Introduction to Data Science

Workshop: Time-series analysis with tsibble and fable

Armande, Benjamin, and Johannes
Hertie School

Our presentation in a nutshell

R Packages for Data Science : Introduction to Tsibble and Fable

Temporal data with tsibble and fable :

1. Introduction to the Tsibble Package for Time-Series Analysis
2. Introduction to the Fable Package for Time-Series forecasting
3. Tutorial

Tsibble Basics: Introduction to the Tsibble Package for Time-Series Analysis

What is the Tsibble Package?

R Packages for Data Science

- Let's take it from the tsibble website:

"The tsibble package provides a data infrastructure for tidy temporal data with wrangling tools."

It is part of the 'tidyverts' packages who has been mostly done by Earo Wang and Mitchell O'Hara-Wild.

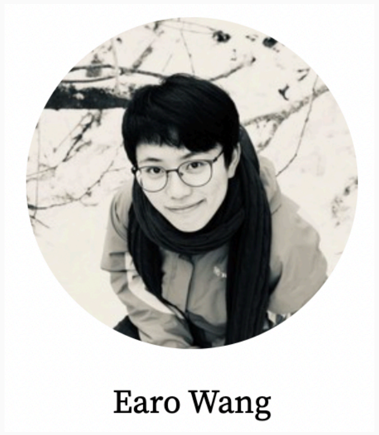


image from: earo

A Guide to Tsibble

Valuable Resources

- Tsibble : RDocumentation
- Tsibble : learn more about tsibble
- Tsibble : tidy temporal data frames and tools
- Tsibble : CRAN
- Tsibble : example
- Tsibble : Research Paper : a new tidy data structure to support exploration and modeling of temporal data
- Tsibble : Tsibble objects

Tsibble Package

Installing and Loading the Tsibble Package

- You can install the `tsibble` package from CRAN using the install.packages("tsibble") command.

```r
install.packages("tsibble")
```

- To load the package, use the library(tsibble) function.

```r
library(tsibble)
```

Purpose of the Tsibble Package

-Set of **tools** and **functions** for creating a structured framework for working with time-series data.

-Integrates information about **time index**, **key**, and other attributes to make it easier to work with time-based data.

-Wrangling tools to get data into a useful form for **visualization** and modeling.

-Part of the **tidyverse**, making it compatible with other tidyverse packages like ggplot2, dplyr, and tidyr.

But what are time series?

A reminder of time series data: Stats I and II

A time series is a series of **discrete-time data** (points or observations) y_t that are gathered or recorded at different moments in time.

1. "Time series allows for **delayed effects** or effects that persist over time."
2. In time series, there is a clear **temporal ordering**, and time series are indexed by **time**.
3. Time series require relatively **many measurements** (long lengths) conversely to longitudinal data (panel data) which often assume fewer measurements but a large number of individuals.
4. Examples in which time series are useful include studying the delayed effect of a gaffe by a political candidate on his popularity for a period.
5. Time series analysis helps understand the data, finding patterns and **trends**.

Some assumptions for time series :

Stationarity :

-Constant mean and constant variance -Mathematically, stationarity can be defined as: $\mu_t = \mu$ & $\sigma_t^2 = \sigma^2$

Independence :

-Each observation in the time series is **not influenced** by previous or future observations.

-Mathematically, $\mathrm{Cov}(y_t, y_{t-k}) = 0$ for $k \neq 0$

Normality :

-Some models assume that the errors or residuals ε_t follow a **normal distribution**.

Most important concepts for time series :

Autocorrelation Function :

Measures the correlation between a time series observation at time t and its **past values** at different lags k. It is denoted as $\rho(k)$ and is expressed as:

$$\rho(k) = \frac{\mathrm{Cov}(y_t, y_{t-k})}{\sigma^2}$$

Seasonality :

Represents repeating patterns in the data at **fixed time intervals**, such as daily, weekly, or yearly. These patterns can be modeled mathematically using periodic functions.

Trends :

Indicates the **long-term direction of change** in a time series. Can be linear or nonlinear, capturing overall increases or decreases in values over time.

Data Structure

A tsibble is an **enhanced version** of a data frame or tibble (i.e., a data structure part of the tidyverse, an alternative to a data frame) specifically designed for **time series data**.

