Applied Economic Forecasting

3. Evaluation of Basic Forecasting Models

Spring 2020

1. Some simple forecasting methods

2. Transformations & Adjustments

3. Residual diagnostics

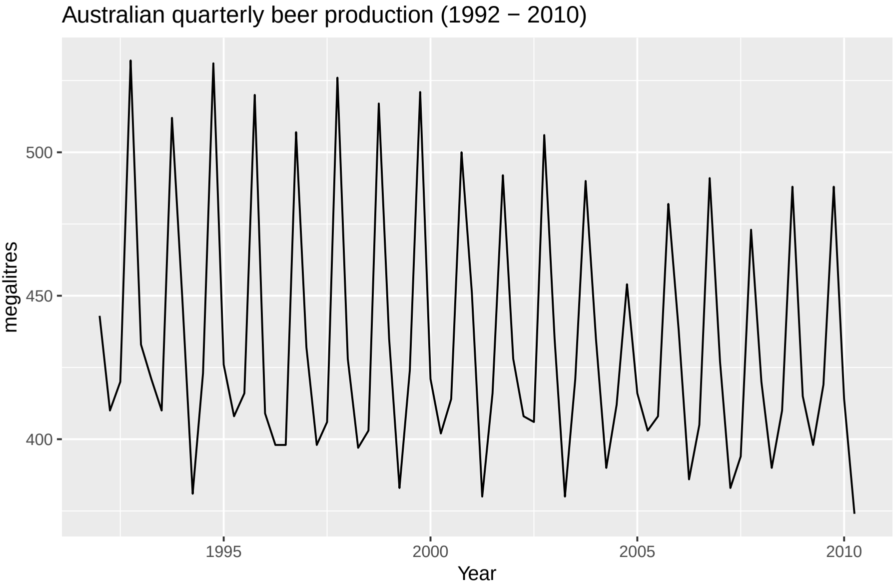
4. Evaluating forecast accuracy

5. Prediction intervals

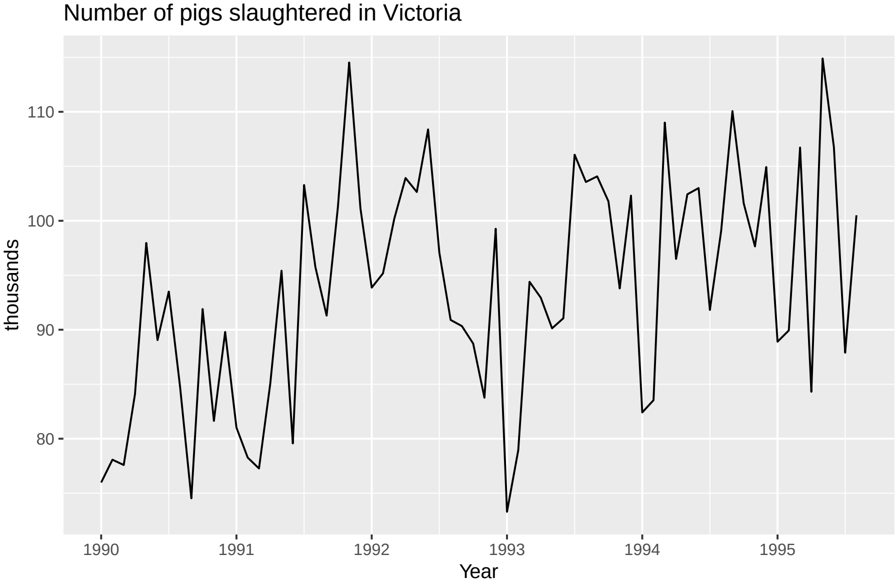
Section 1

Some simple forecasting methods

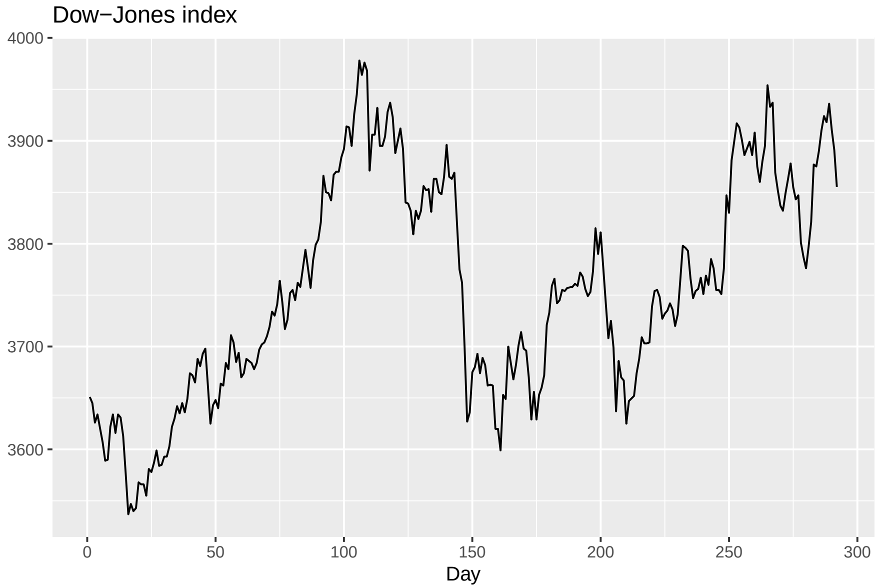
Some simple forecasting methods



Some simple forecasting methods



Some simple forecasting methods



Some simple forecasting methods

1. Average method

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

