

# **ETC3550**

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

Ch3. The forecasters' toolbox

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

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations
- 5 Distributional forecasts

# Outline

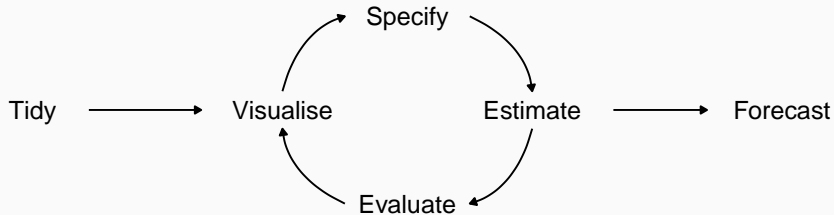
- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations
- 5 Distributional forecasts

# 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

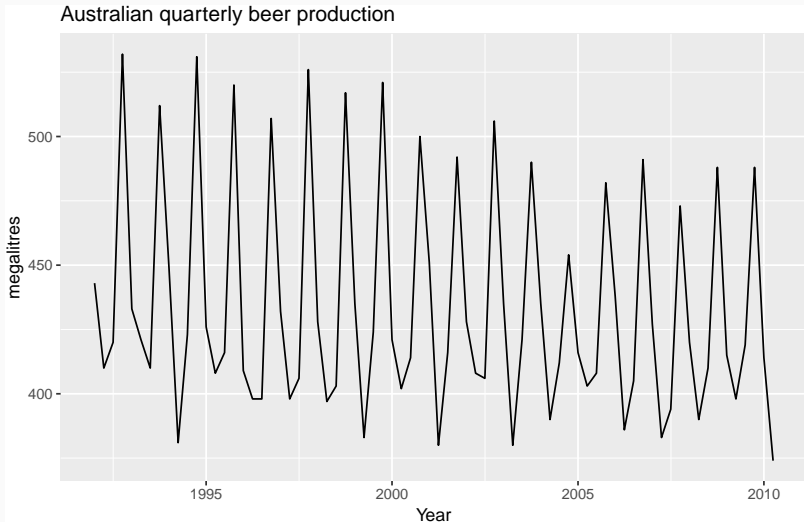
# A tidy forecasting workflow



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations
- 5 Distributional forecasts

# Some simple forecasting methods



How would you forecast these series?

# Some simple forecasting methods



How would you forecast these series?



# Some simple forecasting methods

Facebook closing stock price in 2018

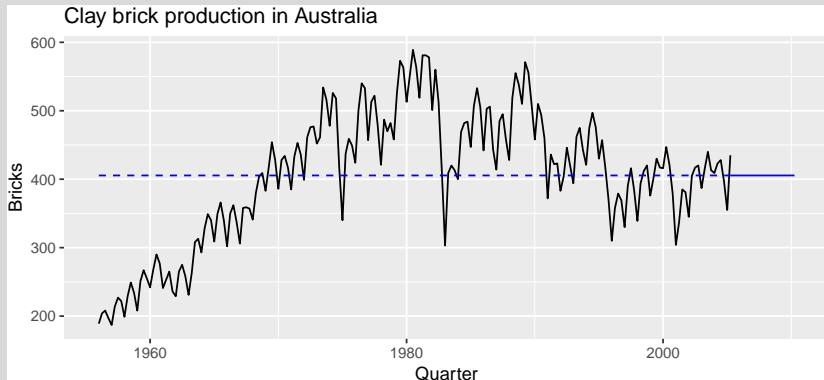


How would you forecast these series?

# Some simple forecasting methods

## MEAN( $y$ ): Average method

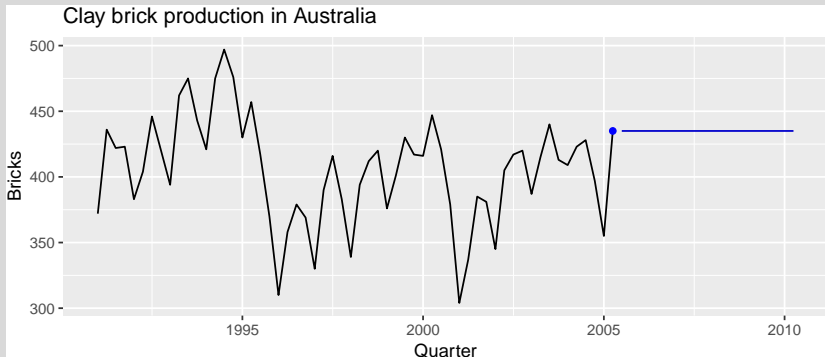
- Forecast of all future values is equal to mean of historical data  $\{y_1, \dots, y_T\}$ .
- Forecasts:  $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \dots + y_T)/T$



