



# ETC3550: Applied forecasting for business and economics

Ch3. The forecasters' toolbox

[OTexts.org/fpp2/](https://OTexts.org/fpp2/)

# Outline

**1** Some simple forecasting methods

**2** Box-Cox transformations

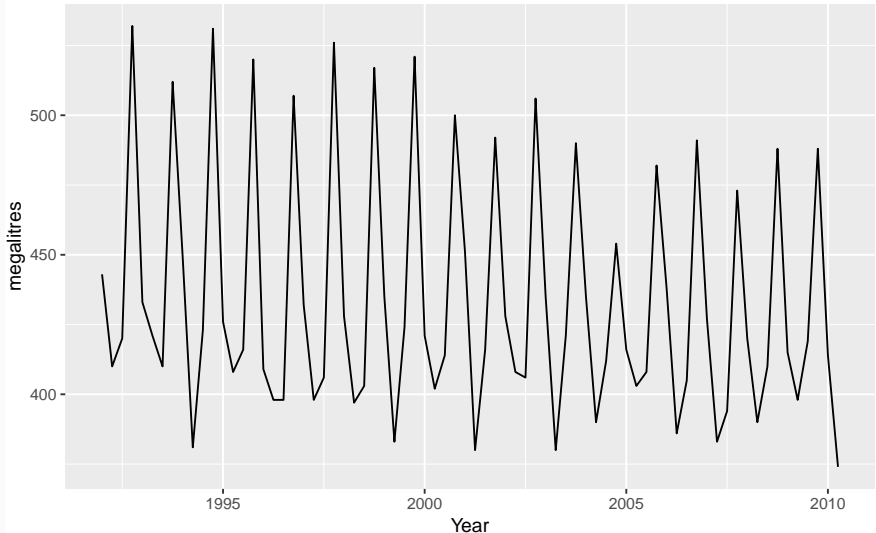
**3** Residual diagnostics

**4** Evaluating forecast accuracy

**5** Prediction intervals

# Some simple forecasting methods

Australian quarterly beer production



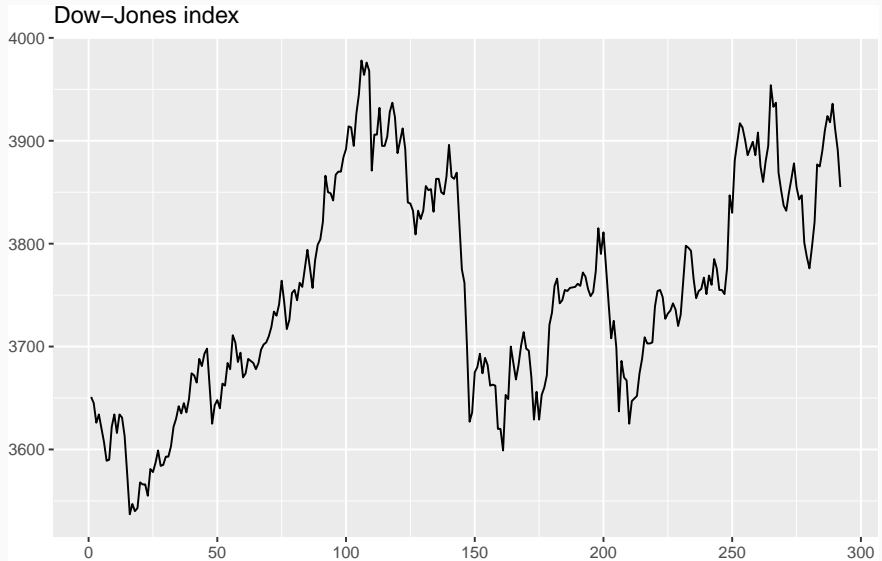
How would you forecast these data?

# Some simple forecasting methods



How would you forecast these data?

# Some simple forecasting methods



How would you forecast these data?

