

# **ETC3550**

## **Applied forecasting for business and economics**

Ch5. The forecasters' toolbox

[OTexts.org/fpp3/](https://OTexts.org/fpp3/)

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Outline

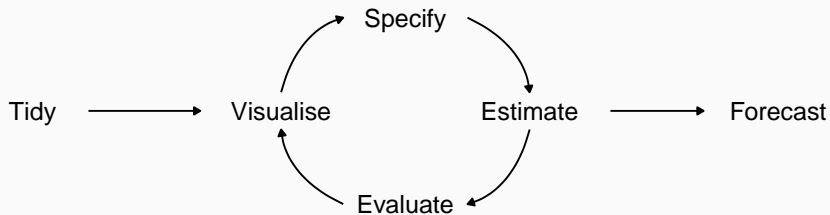
- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# A tidy forecasting workflow

The process of producing forecasts can be split up into a few fundamental steps.

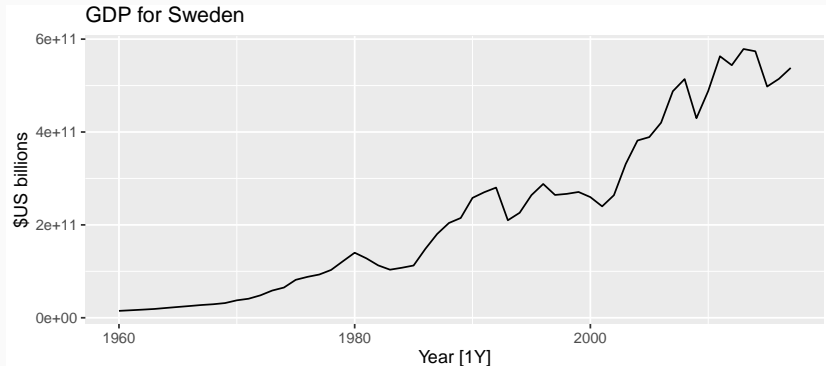
- 1 Preparing data
- 2 Data visualisation
- 3 Specifying a model
- 4 Model estimation
- 5 Accuracy & performance evaluation
- 6 Producing forecasts

# A tidy forecasting workflow



# Data preparation and visualisation

```
global_economy %>%  
  filter(Country=="Sweden") %>%  
  autoplot(GDP) +  
    ggtitle("GDP for Sweden") + ylab("$US billions")
```



# Model estimation

The `model()` function trains models to data.

```
fit <- global_economy %>%  
  model(trend_model = TSLM(GDP ~ trend()))  
fit
```

```
## # A mable: 263 x 2  
## # Key:      Country [263]  
##   Country      trend_model  
##   <fct>        <model>  
## 1 Afghanistan <TSLM>  
## 2 Albania     <TSLM>  
## 3 Algeria     <TSLM>  
## 4 American Samoa <TSLM>  
## 5 Andorra     <TSLM>
```

# Producing forecasts

```
fit %>% forecast(h = "3 years")
```

```
## # A tibble: 789 x 5 [1Y]
```

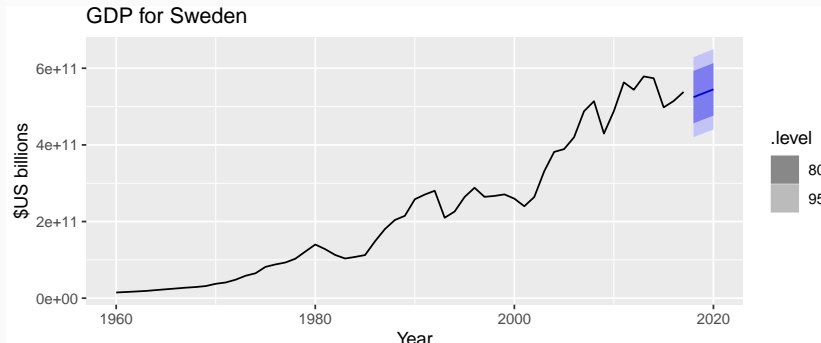
```
## # Key:   Country, .model [263]
```

| ## | Country       | .model     | Year  | GDP     | .distribution    |
|----|---------------|------------|-------|---------|------------------|
| ## | <fct>         | <chr>      | <dbl> | <dbl>   | <dist>           |
| ## | 1 Afghanistan | trend_mod~ | 2018  | 1.62e10 | N(1.6e+10, 1.3e~ |
| ## | 2 Afghanistan | trend_mod~ | 2019  | 1.65e10 | N(1.7e+10, 1.3e~ |
| ## | 3 Afghanistan | trend_mod~ | 2020  | 1.68e10 | N(1.7e+10, 1.3e~ |
| ## | 4 Albania     | trend_mod~ | 2018  | 1.37e10 | N(1.4e+10, 3.9e~ |
| ## | 5 Albania     | trend_mod~ | 2019  | 1.42e10 | N(1.4e+10, 3.9e~ |
| ## | 6 Albania     | trend_mod~ | 2020  | 1.46e10 | N(1.5e+10, 3.9e~ |
| ## | 7 Algeria     | trend_mod~ | 2018  | 1.58e11 | N(1.6e+11, 9.4e~ |
| ## | 8 Algeria     | trend_mod~ | 2019  | 1.61e11 | N(1.6e+11, 9.4e~ |
| ## | 9 Algeria     | trend_mod~ | 2020  | 1.64e11 | N(1.6e+11, 9.4e~ |



# Visualising forecasts

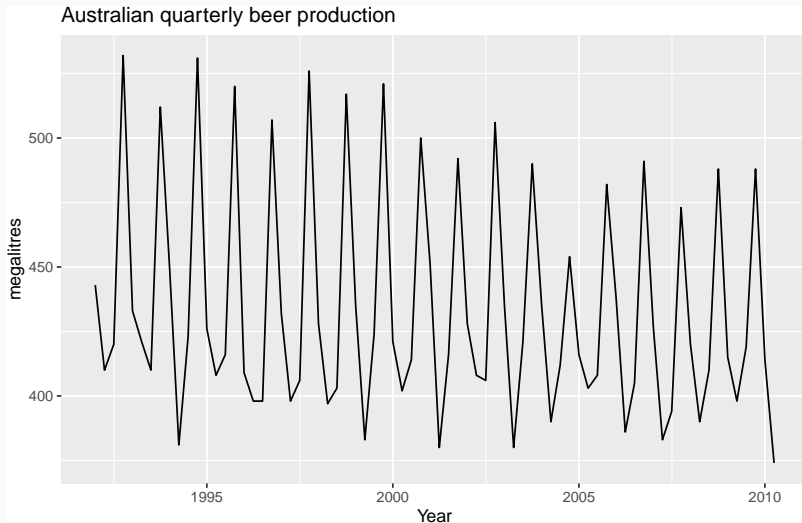
```
fit %>% forecast(h = "3 years") %>%  
  filter(Country=="Sweden") %>%  
  autoplot(global_economy) +  
    ggtitle("GDP for Sweden") + ylab("$US billions")
```



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Some simple forecasting methods



How would you forecast these series?

