Day Four: Principles of Time Series Forecasting

Lumumba Wandera Victor

2023-11-14

## Set up the Document

knitr::opts\_chunk$set(echo = TRUE, warning=FALSE,comment = NA, message=FALSE,  
 fig.height=4, fig.width=6)

## Day Four: Principles of Time Series Forecasting

## FIRST SESSION 10:00 AM TO 11:30 AM

## Exponential Smoothing

##### Summary Notes on Exponential Smoothing

Exponential smoothing is a widely used time series forecasting method that provides a way to make predictions based on past data while giving more weight to recent observations. It is particularly useful for modeling and forecasting time series data with a trend and/or seasonality. Here are detailed notes on exponential smoothing for time modeling and forecasting:

**1. Time Series Data:**

* Exponential smoothing is used for analyzing and forecasting time series data, where observations are recorded at regular intervals (e.g., daily, monthly, yearly).

**2. Smoothing Parameter (α):**

* Exponential smoothing relies on a smoothing parameter (α), which controls the weight given to the most recent observation. A smaller α places more emphasis on older data, while a larger α focuses on recent data.
* 0 < α < 1; typically chosen through trial and error or statistical methods.

**3. Three Main Types:**

* There are three primary forms of exponential smoothing:
* a. **Simple Exponential Smoothing (SES):**
  + Suitable for time series data without a trend or seasonality.
  + Forecast (Ft) and smoothed value (St) are calculated using the previous data point and α.
* b. **Double Exponential Smoothing (Holt’s Exponential Smoothing):**
  + Designed for time series data with a trend but no seasonality.
  + Adds a trend component (Tt) to SES for forecasting.
  + Forecast (Ft) and smoothed value (St) are calculated based on both the level and trend.
* c. **Triple Exponential Smoothing (Holt-Winters Exponential Smoothing):**
  + Applicable to time series data with both trend and seasonality.
  + Includes three components: level (Lt), trend (Tt), and seasonality (St).
  + Requires different equations for forecasting the three components.

**4. Initialization:**

* The initial values for the components (e.g., initial level, trend, seasonality) need to be set or estimated, usually based on historical data.

**5. Forecasting:**

* Exponential smoothing provides forecasts for future periods by updating the components in each time step:
  + Level Update: Lt = α \* Yt + (1 - α) \* (Lt-1), where Yt is the observed value.
  + Trend Update (for double and triple exponential smoothing): Tt = β \* (Lt - Lt-1) + (1 - β) \* Tt-1
  + Seasonality Update (for triple exponential smoothing): St = γ \* (Yt - Lt) + (1 - γ) \* St-m, where m is the seasonality period.

**6. Error Metrics:**

* To evaluate the forecasting accuracy, common error metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE).

**7. Seasonal Decomposition:**

* For triple exponential smoothing, the method decomposes the time series into level, trend, and seasonal components, which can be insightful for understanding the data.

**8. Advantages:**

* Exponential smoothing is simple to implement, computationally efficient, and adaptable to various time series patterns.
* It can be automated and is suitable for short- to medium-term forecasting.

**9. Limitations:**

* Exponential smoothing may not work well for time series data with complex patterns or irregular outliers. Other forecasting methods may be more appropriate in such cases.

**10. Software and Tools:**

* Many statistical software packages, including R, Python (using libraries like Statsmodels), and specialized forecasting software, offer implementations of exponential smoothing techniques.

Exponential smoothing is a valuable method for time modeling and forecasting due to its simplicity, flexibility, and effectiveness in capturing trends and seasonality in time series data. When applied appropriately, it can provide reliable forecasts for planning and decision-making.

library(fpp3)

## fpp3: Data for “Forecasting: Principles and Practice” (3rd Edition)

### Description

All data sets required for the examples and exercises in the book “Forecasting: principles and practice” by Rob J Hyndman and George Athanasopoulos [https://OTexts.com/fpp3/](https://otexts.com/fpp3/). All packages required to run the examples are also loaded.

### Author(s)

**Maintainer**: Rob Hyndman [Rob.Hyndman@monash.edu](mailto:Rob.Hyndman@monash.edu) ([ORCID](https://orcid.org/0000-0002-2140-5352)) [copyright holder]

Other contributors:

* George Athanasopoulos [contributor]
* Mitchell O’Hara-Wild [contributor]
* RStudio [copyright holder]

### See Also

Useful links:

* <https://github.com/robjhyndman/fpp3package>
* [https://OTexts.com/fpp3/](https://otexts.com/fpp3/)
* Report bugs at <https://github.com/robjhyndman/fpp3package>

### Recap from Day 3

### Description of the Data

## Australian domestic overnight trips

### Description

A dataset containing the quarterly overnight trips from 1998 Q1 to 2016 Q4 across Australia.

### Usage

tourism

### Format

A tsibble with 23,408 rows and 5 variables:

* **Quarter**: Year quarter (index)
* **Region**: The tourism regions are formed through the aggregation of Statistical Local Areas (SLAs) which are defined by the various State and Territory tourism authorities according to their research and marketing needs
* **State**: States and territories of Australia
* **Purpose**: Stopover purpose of visit:
  + “Holiday”
  + “Visiting friends and relatives”
  + “Business”
  + “Other reason”
* **Trips**: Overnight trips in thousands

### References

[Tourism Research Australia](https://www.tra.gov.au/)

### View the Data

tourism

# A tsibble: 24,320 x 5 [1Q]  
# Key: Region, State, Purpose [304]  
 Quarter Region State Purpose Trips  
 <qtr> <chr> <chr> <chr> <dbl>  
 1 1998 Q1 Adelaide South Australia Business 135.  
 2 1998 Q2 Adelaide South Australia Business 110.  
 3 1998 Q3 Adelaide South Australia Business 166.  
 4 1998 Q4 Adelaide South Australia Business 127.  
 5 1999 Q1 Adelaide South Australia Business 137.  
 6 1999 Q2 Adelaide South Australia Business 200.  
 7 1999 Q3 Adelaide South Australia Business 169.  
 8 1999 Q4 Adelaide South Australia Business 134.  
 9 2000 Q1 Adelaide South Australia Business 154.  
10 2000 Q2 Adelaide South Australia Business 169.  
# ℹ 24,310 more rows

### Prepare the data by Summarizing to get the Sum

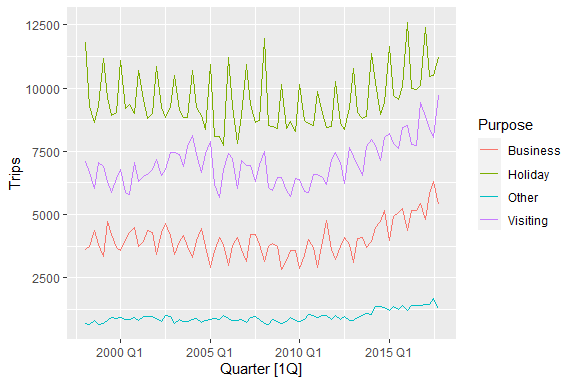
aus\_tourism\_purpose <- tourism|>   
 group\_by(Purpose) |>   
 summarise(Trips =sum(Trips))  
aus\_tourism\_purpose

# A tsibble: 320 x 3 [1Q]  
# Key: Purpose [4]  
 Purpose Quarter Trips  
 <chr> <qtr> <dbl>  
 1 Business 1998 Q1 3599.  
 2 Business 1998 Q2 3724.  
 3 Business 1998 Q3 4356.  
 4 Business 1998 Q4 3796.  
 5 Business 1999 Q1 3335.  
 6 Business 1999 Q2 4714.  
 7 Business 1999 Q3 4190.  
 8 Business 1999 Q4 3701.  
 9 Business 2000 Q1 3562.  
10 Business 2000 Q2 4018.  
# ℹ 310 more rows

The data above is prepared for use. We have the data grouped by purpose of travel and total trips or each purpose in each quarter. We have four times series because we have four purpose of travel.

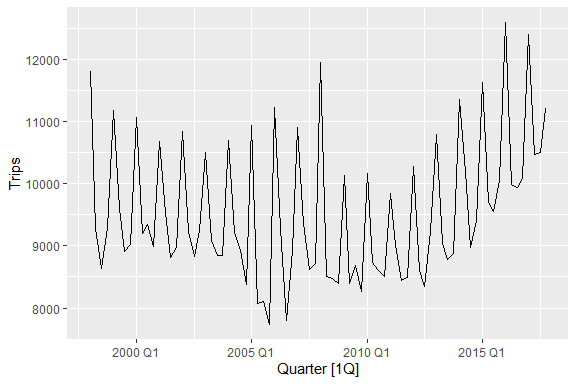
### Plot Sum of Trips

aus\_tourism\_purpose|>  
 autoplot(Trips)



### Filter for Various Purpose of Travel (Holiday

aus\_tourism\_purpose|>  
 filter(Purpose == "Holiday")|>  
 autoplot(Trips)



### Fit the Various Forecast Models

fit <- aus\_tourism\_purpose |>   
 model(average = MEAN(Trips),  
 naive = NAIVE(Trips),  
 seasonal\_naive = SNAIVE(Trips),  
 Regression = TSLM(Trips ~ trend() + season()))

In the model estimation above, the average and naive models are the bench mark model which we would like to use as our reference to compare with other models. Trips is the response variable in this case. In other words, we want to model the total number of trips. The models above also comprised of Time Series Linear Model.

### View the Models that we have

fit

# A mable: 4 x 5  
# Key: Purpose [4]  
 Purpose average naive seasonal\_naive Regression  
 <chr> <model> <model> <model> <model>  
1 Business <MEAN> <NAIVE> <SNAIVE> <TSLM>  
2 Holiday <MEAN> <NAIVE> <SNAIVE> <TSLM>  
3 Other <MEAN> <NAIVE> <SNAIVE> <TSLM>  
4 Visiting <MEAN> <NAIVE> <SNAIVE> <TSLM>

We have four models (average model, naive model, seasonal naive and time series regression model, grouped by purpose of travel

fit |>   
 glance()

# A tibble: 16 × 16  
 Purpose .model sigma2 r\_squared adj\_r\_squared statistic p\_value df  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
 1 Business average 4.86e5 NA NA NA NA NA  
 2 Business naive 3.81e5 NA NA NA NA NA  
 3 Business seasonal\_n… 1.63e5 NA NA NA NA NA  
 4 Business Regression 2.94e5 0.426 0.395 13.9 1.59e- 8 5  
 5 Holiday average 1.23e6 NA NA NA NA NA  
 6 Holiday naive 2.23e6 NA NA NA NA NA  
 7 Holiday seasonal\_n… 2.90e5 NA NA NA NA NA  
 8 Holiday Regression 4.19e5 0.677 0.660 39.4 9.81e-18 5  
 9 Other average 5.14e4 NA NA NA NA NA  
10 Other naive 1.62e4 NA NA NA NA NA  
11 Other seasonal\_n… 2.02e4 NA NA NA NA NA  
12 Other Regression 2.53e4 0.532 0.507 21.3 8.78e-12 5  
13 Visiting average 7.21e5 NA NA NA NA NA  
14 Visiting naive 4.77e5 NA NA NA NA NA  
15 Visiting seasonal\_n… 2.41e5 NA NA NA NA NA  
16 Visiting Regression 3.99e5 0.475 0.447 17.0 5.95e-10 5  
# ℹ 8 more variables: log\_lik <dbl>, AIC <dbl>, AICc <dbl>, BIC <dbl>,  
# CV <dbl>, deviance <dbl>, df.residual <int>, rank <int>

In the above results we have all the four models for every purpose of travel. Besides, information for each models including r-square, adjusted r-square, AIC, AICc, BIC among other information.

fit |>   
 tidy()

# A tibble: 24 × 7  
 Purpose .model term estimate std.error statistic p.value  
 <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
 1 Business average mean 4025. 77.9 51.7 1.17e-62  
 2 Business Regression (Intercept) 2982. 159. 18.8 4.35e-30  
 3 Business Regression trend() 11.3 2.63 4.30 5.07e- 5  
 4 Business Regression season()year2 692. 171. 4.04 1.30e- 4  
 5 Business Regression season()year3 981. 171. 5.72 2.04e- 7  
 6 Business Regression season()year4 670. 172. 3.91 2.03e- 4  
 7 Holiday average mean 9540. 124. 76.9 5.19e-76  
 8 Holiday Regression (Intercept) 10667. 189. 56.3 4.13e-63  
 9 Holiday Regression trend() 9.75 3.14 3.11 2.66e- 3  
10 Holiday Regression season()year2 -1833. 205. -8.96 1.80e-13  
# ℹ 14 more rows

The out above also provide us with information for the models estimated. We can use report function to look the various models. However, it is good to note that report function work for a single model as shown below

### Report the Model (Time Series Regression Model for the Four Purpose of Travel)

fit |> select(Regression) |>   
 report()

# A tibble: 4 × 16  
 Purpose .model r\_squared adj\_r\_squared sigma2 statistic p\_value df log\_lik  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl>  
1 Busine… Regre… 0.426 0.395 2.94e5 13.9 1.59e- 8 5 -615.  
2 Holiday Regre… 0.677 0.660 4.19e5 39.4 9.81e-18 5 -629.  
3 Other Regre… 0.532 0.507 2.53e4 21.3 8.78e-12 5 -517.  
4 Visiti… Regre… 0.475 0.447 3.99e5 17.0 5.95e-10 5 -627.  
# ℹ 7 more variables: AIC <dbl>, AICc <dbl>, BIC <dbl>, CV <dbl>,  
# deviance <dbl>, df.residual <int>, rank <int>

### Report the TSLM for Holiday Purpose of Travel

fit |> select(Regression) |>   
 filter(Purpose == "Holiday") |>   
 report()

