Unit 9 Live Session Solutions

Time Series Analysis Part 4: Mixed Autoregressive Moving Average (ARMA) Models Autoregressive Integrated Moving Average (ARIMA) Models Seasonal ARIMA (SARIMA) Models

Figure 1: South Hall

Class Announcements

- No HW this week

Roadmap

Rearview Mirror

- Mathematical formulation and key properties of AR and MA process
- Comparing AR models and MA models using simulated series
- Discuss the steps to build a time series model: Box-Jenkins' approach

Today

- Mixed Autoregressive Moving Average (ARMA) Models
- An introduction to the non-stationary time series model
- Random walk and integrated processes
- Autoregressive Integrated Moving Average (ARIMA) Models
- Seasonal ARIMA (SARIMA) Models

Looking Ahead

- Regression with multiple trending time series
- Cointegration
- Multivariate Time Series Models: Vector Autoregressive (VAR) model

Start-up Code

```
# Load required libraries
## Load a set of packages including broom, cli, crayon, dbplyr, dplyr, dtplyr, forecast,
#googledrive, googlesheets4, ggplot2, haven, hms, httr, jsonlite, lubridate, magrittr,
#modelr, pillar, purrr, readr, readxl, reprex, rlang, rstudioapi, rvest, stringr, tibble,
#tidyr, xml2
library(tidyverse)
# To create and work with tidy temporal data
library(tsibble)
# To work with date-times and time-spans
library(lubridate)
# Provides a collection of commonly used univariate and multivariate time series models
library(fable )
## To interact directly with the Quandl API and download data
library(Quandl)
# For analyzing tidy time series data.
library(feasts)
# For simulation of SARIMA models
library(astsa)
library(patchwork)
```

An introduction to ARMA, ARIMA, and Seasonal ARIMA (SARIMA) Models

1. Last week, we were introduced to AR, MA, ARMA, and ARIMA models. AR, MA, and ARMA models are only appropriate when a time series is stationary.
2. We are often confronted with time series that are not stationary in the mean and variance (and other forms, such as seasonality and volatility clustering). Luckily, many non-stationary series can be simply transformed into stationary series using straightforward transformations such as differencing.
3. Here are some common reasons why time series might not be stationary in the mean:
    a. The series has a trend
    b. The series contains seasonal elements
    c. The series contains time-varying variance
4. We can take care of some of these problems by de-trending the data or by differencing the data. In the context of ARIMA modeling, first difference the data and then fit an ARMA model to the differenced series. If the data are seasonal, we use additional seasonal parameters, resulting in a SARIMA model - an ARIMA model with seasonal components.
5. Remember, here are the steps to building an ARIMA model (assuming that you already have your questions well-defined and data collected and cleansed):
    i. Conduct an EDA to determine if you need to transform the series to make it stationary.
    ii. Transform the series if needed.
    iii. Estimate several `ARIMA(p, d, q)x(P, D, Q)s` models, with starting values coming from examining time series plot, ACF, and PACF.
    iv. Evaluate the residuals of models with the lowest AIC and/or BIC values and more parsimonious models. Select the model where the residuals resemble a white nose.
    v. Answer your question / generate forecasts!

Seasonal Autoregressive Integrated Moving Average SARIMA(p,d,q)(P,D,Q)s Models

SARIMA models combine ARIMA models with additional parameters to capture seasonal components in time series data. We basically specify an "additional" ARIMA model for the seasonal components of the process.

This means there are seven parameters we need to estimate in the data. There are the normal ARIMA parameters: p the number of non seasonal AR lags, q the number of non seasonal MA lags, and d the non seasonal order of differencing. But then there are also the same components for the seasonal part of the process: P the number of seasonal AR lags, Q the number of seasonal MA lags, and D the seasonal differencing. There is also s the number of observations per year or seasonal period we care about. For monthly data we usually have $s = 12$ and for quarterly data we usually set $s = 4$ and so on.

We can express SARIMA(p,d,q)(P,D,Q)s models using the backshift notation as follows:

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D x_t = \mu + \theta(B)\Theta(B^s)\epsilon_t$$

This shows that SARIMA models just have additional differencing and terms at seasonal lags in the time series process to capture the seasonal effects. Typically we don't want the total differencing $d + D$ in SARIMA models to be much more than 2 and generally do not set $D > 1$. Otherwise we risk overfitting the data.

Parameter Redundancy in ARIMA/SARIMA Models

One thing that can happen in SARIMA (and ARIMA) models is parameter redundancy. For example, suppose we have a random walk process:

$$x_t = \epsilon_t$$

We could also write that as:

$$x_t = x_{t-1} + \epsilon_t - \epsilon_{t-1}$$

The first formulation is an $ARIMA(0,0,0)$ model while the second formulation is an $ARIMA(0,1,1)$ model even though they describe the exact same process.

Hence the same time series can be described by multiple time series models, but there is also a model that represents the most parimonious expression of a time series process. We want to ensure we are estimating the most parismonious version and so prefer models that have less parameters.

To see that a time series process has parameter redundancy and can be reduced, we can use the backshift notational form.

For the process above, we could write it as:

$$x_t = x_{t-1} + \epsilon_t - \epsilon_{t-1}$$

$$x_t - x_{t-1} = \epsilon_t - \epsilon_{t-1}$$

$$(1 - B)x_t = (1 - B)\epsilon_t$$

So we can simplify the backshift notation form by dividing through by $(1 - B)$ to remove the parameter redundancy. This is a general fact about backshift notation and can help in simplifying time series.

An Analytical Analysis

- For following specifications:

1- $y_t = y_{t-1} - \frac{1}{4}y_{t-2} + w_t - \frac{1}{2}w_{t-1}$

2- $y_t = \frac{4}{3}y_{t-1} - \frac{1}{3}y_{t-2} + w_t - w_{t-1} + \frac{1}{4}w_{t-2}$

3- $y_t = \frac{1}{4}y_{t-1} + \frac{1}{10}y_{t-12} - \frac{1}{40}y_{t-13} + w_t + \frac{1}{2}w_{t-1}$

4- $y_t = y_{t-1} + y_{t-4} - y_{t-5} + w_t - \frac{1}{2}w_{t-1} - \frac{1}{2}w_{t-4} + \frac{1}{4}w_{t-5}$

- Where $\{w_t\}$ is white noise with zero means and standard deviation 1.

a) Rewriting the processes in `Seasonal ARIMA(p,d,q)(P,D,Q)` notation.

b) Check for parameter redundancy

c) Check the stationarity and invertibility of these processes

d) Identify each series

e) Compute the Mean of each process (If it is stationary)

f) Simulate each process separately

g) Check ACF and PACF plots of each simulation and discuss the pattern in these two plots

1- $y_t = y_{t-1} - \frac{1}{4}y_{t-2} + w_t - \frac{1}{2}w_{t-1}$

a) `Seasonal ARIMA(p,d,q)(P,D,Q)` notation

- $(1 - B + \frac{1}{4}B^2)y_t = (1 - \frac{1}{2}B)w_t$
- $(1 - \frac{1}{2}B)^2 y_t = (1 - \frac{1}{2}B)w_t$

b) Check for parameter redundancy

- The AR and MA share a common factor, after simplification:

$(1 - \frac{1}{2}B)y_t = w_t$

c) Check the stationarity and invertibility

- It is stationary since $B = 2$ is the solution to the characteristic equation taken from the left-hand side of the equation.