# Some simple forecasting methods



How would you forecast these series?

# Some simple forecasting methods

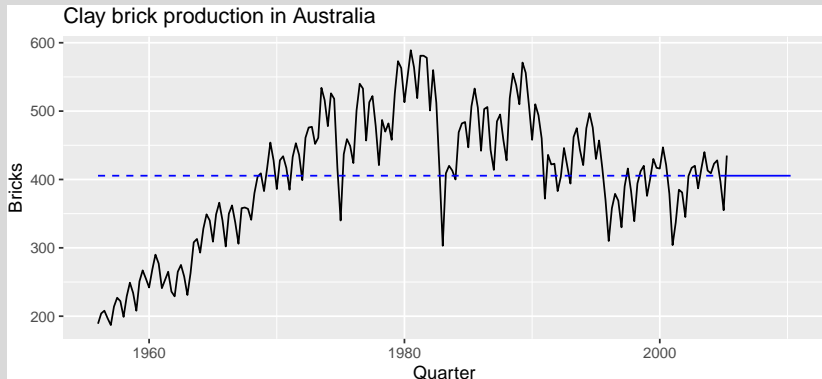


How would you forecast these series?

# Some simple forecasting methods

## MEAN( $y$ ): Average method

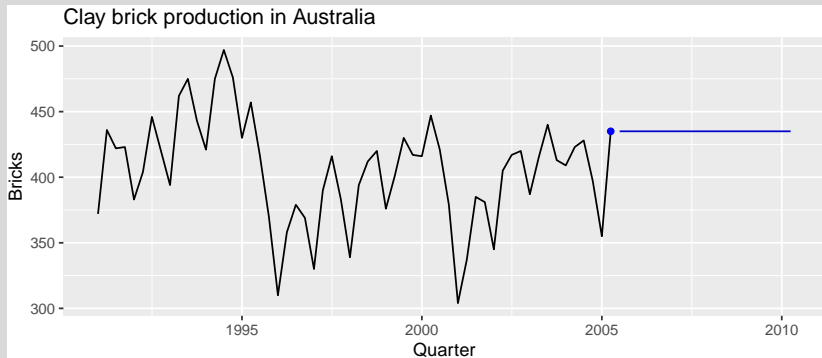
- Forecast of all future values is equal to mean of historical data  $\{y_1, \dots, y_T\}$ .
- Forecasts:  $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \dots + y_T)/T$



# Some simple forecasting methods

## NAIVE(y): Naïve method

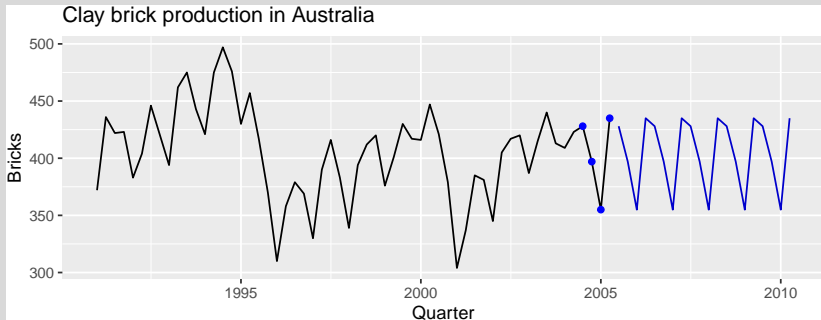
- Forecasts equal to last observed value.
- Forecasts:  $\hat{y}_{T+h|T} = y_T$ .
- Consequence of efficient market hypothesis.



# 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  $(h - 1)/m$ .





# Some simple forecasting methods

## `RW(y ~ drift())`: Drift method

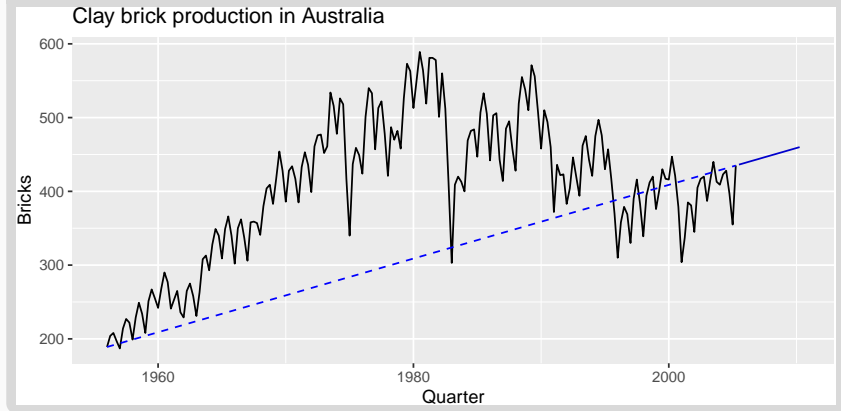
- Forecasts equal to last value plus average change.
- Forecasts:

$$\begin{aligned}\hat{y}_{T+h|T} &= y_T + \frac{h}{T-1} \sum_{t=2}^T (y_t - y_{t-1}) \\ &= y_T + \frac{h}{T-1} (y_T - y_1).\end{aligned}$$

- Equivalent to extrapolating a line drawn between first and last observations.

# Some simple forecasting methods

## Drift method



# Model fitting

The `model()` function trains models to data.

```
brick_fit <- aus_production %>%  
  filter(!is.na(Bricks)) %>%  
  model(  
    `Seasonal_naïve` = SNAIVE(Bricks),  
    `Naïve` = NAIVE(Bricks),  
    Drift = RW(Bricks ~ drift()),  
    Mean = MEAN(Bricks)  
  )
```

```
## # A mable: 1 x 4  
##   Seasonal_naïve Naïve   Drift         Mean  
##   <model>         <model> <model>         <model>  
## 1 <SNAIVE>        <NAIVE> <RW w/ drift> <MEAN>
```

A mable is a model table, each cell corresponds to a fitted model.