# Some simple forecasting methods

## 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

## Average method

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

## 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

## Average method

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

## Naïve method

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

## 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

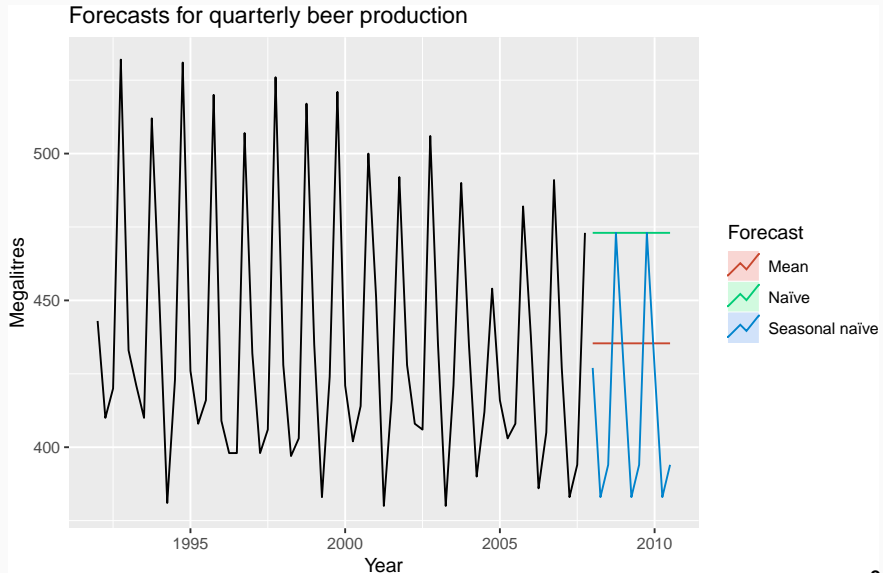
## Drift method

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

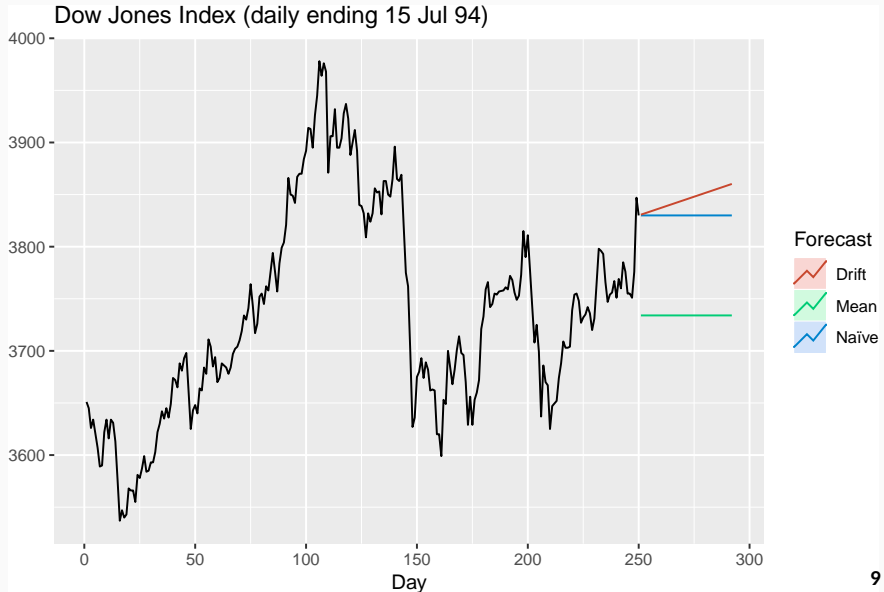
$$\begin{aligned}\hat{y}_{T+h|T} &= y_T + \frac{h}{T-1} \sum_{t=2}^T (y_t - y_{t-1}) \\ &= y_T + \frac{h}{T-1} (y_T - y_1).\end{aligned}$$

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

# Some simple forecasting methods




# Some simple forecasting methods



## Some simple forecasting methods

- Mean: `meanf(y, h=20)`
- Naïve: `naive(y, h=20)`
- Seasonal naïve: `snaive(y, h=20)`
- Drift: `rwf(y, drift=TRUE, h=20)`

# Some simple forecasting methods

- Mean: `meanf(y, h=20)`
- Naïve: `naive(y, h=20)`
- Seasonal naïve: `snaive(y, h=20)` 
- Drift: `rwf(y, drift=TRUE, h=20)`

## Your turn

- Use these four functions to produce forecasts for goog and auscafe.
- Plot the results using `autoplot()`.

# Outline

- 1 Some simple forecasting methods
- 2 Box-Cox transformations
- 3 Residual diagnostics
- 4 Evaluating forecast accuracy
- 5 Prediction intervals

# 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.

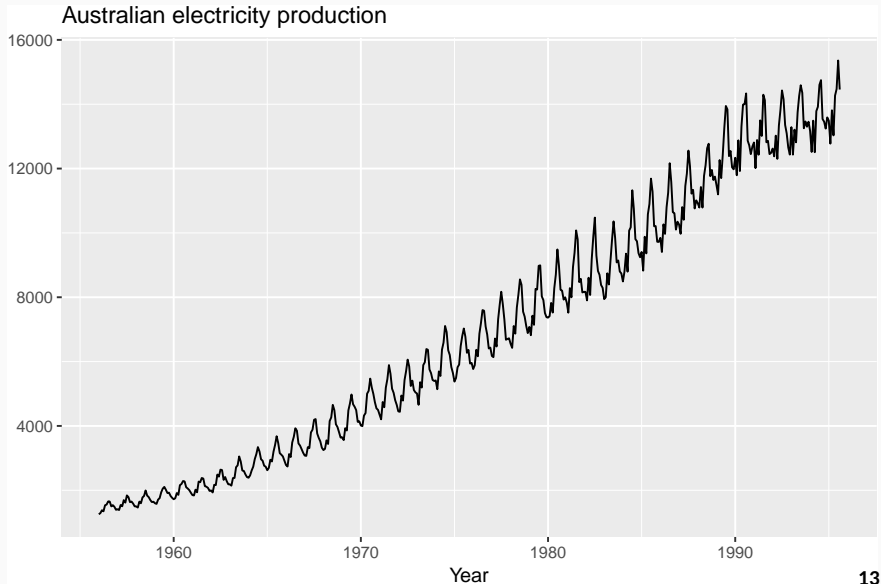
Denote original observations as  $y_1, \dots, y_n$  and transformed observations as  $w_1, \dots, w_n$ .

