Tidy Time Series Forecasting: fable methods

Dr. Kam Tin Seong
Assoc. Professor of Information Systems

School of Computing and Information Systems,
Singapore Management University

2022-7-16 (updated: 2022-07-23)

Content

In this topic, you will learn the basic principles and methods of time series forecasting.

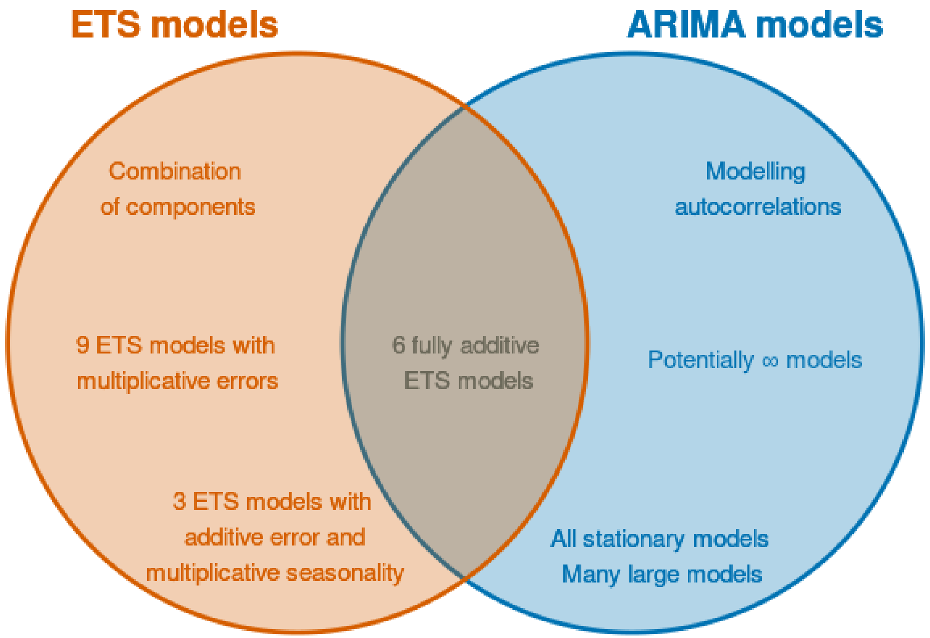
- Exponential Smoothing State Space Models (ETS)
- AutoRegressive Integrated Moving Average(ARIMA)
  - Stationary
  - Differencing

At the same time, you will gain hands-on experience on using:

- functions in **feasts** package to visualise, explore and detect time series patterns,
- functions in **fable** package fit time series forecasting model by using manual and automatic methods,
- functions in **fabletools** package to analyse the models including model comparison,
- functions in **fable** package to forecast the future values of the time series.

Getting to Know the Time Series Forecasting Methods

- Exponential Smoothing: Based on the based on a description of the *trend* and *seasonality* in the time series data.
- ARIMA: Based on the *autocorrelations* in the time series data.



Source: *"9.10 ARIMA vs ETS"* of Rob J Hyndman and George Athanasopoulos (2022) **Forecasting: Principles and Practice** (3rd ed) (online version)

Exponential Smoothing State Space Models (ETS) for Time Series Forecasting

$$Y_t = f(T_t, S_t, R_t)$$

TREND

SEASONAL

ERROR (Irregular)

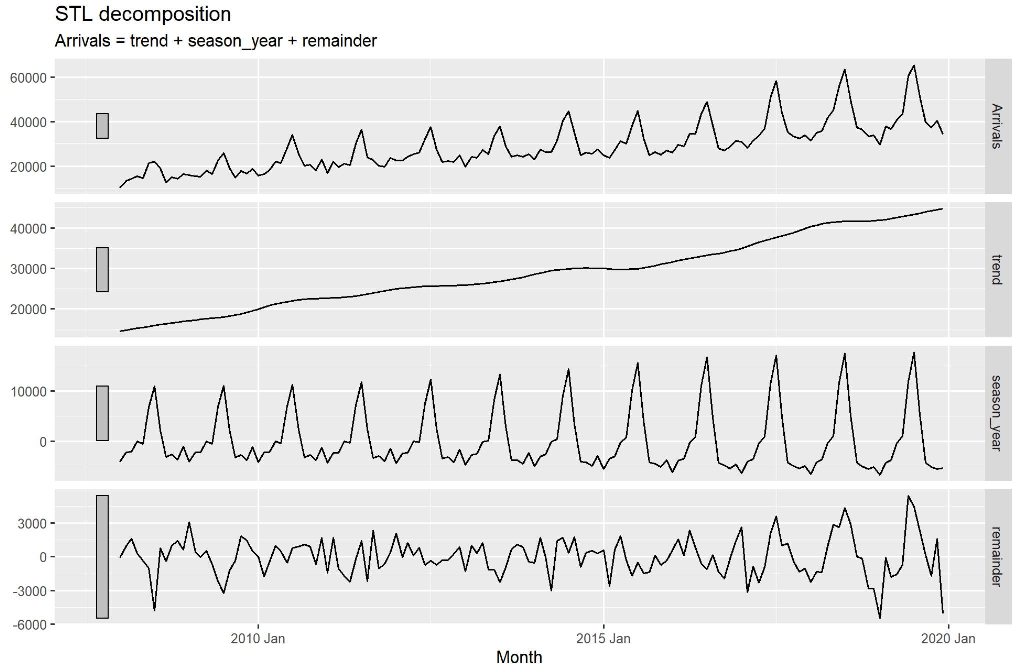
Exponential Smoothing

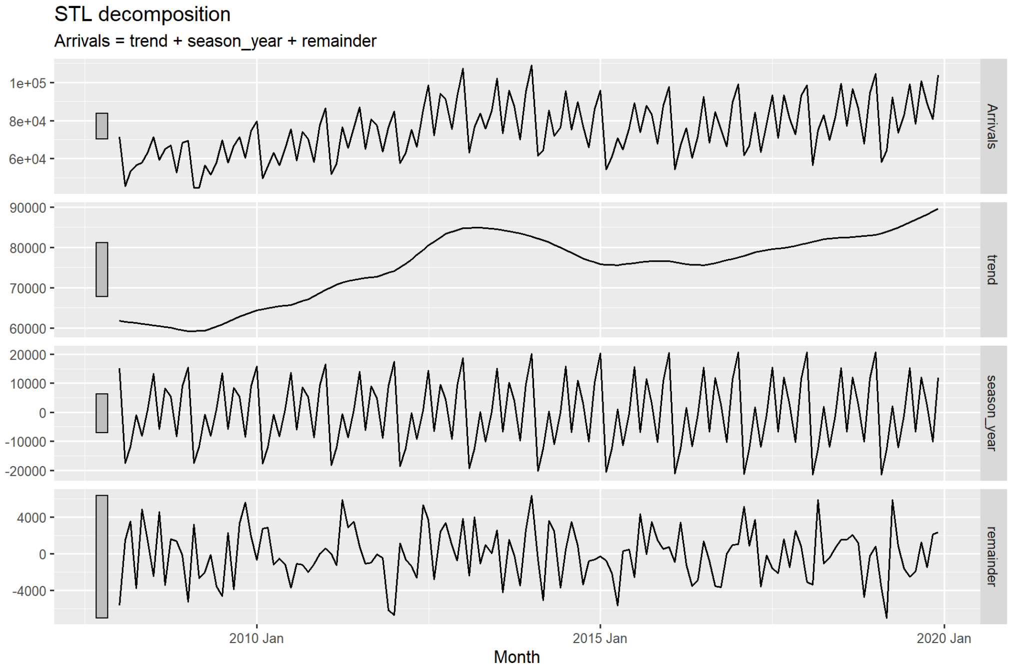
Exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get more remote. Exponential smoothing is a family of methods that vary by their trend and seasonal components.

- There can be no trend (N), an additive (A) linear trend from the forecast horizon, or a damped additive (Ad) trend leveling off from the forecast horizon. The trend could also be multiplicative (M) or multiplicative damped (Md), but Hyndman explains that they do not produced good forecasts.
- There can be no seasonality (N), or it can be additive (A) change, or multiplicative (M) (proportional) change. Apparently seasonality does not have an additive damped version.

	Seasonal Component		
Trend Component	**None (N)**	**Additive (A)**	**Multiplicative (M)**
None (N)	(N, N) Simple Exponential Smoothing	(N, A)	(N, M)
Additive (A)	(A, N) Holt's linear method	(A, A) Additive Holt-Winters' method	(A, M) Multiplicative Holt-Winters' method
Additive Damped (A_d)	(A_d, N) Additive damped trend method	(A_d, A)	(A_d, M) Holt-Winters' damped method

EDA methods for detecting trend, seasonality and errors: STL





AutoRegressive Integrated Moving Average(ARIMA) Methods for Time Series Forecasting

AutoRegressive Integrated Moving Average (ARIMA) models: An overview

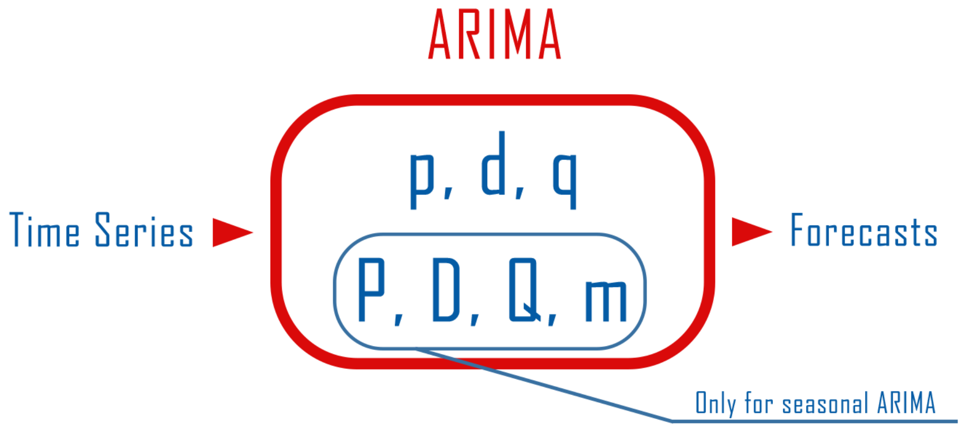
ARIMA models are used to forecast the observation at (t+1) based on the historical data of previous time spots recorded for the same observation.

The key aspects of ARIMA model are the following:

- AR: Autoregression. This indicates that the time series is regressed on its own lagged values.
- I: Integrated: This indicates that the data values have been replaced with the difference between their values and the previous values in order to convert the series into stationary.
- MA: Moving Average. This indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past.

ARIMA models: Basic Principles

The ARIMA model can be applied when we have seasonal or non-seasonal data. The difference is that when we have seasonal data we need to add some more parameters to the model.



