Day Five: Principles of Time Series Forecasting

Lumumba Wandera Victor

2023-11-16

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

## Day Five: Principles of Time Series Forecasting

### Basic Training and Test Accuracy

#### Evaluating forecast accuracy

In order to evaluate the performance of a forecasting model, we compute its forecast accuracy. Forecast accuracy is compared by measuring errors based on the test set. Ideally it should allow comparing benefits from improved accuracy with the cost of obtaining the improvement.

We should be choosing forecast models that lead to better business decisions

▶ least staffing costs, least emission, highest service level, least stock-out, least inventory, fastest response, least change in planing, for example.

However, this is not always easy to obtain, therefore we might simply use methods that provide the most accurate forecast.

### In-sample (training) vs. out-of-sample (test)

Fitting and its residual are not a reliable indication of forecast accuracy. Besides, a model which fits the training data well will not necessarily forecast well. A perfect fit can always be obtained by using a model with enough parameters. On the other hand, over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data. In testing the accuracy of the model, we mimic the real life situation and pretend that we don’t know some part of data (new data). Remember tt must not be used for any aspect of model training. Forecast accuracy is computed only based on the test set

### Evaluating point forecast accuracy

#### Load the following libraries

library(fpp3)

── Attaching packages ────────────────────────────────────────────── fpp3 0.5 ──

✔ tibble 3.2.1 ✔ tsibble 1.1.3  
✔ dplyr 1.1.2 ✔ tsibbledata 0.4.1  
✔ tidyr 1.3.0 ✔ feasts 0.3.1  
✔ lubridate 1.9.2 ✔ fable 0.3.3  
✔ ggplot2 3.4.3 ✔ fabletools 0.3.3

── Conflicts ───────────────────────────────────────────────── fpp3\_conflicts ──  
✖ lubridate::date() masks base::date()  
✖ dplyr::filter() masks stats::filter()  
✖ tsibble::intersect() masks base::intersect()  
✖ tsibble::interval() masks lubridate::interval()  
✖ dplyr::lag() masks stats::lag()  
✖ tsibble::setdiff() masks base::setdiff()  
✖ tsibble::union() masks base::union()

library(ggdist)  
library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ forcats 1.0.0 ✔ readr 2.1.4  
✔ purrr 1.0.1 ✔ stringr 1.5.0

── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ tsibble::interval() masks lubridate::interval()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

### Load 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/)

### 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

### Index by Time and Summarize Total Trips

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

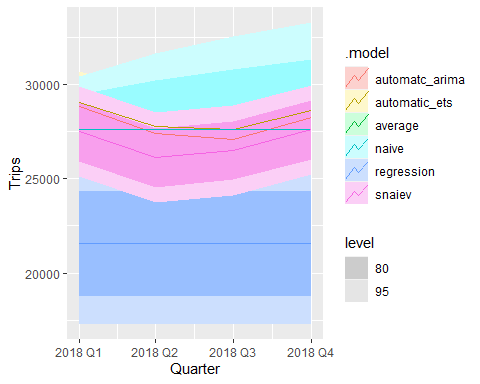
# A tsibble: 80 x 2 [1Q]  
 Quarter Trips  
 <qtr> <dbl>  
 1 1998 Q1 23182.  
 2 1998 Q2 20323.  
 3 1998 Q3 19827.  
 4 1998 Q4 20830.  
 5 1999 Q1 22087.  
 6 1999 Q2 21458.  
 7 1999 Q3 19914.  
 8 1999 Q4 20028.  
 9 2000 Q1 22339.  
10 2000 Q2 19941.  
# ℹ 70 more rows

### Fit the Various Models

fit <- aus\_tourism |>  
 model(average = MEAN(Trips),  
 naive = NAIVE(Trips),  
 snaiev = SNAIVE(Trips),  
 regression = TSLM(Trips),  
 automatic\_ets = ETS(Trips),  
 automatc\_arima = ARIMA(Trips)  
 )

### Plot the Forecast (pay attention on regression)

fit |>   
 forecast(h = 4)|>  
 autoplot()

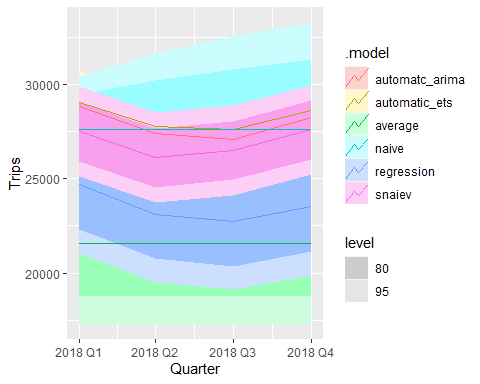


### Estimate the Forecast Model and add trend and seasonal component to it

fit <- aus\_tourism |>  
 model(average = MEAN(Trips),  
 naive = NAIVE(Trips),  
 snaiev = SNAIVE(Trips),  
 regression = TSLM(Trips ~ trend() + season()),  
 automatic\_ets = ETS(Trips),  
 automatc\_arima = ARIMA(Trips)  
 )

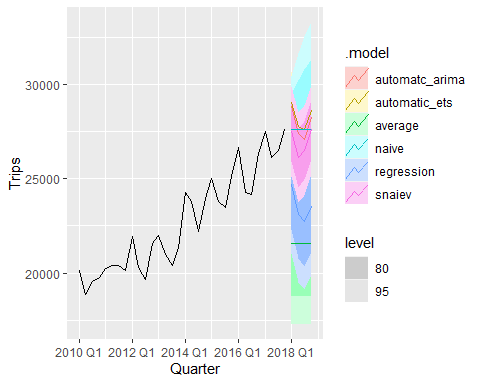
### Plot the Forecast

fit |>   
 forecast(h = 4)|>  
 autoplot()



### Plot the Forecast with the data

fit |>   
 forecast(h = 4)|>  
 autoplot(filter\_index(aus\_tourism, "2010 Q1"~.))



### Various Function to View the Estimated Mode include the Following

1. tidy()
2. glance()
3. report()
4. components()
5. augment()

#### Apply the Function to Navigate the Model

#### tidy() function

fit |>   
 select(automatic\_ets) |>   
 tidy()

# A tibble: 9 × 3  
 .model term estimate  
 <chr> <chr> <dbl>  
1 automatic\_ets alpha 0.450   
2 automatic\_ets beta 0.0445   
3 automatic\_ets gamma 0.000100  
4 automatic\_ets l[0] 21690.   
5 automatic\_ets b[0] -58.5   
6 automatic\_ets s[0] -126.   
7 automatic\_ets s[-1] -816.   
8 automatic\_ets s[-2] -325.   
9 automatic\_ets s[-3] 1267.

#### glance() function

fit |>   
 select(automatic\_ets) |>   
 glance()

# A tibble: 1 × 9  
 .model sigma2 log\_lik AIC AICc BIC MSE AMSE MAE  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 automatic\_ets 699901. -709. 1437. 1439. 1458. 629911. 790554. 604.

fit |>   
 select(automatic\_ets) |>   
 report()

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

The output above outlines a time series forecasting model under the ETS(A,A,A) framework, representing an Exponential Smoothing State Space model with additive errors and additive trend and seasonality components. The model is characterized by three smoothing parameters: alpha, beta, and gamma, set to 0.4495675, 0.04450178, and 0.0001000075, respectively. These parameters govern the weights assigned to the current observation, trend, and seasonal components during the forecasting process.