# Producing forecasts

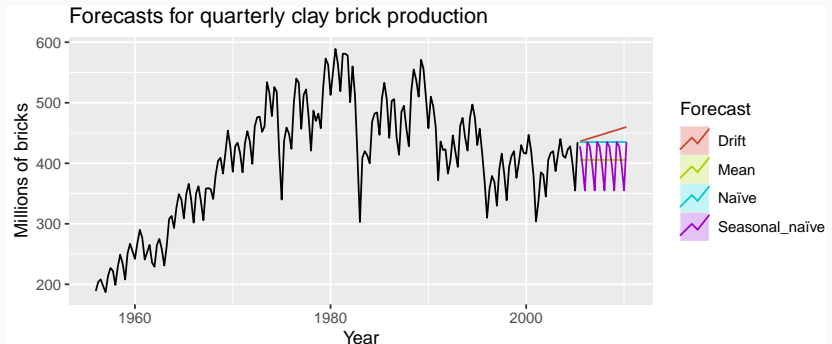
```
brick_fc <- brick_fit %>%  
  forecast(h = "5 years")
```

```
## # A fable: 80 x 4 [1Q]  
## # Key:      .model [4]  
##   .model      Quarter Bricks .distribution  
##   <chr>         <qtr>   <dbl> <dist>  
## 1 Seasonal_naïve 2005 Q3     428 N(428, 2336)  
## 2 Seasonal_naïve 2005 Q4     397 N(397, 2336)  
## 3 Seasonal_naïve 2006 Q1     355 N(355, 2336)  
## 4 Seasonal_naïve 2006 Q2     435 N(435, 2336)  
## # ... with 76 more rows
```

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

# Visualising forecasts

```
brick_fc %>%  
  autoplot(aus_production, level = NULL) +  
  ggtitle("Forecasts for quarterly clay brick production") +  
  xlab("Year") + ylab("Millions of bricks") +  
  guides(colour = guide_legend(title = "Forecast"))
```



# Facebook closing stock price

```
# Extract training data
fb_stock <- gafa_stock %>%
  group_by(Symbol) %>%
  mutate(trading_day = row_number()) %>%
  update_tsibble(index=trading_day, regular=TRUE) %>%
  ungroup() %>%
  filter(Symbol == "FB")

# Specify, estimate and forecast
fb_stock %>%
  model(
    Mean = MEAN(Close),
    `Naïve` = NAIVE(Close),
    Drift = RW(Close ~ drift())
  ) %>%
  forecast(h=42) %>%
  autoplot(fb_stock, level = NULL) +
  ggtitle("Facebook closing stock price") +
  xlab("Day") + ylab("") +
  guides(colour=guide_legend(title="Forecast"))
```

# Facebook closing stock price



## Your turn

- Produce forecasts using an appropriate benchmark method for household wealth (`hh_budget`). Plot the results using `autoplot()`.
- Produce forecasts using an appropriate benchmark method for Australian takeaway food turnover (`aus_retail`). Plot the results using `autoplot()`.



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Fitted values

- $\hat{y}_{t|t-1}$  is the forecast of  $y_t$  based on observations  $y_1, \dots, y_t$ .
- 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.

# Forecasting residuals

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

# Forecasting residuals

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

## Assumptions

- 1  $\{e_t\}$  uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
- 2  $\{e_t\}$  have mean zero. If they don't, then forecasts are biased.

# Forecasting residuals

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

## Assumptions

- 1  $\{e_t\}$  uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
- 2  $\{e_t\}$  have mean zero. If they don't, then forecasts are biased.

## Useful properties (for distributions & prediction intervals)

- 3  $\{e_t\}$  have constant variance.
- 4  $\{e_t\}$  are normally distributed.

# Facebook closing stock price

```
fb_stock <- gafa_stock %>%  
  filter(Symbol == "FB") %>%  
  mutate(trading_day = row_number()) %>%  
  update_tsibble(index = trading_day, regular = TRUE)  
fb_stock %>% autoplot(Close)
```



# Facebook closing stock price

```
fit <- fb_stock %>% model(NAIVE(Close))  
augment(fit)
```

```
## # A tibble: 1,258 x 6 [1]  
## # Key:      Symbol, .model [1]  
##   Symbol .model      trading_day Close .fitted .resid  
##   <chr>   <chr>          <int> <dbl>   <dbl>   <dbl>  
## 1 FB     NAIVE(Close)         1  54.7    NA     NA  
## 2 FB     NAIVE(Close)         2  54.6    54.7 -0.150  
## 3 FB     NAIVE(Close)         3  57.2    54.6  2.64  
## 4 FB     NAIVE(Close)         4  57.9    57.2  0.720  
## 5 FB     NAIVE(Close)         5  58.2    57.9  0.310  
## 6 FB     NAIVE(Close)         6  57.2    58.2 -1.01  
## 7 FB     NAIVE(Close)         7  57.9    57.2  0.720  
## 8 FB     NAIVE(Close)         8  55.9    57.9 -2.03  
## 9 FB     NAIVE(Close)         9  57.7    55.9  1.83  
## 10 FB    NAIVE(Close)        10  57.6    57.7 -0.140  
## # ... with 1,248 more rows
```

# Facebook closing stock price

```
fit <- fb_stock %>% model(NAIVE(Close))  
augment(fit)
```

```
## # A tibble: 1,258 x 6 [1]  
## # Key:      Symbol, .model [1]  
##   Symbol .model      trading_day Close .fitted .resid  
##   <chr>   <chr>          <int> <dbl>   <dbl>   <dbl>  
## 1 FB     NAIVE(Close)         1  54.7    NA     NA  
## 2 FB     NAIVE(Close)         2  54.6    54.7  -0.150  
## 3 FB     NAIVE(Close)         3  57.2    54.6   2.64  
## 4 FB     NAIVE(Close)         4  57.9    57.2   0.720  
## 5 FB     NAIVE(Close)         5  58.2    57.9   0.310  
## 6 FB     NAIVE(Close)         6  57.2    58.2  -1.01  
## 7 57.9    57.2   0.720  
## 8 55.9    57.9  -2.03  
## 9 57.7    55.9   1.83  
## 10 57.6    57.7  -0.140
```

## Naïve forecasts:

$$\hat{y}_{t|t-1} = y_{t-1}$$

$$e_t = y_t - \hat{y}_{t|t-1} = y_t - y_{t-1}$$



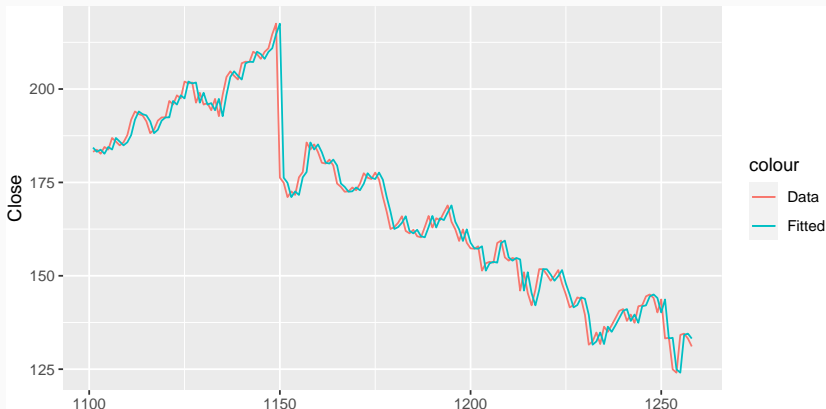
# Facebook closing stock price

```
augment(fit) %>%  
  ggplot(aes(x = trading_day)) +  
  geom_line(aes(y = Close, colour = "Data")) +  
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



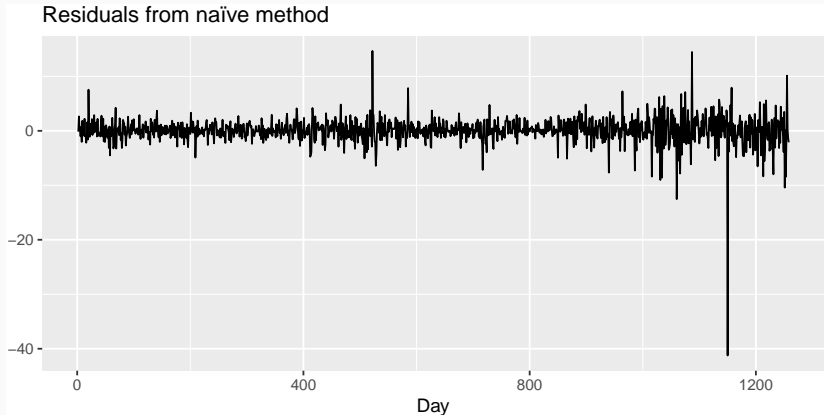
# Facebook closing stock price