# Some simple forecasting methods

## NAIVE(y): Naïve method

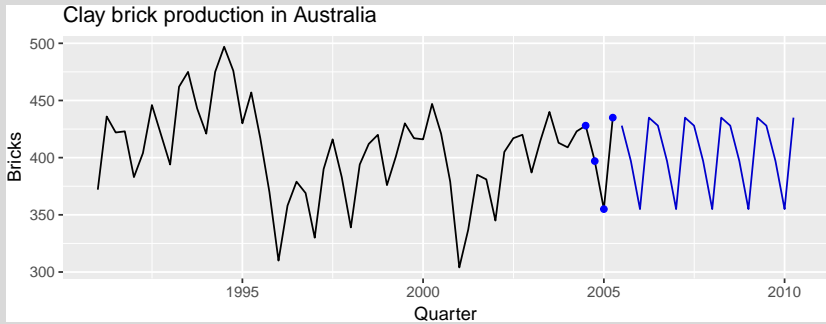
- Forecasts equal to last observed value.
- Forecasts:  $\hat{y}_{T+h|T} = y_T$ .
- Consequence of efficient market hypothesis.



# 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

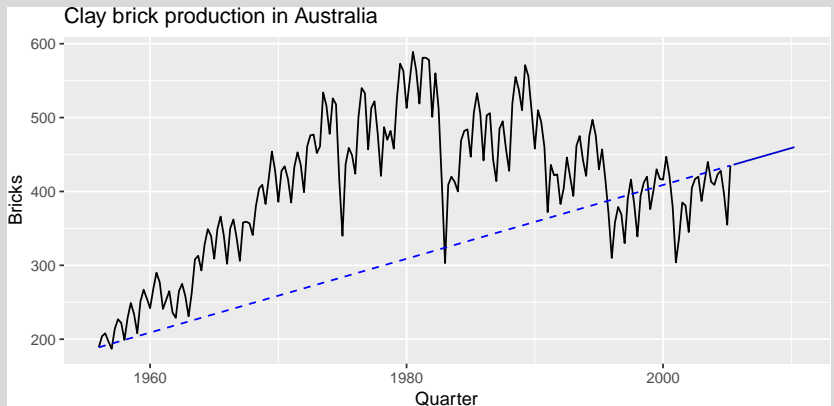
- Forecasts equal to last value plus average change.
- Forecasts:

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

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

# Some simple forecasting methods

## Drift method

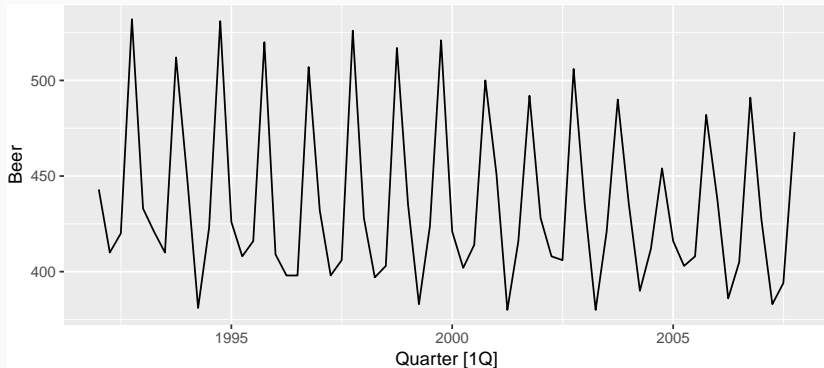


# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations
- 5 Distributional forecasts

# Data preparation and visualisation

```
# Set training data from 1992 to 2007  
train <- aus_production %>%  
  filter(between(year(Quarter), 1992, 2007))  
train %>% autoplot(Beer)
```





# Model estimation

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

```
# Fit the models
beer_fit <- train %>%
  model(
    Mean = MEAN(Beer),
    Naïve = NAIVE(Beer),
    Seasonal naïve = SNAIVE(Beer),
    Drift = RW(Beer ~ drift())
  )
```