The initial states of the model include level (l), trend (b), and seasonal (s) components, with specific values provided for each. These initial states influence the starting point of the forecasting process. Notably, the sigma^2 value of 699901.4 represents the estimated variance of the error term in the model.

The output has the three model evaluation metrics: AIC (Akaike Information Criterion), AICc (corrected AIC), and BIC (Bayesian Information Criterion). These metrics serve as measures of the model’s goodness of fit, with lower values indicating a better fit. In this case, the AIC, AICc, and BIC values are 1436.829, 1439.400, and 1458.267, respectively. Model selection would involve choosing the model with the lowest value among these criteria. The provided values suggest a reasonably good fit, but further comparison with alternative models and consideration of the specific context of the time series data would be essential for a comprehensive interpretation and decision-making.

The coefficients and parameters in the output pertain to an Exponential Smoothing State Space model with additive errors, additive trend, and additive seasonality (ETS(A,A,A)). Let’s break down the interpretation of each:

1. **Smoothing Parameters:**
   * **Alpha (α):** Represents the smoothing parameter for the level (l) component. It determines the weight given to the most recent observation when updating the level.
   * **Beta (β):** The smoothing parameter for the trend (b) component. It controls the weight assigned to the most recent trend when updating.
   * **Gamma (γ):** The smoothing parameter for the seasonal (s) component. It governs the weight assigned to the most recent seasonal observation when updating the seasonal component.
2. **Initial States:**
   * **l[0]:** Initial level of the time series.
   * **b[0]:** Initial trend of the time series.
   * **s[0]:** Initial seasonal component.
   * **s[-1], s[-2], s[-3]:** Initial seasonal components for the previous three seasons.
3. **Sigma^2:**
   * **Sigma^2:** Represents the estimated variance of the error term in the model. In this context, it is 699901.4, indicating the variability or volatility of the residuals.
4. **Evaluation Metrics:**
   * **AIC (Akaike Information Criterion):** A measure of the model’s goodness of fit, balancing the trade-off between accuracy and complexity. Lower AIC values suggest a better-fitting model.
   * **AICc (Corrected AIC):** A modification of AIC for small sample sizes, penalizing the model for additional parameters.
   * **BIC (Bayesian Information Criterion):** Similar to AIC but with a stronger penalty for model complexity. Like AIC, lower BIC values indicate a better-fitting model.

In summary, these parameters and coefficients collectively define the structure of the ETS(A,A,A) model and provide information on how the model captures and updates the level, trend, and seasonality of the time series data. The smoothing parameters control the influence of recent observations on these components, while the initial states set the starting values. The evaluation metrics help assess the model’s overall fit, with lower values indicating a more parsimonious and effective model.

#### component() function

fit |>   
 select(automatic\_ets) |>   
 components()

# A dable: 84 x 7 [1Q]  
# Key: .model [1]  
# : Trips = lag(level, 1) + lag(slope, 1) + lag(season, 4) + remainder  
 .model Quarter Trips level slope season remainder  
 <chr> <qtr> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 automatic\_ets 1997 Q1 NA NA NA 1267. NA   
 2 automatic\_ets 1997 Q2 NA NA NA -325. NA   
 3 automatic\_ets 1997 Q3 NA NA NA -816. NA   
 4 automatic\_ets 1997 Q4 NA 21690. -58.5 -126. NA   
 5 automatic\_ets 1998 Q1 23182. 21759. -45.8 1267. 284.   
 6 automatic\_ets 1998 Q2 20323. 21234. -93.2 -325. -1065.   
 7 automatic\_ets 1998 Q3 19827. 20917. -115. -816. -498.   
 8 automatic\_ets 1998 Q4 20830. 20871. -109. -126. 154.   
 9 automatic\_ets 1999 Q1 22087. 20789. -106. 1267. 58.0  
10 automatic\_ets 1999 Q2 21458. 21177. -57.0 -325. 1100.   
# ℹ 74 more rows

#### Augment() Function

fit |>   
 select(automatic\_ets) |>   
 augment()

# A tsibble: 80 x 6 [1Q]  
# Key: .model [1]  
 .model Quarter Trips .fitted .resid .innov  
 <chr> <qtr> <dbl> <dbl> <dbl> <dbl>  
 1 automatic\_ets 1998 Q1 23182. 22898. 284. 284.   
 2 automatic\_ets 1998 Q2 20323. 21389. -1065. -1065.   
 3 automatic\_ets 1998 Q3 19827. 20325. -498. -498.   
 4 automatic\_ets 1998 Q4 20830. 20676. 154. 154.   
 5 automatic\_ets 1999 Q1 22087. 22029. 58.0 58.0  
 6 automatic\_ets 1999 Q2 21458. 20358. 1100. 1100.   
 7 automatic\_ets 1999 Q3 19914. 20304. -390. -390.   
 8 automatic\_ets 1999 Q4 20028. 20745. -717. -717.   
 9 automatic\_ets 2000 Q1 22339. 21709. 630. 630.   
10 automatic\_ets 2000 Q2 19941. 20323. -382. -382.   
# ℹ 70 more rows

#### Use the Model to Forecast the Next Four Seasons (Four quarters ahead)

fcst <-   
 fit |>   
 forecast(h = 4)  
fcst

# A fable: 24 x 4 [1Q]  
# Key: .model [6]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
 1 average 2018 Q1 N(21553, 4751393) 21553.  
 2 average 2018 Q2 N(21553, 4751393) 21553.  
 3 average 2018 Q3 N(21553, 4751393) 21553.  
 4 average 2018 Q4 N(21553, 4751393) 21553.  
 5 naive 2018 Q1 N(27594, 2115987) 27594.  
 6 naive 2018 Q2 N(27594, 4231973) 27594.  
 7 naive 2018 Q3 N(27594, 6347960) 27594.  
 8 naive 2018 Q4 N(27594, 8463947) 27594.  
 9 snaiev 2018 Q1 N(27496, 1475025) 27496.  
10 snaiev 2018 Q2 N(26114, 1475025) 26114.  
# ℹ 14 more rows

The output above shows the point focus and the distribution of the forecast for each model. We have the normal distribution with mean and variance as indicated. This is kind of focus assumes that residuals are following a normal distribution. If we assume that the residuals are not following normal distribution, then we have to use bootstrapping, as shown below.

### Bootstrap

fcst1 <- fit |>   
 forecast(h = 4, bootstrap = TRUE)  
fcst1

# A fable: 24 x 4 [1Q]  
# Key: .model [6]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
 1 average 2018 Q1 sample[5000] 21570.  
 2 average 2018 Q2 sample[5000] 21545.  
 3 average 2018 Q3 sample[5000] 21578.  
 4 average 2018 Q4 sample[5000] 21546.  
 5 naive 2018 Q1 sample[5000] 27592.  
 6 naive 2018 Q2 sample[5000] 27598.  
 7 naive 2018 Q3 sample[5000] 27591.  
 8 naive 2018 Q4 sample[5000] 27567.  
 9 snaiev 2018 Q1 sample[5000] 27507.  
10 snaiev 2018 Q2 sample[5000] 26130.  
# ℹ 14 more rows

From the results above, the bootstrap method takes 5000 samples to make the prediction.