```
augment(fit) %>%  
  filter(trading_day > 1100) %>%  
  ggplot(aes(x = trading_day)) +  
  geom_line(aes(y = Close, colour = "Data")) +  
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



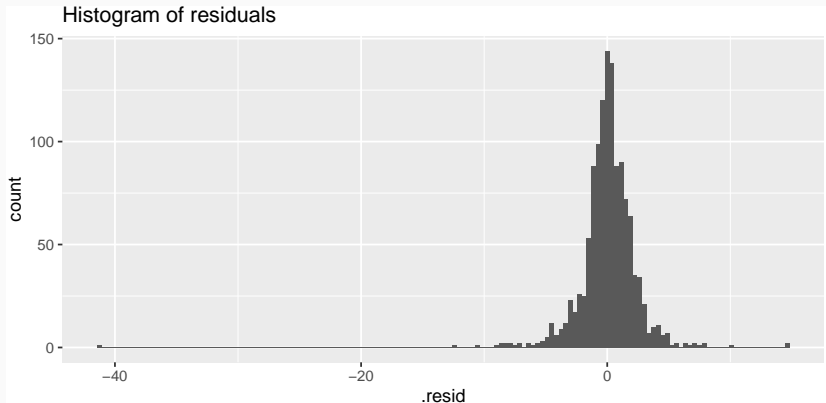
# Facebook closing stock price

```
augment(fit) %>%  
  autoplot(.resid) + xlab("Day") + ylab("") +  
  ggtitle("Residuals from naïve method")
```



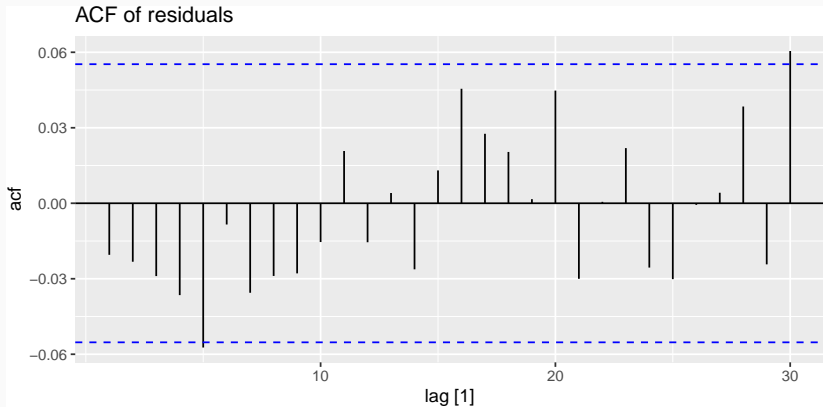
# Facebook closing stock price

```
augment(fit) %>%  
  ggplot(aes(x = .resid)) +  
  geom_histogram(bins = 150) +  
  ggtitle("Histogram of residuals")
```



# Facebook closing stock price

```
augment(fit) %>%  
  ACF(.resid) %>%  
  autoplot() + ggtitle("ACF of residuals")
```



# ACF of residuals

- We assume that the residuals are white noise (uncorrelated, mean zero, constant variance). If they aren't, then there is information left in the residuals that should be used in computing forecasts.
- So a standard residual diagnostic is to check the ACF of the residuals of a forecasting method.
- We *expect* these to look like white noise.

# Portmanteau tests

Consider a *whole set* of  $r_k$  values, and develop a test to see whether the set is significantly different from a zero set.

# Portmanteau tests

Consider a *whole* set of  $r_k$  values, and develop a test to see whether the set is significantly different from a zero set.

## Box-Pierce test

$$Q = T \sum_{k=1}^h r_k^2$$

where  $h$  is max lag being considered and  $T$  is number of observations.

- If each  $r_k$  close to zero,  $Q$  will be **small**.
- If some  $r_k$  values large (positive or negative),  $Q$  will be **large**.



# Portmanteau tests

Consider a *whole* set of  $r_k$  values, and develop a test to see whether the set is significantly different from a zero set.

## Ljung-Box test

$$Q^* = T(T+2) \sum_{k=1}^h (T-k)^{-1} r_k^2$$

where  $h$  is max lag being considered and  $T$  is number of observations.

- My preferences:  $h = 10$  for non-seasonal data,  $h = 2m$  for seasonal data.
- Better performance, especially in small samples. 37

# Portmanteau tests

- If data are WN,  $Q^*$  has  $\chi^2$  distribution with  $(h - K)$  degrees of freedom where  $K$  = no. parameters in model.
- When applied to raw data, set  $K = 0$ .

```
# lag=h and fitdf=K
```

```
Box.test(augment(fit)$resid,  
lag = 10, fitdf = 0, type = "Lj")
```

```
##
```

```
## Box-Ljung test
```

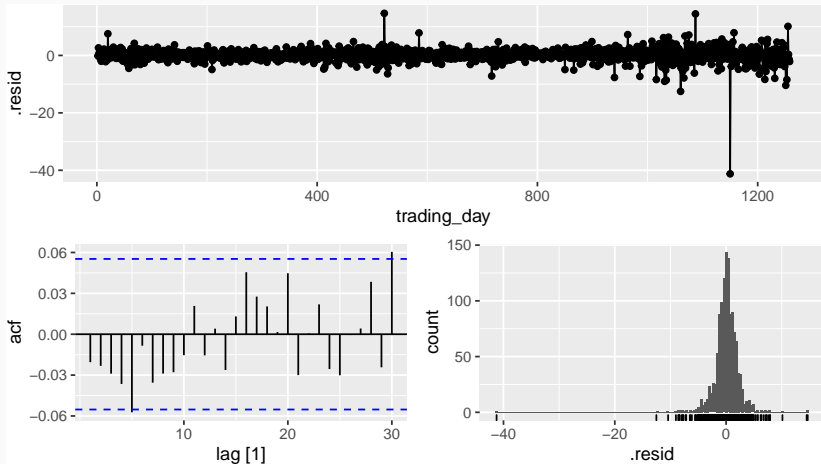
```
##
```

```
## data: augment(fit)$resid
```

```
## X-squared = 12.136, df = 10, p-value = 0.276
```

# gg\_tsresiduals function

```
gg_tsresiduals(fit)
```



# Your turn

Compute seasonal naïve forecasts for quarterly Australian beer production from 1992.

```
recent <- aus_production %>% filter(year(Quarter) >= 1992)
fit <- recent %>% model(SNAIVE(Beer))
fit %>% forecast() %>% autoplot(recent)
```

Test if the residuals are white noise.

```
Box.test(augment(fit)$resid, lag=10, fitdf=0, type="Lj")
gg_tsresiduals(fit)
```

What do you conclude?