d) Identify each series

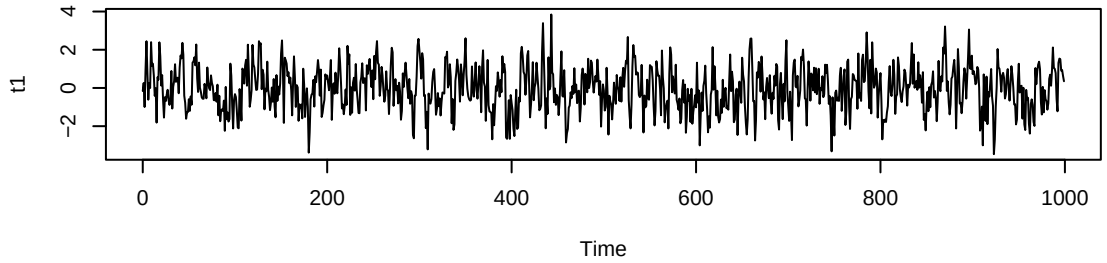
- It's a AR(1) process

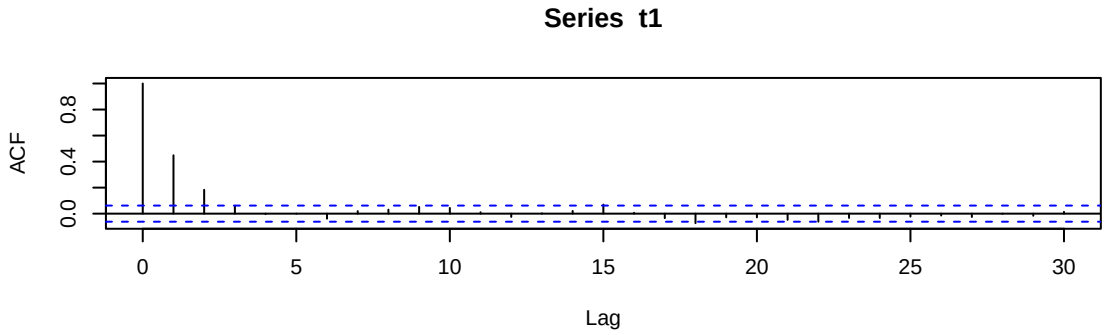
e) Compute the Mean

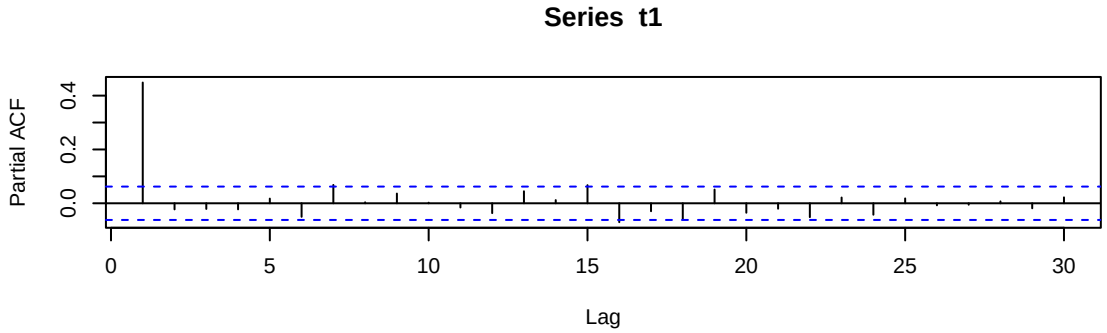
- $E(y_t) - \frac{1}{2}E(y_{t-1}) = E(w_t) - \frac{1}{2}E(w_{t-1})$
- Because it's stationary process $E(y_t) = E(y_{t-1}) = \mu$, so $\mu - \frac{1}{2}\mu = 0$, then $\mu = 0$

f) & g) Simulate and check ACF and PACF plots

```
set.seed(1)
t1<-sarima.sim(n = 1000, ar = c(1/2), d = 0, ma = NULL, sar = NULL, D = 0, sma = NULL, S = NULL)

par(mfrow=c(3,1))
plot(t1, type ="l")
acf(t1)
pacf(t1)
```







The time series plot fluctuates around its mean of 1, and each period oscillates around it with mean reversion. The ACF declines kind of fast over time. the PACF plot is significant up to 1 lag, which is the number of terms in the AR(1) process.

2- $y_t = \frac{4}{3}y_{t-1} - \frac{1}{3}y_{t-2} + w_t - w_{t-1} + \frac{1}{4}w_{t-2}$

a) `Seasonal ARIMA(p,d,q)(P,D,Q)` notation

- $(1 - \frac{4}{3}B + \frac{1}{3}B^2)y_t = (1 - B + \frac{1}{4}B^2)w_t$
- $(1 - B)(1 - \frac{1}{3}B)y_t = (1 - \frac{1}{2}B)^2 w_t$

b) Check for parameter redundancy

- No common factor

c) Check the stationarity and invertibility

- It is non-stationary since the characteristic equation taken from the left-hand-side has a unit root $B = 1$.
- It is invertible since the roots of the right-hand-side equation exceed unity $B = 2$.

d) Identify each series

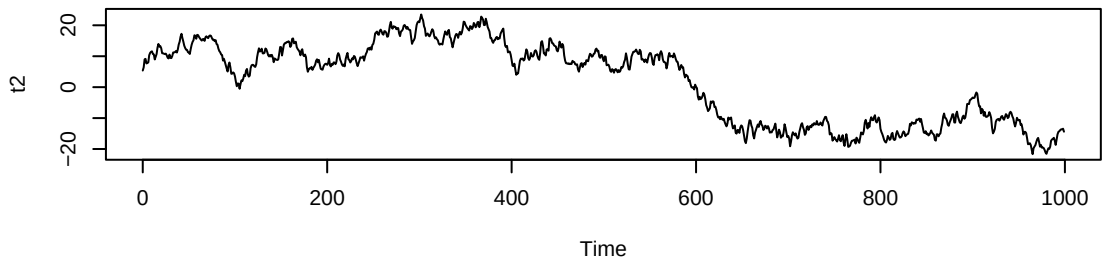
- It's a ARIMA(1,1,2) process

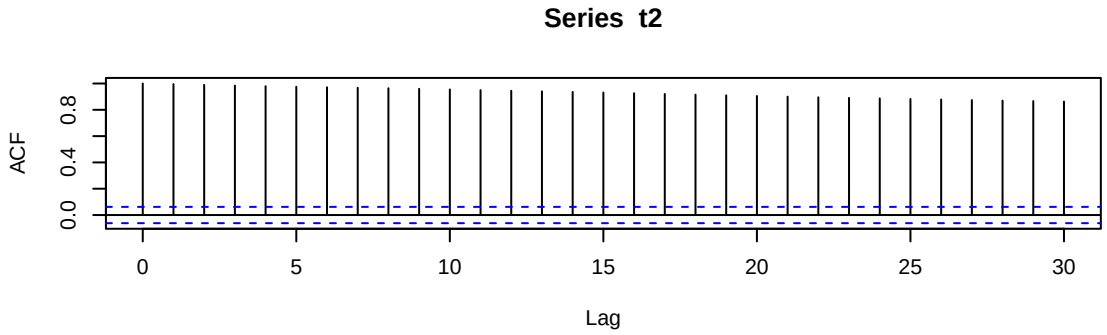
e) Compute the Mean

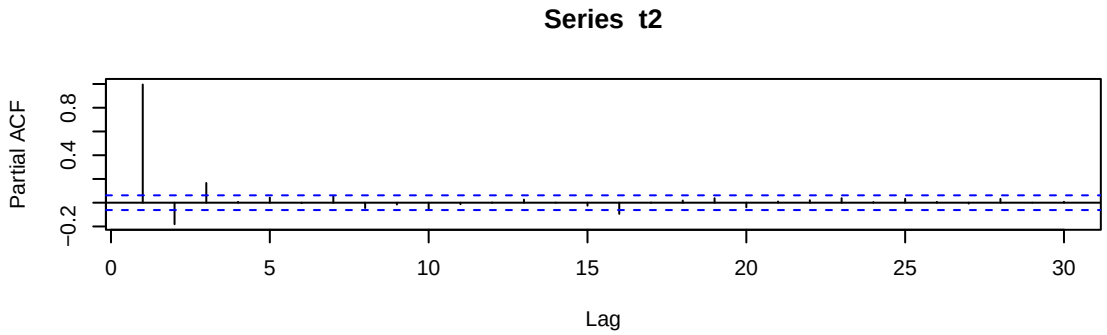
- It is a non-stationary process with a non-constant mean

f) & g) Simulate and check ACF and PACF plots

```
set.seed(1)
t2<- sarima.sim(n = 1000, ar = c(1/3), d = 1, ma = c(0,-1/4), sar = NULL, D = 0, sma = NULL, S = NULL)

par(mfrow=c(3,1))
plot(t2, type ="l")
acf(t2)
pacf(t2)
```







Since this process has a unit root, the time series plot has apparent trends up or down with sudden and unpredictable changes in direction. The ACF decays very slowly, and the first bar of the PACF is almost equal to 1.