For non-seasonal data the parameters are:

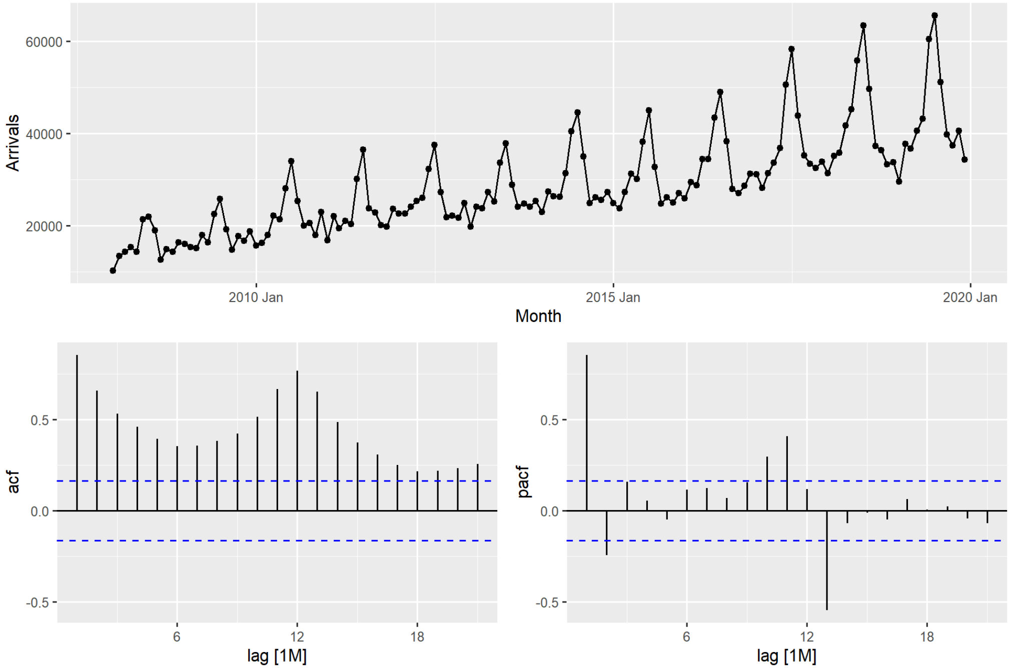
- p: The number of lag observations the model will use
- d: The number of times that the raw observations are differenced till stationarity.
- q: The size of the moving average window.

For seasonal data we need to add also the following:

- P: The number of seasonal lag observations the model will use
- D: The number of times that the seasonal observations are differenced till stationarity.
- Q: The size of the seasonal moving average window.
- m: The number of observations of 1 season

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

ACF and PACF plots are the two most used methods for choosing the best q and p values from them.

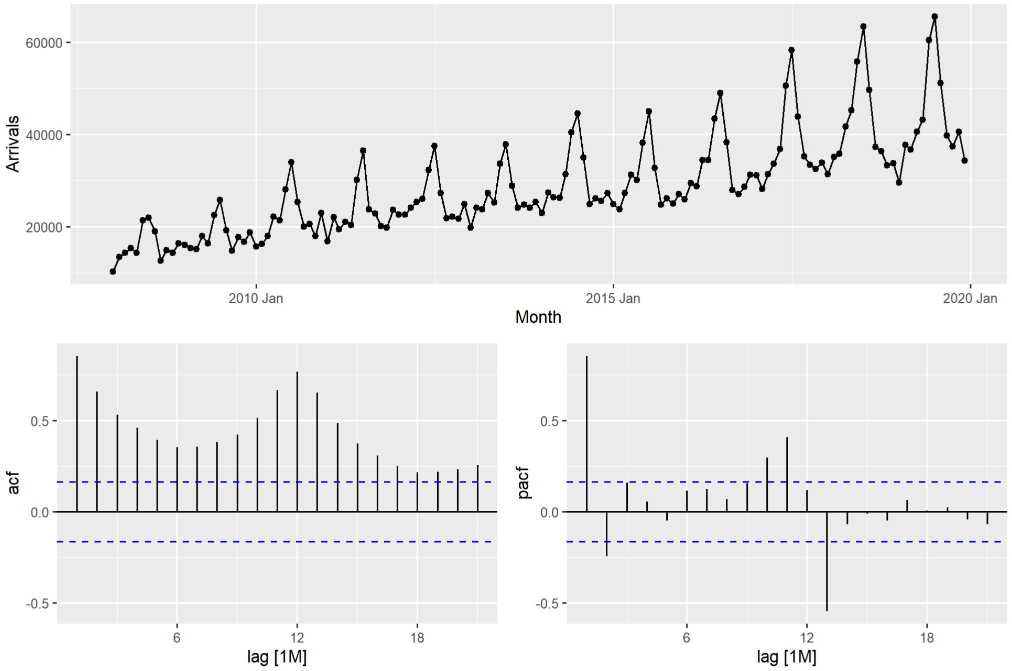


The ACF plots the correlation coefficient against the lag, which is measured in terms of a number of periods or units. A lag corresponds to a certain point in time after which we observe the first value in the time series.

PACF is a statistical measure that captures the correlation between two variables after controlling for the effects of other variables. In short, a PACF captures a “direct” correlation between time series and a lagged version of itself.

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

ACF and PACF plots are the two most used methods for choosing the best q and p values from them.

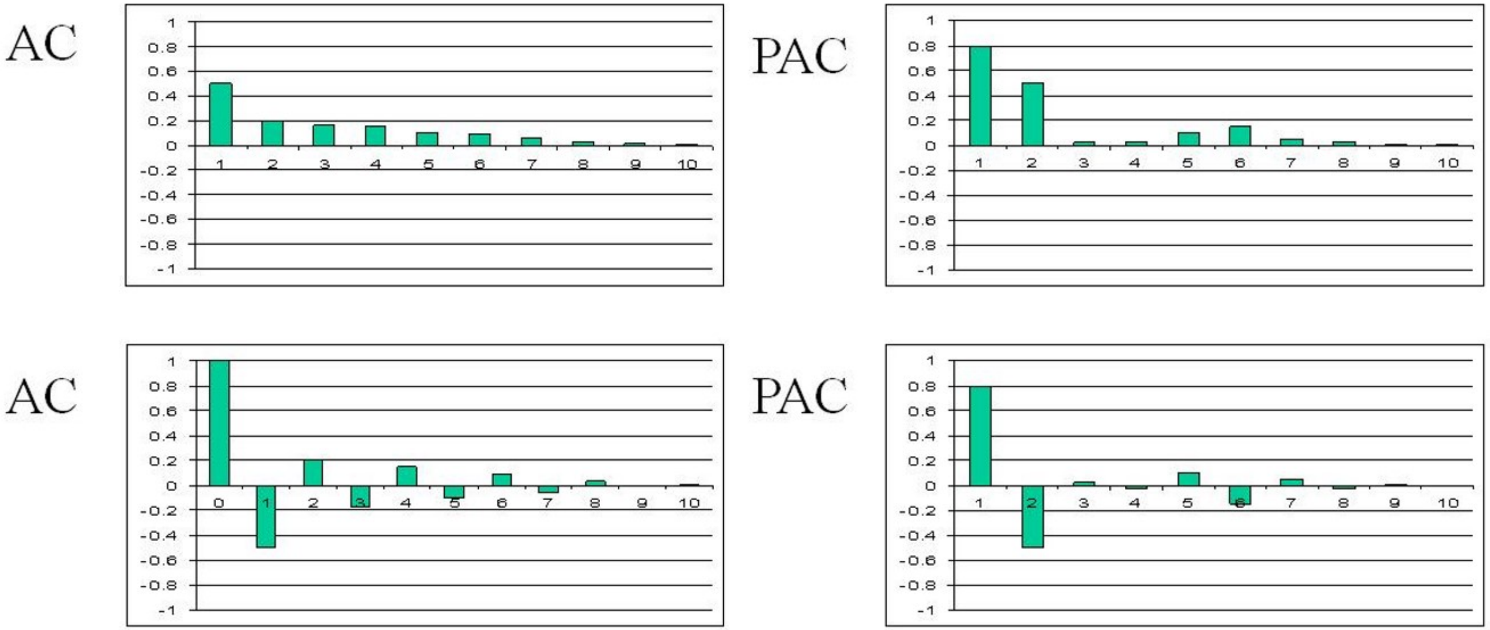


- When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in value. So the ACF of a trended time series tends to have positive values that slowly decrease as the lags increase.
- When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal period) than for other lags.
- When data are both trended and seasonal, you see a combination of these effects. The visitor from Vietnam data plotted on left shows both trend and seasonality. The slow decrease in the ACF as the lags increase is due to the trend, while the “scalloped” shape is due to the seasonality.

AR Model Fit

$$y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \cdots + \emptyset_p y_{t-p} + e_t$$

where e_t is white noise

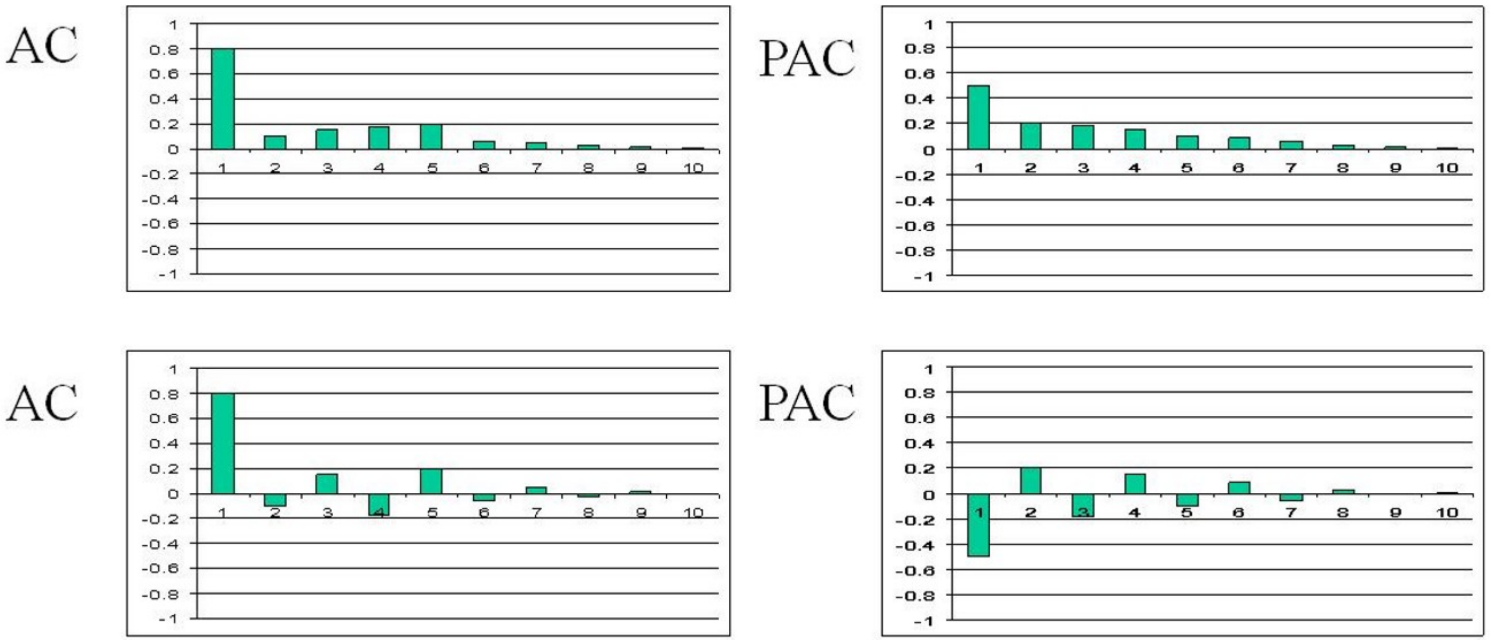


- When the autocorrelation coefficients gradually fall to 0, and the partial correlation has spikes, an AR model is appropriate. The order of the model depends on the number of spikes.
- Figure above reveals an *AR(2)* model.