Series: Trips   
Model: TSLM   
  
Residuals:  
 Min 1Q Median 3Q Max   
-1337.70 -422.92 49.49 390.46 1807.66   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 10667.368 189.484 56.297 < 2e-16 \*\*\*  
trend() 9.751 3.137 3.108 0.00266 \*\*   
season()year2 -1833.194 204.662 -8.957 1.80e-13 \*\*\*  
season()year3 -2210.185 204.734 -10.795 < 2e-16 \*\*\*  
season()year4 -2044.254 204.855 -9.979 2.09e-15 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 647.1 on 75 degrees of freedom  
Multiple R-squared: 0.6775, Adjusted R-squared: 0.6603  
F-statistic: 39.39 on 4 and 75 DF, p-value: < 2.22e-16

### Forecast using the Esytimated Model Above

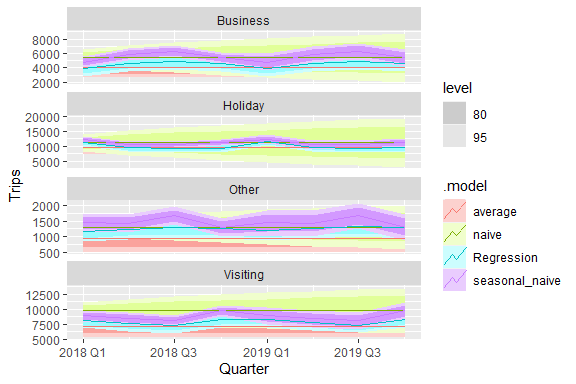
fcst <- fit |>   
 forecast(h = "2 years")  
fcst

# A fable: 128 x 5 [1Q]  
# Key: Purpose, .model [16]  
 Purpose .model Quarter Trips .mean  
 <chr> <chr> <qtr> <dist> <dbl>  
 1 Business average 2018 Q1 N(4025, 491580) 4025.  
 2 Business average 2018 Q2 N(4025, 491580) 4025.  
 3 Business average 2018 Q3 N(4025, 491580) 4025.  
 4 Business average 2018 Q4 N(4025, 491580) 4025.  
 5 Business average 2019 Q1 N(4025, 491580) 4025.  
 6 Business average 2019 Q2 N(4025, 491580) 4025.  
 7 Business average 2019 Q3 N(4025, 491580) 4025.  
 8 Business average 2019 Q4 N(4025, 491580) 4025.  
 9 Business naive 2018 Q1 N(5378, 376327) 5378.  
10 Business naive 2018 Q2 N(5378, 752655) 5378.  
# ℹ 118 more rows

Here we need to know what each column represents. The model column correspond to the list of models estimated (four models). Quarter column are the quarters we are forecasting a head. We also get the distribution of the forecast based on the normal distribution, that is, the Trips column. We also have the mean column which is the point focus.

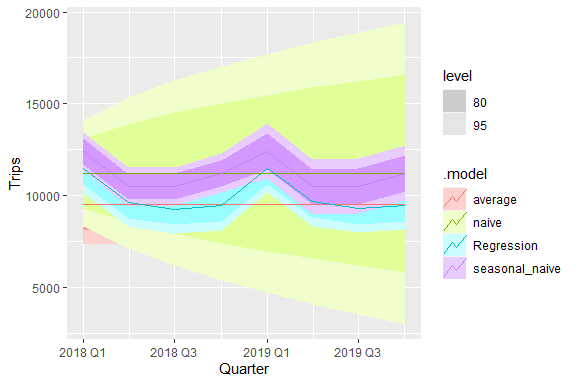
### Visualize the Focus

fcst|>  
 autoplot()



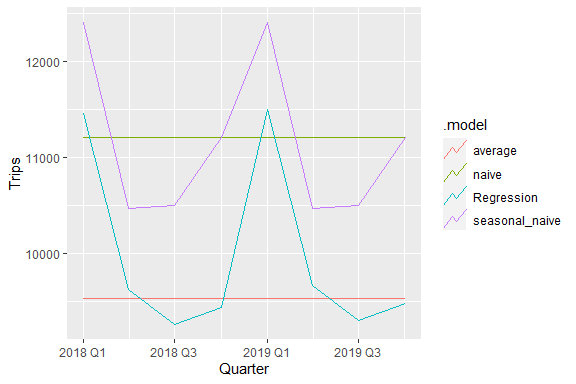
### Visualize the Model for Only One Purpose say Holiday

fcst|>  
 filter(Purpose == "Holiday")|>  
 autoplot()



In the plot above, average and naive model are not doing so well. Let us remove the level of confidence and have a closer look at the plot

fcst|>  
 filter(Purpose == "Holiday")|>  
 autoplot(level = NULL)



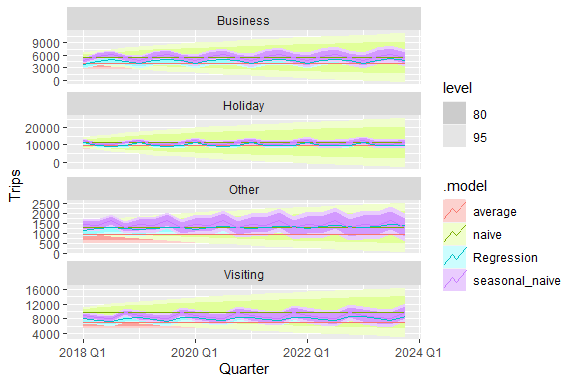
It is now clear that naive model and regression are not doing quite better. Seasonal naive and time series regression model are behaving quite the same. Let us now forecast using (h=24). There is a difference between h=24 and h = “2 years”. The latter shows two year with results in four quarters while the former implies we are forecasting 24 quarters ahead, which is five years. If the granularity of your tsibble is quarterly data then h = “2 years” would mean that you want to forecast the next eight quarters.

fcst <- fit |>   
 forecast(h = 24)  
fcst

# A fable: 384 x 5 [1Q]  
# Key: Purpose, .model [16]  
 Purpose .model Quarter Trips .mean  
 <chr> <chr> <qtr> <dist> <dbl>  
 1 Business average 2018 Q1 N(4025, 491580) 4025.  
 2 Business average 2018 Q2 N(4025, 491580) 4025.  
 3 Business average 2018 Q3 N(4025, 491580) 4025.  
 4 Business average 2018 Q4 N(4025, 491580) 4025.  
 5 Business average 2019 Q1 N(4025, 491580) 4025.  
 6 Business average 2019 Q2 N(4025, 491580) 4025.  
 7 Business average 2019 Q3 N(4025, 491580) 4025.  
 8 Business average 2019 Q4 N(4025, 491580) 4025.  
 9 Business average 2020 Q1 N(4025, 491580) 4025.  
10 Business average 2020 Q2 N(4025, 491580) 4025.  
# ℹ 374 more rows

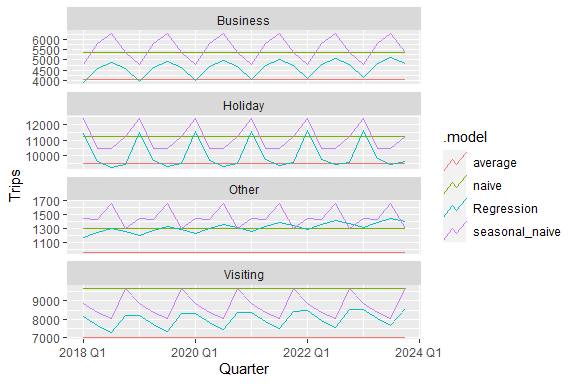
### Plot the Forecast for the 24 Quaters

fcst |>   
 autoplot()



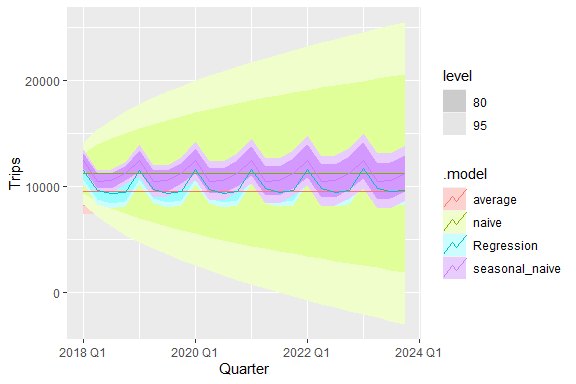
#### Remove the Confidence Bound

fcst |>   
 autoplot(level = NULL)



We can now easily extract the forecast one specific purpose of travel (Holiday)

fcst |>   
 filter(Purpose == "Holiday")|>  
 autoplot()



The naive forest has a very wide forecast interval. The wider the forest interval, the uncertain we become about our forecast. When a forecast interval is very wide, it typically means that there is a significant amount of uncertainty or variability associated with the forecasted value. Several factors contribute to a wide forecast interval, and the implications of this wide interval can vary depending on the specific context and application. Here are some things that can happen when the forecast interval is very wide:

1. **High Uncertainty:** A wide forecast interval suggests that the forecast model or method is unable to make precise predictions. This could be due to the complexity of the data or the limitations of the forecasting technique. In practical terms, this means that the actual outcome could fall within a broad range of values.
2. **Limited Predictive Power:** A wide forecast interval can indicate that the available data is not highly informative for making accurate predictions. This could be because of data quality issues, missing information, or the presence of multiple factors that affect the outcome.
3. **Risk Management:** A wide forecast interval is often seen in situations with a high degree of risk. Decision-makers need to be aware of this uncertainty and plan accordingly. For example, in financial markets, a wide forecast interval can signal a high level of volatility, which may lead to cautious investment decisions.
4. **Need for Sensitivity Analysis:** When the forecast interval is wide, it may be advisable to conduct sensitivity analysis. This involves assessing how changes in various input variables or assumptions affect the forecasted outcome. Sensitivity analysis helps decision-makers understand the potential impact of different scenarios.
5. **Communication Challenges:** Wide forecast intervals can pose communication challenges, as it can be difficult to convey the level of uncertainty to stakeholders. Decision-makers may need to balance the desire for precise forecasts with the need to acknowledge uncertainty.
6. **Risk Mitigation:** A wide forecast interval can lead to a greater focus on risk mitigation strategies. In situations with substantial uncertainty, organizations may invest in risk management, diversification, or hedging to protect against adverse outcomes.
7. **Data Collection and Model Improvement:** In cases where the forecast interval remains consistently wide, it may be necessary to collect more relevant data, improve modeling techniques, or consider alternative forecasting methods to reduce uncertainty.

In summary, a wide forecast interval implies significant uncertainty and the potential for a range of outcomes. Decision-makers should consider the implications of this uncertainty in their planning and risk management efforts. It is essential to understand the underlying reasons for the wide forecast interval and take appropriate actions to manage risk and make informed decisions.

### Get the 95% Prediction Interval on Separate Columns

fcst |>   
 hilo(level = 95) |>   
 unpack\_hilo("95%")

# A tsibble: 384 x 7 [1Q]  
# Key: Purpose, .model [16]  
 Purpose .model Quarter Trips .mean `95%\_lower` `95%\_upper`  
 <chr> <chr> <qtr> <dist> <dbl> <dbl> <dbl>  
 1 Business average 2018 Q1 N(4025, 491580) 4025. 2651. 5399.  
 2 Business average 2018 Q2 N(4025, 491580) 4025. 2651. 5399.  
 3 Business average 2018 Q3 N(4025, 491580) 4025. 2651. 5399.  
 4 Business average 2018 Q4 N(4025, 491580) 4025. 2651. 5399.  
 5 Business average 2019 Q1 N(4025, 491580) 4025. 2651. 5399.  
 6 Business average 2019 Q2 N(4025, 491580) 4025. 2651. 5399.  
 7 Business average 2019 Q3 N(4025, 491580) 4025. 2651. 5399.  
 8 Business average 2019 Q4 N(4025, 491580) 4025. 2651. 5399.  
 9 Business average 2020 Q1 N(4025, 491580) 4025. 2651. 5399.  
10 Business average 2020 Q2 N(4025, 491580) 4025. 2651. 5399.  
# ℹ 374 more rows

So now, we have the entire forecast distribution, stored in Trips column, the point forecast stored in mean column and the 95% prediction interval.

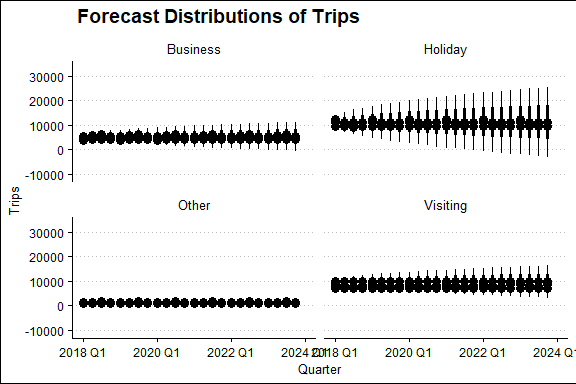
### Forecast the Distribution

### Load the Following Library

library(ggdist)

#### Make the Plot to show the Distribution of Forecast

fcst |> ggplot(aes(x = Quarter, ydist =Trips))+  
 stat\_halfeye()+  
 facet\_wrap(vars(Purpose))+  
 labs(x = "Quarter", y="Trips", title = " Forecast Distributions of Trips")+  
 ggthemes::theme\_clean()



You can interpret the distribuction of the forecast for various purposes.