# Model estimation

```
beer_fit
```

```
## # A mable: 1 x 4
```

```
##   Mean                Naïve   Seasonal naïve Drift
```

```
##   <model>             <model> <model>             <model>
```

```
## 1 <MEAN,  $\mu=435.375$ > <NAIVE> <SNAIVE>             <RW w/ drift>
```

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

# Producing forecasts

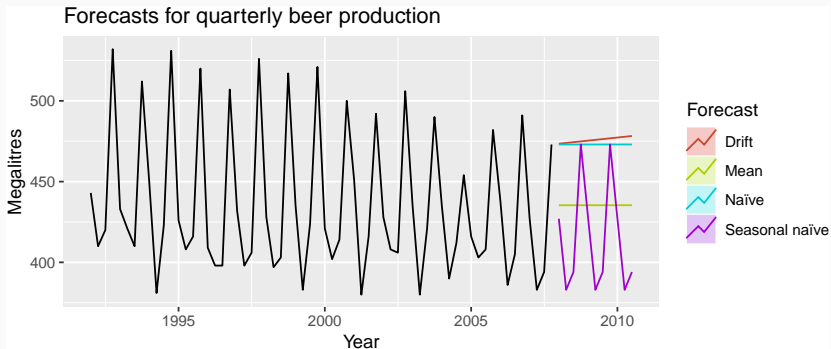
```
beer_fc <- beer_fit %>%  
  forecast(h = 11)
```

```
## # A fable: 44 x 4 [1Q]  
## # Key:      .model [4]  
##   .model Quarter Beer .distribution  
##   <chr>      <qtr> <dbl> <dist>  
## 1 Mean      2008 Q1  435. N(435, 1964)  
## 2 Mean      2008 Q2  435. N(435, 1964)  
## 3 Mean      2008 Q3  435. N(435, 1964)  
## 4 Mean      2008 Q4  435. N(435, 1964)  
## # ... with 40 more rows
```

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

# Visualising forecasts

```
beer_fc %>%  
  autoplot(train, level = NULL) +  
  ggtitle("Forecasts for quarterly beer production") +  
  xlab("Year") + ylab("Megalitres") +  
  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) %>%
  filter(Symbol == "FB",
         between(Date, ymd("2018-01-01"), ymd("2018-09-01")))

# 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 (daily ending Sep 2018)") +
  xlab("Day") + ylab("") +
  guides(colour=guide_legend(title="Forecast"))
```

# Facebook closing stock price



## Your turn

- Produce forecasts from the appropriate method for Amazon closing price (`gafa_stock`) and Australian takeaway food turnover (`aus_retail`).
- Plot the results using `autoplot()`.

# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations**
- 5 Distributional forecasts



# Variance stabilization

If the data show different variation at different levels of the series, then a transformation can be useful.

# Variance stabilization

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as  $y_1, \dots, y_n$  and transformed observations as  $w_1, \dots, w_n$ .

# Variance stabilization

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as  $y_1, \dots, y_n$  and transformed observations as  $w_1, \dots, w_n$ .

## Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	↓
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

# Variance stabilization

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as  $y_1, \dots, y_n$  and transformed observations as  $w_1, \dots, w_n$ .

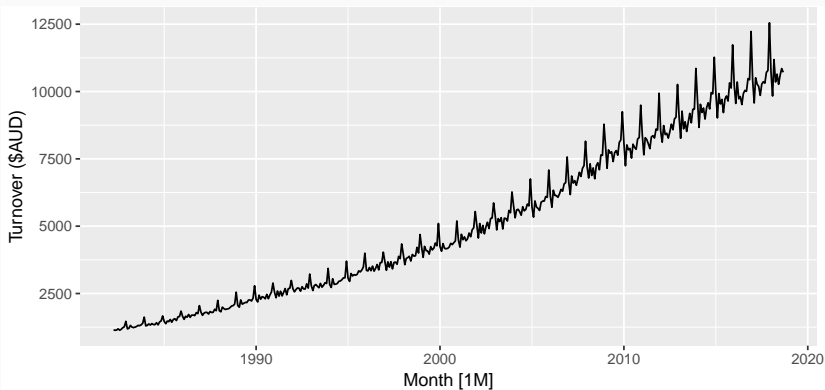
## Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	↓
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale**.