3- $y_t = \frac{1}{4}y_{t-1} + \frac{1}{10}y_{t-12} - \frac{1}{40}y_{t-13} + w_t + \frac{1}{2}w_{t-1}$

a) `Seasonal ARIMA(p,d,q)(P,D,Q)` notation

- $(y_t - \frac{1}{4}y_{t-1}) - \frac{1}{10}(y_{t-12} - \frac{1}{4}y_{t-13}) = w_t + \frac{1}{2}w_{t-1}$
- $(y_t - \frac{1}{4}y_{t-1}) - \frac{1}{10}B^{12}(y_t - \frac{1}{4}y_{t-1}) = w_t + \frac{1}{2}w_{t-1}$
- $(y_t - \frac{1}{4}y_{t-1})(1 - \frac{1}{10}B^{12}) = w_t + \frac{1}{2}w_{t-1}$
- $y_t(1 - \frac{1}{4}B)(1 - \frac{1}{10}B^{12}) = (1 + \frac{1}{2}B)w_t$

b) Check for parameter redundancy

- No common factor

c) Check the stationarity and invertibility

- It's stationary since left-hand side contains the term $(1 - \frac{1}{4}B)\ (1 - \frac{1}{10}B^{12})$, which means both roots $B = 4\ B = 10^{\frac{1}{12}}$ exceed unity .
- It is invertible since the roots of the right-hand side are less than unity $B = -2$.

d) Identify each series

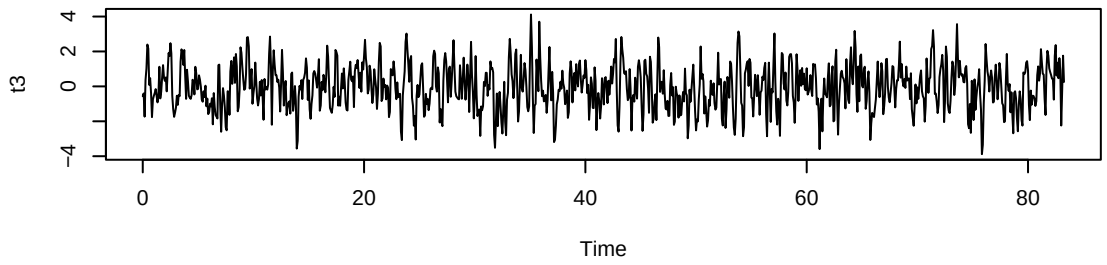
- It is a SARIMA(1,0,1)(1,0,0)12 process

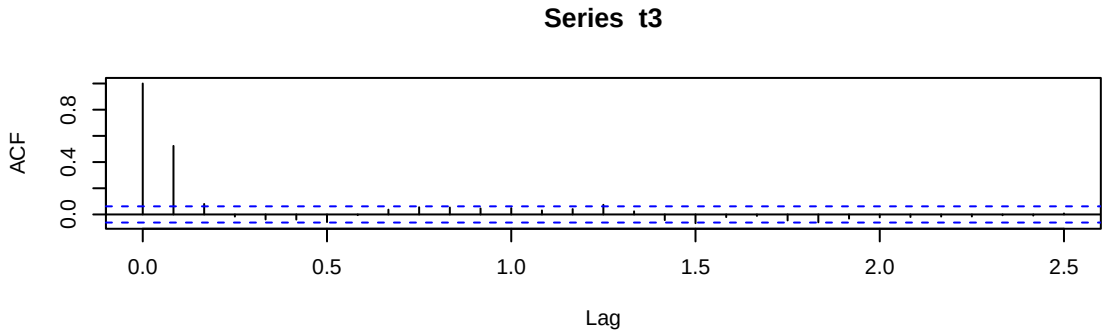
e) Compute the Mean

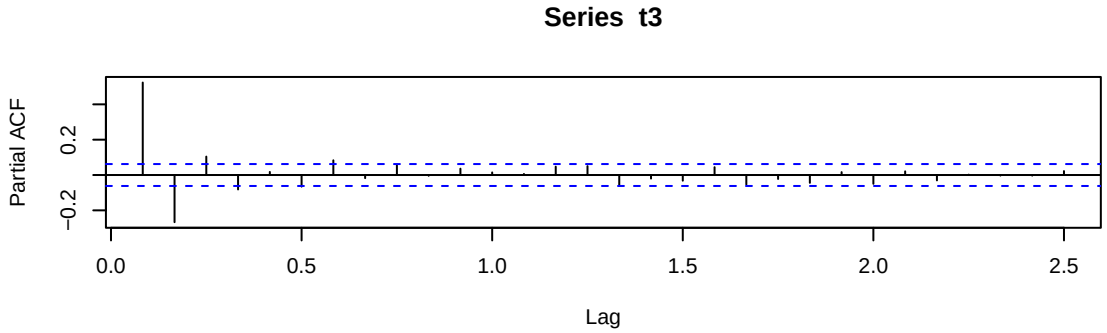
- $E(y_t) - \frac{1}{4}E(y_{t-1}) - \frac{1}{10}E(y_{t-12}) - \frac{1}{40}E(y_{t-13}) = E(w_t) + \frac{1}{2}E(w_{t-1})$
- Because it's a stationary process:
- $E(y_t) = E(y_{t-1}) = E(y_{t-12}) = E(y_{t-13}) = \mu$
- $\mu - \frac{1}{4}\mu - \frac{1}{10}\mu - \frac{1}{40}\mu = 0$, then $\mu = 0$

f) & g) Simulate and check ACF and PACF plots

```
set.seed(1)
t3<-sarima.sim(n = 1000, ar = c(1/4), d = 0, ma = c(1/2), sar = c(1/12), D = 0, sma = NULL, S = 12)

par(mfrow=c(3,1))
plot(t3, type ="l")
acf(t3)
pacf(t3)
```







The time series plot fluctuates around its mean (1.2). The ACF and PACF both decay pretty fast, which is a usual pattern in the ARMA process, and also we can see an oscillations pattern in the ACF, which could be a sign of a seasonal component

4- $y_t = y_{t-1} + y_{t-4} - y_{t-5} + w_t - \frac{1}{2}w_{t-1} - \frac{1}{2}w_{t-4} + \frac{1}{4}w_{t-5}$

a) `Seasonal ARIMA(p,d,q)(P,D,Q)` notation.