Some simple forecasting methods

1. Average method

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

2. 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

1. Average method

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

2. Naïve method

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

3. 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

4. Drift method

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

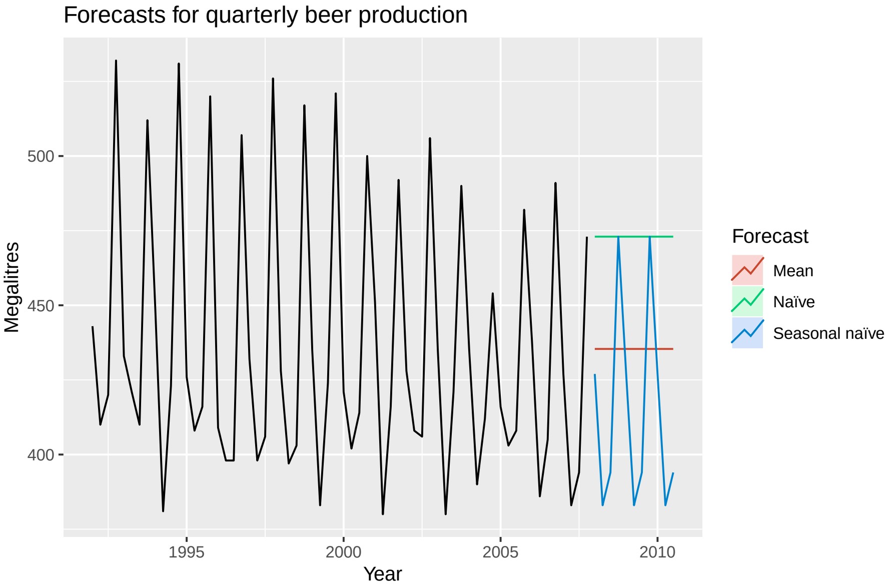
$$\begin{aligned}\hat{y}_{T+h|T} &= y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) \\ &= y_T + \frac{h}{T-1}(y_T - y_1).\end{aligned}$$