MA Model Fit

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

where e_t is white noise

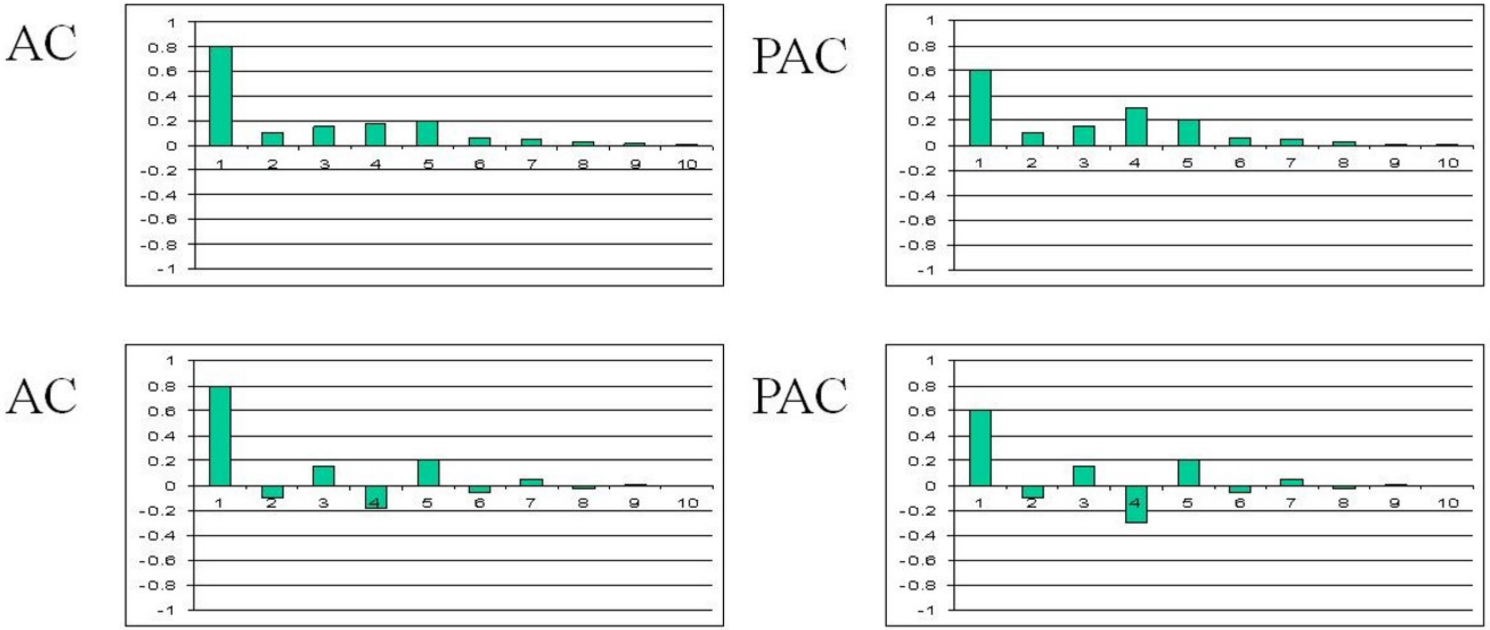


- When the partial correlation coefficients gradually fall to 0, and the autocorrelation has spikes, a MA model is appropriate. The order of the model depends on the number of spikes.
- Figure above shows a *MA(1)* model.

ARIMA Model Fit

$$\boldsymbol{y'_t = c + \emptyset_1 y'_{t-1} + \emptyset_2 y'_{t-2} + \cdots + \emptyset_p y'_{t-p} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} + e_t}$$

where y'_t is the differenced series p = order of the autoregressive part; d = degree of first differencing involved; q = order of the moving average part.

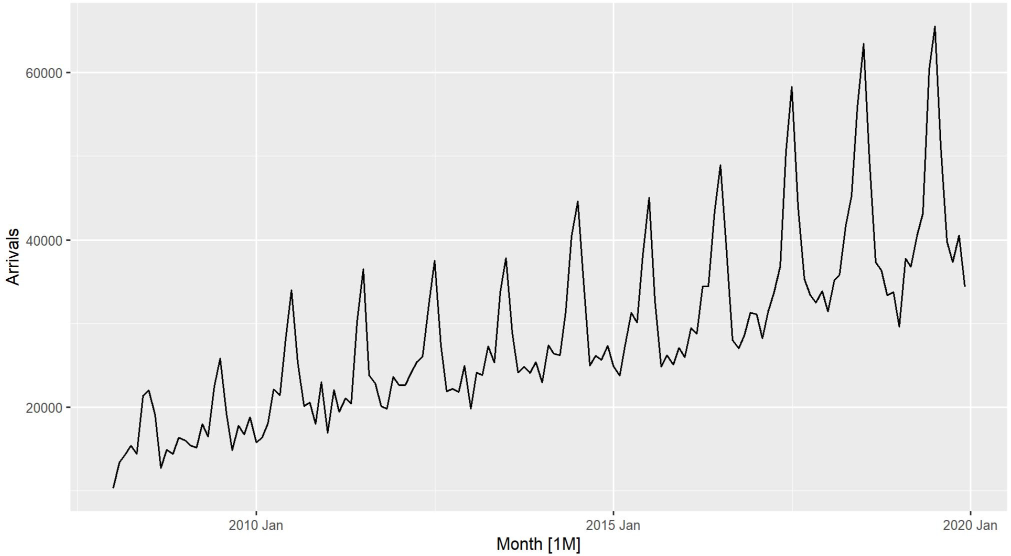


- When both the autocorrelation and the partial correlograms show irregular patterns, then an ARIMA model is appropriate. The order of the model depends on the number of spikes.
- Figure above reveals an *ARIMA(1,0,1)*.

What is Stationary in Time Series?

A **stationary** time series is one whose statistical properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary. On the other hand, a white noise series is stationary.

Figure on the right shows that the monthly visitor arrivals from Vietnam by air are non-stationary.



In general, it is necessary to make sure that the time series is stationary over the historical data of observation overtime period. If the time series is not stationary then we could apply the **differencing** factor on the records and see if the graph of the time series is a stationary overtime period.

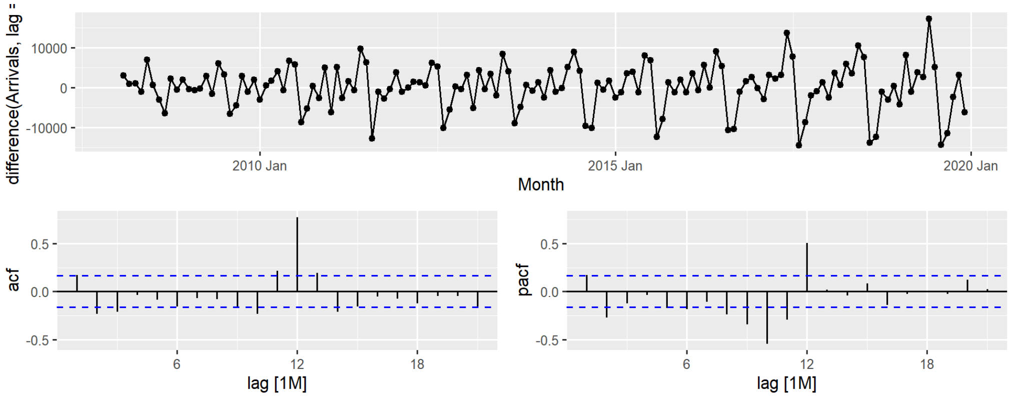
What is differencing?

On way to make a time series non-stationary is to compute the differences between consecutive observations. This is known as **differencing**. Differencing can help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

Differencing methods

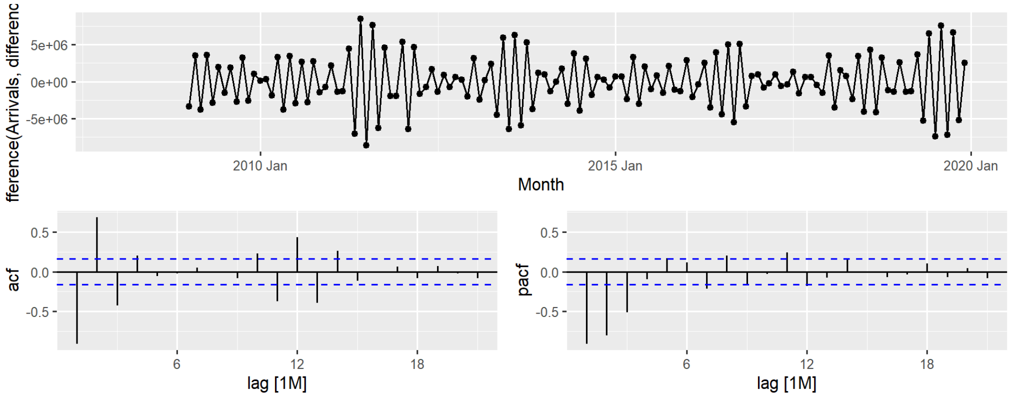
Trend differencing

$$d_t = Y_t - Y_{t-1}$$



Seasonal differencing

$$d_t = Y_t - Y_{t-12}$$



The PACF is suggestive of an AR(1) model; so an initial candidate model is an ARIMA(1,1,0). The ACF suggests an MA(1) model; so an alternative candidate is an ARIMA(0,1,1).

Where are the equivalence relationships between ETS and ARIMA models?

Table 9.4: Equivalence relationships between ETS and ARIMA models.

ETS model	ARIMA model	Parameters
ETS(A,N,N)	ARIMA(0,1,1)	$\theta_1 = \alpha - 1$
ETS(A,A,N)	ARIMA(0,2,2)	$\theta_1 = \alpha + \beta - 2$ $\theta_2 = 1 - \alpha$
ETS(A,A$_d$,N)	ARIMA(1,1,2)	$\phi_1 = \phi$ $\theta_1 = \alpha + \phi\beta - 1 - \phi$ $\theta_2 = (1 - \alpha)\phi$
ETS(A,N,A)	ARIMA(0,1,m)(0,1,0)$_m$	
ETS(A,A,A)	ARIMA(0,1,$m + 1$)(0,1,0)$_m$	
ETS(A,A$_d$,A)	ARIMA(1,0,$m + 1$)(0,1,0)$_m$	

Source: *"9.10 ARIMA vs ETS"* of Rob J Hyndman and George Athanasopoulos (2022) **Forecasting: Principles and Practice** (3rd ed) (online version)

Setting Up R Environment

For the purpose of this hands-on exercise, the following R packages will be used.

```
pacman::p_load(tidyverse, lubridate, zoo,
               seasonal, tsibble, feasts,
               fable)
```

Two R packages are added in the list, they are:

- **fable** provides a collection of commonly used univariate and multivariate time series forecasting models including exponential smoothing via state space models and automatic ARIMA modelling. It is a tidy version of the popular **forecast** package and a member of **tidyverts**.
- **fabletools** provides tools for building modelling packages, with a focus on time series forecasting. This package allows package developers to extend fable with additional models, without needing to depend on the models supported by fable.