- The tsibble package is used at the **tidy stage** to "verify if the raw temporal data is appropriate for downstream analytics" ie if the raw time-related data is well-structured enough for more advanced analysis that comes later in the data processing pipelines. So `tsibble` assesses if the data is in a good format for further analysis. Tsibble Research Paper:

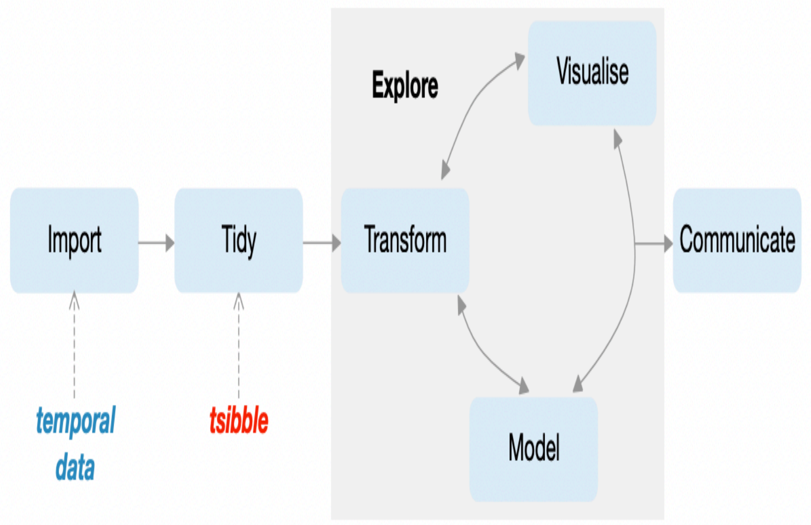
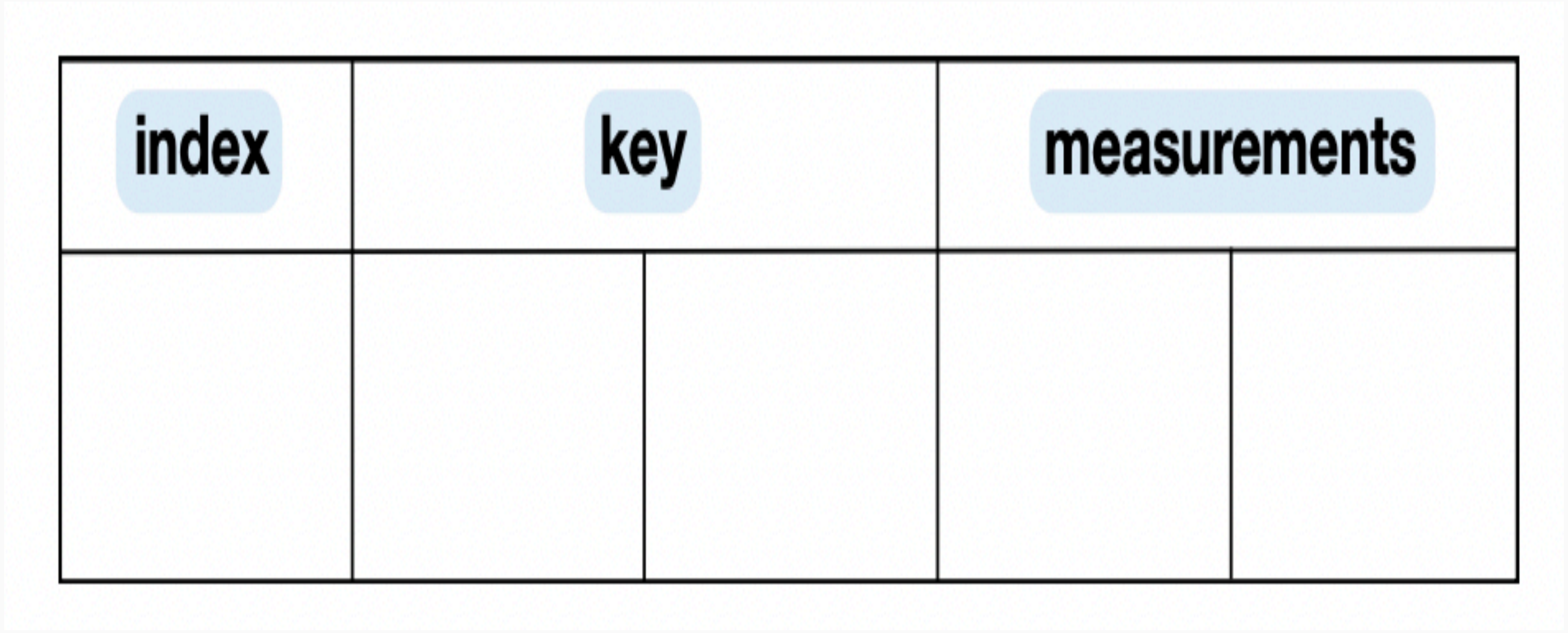