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

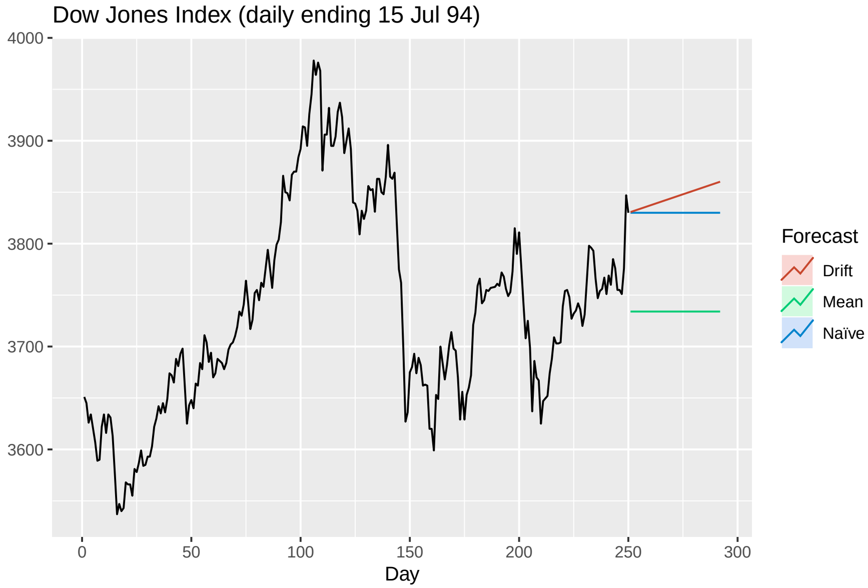
Some simple forecasting methods

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

Some simple forecasting methods



Some simple forecasting methods



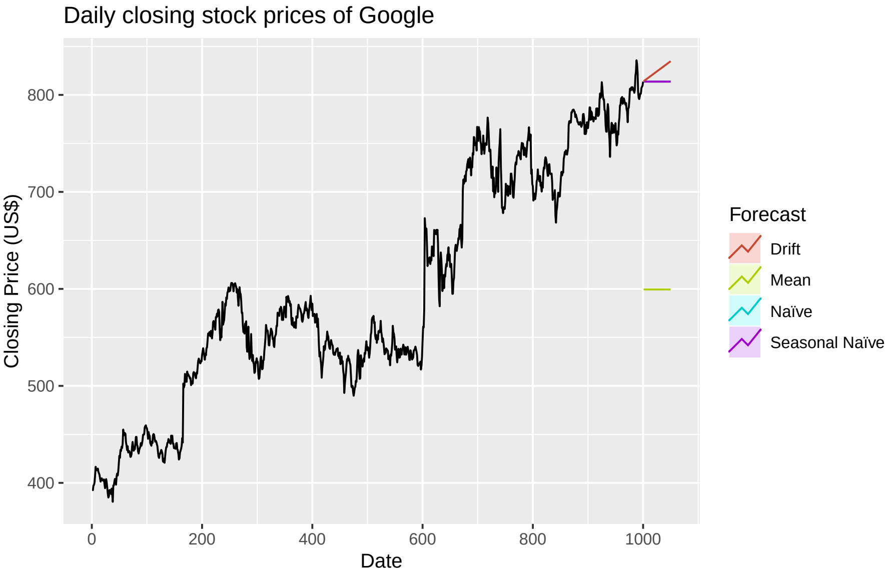
Some simple forecasting methods

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

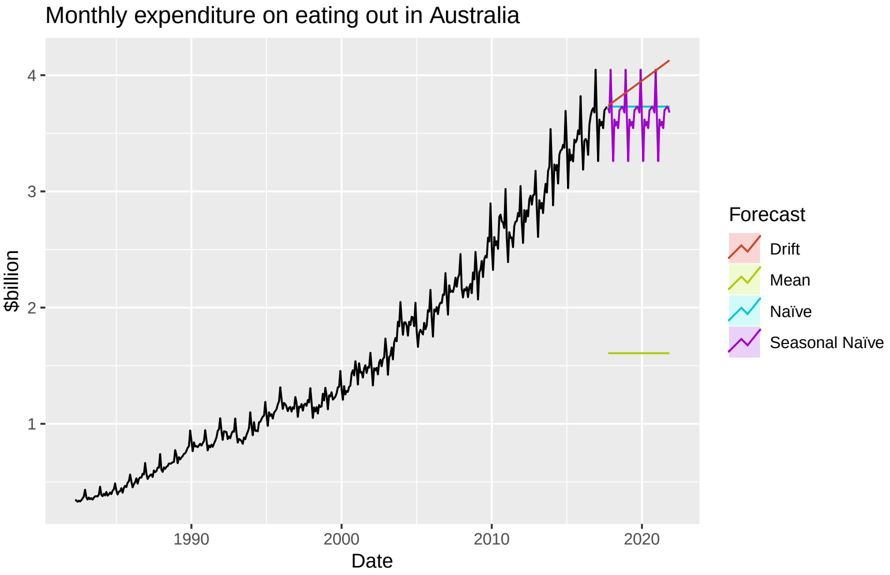
Your turn

- Use these four functions to produce forecasts for `goog` and `auscafe`. Set h=50.
- Plot the results using `autoplot()`.

