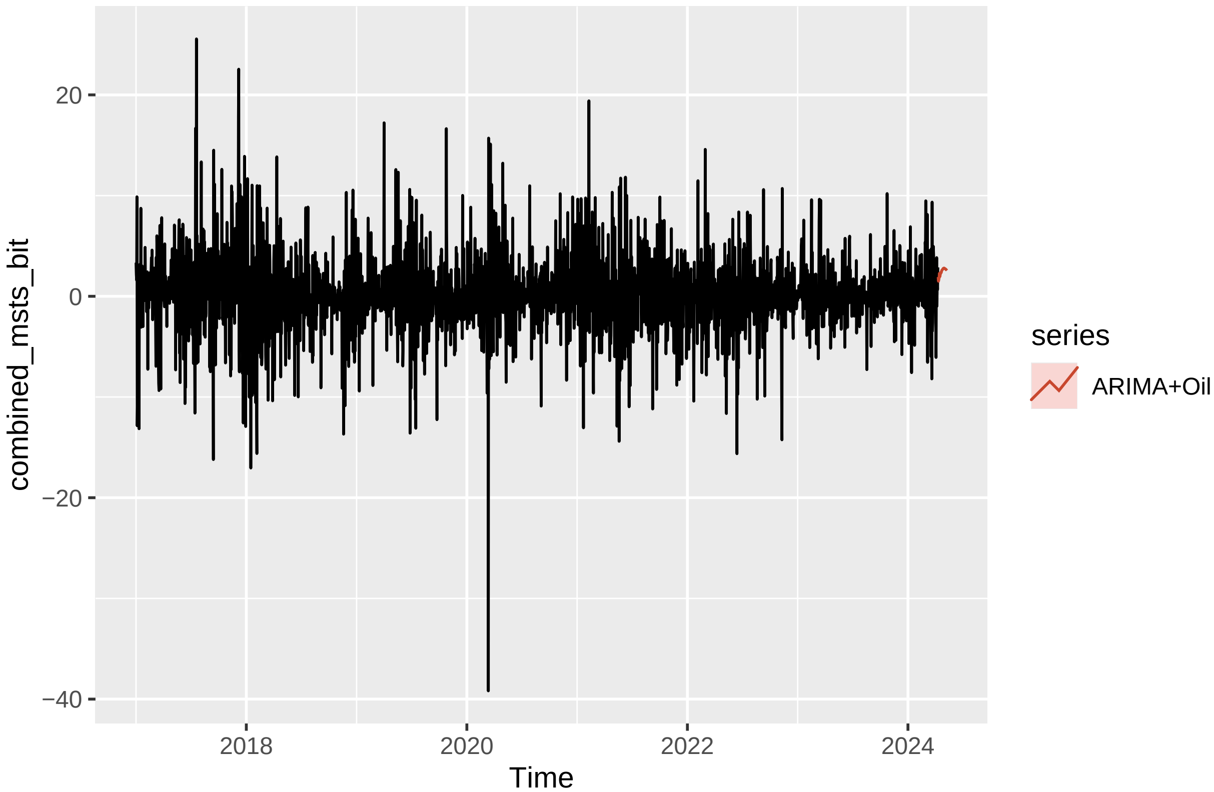

```r
autoplot(combined_msts_bit) + autolayer(arima_forecast_r_pd, series="ARIMA+Oil", PI=TRUE)
```