Importing the data

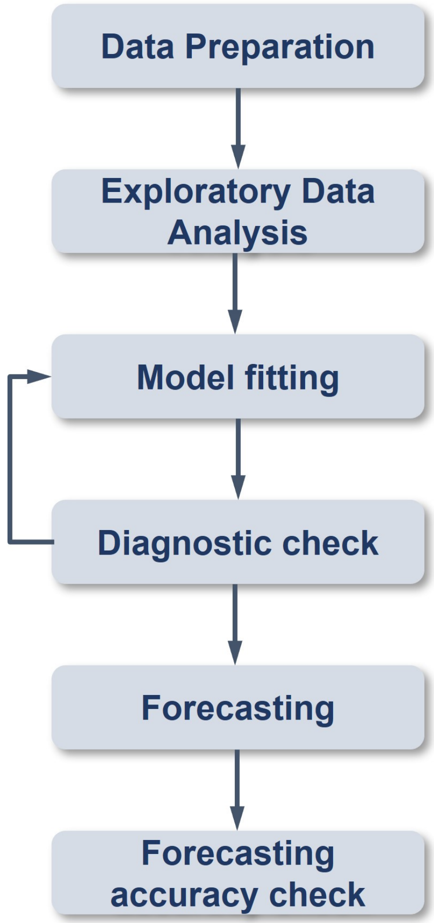
For the purpose of this hands-on session, visitor arrivals data to Singapore between from Jan 2008 until Dec 2019 will be used.

- *tsbl_longer*: Visitor arrivals data set in tsibble data frame format. It is in rds format.

In the code chunk below, `read_rds()` of **readr** package is used to import both data sets into R environment.

```
tsbl_longer <- read_rds(
  "data/tsbl_longer.rds")
```

Time Series Forecasting Processes



Time Series Forecasting: fable (tidyverts) methods

Time Series Data Sampling

In forecasting, we will split the time series into two parts. The first part is called **estimate sample** (also known as training data). It will be used to estimate the starting values and the smoothing parameters. This sample typically contains 75-80 percent of the observation, although the forecaster may choose to use a smaller percentage for longer series.
The second part of the time series is called **hold-out sample**. It is used to check the forecasting performance. No matter how many parameters are estimated with the estimation sample, each method under consideration can be evaluated with the use of the “new” observation contained in the hold-out sample.

Step 1: Time Series Data Sampling

A good practice in forecasting is to split the original data set into a training and a hold-out data sets. In this example we will use the last 12 months for hold-out and the rest for training.

First, an extra column called *Type* indicating training or hold-out will be created by using `mutate()` of **dplyr** package. It will be extremely useful for subsequent data visualisation.

```
vietnam_ts <- tsbl_longer %>%
  filter(Country == "Vietnam") %>%
  mutate(Type = if_else(
    `Month-Year` >= "2019-01-01",
    "Hold-out", "Training"))
```

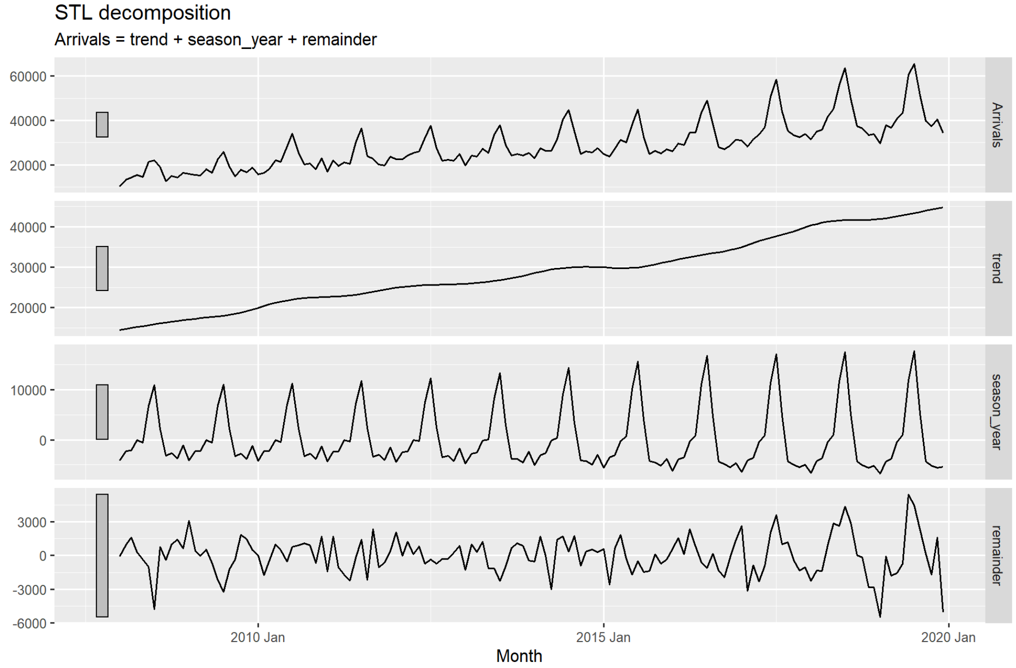
Next, a training data set is extracted from the original data set by using `filter()` of **dplyr** package.

```
vietnam_train <- vietnam_ts %>%
  filter(`Month-Year` < "2019-01-01")
```

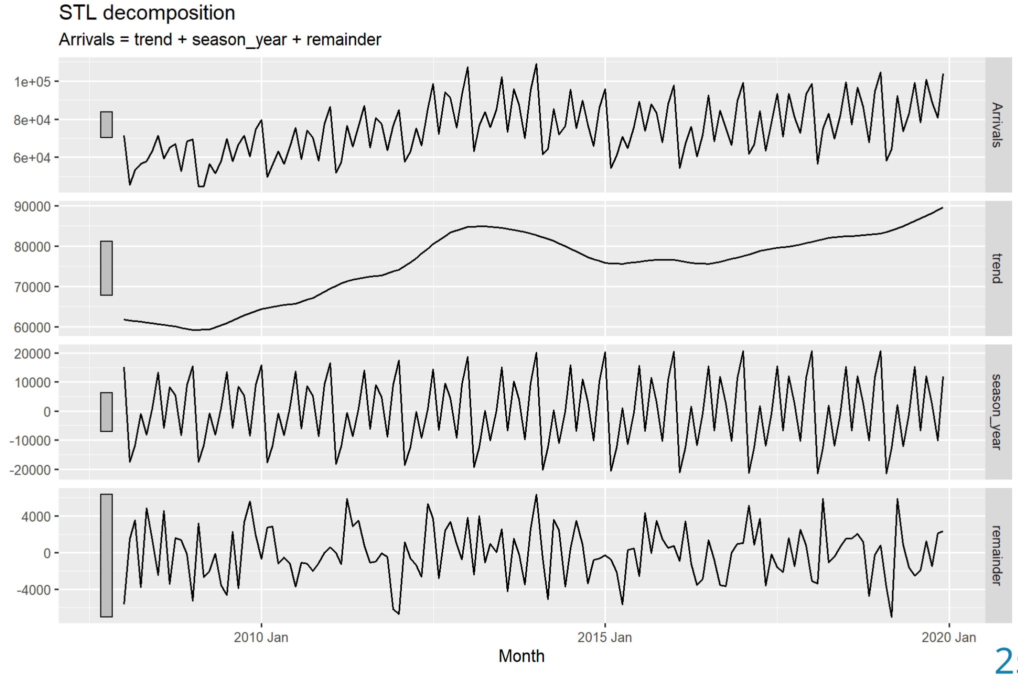
Step 2: Exploratory Data Analysis (EDA): Time Series Data

Before fitting forecasting models, it is a good practice to analysis the time series data by using EDA methods.

```
tsbl_longer %>%
  filter(`Country` == "Vietnam") %>%
  model(stl = STL(Arrivals)) %>%
  components() %>%
  autoplot()
```




```
tsbl_longer %>%
  filter(`Country` == "Australia") %>%
  model(stl = STL(Arrivals)) %>%
  components() %>%
  autoplot()
```



Step 3: Fitting Exponential Smoothing State Space (ETS) Models: fable methods

In fable, Exponential Smoothing State Space Models are supported by `ETS()`. The combinations are specified through the formula:

```
ETS(y ~ error(c("A", "M"))
    + trend(c("N", "A", "Ad"))
    + season(c("N", "A", "M"))
```

Fitting a simple exponential smoothing (SES)

```
fit_ses <- vietnam_train %>%
  model(ETS(Arrivals ~ error("A")
            + trend("N")
            + season("N")))
fit_ses
```

```
## # A mable: 1 x 2
## # Key:     Country [1]
##   Country `ETS(Arrivals ~ error("A") + trend("N") + season("N"))`
##   <chr>                                                   <model>
## 1 Vietnam                                             <ETS(A,N,N)>
```

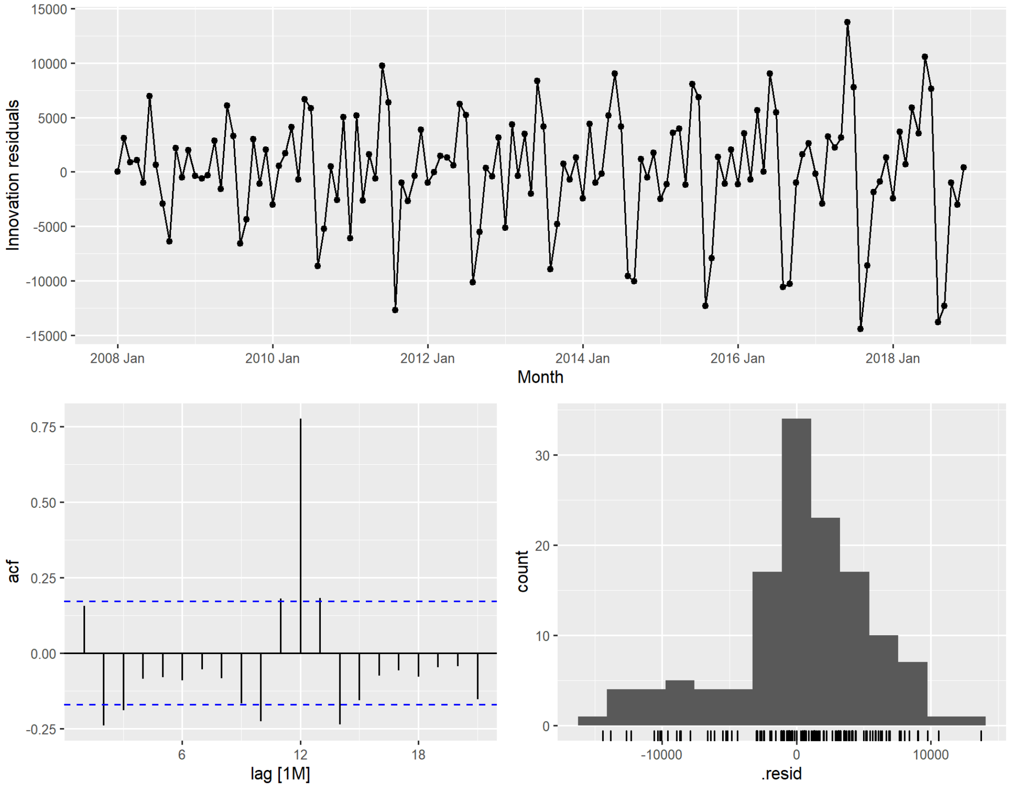
Notice that `model()` of fable package is used to estimate the ETS model on a particular dataset, and returns a **mable** (model table) object.