## EXPONENTIAL SMOOTHING

### From simple methods to Exponential Smoothing

Naive use only the last observation to makes prediction for the indicated horizone. On the other hand, average method use all observations to make predi ction. In this case, the average of the entire observations is considered as the preeiction for the future for the indicated horizon. But now in this case, we want something in between naive and average methods. Most recent data should have more weight. This is exactly the concept behind exponential smoothing

### Load Global Economic Data

global\_economy

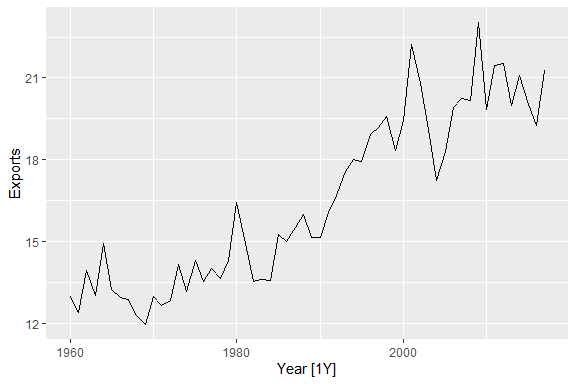
# A tsibble: 15,150 x 9 [1Y]  
# Key: Country [263]  
 Country Code Year GDP Growth CPI Imports Exports Population  
 <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Afghanistan AFG 1960 537777811. NA NA 7.02 4.13 8996351  
 2 Afghanistan AFG 1961 548888896. NA NA 8.10 4.45 9166764  
 3 Afghanistan AFG 1962 546666678. NA NA 9.35 4.88 9345868  
 4 Afghanistan AFG 1963 751111191. NA NA 16.9 9.17 9533954  
 5 Afghanistan AFG 1964 800000044. NA NA 18.1 8.89 9731361  
 6 Afghanistan AFG 1965 1006666638. NA NA 21.4 11.3 9938414  
 7 Afghanistan AFG 1966 1399999967. NA NA 18.6 8.57 10152331  
 8 Afghanistan AFG 1967 1673333418. NA NA 14.2 6.77 10372630  
 9 Afghanistan AFG 1968 1373333367. NA NA 15.2 8.90 10604346  
10 Afghanistan AFG 1969 1408888922. NA NA 15.0 10.1 10854428  
# ℹ 15,140 more rows

When modeling an ExponenTial Smoothing model has three main components and these are Error, Trend and Season.

1. Error: Additive (“A”) or multiplicative (“M”)
2. Trend: None (“N”), additive (“A”), multiplicative (“M”), or damped (“Ad” or “Md”).
3. Seasonality: None (“N”), additive (“A”) or multiplicative (“M”)

### Plot Australian Exports

global\_economy|>  
 filter(Country == "Australia")|>  
 autoplot(Exports)



dcmp <- global\_economy |>  
 filter(Country == "Australia")|>  
 filter(Year >= year(1960)) |>  
 model(STL(Exports))  
dcmp

# A mable: 1 x 2  
# Key: Country [1]  
 Country `STL(Exports)`  
 <fct> <model>  
1 Australia <STL>

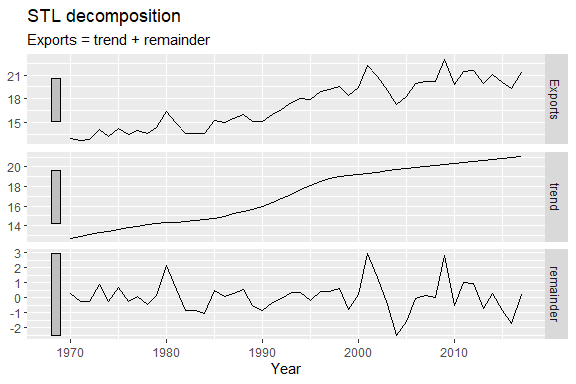
### View the Decomposed Time Series

dcmp |>  
 components()

# A dable: 48 x 7 [1Y]  
# Key: Country, .model [1]  
# : Exports = trend + remainder  
 Country .model Year Exports trend remainder season\_adjust  
 <fct> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Australia STL(Exports) 1970 13.0 12.7 0.254 13.0  
 2 Australia STL(Exports) 1971 12.7 12.9 -0.234 12.7  
 3 Australia STL(Exports) 1972 12.8 13.1 -0.241 12.8  
 4 Australia STL(Exports) 1973 14.2 13.2 0.909 14.2  
 5 Australia STL(Exports) 1974 13.2 13.4 -0.282 13.2  
 6 Australia STL(Exports) 1975 14.3 13.6 0.661 14.3  
 7 Australia STL(Exports) 1976 13.5 13.8 -0.272 13.5  
 8 Australia STL(Exports) 1977 14.0 13.9 0.0858 14.0  
 9 Australia STL(Exports) 1978 13.6 14.1 -0.443 13.6  
10 Australia STL(Exports) 1979 14.3 14.2 0.136 14.3  
# ℹ 38 more rows

### Plot the Decomposition

dcmp |>  
 components() |>  
 autoplot(Exports)



From the plot above, our time series has no seasonal component, so we remain with the trend and remainder.

#### General ETS Model (Automatic ETS)

fit <- global\_economy |>   
 filter(Country == "Australia") |>   
 model(ses = ETS(Exports ~ error("A") + trend("N") + season("N")))  
  
fit

# A mable: 1 x 2  
# Key: Country [1]  
 Country ses  
 <fct> <model>  
1 Australia <ETS(A,N,N)>

From the plot, we can see that the error term is additive, trend is none, seasonal is also none. In other words, the models assume trend and seasonality is none.

#### Glance()

Glance will return will return several metrics for model evaluation including AIC , BIC and other metrics.

fit |>   
 glance()

# A tibble: 1 × 10  
 Country .model sigma2 log\_lik AIC AICc BIC MSE AMSE MAE  
 <fct> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Australia ses 1.36 -126. 257. 258. 264. 1.32 1.79 0.914

#### Use tidy() to view the alpha parameter and other information

fit |>   
 tidy()

# A tibble: 2 × 4  
 Country .model term estimate  
 <fct> <chr> <chr> <dbl>  
1 Australia ses alpha 0.566  
2 Australia ses l[0] 13.0

The output above gives us alpha value and l(0). It is however, important to note that when we have an exponential smoothing, basically the process has to start from a particular point. Thus the process goes to the beginning of the time series. Remember exponential smoothing are recursive calculations, we start from the beginning of the times series and go on doing calculation to the last observation of the time series in order to get the fitted values. To have this process, we need something in exponential smoothing called l(0). If you follow the calculation you can as well derive the l(0). The estimate will have l1, l2, ….ln. Now, remember, to calculate l1, you mst l(0) value.

#### Report

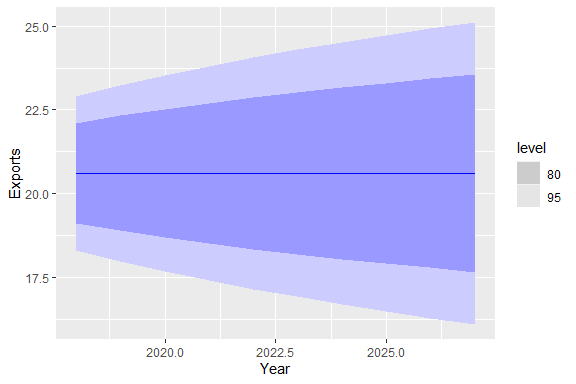
fit |>   
 report()

Series: Exports   
Model: ETS(A,N,N)   
 Smoothing parameters:  
 alpha = 0.5659948   
  
 Initial states:  
 l[0]  
 12.98943  
  
 sigma^2: 1.3621  
  
 AIC AICc BIC   
257.3943 257.8387 263.5756

The report tell us what is the model that we estimated and also give us some parameters like apha that we have seen before.

### Forecast

fit |>   
 forecast(h=10) |>   
 autoplot()



### Australian Economy

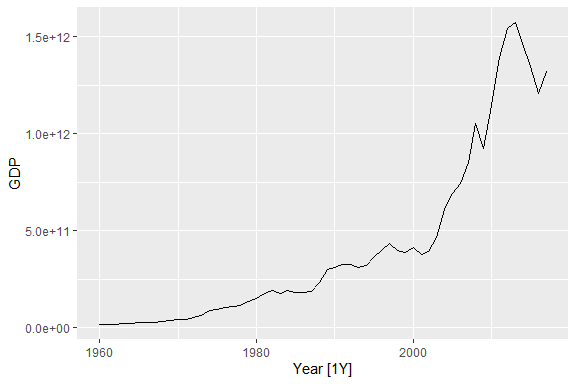
aus\_economy <- global\_economy |>   
 filter(Country == "Australia") |>  
 mutate(Pop = Population / 1e6)

aus\_economy

# A tsibble: 58 x 10 [1Y]  
# Key: Country [1]  
 Country Code Year GDP Growth CPI Imports Exports Population Pop  
 <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Australia AUS 1960 1.86e10 NA 7.96 14.1 13.0 10276477 10.3  
 2 Australia AUS 1961 1.96e10 2.49 8.14 15.0 12.4 10483000 10.5  
 3 Australia AUS 1962 1.99e10 1.30 8.12 12.6 13.9 10742000 10.7  
 4 Australia AUS 1963 2.15e10 6.21 8.17 13.8 13.0 10950000 11.0  
 5 Australia AUS 1964 2.38e10 6.98 8.40 13.8 14.9 11167000 11.2  
 6 Australia AUS 1965 2.59e10 5.98 8.69 15.3 13.2 11388000 11.4  
 7 Australia AUS 1966 2.73e10 2.38 8.98 15.1 12.9 11651000 11.7  
 8 Australia AUS 1967 3.04e10 6.30 9.29 13.9 12.9 11799000 11.8  
 9 Australia AUS 1968 3.27e10 5.10 9.52 14.5 12.3 12009000 12.0  
10 Australia AUS 1969 3.66e10 7.04 9.83 13.3 12.0 12263000 12.3  
# ℹ 48 more rows

### Autoplot Australian Economic Growth (GDP

autoplot(aus\_economy)



fit <- aus\_economy |>  
 model(ETS(Pop))  
fit

# A mable: 1 x 2  
# Key: Country [1]  
 Country `ETS(Pop)`  
 <fct> <model>  
1 Australia <ETS(A,A,N)>

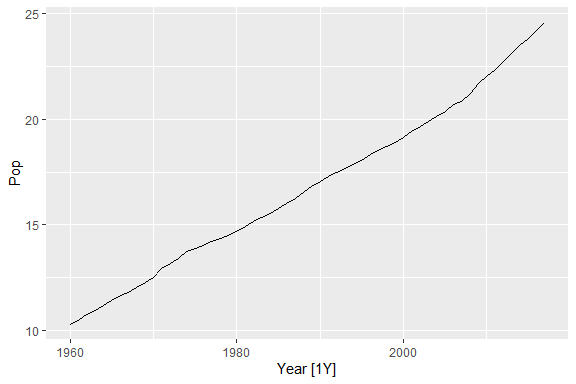
#### Report the Model

report(fit)

Series: Pop   
Model: ETS(A,A,N)   
 Smoothing parameters:  
 alpha = 0.9999   
 beta = 0.3266366   
  
 Initial states:  
 l[0] b[0]  
 10.05414 0.2224818  
  
 sigma^2: 0.0041  
  
 AIC AICc BIC   
-76.98569 -75.83184 -66.68347

The output above shows that ther error of our model is additive, the trends is also additive as well, however, we do not have seasonal component in the data. Let us plot and have a look at the plot.

aus\_economy|>  
 autoplot(Pop)



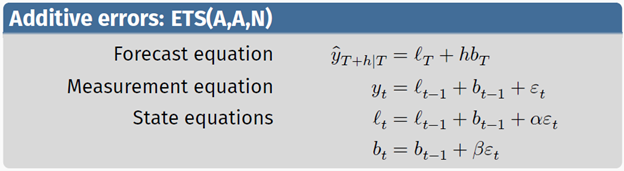
#### Extract the Components

fit |>   
 components()

# A dable: 59 x 7 [1Y]  
# Key: Country, .model [1]  
# : Pop = lag(level, 1) + lag(slope, 1) + remainder  
 Country .model Year Pop level slope remainder  
 <fct> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Australia ETS(Pop) 1959 NA 10.1 0.222 NA   
 2 Australia ETS(Pop) 1960 10.3 10.3 0.222 -0.000145  
 3 Australia ETS(Pop) 1961 10.5 10.5 0.217 -0.0159   
 4 Australia ETS(Pop) 1962 10.7 10.7 0.231 0.0418   
 5 Australia ETS(Pop) 1963 11.0 11.0 0.223 -0.0229   
 6 Australia ETS(Pop) 1964 11.2 11.2 0.221 -0.00641   
 7 Australia ETS(Pop) 1965 11.4 11.4 0.221 -0.000314  
 8 Australia ETS(Pop) 1966 11.7 11.7 0.235 0.0418   
 9 Australia ETS(Pop) 1967 11.8 11.8 0.206 -0.0869   
10 Australia ETS(Pop) 1968 12.0 12.0 0.208 0.00350   
# ℹ 49 more rows

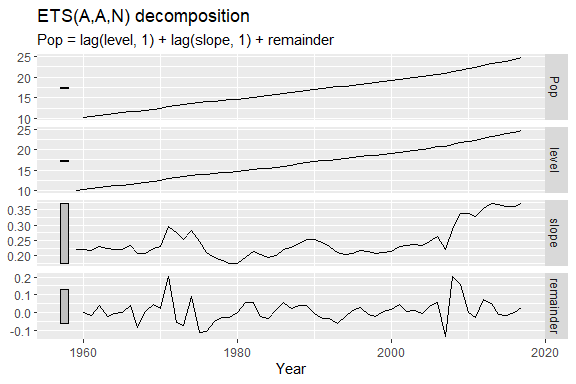
Level correponse to the state equation which has lt in it. Slope column correspond to the equation that has bt in it, and finnaly, the remainder correspond to the epsilon part. Consider the information below.

knitr::include\_graphics("ets.png")



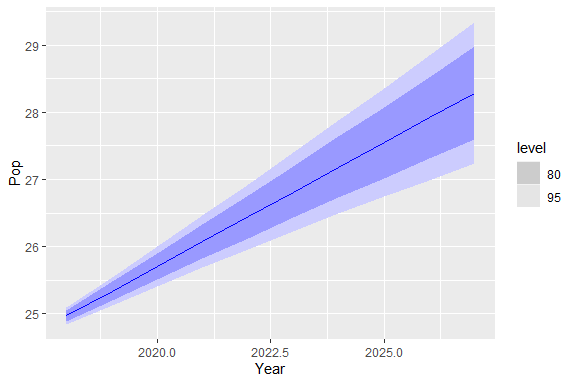
#### Plot the Component of the Time Series

fit |>   
 components() |>  
 autoplot()



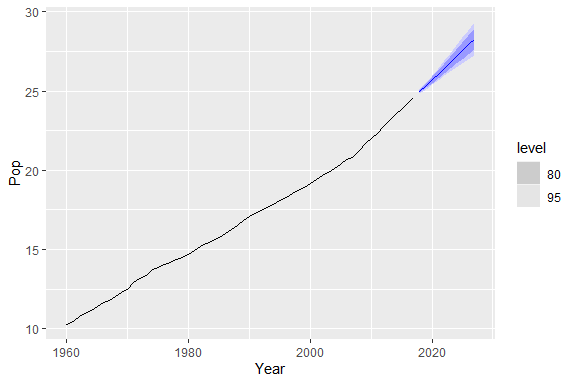
### Forecast the Next ten Periods and Create the Plot

fit |>   
 forecast(h=10) |>   
 autoplot()



### Add the Original Data to the Plot

fit |>   
 forecast(h=10) |>   
 autoplot(aus\_economy)



The plot above seems to be a good prediction model since the prediction interval are quite narrower as required.