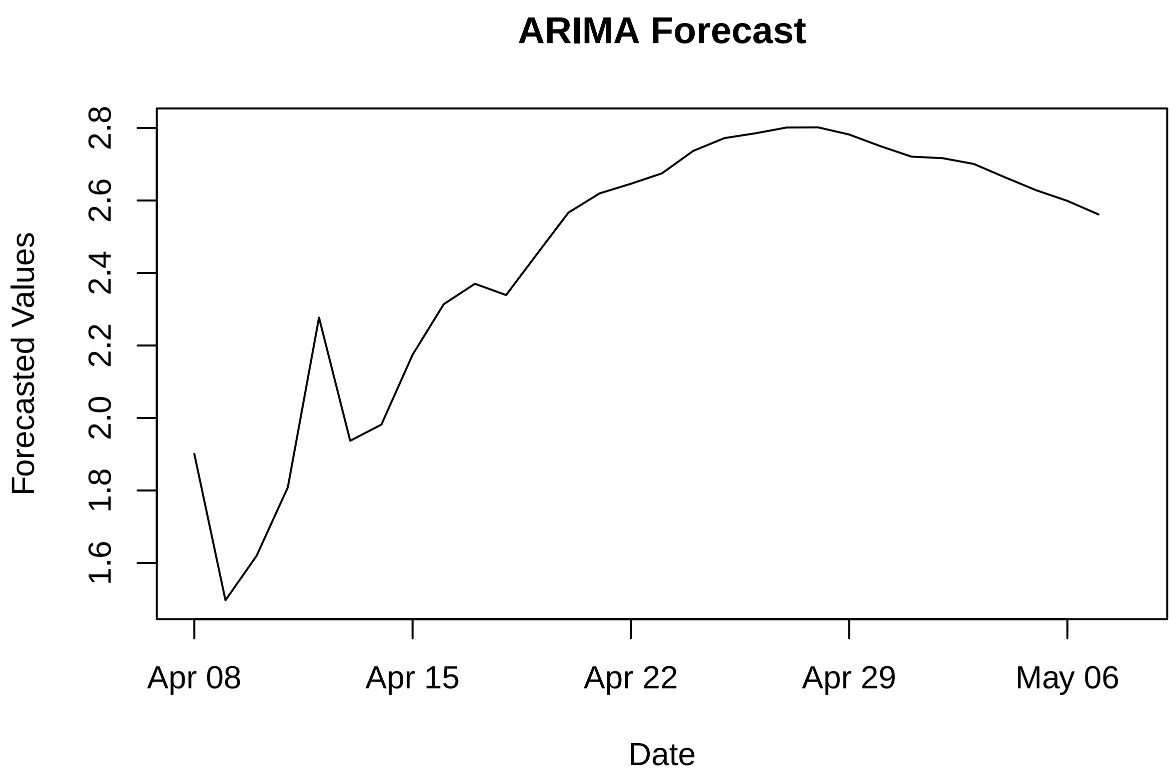

```r
autoplot(arima_forecast_r_pd)
```



```r
autoplot(combined_msts_bit) +
  autolayer(arima_forecast_r_pd, series="ARIMA+Oil",PI=FALSE)
```




```r
# Extract the mean forecast values
mean_forecast <- arima_forecast_r_pd$mean

# Length of mean_forecast
n <- length(mean_forecast)

# Extract the dates for the x-axis
dates <- seq(from = as.Date("2024-04-08"), by = "day", length.out = n)

# Plot the mean forecast with specified x-axis range
plot(dates, mean_forecast, type = "l", xlab = "Date", ylab = "Forecasted Values",
     main = "ARIMA Forecast", xlim = c(as.Date("2024-04-08"), as.Date("2024-05-08")))
```



6. Table

```
# Load required packages
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
# Combine scores
combined_scores <- rbind(
  ETS_scores,
  TBATS_scores,
  ARIMA_scores,
  ARIMA_scores1,
  NN_scores1,
  NN_scores3,
  ARIMA_scores_oil,
  ARIMA_scores_co,
  NN_scores_o,
  NN_scores_co
)

# Define row names
rownames(combined_scores) <- c(
  "ETS", "TBATS", "ARIMA", "ARIMA with Covid",
  "Neural Network with Fourier", "Neural Network with Covid",
  "ARIMA with Oil", "ARIMA with Covid and Oil",
```

```
  "Neural Network with Oil", "Neural Network with Covid and Oil"
)

knitr::kable(
  combined_scores,
  format = "latex",
  caption = "Combined Scores"
) %>%
  kable_styling(
    latex_options = c("hold_position", "scale_down"),
    full_width = FALSE
)
```

Table 1: Combined Scores

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ETS	0.6482149	3.360422	2.599518	86.34347	498.0130	-0.1136271	1.283959
TBATS	0.5822264	2.701150	1.852876	99.08521	205.2798	0.0192255	1.078225
ARIMA	0.8172699	4.580938	3.314904	414.25838	1024.7176	0.0374629	1.634431
ARIMA with Covid	1.5937054	3.013271	2.235520	61.13900	464.8829	-0.0445094	1.155884
Neural Network with Fourier	0.3654727	3.079558	2.242687	185.61673	414.9941	0.0152402	1.037852
Neural Network with Covid	0.4074562	2.823624	2.128092	114.48023	445.0392	-0.0417718	1.035712
ARIMA with Oil	-0.1168176	2.511553	1.699530	95.62725	177.5840	-0.0847061	1.115706
ARIMA with Covid and Oil	-1.2261912	2.803823	2.148202	100.87372	472.3427	-0.0731229	1.658998
Neural Network with Oil	0.3001861	2.964914	2.200633	205.13294	403.2813	0.0648576	1.538607
Neural Network with Covid and Oil	0.3705762	2.995247	2.197752	246.33732	412.4288	0.0413739	1.300681