- $y_t = y_{t-1} + y_{t-4} - y_{t-5} + w_t - \frac{1}{2}w_{t-1} - \frac{1}{2}w_{t-4} + \frac{1}{4}w_{t-5}$
- $(y_t - y_{t-1}) - (y_{t-4} - y_{t-5}) = (w_t - \frac{1}{2}w_{t-1}) - \frac{1}{2}(w_{t-4} - \frac{1}{2}w_{t-5})$
- $(y_t - y_{t-1}) - B^4(y_t - y_{t-1}) = (w_t - \frac{1}{2}w_{t-1}) - \frac{1}{2}B^4(w_t - \frac{1}{2}w_{t-1})$
- $(y_t - y_{t-1})(1 - B^4) = (w_t - \frac{1}{2}w_{t-1})(1 - B^4)$
- $y_t(1 - B)(1 - B^4) = w_t(1 - \frac{1}{2}B)(1 - \frac{1}{2}B^4)$

b) Check for parameter redundancy

- No common factor

c) Check the stationarity and invertibility

- It is non-stationary since the polynomial on the left-hand side contains the term $(1 - B)$ and $(1 - B^4)$, which implies that there exists a unit root $B = 1$ in both non-seasonal and seasonal components.
- The non-seasonal MA is invertible since its roots are greater than unity $B = 2$, also the seasonal MA is invertible since its root $B = 2^{\frac{1}{4}}$ is greater than unity.

d) Identify each series

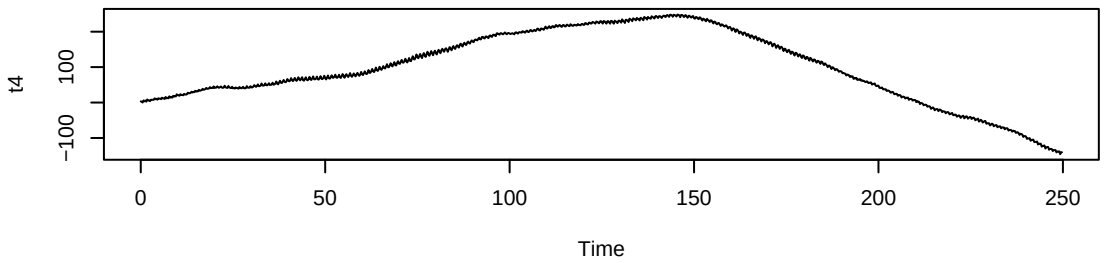
- It's a SARIMA(0,1,1)(0,1,1)4 process

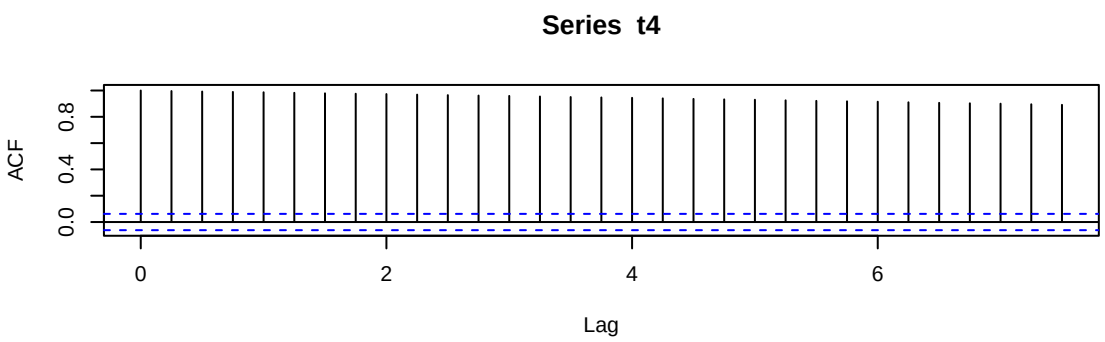
e) Compute the Mean

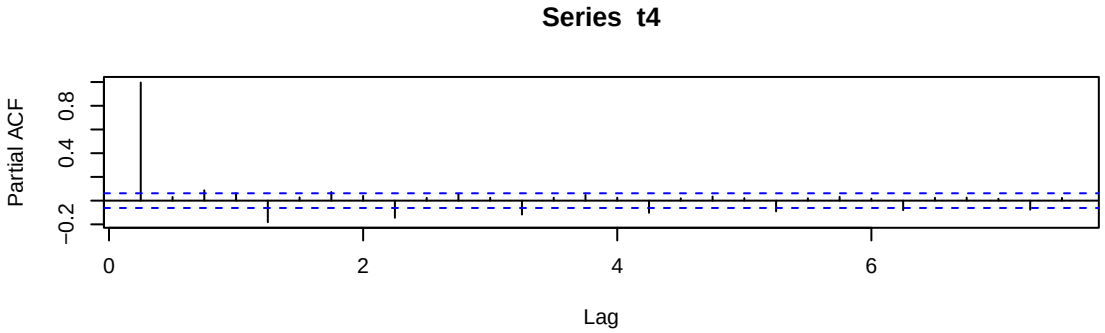
- It is a non-stationary process with a non-constant mean

f) & g) Simulate and check ACF and PACF plots

```
set.seed(1)
t4<-sarima.sim(n = 1000, ar = NULL, d = 1, ma = c(-1/2), sar=, D = 1, sma = c(-1/2), S = 4)

par(mfrow=c(3,1))
plot(t4,type="l")
acf(t4)
pacf(t4)
```







The ACF decays very slowly, and the first bar of the PACF is almost equal to 1.

Is the efficient market hypothesis true in the cryptocurrency market?

Bitcoin has become a fad among investors despite its nature and characteristics ambiguity. Investors can speculate on bitcoin prices and buy when they think prices are going up and sell when they think prices are decreasing. This would be profitable for investors if they could accurately predict bitcoin prices. However, the predictability of bitcoin prices has been questioned. According to the efficient market hypothesis, asset prices are unpredictable, so one can not beat the market.

The efficient market hypothesis is associated with the "random walk hypothesis," which states that asset prices are random walk, like the steps a drunk takes, and, therefore, unpredictable.

$$P_t = P_{t-1} + w_t$$

- where $\{w_t\}$ is a Gaussian white noise $w_t \sim N(0, \sigma^2)$.

The current consensus is that the random walk is explained by the efficient market hypothesis, that the markets quickly and efficiently react to new information about assets, so most of the fluctuations in prices are explained by the changes in the instantaneous demand and supply of any given asset, causing the random walk-in prices. An implication is that technical analysis is undependable.

This exercise aims to investigate whether the cryptocurrency market is predictable and if the efficient market hypothesis is true

To test the null hypothesis of random walk or efficient market, we estimate an ARIMA and a random walk model for the period 01/2017 to 01/2022 and compare the six-month a head forecast performances of these two models. If the ARIMA model outperforms the random walk model, it suggests that the cryptocurrency market is not efficient during the sample period.

What does unpredictable price mean?

To better understate the random walk hypothesis, answer the following questions.

a) If the crypto price P_t follow a random walk process, what are the forecasts of P_{t+1}, P_{t+2} and P_{t+h}

- One step ahead forecast:

$E_t(P_{t+1}) = E_t(P_t + w_t) = E_t(P_t) + E_t(w_t) = p_t$,

- Two steps ahead forecast:

$E_t(P_{t+2}) = E_t(P_{t+1} + w_{t+1}) = E_t(P_{t+1}) + E_t(w_{t+1}) = E_t(P_t + w_t) + E_t(w_{t+1}) = P_t$

- H steps ahead forecast:

$E_t(P_{t+h}) = P_t$

So if the bitcoin price follows a random walk, the changes in the future price are white noise and unpredictable, and our best guess about the future price is today's price.

b) If the crypto price P_t follow a random walk process, what are the variances of forecast of P_{t+1}, P_{t+2}, and P_{t+h}

- Variance of one step ahead forecast:

$Var_t(P_{t+1}) = Var_t(P_t + w_t) = \sigma^2$,

- Variance of two steps ahead forecast:

$Var_t(P_{t+2}) = Var_t(P_{t+1}) + Var(w_{t+1}) = Var_t(P_t + w_t) + Var_t(w_{t+1}) = Var_t(P_t) + Var_t(w_t) + Var_t(w_{t+1})$

$Var_t(P_{t+2}) = \sigma^2 + \sigma^2 = 2\sigma^2$

- Variance of h step ahead forecast:

$Var_t(P_{t+h}) = h\sigma^2$

The variance of our forecasts increases as the forecast horizon increases, and as a result, the prediction interval also increases with the forecast horizon.