Figure 1: *Annotation of the data science workflow regarding temporal data, drawn from Wickham & Grolemund (2016). The new data structure, tsibble, makes the connection between temporal data input, and downstream analytics. It provides elements at the "tidy" step, which produce tidy temporal data for temporal visualization and modeling.*

Data Principles

- In tsibble : tidy data principles

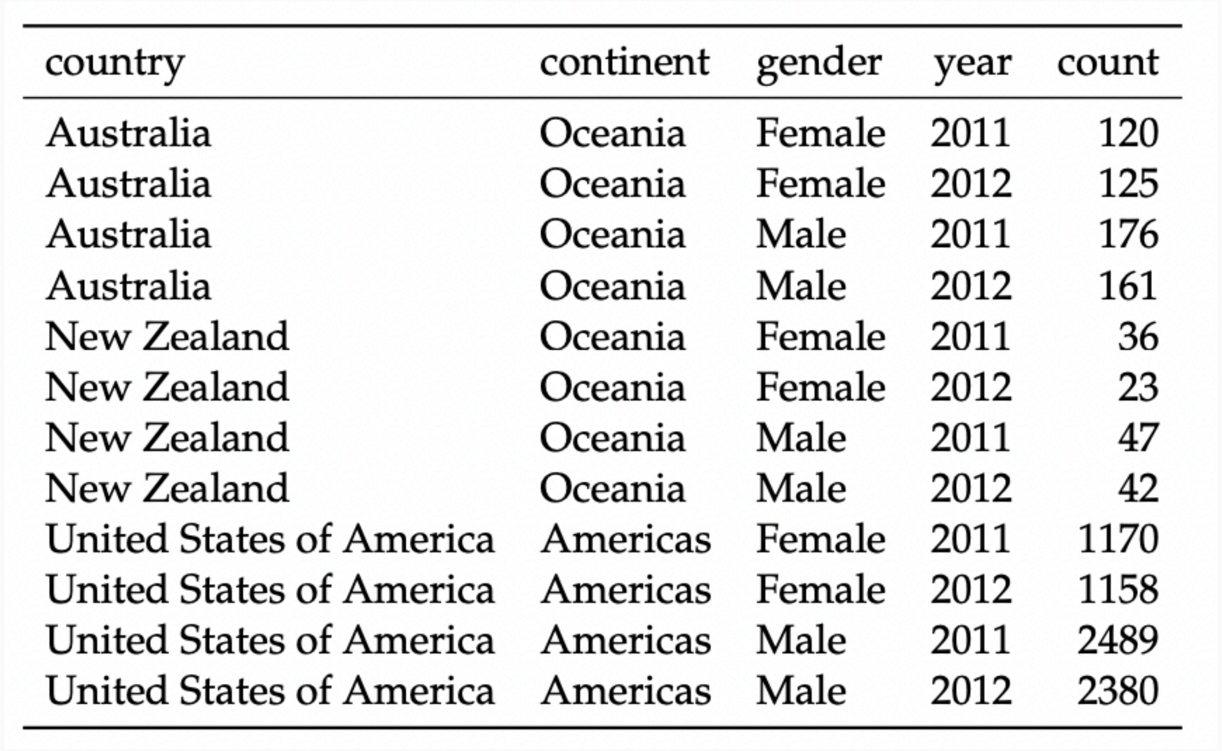
1. **Index** is a variable with inherent **ordering** from past to present.
2. **Key** is a set of variables that define observational units over time.
3. Each observation should be **uniquely identified** by index and key.
4. Each observational unit should be measured at a **common interval**, if regularly spaced.



index	key		measurements	

So we have two variables : Index and Key. Variables other than index and key are considered as measurements.