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Forecast distributions

- A forecast  $\hat{y}_{T+h|T}$  is (usually) the mean of the conditional distribution  $y_{T+h} \mid y_1, \dots, y_T$ .
- Most time series models produce normally distributed forecasts.
- The forecast distribution describes the probability of observing any future value.

# Forecast distributions

Assuming residuals are normal, uncorrelated,  $\text{sd} = \hat{\sigma}$ :

**Mean:**  $\hat{y}_{T+h|T} \sim N(\bar{y}, (1 + 1/T)\hat{\sigma}^2)$

**Naïve:**  $\hat{y}_{T+h|T} \sim N(y_T, h\hat{\sigma}^2)$

**Seasonal naïve:**  $\hat{y}_{T+h|T} \sim N(y_{T+h-m(k+1)}, (k+1)\hat{\sigma}^2)$

**Drift:**  $\hat{y}_{T+h|T} \sim N(y_T + \frac{h}{T-1}(y_T - y_1), h\frac{T+h}{T}\hat{\sigma}^2)$

where  $k$  is the integer part of  $(h - 1)/m$ .

Note that when  $h = 1$  and  $T$  is large, these all give the same approximate forecast variance:  $\hat{\sigma}^2$ .

# Prediction intervals

- A prediction interval gives a region within which we expect  $y_{T+h}$  to lie with a specified probability.
- Assuming forecast errors are normally distributed, then a 95% PI is

$$\hat{y}_{T+h|T} \pm 1.96\hat{\sigma}_h$$

where  $\hat{\sigma}_h$  is the st dev of the  $h$ -step distribution.

- When  $h = 1$ ,  $\hat{\sigma}_h$  can be estimated from the residuals.



# Prediction intervals

```
brick_fc %>% hilo(level = 95)
```

```
## # A tsibble: 80 x 4 [1Q]
```

```
## # Key:           .model [4]
```

| ## | .model            | Quarter | Bricks | `95%`                  |
|----|-------------------|---------|--------|------------------------|
| ## | <chr>             | <qtr>   | <dbl>  | <hilo>                 |
| ## | 1 Seasonal_naïve  | 2005 Q3 | 428    | [333.2737, 522.7263]95 |
| ## | 2 Seasonal_naïve  | 2005 Q4 | 397    | [302.2737, 491.7263]95 |
| ## | 3 Seasonal_naïve  | 2006 Q1 | 355    | [260.2737, 449.7263]95 |
| ## | 4 Seasonal_naïve  | 2006 Q2 | 435    | [340.2737, 529.7263]95 |
| ## | 5 Seasonal_naïve  | 2006 Q3 | 428    | [294.0368, 561.9632]95 |
| ## | 6 Seasonal_naïve  | 2006 Q4 | 397    | [263.0368, 530.9632]95 |
| ## | 7 Seasonal_naïve  | 2007 Q1 | 355    | [221.0368, 488.9632]95 |
| ## | 8 Seasonal_naïve  | 2007 Q2 | 435    | [301.0368, 568.9632]95 |
| ## | 9 Seasonal_naïve  | 2007 Q3 | 428    | [263.9292, 592.0708]95 |
| ## | 10 Seasonal_naïve | 2007 Q4 | 397    | [232.9292, 561.0708]95 |

# Prediction intervals

- Point forecasts are often useless without a measure of uncertainty (such as prediction intervals).
- Prediction intervals require a stochastic model (with random errors, etc).
- Multi-step forecasts for time series require a more sophisticated approach (with PI getting wider as the forecast horizon increases).

# Prediction intervals

- Computed automatically from the forecast distribution.
- Use `level` argument to control coverage.
- Check residual assumptions before believing them (we will see this next class).
- Usually too narrow due to unaccounted uncertainty.

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations**
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Modelling with transformations

Transformations used in the left of the formula will be automatically back-transformed. To model log-transformed food retailing turnover, you could use:

```
food <- aus_retail %>%  
  filter(Industry == "Food retailing") %>%  
  summarise(Turnover = sum(Turnover))
```

```
fit <- food %>%  
  model(SNAIVE(log(Turnover)))
```

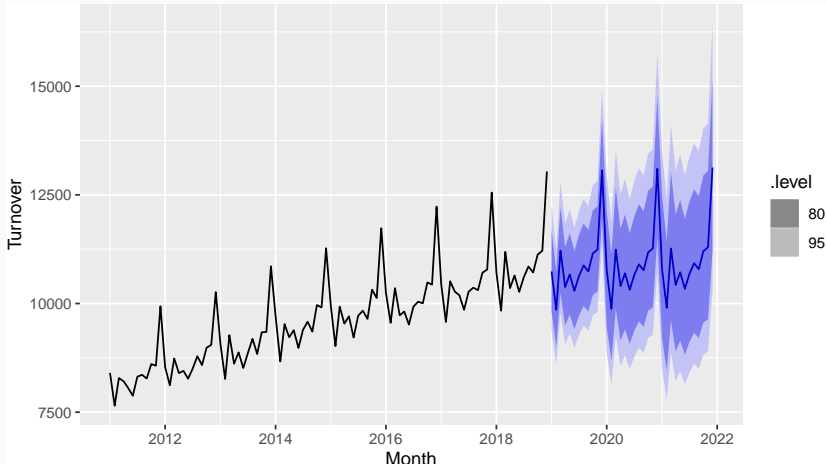
# Forecasting with transformations

```
fc <- fit %>%  
  forecast(h = "3 years")
```

```
## # A tibble: 36 x 4 [1M]  
## # Key:   .model [1]  
##   .model                Month Turnover .distribution  
##   <chr>                <mth>      <dbl> <dist>  
## 1 SNAIVE(log(Turnover)) 2019 Jan    10738. t(N(9.3, 0.0047))  
## 2 SNAIVE(log(Turnover)) 2019 Feb     9856. t(N(9.2, 0.0047))  
## 3 SNAIVE(log(Turnover)) 2019 Mar    11214. t(N(9.3, 0.0047))  
## 4 SNAIVE(log(Turnover)) 2019 Apr    10378. t(N(9.2, 0.0047))  
## 5 SNAIVE(log(Turnover)) 2019 May    10670. t(N(9.3, 0.0047))  
## 6 SNAIVE(log(Turnover)) 2019 Jun    10292. t(N(9.2, 0.0047))  
## # ... with 30 more rows
```

# Forecasting with transformations

```
fc %>% autoplot(filter(food, year(Month) > 2010))
```



# Bias adjustment

- Back-transformed point forecasts are medians.
- Back-transformed PI have the correct coverage.