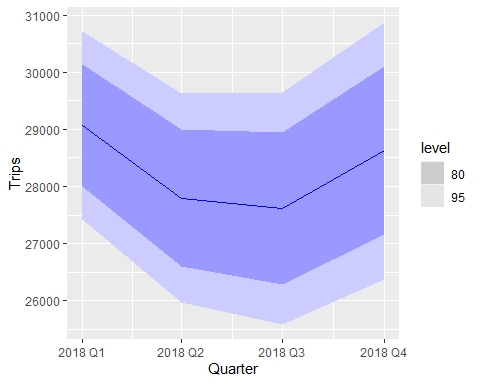
#### View the Forecast for the Automatic ETS

fcst\_ets <-   
 fcst |>   
 filter(.model == "automatic\_ets")  
fcst\_ets

# A fable: 4 x 4 [1Q]  
# Key: .model [1]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
1 automatic\_ets 2018 Q1 N(29068, 7e+05) 29068.  
2 automatic\_ets 2018 Q2 N(27794, 870750) 27794.  
3 automatic\_ets 2018 Q3 N(27619, 1073763) 27619.  
4 automatic\_ets 2018 Q4 N(28627, 1311711) 28627.

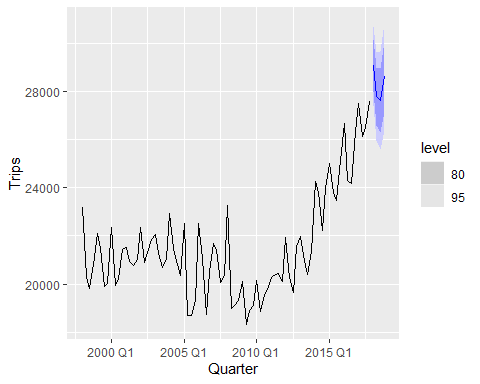
### Plot the Forecast

fcst\_ets|>  
 autoplot()



### Plot the Forecast together with the data

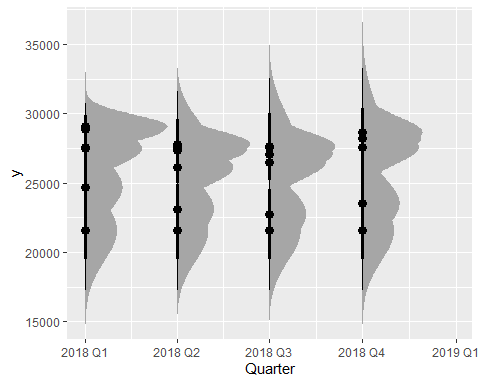
fcst\_ets|>  
 autoplot(aus\_tourism)



### Plot the Distribution of the Forecast for all the models

ggplot(data = fcst, mapping = aes(x = Quarter, ydist = Trips))+  
 stat\_halfeye()

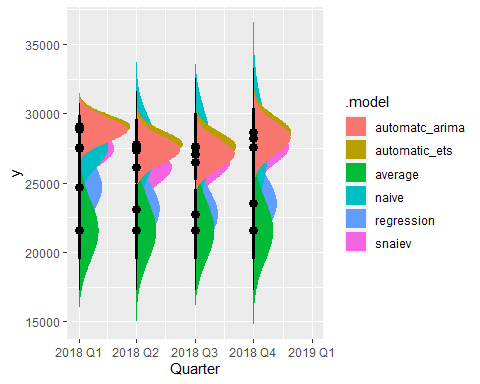
Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.



As you can see, the plot above shows the distribution of the focus for various models and you can see, the distribution has several peaks indicating various models. Let us add colors to distinguish distribution of the various forecast

ggplot(data = fcst, mapping = aes(x = Quarter, ydist = Trips, fill = .model))+  
 stat\_halfeye()

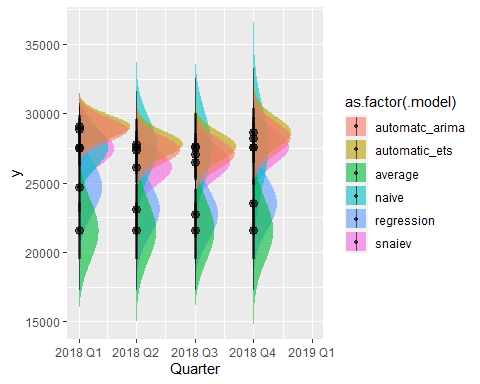
Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.



We can now see the forecast distribution for various models. Let us now view the distribution with overlapping view.

ggplot(data = fcst, mapping = aes(x = Quarter, ydist = Trips, fill = as.factor(.model)))+  
 stat\_halfeye(alpha = 0.6)

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.



From the plot above, ARIMA and ETS models proves to be the best. The distribution is very narrow, showing higher certainty in our forecast.

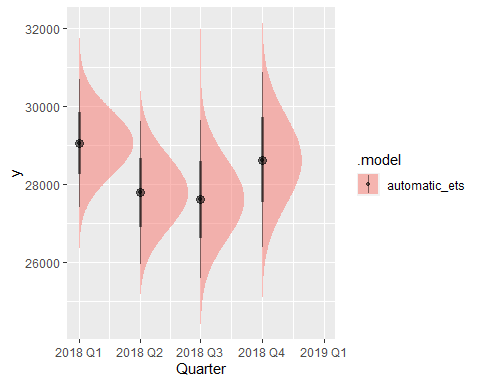
### Extract for ETS and Plot

fcst\_ets <-   
 fcst |>   
 filter(.model == "automatic\_ets")  
fcst\_ets

# A fable: 4 x 4 [1Q]  
# Key: .model [1]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
1 automatic\_ets 2018 Q1 N(29068, 7e+05) 29068.  
2 automatic\_ets 2018 Q2 N(27794, 870750) 27794.  
3 automatic\_ets 2018 Q3 N(27619, 1073763) 27619.  
4 automatic\_ets 2018 Q4 N(28627, 1311711) 28627.

ggplot(data = fcst\_ets, mapping = aes(x = Quarter, ydist = Trips, fill = .model))+  
 stat\_halfeye(alpha =.5)

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.