A mable contains a row for each time series (uniquely identified by the key variables), and a column for each model specification. A model is contained within the cells of each model column.

Examine Model Assumptions

Next, `gg_tsresiduals()` of **feasts** package is used to check the model assumptions with residuals plots.

```
gg_tsresiduals(fit_ses)
```



The model details

`report()` of fabletools is be used to reveal the model details.

```
fit_ses %>%
  report()
```

```
## Series: Arrivals
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.9998995
##
##   Initial states:
##      l[0]
##  10312.99
##
##   sigma^2:  27939164
##
##      AIC     AICc      BIC
## 2911.726 2911.913 2920.374
```

Fitting ETS Methods with Trend: Holt's Linear

Trend methods

```
vietnam_H <- vietnam_train %>%
  model(`Holt's method` =
          ETS(Arrivals ~ error("A") +
                trend("A") +
                season("N")))
vietnam_H %>% report()
```

```
## Series: Arrivals
## Model: ETS(A,A,N)
##   Smoothing parameters:
##     alpha = 0.9998995
##     beta  = 0.0001004625
##
##   Initial states:
##      l[0]     b[0]
##  13673.29 525.8859
##
##   sigma^2:  28584805
##
##      AIC     AICc      BIC
## 2916.695 2917.171 2931.109
```

Damped Trend methods

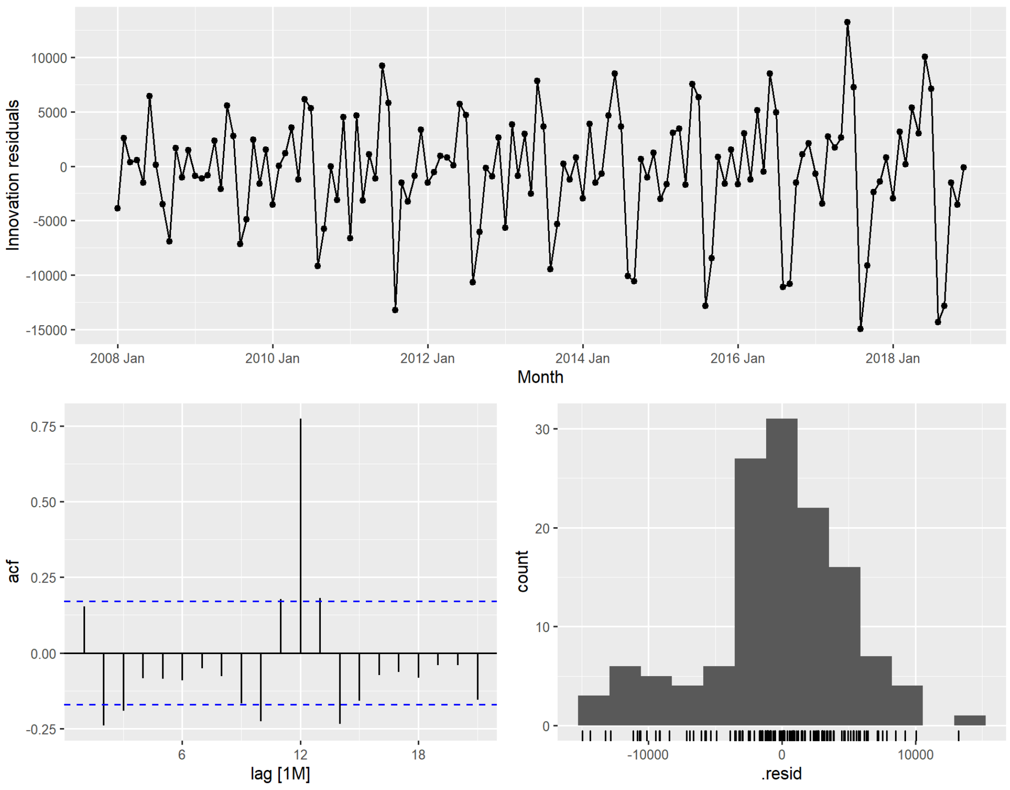
```
vietnam_HAd <- vietnam_train %>%
  model(`Holt's method` =
          ETS(Arrivals ~ error("A") +
                trend("Ad") +
                season("N")))
vietnam_HAd %>% report()
```

```
## Series: Arrivals
## Model: ETS(A,Ad,N)
##   Smoothing parameters:
##     alpha = 0.9998999
##     beta  = 0.0001098602
##     phi   = 0.9784562
##
##   Initial states:
##      l[0]   b[0]
##  13257.28 523.54
##
##   sigma^2:  28641536
##
##      AIC     AICc      BIC
```

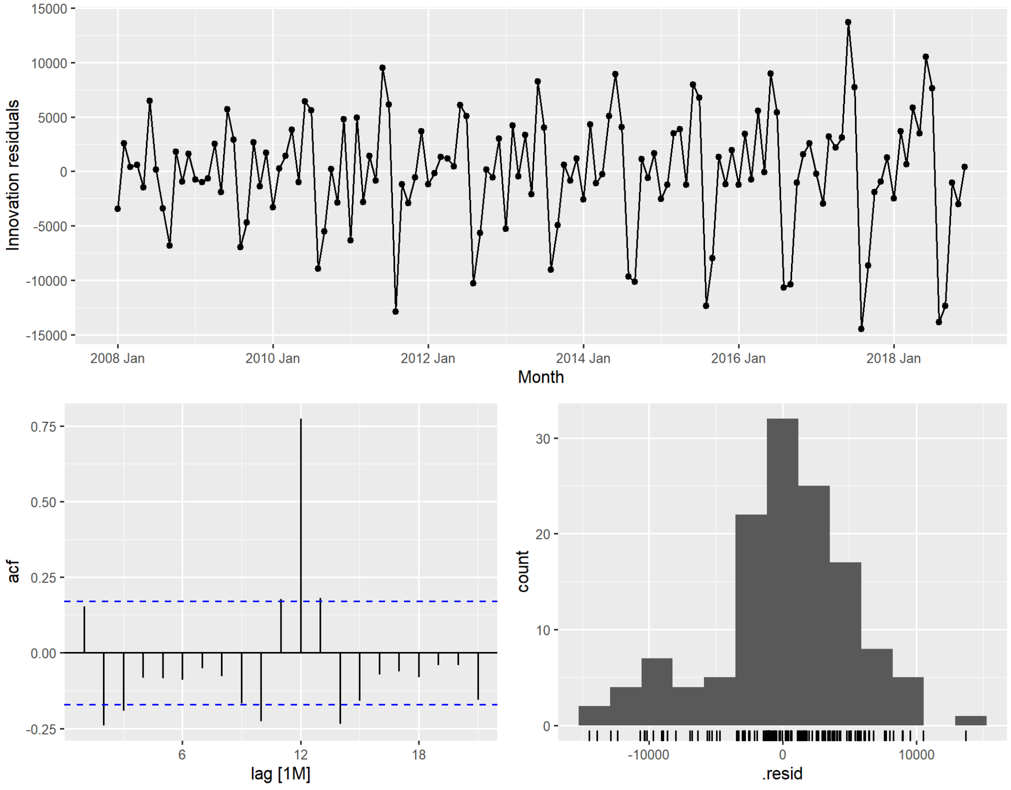
Checking for results

Check the model assumptions with residuals plots.

```
gg_tsresiduals(vietnam_H)
```




```
gg_tsresiduals(vietnam_HAd)
```



Fitting ETS Methods with Season: Holt-Winters

```
Vietnam_WH <- vietnam_train %>%
  model(
    Additive = ETS(Arrivals ~ error("A")
                  + trend("A")
                  + season("A")),
    Multiplicative = ETS(Arrivals ~ error("M")
                        + trend("A")
                        + season("M"))
    )

Vietnam_WH %>% report()
```

```
## # A tibble: 2 × 10
##   Country .model          sigma2 log_lik   AIC  AICc   BIC    MSE   AMSE     MAE
##   <chr>   <chr>            <dbl>   <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>   <dbl>
## 1 Vietnam Additive       5.33e+6  -1336. 2706. 2711. 2755. 4.68e6 8.56e6 1.72e+3
## 2 Vietnam Multiplicative 4.55e-3  -1300. 2635. 2640. 2684. 3.05e6 3.42e6 5.20e-2
```

Fitting multiple ETS Models

```
fit_ETS <- vietnam_train %>%
  model(`SES` = ETS(Arrivals ~ error("A") +
                    trend("N") +
                    season("N")),
        `Holt`= ETS(Arrivals ~ error("A") +
                    trend("A") +
                    season("N")),
        `damped Holt` =
          ETS(Arrivals ~ error("A") +
                trend("Ad") +
                season("N")),
        `WH_A` = ETS(
          Arrivals ~ error("A") +
            trend("A") +
            season("A")),
        `WH_M` = ETS(Arrivals ~ error("M")
                     + trend("A")
                     + season("M"))
  )
```

The model coefficient

`tidy()` of fabletools is be used to extract model coefficients from a mable.

```
fit_ETS %>%
  tidy()
```

```
## # A tibble: 45 × 4
##    Country .model      term      estimate
##    <chr>   <chr>       <chr>        <dbl>
##  1 Vietnam SES         alpha     1.00
##  2 Vietnam SES         l[0]  10313.
##  3 Vietnam Holt        alpha     1.00
##  4 Vietnam Holt        beta      0.000100
##  5 Vietnam Holt        l[0]  13673.
##  6 Vietnam Holt        b[0]    526.
##  7 Vietnam damped Holt alpha     1.00
##  8 Vietnam damped Holt beta      0.000110
##  9 Vietnam damped Holt phi       0.978
## 10 Vietnam damped Holt l[0]  13257.
## # … with 35 more rows
```