# Bias adjustment

- Back-transformed point forecasts are medians.
- Back-transformed PI have the correct coverage.

## Back-transformed means

Let  $X$  be have mean  $\mu$  and variance  $\sigma^2$ .

Let  $f(x)$  be back-transformation function, and  $Y = f(X)$ .

Taylor series expansion about  $\mu$ :

$$f(X) = f(\mu) + (X - \mu)f'(\mu) + \frac{1}{2}(X - \mu)^2f''(\mu).$$

# Bias adjustment

- Back-transformed point forecasts are medians.
- Back-transformed PI have the correct coverage.

## Back-transformed means

Let  $X$  be have mean  $\mu$  and variance  $\sigma^2$ .

Let  $f(x)$  be back-transformation function, and  $Y = f(X)$ .

Taylor series expansion about  $\mu$ :

$$f(X) = f(\mu) + (X - \mu)f'(\mu) + \frac{1}{2}(X - \mu)^2f''(\mu).$$

$$E[Y] = E[f(X)] = f(\mu) + \frac{1}{2}\sigma^2f''(\mu)$$

# Bias adjustment

**Box-Cox back-transformation:**

$$y_t = \begin{cases} \exp(w_t) & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

$$f(x) = \begin{cases} e^x & \lambda = 0; \\ (\lambda x + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

$$f''(x) = \begin{cases} e^x & \lambda = 0; \\ (1 - \lambda)(\lambda x + 1)^{1/\lambda - 2} & \lambda \neq 0. \end{cases}$$

# Bias adjustment

**Box-Cox back-transformation:**

$$y_t = \begin{cases} \exp(w_t) & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

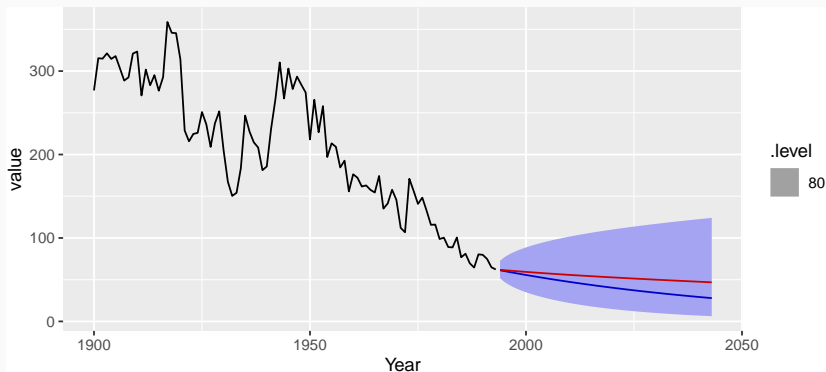
$$f(x) = \begin{cases} e^x & \lambda = 0; \\ (\lambda x + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

$$f''(x) = \begin{cases} e^x & \lambda = 0; \\ (1 - \lambda)(\lambda x + 1)^{1/\lambda - 2} & \lambda \neq 0. \end{cases}$$

$$E[Y] = \begin{cases} e^{\mu} \left[ 1 + \frac{\sigma^2}{2} \right] & \lambda = 0; \\ (\lambda \mu + 1)^{1/\lambda} \left[ 1 + \frac{\sigma^2(1-\lambda)}{2(\lambda \mu + 1)^2} \right] & \lambda \neq 0. \end{cases}$$

# Bias adjustment

```
eggs <- as_tsibble(fma::eggs)
fit <- eggs %>% model(RW(log(value) ~ drift()))
fc <- fit %>% forecast(h=50)
fc_biased <- fit %>% forecast(h=50, bias_adjust = FALSE)
eggs %>% autoplot(value) + xlab("Year") +
  autolayer(fc_biased, level = 80) +
  autolayer(fc, colour = "red", level = NULL)
```



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Forecasting and decomposition

- Forecast seasonal component by repeating the last year
- Forecast seasonally adjusted data using non-seasonal time series method.
- Combine forecasts of seasonal component with forecasts of seasonally adjusted data to get forecasts of original data.
- Sometimes a decomposition is useful just for understanding the data before building a separate forecasting model.

# US Retail Employment

```
us_retail_employment <- us_employment %>%  
  filter(year(Month) >= 1990, Title == "Retail Trade") %>%  
  select(-Series_ID)  
us_retail_employment
```

```
## # A tsibble: 357 x 3 [1M]  
##      Month Title      Employed  
##      <mth> <chr>      <dbl>  
## 1 1990 Jan Retail Trade 13256.  
## 2 1990 Feb Retail Trade 12966.  
## 3 1990 Mar Retail Trade 12938.  
## 4 1990 Apr Retail Trade 13012.  
## 5 1990 May Retail Trade 13108.  
## 6 1990 Jun Retail Trade 13183.  
## 7 1990 Jul Retail Trade 13170.  
## 8 1990 Aug Retail Trade 13160.  
## 9 1990 Sep Retail Trade 13113.  
## 10 1990 Oct Retail Trade 13185.
```



# US Retail Employment

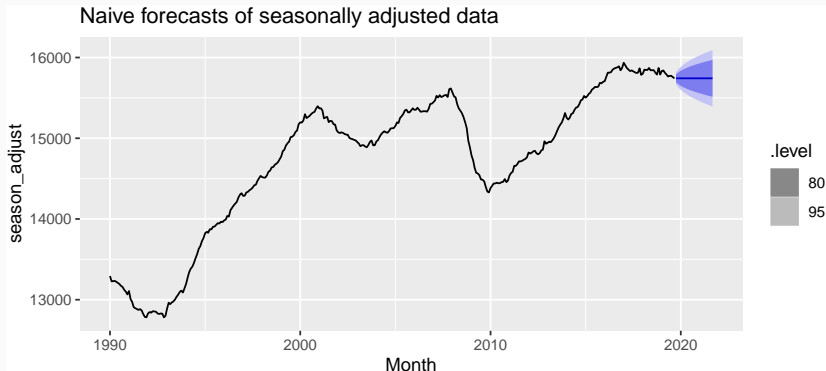
```
dcmp <- us_retail_employment %>%  
  model(STL(Employed)) %>%  
  components() %>% select(-.model)  
dcmp
```

```
## # A tsibble: 357 x 6 [1M]
```

| ## |    | Month    | Employed | trend  | season_year | remainder |
|----|----|----------|----------|--------|-------------|-----------|
| ## |    | <mth>    | <dbl>    | <dbl>  | <dbl>       | <dbl>     |
| ## | 1  | 1990 Jan | 13256.   | 13291. | -38.1       | 3.08      |
| ## | 2  | 1990 Feb | 12966.   | 13272. | -261.       | -44.2     |
| ## | 3  | 1990 Mar | 12938.   | 13252. | -291.       | -23.0     |
| ## | 4  | 1990 Apr | 13012.   | 13233. | -221.       | 0.0892    |
| ## | 5  | 1990 May | 13108.   | 13213. | -115.       | 9.98      |
| ## | 6  | 1990 Jun | 13183.   | 13193. | -25.6       | 15.7      |
| ## | 7  | 1990 Jul | 13170.   | 13173. | -24.4       | 22.0      |
| ## | 8  | 1990 Aug | 13160.   | 13152. | -11.8       | 19.5      |
| ## | 9  | 1990 Sep | 13113.   | 13131. | -43.4       | 25.7      |
| ## | 10 | 1990 Oct | 13185.   | 13110. | -62.5       | 12.2      |