In the same Paper, the authors give an example of tsibble data.



country	continent	gender	year	count
Australia	Oceania	Female	2011	120
Australia	Oceania	Female	2012	125
Australia	Oceania	Male	2011	176
Australia	Oceania	Male	2012	161
New Zealand	Oceania	Female	2011	36
New Zealand	Oceania	Female	2012	23
New Zealand	Oceania	Male	2011	47
New Zealand	Oceania	Male	2012	42
United States of America	Americas	Female	2011	1170
United States of America	Americas	Female	2012	1158
United States of America	Americas	Male	2011	2489
United States of America	Americas	Male	2012	2380

More explanations of each principle

1. *Index* : Introduces **temporal structure** in tidy data frames. We should understand easily what is the index and it should be easy to extract this information.

2. *Key* : Way to **store multiple time series** in one dataset. The key identifies **unique** entities or groups within the data, especially when there are different types of entities. In the past example, the key would include variables like "gender" or "country" since the data records tuberculosis cases for each gender in different countries every year. The tsibble key helps identifying and distinguishing each entity over time.

3. *Interval* : We have two main categories of time data : regular and irregular time intervals. Examples of irregular time intervals are flight schedules. The tsibble will reports a **single interval** even if the data has a mix of different intervals.

Key Features of "tsibble"

Index and Key:

The tsibble structure allows us to specify a time index and a key that **uniquely identifies each time series** in the data. It's designed to simplify time-series data handling and includes the temporal aspect of the data.

Time-Based Operations:

The package provides a range of operations for **manipulating and transforming** time series data, including handling irregular time intervals, filling gaps, and handling missing values.

Visualization:

The package includes functions for visualizing time series data, making it easier to explore and understand temporal patterns.

Why Should We Use It?

- Makes it easier to **prepare time-series data** for analysis.
- A good way to manipulate and **visualize** time-series data.
- Helps keep your code **tidy** and **clean**.
- Particularly useful when dealing with **large volumes** of time-series data.

Main functions provided by tsibble :

`as_tsibble()` : converting data into a tsibble ie into a time series data frame with time intervals and key columns.

`fill_gaps()` : fill gaps in a time series, ensuring that there are no missing time points. Useful for creating a complete time series with regular intervals.

`index_by()` : helps to specify the time index of our tsibble. We can use it to define the time variable and the key variable that uniquely identifies each time series within our data.

`interval()` : used to specify the time interval of our time series data. Crucial to work with regularly spaced time series.

`aggregate_key()` : used to aggregate a tsibble based on specific keys. Good for summarizing or grouping time series data.

`index()` and `time_index()` : help extract or manipulate the time index in a tsibble.

Example : turning a dataframe into tsibble

Tsibble with Hypothetical Data

Let's create a tsibble with a hypothetical dataset:

library(tsibble) library(dplyr)

```r
# We first have to create a tsibble obj
# Create a dataframe sales_data with th
sales_data ← data.frame(
  Date = seq.Date(from = as.Date("2023-
  Product_ID = rep(1:3, each = 10),
  Sales = rnorm(30)
) %>%
  as_tsibble(index = Date, key = Produc
```

Visualization with Tsibble

Tsibble can be used to create time-series visualizations. Here's an example using ggplot2:

```r
# We can create a line plot.
library(ggplot2)
sales_data %>%
  ggplot(aes(x = Date, y = Sales, color
  geom_line() +
  labs(title = "Sales over Time", x = "
```

Tidy time series data using tsibbles :

Rob J Hyndman

`tsibble` is a new way of organizing time date easier than some older method such as ts. It allows us to switch data from the older 'ts' format to the `tsibble` format by using the function `as_tsibble`.

Here, in the example :

```
#To illustrate this idea : let's take an example :
library(tidyverse)
library(tsibble)
USAccDeaths %>% as_tsibble()
#We just created an Index column which species the time or date index. Our second va
#Moreover it is also possible to create tsibbles from csv files by using readr::read_
```

Different data types can be converted into a tsibble object such as Data frames, Time series objects as we just saw, tidy data or numeric vectors with a time index.