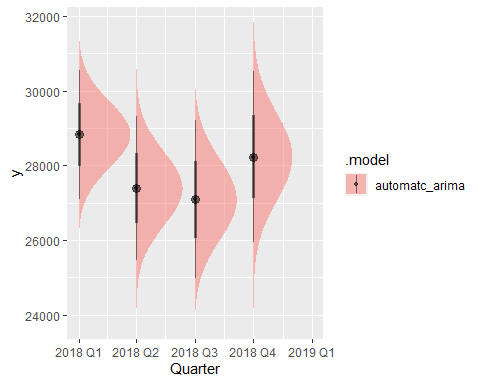
### Extract ARIMA Forecast and Plot

fcst\_arima <-   
 fcst |>   
 filter(.model == "automatc\_arima")  
fcst\_arima

# A fable: 4 x 4 [1Q]  
# Key: .model [1]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
1 automatc\_arima 2018 Q1 N(28832, 772625) 28832.  
2 automatc\_arima 2018 Q2 N(27383, 969648) 27383.  
3 automatc\_arima 2018 Q3 N(27092, 1166670) 27092.  
4 automatc\_arima 2018 Q4 N(28230, 1363693) 28230.

ggplot(data = fcst\_arima, mapping = aes(x = Quarter, ydist = Trips, fill = .model))+  
 stat\_halfeye(alpha = .5)

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.

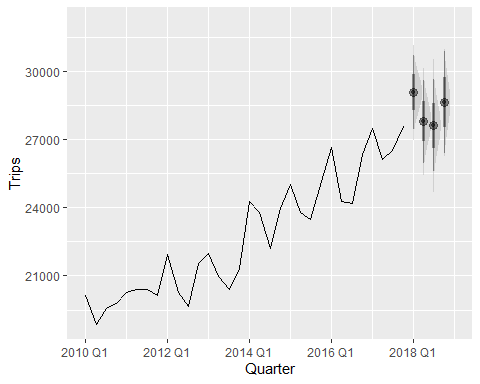


From the plot above, we can get the distribution interval and the probabilities as well.

ggplot(data = fcst\_ets, mapping = aes(x = Quarter, ydist = Trips))+  
 stat\_halfeye(alpha = .4)+  
 autolayer(filter\_index(aus\_tourism, "2010 Q1"~.))

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.

Plot variable not specified, automatically selected `.vars = Trips`



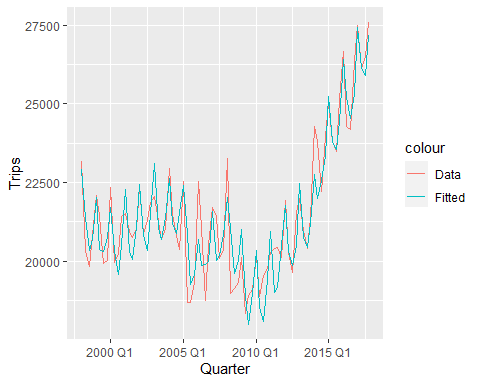
### get the fitted ets and plot with the data

fitted\_ets <- fit|>  
 select(automatic\_ets)|>  
 augment()  
fitted\_ets

# A tsibble: 80 x 6 [1Q]  
# Key: .model [1]  
 .model Quarter Trips .fitted .resid .innov  
 <chr> <qtr> <dbl> <dbl> <dbl> <dbl>  
 1 automatic\_ets 1998 Q1 23182. 22898. 284. 284.   
 2 automatic\_ets 1998 Q2 20323. 21389. -1065. -1065.   
 3 automatic\_ets 1998 Q3 19827. 20325. -498. -498.   
 4 automatic\_ets 1998 Q4 20830. 20676. 154. 154.   
 5 automatic\_ets 1999 Q1 22087. 22029. 58.0 58.0  
 6 automatic\_ets 1999 Q2 21458. 20358. 1100. 1100.   
 7 automatic\_ets 1999 Q3 19914. 20304. -390. -390.   
 8 automatic\_ets 1999 Q4 20028. 20745. -717. -717.   
 9 automatic\_ets 2000 Q1 22339. 21709. 630. 630.   
10 automatic\_ets 2000 Q2 19941. 20323. -382. -382.   
# ℹ 70 more rows

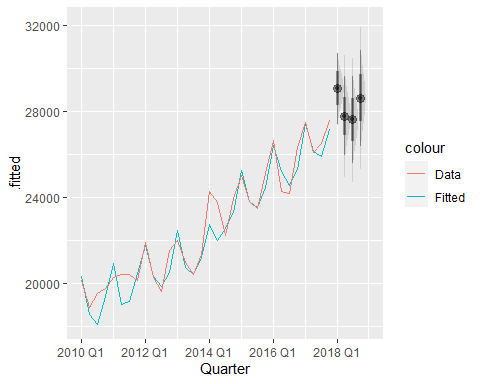
### Plot the Fitted and the Data

fitted\_ets |>  
 ggplot(aes(x = Quarter))+  
 geom\_line(aes(y = Trips, color = "Data"))+  
 geom\_line(aes(y = .fitted, color = "Fitted"))



ggplot(data = fcst\_ets, mapping = aes(x = Quarter, ydist = Trips))+  
 stat\_halfeye(alpha = .4)+  
 geom\_line(aes(y = .fitted, colour ="Fitted"), data = filter\_index(fitted\_ets, "2010 Q1" ~ .))+  
 geom\_line(aes(y = Trips, colour ="Data"),data = filter\_index(fitted\_ets, "2010 Q1" ~ .))

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.

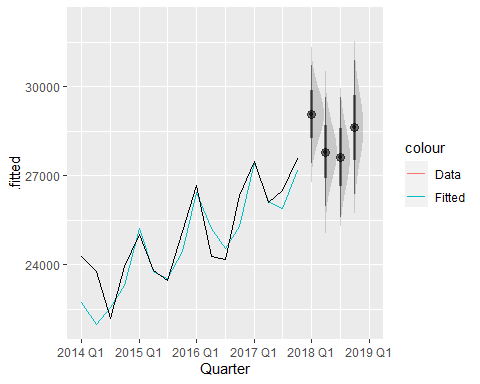


## Alternatively

ggplot(data = fcst\_ets, mapping = aes(x = Quarter, ydist = Trips))+  
 stat\_halfeye(alpha = .5)+  
 geom\_line(aes(y = .fitted, color = "Fitted"),data = filter\_index(fitted\_ets, "2014 Q1"~.))+  
 geom\_line(aes(y = Trips, color = "Data"), data = filter\_index(fitted\_ets, "2014 Q1"~.))+  
 autolayer(filter\_index(aus\_tourism, "2014 Q1"~.))

Warning: The output of `fortify(<fable>)` has changed to better suit usage with the ggdist package.  
If you're using it to extract intervals, consider using `hilo()` to compute intervals, and `unpack\_hilo()` to obtain values.

Plot variable not specified, automatically selected `.vars = Trips`



### Basic accuracy- split to Test and Train

### Testing dataset

forecast\_horizon <- 4  
test <- aus\_tourism |>  
 filter\_index(as.character(max(aus\_tourism$Quarter)-forecast\_horizon +1) ~ .)  
test

# A tsibble: 4 x 2 [1Q]  
 Quarter Trips  
 <qtr> <dbl>  
1 2017 Q1 27496.  
2 2017 Q2 26114.  
3 2017 Q3 26506.  
4 2017 Q4 27594.

### Training data set

train <- aus\_tourism |> filter\_index(. ~ as.character(max(aus\_tourism$Quarter)-forecast\_horizon))  
train

# A tsibble: 76 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 66 more rows

Our test goes from the beginning of the times series to 2016 quarter 4. On the other hand, the test data takes the last four observation of the series, that is, 2017 Q1 to 2017 Q4. We have managed to split our data into test and train data set.