Some simple forecasting methods



Some simple forecasting methods



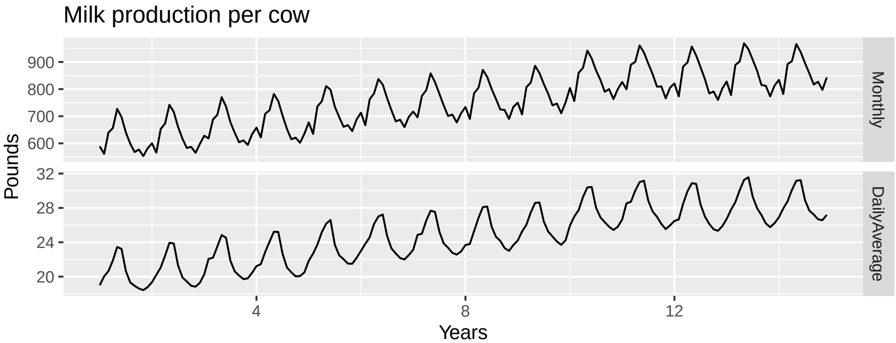
Section 2

Transformations & Adjustments

Calendar Adjustments

Some of the observed variation in seasonal data may be due to simple issues such as the number of calendar days. Usually much easier to remove the variation before fitting a forecasting model. The `monthdays()` function will compute the number of days in each month or quarter.

```
dframe <- cbind(Monthly = milk, DailyAverage = milk/monthdays(milk))
autoplot(dframe, facet = TRUE) + labs(x = "Years",
    y = "Pounds") + ggtitle("Milk production per cow")
```



Population adjustments

Data affected by population change (or size) are better expressed on a per-capita basis. For example, if you are studying the relationship between income and literacy across two regions, rather than compare the total GDP, it might be more constructive to look at income per person. With such a large population, China's total GDP will definitely be larger than a Luxembourg for example. After removing the effect of population size however, Luxembourg would be the leader (IMF 2019).

Inflation adjustments

- Data which are affected by the value of money are best adjusted before modelling. For example, the average cost of a new house will have increased over the last few decades due to inflation.
- A \$1 this year is not the same as a \$1 twenty years ago. For this reason, financial time series are usually adjusted so that all values are stated in dollar values from a particular year (real prices).
- Price indexes are often constructed by government agencies. For consumer goods, a common price index is the Consumer Price Index (or CPI).
- To convert current prices(y_t) to real prices (in 2010 dollars - x_{2020}) we can use the transformation:

$$x_{2020} = y_{2020} \times \frac{CPI_{2010}}{CPI_{2020}}$$

Box-Cox transformations

Variance stabilization

If the data show different variation at different levels of the series, then a transformation can be useful.
Denote original observations as $y_1, \ldots, y_n$ and transformed observations as $w_1, \ldots, w_n$.

Box-Cox transformations

Variance stabilization

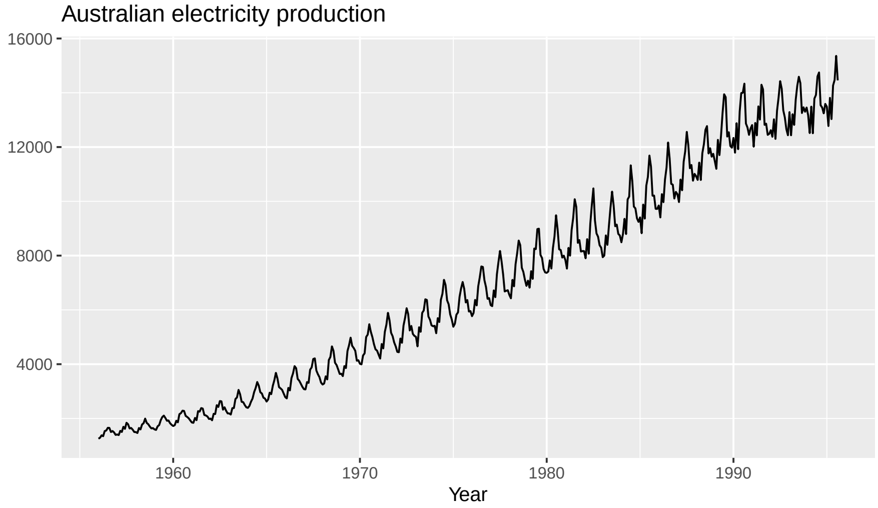
If the data show different variation at different levels of the series, then a transformation can be useful.
Denote original observations as $y_1, \ldots, y_n$ and transformed observations as $w_1, \ldots, w_n$.

