

# Time Series Analysis & Forecasting Using R

An aerial photograph of a city skyline during sunset. The sky is a mix of orange, yellow, and light blue. In the foreground, there's a green field and some trees. The middle ground is filled with various buildings, including several tall skyscrapers. The background shows more distant city structures and hills under a hazy sky.

The forecasters' toolbox-Part 1

Bahman Rostami-Tabar

# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5

# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5

# Learning outcome

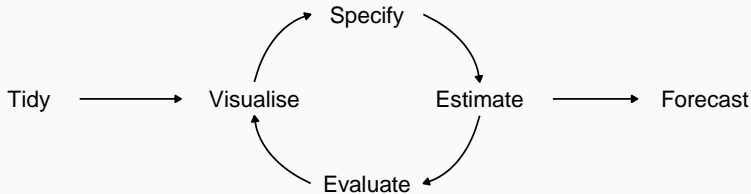
You should be able to:

- 1 Explain simple forecasting methods (benchmarks)
- 2 Specify and estimate models using R functions in `fable`
- 3 Recognise and extract fitted values and residuals
- 4 Produce point and prediction interval forecasts

# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5

# A tidy forecasting workflow



# Outline

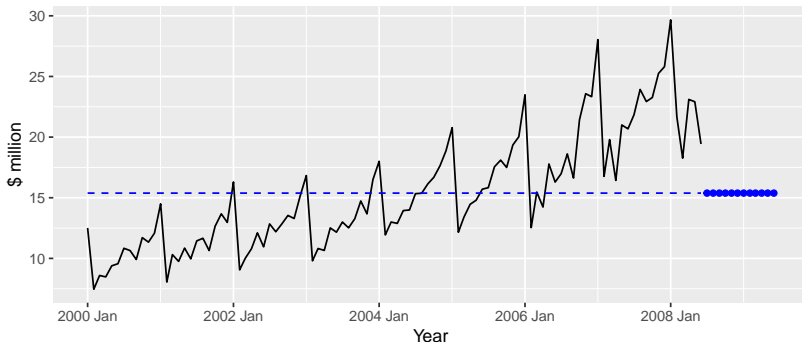
- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)**
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5

# Some simple forecasting methods

## MEAN(y): Average method

- Forecast of all future values is equal to mean of historical data  $\{y_1, \dots, y_T\}$ .
- Forecasts:  $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \dots + y_T)/T$

Antidiabetic drug sales using simple average



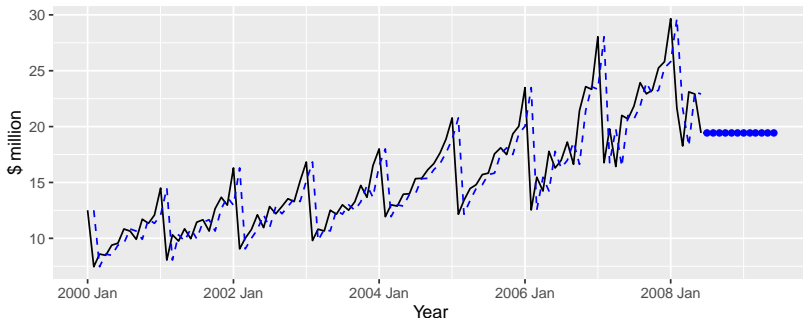


# Some simple forecasting methods

## NAIVE(y): Naïve method

- Forecasts equal to last observed value.
- Forecasts:  $\hat{y}_{T+h|T} = y_T$ .
- Consequence of efficient market hypothesis.

Antidiabetic drug sales using Naive method

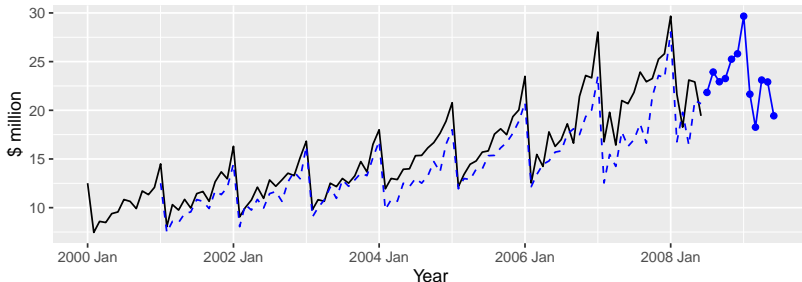


# Some simple forecasting methods

## SNAIVE( $y \sim \text{lag}(m)$ ): Seasonal naïve method

- Forecasts equal to last value from same season.
- Forecasts:  $\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$ , where  $m$  = seasonal period and  $k$  is the integer part of  $(h - 1)/m$ .