Step 4: Model Comparison

`glance()` of fabletool

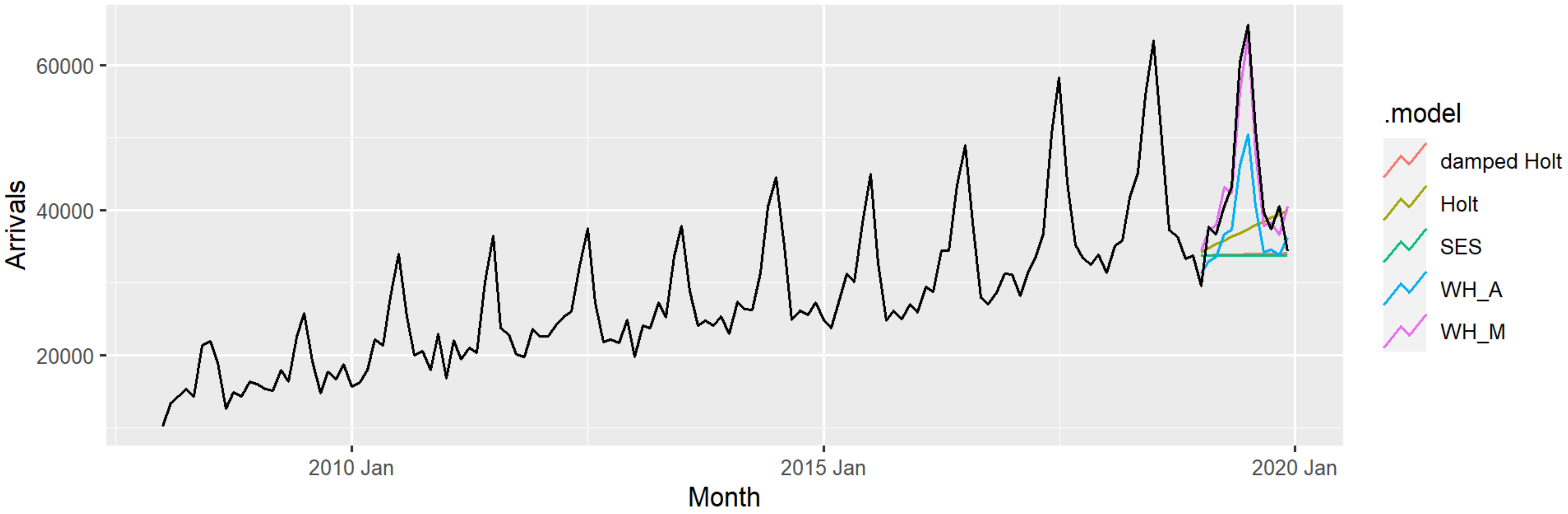
```
fit_ETS %>%
  report()
```

```
## # A tibble: 5 × 10
##   Country .model       sigma2 log_lik   AIC  AICc   BIC       MSE   AMSE     MAE
##   <chr>   <chr>         <dbl>   <dbl> <dbl> <dbl> <dbl>     <dbl>  <dbl>   <dbl>
## 1 Vietnam SES         2.79e+7  -1453. 2912. 2912. 2920. 27515844. 5.99e7 3.91e+3
## 2 Vietnam Holt        2.86e+7  -1453. 2917. 2917. 2931. 27718599. 6.03e7 3.92e+3
## 3 Vietnam damped Holt 2.86e+7  -1453. 2918. 2919. 2935. 27556629. 5.97e7 3.92e+3
## 4 Vietnam WH_A        5.33e+6  -1336. 2706. 2711. 2755.  4684271. 8.56e6 1.72e+3
## 5 Vietnam WH_M        4.55e-3  -1300. 2635. 2640. 2684.  3046059. 3.42e6 5.20e-2
```

Step 5: Forecasting future values

To forecast the future values, `forecast()` of fable will be used. Notice that the forecast period is 12 months.

```
fit_ETS %>%
  forecast(h = "12 months") %>%
  autoplot(vietnam_ts,
           level = NULL)
```



Fitting ETS Automatically

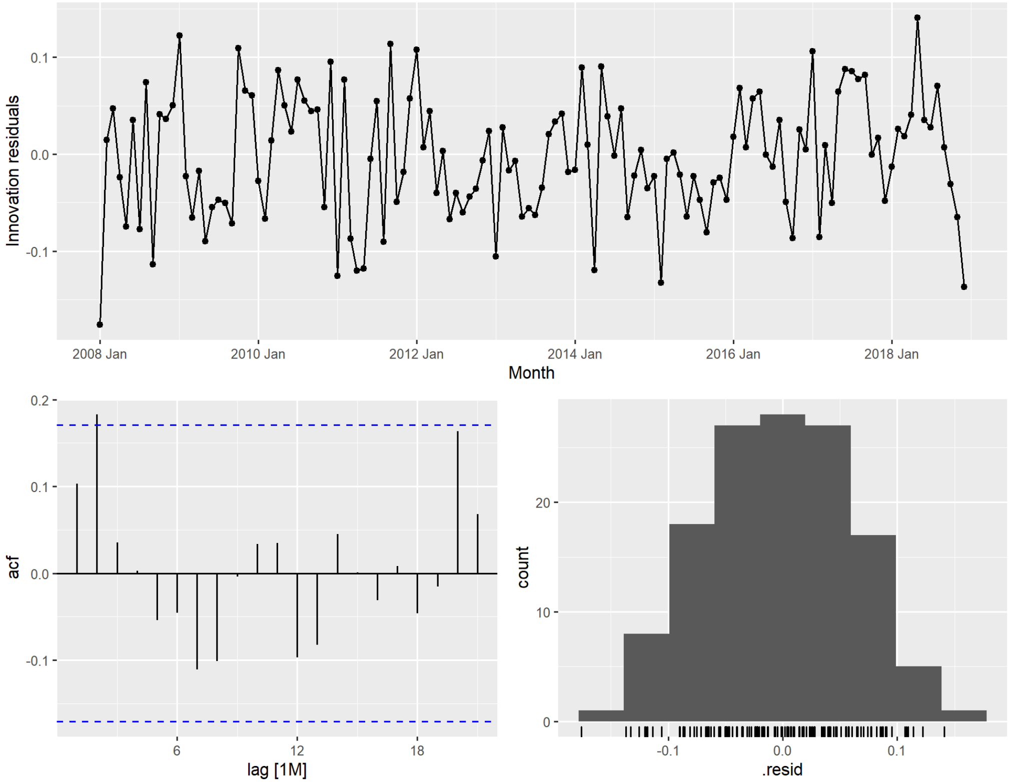
```
fit_autoETS <- vietnam_train %>%
  model(ETS(Arrivals))
fit_autoETS %>% report()
```

```
## Series: Arrivals
## Model: ETS(M,A,M)
##   Smoothing parameters:
##     alpha = 0.1613503
##     beta  = 0.0001021811
##     gamma = 0.0001030996
##
##   Initial states:
##      l[0]     b[0]      s[0]     s[-1]     s[-2]     s[-3]    s[-4]    s[-5]
##  15001.12 212.3552 0.9167302 0.8311728 0.8739287 0.8690543 1.104668 1.485584
##     s[-6]     s[-7]    s[-8]     s[-9]    s[-10]    s[-11]
##  1.311207 0.9917759 1.014187 0.8973028 0.8816768 0.8227129
##
##   sigma^2:  0.0046
##
##      AIC     AICc      BIC
## 2634.751 2640.119 2683.759
```

Fitting Fitting ETS Automatically

Next, we will check the model assumptions with residuals plots by using `gg_tsresiduals()` of **feasts** package

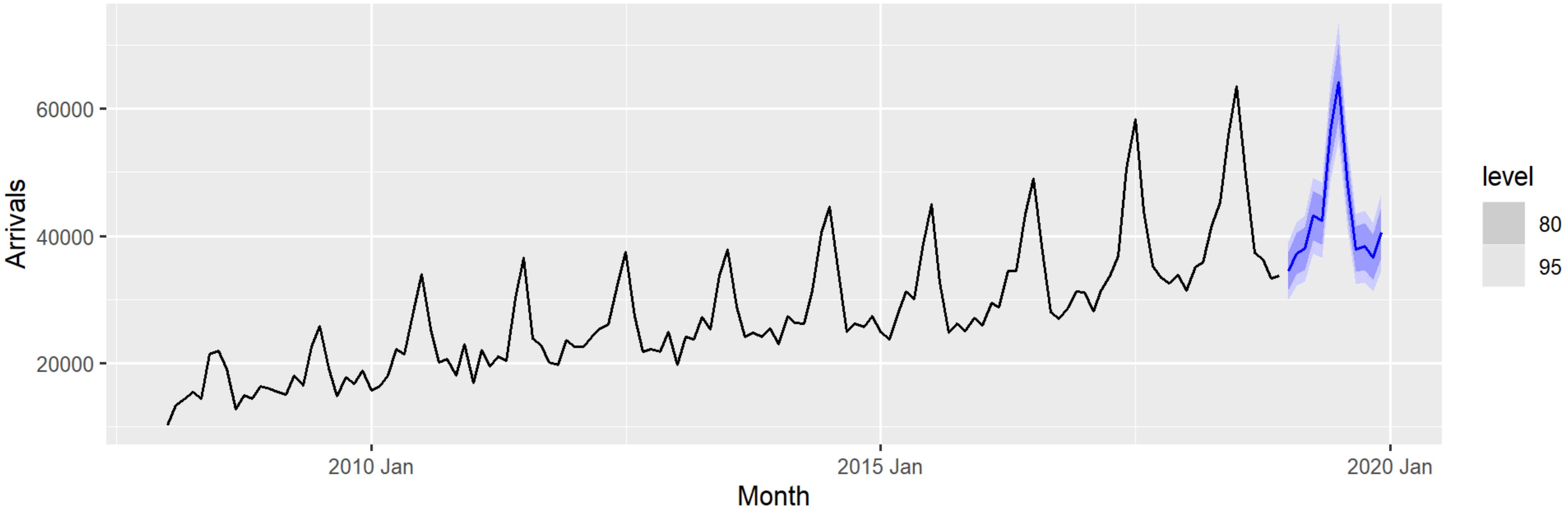
```
gg_tsresiduals(fit_autoETS)
```



Forecast the future values

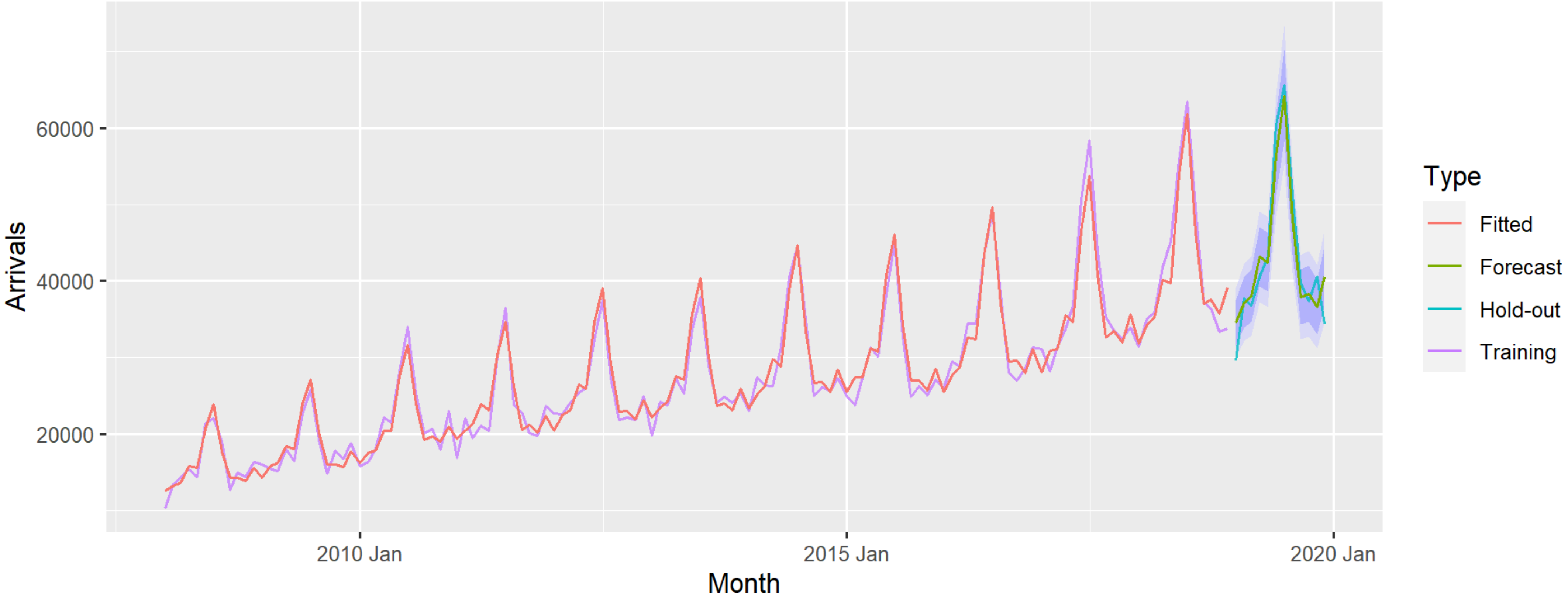
In the code chunk below, `forecast()` of **fable** package is used to forecast the future values. Then, `autoplot()` of **feasts** package is used to see the training data along with the forecast values.

```
fit_autoETS %>%
  forecast(h = "12 months") %>%
  autoplot(vietnam_train)
```