From the code above, we are creating a training and test sets for forecasting. From the code, we are filtering the data to create separate sets for training and testing, likely for a time series forecasting model.

Here’s a breakdown of your code:

1. **forecast\_horizon <- 4**: We are defining a variable **forecast\_horizon** with a value of 4. This variable is representing the number of periods into the future you want to forecast.
2. **test <- aus\_tourism |> filter\_index(as.character(max(aus\_tourism$Quarter) - forecast\_horizon + 1) ~ .)**: We are creating the test set. Again we are filtering the **aus\_tourism** data based on the index (Quarter) to select observations for the test set. The filter condition used in this code helps us select data points starting from the quarter which is **forecast\_horizon** quarters before the maximum quarter.
3. **train <- aus\_tourism |> filter\_index(. ~ as.character(max(aus\_tourism$Quarter) - forecast\_horizon))**: In this code, we are creating the training set. Similar to the test set, where we are filtering the **aus\_tourism** data, but this time we are selecting data points up to the quarter which is **forecast\_horizon** quarters before the maximum quarter.

It’s important to note that the success of a time series forecasting model often depends on how well you structure your training and test sets. Make sure that the temporal order of your data is maintained, with the training set covering the period before the test set.

Also, ensure that the data types and formats are consistent, especially when dealing with time-related variables such as quarters. If **Quarter** is a date or time-related variable, consider converting it to an appropriate date format for better handling of time series data.

If you have specific questions or if there’s anything else you would like assistance with, feel free to ask!

### Estimate the Model

fit <- train |>  
 model(average = MEAN(Trips),  
 naive = NAIVE(Trips),  
 snaiev = SNAIVE(Trips),  
 regression = TSLM(Trips ~ trend() + season()),  
 automatic\_ets = ETS(Trips),  
 automatic\_arima = ARIMA(Trips)  
 )  
fit

# A mable: 1 x 6  
 average naive snaiev regression automatic\_ets automatic\_arima  
 <model> <model> <model> <model> <model> <model>  
1 <MEAN> <NAIVE> <SNAIVE> <TSLM> <ETS(M,N,M)> <ARIMA(0,1,1)(0,1,1)[4]>

### Recap: View the arima model

fit|>  
 select(automatic\_arima)|>  
 report()

Series: Trips   
Model: ARIMA(0,1,1)(0,1,1)[4]   
  
Coefficients:  
 ma1 sma1  
 -0.5017 -0.8669  
s.e. 0.1041 0.1271  
  
sigma^2 estimated as 791728: log likelihood=-584.87  
AIC=1175.73 AICc=1176.09 BIC=1182.52

### Create the Forecast Using the Models Above

fcst <- fit |>   
 forecast(h = forecast\_horizon)  
fcst

# A fable: 24 x 4 [1Q]  
# Key: .model [6]  
 .model Quarter Trips .mean  
 <chr> <qtr> <dist> <dbl>  
 1 average 2017 Q1 N(21270, 3343105) 21270.  
 2 average 2017 Q2 N(21270, 3343105) 21270.  
 3 average 2017 Q3 N(21270, 3343105) 21270.  
 4 average 2017 Q4 N(21270, 3343105) 21270.  
 5 naive 2017 Q1 N(26348, 2167931) 26348.  
 6 naive 2017 Q2 N(26348, 4335862) 26348.  
 7 naive 2017 Q3 N(26348, 6503793) 26348.  
 8 naive 2017 Q4 N(26348, 8671725) 26348.  
 9 snaiev 2017 Q1 N(26661, 1404835) 26661.  
10 snaiev 2017 Q2 N(24285, 1404835) 24285.  
# ℹ 14 more rows

### Estimate Models’ Accuracy Metrics

In this case, we use the accuracy() function to estimate the models’ accuracy as shown below.

fcst\_accuracy <- fcst |>  
 accuracy(aus\_tourism,  
 measures = list(point\_accuracy\_measures,  
 interval\_accuracy\_measures,  
 distribution\_accuracy\_measures))  
fcst\_accuracy

# A tibble: 6 × 13  
 .model .type ME RMSE MAE MPE MAPE MASE RMSSE ACF1 winkler  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 automatic\_arima Test 784. 900. 784. 2.91 2.91 0.856 0.759 0.281 4068.  
2 automatic\_ets Test 1108. 1336. 1108. 4.11 4.11 1.21 1.13 0.300 4841.  
3 average Test 5658. 5693. 5658. 21.0 21.0 6.18 4.80 -0.249 90136.  
4 naive Test 580. 859. 697. 2.10 2.55 0.761 0.725 -0.249 8869.  
5 regression Test 4419. 4454. 4419. 16.4 16.4 4.83 3.76 0.307 55871.  
6 snaiev Test 1556. 1655. 1556. 5.82 5.82 1.70 1.40 -0.178 4646.  
# ℹ 2 more variables: percentile <dbl>, CRPS <dbl>

#View(fcst\_accuracy)

The table give various accuracy metrics but we can just extract a few that we interested in, because we do not need everything here.

### Extract a Few Accuracy Metrics of Interest

acc <-fcst\_accuracy |>   
 select(.model, MAE, RMSE, MAPE, winkler, CRPS)  
acc

# A tibble: 6 × 6  
 .model MAE RMSE MAPE winkler CRPS  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 automatic\_arima 784. 900. 2.91 4068. 509.  
2 automatic\_ets 1108. 1336. 4.11 4841. 759.  
3 average 5658. 5693. 21.0 90136. 4628.  
4 naive 697. 859. 2.55 8869. 669.  
5 regression 4419. 4454. 16.4 55871. 3509.  
6 snaiev 1556. 1655. 5.82 4646. 1038.

#View(acc)

If we look at CRPS, the lowest value if for automatic arima. This is mostly used if you are interested in the entire distribution. However, if we are interested in the prediction interval, arima is doing good as well. For the point forecast, we have RMSE from which naive is doing better. That is, the model that has the lowest error measure is naive.

## Short Notes

1. **MAE (Mean Absolute Error):**
   * **Definition:** The average absolute difference between the predicted and actual values.
   * **Interpretation:** Lower MAE indicates better model accuracy. It is easy to understand and gives equal weight to all errors.
2. **RMSE (Root Mean Squared Error):**
   * **Definition:** The square root of the average of squared differences between predicted and actual values.
   * **Interpretation:** Similar to MAE, but it penalizes larger errors more heavily. It provides a measure of the spread of errors.
3. **MAPE (Mean Absolute Percentage Error):**
   * **Definition:** The average percentage difference between predicted and actual values, expressed as a percentage of the actual values.
   * **Interpretation:** MAPE is useful when the scale of the variable being predicted varies. It represents the relative size of the errors.
4. **Winkler Score:**
   * **Definition:** A scoring metric that penalizes predictions that deviate from actual values.
   * **Interpretation:** Lower Winkler score is desirable. It is particularly used in forecasting tasks where deviations from actual values are penalized.
5. **CRPS (Continuous Ranked Probability Score):**
   * **Definition:** A probabilistic accuracy metric often used for probabilistic forecasts.
   * **Interpretation:** It measures the difference between the predicted and actual cumulative distribution functions. Lower CRPS indicates better probabilistic forecast accuracy.