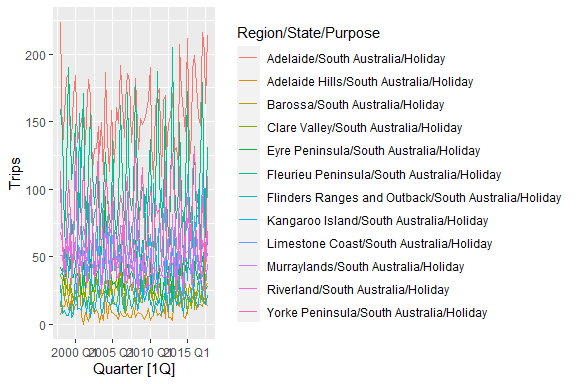
### Model the Exponential Smoothing with Holidays Data set

holiday <- tourism |>   
 filter(Purpose == "Holiday")  
holiday

# A tsibble: 6,080 x 5 [1Q]  
# Key: Region, State, Purpose [76]  
 Quarter Region State Purpose Trips  
 <qtr> <chr> <chr> <chr> <dbl>  
 1 1998 Q1 Adelaide South Australia Holiday 224.  
 2 1998 Q2 Adelaide South Australia Holiday 130.  
 3 1998 Q3 Adelaide South Australia Holiday 156.  
 4 1998 Q4 Adelaide South Australia Holiday 182.  
 5 1999 Q1 Adelaide South Australia Holiday 185.  
 6 1999 Q2 Adelaide South Australia Holiday 135.  
 7 1999 Q3 Adelaide South Australia Holiday 136.  
 8 1999 Q4 Adelaide South Australia Holiday 169.  
 9 2000 Q1 Adelaide South Australia Holiday 184.  
10 2000 Q2 Adelaide South Australia Holiday 134.  
# ℹ 6,070 more rows

### Plot the Trips for Holiday Purpose

holiday|>  
 #filter(Region == "Adelaide")|>  
 filter(State == "South Australia")|>  
 autoplot()



### Estimate the ETS Model

fit <- holiday |>   
 model(ets = ETS(Trips))

### Glance

fit|>  
 glance()

# A tibble: 76 × 12  
 Region State Purpose .model sigma2 log\_lik AIC AICc BIC MSE AMSE  
 <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Adelaide Sout… Holiday ets 4.57e+2 -417. 848. 850. 865. 422. 421.   
 2 Adelaid… Sout… Holiday ets 3.63e+1 -317. 644. 645. 656. 34.5 34.5  
 3 Alice S… Nort… Holiday ets 1.21e-1 -355. 724. 725. 740. 97.4 97.9  
 4 Austral… West… Holiday ets 4.19e-2 -421. 856. 857. 873. 502. 553.   
 5 Austral… West… Holiday ets 8.13e-2 -399. 812. 813. 829. 285. 294.   
 6 Austral… West… Holiday ets 2.20e+2 -388. 790. 791. 806. 203. 236.   
 7 Austral… West… Holiday ets 2.11e-2 -472. 958. 960. 975. 1946. 2163.   
 8 Ballarat Vict… Holiday ets 6.65e-2 -375. 765. 766. 781. 162. 161.   
 9 Barkly Nort… Holiday ets 3.01e+1 -308. 631. 632. 647. 27.9 28.1  
10 Barossa Sout… Holiday ets 8.44e+1 -352. 709. 710. 717. 82.3 82.2  
# ℹ 66 more rows  
# ℹ 1 more variable: MAE <dbl>

### Augment

fit|>  
 augment()

# A tsibble: 6,080 x 9 [1Q]  
# Key: Region, State, Purpose, .model [76]  
 Region State Purpose .model Quarter Trips .fitted .resid .innov  
 <chr> <chr> <chr> <chr> <qtr> <dbl> <dbl> <dbl> <dbl>  
 1 Adelaide South Australia Holiday ets 1998 Q1 224. 189. 34.8 34.8   
 2 Adelaide South Australia Holiday ets 1998 Q2 130. 158. -27.8 -27.8   
 3 Adelaide South Australia Holiday ets 1998 Q3 156. 149. 7.45 7.45  
 4 Adelaide South Australia Holiday ets 1998 Q4 182. 167. 15.6 15.6   
 5 Adelaide South Australia Holiday ets 1999 Q1 185. 194. -9.06 -9.06  
 6 Adelaide South Australia Holiday ets 1999 Q2 135. 156. -20.9 -20.9   
 7 Adelaide South Australia Holiday ets 1999 Q3 136. 147. -11.3 -11.3   
 8 Adelaide South Australia Holiday ets 1999 Q4 169. 162. 6.81 6.81  
 9 Adelaide South Australia Holiday ets 2000 Q1 184. 188. -3.81 -3.81  
10 Adelaide South Australia Holiday ets 2000 Q2 134. 150. -15.7 -15.7   
# ℹ 6,070 more rows

fit |>   
 filter(Region == "Snowy Mountains") |>   
 report()

Series: Trips   
Model: ETS(M,N,A)   
 Smoothing parameters:  
 alpha = 0.1571013   
 gamma = 0.0001000991   
  
 Initial states:  
 l[0] s[0] s[-1] s[-2] s[-3]  
 141.6782 -60.95904 130.8567 -42.23776 -27.65986  
  
 sigma^2: 0.0388  
  
 AIC AICc BIC   
852.0452 853.6008 868.7194

### Use Component Function to Inspect the Model

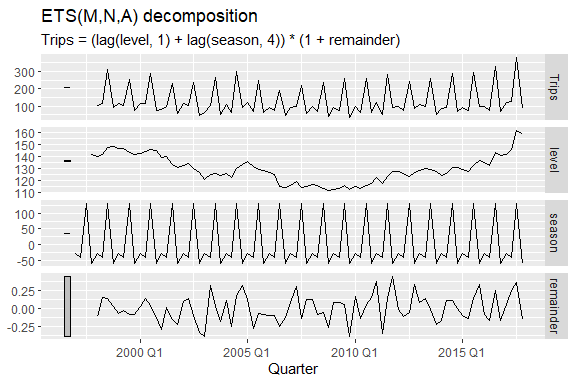
fit |>   
 filter(Region == "Snowy Mountains") |>   
 components()

# A dable: 84 x 9 [1Q]  
# Key: Region, State, Purpose, .model [1]  
# : Trips = (lag(level, 1) + lag(season, 4)) \* (1 + remainder)  
 Region State Purpose .model Quarter Trips level season remainder  
 <chr> <chr> <chr> <chr> <qtr> <dbl> <dbl> <dbl> <dbl>  
 1 Snowy Mountains New Sout… Holiday ets 1997 Q1 NA NA -27.7 NA   
 2 Snowy Mountains New Sout… Holiday ets 1997 Q2 NA NA -42.2 NA   
 3 Snowy Mountains New Sout… Holiday ets 1997 Q3 NA NA 131. NA   
 4 Snowy Mountains New Sout… Holiday ets 1997 Q4 NA 142. -61.0 NA   
 5 Snowy Mountains New Sout… Holiday ets 1998 Q1 101. 140. -27.7 -0.113   
 6 Snowy Mountains New Sout… Holiday ets 1998 Q2 112. 142. -42.2 0.154   
 7 Snowy Mountains New Sout… Holiday ets 1998 Q3 310. 148. 131. 0.137   
 8 Snowy Mountains New Sout… Holiday ets 1998 Q4 89.8 148. -61.0 0.0335  
 9 Snowy Mountains New Sout… Holiday ets 1999 Q1 112. 147. -27.7 -0.0687  
10 Snowy Mountains New Sout… Holiday ets 1999 Q2 103. 147. -42.2 -0.0199  
# ℹ 74 more rows

In the output above , we have seasonal part, and the remainder part as well. These are the s components

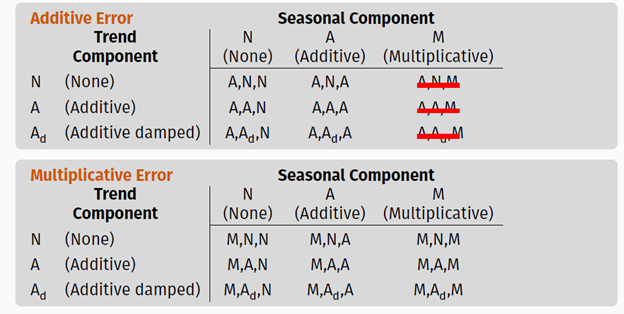
### Plot the Components

fit |>   
 filter(Region == "Snowy Mountains") |>   
 components()|>   
 autoplot()



### Summary of Trend Component with Both Additive Error and Multiplicative Error

knitr::include\_graphics("trend.png")



### Now, how does ETS Estimates Parameters

1. Smoothing parameters 𝛼, 𝛽, 𝛾 and 𝜙, and the initial states ℓ0, 𝑏0, 𝑠0, 𝑠−1, … , 𝑠−𝑚+1 are estimated by maximizing the “likelihood” = the probability of the data arising from the specified model.
2. For models with additive errors equivalent to minimizing SSE.
3. For models with multiplicative errors, not equivalent to minimizing SSE.

#### Note!!!

The parameter 𝛼 is for level, 𝛽 for trend, 𝛾 for seanality and and 𝜙 for dumped trend. We also have the initial values, like ℓ0 for the level , we need 𝑏0 to start the calculations for the levels. We also need a collection of the seasonal indices, and these 𝑠0, 𝑠−1, … , 𝑠−𝑚+1. These collection of seasonal indices depends on the temporal granularity. In the equation referes to the number of seasons, if we have quarterly data so m=4, if you monthly data = 12. We indtroduce these parameter like gamma, beta and alpha to control the change of these components over time. We leave this to the ETS function to do the job automatically. All these value are automatically estimated by the function using maximum likelihood estimation. How are models selected in ETS, we use the following,

1. Akaike’s Information Criterion (AIC)
2. Corrected Akaike’s Information Criterion (AICc)
3. Bayesian Information Criterion (BIC)

### AIC and Cross-Validation

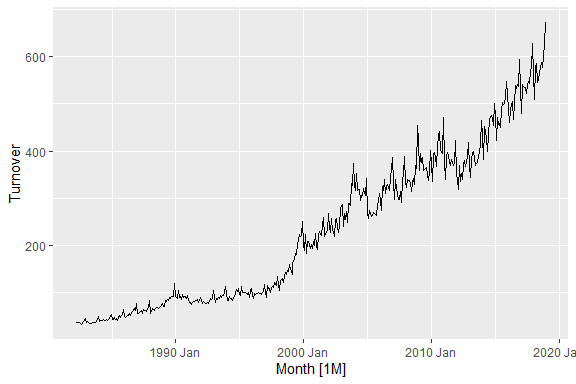
Minimizing the AIC assuming Gaussian residuals is asymptotically equivalent to minimizing one-step time series cross validation MSE. The best model is the one that minimzes the mean square error (MSE)

## Retail Data

cafe\_vic <- tsibbledata::aus\_retail |>   
 filter(State == "Victoria",  
 Industry == "Cafes, restaurants and catering services"  
 ) |> select(Month,Turnover)  
cafe\_vic

# A tsibble: 441 x 2 [1M]  
 Month Turnover  
 <mth> <dbl>  
 1 1982 Apr 36.4  
 2 1982 May 36.2  
 3 1982 Jun 35.7  
 4 1982 Jul 34.6  
 5 1982 Aug 32.5  
 6 1982 Sep 33.9  
 7 1982 Oct 37.7  
 8 1982 Nov 40.3  
 9 1982 Dec 45.2  
10 1983 Jan 36.9  
# ℹ 431 more rows

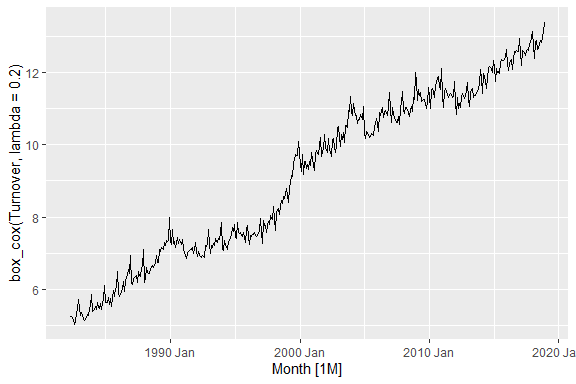
cafe\_vic|>  
 autoplot()



You can see we have multiplicative seasonality. So before modeling we will need a bit of transformation as shown below.