## Mathematical transformations for stabilizing variation

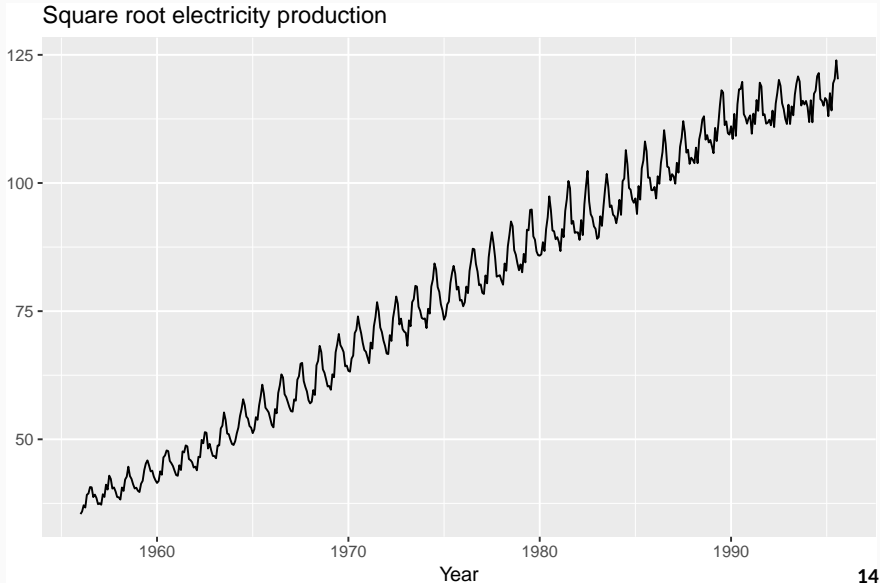
|             |                       |            |
|-------------|-----------------------|------------|
| Square root | $w_t = \sqrt{y_t}$    | ↓          |
| Cube root   | $w_t = \sqrt[3]{y_t}$ | Increasing |
| Logarithm   | $w_t = \log(y_t)$     | strength   |