Mathematical transformations for stabilizing variation

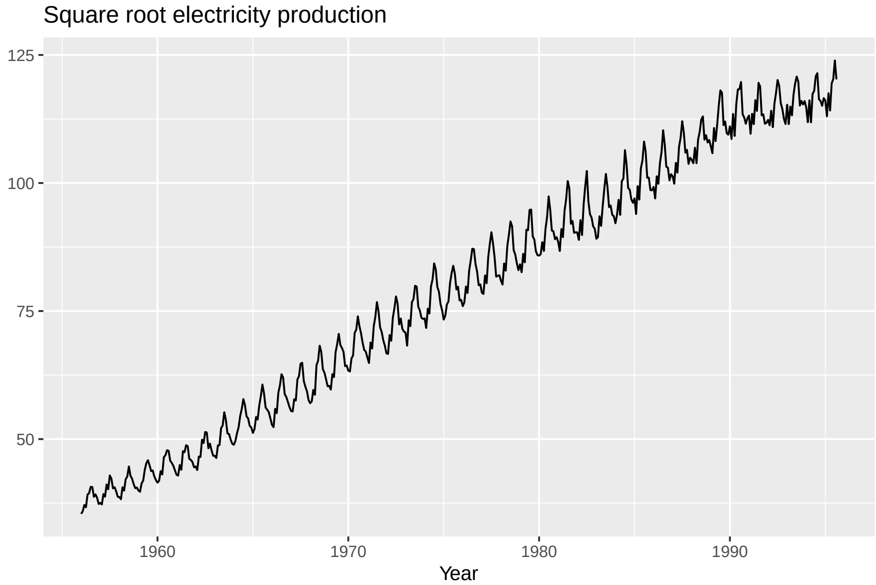
Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