Visualising AutoETS model with ggplot2

There are time that we are interested to visualise relationship between training data and fit data and forecasted values versus the hold-out data.



Visualising AutoETS model with ggplot2

Code chunk below is used to create the data visualisation in previous slide.

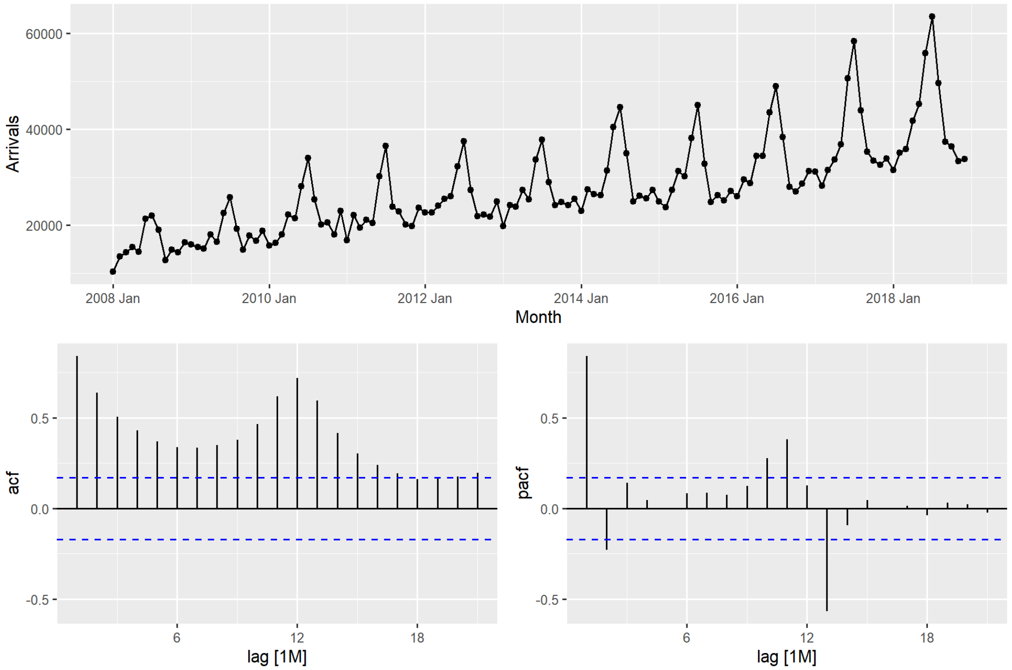
```
fc_autoETS <- fit_autoETS %>%
  forecast(h = "12 months")

vietnam_ts %>%
  ggplot(aes(x=`Month`,
             y=Arrivals)) +
  autolayer(fc_autoETS,
            alpha = 0.6) +
  geom_line(aes(
    color = Type),
    alpha = 0.8) +
  geom_line(aes(
    y = .mean,
    colour = "Forecast"),
    data = fc_autoETS) +
  geom_line(aes(
    y = .fitted,
    colour = "Fitted"),
    data = augment(fit_autoETS))
```

AutoRegressive Integrated Moving Average(ARIMA) Methods for Time Series Forecasting: fable (tidyverts) methods

Visualising Autocorrelations: feasts methods

feasts package provides a very handy function for visualising ACF and PACF of a time series called `gg_tsdiaply()`.

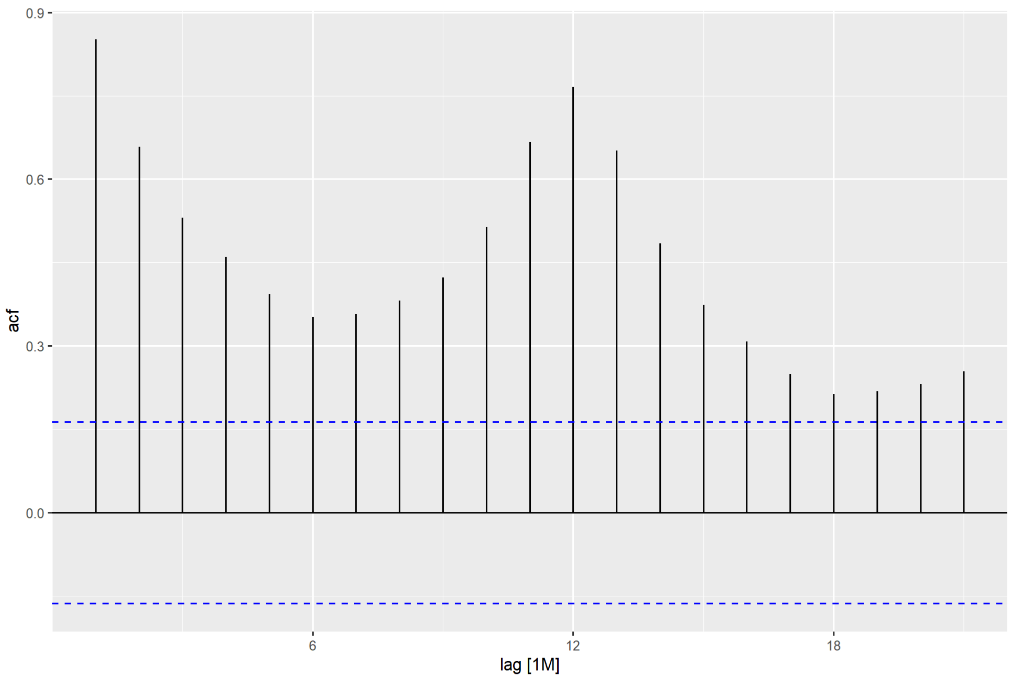
```
vietnam_train %>%
  gg_tsdisplay(plot_type='partial')
```

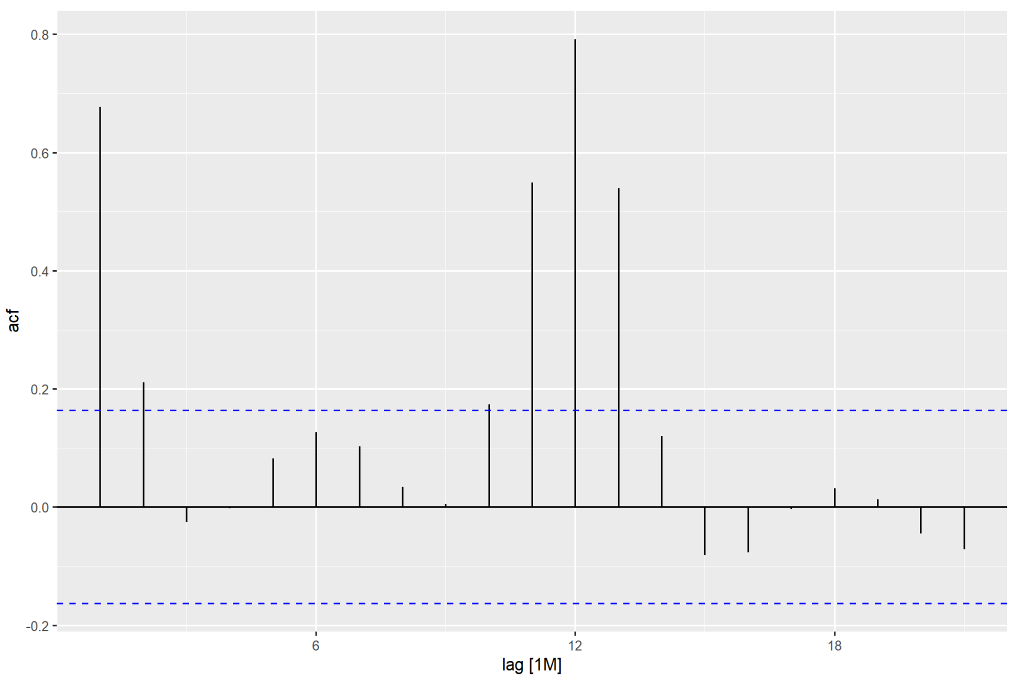


Visualising Autocorrelations: feasts methods

```
tsbl_longer %>%
  filter(`Country` == "Vietnam") %>%
  ACF(Arrivals) %>%
  autoplot()
```

```
tsbl_longer %>%
  filter(`Country` == "United Kingdom") %>%
  ACF(Arrivals) %>%
  autoplot()
```



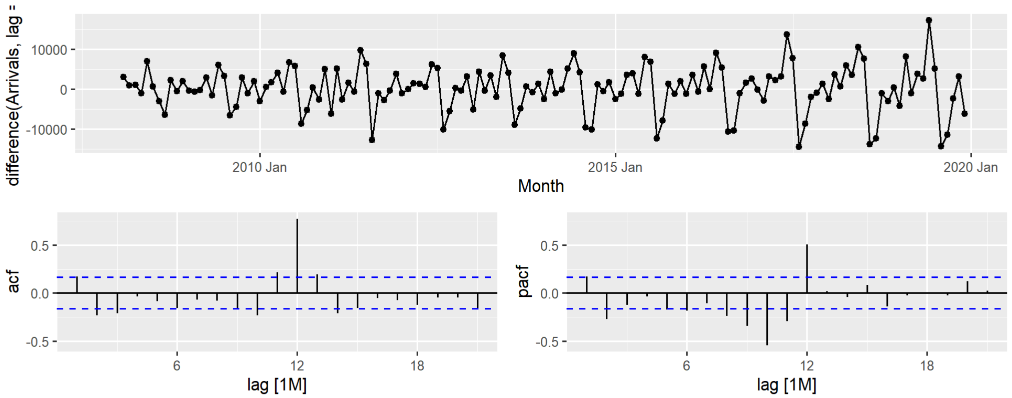


By comparing both ACF plots, it is clear that visitor arrivals from United Kingdom were very seasonal and relatively weaker trend as compare to visitor arrivals from Vietnam.