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

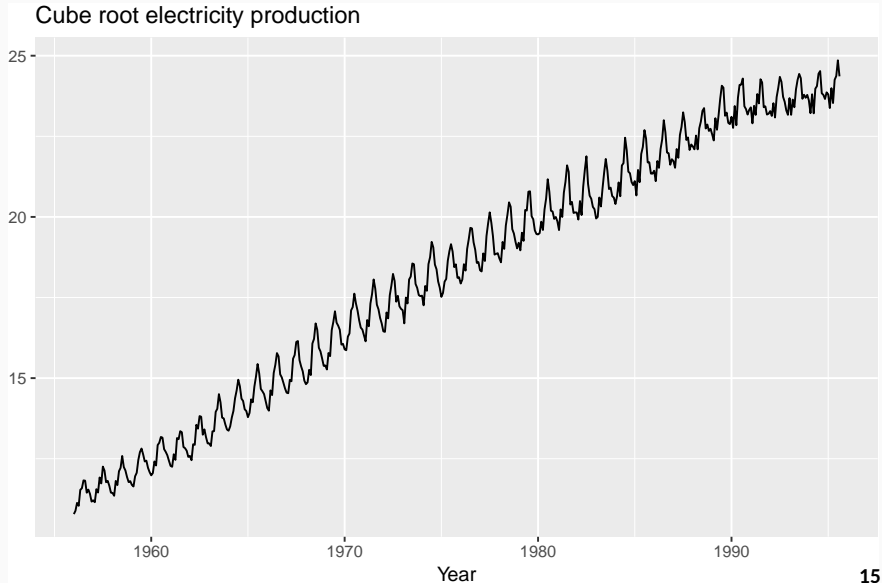
# Variance stabilization



# Variance stabilization

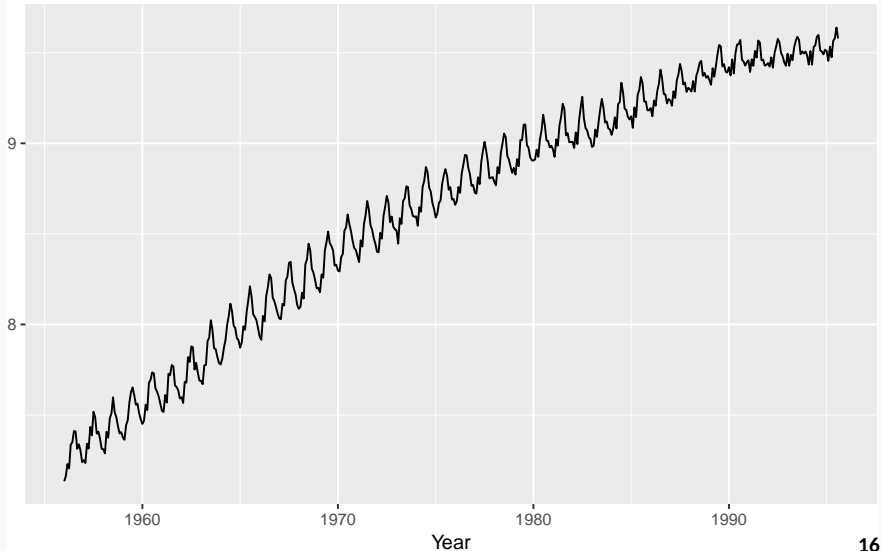


# Variance stabilization



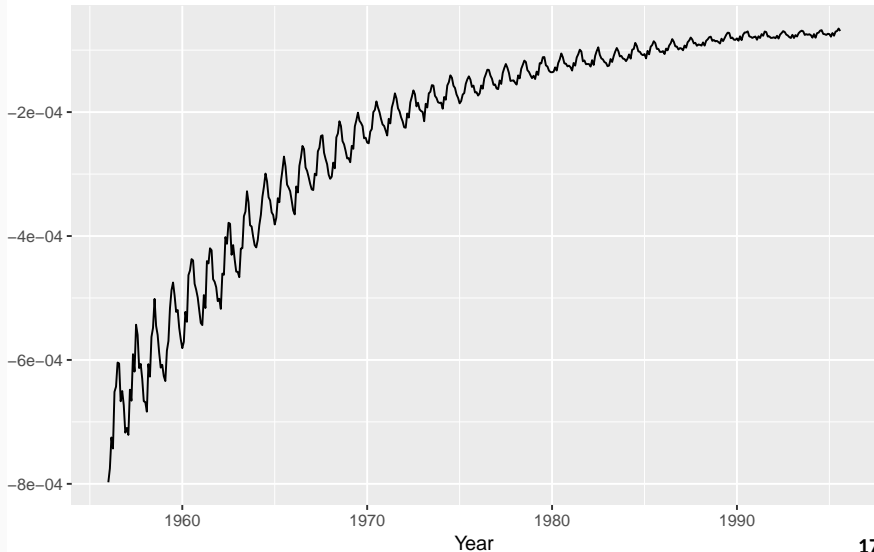
# Variance stabilization

Log electricity production



# Variance stabilization

Inverse electricity production



# 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**:

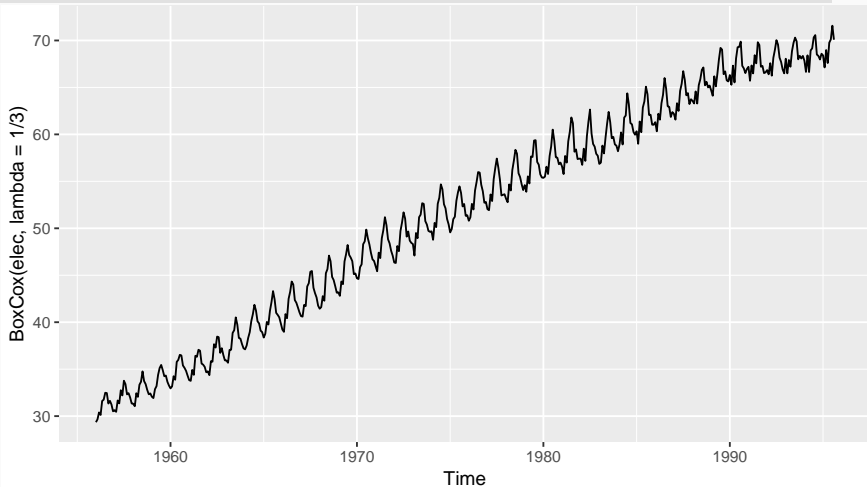
$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$ : (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

```
autoplot(BoxCox(elec, lambda=1/3))
```



# 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

# Automated Box-Cox transformations

```
(BoxCox.lambda(elec))
```

```
## [1] 0.2654076
```

# Automated Box-Cox transformations

```
(BoxCox.lambda(elec))
```

```
## [1] 0.2654076
```

- 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

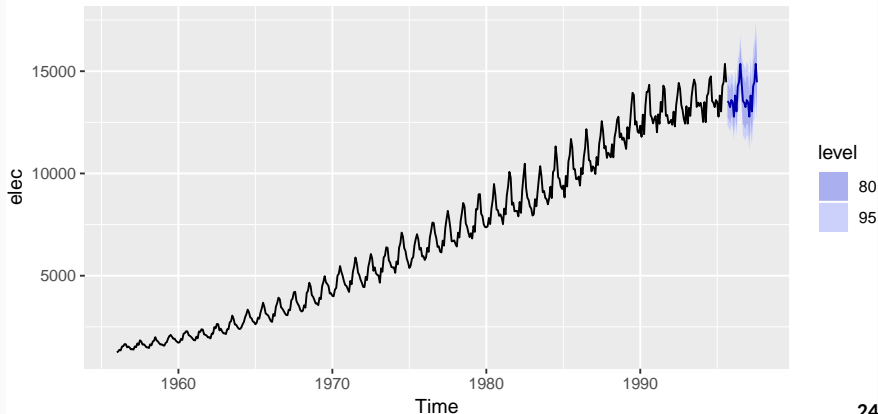
We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

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

# Back-transformation

```
fit <- snaive(elec, lambda=1/3)  
autoplot(fit)
```

Forecasts from Seasonal naive method

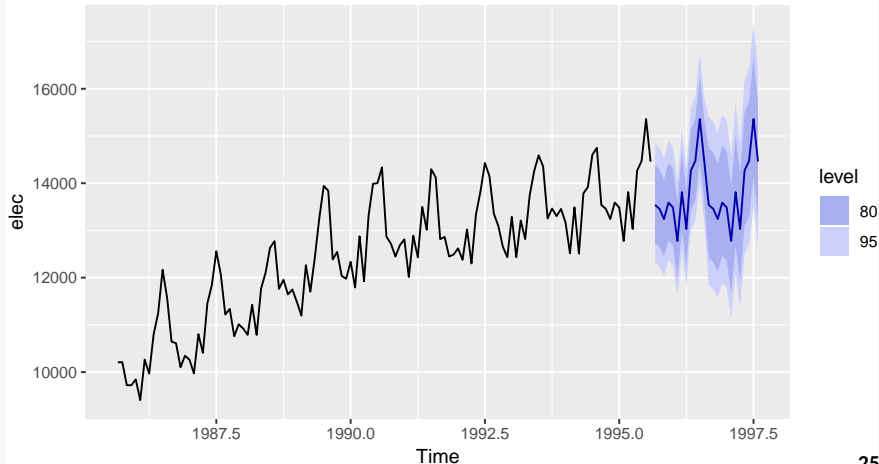




# Back-transformation

```
autoplot(fit, include=120)
```

Forecasts from Seasonal naive method



## Your turn

Find a Box-Cox transformation that works for the gas data.