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

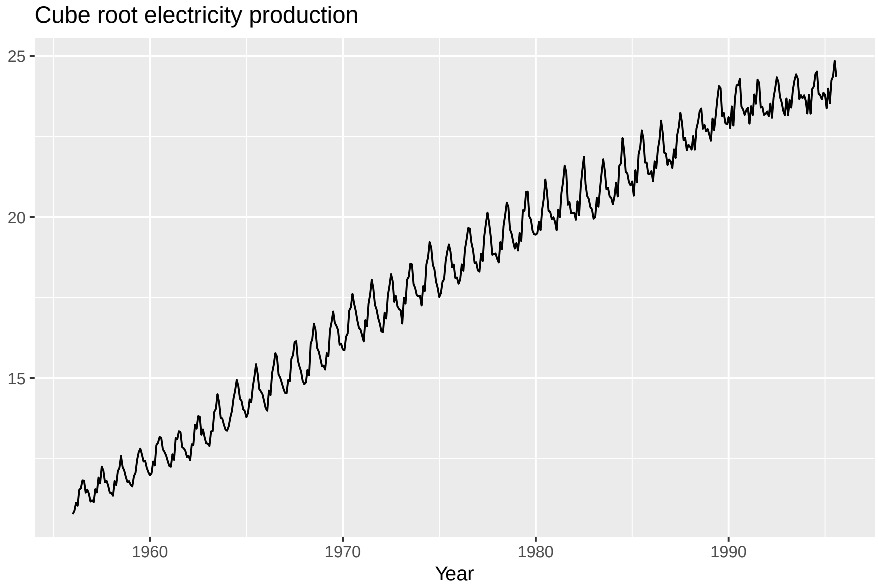
Variance stabilization



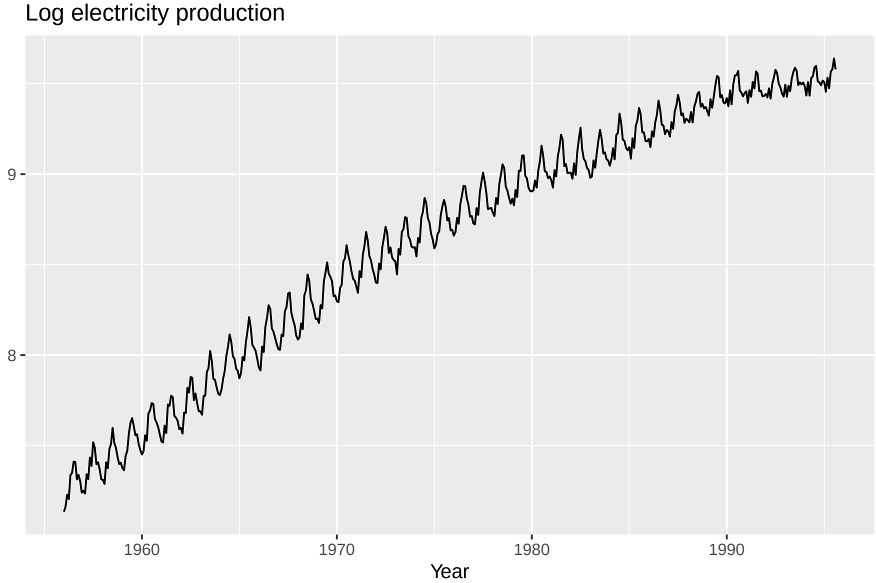
Variance stabilization



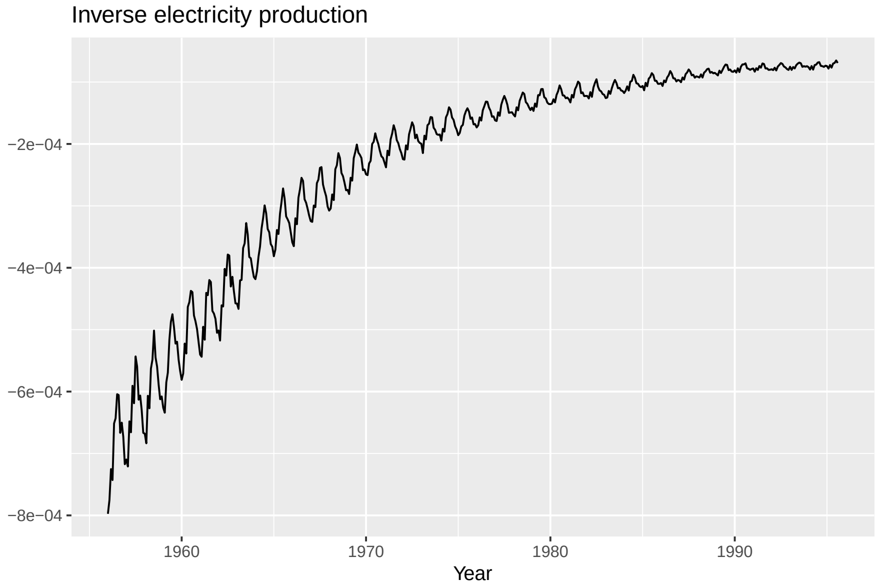
Variance stabilization



Variance stabilization



Variance stabilization



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)

Section 3

Residual diagnostics

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

Assumptions

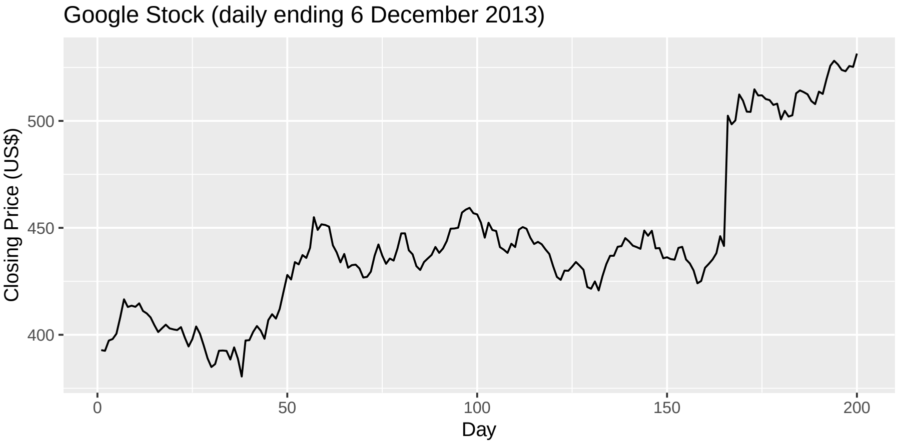
1. $\{e_t\}$ uncorrelated.
   - If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero.
   - If they don't, then forecasts are biased.