### How to forecast with transformations

cafe\_vic |>  
 autoplot(box\_cox(Turnover, lambda = .2))



The plot now looks better after transformation. The variance is stabalized.

fit <- cafe\_vic |>   
 model(ets = ETS(box\_cox(Turnover, lambda = .2)))  
fit

# A mable: 1 x 1  
 ets  
 <model>  
1 <ETS(A,A,A)>

The model above has additive trend and additive seasonality.

### Fit the Model without Transformation

fit <- cafe\_vic |>  
 model(ets = ETS(Turnover))  
fit

# A mable: 1 x 1  
 ets  
 <model>  
1 <ETS(M,A,M)>

Developing the model with transformation results into a model with multiplicative error, additive trend and multiple seasonality. However, for the purpose of this training, we will use the transformed model

### Forecasting (12 Periods)

fct <- fit |>   
 forecast(h = 12)

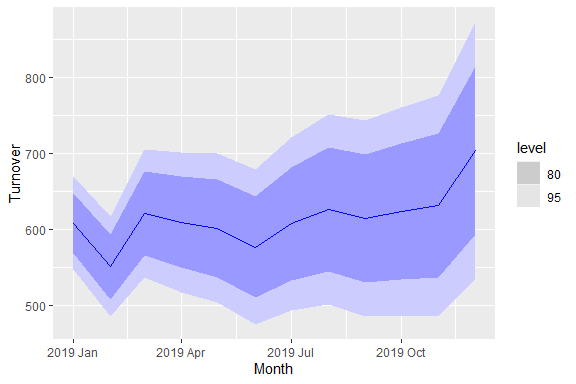
### View the Forecast

fct

# A fable: 12 x 4 [1M]  
# Key: .model [1]  
 .model Month Turnover .mean  
 <chr> <mth> <dist> <dbl>  
 1 ets 2019 Jan N(608, 978) 608.  
 2 ets 2019 Feb N(551, 1129) 551.  
 3 ets 2019 Mar N(622, 1856) 622.  
 4 ets 2019 Apr N(609, 2190) 609.  
 5 ets 2019 May N(602, 2539) 602.  
 6 ets 2019 Jun N(577, 2704) 577.  
 7 ets 2019 Jul N(607, 3413) 607.  
 8 ets 2019 Aug N(626, 4072) 626.  
 9 ets 2019 Sep N(614, 4358) 614.  
10 ets 2019 Oct N(624, 4942) 624.  
11 ets 2019 Nov N(632, 5535) 632.  
12 ets 2019 Dec N(704, 7460) 704.

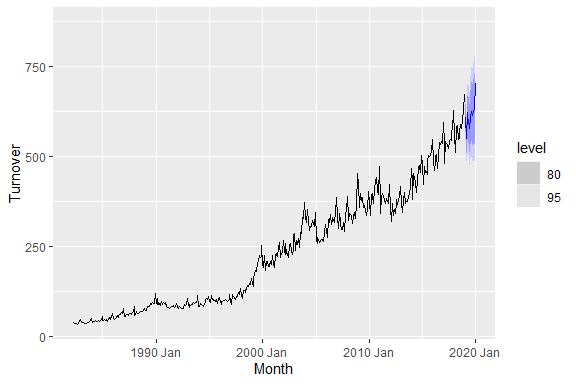
### Plot the Forecast

fct |>   
 autoplot()



### Plot the Forecast with the Data as well

fct |>   
 autoplot(filter\_index(cafe\_vic, "1982 Jan" ~ .))



### Generate function to produce forecasts using bootstrapping

simulation <- fit |>   
 generate(h= 12, times = 1000, bootstrap = TRUE)

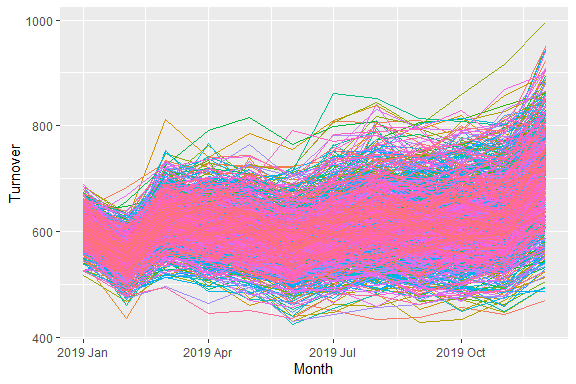
### View the Forecast

simulation

# A tsibble: 12,000 x 5 [1M]  
# Key: .model, .rep [1,000]  
 .model .rep Month .innov .sim  
 <chr> <chr> <mth> <dbl> <dbl>  
 1 ets 1 2019 Jan 0.0283 625.  
 2 ets 1 2019 Feb 0.0327 579.  
 3 ets 1 2019 Mar -0.0195 633.  
 4 ets 1 2019 Apr 0.0324 646.  
 5 ets 1 2019 May -0.00496 628.  
 6 ets 1 2019 Jun -0.0439 576.  
 7 ets 1 2019 Jul 0.0226 631.  
 8 ets 1 2019 Aug 0.0647 687.  
 9 ets 1 2019 Sep 0.0114 667.  
10 ets 1 2019 Oct -0.0164 663.  
# ℹ 11,990 more rows

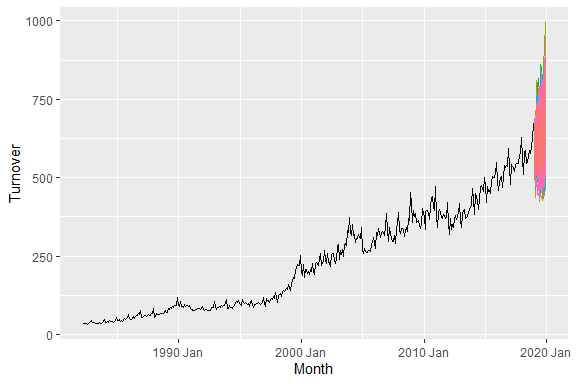
### Plot the 1000 Possible Forecast

cafe\_vic |> filter\_index("2019 Dec" ~ .) |>   
 ggplot(aes(x = Month))+  
 geom\_line(aes(y =Turnover))+  
 geom\_line(aes( y = .sim, colour = as.factor(.rep)), data = simulation)+  
 guides(col = FALSE)



### Plot the Simulation together with the Data

cafe\_vic |> filter\_index("1980 Jan" ~ .) |>   
 ggplot(aes(x = Month))+  
 geom\_line(aes(y =Turnover))+  
 geom\_line(aes( y = .sim, colour = as.factor(.rep)), data = simulation)+  
 guides(col = FALSE)



fct\_boot <- fit |>   
 forecast(h= 12, bootstrap = TRUE)  
fct\_boot

# A fable: 12 x 4 [1M]  
# Key: .model [1]  
 .model Month Turnover .mean  
 <chr> <mth> <dist> <dbl>  
 1 ets 2019 Jan sample[5000] 607.  
 2 ets 2019 Feb sample[5000] 551.  
 3 ets 2019 Mar sample[5000] 622.  
 4 ets 2019 Apr sample[5000] 610.  
 5 ets 2019 May sample[5000] 603.  
 6 ets 2019 Jun sample[5000] 578.  
 7 ets 2019 Jul sample[5000] 609.  
 8 ets 2019 Aug sample[5000] 628.  
 9 ets 2019 Sep sample[5000] 617.  
10 ets 2019 Oct sample[5000] 625.  
11 ets 2019 Nov sample[5000] 634.  
12 ets 2019 Dec sample[5000] 707.

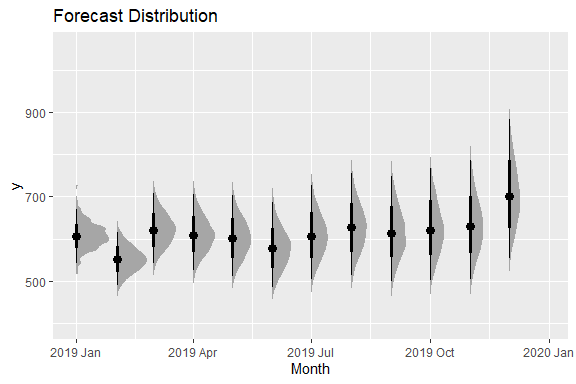
The forecast was determined from a sample of 5000 trials.

fct\_boot |>   
 hilo(level = 90)

# A tsibble: 12 x 5 [1M]  
# Key: .model [1]  
 .model Month Turnover .mean `90%`  
 <chr> <mth> <dist> <dbl> <hilo>  
 1 ets 2019 Jan sample[5000] 607. [559.3853, 657.5942]90  
 2 ets 2019 Feb sample[5000] 551. [497.9878, 605.6716]90  
 3 ets 2019 Mar sample[5000] 622. [555.5469, 693.6700]90  
 4 ets 2019 Apr sample[5000] 610. [538.2574, 688.4091]90  
 5 ets 2019 May sample[5000] 603. [524.5439, 686.4129]90  
 6 ets 2019 Jun sample[5000] 578. [497.4990, 664.6892]90  
 7 ets 2019 Jul sample[5000] 609. [520.3519, 706.8934]90  
 8 ets 2019 Aug sample[5000] 628. [530.0176, 732.2234]90  
 9 ets 2019 Sep sample[5000] 617. [516.3165, 725.2165]90  
10 ets 2019 Oct sample[5000] 625. [519.1912, 743.0346]90  
11 ets 2019 Nov sample[5000] 634. [522.0296, 759.6890]90  
12 ets 2019 Dec sample[5000] 707. [578.2979, 850.4893]90

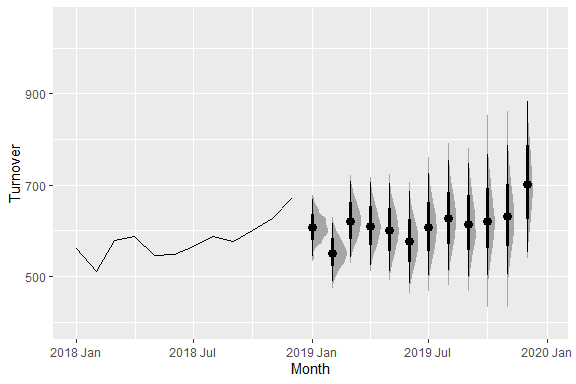
### Distribution of the Forecast

fct\_boot |>   
 ggplot(aes(x = Month, ydist =Turnover))+  
 stat\_halfeye()+  
 labs(title = "Forecast Distribution")



### Forecast Distribution together with the data as well

fct\_boot |>   
 ggplot(aes(x = Month, ydist =Turnover))+  
 stat\_halfeye()+  
 autolayer(cafe\_vic |> filter\_index("2018 Jan" ~ .))



In the beginning of the forecast, the distribution is very close to the point focus, however, as the forecast progresses, the distribution of the forecasts widens creating uncertainty in the forecast. The wider indicated a higher variability in the forecast.

## SECOND SESSION 11:45 AM TO 1:15 PM

## AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

### Global Economic Data

global\_economy

# A tsibble: 15,150 x 9 [1Y]  
# Key: Country [263]  
 Country Code Year GDP Growth CPI Imports Exports Population  
 <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Afghanistan AFG 1960 537777811. NA NA 7.02 4.13 8996351  
 2 Afghanistan AFG 1961 548888896. NA NA 8.10 4.45 9166764  
 3 Afghanistan AFG 1962 546666678. NA NA 9.35 4.88 9345868  
 4 Afghanistan AFG 1963 751111191. NA NA 16.9 9.17 9533954  
 5 Afghanistan AFG 1964 800000044. NA NA 18.1 8.89 9731361  
 6 Afghanistan AFG 1965 1006666638. NA NA 21.4 11.3 9938414  
 7 Afghanistan AFG 1966 1399999967. NA NA 18.6 8.57 10152331  
 8 Afghanistan AFG 1967 1673333418. NA NA 14.2 6.77 10372630  
 9 Afghanistan AFG 1968 1373333367. NA NA 15.2 8.90 10604346  
10 Afghanistan AFG 1969 1408888922. NA NA 15.0 10.1 10854428  
# ℹ 15,140 more rows

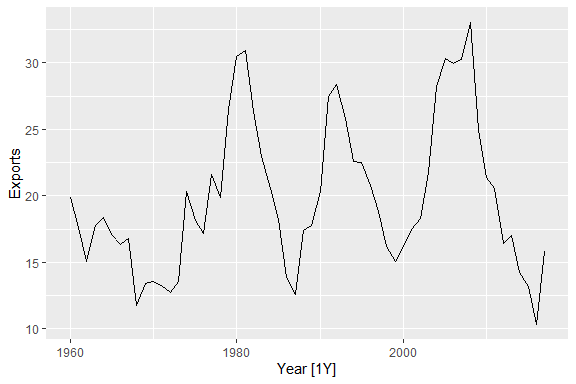
### Extract the Data for Egypt

egy\_economy <- global\_economy |>   
 filter(Code == "EGY")   
egy\_economy

# A tsibble: 58 x 9 [1Y]  
# Key: Country [1]  
 Country Code Year GDP Growth CPI Imports Exports Population  
 <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 Egypt, Arab Rep. EGY 1960 NA NA 1.21 21.8 19.9 26996533  
 2 Egypt, Arab Rep. EGY 1961 NA 5.16 1.22 19.7 17.6 27744712  
 3 Egypt, Arab Rep. EGY 1962 NA 3.88 1.18 19.5 15.1 28506176  
 4 Egypt, Arab Rep. EGY 1963 NA 10.5 1.19 23.6 17.8 29281250  
 5 Egypt, Arab Rep. EGY 1964 NA 11.5 1.23 24.9 18.4 30071102  
 6 Egypt, Arab Rep. EGY 1965 4.95e9 4.91 1.42 20.1 17.1 30875964  
 7 Egypt, Arab Rep. EGY 1966 5.28e9 5.05 1.54 20.5 16.3 31697616  
 8 Egypt, Arab Rep. EGY 1967 5.61e9 0.805 1.55 19.1 16.8 32534021  
 9 Egypt, Arab Rep. EGY 1968 5.93e9 -1.61 1.53 17.5 11.8 33377259  
10 Egypt, Arab Rep. EGY 1969 6.52e9 5.28 1.58 16.5 13.5 34216826  
# ℹ 48 more rows

### Autoplot Export Volume Relating to Egypt

egy\_economy |>   
 autoplot(Exports)



## ARIMA Model

AR: autoregressive (lagged observations as inputs)

I: integrated (differencing to make series stationary)

MA: moving average (lagged errors as inputs)

An ARIMA model is rarely interpretable in terms of visible data structures like trend and seasonality. But it can capture a huge range of time series patterns.