# 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:**

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

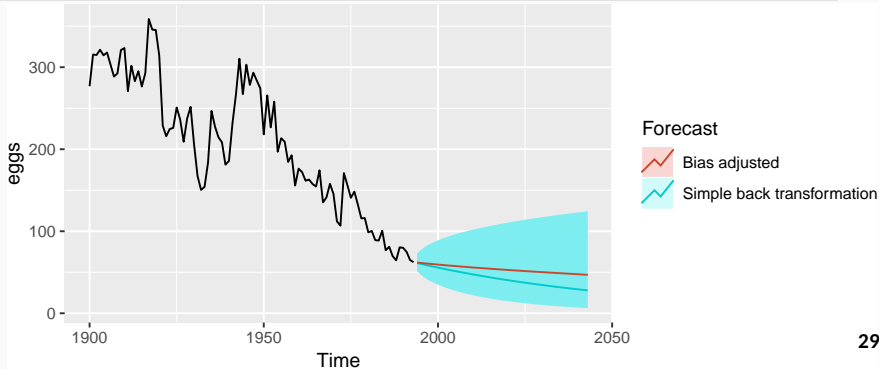
$$f(x) = \begin{cases} e^x & \lambda = 0; \\ (\lambda x + 1)^{1/\lambda} & \lambda \neq 0. \end{cases}$$

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

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

# Bias adjustment

```
fc <- rwf(eggs, drift=TRUE, lambda=0, h=50, level=80)
fc2 <- rwf(eggs, drift=TRUE, lambda=0, h=50, level=80,
  biasadj=TRUE)
autoplot(eggs) +
  autolayer(fc, series="Simple back transformation") +
  autolayer(fc2, series="Bias adjusted", PI=FALSE) +
  guides(colour=guide_legend(title="Forecast"))
```





# Outline

- 1 Some simple forecasting methods
- 2 Box-Cox transformations
- 3 Residual diagnostics
- 4 Evaluating forecast accuracy
- 5 Prediction intervals

# 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.

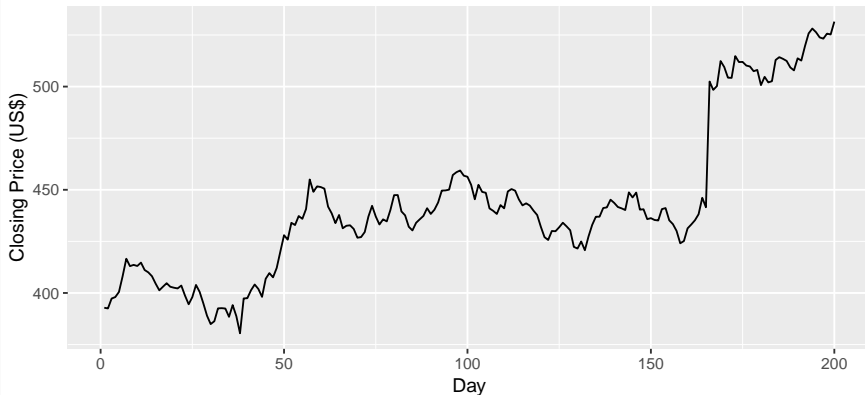
## Useful properties (for prediction intervals)

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

# Example: Google stock price

```
autoplot(goog200) +  
  xlab("Day") + ylab("Closing Price (US$)") +  
  ggtitle("Google Stock (daily ending 6 December 2013)")
```

Google Stock (daily ending 6 December 2013)



# Example: Google stock price

Naïve forecast:

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

# Example: Google stock price

Naïve forecast:

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

$$e_t = y_t - y_{t-1}$$



# Example: Google stock price

Naïve forecast:

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

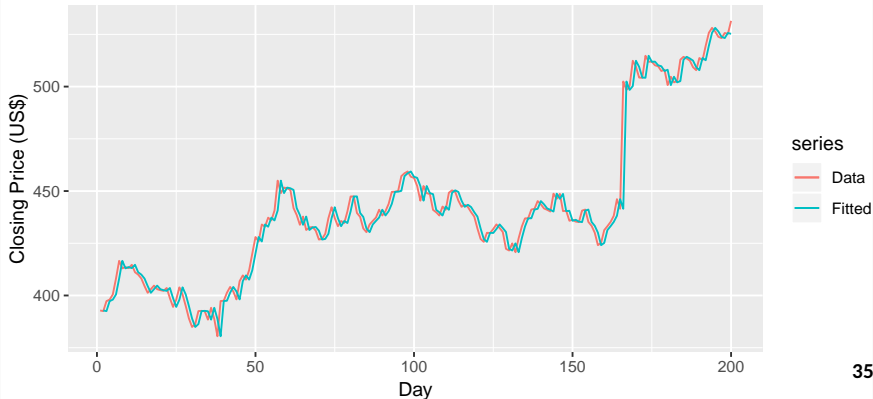
$$e_t = y_t - y_{t-1}$$

Note:  $e_t$  are one-step-forecast residuals

# Example: Google stock price

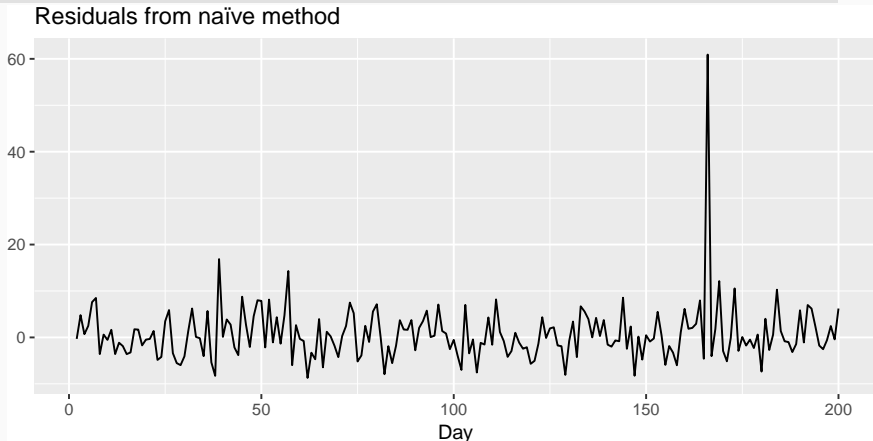
```
fits <- fitted(naive(goog200))  
autoplot(goog200, series="Data") +  
  autolayer(fits, series="Fitted") +  
  xlab("Day") + ylab("Closing Price (US$)") +  
  ggtitle("Google Stock (daily ending 6 December 2013)")
```

Google Stock (daily ending 6 December 2013)



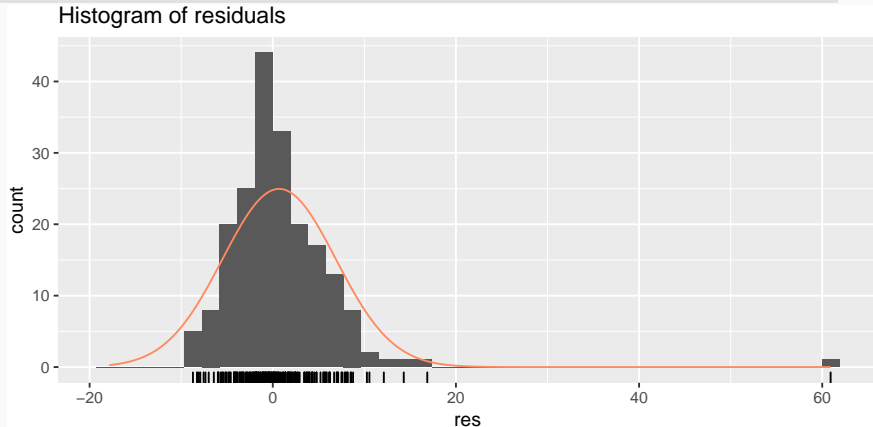
# Example: Google stock price

```
res <- residuals(naive(goog200))  
autoplot(res) + xlab("Day") + ylab("") +  
  ggtitle("Residuals from naïve method")
```



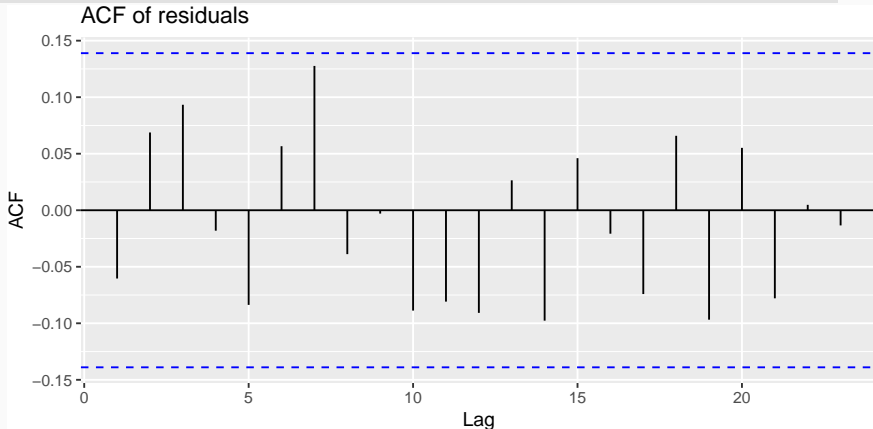
# Example: Google stock price

```
gghistogram(res, add.normal=TRUE) +  
  ggtitle("Histogram of residuals")
```



# Example: Google stock price

```
ggAcf(res) + 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.

# 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  $K = 0$ .
- For the Google example:

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

```
Box.test(res, lag=10, fitdf=0, type="Lj")
```

```
##
```

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

```
##
```

```
## data: res
```

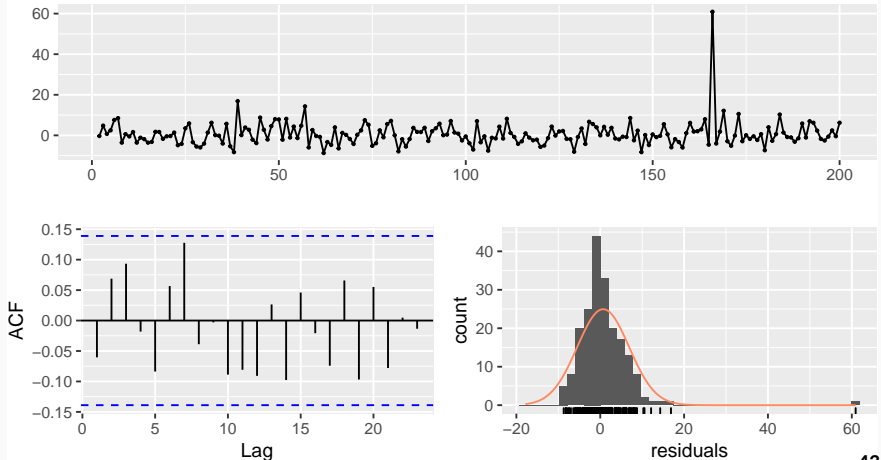
```
## X-squared = 11.031, df = 10, p-value =
```