Useful properties (for prediction intervals)

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

Example: Google stock price

```r
autoplot(goog200) + labs(x = "Day", y = "Closing Price (US$)") +
  ggtitle("Google Stock (daily ending 6 December 2013)")
```



Example: Google stock price

Naïve forecast:

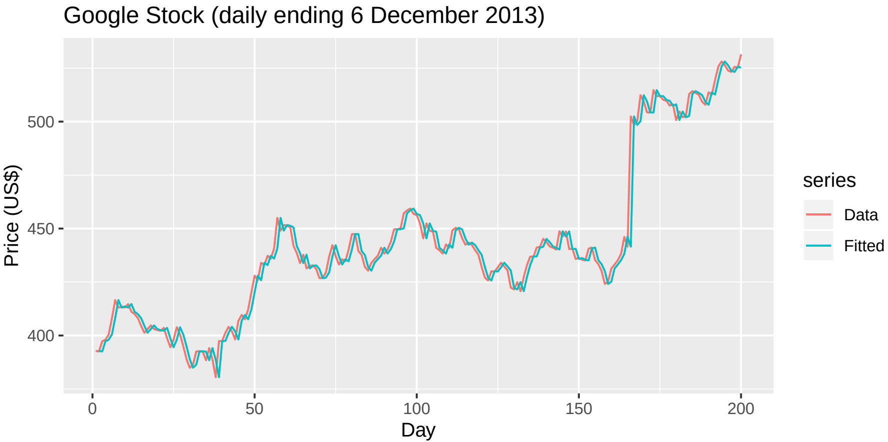
$$\hat{y}_{t|t-1} = y_{t-1}$$

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

Note: e_t are one-step-forecast residuals

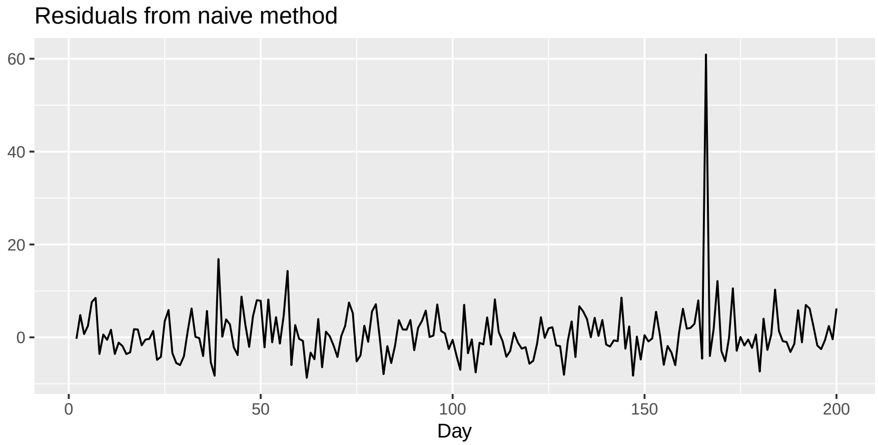
Example: Google stock price

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



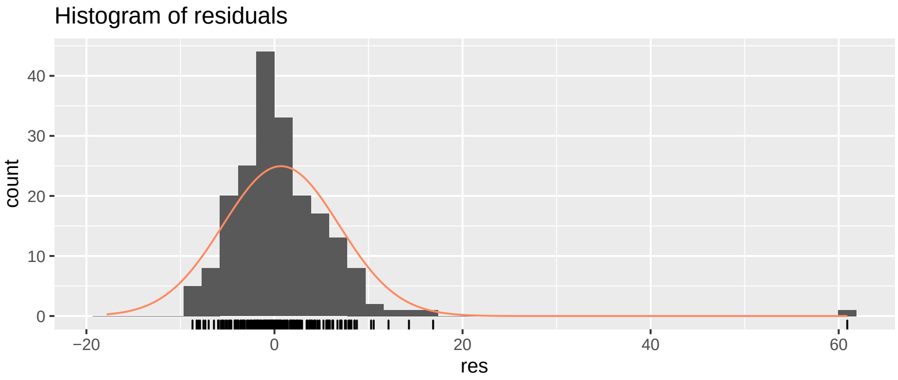
Example: Google stock price

```r
res <- residuals(naive(goog200))
autoplot(res) + labs(x = "Day", y = "") +
    ggtitle("Residuals from naive method")
```