Antidiabetic drug sales usign Snaive method



# Model specification

- Model specification in fable supports a formula based interface
- A model formula in R is expressed using `response ~ terms`
  - ▶ the formula's left side describes the response
  - ▶ the right describes terms used to model the response.
- Attention: `MODEL_NAME` is in capital letters, e.g. `SNAIVE`

```
MODEL_NAME(response_variable ~ term1+term2+...)  
SNAIVE(Beer ~ lag("year"))
```

# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5

# Model estimation: template

The `model()` function trains models to data. - It returns a model table or a `mable` object.

```
# Fit the models
my_mable <- my_data %>%
  model(
    choose_name1 = MODEL_1(response_variable ~ term1+...),
    choose_name2 = MODEL_2(response_variable ~ term1+...),
    choose_name3 = MODEL_3(response_variable ~ term1+...),
    choose_name4 = MODEL_4(response_variable ~ term1+...)
  )
```

# Model estimation

```
# Fit the models
beer_fit <- aus_production %>%
  model(
    mean = MEAN(Beer),
    naive = NAIVE(Beer),
    snaive = SNAIVE(Beer, lag="year")
  )

#beer_fit <- beer_fit %>% stream(new_data),
#we can update the fitted models once we have new data
```

# mable: a model object

```
beer_fit
```

```
## # A mable: 1 x 3
##      mean    naive    snaive
##    <model> <model> <model>
## 1  <MEAN> <NAIVE> <SNAIVE>
```

- A mable is a model table, each cell corresponds to a fitted model.
- A mable contains
  - ▶ a row for each time series
  - ▶ a column for each model specification

# Extract coefficients from mab1e

```
beer_fit %>% select(snaive) %>% report()  
beer_fit %>% tidy()  
beer_fit %>% glance()
```

- The `report()` function gives a formatted model-specific display.
- The `tidy()` function is used to extract the coefficients from the models.
- The `glance()` shows a summary from the models.
- We can extract information about some specific model using the `filter()` and `select()` functions.



# Producing forecasts

- The `forecast()` function is used to produce forecasts from estimated models.
- `h` can be specified with a number (the number of future observations) or natural language (the length of time to predict).

```
beer_fc <- beer_fit %>%  
  forecast(h = "3 years")  
#h = "3 years" is equivalent to setting h = 12.
```

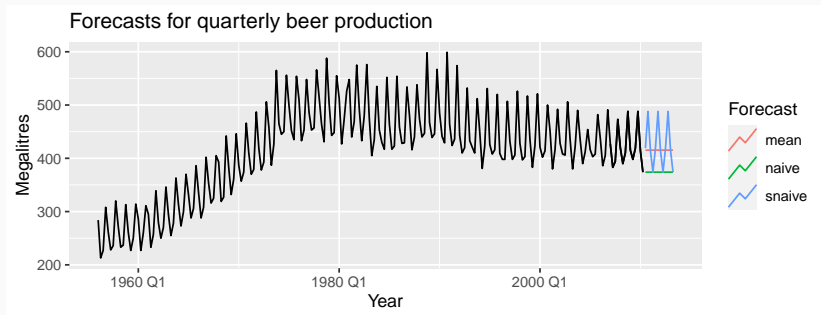
# Producing forecasts

```
## # A fable: 36 x 4 [1Q]
## # Key:      .model [3]
##   .model Quarter      Beer .mean
##   <chr>      <qtr>      <dist> <dbl>
## 1 mean      2010 Q3 N(415, 7409) 415.
## 2 mean      2010 Q4 N(415, 7409) 415.
## 3 mean      2011 Q1 N(415, 7409) 415.
## 4 mean      2011 Q2 N(415, 7409) 415.
## # ... with 32 more rows
```

A fable is a forecast table with point forecasts and distributions.