Tsibble object containing quaterly time

```r
#Let's see an example of a tsibble object containing quaterly overnight trips across
tourism
#There are three keys : Region, State, Purpose and one measurement : Trips.
#Say we want to get the total visitor nights spent on Holiday by State for each quar
tourism %>%
  filter(Purpose == "Holiday") %>%
  group_by(State) %>%
  summarise(Trips = sum(Trips))
#Important note : we don't have to explicitly group by the time index because it is
#Let's switch to annual data and thus re-index the tsibble :
tourism %>%
  mutate(Year = lubridata::year(Quarter)) %>%
  index_by(Year) %>% #here, index_by plays the same role as the group_by function
  group_by(Region, State, Purpose) %>%
  summarise(Trips = sum(Trips)) %>%
  ungroup()
```

Tsibble object with daily and sub-daily

```r
#The tsibble package also handles very well daily and sub-daily data. Let's take the
pedestrian
pedestrian %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat")
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) + #Here, Day and Time variables spl
    geom_line(aes(col = Weekend)) +
    facet_grid(Sensor ~ .)

  )
```

Learn more about tsibble

- There is a whole ecosystem the `tidyverts`, around the `tsibble` package, aiming to tidy time series analysis : Tsibble Tidyverts

  -The `tsibbledata` package which compiles various examples of tsibble data

  -The `feasts` package which allows to visualize data and extract time series features

  -The `fable` package which offers a popular forecasting techniques for tsibble including ARIMA and ETS while its foundation, the fabletools package, simplifies the modeling process when working with tsibble data.

  We will now learn more about the fable package, but before this do you have questions on the tsibble package

FAQ

Q: : What other packages work well with tsibble for time-series analysis?

A: Tsibble is part of the `tidyverse`, so it integrates seamlessly with other tidyverse packages like ggplot2, dplyr, and tidyr. Additionally, tsibble is compatible with time-series analysis packages like forecast and fable.

Q: Is tsibble suitable for handling **large volumes** of time-series data?

A: Yes, tsibble is a great choice for working with large volumes of time-series data. Its

Q: Are there **alternatives to dplyr** for data wrangling when working with tsibble?

A: Yes, when dealing with large datasets, we can consider the `data.table` package as an alternative to dplyr. Another option is to use dtplyr, which is a data.table backend for dplyr. It automatically translates dplyr code to data.table code for faster performance.

Presentation on Fable Package for Time series Forecasting

Time series Forecasting

`Time Series forecasting` is a crucial component of data analysis, enabling us to make predictions about future values based on historical data points.

In R, the `fable` package provides a powerful framework for time series forecasting, making it easier than ever to create accurate and reliable forecasts.

Much like `tsibble` implements tidy time series data, the fable package applies tidyverse principles to time series modeling, making the forecasting workflow seamlessly integrate with other tidyverse packages.

image from: Mitchell O'Hara-Wild

Introduction to "fable"

`fable` is a comprehensive time series forecasting package in R, designed to make forecasting tasks more accessible, flexible, and efficient.

It extends the functionalities of the "forecast" package and the `tsibble` package, making it a versatile tool for time series analysis and forecasting.

The fable package provides some commonly used univariate and multivariate time series forecasting models which can be used with tidy temporal data in the tsibble format.

image from: Mitchell O'Hara-Wild

These models are used within a consistent and tidy modeling framework, allowing several models to be estimated, compared, combined, forecasted and otherwise worked across many time series.

Introduction to "fable"

The Role of Fable in R

- Fable is an R package provides a wide range of forecasting models and methods, from simple ones like exponential smoothing to more complex models such as ARIMA and state space models. This diversity of approaches allows users to choose the most appropriate model for their specific data set and forecasting needs.

- Additionally, Fable makes it easy to visualize forecasts, evaluate model performance, and create production-ready reports. With the `fable` package, you can unlock the power of time series forecasting in R to gain insights and make data-driven decisions.

Installing and Loading

You can install the `fable` package from CRAN using the install.packages("fable") command.

To load the package, use the library(fable) function.

Functions & Model Diagnostic Tools

`fable` includes an array of diagnostic tools to assess the quality and reliability of your forecasting models.

These tools may include visualizations, such as residual plots and forecasting accuracy metrics, like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), Cross-Validation, Autocorrelation etc. These diagnostics help you fine-tune your models.

Some functions in the Fable package

`tsibble`: The tsibble (time seriesibble) is a core data structure in fable. It extends the data frame concept to handle time series data efficiently.