```
## 0.3551
```

# checkresiduals function

```
checkresiduals(naive(goog200))
```

Residuals from Naive method



## checkresiduals function

```
##  
##  Ljung-Box test  
##  
## data:  Residuals from Naive method  
## Q* = 11.031, df = 10, p-value = 0.3551  
## Model df: 0.    Total lags used: 10
```

## Your turn

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

```
beer <- window(ausbeer, start=1992)
fc <- snaive(beer)
autoplot(fc)
```

Test if the residuals are white noise.

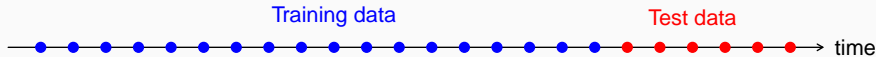
```
checkresiduals(fc)
```

What do you conclude?

# Outline

- 1 Some simple forecasting methods
- 2 Box-Cox transformations
- 3 Residual diagnostics
- 4 **Evaluating forecast accuracy**
- 5 Prediction intervals

# 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.

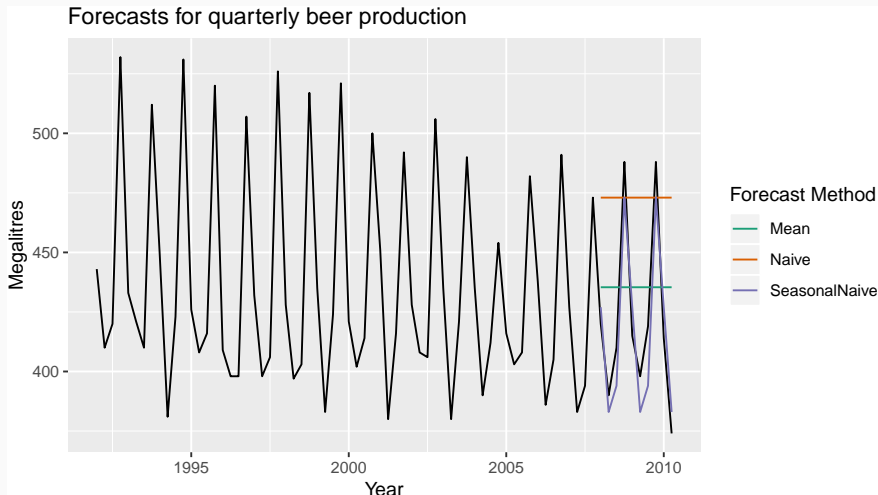
$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

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  $\{y_t\}$ .

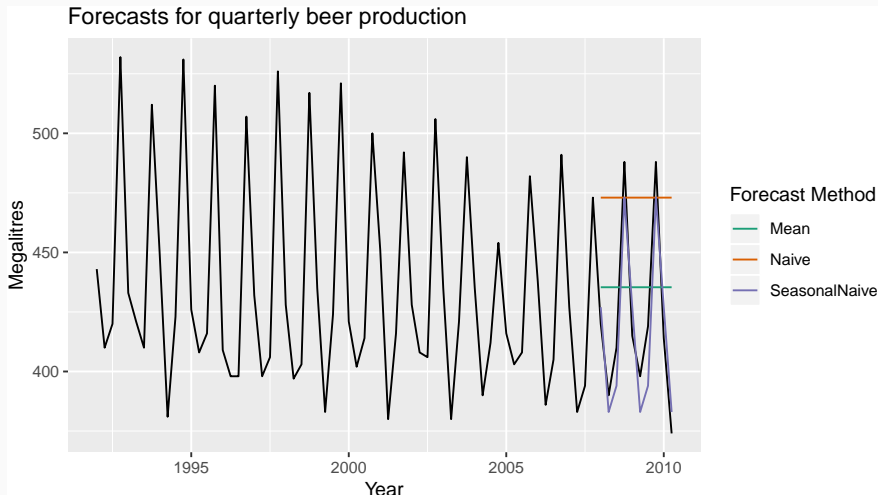
Proposed by Hyndman and Koehler (IJF, 2006).

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