# Visualising forecasts

```
beer <- aus_production |> select(Beer)
# Plot forecasts against actual values
beer_fc %>%
  autoplot(beer, level = NULL) +
  autolayer(filter_index(aus_production, "2007 Q1" ~ .), color = "black") +
  ggtitle("Forecasts for quarterly beer production") +
  xlab("Year") + ylab("Megalitres") +
  guides(colour=guide_legend(title="Forecast"))
```



# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals**
- 6 Prediction intervals
- 7 lab session 5

# Fitted values

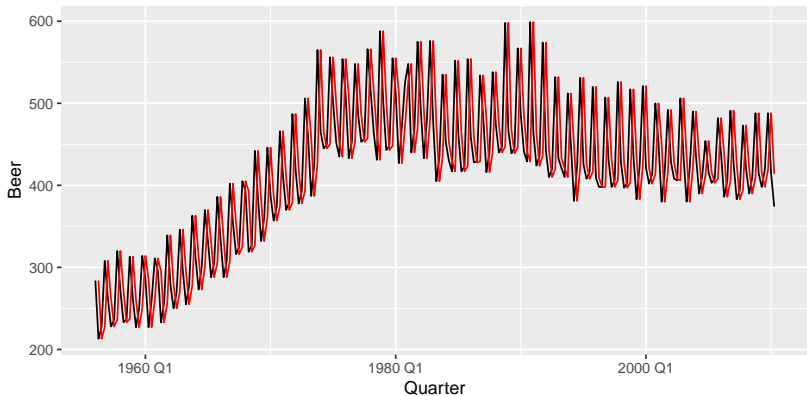
- $\hat{y}_{T|T-1}$  is the forecast of  $y_T$  based on observations  $y_1, \dots, y_{T-1}$ .
- We call these “fitted values”.
- Sometimes drop the subscript:  $\hat{y}_T \equiv \hat{y}_{T|T-1}$ .
- Often not true forecasts since parameters are estimated on all data.

## For example:

- $\hat{y}_T = \bar{y}$  for average method.
- $\hat{y}_T = y_{T-1} + (y_T - y_1)/(T - 1)$  for drift method.
- $\hat{y}_T = y_{T-1}$  for naive method.

# Fitted values

```
beer_fit %>% select(naive) %>% augment() %>%  
  ggplot(aes(x=Quarter, y=Beer))+  
  geom_line()+  
  geom_line(aes(y=.fitted), colour="red")
```



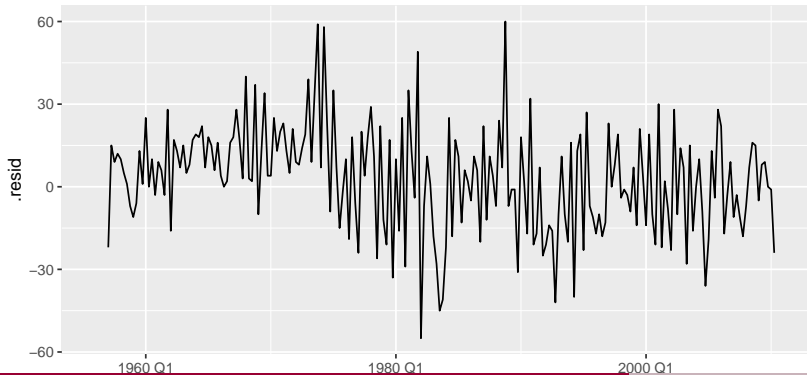
# Residuals

- The “residuals” in a time series model are what is left over after fitting a model.
- Residuals are useful in checking whether a model has adequately captured the information in the data.

**Residuals in forecasting:** difference between observed value and its fitted value:  $e_t = y_t - \hat{y}_{t|t-1}$ .

# Residuals

```
#beer_fit %>% fitted  
#`augment()` function gets residuals and fitted values  
beer_fit %>% select(snaive) %>% augment() %>%  
  ggplot(aes(x=Quarter, y=.resid))+  
  geom_line()
```





# Extract fitted values and residuals

```
beer_fit %>% augment()
```

```
## # A tsibble: 654 x 6 [1Q]
## # Key:           .model [3]
##   .model Quarter Beer .fitted .resid .innov
##   <chr>    <qtr> <dbl>   <dbl>  <dbl>  <dbl>
## 1 mean    1956 Q1   284    415. -131. -131.
## 2 mean    1956 Q2   213    415. -202. -202.
## 3 mean    1956 Q3   227    415. -188. -188.
## 4 mean    1956 Q4   308    415. -107. -107.
## 5 mean    1957 Q1   262    415. -153. -153.
## 6 mean    1957 Q2   228    415. -187. -187.
## 7 mean    1957 Q3   236    415. -179. -179.
## 8 mean    1957 Q4   320    415.  -95.4 -95.4
## 9 mean    1958 Q1   272    415. -143. -143.
## 10 mean   1958 Q2   233    415. -182. -182.
```