`fable()`: This function is used to create a fable, which is a tidy representation of a model. A fable contains point forecasts, prediction intervals, and various components of a forecast.

`model()`: The model() function is used to specify time series models for forecasting. It's used to fit models to the data and extract forecasts.

Functions & Model Diagnostic Tools

`as_fable()`: This function converts time series objects into fable objects. It's particularly useful when you want to use fable functions with data in other formats.

`forecast()`: This function generates point forecasts and prediction intervals for time series data based on a fitted model.

`combine()`: Used to combine multiple fable objects into a single fable, which can be useful when working with multiple time series or models.

`autoplot()`: An essential function for visualization. It's used to create time series plots and forecast visualizations from fable objects.

`accuracy()`: This function is used to calculate accuracy metrics to evaluate the performance of forecasting models.

`filter_fable()` and summarise_fable(): These functions allow you to filter and summarize the components of a fable, helping you extract specific information or insights.

Functions & Model Diagnostic Tools

`refit()`: This function takes the existing model and incorporates the new data, updating the model parameters as needed.

Visualization:

The package empowers users with rich visualization capabilities, enabling a better understanding of time series data. Some commonly used packages with the fable are listed below:

- Ggplot2
- Ggfortify: This package allows us to blend the use of some basic functions with the package eg. autoplot() function with forecast objects.
- feasts: stands for Feature Extraction and Statistics for Time Series, which is used for extracting features from time series data and generating plots like seasonal decomposition plots and autocorrelation plots.

Functions & Model Diagnostic Tools

The `fable` package in R provides a variety of time series models for forecasting. Some of the key models available in the `fable` package include:

ETS Models

The ETS (Error-Trend-Seasonal) model is a popular and widely used time series forecasting model for analyzing and forecasting univariate time series data.

In the context of the R package fable, the ETS model is part of the forecasting framework provided by the fable package, which offers a unified interface for various time series forecasting methods.

Functioning

The ETS model decomposes a time series into three components: Error (E), Trend (T), and Seasonal (S). The model assumes that these components follow a certain structure, which

Functions & Model Diagnostic Tools

6 possible combinations of these components in ETS models

- ETS(A,A,A): Additive errors, additive trend, and additive seasonality.
- ETS(M,A,A): Multiplicative errors, additive trend, and additive seasonality.
- ETS(A,A,M): Additive errors, additive trend, and multiplicative seasonality.
- ETS(M,A,M): Multiplicative errors, additive trend, and multiplicative seasonality.
- ETS(A,M,A): Additive errors, multiplicative trend, and additive seasonality.
- ETS(M,M,A): Multiplicative errors, multiplicative trend, and additive seasonality.

`The ETS model` assumes that the time series can be decomposed into the three components mentioned above. It assumes that the `errors are normally distributed and have constant variance` and the `trend and seasonal components`are assumed to follow a `specific structure`, which can be additive or multiplicative.

Functions & Model Diagnostic Tools

ARIMA Model

ARIMA (AutoRegressive Integrated Moving Average) is a widely used time series forecasting model, and it can be used with the fable package in R for time series analysis.

Functioning

- ARIMA models are a combination of Autoregressive (AR) and Moving Average (MA) models, with an optional differencing (I) component for stationarity.
- The AR component models the relationship between the current value and its past values.
- The MA component models the relationship between the current value and past forecast errors.
- The I component deals with differencing to make the time series stationary (i.e.,

Forecasting with fable

```
pacman::p_load(fable,tidyverse,tsibble)
```

It is a good practice to observe your data before you start to work with it (modelling).

```
trips = tourism |>
  summarise(Trips = sum(Trips))
#trips
```

The terms used with the fable model somewhat include model specific functions called ‘specials, which describes how the time series dynamics are captured by a model.

`Fable` allows model specification that supports formula based interface. eg lm()

`ETS()` function is used to define `exponential smoothing models` which provides ‘specials’ for controlling the error(), trend() and season().

Note finding an appropriate model specification can be tricky as it requires some prior

Forecasting with fable

Model specification:

This is the process of defining and describing the structure, components, and assumptions of a statistical or mathematical model.