Data Description and wrangling

This study only incorporates Bitcoin (BTC) prices but can be replicated for other cryptocurrencies to check for the robustness of its results.

We use a Quandl package to download the bitcoin prices from Quandl(). Quandl is a platform that provides its users with economic, financial, and alternative datasets. Users can download free data, buy paid data or sell data to Quandl. So the first step is to create an account that allows you to access their API, libraries, and tools.

```
# Quandl.api_key("API KEY")
# bitcoin_daily = Quandl("BCHAIN/MKPRU", start_date="2017-01-01") %>%
#    as_tsibble(index=Date)


# ## Importing form the file
bitcoin_daily <- read_csv("./data/Bitcoin_prices.csv") %>%
   as_tsibble(index=Date)
```

```
## Rows: 2007 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): Value
## date (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#class(bitcoin_daily)
#head(bitcoin_daily)
#tail(bitcoin_daily)
```

In order to use Quandl() to download the bitcoin prices, please go to https://data.nasdaq.com/, create an account, go to the account setting and copy your API key here

Training and test sets

In order to evaluate forecast accuracy, we separate the data set into training and test data. The training data is used to estimate model parameters, and it is for 01/2017-01/2022. The test data is used to evaluate its accuracy, and it is for 01/2022-current.

```
bitcoin_training <-bitcoin_daily %>%
  filter( Date < ymd(20220101))

tail(bitcoin_training)
```

```
## # A tsibble: 6 x 2 [1D]
##   Date        Value
##   <date>      <dbl>
## 1 2021-12-26 50471.
## 2 2021-12-27 50801.
## 3 2021-12-28 50692.
## 4 2021-12-29 47601.
## 5 2021-12-30 46409.
## 6 2021-12-31 47133.
```

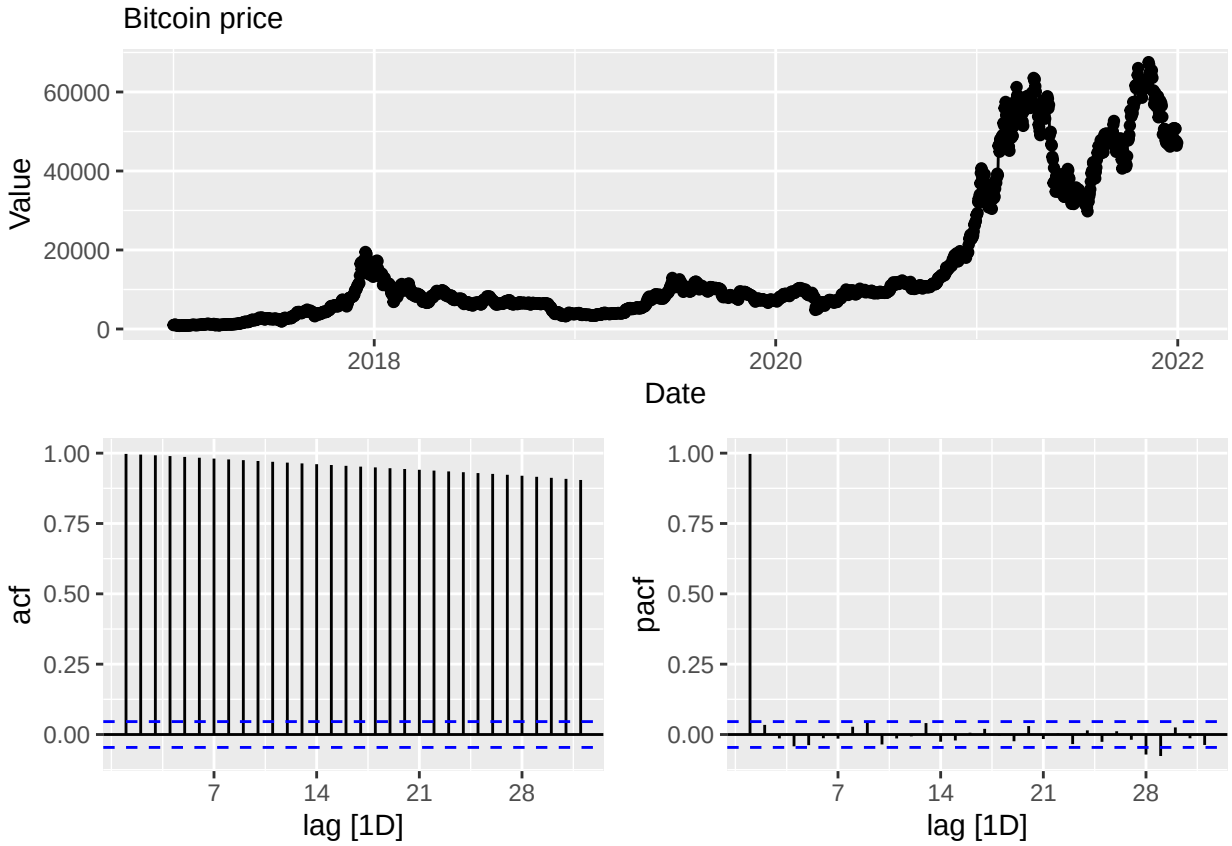
```
bitcoin_test <-bitcoin_daily %>%
  filter( Date >= ymd(20220101))

#head(bitcoin_test)
#tail(bitcoin_test)
```

We use the training set to estimate models and then compare their forecasting accuracy in the test set

Model Development

a) Plot graphs of the training data set and the ACF/PACF. What do you notice? Is there any transformation necessary?