Now, let’s apply these interpretations to your models:

1. **automatic\_arima:**
   * Performs well in terms of MAE, RMSE, and MAPE.
   * Achieves a relatively low Winkler score and CRPS, indicating good performance in penalizing deviations and probabilistic forecasting.
2. **automatic\_ets:**
   * Similar to automatic\_arima but with higher MAE, RMSE, and MAPE.
   * Scores higher on Winkler and CRPS, indicating higher penalties for deviations and potentially less accurate probabilistic forecasting.
3. **average:**
   * Has high values across all metrics, indicating poor performance in terms of MAE, RMSE, MAPE, Winkler, and CRPS.
   * This model might be a benchmark or a simple baseline, but it is not performing well compared to the other models.
4. **naive:**
   * Performs well, especially in terms of MAE and Winkler.
   * Has a low MAPE, indicating good relative performance. The CRPS value is also relatively low.
5. **regression:**
   * Has high values for all metrics, suggesting poorer performance compared to other models.
   * It is likely not the best-performing model based on these metrics.
6. **snaiev:**
   * Performs reasonably well in terms of MAE, RMSE, and MAPE.
   * Has a lower Winkler score and CRPS compared to the average model, indicating better penalization of deviations and probabilistic forecasting.

Remember that the choice of the appropriate metric depends on the specific characteristics of your forecasting problem and the importance you assign to different types of errors.

### Time Series Cross Validation (Advanced Accuracy Testing)

It is too risky to make the prediction and conclusion based on only one forecast horizon, that is very risky. Such a method cannot be recommended. We have to make sure that the model we recommend to people has been tested to various forecast horizon. That is the concept in time series called cross validation. We started with a simple train and test model, now we mover to a more advanced model. In this process, we start by defining the forecast horizon and the percentage\_test.

forecast\_horizon <- 4  
percentage\_test <- 0.3

test <- aus\_tourism |> filter\_index(as.character(max(aus\_tourism$Quarter)))  
test

# A tsibble: 1 x 2 [1Q]  
 Quarter Trips  
 <qtr> <dbl>  
1 2017 Q4 27594.

### Test

test <- aus\_tourism |> filter\_index(as.character(max(aus\_tourism$Quarter) -  
 round(percentage\_test\*length(unique(aus\_tourism$Quarter)))+1) ~ .)  
test

# A tsibble: 24 x 2 [1Q]  
 Quarter Trips  
 <qtr> <dbl>  
 1 2012 Q1 21924.  
 2 2012 Q2 20274.  
 3 2012 Q3 19646.  
 4 2012 Q4 21558.  
 5 2013 Q1 21984.  
 6 2013 Q2 20972.  
 7 2013 Q3 20397.  
 8 2013 Q4 21338.  
 9 2014 Q1 24279.  
10 2014 Q2 23790.  
# ℹ 14 more rows

### Train

train <- aus\_tourism |> filter\_index(. ~ as.character(max(aus\_tourism$Quarter) -  
 round(percentage\_test\*length(unique(aus\_tourism$Quarter)))))  
train

# A tsibble: 56 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 46 more rows

### Alternatively

# Set the proportion for the training set  
training\_proportion <- 0.7  
  
# Calculate the splitting index  
split\_index <- floor(nrow(aus\_tourism) \* training\_proportion)  
  
# Split the tsibble into training and test sets  
train\_data <- aus\_tourism[1:split\_index, ]  
test\_data <- aus\_tourism[(split\_index + 1):nrow(aus\_tourism), ]

### View the train data

train\_data

# A tsibble: 56 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 46 more rows

### View the test data

View(test\_data)

## stretch\_tsibble() function

### Proceed with previous method of splitting the data

tscv\_aus\_tourism <- aus\_tourism |>  
 filter\_index(. ~ as.character(max(aus\_tourism$Quarter)-forecast\_horizon)) |>  
 stretch\_tsibble(.init = length(unique(train$Quarter)), .step = 1)  
tscv\_aus\_tourism

# A tsibble: 1,386 x 3 [1Q]  
# Key: .id [21]  
 Quarter Trips .id  
 <qtr> <dbl> <int>  
 1 1998 Q1 23182. 1  
 2 1998 Q2 20323. 1  
 3 1998 Q3 19827. 1  
 4 1998 Q4 20830. 1  
 5 1999 Q1 22087. 1  
 6 1999 Q2 21458. 1  
 7 1999 Q3 19914. 1  
 8 1999 Q4 20028. 1  
 9 2000 Q1 22339. 1  
10 2000 Q2 19941. 1  
# ℹ 1,376 more rows

### View

View(tscv\_aus\_tourism)

From the data above, we have 21 different ids. In other words, we will evaluate the forecast in 21 different forecast horizons in 21 different four steps ahead forecast. So will have different sample with different situations.

fit <- tscv\_aus\_tourism |>  
 model(average = MEAN(Trips),  
 naive = NAIVE(Trips),  
 snaiev = SNAIVE(Trips),  
 regression = TSLM(Trips ~ trend() + season()),  
 automatic\_ets = ETS(Trips),  
 automatc\_arima = ARIMA(Trips)  
 ) |>  
 mutate(combination = (automatc\_arima+automatic\_ets+snaiev)/3)

### View the output for ARIMA

fit|>  
 select(automatc\_arima)|>  
 report()

Warning in report.mdl\_df(select(fit, automatc\_arima)): Model reporting is only  
supported for individual models, so a glance will be shown. To see the report  
for a specific model, use `select()` and `filter()` to identify a single model.

# A tibble: 21 × 9  
 .id .model sigma2 log\_lik AIC AICc BIC ar\_roots ma\_roots   
 <int> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <list> <list>   
 1 1 automatc\_arima 832760. -429. 864. 865. 870. <cpl [1]> <cpl [4]>  
 2 2 automatc\_arima 732977. -433. 878. 880. 890. <cpl [3]> <cpl [5]>  
 3 3 automatc\_arima 716231. -441. 894. 895. 905. <cpl [3]> <cpl [5]>  
 4 4 automatc\_arima 695360. -448. 909. 911. 921. <cpl [3]> <cpl [5]>  
 5 5 automatc\_arima 975021. -465. 937. 938. 945. <cpl [6]> <cpl [0]>  
 6 6 automatc\_arima 768140. -469. 944. 945. 951. <cpl [1]> <cpl [4]>  
 7 7 automatc\_arima 944538. -480. 969. 969. 977. <cpl [6]> <cpl [0]>  
 8 8 automatc\_arima 927599. -488. 984. 985. 992. <cpl [6]> <cpl [0]>  
 9 9 automatc\_arima 912076. -496. 1000. 1000. 1008. <cpl [6]> <cpl [0]>  
10 10 automatc\_arima 971926. -506. 1020. 1021. 1029. <cpl [6]> <cpl [0]>  
# ℹ 11 more rows

### Extract a Particular ARIMA model for a specific ID

fit|>  
 select(automatc\_arima)|>  
 filter(.id == 20)|>  
 report()

Series: Trips   
Model: ARIMA(0,1,1)(2,1,0)[4]   
  
Coefficients:  
 ma1 sar1 sar2  
 -0.5805 -0.6013 -0.2499  
s.e. 0.1104 0.1195 0.1203  
  
sigma^2 estimated as 953842: log likelihood=-580.74  
AIC=1169.48 AICc=1170.09 BIC=1178.47

This model is different from what we saw before. Initially, we had only one time series, but in this case we have 21 different ids and there we have 21 different time series. In other words, each id is treated as time series. Each row is one time series and each column is one method. In this case we will therefore have 21 different models for each method.