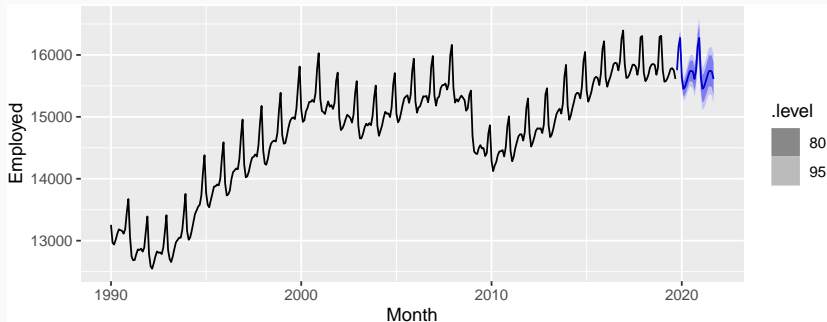
# US Retail Employment

```
dcmp %>%  
  model(NAIVE(season_adjust)) %>%  
  forecast() %>%  
  autoplot(dcmp) +  
  ggtitle("Naive forecasts of seasonally adjusted data")
```



# US Retail Employment

```
us_retail_employment %>%  
  model(stlf = decomposition_model(  
    STL(Employed ~ trend(window = 7), robust = TRUE),  
    NAIVE(season_adjust)  
  )) %>%  
  forecast() %>%  
  autoplot(us_retail_employment)
```



# Decomposition models

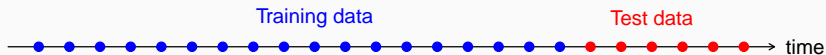
`decomposition_model()` creates a decomposition model

- You must provide a method for forecasting the `season_adjust` series.
- A seasonal naive method is used by default for the `seasonal` components.
- The variances from both the seasonally adjusted and seasonal forecasts are combined.

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Training and test sets



- 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.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.
- The test set must not be used for *any* aspect of model development or calculation of forecasts.
- Forecast accuracy is based only on the test set.

# Forecast errors

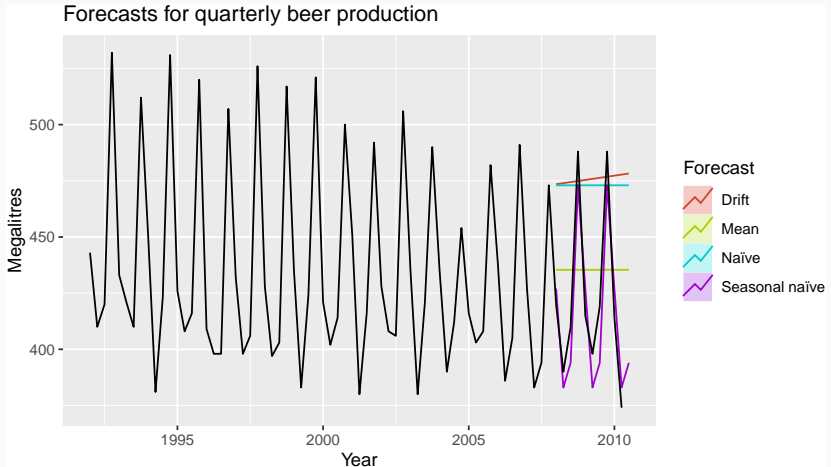
Forecast “error”: the difference between an observed value and its forecast.

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

where the training data is given by  $\{y_1, \dots, y_T\}$

- Unlike residuals, forecast errors on the test set involve multi-step forecasts.
- These are *true* forecast errors as the test data is not used in computing  $\hat{y}_{T+h|T}$ .

# Measures of forecast accuracy





# Measures of forecast accuracy

$y_{T+h}$  =  $(T + h)$ th observation,  $h = 1, \dots, H$

$\hat{y}_{T+h|T}$  = its forecast based on data up to time  $T$ .

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T}$$

$$\text{MAE} = \text{mean}(|e_{T+h}|)$$

$$\text{MSE} = \text{mean}(e_{T+h}^2)$$

$$\text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)}$$

$$\text{MAPE} = 100\text{mean}(|e_{T+h}|/|y_{T+h}|)$$

# Measures of forecast accuracy

$y_{T+h}$  =  $(T + h)$ th observation,  $h = 1, \dots, H$

$\hat{y}_{T+h|T}$  = its forecast based on data up to time  $T$ .

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T}$$

$$\text{MAE} = \text{mean}(|e_{T+h}|)$$

$$\text{MSE} = \text{mean}(e_{T+h}^2)$$

$$\text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)}$$

$$\text{MAPE} = 100\text{mean}(|e_{T+h}|/|y_{T+h}|)$$

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if  $y_t \gg 0$  for all  $t$ , and  $y$  has a natural zero.

# Measures of forecast accuracy

## Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where  $Q$  is a stable measure of the scale of the time series  $\{y_t\}$ .

Proposed by Hyndman and Koehler (IJF, 2006).

For non-seasonal time series,

$$Q = (T - 1)^{-1} \sum_{t=2}^T |y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naïve method.

# Measures of forecast accuracy

## Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where  $Q$  is a stable measure of the scale of the time series  $\{y_t\}$ .

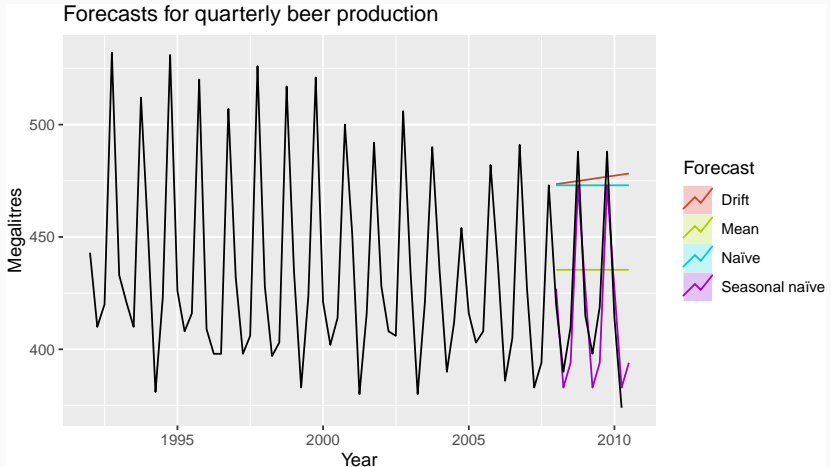
Proposed by Hyndman and Koehler (IJF, 2006).