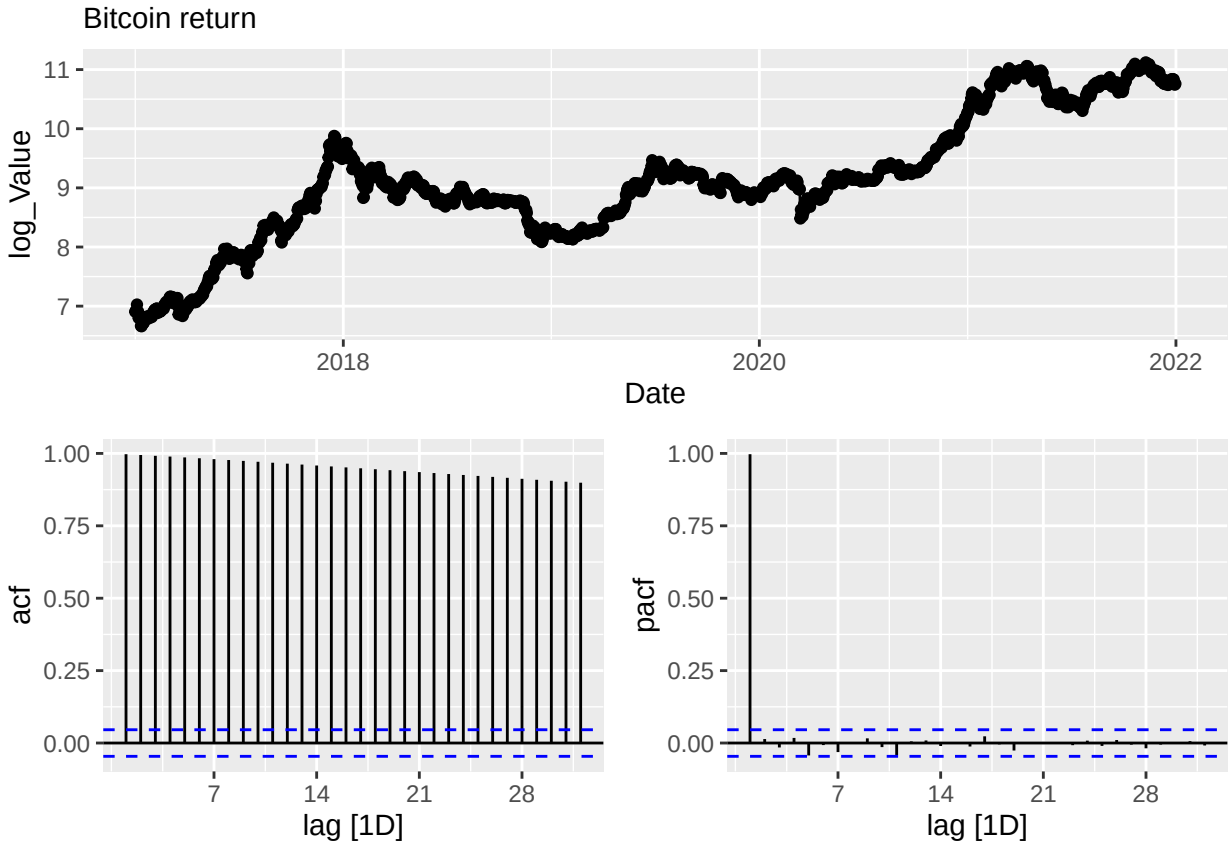
```
bitcoin_training <- bitcoin_training %>%
  mutate(log_Value = log(Value))
bitcoin_training %>% gg_tsdisplay(Value, plot_type="partial") +labs(subtitle = "Bitcoin price")
```



The time series plot of bitcoin price shows some non-stationarity, with an overall price increase.

It seems that variance or fluctuation in bitcoin price increases over the time, so we will use log transformation

```
bitcoin_training %>% gg_tsdisplay(log_Value, plot_type="partial") +labs(subtitle = "Bitcoin return")
```



The ACF decay very slowly, and the first lag of PACF is 1 in both price series and log of price series, suggesting that these series have a unit root.

b) Use a unit root test to determine whether differencing is required more objectively. If data is not stationary, difference the data, and apply the test again until it becomes stationary? How many differences are needed to make data stationary?

```
bitcoin_training %>%
  features(log_Value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      14.1        0.01
```

```
bitcoin_training %>%
  mutate(diff_log = difference(log_Value)) %>%
  features(diff_log, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.127         0.1
```

```
bitcoin_training %>%
  features(log_Value, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

We use the KPSS unit root test, and in this test, the null hypothesis is that the series is stationary. It is the opposite of the null hypothesis in the ADF and PP unit root test.

In the KPSS unit root test for the log of the prices, the p-value is 0.01, which is less than 0.05, indicating that the null hypothesis of stationary is rejected and the log series is not stationary. So first difference is required.

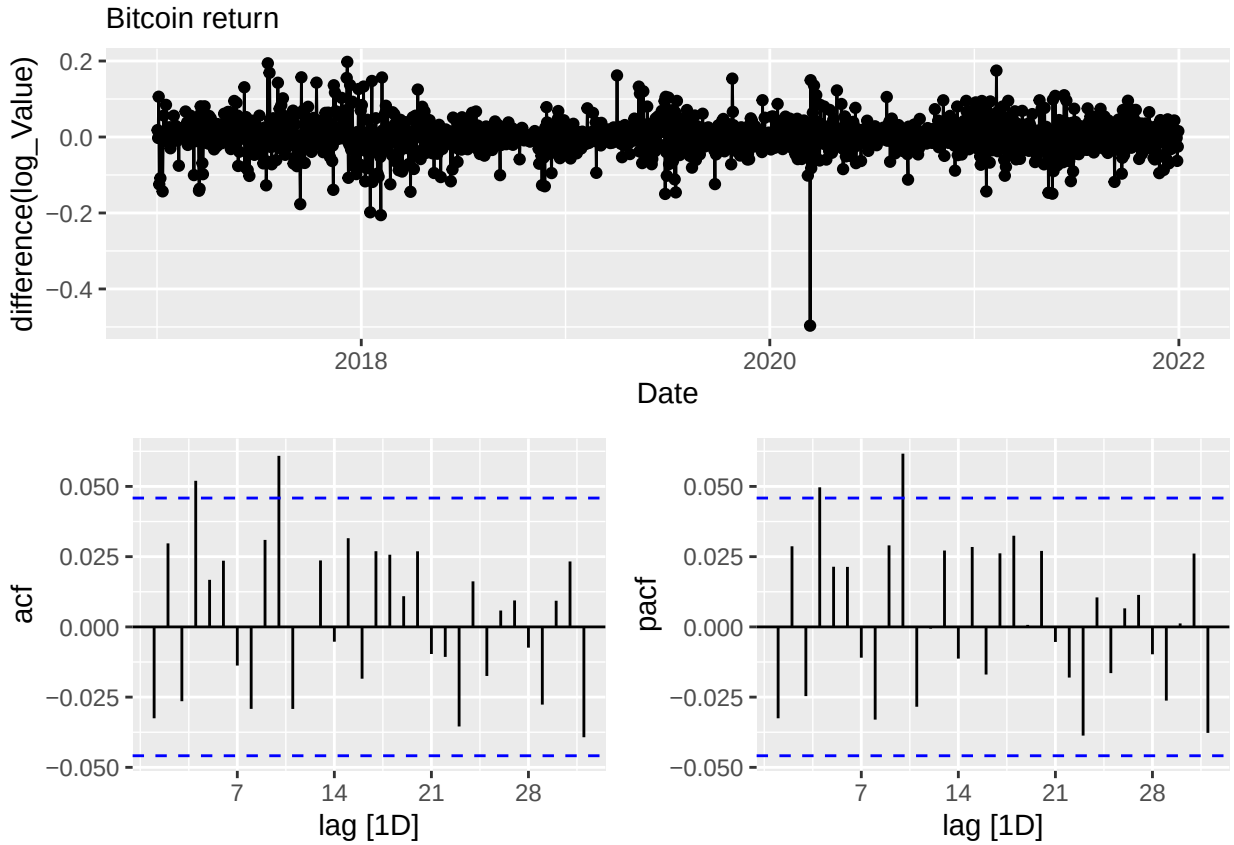
In the KPSS unit root test for the difference of the log of price, the p-value is 0.1, and it is greater than 0.05, so we fail to reject the null hypothesis of stationary

Only one difference is required to make the bitcoin price stationary.

c) Plot graphs of the differenced data and try to identify an appropriate ARIMA model.

```
bitcoin_training %>% gg_tsdisplay(difference(log_Value), plot_type="partial") +labs(subtitle = "Bitcoin return")
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
## Warning: Removed 1 rows containing missing values (geom_point).
```



The ACF suggests no lag for non-seasonal MA since first lags are insignificant and maybe some lag of seasonal MA, but it is not clear. The PACF suggests the same thing for AR lags, no non-seasonal AR lags but probably some seasonal AR lag. In this case, we need to search a set of different possible models, compare their AIC/BIC, and select the model with the lowest values. But we do not need to do that, thanks to ARIMA()

d) Use ARIMA() to estimate a random walk model and find an appropriate ARIMA model. What model was selected? Check the in-sample performance of the models using information criteria. Which one performs better?