## Forecast and Plot the Forecast

fcst <- fit |> forecast(h = forecast\_horizon)  
fcst

# A fable: 588 x 5 [1Q]  
# Key: .id, .model [147]  
 .id .model Quarter Trips .mean  
 <int> <chr> <qtr> <dist> <dbl>  
 1 1 average 2012 Q1 N(20629, 1529860) 20629.  
 2 1 average 2012 Q2 N(20629, 1529860) 20629.  
 3 1 average 2012 Q3 N(20629, 1529860) 20629.  
 4 1 average 2012 Q4 N(20629, 1529860) 20629.  
 5 1 naive 2012 Q1 N(20124, 2139220) 20124.  
 6 1 naive 2012 Q2 N(20124, 4278440) 20124.  
 7 1 naive 2012 Q3 N(20124, 6417661) 20124.  
 8 1 naive 2012 Q4 N(20124, 8556881) 20124.  
 9 1 snaiev 2012 Q1 N(20266, 1208426) 20266.  
10 1 snaiev 2012 Q2 N(20399, 1208426) 20399.  
# ℹ 578 more rows

### View the Forecast of Model for ID 1

fcst|>  
 filter(.id == 1)

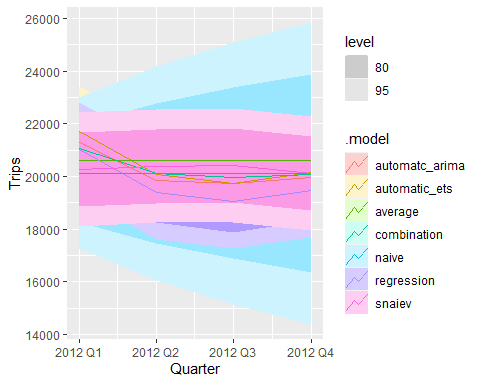
# A fable: 28 x 5 [1Q]  
# Key: .id, .model [7]  
 .id .model Quarter Trips .mean  
 <int> <chr> <qtr> <dist> <dbl>  
 1 1 average 2012 Q1 N(20629, 1529860) 20629.  
 2 1 average 2012 Q2 N(20629, 1529860) 20629.  
 3 1 average 2012 Q3 N(20629, 1529860) 20629.  
 4 1 average 2012 Q4 N(20629, 1529860) 20629.  
 5 1 naive 2012 Q1 N(20124, 2139220) 20124.  
 6 1 naive 2012 Q2 N(20124, 4278440) 20124.  
 7 1 naive 2012 Q3 N(20124, 6417661) 20124.  
 8 1 naive 2012 Q4 N(20124, 8556881) 20124.  
 9 1 snaiev 2012 Q1 N(20266, 1208426) 20266.  
10 1 snaiev 2012 Q2 N(20399, 1208426) 20399.  
# ℹ 18 more rows

### View the aus\_tourism dataset

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

fcst|>  
 filter(.id == 1)|>  
 autoplot()



### Performance Metrics and Models’ Accuracy

fcst\_accuracy <- fcst |>  
 accuracy(aus\_tourism,  
 measures = list(point\_accuracy\_measures,  
 interval\_accuracy\_measures,  
 distribution\_accuracy\_measures))

### View the Performance and Accuracy of each model

fcst\_accuracy

# A tibble: 7 × 13  
 .model .type ME RMSE MAE MPE MAPE MASE RMSSE ACF1 winkler  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 automatc\_arima Test 827. 1188. 921. 3.48 3.89 1.01 1.00 0.382 6432.  
2 automatic\_ets Test 1020. 1308. 1055. 4.21 4.36 1.15 1.10 0.278 6719.  
3 average Test 2825. 3469. 2938. 11.2 11.8 3.21 2.93 0.667 37098.  
4 combination Test 980. 1263. 1019. 4.06 4.25 1.11 1.07 0.293 6854.  
5 naive Test 744. 1674. 1378. 2.90 5.77 1.50 1.41 0.466 8860.  
6 regression Test 2915. 3175. 2915. 12.0 12.0 3.18 2.68 0.698 37589.  
7 snaiev Test 1093. 1420. 1176. 4.50 4.91 1.28 1.20 0.190 6877.  
# ℹ 2 more variables: percentile <dbl>, CRPS <dbl>

1. **Automatic ARIMA:** Moderate errors across metrics. Low MAE and RMSE compared to some models, indicating relatively good performance in predicting close values. The ACF1 value suggests some autocorrelation in residuals. Moderate overall performance.
2. **Automatic ETS:** Similar performance to ARIMA but slightly higher errors. ACF1 is also slightly higher, indicating residual autocorrelation.
3. **Mean Model:** Highest errors among all models. Considerably higher MAE, RMSE, and MAPE values, indicating poor performance in capturing the pattern.
4. **Combination:** Moderate errors, lower than Mean Model. Shows better performance compared to the mean model but slightly higher errors than ARIMA and ETS.
5. **Naive:** Lower errors in MAE and RMSE compared to some models, suggesting better performance in predicting close values. However, relatively high MAPE indicates higher percentage errors.
6. **Regression:** Shows high errors across metrics, similar to the Mean Model. It seems to perform poorly compared to other models.
7. **Seasonal Naive:** Moderate errors overall, similar to ARIMA and ETS. ACF1 is relatively low, indicating less autocorrelation in residuals compared to some models.

Among these models, Automatic ARIMA and Automatic ETS seem to perform relatively well overall, having moderate errors across metrics and lower autocorrelation in residuals. Naive and Combination models also exhibit moderate performance. The Mean Model, Regression, and Seasonal Naive models have higher errors and might not be suitable choices based on these metrics.

## Note!!!

### **Metrics Focused on the Entire Distribution:**

1. **CRPS (Continuous Ranked Probability Score):**
   * **Focus:** Measures the discrepancy between the predicted and observed cumulative distribution functions. It evaluates probabilistic forecasts by assessing the entire distribution of predicted probabilities.
   * **Application:** Ideal for assessing the accuracy of probabilistic forecasts, giving an overview of the model’s performance across the entire range of predicted probabilities.
2. **RMSSE (Root Mean Squared Scaled Error):**
   * **Focus:** Measures the relative forecast performance by comparing the RMSE of the forecast to the RMSE of a naïve forecast.
   * **Application:** Evaluates the forecasting performance relative to a baseline (naïve forecast) by considering the scale of the data.