# Outline

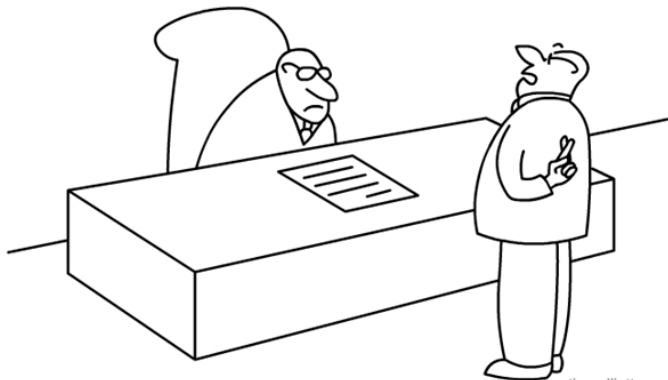
- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals**
- 7 lab session 5

# What is wrong with point forecasts?

- Resource allocation in A&E is assymmetric
  - ▶ The cost of over-allocating resources(over estimation) can vastly differ from the cost of under-allocating(under estimation)
- The disadvantage of point forecast:
  - ▶ it ignores additional information in future demand;
  - ▶ it does not explain uncertainties around future demand
  - ▶ it can not deal with assymmetric.

# Importance of providing interval forecast

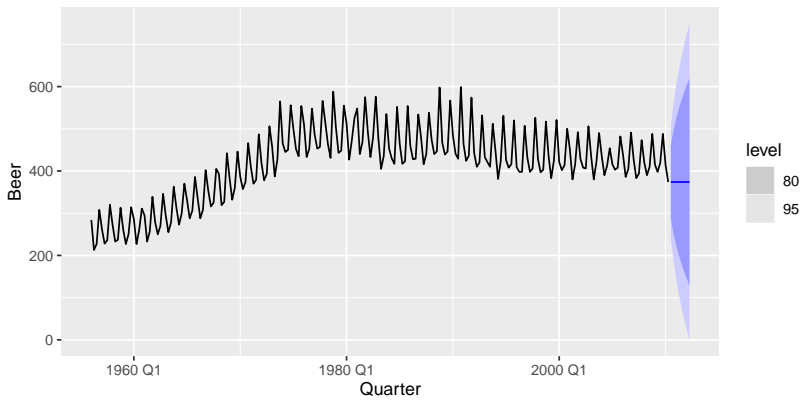
Point forecasts are often useless without a measure of uncertainty



*"Yes sir, you can absolutely trust those numbers"*

# Prediction intervals

- A prediction interval gives a region within which we expect  $y_{T+h}$  to lie with a specified probability
- It consists of an upper and a lower limit between which the future value is expected to lie



# Prediction intervals

- Assuming forecast errors are normally distributed, then a c% PI is:

$$\hat{y}_{T+h|T} \pm c\hat{\sigma}_h$$

where the multiplier  $c$  depends on the coverage probability and  $\hat{\sigma}_h$  is the st dev of the  $h$ -step distribution.

# Prediction intervals

- Forecast intervals can be extracted using the `hilo()` function
- Use `level` argument to control coverage.

```
fit <- aus_production %>% model(NAIVE(Beer))  
forecast(fit) %>% hilo(level = c(80, 95))
```

```
## # A tsibble: 8 x 6 [1Q]  
## # Key:           .model [1]  
##   .model Quarter      Beer .mean      '80%'  
##   <chr>      <qtr>      <dist> <dbl>      <hilo>  
## 1 NAIVE(~ 2010 Q3  N(374, 4580)   374 [287.2735, 460.7265] 80  
## 2 NAIVE(~ 2010 Q4  N(374, 9159)   374 [251.3502, 496.6498] 80  
## 3 NAIVE(~ 2011 Q1  N(374, 13739)  374 [223.7853, 524.2147] 80  
## 4 NAIVE(~ 2011 Q2  N(374, 18210)  374 [200.5470, 547.4530] 80
```

# Outline

- 1 Learning outcome
- 2 A tidy forecasting workflow
- 3 Define the model (specify)
- 4 Train the model (estimate)
- 5 Fitted values and Residuals
- 6 Prediction intervals
- 7 lab session 5



## lab session 5

- Produce forecasts from the bechmark methods for daily A&E series for 42 days
- Plot the results using `autoplot()`.
  - ▶ Use `filter_index()` to show the plot from 2016
- Use `augment()` to extract fitted values for snaive method
- Extract residuals for mean method