```
bit_fit <- bitcoin_training %>%
  model(
  random_walk = NAIVE(log_Value),
  random_walk_1 = ARIMA(log_Value~0+pdq(0,1,0)+PDQ(0,0,0)),
  arima_fit = ARIMA(log_Value)
  )

report(bit_fit[1])
```

```
## Series: log_Value
## Model: NAIVE
##
## sigma^2: 0.0018
```

```
report(bit_fit[2])
```

```
## Series: log_Value
## Model: ARIMA(0,1,0)
##
## sigma^2 estimated as 0.001816:  log likelihood=3169.54
## AIC=-6337.09   AICc=-6337.09   BIC=-6331.58
```

```
report(bit_fit[3])
```

```
## Series: log_Value
## Model: ARIMA(0,1,0)(2,0,0)[7] w/ drift
##
## Coefficients:
##          sar1     sar2  constant
##       -0.0139  -0.0055    0.0022
## s.e.   0.0235   0.0236    0.0010
##
## sigma^2 estimated as 0.001814:  log likelihood=3171.99
## AIC=-6335.98   AICc=-6335.96   BIC=-6313.95
```

To estimate the random walk, we can use either NAIVE() or ARIMA(). When we use ARIMA() to estimate the random walk model without drift, we have to include a zero intercept explicitly.

Both NAIVE() and ARIMA() report the only one parameter, which is the variance of the white noise component, and it is 0.0018

in both

The automated model selection has found that an ARIMA(0,1,0)(2,0,0) gives the lowest AICc value, which is pretty close to what we guess; there is no non-seasonal AR and MA term in the model.

Since the selected model by ARIMA() and Random walk has the same degree of differencing, we can use AIC/BIC/cAIC to compare them. The AIC/BIC/cAIC are slightly smaller for the random walk process, which confirms that the bitcoin price follows a random walk.

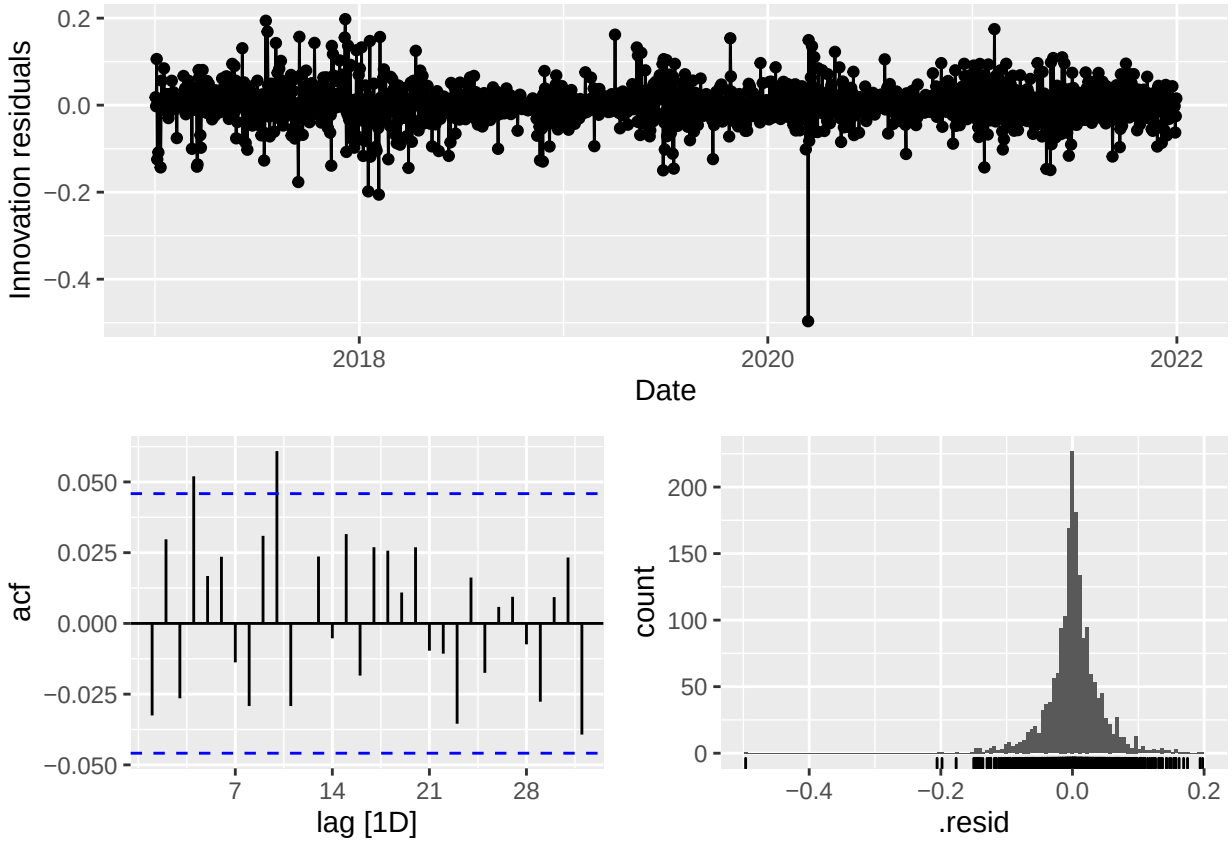
e) Do residual diagnostic checking of random walk and ARIMA model. Are the residuals white noise? Use the Ljung-box test to check if the residuals are white noise.

```
bit_fit%>%
   select(random_walk)%>%
   gg_tsresiduals()
```

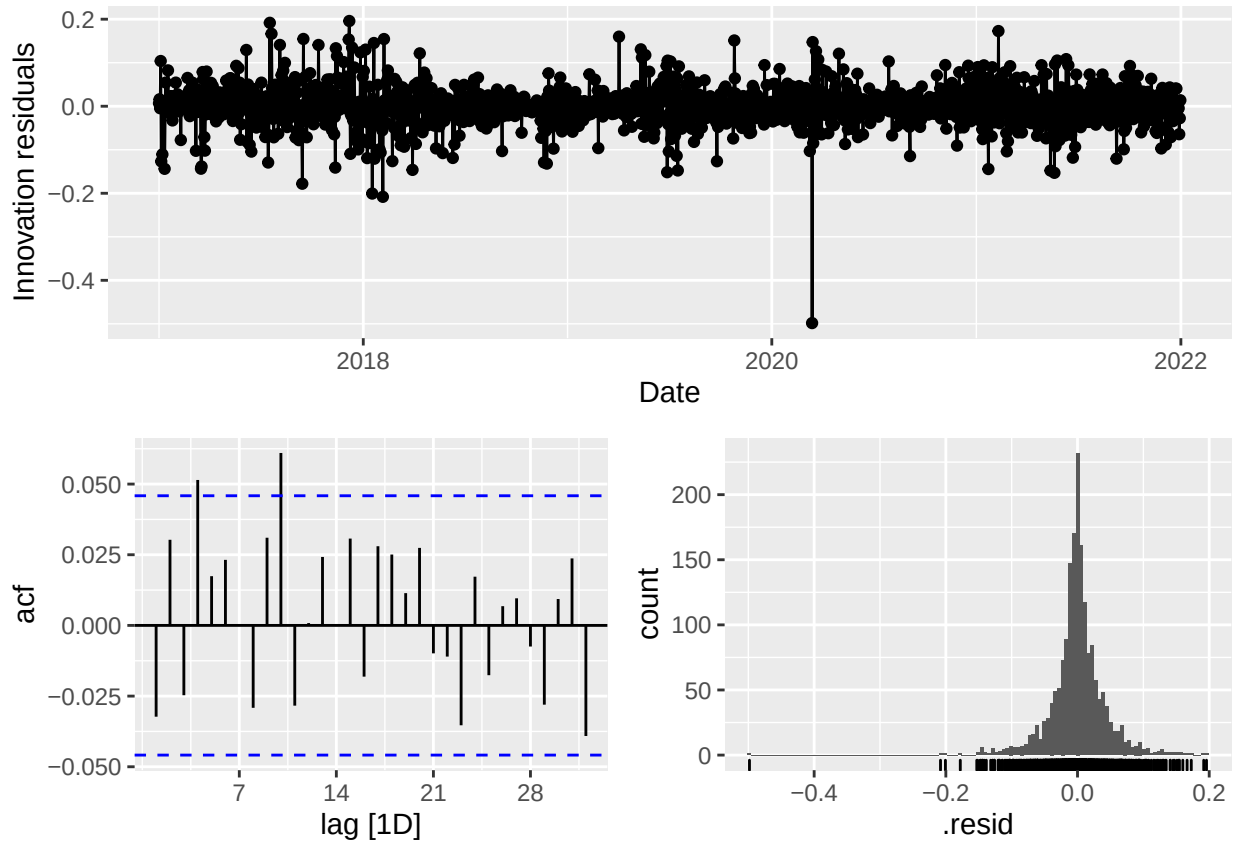
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