### **Metrics Focused on Point Forecasts:**

1. **ME (Mean Error):**
   * **Focus:** Measures the average error, indicating the tendency of the forecasts to overestimate or underestimate the actual values.
   * **Application:** Provides information on the average bias of the forecasts. If consistently positive or negative, it indicates a systematic over- or underestimation.
2. **RMSE (Root Mean Squared Error):**
   * **Focus:** Measures the square root of the average of squared differences between predicted and actual values.
   * **Application:** It provides a measure of the overall accuracy of the model’s point forecasts, penalizing larger errors more heavily.
3. **MAE (Mean Absolute Error):**
   * **Focus:** Measures the average absolute difference between predicted and actual values.
   * **Application:** Similar to RMSE but without squaring errors, offering an alternative view of the model’s accuracy without magnifying larger errors.
4. **MAPE (Mean Absolute Percentage Error):**
   * **Focus:** Measures the average percentage difference between predicted and actual values.
   * **Application:** Provides insights into the relative size of errors, useful when comparing models predicting different scales.
5. **MPE (Mean Percentage Error):**
   * **Focus:** Measures the average of percentage errors.
   * **Application:** Offers information on the tendency of the model to overestimate or underestimate and the magnitude of these errors.

### **Weighted or Penalizing Metrics:**

1. **Winkler Score:**
   * **Focus:** A scoring metric that penalizes predictions that deviate from actual values.
   * **Application:** Provides a weighted assessment that penalizes significant deviations, particularly useful when specific focus on larger deviations is needed.

Each metric contributes a different perspective on the model’s performance, focusing either on overall distribution assessment or pinpointing accuracy at specific points. The choice of which metric to prioritize depends on the specific forecasting goals and the critical aspects of model performance you wish to emphasize.

### Extract Specific Accuracy Metrics

fcst\_accuracy |>   
 select(.model, MAE, RMSE, winkler, CRPS)

# A tibble: 7 × 5  
 .model MAE RMSE winkler CRPS  
 <chr> <dbl> <dbl> <dbl> <dbl>  
1 automatc\_arima 921. 1188. 6432. 666.  
2 automatic\_ets 1055. 1308. 6719. 748.  
3 average 2938. 3469. 37098. 2307.  
4 combination 1019. 1263. 6854. 726.  
5 naive 1378. 1674. 8860. 997.  
6 regression 2915. 3175. 37589. 2298.  
7 snaiev 1176. 1420. 6877. 831.

### Winkler score alone

fcst |>  
 accuracy(aus\_tourism, list(qs = winkler\_score), level = .9)

# A tibble: 7 × 3  
 .model .type qs  
 <chr> <chr> <dbl>  
1 automatc\_arima Test 1858.  
2 automatic\_ets Test 2128.  
3 average Test 5929.  
4 combination Test 2056.  
5 naive Test 2781.  
6 regression Test 5883.  
7 snaiev Test 2373.

### Quantile Score alone

fcst |>  
 accuracy(aus\_tourism, list(qs = quantile\_score), probs = .9)

# A tibble: 7 × 3  
 .model .type qs  
 <chr> <chr> <dbl>  
1 automatc\_arima Test 447.  
2 automatic\_ets Test 457.  
3 average Test 2632.  
4 combination Test 463.  
5 naive Test 530.  
6 regression Test 2673.  
7 snaiev Test 523.

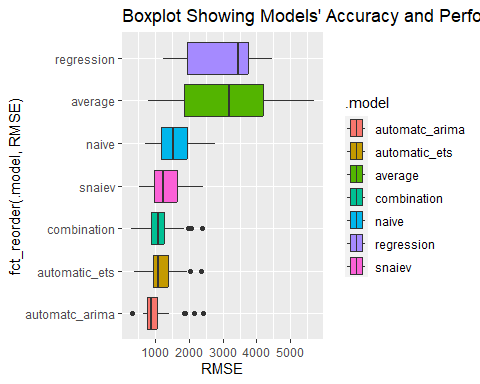
### Accuracy by model and .id

fcst\_accuracy <- fcst |>  
 accuracy(aus\_tourism, by = c(".model", ".id"))  
fcst\_accuracy

# A tibble: 147 × 11  
 .model .id .type ME RMSE MAE MPE MAPE MASE RMSSE ACF1  
 <chr> <int> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
 1 automatc\_arima 1 Test 635. 883. 688. 2.95 3.22 0.824 0.803 -0.365   
 2 automatc\_arima 2 Test 459. 789. 649. 2.09 3.06 0.764 0.709 -0.274   
 3 automatc\_arima 3 Test 574. 878. 801. 2.63 3.78 0.957 0.796 -0.496   
 4 automatc\_arima 4 Test 907. 988. 907. 4.27 4.27 1.08 0.900 -0.551   
 5 automatc\_arima 5 Test 285. 318. 285. 1.33 1.33 0.336 0.288 -0.208   
 6 automatc\_arima 6 Test 1163. 1406. 1163. 5.13 5.13 1.40 1.28 0.0501  
 7 automatc\_arima 7 Test 1349. 1864. 1349. 5.65 5.65 1.62 1.71 0.297   
 8 automatc\_arima 8 Test 1861. 2133. 1861. 7.95 7.95 2.24 1.97 0.0256  
 9 automatc\_arima 9 Test 2396. 2423. 2396. 10.2 10.2 2.93 2.25 -0.711   
10 automatc\_arima 10 Test 1823. 1871. 1823. 7.67 7.67 2.16 1.69 -0.666   
# ℹ 137 more rows

### Boxplot

fcst\_accuracy |> ggplot(aes( x = RMSE, y = fct\_reorder(.model, RMSE), fill = .model))+  
 geom\_boxplot()+  
 labs(title = "Boxplot Showing Models' Accuracy and Performance")



In the realm of forecasting models, the evaluation of performance metrics often serves as a guiding light to discern the efficacy of various methodologies. In this scenario, the Automatic ARIMA model emerges as the forerunner, showcasing a superior predictive prowess among its peers. Its adeptness in minimizing errors across key metrics, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE), positions it as a reliable choice for accurate point forecasts. The model’s demonstrated ability to capture patterns within the dataset without succumbing to significant biases underscores its robustness.

Following closely in the wake of Automatic ARIMA, the Automatic ETS model and the combined model exhibit commendable performance. While both models present slightly higher errors compared to the ARIMA, their overall accuracy and nuanced forecasting capabilities place them in the upper echelon of the lineup.

The Seasonal NAIVE and the NAIVE models secure their positions with respectable performances, albeit with slightly elevated errors when predicting values. Their simplicity and baseline-level performance make them suitable options for initial benchmarks but may fall short when aiming for finer predictive precision.

Contrarily, the Mean Model and Regression model display considerable shortcomings in forecasting accuracy, marked by notably higher errors across metrics. These models, albeit functional, lag behind in capturing the nuances of the dataset, signaling their limitations in providing accurate predictions.

In summary, the hierarchy of model performance, led by the Automatic ARIMA, suggests a tiered evaluation where models with more sophisticated methodologies and superior adaptability in capturing underlying patterns shine brighter, whereas simpler or less adaptive models lag behind in predictive accuracy.