For seasonal time series,

$$Q = (T - m)^{-1} \sum_{t=m+1}^T |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naïve method.

# Measures of forecast accuracy



# Measures of forecast accuracy

```
recent_production <- aus_production %>%  
  filter(year(Quarter) >= 1992)  
train <- recent_production %>%  
  filter(year(Quarter) <= 2007)  
beer_fit <- train %>%  
  model(  
    Mean = MEAN(Beer),  
    `Naïve` = NAIVE(Beer),  
    `Seasonal naïve` = SNAIVE(Beer),  
    Drift = RW(Beer ~ drift())  
  )  
beer_fc <- beer_fit %>%  
  forecast(h = 10)
```

# Measures of forecast accuracy

```
accuracy(beer_fit)
```

```
## # A tibble: 4 x 6
##   .model      .type    RMSE    MAE    MAPE    MASE
##   <chr>      <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 Drift      Training  65.3   54.8  12.2   3.83
## 2 Mean      Training  43.6   35.2   7.89   2.46
## 3 Naïve     Training  65.3   54.7  12.2   3.83
## 4 Seasonal naïve Training  16.8   14.3   3.31   1
```

```
accuracy(beer_fc, recent_production)
```

```
## # A tibble: 4 x 6
##   .model      .type    RMSE    MAE    MAPE    MASE
##   <chr>      <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 Drift      Test     64.9   58.9  14.6   4.12
## 2 Mean      Test     38.4   34.8   8.28   2.44
## 3 Naïve     Test     62.7   57.4  14.2   4.01
## 4 Seasonal naïve Test     14.3   13.4   3.17  0.937
```

## Poll: true or false?

- 1 Good forecast methods should have normally distributed residuals.
- 2 A model with small residuals will give good forecasts.
- 3 The best measure of forecast accuracy is MAPE.
- 4 If your model doesn't forecast well, you should make it more complicated.
- 5 Always choose the model with the best forecast accuracy as measured on the test set.



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 Residual diagnostics
- 4 Distributional forecasts and prediction intervals
- 5 Forecasting with transformations
- 6 Forecasting and decomposition
- 7 Evaluating forecast accuracy
- 8 Time series cross-validation

# Time series cross-validation

## Traditional evaluation

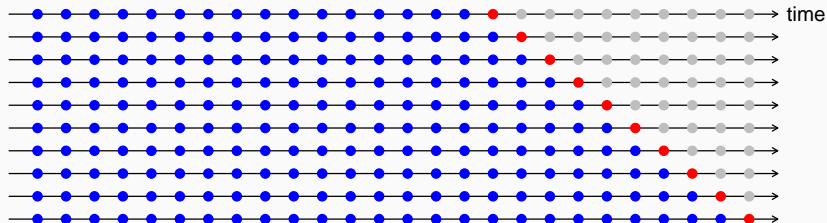


# Time series cross-validation

## Traditional evaluation



## Time series cross-validation

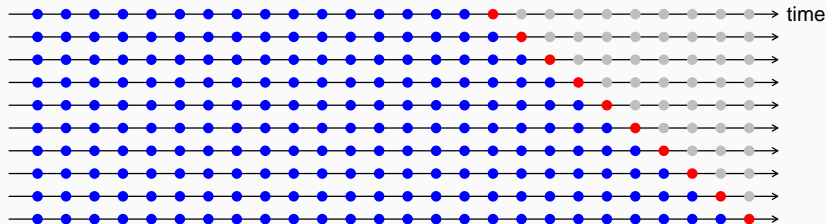


# Time series cross-validation

## Traditional evaluation



## Time series cross-validation



- Forecast accuracy averaged over test sets.
- Also known as “evaluation on a rolling forecasting origin”

# Creating the rolling training sets

There are three main rolling types which can be used.

- Stretch: extends a growing length window with new data.
- Slide: shifts a fixed length window through the data.
- Tile: moves a fixed length window without overlap.

Three functions to roll a tsibble: `stretch_tsibble()`, `slide_tsibble()`, and `tile_tsibble()`.

For time series cross-validation, stretching windows are most commonly used.

# Creating the rolling training sets

# Time series cross-validation

Stretch with a minimum length of 3, growing by 1 each step.

```
fb_stretch <- fb_stock %>%  
  stretch_tsibble(.init = 3, .step = 1) %>%  
  filter(.id != max(.id))
```

```
## # A tsibble: 790,650 x 4 [1]  
## # Key:           .id [1,255]  
##   Date           Close trading_day   .id  
##   <date>         <dbl>         <int> <int>  
## 1 2014-01-02    54.7             1     1  
## 2 2014-01-03    54.6             2     1  
## 3 2014-01-06    57.2             3     1  
## 4 2014-01-02    54.7             1     2  
## 5 2014-01-03    54.6             2     2  
## 6 2014-01-06    57.2             3     2  
## 7 2014-01-07    57.9             4     2
```

# Time series cross-validation

Estimate RW w/ drift models for each window.

```
fit_cv <- fb_stretch %>%  
  model(RW(Close ~ drift()))
```

```
## # A mable: 1,255 x 3  
## # Key:      .id, Symbol [1,255]  
##      .id Symbol `RW(Close ~ drift())`  
##    <int> <chr>   <model>  
## 1      1 FB      <RW w/ drift>  
## 2      2 FB      <RW w/ drift>  
## 3      3 FB      <RW w/ drift>  
## 4      4 FB      <RW w/ drift>  
## # ... with 1,251 more rows
```



# Time series cross-validation

Produce one step ahead forecasts from all models.

```
fc_cv <- fit_cv %>%  
  forecast(h=1)
```

```
## # A tibble: 1,255 x 5  
## # Key:   .id, Symbol [1,255]  
##       .id Symbol trading_day Close .distribution  
##   <int> <chr>          <dbl> <dbl> <dist>  
## 1     1  1 FB              4  58.4 N(58, 3.9)  
## 2     2  2 FB              5  59.0 N(59, 2)  
## 3     3  3 FB              6  59.1 N(59, 1.5)  
## 4     4  4 FB              7  57.7 N(58, 1.8)  
## # ... with 1,251 more rows
```

# Time series cross-validation

```
# Cross-validated  
fc_cv %>% accuracy(fb_stock)  
# Training set  
fb_stock %>% model(RW(Close ~ drift())) %>% accuracy()
```

|                  | RMSE  | MAE   | MAPE  |
|------------------|-------|-------|-------|
| Cross-validation | 2.418 | 1.469 | 1.266 |
| Training         | 2.414 | 1.465 | 1.261 |

A good way to choose the best forecasting model is to find the model with the smallest RMSE computed using time series cross-validation.