```
bit_fit %>%
  select(arima_fit)%>%
   gg_tsresiduals()
```




```
augment(bit_fit) %>%
  filter(.model == "random_walk") %>%
  features(.innov, ljung_box, lag = 10, dof = 0)
```

```
## # A tibble: 1 x 3
##   .model      lb_stat lb_pvalue
##   <chr>         <dbl>     <dbl>
```

```
## 1 random_walk    21.8    0.0162
```

```
bit_fit[2]%>% report()
```

```
## Series: log_Value
## Model: ARIMA(0,1,0)
##
## sigma^2 estimated as 0.001816:  log likelihood=3169.54
## AIC=-6337.09   AICc=-6337.09   BIC=-6331.58
```

```
augment(bit_fit) %>%
   filter(.model == "arima_fit") %>%
   features(.innov, ljung_box, lag = 10, dof = 3)
```

```
## # A tibble: 1 x 3
##   .model    lb_stat lb_pvalue
##   <chr>       <dbl>     <dbl>
## 1 arima_fit    21.3   0.00340
```

There are a few significant spikes in the ACF of both models, suggesting that the residuals are correlated.

Based on the Ljung-Box test, the p-value is less than 5% for both models, and we reject the null hypothesis of no serial correlation. These results suggest that the residuals are not white noise. So we need to adjust the ARIMA model to include some MA terms to include this serial correlation in our model.

The histograms of the residual look like normal distribution for both models, so our prediction interval would be correct if we modify the model and remove the serial correlation in the residual

f) Use the random walk models and chosen ARIMA model to forecast the next month and evaluate the forecast accuracy of these models.

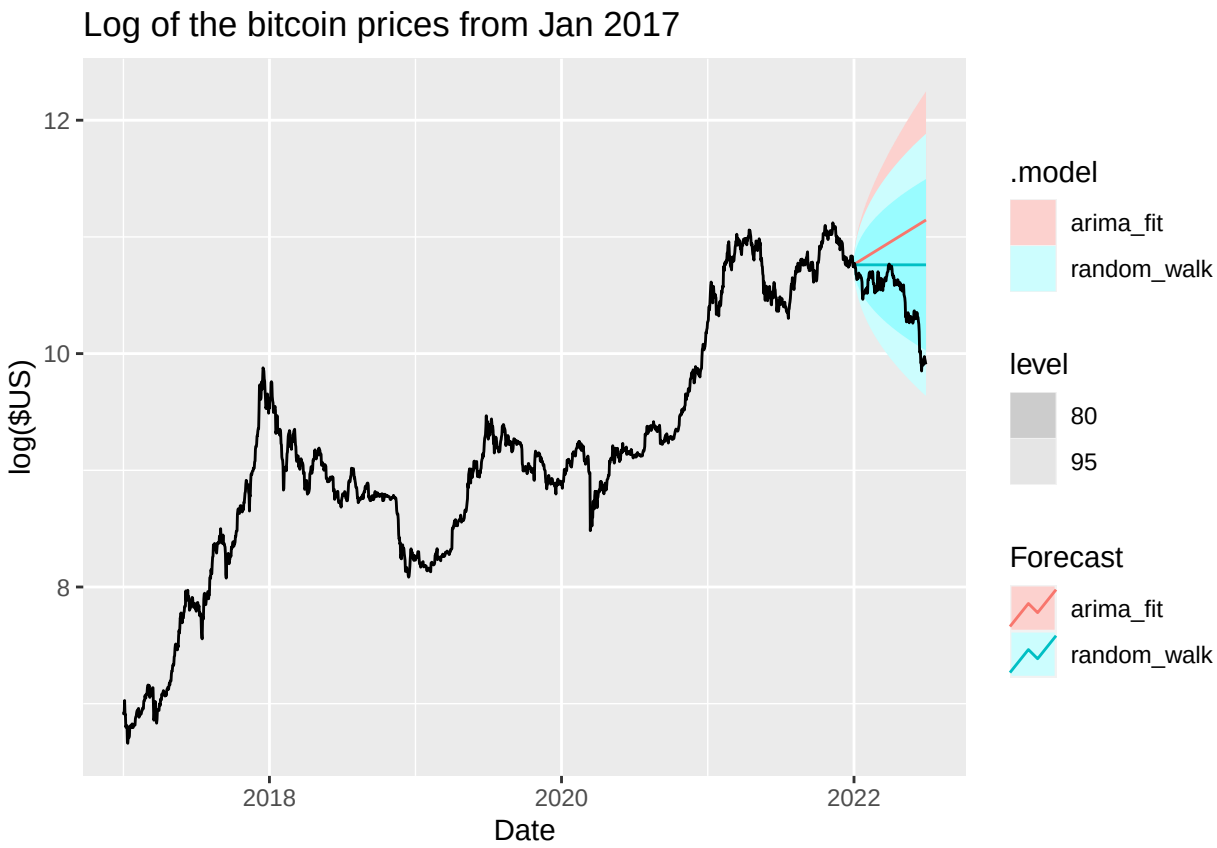
- Recall that the two most commonly used measures of forecast accuracy are MAE(Mean absolute error) and RMSE(Root mean squared error) given by:

$$MAE = mean(|e_t|)$$

$$RMSE = \sqrt{mean(e_t^2)}$$

-Where e_t is forecast "error."

```
bit_forcast <- bit_fit %>%
  forecast(bitcoin_test)

bitcoin_daily <- bitcoin_daily %>%
  mutate(log_Value = log(Value))

bit_forcast %>%
  filter(.model %in% c("random_walk", "arima_fit")) %>%
  autoplot(bitcoin_daily)+
  labs(y = "log($US)",title = "Log of the bitcoin prices from Jan 2017") +
  guides(colour = guide_legend(title = "Forecast"))
```




```
accuracy(bit_forcast, bitcoin_daily)
```

```
## # A tibble: 3 x 10
##   .model        .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>         <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 arima_fit     Test  -0.459 0.559 0.459 -4.44  4.44  5.46  4.88 0.971
## 2 random_walk   Test  -0.265 0.349 0.265 -2.58  2.58  3.16  3.04 0.964
## 3 random_walk_1 Test  -0.265 0.349 0.265 -2.58  2.58  3.16  3.04 0.964
```

The random walk forecast are constant equal to the last price in the training set, and the prediction interval increase with the forecast's horizon.

The forecasts of the selected model by ARIMA() show an increasing trend mimicking the general upward trend in the data.

Finally, we use the accuracy() to compare their prediction accuracy, and both RMSE(Root mean squared error) and MAE(Mean absolute error) are smaller for the random walk process.

So, based on our findings, we could not reject the random walk hypothesis for daily bitcoin prices, hence validating the efficiency of bitcoin prices.

Reminders

1. Complete all videos and reading for unit 10