Differencing: fable methods

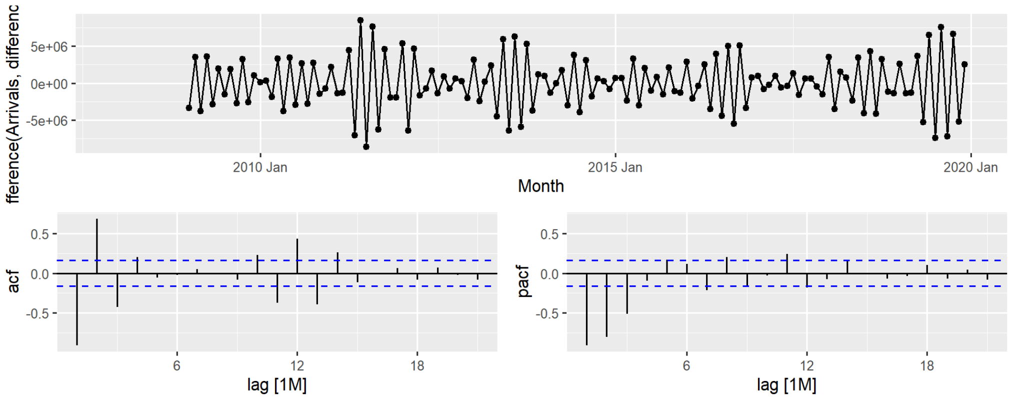
Trend differencing

```r
tsbl_longer %>%
  filter(Country == "Vietnam") %>%
  gg_tsdisplay(difference(
    Arrivals,
    lag = 1),
    plot_type='partial')
```



Seasonal differencing

```r
tsbl_longer %>%
  filter(Country == "Vietnam") %>%
  gg_tsdisplay(difference(
    Arrivals,
    difference = 12),
    plot_type='partial')
```



The PACF is suggestive of an AR(1) model; so an initial candidate model is an ARIMA(1,1,0). The ACF suggests an MA(1) model; so an alternative candidate is an ARIMA(0,1,1).

Fitting ARIMA models manually: fable methods

```
fit_arima <- vietnam_train %>%
  model(
    arima200 = ARIMA(Arrivals ~ pdq(2,0,0)),
    sarima210 = ARIMA(Arrivals ~ pdq(2,0,0) +
                        PDQ(2,1,0))
    )
report(fit_arima)
```

```
## # A tibble: 2 × 9
##   Country .model      sigma2 log_lik   AIC  AICc   BIC ar_roots    ma_roots
##   <chr>   <chr>        <dbl>   <dbl> <dbl> <dbl> <dbl> <list>      <list>
## 1 Vietnam arima200  4173906.  -1085. 2181. 2182. 2198. <cpl [26]> <cpl [0]>
## 2 Vietnam sarima210 4173906.  -1085. 2181. 2182. 2198. <cpl [26]> <cpl [0]>
```

Fitting ARIMA models automatically: fable methods

```
fit_autoARIMA <- vietnam_train %>%
  model(ARIMA(Arrivals))
report(fit_autoARIMA)
```

```
## Series: Arrivals
## Model: ARIMA(2,0,0)(2,1,0)[12] w/ drift
##
## Coefficients:
##          ar1     ar2     sar1     sar2   constant
##       0.4748  0.1892  -0.5723  -0.1578  1443.2068
## s.e.  0.0924  0.0903   0.0989   0.1032   188.9468
##
## sigma^2 estimated as 4173906:  log likelihood=-1084.6
## AIC=2181.19   AICc=2181.94   BIC=2197.92
```

Model Comparison

```
bind_rows(
    fit_autoARIMA %>% accuracy(),
    fit_autoETS %>% accuracy(),
    fit_autoARIMA %>%
      forecast(h = 12) %>%
      accuracy(vietnam_ts),
    fit_autoETS %>%
      forecast(h = 12) %>%
      accuracy(vietnam_ts)) %>%
  select(-ME, -MPE, -ACF1)
```

```
## # A tibble: 4 × 8
##   Country .model          .type     RMSE   MAE  MAPE  MASE RMSSE
##   <chr>   <chr>           <chr>    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Vietnam ARIMA(Arrivals) Training 1907. 1458.  5.37 0.491 0.517
## 2 Vietnam ETS(Arrivals)   Training 1745. 1386.  5.29 0.467 0.473
## 3 Vietnam ARIMA(Arrivals) Test     2647. 2136.  5.17 0.720 0.717
## 4 Vietnam ETS(Arrivals)   Test     3163. 2636.  6.64 0.889 0.857
```

Forecast Multiple Time Series

In this section, we will perform time series forecasting on multiple time series at one goal. For the purpose of the hand-on exercise, visitor arrivals from five selected ASEAN countries will be used.

First, `filter()` is used to extract the selected countries' data.

```
ASEAN <- tsbl_longer %>%
  filter(Country == "Vietnam" |
         Country == "Malaysia" |
         Country == "Indonesia" |
         Country == "Thailand" |
         Country == "Philippines")
```

Next, `mutate()` is used to create a new field called Type and populates their respective values. Lastly, `filter()` is used to extract the training data set and save it as a tsibble object called *ASEAN_train*.

```
ASEAN_train <- ASEAN %>%
  mutate(Type = if_else(
    `Month-Year` >= "2019-01-01",
    "Hold-out", "Training")) %>%
  filter(Type == "Training")
```

Fitting Mulltiple Time Series

In the code chunk below auto ETS and ARIMA models are fitted by using `model()`.

```
ASEAN_fit <- ASEAN_train %>%
  model(
    ets = ETS(Arrivals),
    arima = ARIMA(Arrivals)
  )
```

Examining Models

The `glance()` of **fabletools** provides a one-row summary of each model, and commonly includes descriptions of the model's fit such as the residual variance and information criteria.

```
ASEAN_fit %>%
  glance()
```

```
## # A tibble: 10 × 12
##    Country     .model  sigma2 log_lik   AIC  AICc   BIC       MSE     AMSE     MAE
##    <chr>       <chr>    <dbl>   <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>   <dbl>
##  1 Indonesia   ets     1.02e-2  -1561. 3156. 3161. 3205.   1.74e8  1.80e8  0.0732
##  2 Indonesia   arima   1.48e+8  -1290. 2589. 2590. 2603.  NA      NA      NA
##  3 Malaysia    ets     4.67e-3  -1430. 2894. 2899. 2943.   2.04e7  2.00e7  0.0506
##  4 Malaysia    arima   2.62e+7  -1185. 2378. 2379. 2390.  NA      NA      NA
##  5 Philippines ets     3.56e-3  -1343. 2722. 2728. 2774.   5.28e6  7.58e6  0.0461
##  6 Philippines arima   8.04e+6  -1122. 2260. 2262. 2283.  NA      NA      NA
##  7 Thailand    ets     6.63e-3  -1343. 2722. 2728. 2774.   5.40e6  6.33e6  0.0584
##  8 Thailand    arima   8.51e+6  -1127. 2269. 2270. 2288.  NA      NA      NA
##  9 Vietnam     ets     4.55e-3  -1300. 2635. 2640. 2684.   3.05e6  3.42e6  0.0520
## 10 Vietnam     arima   4.17e+6  -1085. 2181. 2182. 2198.  NA      NA      NA
## # … with 2 more variables: ar_roots <list>, ma_roots <list>
```

Be wary though, as information criteria (AIC, AICc, BIC) are only comparable between the same model class and only if those models share the same response (after transformations and differencing).

Extracintg fitted and residual values

The fitted values and residuals from a model can obtained using fitted() and residuals() respectively. Additionally, the augment() function may be more convenient, which provides the original data along with both fitted values and their residuals.

```
ASEAN_fit %>%
  augment()
```

```
## # A tsibble: 1,320 x 7 [1M]
## # Key:       Country, .model [10]
##    Country   .model    Month Arrivals .fitted  .resid  .innov
##    <chr>     <chr>     <mth>    <dbl>   <dbl>   <dbl>   <dbl>
##  1 Indonesia ets    2008 Jan    62683  56534.   6149.  0.109
##  2 Indonesia ets    2008 Feb    47834  46417.   1417.  0.0305
##  3 Indonesia ets    2008 Mar    64688  62660.   2028.  0.0324
##  4 Indonesia ets    2008 Apr    58074  61045.  -2971. -0.0487
##  5 Indonesia ets    2008 May    57089  62280.  -5191. -0.0833
##  6 Indonesia ets    2008 Jun    70118  75791.  -5673. -0.0749
##  7 Indonesia ets    2008 Jul    73805  78691.  -4886. -0.0621
##  8 Indonesia ets    2008 Aug    58015  61910.  -3895. -0.0629
##  9 Indonesia ets    2008 Sep    63730  74518. -10788. -0.145
## 10 Indonesia ets    2008 Oct    71206  67869.   3337.  0.0492
## # … with 1,310 more rows
```

Comparing Fit Models

In the code chunk below, `accuracy()` is used to compare the performances of the models.

```
ASEAN_fit %>%
  accuracy() %>%
  arrange(Country)
```

```
## # A tibble: 10 × 11
##    Country    .model .type       ME   RMSE   MAE     MPE  MAPE  MASE RMSSE     ACF1
##    <chr>      <chr>  <chr>    <dbl>  <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>
##  1 Indonesia ets    Trai… -1.81e3 13187. 9665. -1.83    7.57 0.556 0.619 -0.236
##  2 Indonesia arima  Trai… -9.54e1 11351. 8382. -0.136   6.38 0.482 0.533 -0.00802
##  3 Malaysia  ets    Trai… -6.78e2  4515. 3538. -1.25    5.15 0.529 0.527 -0.288
##  4 Malaysia  arima  Trai… -2.33e1  4801. 3684. -0.109   5.20 0.551 0.561 -0.00933
##  5 Philippi… ets    Trai… -2.35e0  2298. 1897. -0.334   4.64 0.464 0.408  0.0400
##  6 Philippi… arima  Trai…  9.53e0  2624. 1934. -0.269   4.60 0.473 0.466  0.00717
##  7 Thailand  ets    Trai…  1.97e1  2323. 1773. -0.511   5.89 0.489 0.485 -0.0812
##  8 Thailand  arima  Trai…  5.88e1  2710. 1932. -0.562   6.16 0.532 0.565 -0.0112
##  9 Vietnam   ets    Trai… -3.52e1  1745. 1386. -0.728   5.29 0.467 0.473  0.279
## 10 Vietnam   arima  Trai…  1.95e0  1907. 1458. -0.671   5.37 0.491 0.517  0.0136
```

Forecast Future Values

Forecasts from these models can be produced directly as our specified models do not require any additional data.

```
ASEAN_fc <- ASEAN_fit %>%
  forecast(h = "12 months")
```

Visualising the forecasted values

In the code chunk below `autoplot()` of feasts package is used to plot the raw and fitted values.

```
ASEAN_fc %>%
  autoplot(ASEAN)
```