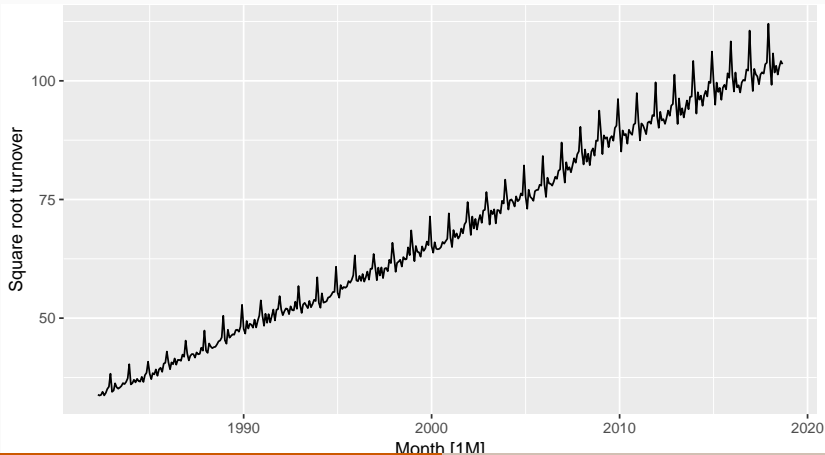
# Variance stabilization

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



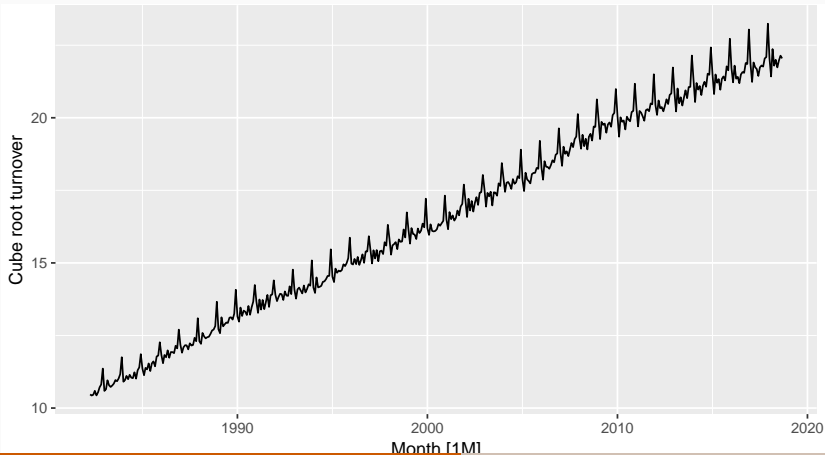
# Variance stabilization

```
food %>% autoplot(sqrt(Turnover)) +  
  labs(y = "Square root turnover")
```



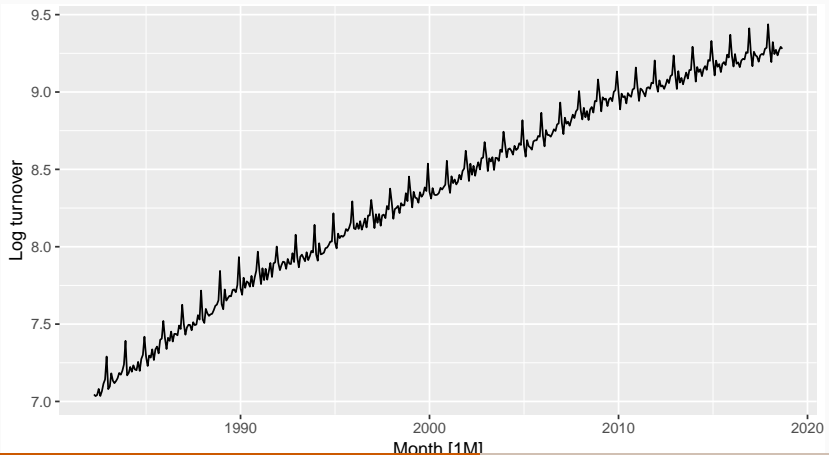
# Variance stabilization

```
food %>% autoplot(Turnover^(1/3)) +  
  labs(y = "Cube root turnover")
```