```
beer2 <- window(ausbeer, start=1992, end=c(2007,4))  
beer3 <- window(ausbeer, start=2008)  
beerfit1 <- meanf(beer2, h=10)  
beerfit2 <- rwf(beer2, h=10)  
beerfit3 <- snaive(beer2, h=10)  
accuracy(beerfit1, beer3)  
accuracy(beerfit2, beer3)  
accuracy(beerfit3, beer3)
```

|                       | RMSE  | MAE   | MAPE  | MASE |
|-----------------------|-------|-------|-------|------|
| Mean method           | 38.45 | 34.83 | 8.28  | 2.44 |
| Naïve method          | 62.69 | 57.40 | 14.18 | 4.01 |
| Seasonal naïve method | 14.31 | 13.40 | 3.17  | 0.94 |

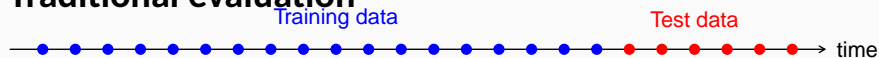
## Poll: true or false?

- 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.



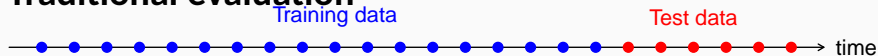
# Time series cross-validation

## Traditional evaluation

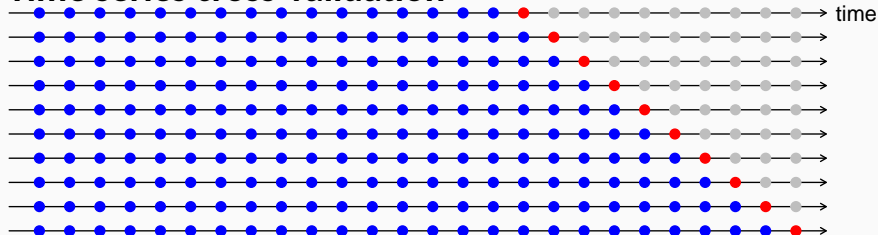


# Time series cross-validation

## Traditional evaluation

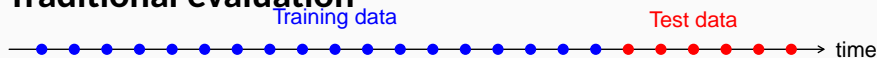


## Time series cross-validation

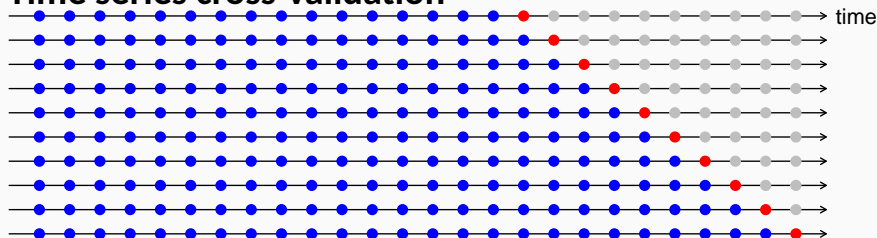


# Time series cross-validation

## Traditional evaluation



## Time series cross-validation



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

## tsCV function:

```
e <- tsCV(goog200, rwf, drift=TRUE, h=1)
sqrt(mean(e^2, na.rm=TRUE))
```

```
## [1] 6.233245
```

```
sqrt(mean(residuals(rwf(goog200, drift=TRUE))^2,
          na.rm=TRUE))
```

```
## [1] 6.168928
```

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

# Pipe function

Ugly code:

```
e <- tsCV(goog200, rwf, drift=TRUE, h=1)
sqrt(mean(e^2, na.rm=TRUE))
sqrt(mean(residuals(rwf(goog200, drift=TRUE))^2,
           na.rm=TRUE))
```

Better with a pipe:

```
goog200 %>%
  tsCV(forecastfunction=rwf, drift=TRUE, h=1) -> e
e^2 %>% mean(na.rm=TRUE) %>% sqrt
goog200 %>% rwf(drift=TRUE) %>% residuals -> res
res^2 %>% mean(na.rm=TRUE) %>% sqrt
```

# Outline

- 1 Some simple forecasting methods
- 2 Box-Cox transformations
- 3 Residual diagnostics
- 4 Evaluating forecast accuracy
- 5 Prediction intervals

# Prediction intervals

- A forecast  $\hat{y}_{T+h|T}$  is (usually) the mean of the conditional distribution  $y_{T+h} \mid y_1, \dots, y_T$ .
- 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

## Naive forecast with prediction interval:

```
res_sd <- sqrt(mean(res^2, na.rm=TRUE))  
c(tail(goog200,1)) + 1.96 * res_sd * c(-1,1)
```

```
## [1] 519.3103 543.6462
```

```
naive(goog200, level=95)
```

| ##     | Point Forecast | Lo 95    | Hi 95    |
|--------|----------------|----------|----------|
| ## 201 | 531.4783       | 519.3104 | 543.6461 |
| ## 202 | 531.4783       | 514.2703 | 548.6862 |
| ## 203 | 531.4783       | 510.4029 | 552.5536 |
| ## 204 | 531.4783       | 507.1425 | 555.8140 |
| ## 205 | 531.4783       | 504.2701 | 558.6865 |
| ## 206 | 531.4783       | 501.6732 | 561.2833 |
| ## 207 | 531.4783       | 499.2851 | 563.6714 |



# Prediction intervals

- Point forecasts are often useless without 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

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

**Mean forecasts:**  $\hat{\sigma}_h = \hat{\sigma} \sqrt{1 + 1/T}$

**Naïve forecasts:**  $\hat{\sigma}_h = \hat{\sigma} \sqrt{h}$

**Seasonal naïve forecasts**  $\hat{\sigma}_h = \hat{\sigma} \sqrt{k + 1}$

**Drift forecasts:**  $\hat{\sigma}_h = \hat{\sigma} \sqrt{h(1 + h/T)}$ .

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 value  $\hat{\sigma}$ .

# Prediction intervals

- Computed automatically using: `naive()`, `snaive()`, `rwf()`, `meanf()`, etc.
- Use `level` argument to control coverage.
- Check residual assumptions before believing them.
- Usually too narrow due to unaccounted uncertainty.