ARIMA, or Autoregressive Integrated Moving Average, is a widely used time series forecasting model that combines autoregressive (AR), differencing (I), and moving average (MA) components. This model is particularly effective for capturing and predicting patterns in time-dependent data. The ARIMA model is specified by three parameters: p, d, and q, where p represents the autoregressive order, d is the differencing order, and q denotes the moving average order.

The autoregressive (AR) component involves predicting a future value based on its past values. The order of autoregression (p) signifies how many past observations are considered for prediction. The differencing (I) component is applied to make the time series stationary, which is crucial for accurate forecasting. The differencing order (d) indicates how many times differencing is performed to achieve stationarity. Finally, the moving average (MA) component models the relationship between an observation and a residual error from a moving average model applied to lagged observations. The order of the moving average (q) determines how many past residual errors are considered in the prediction.

The strength of ARIMA lies in its flexibility to handle various time series patterns, including trend, seasonality, and autocorrelation. By adjusting the p, d, and q parameters, analysts can tailor the model to the specific characteristics of the time series data they are working with. Despite its effectiveness, ARIMA assumes that the underlying data is linear and stationary, and it may not perform optimally in the presence of nonlinearities or structural breaks. Additionally, careful consideration of model diagnostics and validation is essential to ensure the reliability of the forecasts generated by the ARIMA model.

### Stationarity

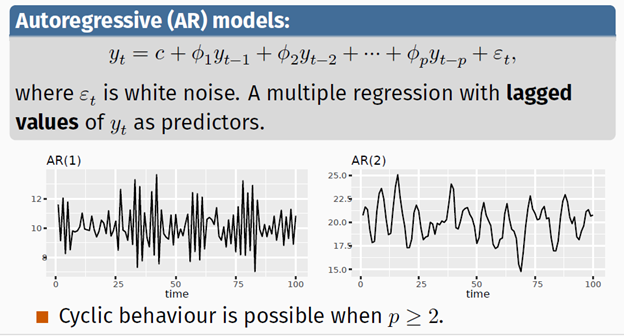
Stationarity is a fundamental concept in time series analysis, indicating that the statistical properties of a time series remain constant over time. A stationary time series exhibits constant mean, variance, and autocorrelation structure, making it more amenable to analysis and modeling. The absence of trends or seasonality in a stationary series simplifies the task of making predictions, as statistical patterns observed in the past are expected to continue into the future. Achieving stationarity often involves differencing the data to remove trends and seasonal effects. Statistical models, such as ARIMA, assume stationarity to provide accurate forecasts. Identifying and addressing non-stationarity is a critical step in time series analysis, enhancing the reliability and interpretability of models applied to temporal data.

### Why Difference the Series

Differencing helps to stabilize the mean. The differenced series is the change between each observation in the original series. Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time. In practice, it is almost never necessary to go beyond second-order differences otherwise will information from the data.

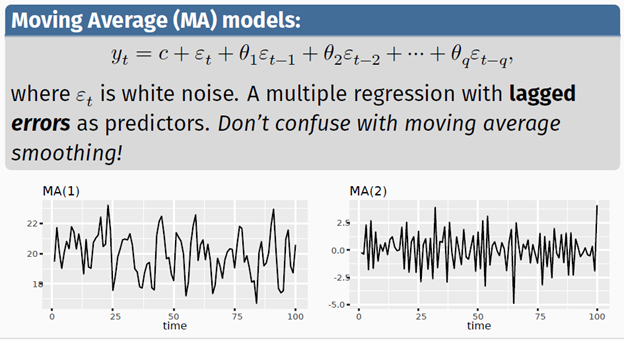
### Autoregressive Component

knitr::include\_graphics("ar.png")



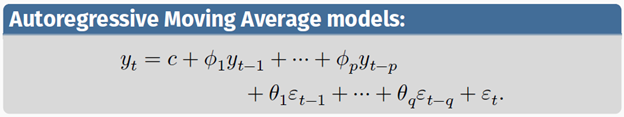
### Moving Average Component

knitr::include\_graphics("ma.png")



When the two models (AR and MA) are combine, we have the autoregressive moving average (ARMA) model as shown below

knitr::include\_graphics("arma.png")



We also a variant of ARMA model called ARIMA model which include the differencing part. The d-differencing series follows an ARMA model therefore we need to choose the value of p,d,q. In other words, we have to choose the order for autoregressive part and moving average part. Differencing is done only when the time series is not stationary. p is the order of the autoregressive part. d is the first differencing involved and q is the order of the moving average part.

## Fit the ARIMA Model

fit <- egy\_economy |>   
 model(arima = ARIMA(Exports))

### View the Model Output Using report() Function

report(fit)

Series: Exports   
Model: ARIMA(2,0,1) w/ mean   
  
Coefficients:  
 ar1 ar2 ma1 constant  
 1.6764 -0.8034 -0.6896 2.5623  
s.e. 0.1111 0.0928 0.1492 0.1161  
  
sigma^2 estimated as 8.046: log likelihood=-141.57  
AIC=293.13 AICc=294.29 BIC=303.43

The given ARIMA(2,0,1) model with a mean includes autoregressive (AR) and moving average (MA) components. Specifically, it is defined by the following equations:

knitr::include\_graphics("model.png")



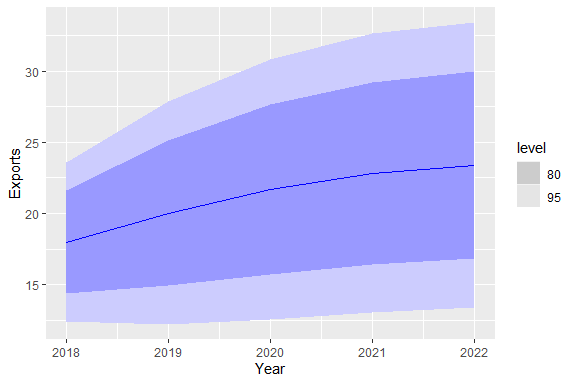
Here:

* Y\_t​ represents the exports at time t.
* The constant term is 2.5623.
* The autoregressive component includes lagged values Yt−1​ and Yt−2​ with coefficients 1.6764 and -0.8034, respectively.
* The moving average component includes the lagged error term ϵt−1​ with a coefficient of -0.6896.
* e\_t is the white noise error term at time t.
* The variance of the error term sigma square is estimated as 8.046.

The Akaike Information Criterion (AIC) is a measure of the model’s goodness of fit, and in this case, AIC=293.13. AICc and BIC are corrected versions of AIC to account for sample size, with AICc=294.29 and BIC=303.43. These metrics can be used for model comparison, where lower values indicate a better fit. The log-likelihood is -141.57, representing the maximized value of the likelihood function given the data and model parameters. Overall, the ARIMA(2,0,1) model provides a statistical description of the exports time series, incorporating lagged values and error terms for forecasting and analysis.

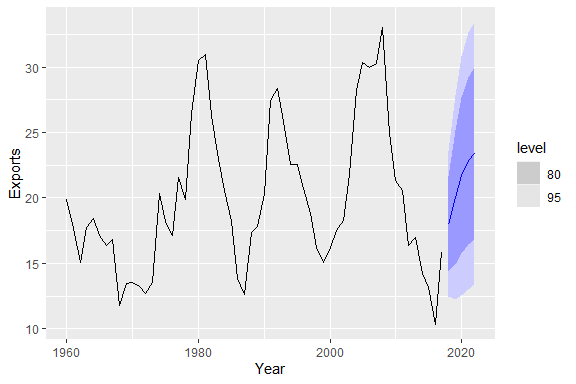
### Plot the Forecast for the Next Five Periods

fit |>   
 forecast(h=5) |>   
 autoplot()



### Plot together with the data

fit |>   
 forecast(h=5) |>   
 autoplot(egy\_economy)



## Understanding ARIMA models

* If 𝑐 = 0 and 𝑑 = 0, the long-term forecasts will go to zero.
* If 𝑐 = 0 and 𝑑 = 1, the long-term forecasts will go to a non-zero constant.
* If 𝑐 = 0 and 𝑑 = 2, the long-term forecasts will follow a straight line.
* If 𝑐 ≠ 0 and 𝑑 = 0, the long-term forecasts will go to the mean of the data.
* If 𝑐 ≠ 0 and 𝑑 = 1, the long-term forecasts will follow a straight line.
* If 𝑐 ≠ 0 and 𝑑 = 2, the long-term forecasts will follow a quadratic trend.

### Forecast variance and 𝑑

The higher the value of 𝑑, the more rapidly the prediction intervals increase in size. For 𝑑 = 0, the long-term forecast standard deviation will go to the standard deviation of the historical data. Most importantly is understanding how ARIMA works.

## SEASONAL ARIMA MODEL

| ARIMA | P, d, q | (P, D, Q)\_m |
| --- | --- | --- |
|  | Non -seasonal Part of the model | Seasonal part of the model |

### Note!!

For monthly data, m =12, quarterly data m = 4, and for annual data m = 1.

* 𝑚 = number of observations per year.
* 𝑑 first differences, 𝐷 seasonal differences
* 𝑝 AR lags, 𝑞 MA lags
* 𝑃 seasonal AR lags, 𝑄 seasonal MA lags

The first difference (d) is the difference between adjacent observations, i.e, the value for this month minus the one for the month before. On the other, the difference denoted as (D), if the difference between seasons. i.e, the value for January 2023 minus the value for January 2022 and so on. Seasonal and non-seasonal terms combine multiplicatively

### ARIMA MODEL ON PBS DATA SET

## Monthly Medicare Australia prescription data

### Description

PBS is a monthly tsibble with two values:

|  |  |
| --- | --- |
| Scripts: | Total number of scripts |
| Cost: | Cost of the scripts in $AUD |
|  |  |

### Format

Time series of class tsibble

### Details

The data is disaggregated using four keys:

|  |  |
| --- | --- |
| Concession: | Concessional scripts are given to pensioners, unemployed, dependents, and other card holders |
| Type: | Co-payments are made until an individual’s script expenditure hits a threshold ($290.00 for concession, $1141.80 otherwise). Safety net subsidies are provided to individuals exceeding this amount. |
| ATC1: | Anatomical Therapeutic Chemical index (level 1) |
| ATC2: | Anatomical Therapeutic Chemical index (level 2) |
|  |  |

### Source

Medicare Australia

### View the Pharmaceutical Data

PBS

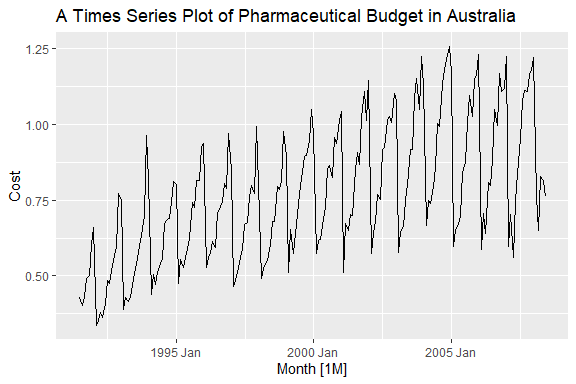
# A tsibble: 67,596 x 9 [1M]  
# Key: Concession, Type, ATC1, ATC2 [336]  
 Month Concession Type ATC1 ATC1\_desc ATC2 ATC2\_desc Scripts Cost  
 <mth> <chr> <chr> <chr> <chr> <chr> <chr> <dbl> <dbl>  
 1 1991 Jul Concessional Co-payme… A Alimenta… A01 STOMATOL… 18228 67877  
 2 1991 Aug Concessional Co-payme… A Alimenta… A01 STOMATOL… 15327 57011  
 3 1991 Sep Concessional Co-payme… A Alimenta… A01 STOMATOL… 14775 55020  
 4 1991 Oct Concessional Co-payme… A Alimenta… A01 STOMATOL… 15380 57222  
 5 1991 Nov Concessional Co-payme… A Alimenta… A01 STOMATOL… 14371 52120  
 6 1991 Dec Concessional Co-payme… A Alimenta… A01 STOMATOL… 15028 54299  
 7 1992 Jan Concessional Co-payme… A Alimenta… A01 STOMATOL… 11040 39753  
 8 1992 Feb Concessional Co-payme… A Alimenta… A01 STOMATOL… 15165 54405  
 9 1992 Mar Concessional Co-payme… A Alimenta… A01 STOMATOL… 16898 61108  
10 1992 Apr Concessional Co-payme… A Alimenta… A01 STOMATOL… 18141 65356  
# ℹ 67,586 more rows

### Filter the Data and Summarize (Data Preparations)

h02 <- PBS |>   
 filter(ATC2 == "H02") |>   
 summarise(Cost = sum(Cost)/ 1e6)  
h02

# A tsibble: 204 x 2 [1M]  
 Month Cost  
 <mth> <dbl>  
 1 1991 Jul 0.430  
 2 1991 Aug 0.401  
 3 1991 Sep 0.432  
 4 1991 Oct 0.493  
 5 1991 Nov 0.502  
 6 1991 Dec 0.603  
 7 1992 Jan 0.660  
 8 1992 Feb 0.336  
 9 1992 Mar 0.351  
10 1992 Apr 0.380  
# ℹ 194 more rows

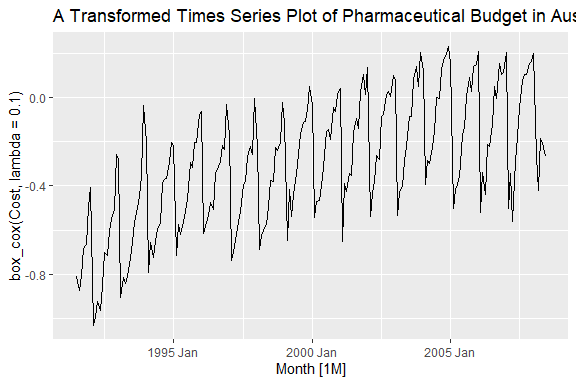
h02 |>   
 autoplot(Cost)+  
 labs(title = "A Times Series Plot of Pharmaceutical Budget in Australia")



We have an increasing trends with seasonality. As trend increases, seasonality increases as well.