# Variance stabilization

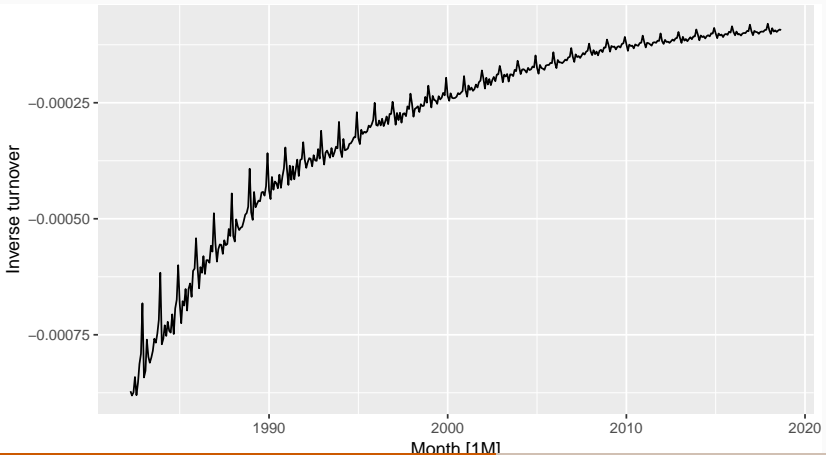
```
food %>% autoplot(log(Turnover)) +  
  labs(y = "Log turnover")
```





# Variance stabilization

```
food %>% autoplot(-1/Turnover) +  
  labs(y = "Inverse turnover")
```



# Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

# Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

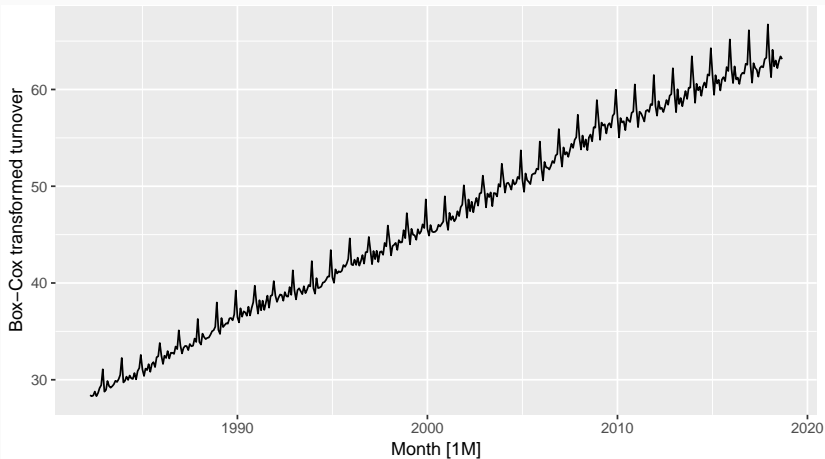
$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$ : (No substantive transformation)
- $\lambda = \frac{1}{2}$ : (Square root plus linear transformation)
- $\lambda = 0$ : (Natural logarithm)
- $\lambda = -1$ : (Inverse plus 1)

# Box-Cox transformations

# Box-Cox transformations

```
food %>% autoplot(box_cox(Turnover, 1/3)) +  
  labs(y = "Box-Cox transformed turnover")
```



# Box-Cox transformations

- $y_t^\lambda$  for  $\lambda$  close to zero behaves like logs.
- If some  $y_t = 0$ , then must have  $\lambda > 0$
- if some  $y_t < 0$ , no power transformation is possible unless all  $y_t$  adjusted by **adding a constant to all values**.
- Simple values of  $\lambda$  are easier to explain.
- Results are relatively insensitive to  $\lambda$ .
- Often no transformation ( $\lambda = 1$ ) needed.
- Transformation can have very large effect on PI.
- Choosing  $\lambda = 0$  is a simple way to force forecasts to be positive

# Box-Cox transformations

```
food %>%  
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1  
##   lambda_guerrero  
##               <dbl>  
## 1               0.00762
```

# Box-Cox transformations

```
food %>%  
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1  
##   lambda_guerrero  
##               <dbl>  
## 1             0.00762
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of  $\lambda$  can give extremely large prediction intervals.



# Back-transformation

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

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

# 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:

```
fit <- food %>%  
  model(SNAIVE(log(Turnover) ~ lag("year")))
```

```
## # A mable: 1 x 1  
##   SNAIVE(log(Turnover) ~ lag("year"))  
##   <model>  
## 1 <SNAIVE>
```

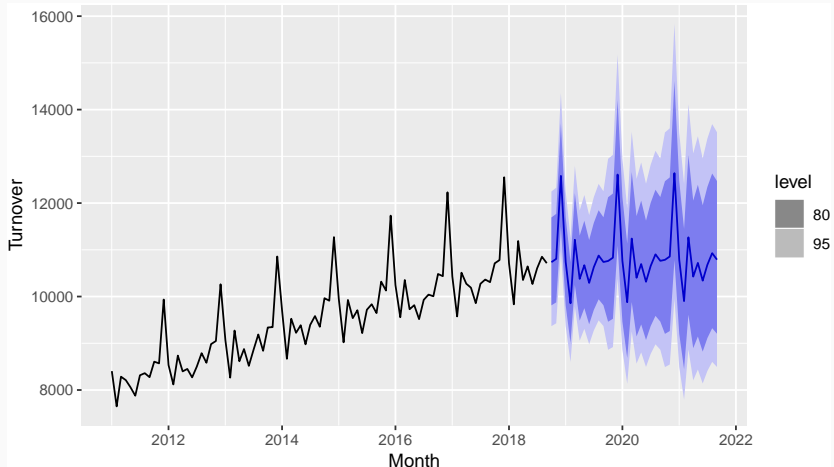
# Forecasting with transformations

```
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) ~ 2018 Oct    10736. t(N(9.3, 0.004~  
## 2 "SNAIVE(log(Turnover) ~ 2018 Nov    10806. t(N(9.3, 0.004~  
## 3 "SNAIVE(log(Turnover) ~ 2018 Dec    12581. t(N(9.4, 0.004~  
## 4 "SNAIVE(log(Turnover) ~ 2019 Jan    10738. t(N(9.3, 0.004~  
## 5 "SNAIVE(log(Turnover) ~ 2019 Feb      9856. t(N(9.2, 0.004~  
## 6 "SNAIVE(log(Turnover) ~ 2019 Mar    11215. t(N(9.3, 0.004~  
## # ... with 30 more rows
```

# Forecasting with transformations

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



# Your turn

Find a transformation that works for the Australian gas production (`aus_production`).

# 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:**

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

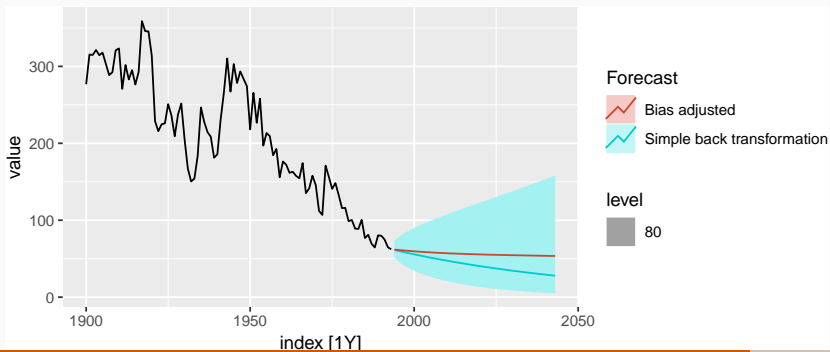
$$f(x) = \begin{cases} e^x & \lambda = 0; \\ (\lambda x + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

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

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

# 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) +
  autolayer(fc_biased, series="Simple back transformation", level=80) +
  autolayer(fc, series="Bias adjusted", level=NULL) +
  guides(colour=guide_legend(title="Forecast"))
```



# Outline

- 1 A tidy forecasting workflow
- 2 Some simple forecasting methods
- 3 The workflow in action
- 4 Transformations
- 5 Distributional forecasts

# 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

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

```
## # A tibble: 2 x 5 [1]  
## # Key:      Symbol, .model [1]  
##   Symbol .model trading_day Close .distribution  
##   <fct>  <chr>          <int> <dbl> <dist>  
## 1 FB     NAIVE(Cl~       3693  176. N(176, 21)  
## 2 FB     NAIVE(Cl~       3694  176. N(176, 42)
```



# Prediction intervals

```
res_sd <- sqrt(mean(augment(fit)$resid^2, na.rm = TRUE))  
last(fb_stock$Close) + 1.96 * res_sd * c(-1,1)
```

```
## [1] 166.7196 184.7404
```

```
forecast(fit, h = 1) %>%  
  transmute(interval = hilo(.distribution, level = 95))
```

```
## # A tsibble: 1 x 4 [?]  
## # Key:      Symbol, .model [1]  
##   Symbol .model      trading_day      interval  
##   <fct>  <chr>          <int>          <hilo>  
## 1 FB     NAIVE(Close)    3693 [166.7198, 184.7402]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.