`ETS()` and other models automatically choose the best specification if several options are available. `fable` can determine if the seasonality is additive (season("A")) or multiplicative (season("M")), using:

ETS model can be implement with: `ETS(Trips)`

Model Estimation

This is training one or more unique models with a dataset. To do this we can use the `model()` function. eg:

Forecasting with fable Forecasting with

- fitting a model with the fit `model()` function

```
trips = tourism ▷
  summarise(Trips = sum(Trips))

fit = trips ▷
  model(auto_ets = ETS(Trips))
fit
```

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

The result informs us that model ETS(A,A,A) has been automatically selected. We can use the `report()` function to provide

```
report(fit)
```

```
## 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]
##   21689.64 -58.46946 -125.8548 -816.3416 -
##
##   sigma^2:  699901.4
##
##      AIC      AICc       BIC
```

Forecasting with fable

Fable also supports verbs from `broom package` which enable us to use various functions: `tidy()` your coefficients, `glance()` your model summary statistics, and `augment()` your data with predictions.

- **forecasting**

Use the `forecast()` function to produce a forecast for the estimated models.

```
fore_cast = fit ▷
  forecast(h = "1 year")
fore_cast
```

```
## # A fable: 4 x 4 [1Q]
## # Key:     .model [1]
##   .model   Quarter              Trips  .mean
##   <chr>      <qtr>             <dist>  <dbl>
## 1 auto_ets 2018 Q1   N(29068, 7e+05) 29068.
```

- **Plot time series**

```
#fore_cast ▷
  #autoplot(trips)
```

Forecasting with fable

You can also try forecasting with different models; eg ARIMA, TSLM (linear model) etc.

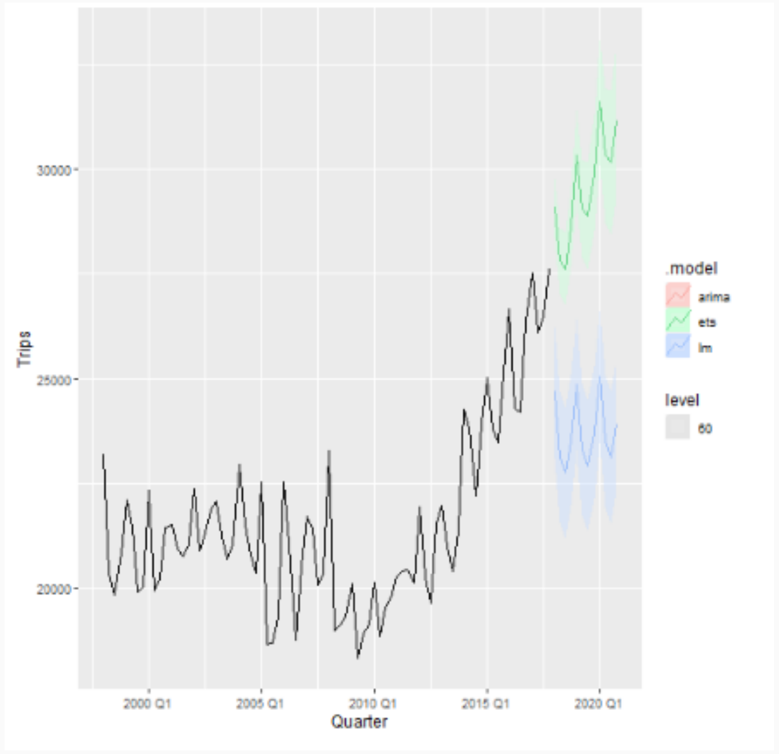
```
fit = trips ▷
  model(
    arima = ARIMA(Trips),
    ets = ETS(Trips),
    lm = TSLM(Trips ~ trend() + season())
  )
fit
```

```
## # A mable: 1 x 3
##                      arima           ets       lm
##                    <model>       <model> <model>
## 1 <ARIMA(0,1,1)(0,1,1)[4]> <ETS(A,A,A)>  <TSLM>
```

The results now show the 3 models. We can also make a plot with respect to each model.

Forecasting with fable

```
#fit ▷
  #forecast(h = "3 year") ▷
 # autoplot((trips), level = 60, alpha = 0.5)
```



Final comment A lot more could be done with respect to whatever your problem set maybe.

- Forecasting (forecast())
- Missing value interpolation (interpolate())
- Reporting model output (report())
- Simulation of future paths (generate())
- Streaming new data (stream())
- Re-estimation (refit())
- Decomposition of model components