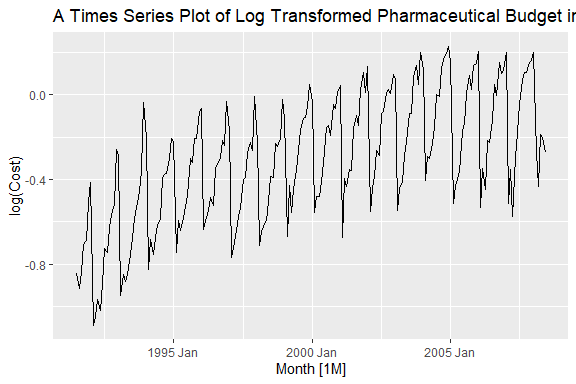
### Plot the Tranformed Series

h02 |>   
 autoplot(box\_cox(Cost, lambda = .1))+  
 labs(title = "A Transformed Times Series Plot of Pharmaceutical Budget in Australia")



### Alternatively

h02 |>   
 autoplot(log(Cost))+  
 labs(title = "A Times Series Plot of Log Transformed Pharmaceutical Budget in Australia")



Taking the logarithm of time series data is a common practice in various fields for several reasons:

1. **Stabilizing Variance:** In many time series analyses, the variance of the data may not be constant over time, leading to heteroscedasticity. Taking the logarithm often helps stabilize the variance, making the data more suitable for statistical techniques that assume homoscedasticity. This is particularly important in financial and economic data where the variability of measurements tends to increase with their level.
2. **Linearizing Trends:** The logarithmic transformation can linearize exponential trends in the data. For example, if the underlying process exhibits exponential growth or decay, taking the logarithm converts it into a linear trend, making it easier to model and interpret. Linear models are often more convenient for analysis and forecasting.
3. **Interpretable Percentage Changes:** When dealing with economic or financial data, the logarithmic transformation converts absolute changes into percentage changes. This is especially useful when comparing the relative impact of changes over time, providing a more interpretable measure of proportional differences.
4. **Normalizing Skewed Distributions:** Logarithmic transformations are effective in normalizing positively skewed distributions. This can be beneficial when applying statistical methods that assume normality, such as linear regression.
5. **Additivity of Logarithms:** The logarithm of the product of two variables is equal to the sum of their logarithms log(ab)=log(a)+log(b)). This property can simplify multiplicative relationships in time series data, converting them into additive relationships that are easier to work with.

However, it’s important to note that while log-transforming time series data has its advantages, it may not be appropriate for all datasets. The choice to use logarithms should be based on the characteristics of the data and the goals of the analysis. Additionally, interpretation of results from log-transformed data should consider the transformation applied during analysis.

## Note!!!

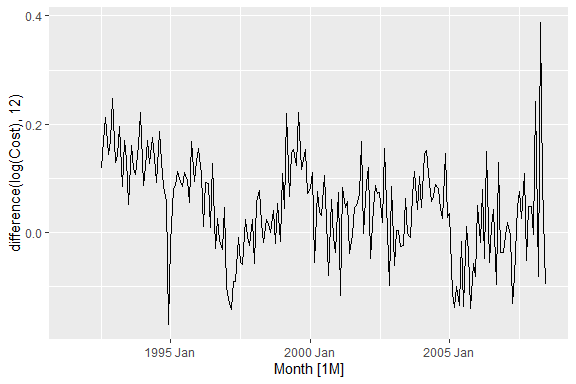
Pay attention on the first two.

### Difference the Series

Here we use the function, differencing as shown below

### Seasonal Difference

h02 |>   
 autoplot(log(Cost) |>   
 difference(12))



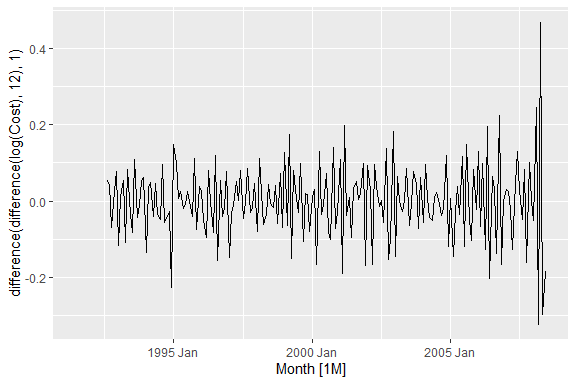
We still have some sort of patterns that we can see. So we need to take another difference and this time it will be meant to stabilize the mean and variance. The first difference is to remove seasonality and the second one is to stabilize the mean and variance to make it stationary.

### Stabilize the Variance by Taking the Normal Difference

### Note!!

It is recommended that you take the seasonal difference (quarter, m = 4, monthly, m = 12 and so on) and then take the first difference.

h02 |>   
 autoplot(log(Cost) |>   
 difference(12) |>   
 difference(1))



### Stationarity test (kpss function)

h02 |>   
 features(Cost, unitroot\_kpss)

# A tibble: 1 × 2  
 kpss\_stat kpss\_pvalue  
 <dbl> <dbl>  
1 2.60 0.01

### Null and Alternative hypothesis

| Hypotheses | Statement |
| --- | --- |
| Null | The time series is stationary |
| Alternative | The time series is not stationary |
|  |  |

From the results above, the times series is not stationary since the p-value is less than 0.05.

### Test How many Differencing we need to make the series stationary

h02 |>   
 features(Cost, unitroot\_ndiffs)

# A tibble: 1 × 1  
 ndiffs  
 <int>  
1 1

The results above shows that we need one differencing to make the series stationary. Now, how many seasonal differencing we need to make the series stationary

h02 |>   
 features(Cost, unitroot\_nsdiffs)

# A tibble: 1 × 1  
 nsdiffs  
 <int>  
1 1

Similarly, the results shows that we need only seasonal differencing to make the series stationary. Lets us now test stationarity after taking one seasonal differencing and one normal differencing.

### Test with a Differenced Series

h02 |>   
 features(log(Cost) |>   
 difference(12) |>   
 difference(1),unitroot\_kpss)

# A tibble: 1 × 2  
 kpss\_stat kpss\_pvalue  
 <dbl> <dbl>  
1 0.0236 0.1

The results shows that the series is stationary after the seasonal difference and first difference. Remember, in this case, differencing is done after series transformation.

## ACF and PACF

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are essential tools in time series analysis, particularly in the context of building and diagnosing autoregressive integrated moving average (ARIMA) models. These models are commonly used to understand and forecast time series data.

### **Autocorrelation Function (ACF):**

* **Definition:** ACF measures the correlation between a time series and its lagged values.
* **Interpretation:** Peaks in the ACF plot indicate significant relationships between the time series and its past values.
* **Use in determining parameters (p, d, q):**
  + **p (Autoregressive Order):** The lag value where the ACF plot crosses the upper confidence interval for the first time. This suggests the number of autoregressive terms needed to capture the underlying patterns.
  + **q (Moving Average Order):** The lag value where the ACF plot crosses the lower confidence interval for the first time. This indicates the number of lagged forecast errors needed to capture the moving average components.

### **Partial Autocorrelation Function (PACF):**

* **Definition:** PACF measures the correlation between a time series and its lagged values, excluding the effects of intermediate lags.
* **Interpretation:** Peaks in the PACF plot indicate significant relationships between the time series and its past values, removing the influence of intervening lags.
* **Use in determining parameters (p, d, q):**
  + **p (Autoregressive Order):** The lag value where the PACF plot crosses the upper confidence interval for the first time after a sharp drop. This helps identify the number of significant autoregressive terms.
  + **q (Moving Average Order):** Similar to ACF, the lag value where the PACF plot crosses the lower confidence interval for the first time. This identifies the number of significant lagged forecast errors.

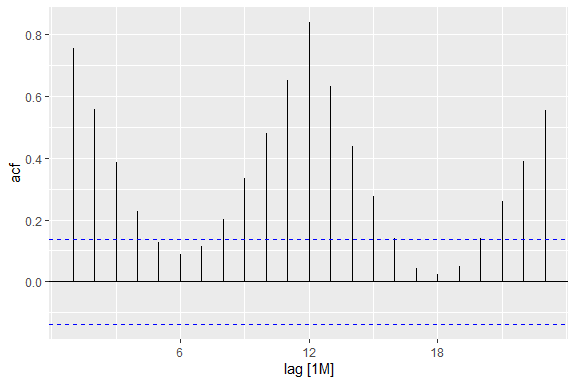
### **General Guidelines:**

* **p (Autoregressive Order):** Look for the lag value at which the ACF or PACF sharply cuts off after a significant number of lags.
* **d (Differencing Order):** The differencing order is determined by the number of times the series needs to be differenced to achieve stationarity (constant mean and variance).
* **q (Moving Average Order):** Examine the lag value at which the ACF or PACF sharply cuts off after a significant number of lags.

By analyzing ACF and PACF plots and identifying significant lag values, analysts can make informed decisions about the order of autoregressive, differencing, and moving average components in ARIMA models, contributing to more accurate and reliable time series forecasts.

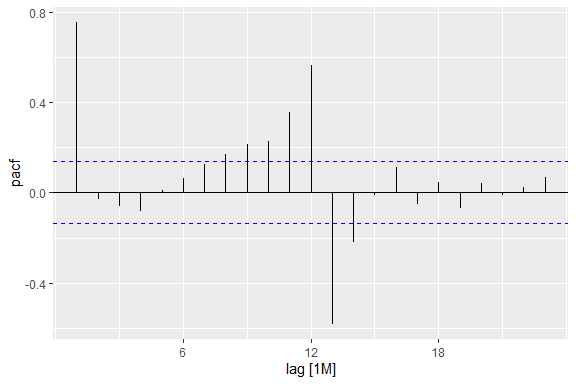
### Autocorrelation Function (ACF)

h02 |>   
 ACF(Cost) |>   
 autoplot()



### Parctial Autocorrelation Function

h02 |>   
 PACF(Cost) |>   
 autoplot()



### Fit the ARIMA Model

fit <- h02 |>   
 model(arima= ARIMA(log(Cost)))  
fit

# A mable: 1 x 1  
 arima  
 <model>  
1 <ARIMA(2,1,0)(0,1,1)[12]>

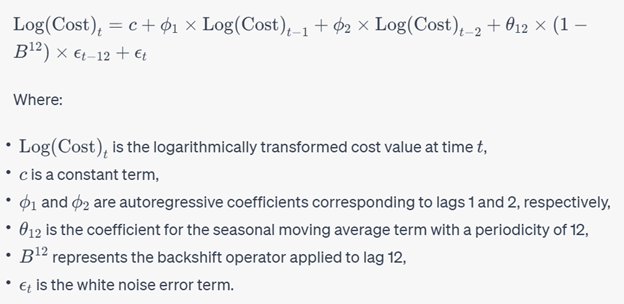
### View the Model using the report() Function

report(fit)

Series: Cost   
Model: ARIMA(2,1,0)(0,1,1)[12]   
Transformation: log(Cost)   
  
Coefficients:  
 ar1 ar2 sma1  
 -0.8491 -0.4207 -0.6401  
s.e. 0.0712 0.0714 0.0694  
  
sigma^2 estimated as 0.004387: log likelihood=245.39  
AIC=-482.78 AICc=-482.56 BIC=-469.77

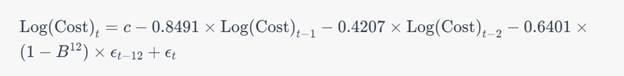
### The Model Specified

knitr::include\_graphics("sarima.png")



### Actual Model

knitr::include\_graphics("mod.png")



The given information presents the results of an ARIMA (AutoRegressive Integrated Moving Average) model with seasonal components applied to a time series of cost values after a logarithmic transformation. Here’s a breakdown of the key elements:

1. **ARIMA Model Specification:**
   * The model is specified as ARIMA(2,1,0)(0,1,1)[12].
   * The non-seasonal part is denoted by (2,1,0), indicating two autoregressive (AR) terms, one differencing (I) order, and no moving average (MA) terms.
   * The seasonal part is denoted by (0,1,1)[12], indicating no seasonal autoregressive terms, one seasonal differencing order, and one seasonal moving average term with a periodicity of 12 (denoted by [12]).
2. **Transformation:**
   * The series “Cost” has undergone a logarithmic transformation, likely to stabilize variance and make the series more amenable to modeling.
3. **Model Coefficients:**
   * **AR Coefficients:** The autoregressive coefficients are estimated as -0.8491 for the first lag (ar1) and -0.4207 for the second lag (ar2).
   * **SMA Coefficient:** The seasonal moving average coefficient (sma1) is estimated as -0.6401.
   * **Standard Errors (s.e.):** These values represent the standard errors associated with each coefficient estimate.
4. **Model Performance:**
   * **Sigma^2:** The estimated variance of the residuals is 0.004387.
   * **Log Likelihood:** The log-likelihood of the model is 245.39.
   * **Information Criteria:**
     + AIC (Akaike Information Criterion): -482.78
     + AICc (Corrected AIC): -482.56
     + BIC (Bayesian Information Criterion): -469.77

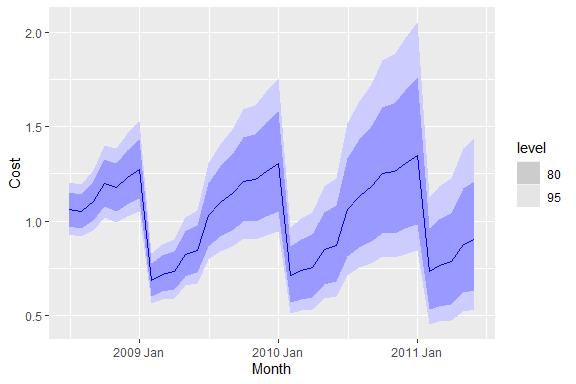
### **Interpretation:**

* The negative AR coefficients suggest a declining impact of past observations on the current value, and the negative SMA coefficient indicates a decreasing effect of seasonal deviations on the current value.
* The small standard errors indicate a relatively precise estimation of the coefficients.
* The information criteria (AIC, AICc, BIC) provide a measure of the model’s goodness of fit and complexity, with lower values indicating a better-fitting model. In this case, the AICc and BIC values are close, suggesting a parsimonious model that adequately fits the data.

This ARIMA model with logarithmic transformation seems to capture the temporal patterns and seasonality in the cost time series data effectively.

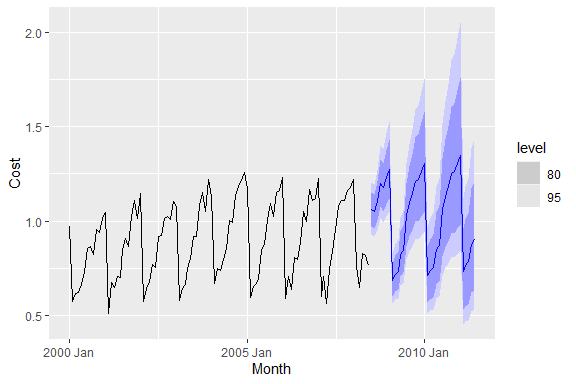
### Pot the Forecast

fit|>   
 forecast(h= "3 years")|>  
 autoplot()



### Plot the Forecast together with data

fit|>   
 forecast(h= "3 years")|>  
 autoplot(filter\_index(h02, "2000 Jan" ~ .))



## Ensemble / Forecast Combination

An ensemble model in time series forecasting is a technique that combines the predictions of multiple individual forecasting models to improve the overall accuracy and reliability of the forecast. Time series data involves patterns, trends, and seasonality that can be challenging to capture with a single forecasting model, and ensemble methods aim to mitigate the weaknesses of individual models by leveraging the strengths of different approaches.

Ensemble models can be particularly effective when dealing with time series data because they can capture various aspects of the underlying patterns and reduce the risk of making poor forecasts due to the limitations of a single model. However, it’s important to carefully select and fine-tune the individual models within the ensemble and consider factors like model diversity, model performance, and the specific characteristics of the time series data being forecasted.

## Tourism data

## Australian domestic overnight trips

### Description

A dataset containing the quarterly overnight trips from 1998 Q1 to 2016 Q4 across Australia.

### Usage

tourism

### Format

A tsibble with 23,408 rows and 5 variables:

* **Quarter**: Year quarter (index)
* **Region**: The tourism regions are formed through the aggregation of Statistical Local Areas (SLAs) which are defined by the various State and Territory tourism authorities according to their research and marketing needs
* **State**: States and territories of Australia
* **Purpose**: Stopover purpose of visit:
  + “Holiday”
  + “Visiting friends and relatives”
  + “Business”
  + “Other reason”
* **Trips**: Overnight trips in thousands

### References

[Tourism Research Australia](https://www.tra.gov.au/)

### Prepare data for Forecasting

tourism

# A tsibble: 24,320 x 5 [1Q]  
# Key: Region, State, Purpose [304]  
 Quarter Region State Purpose Trips  
 <qtr> <chr> <chr> <chr> <dbl>  
 1 1998 Q1 Adelaide South Australia Business 135.  
 2 1998 Q2 Adelaide South Australia Business 110.  
 3 1998 Q3 Adelaide South Australia Business 166.  
 4 1998 Q4 Adelaide South Australia Business 127.  
 5 1999 Q1 Adelaide South Australia Business 137.  
 6 1999 Q2 Adelaide South Australia Business 200.  
 7 1999 Q3 Adelaide South Australia Business 169.  
 8 1999 Q4 Adelaide South Australia Business 134.  
 9 2000 Q1 Adelaide South Australia Business 154.  
10 2000 Q2 Adelaide South Australia Business 169.  
# ℹ 24,310 more rows

### Summarized Data

aus\_tourism <- tourism |>   
 index\_by(Quarter) |>   
 summarise(Trips = sum(Trips))  
aus\_tourism

# A tsibble: 80 x 2 [1Q]  
 Quarter Trips  
 <qtr> <dbl>  
 1 1998 Q1 23182.  
 2 1998 Q2 20323.  
 3 1998 Q3 19827.  
 4 1998 Q4 20830.  
 5 1999 Q1 22087.  
 6 1999 Q2 21458.  
 7 1999 Q3 19914.  
 8 1999 Q4 20028.  
 9 2000 Q1 22339.  
10 2000 Q2 19941.  
# ℹ 70 more rows

#### Ensemble Model

fit <- aus\_tourism |> model(  
 snaive = SNAIVE(Trips),  
 ets= ETS(Trips),  
 arima = ARIMA(Trips),  
 regression = TSLM(Trips ~ trend() + season()),  
 ) |>   
 mutate(combination = (regression+snaive+ets+arima)/4)

### Augment Function

fit|>  
augment()

# A tsibble: 400 x 6 [1Q]  
# Key: .model [5]  
 .model Quarter Trips .fitted .resid .innov  
 <chr> <qtr> <dbl> <dbl> <dbl> <dbl>  
 1 snaive 1998 Q1 23182. NA NA NA   
 2 snaive 1998 Q2 20323. NA NA NA   
 3 snaive 1998 Q3 19827. NA NA NA   
 4 snaive 1998 Q4 20830. NA NA NA   
 5 snaive 1999 Q1 22087. 23182. -1095. -1095.   
 6 snaive 1999 Q2 21458. 20323. 1135. 1135.   
 7 snaive 1999 Q3 19914. 19827. 87.6 87.6  
 8 snaive 1999 Q4 20028. 20830. -802. -802.   
 9 snaive 2000 Q1 22339. 22087. 252. 252.   
10 snaive 2000 Q2 19941. 21458. -1517. -1517.   
# ℹ 390 more rows

### View the Model (Select a specific model of interest)

fit|>  
 select(combination)|>  
 report()

Series: Trips   
Model: COMBINATION   
Combination: (Trips + Trips + Trips + Trips) \* 0.25  
  
===================================================  
  
Series: Trips + Trips + Trips + Trips   
Model: COMBINATION   
Combination: Trips + Trips + Trips + Trips  
  
==========================================  
  
Series: Trips + Trips + Trips   
Model: COMBINATION   
Combination: Trips + Trips + Trips  
  
==================================  
  
Series: Trips + Trips   
Model: COMBINATION   
Combination: Trips + Trips  
  
==========================  
  
Series: Trips   
Model: TSLM   
  
Residuals:  
 Min 1Q Median 3Q Max   
-3119.1 -1618.4 308.3 1030.5 4264.0   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 20895.702 518.741 40.282 < 2e-16 \*\*\*  
trend() 46.749 8.587 5.444 6.32e-07 \*\*\*  
season()year2 -1603.189 560.294 -2.861 0.005463 \*\*   
season()year3 -2036.754 560.491 -3.634 0.000509 \*\*\*  
season()year4 -1306.052 560.820 -2.329 0.022565 \*   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 1772 on 75 degrees of freedom  
Multiple R-squared: 0.3651, Adjusted R-squared: 0.3312  
F-statistic: 10.78 on 4 and 75 DF, p-value: 5.8841e-07  
  
Series: Trips   
Model: SNAIVE   
  
sigma^2: 1397413.9063   
  
  
Series: Trips   
Model: ETS(A,A,A)   
 Smoothing parameters:  
 alpha = 0.4495675   
 beta = 0.04450178   
 gamma = 0.0001000075   
  
 Initial states:  
 l[0] b[0] s[0] s[-1] s[-2] s[-3]  
 21689.64 -58.46946 -125.8548 -816.3416 -324.5553 1266.752  
  
 sigma^2: 699901.4  
  
 AIC AICc BIC   
1436.829 1439.400 1458.267   
  
  
Series: Trips   
Model: ARIMA(0,1,1)(0,1,1)[4]   
  
Coefficients:  
 ma1 sma1  
 -0.4950 -0.8369  
s.e. 0.1043 0.1087  
  
sigma^2 estimated as 772368: log likelihood=-616.4  
AIC=1238.8 AICc=1239.14 BIC=1245.76

### Forecast

fcst <- fit |>   
 forecast(h=12)   
fcst

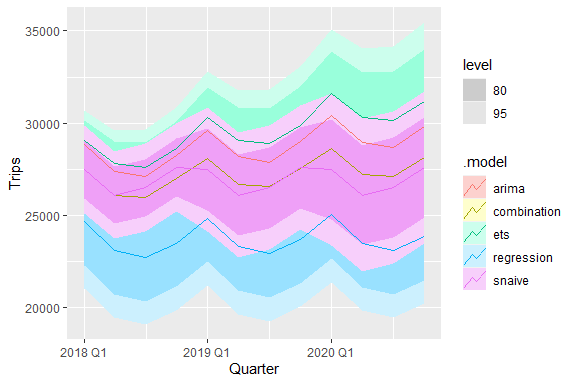
# A fable: 60 x 4 [1Q]  
# Key: .model [5]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
 1 snaive 2018 Q1 N(27496, 1475025) 27496.  
 2 snaive 2018 Q2 N(26114, 1475025) 26114.  
 3 snaive 2018 Q3 N(26506, 1475025) 26506.  
 4 snaive 2018 Q4 N(27594, 1475025) 27594.  
 5 snaive 2019 Q1 N(27496, 3e+06) 27496.  
 6 snaive 2019 Q2 N(26114, 3e+06) 26114.  
 7 snaive 2019 Q3 N(26506, 3e+06) 26506.  
 8 snaive 2019 Q4 N(27594, 3e+06) 27594.  
 9 snaive 2020 Q1 N(27496, 4425075) 27496.  
10 snaive 2020 Q2 N(26114, 4425075) 26114.  
# ℹ 50 more rows

### View the Forecast using View() Function

View(fcst)

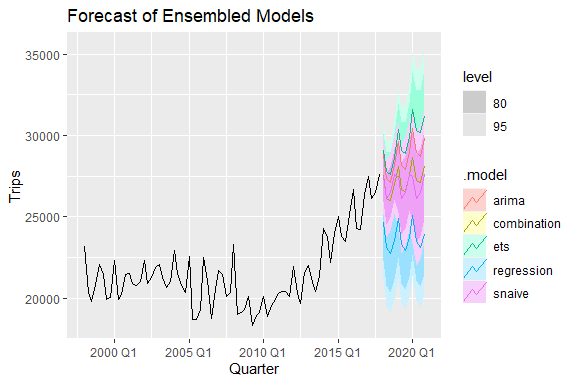
### Plot the Forecast

fcst|>   
 autoplot()



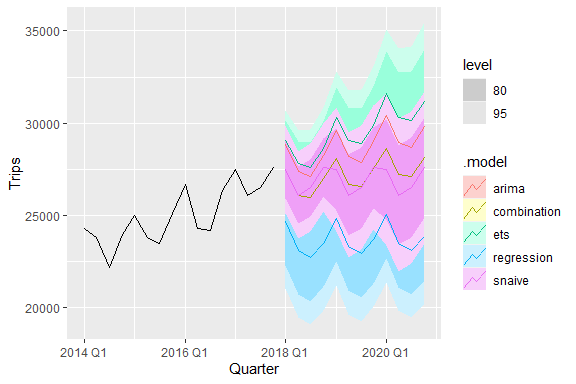
### Plot the Forecast together with the data

fcst|>   
 autoplot(aus\_tourism)+  
 labs(title = "Forecast of Ensembled Models")



### Plot the Forecast with a Section of data of interest

fcst|>   
 autoplot(filter\_index(aus\_tourism, "2014 Jan" ~ .))



## Note!!

After estimating the models, when predicting, everything else goes back to the original state.