Example: Google stock price

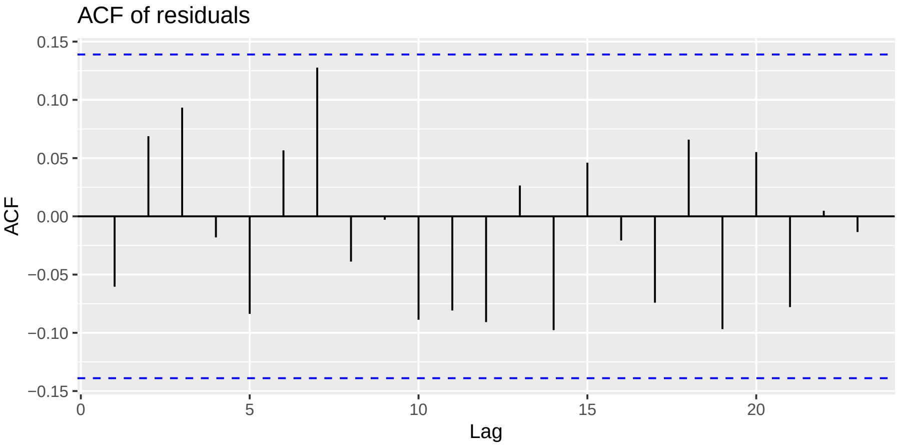
```r
gghistogram(res, add.normal = TRUE) +
    ggtitle("Histogram of residuals")
```



Forecast with this model will be good but the non-normality of the residuals will prove problematic when calculating the prediction intervals.

Example: Google stock price

```r
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 for autocorrelation

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 for autocorrelation

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 for autocorrelation

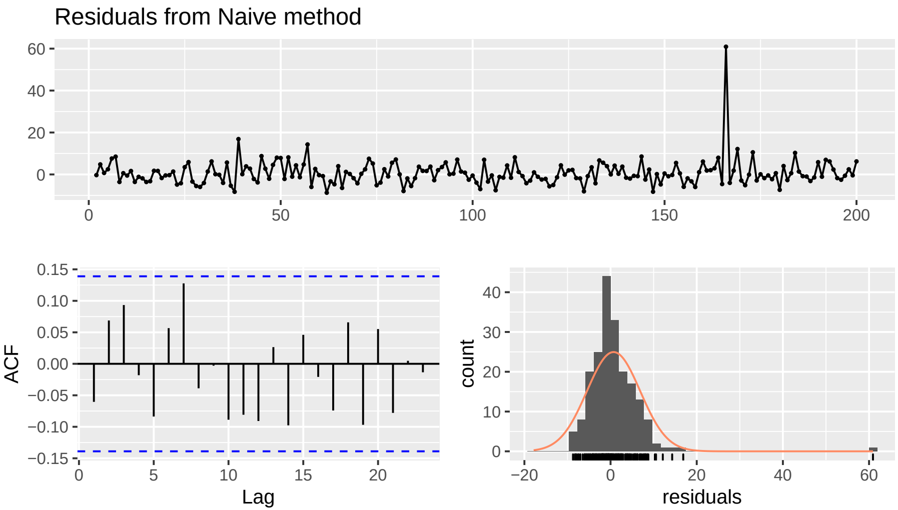
- If data are WN, Q^* has χ^2 distribution with $(h - K)$ degrees of freedom where $K =$ no. parameters in model.
- When applied to raw data, set $K = 0$.
- For the Google example:

```
# 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), test = FALSE)
```



checkresiduals function

```r
checkresiduals(naive(goog200), plot = FALSE)
```

```
##
##  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)
fbeer <- snaive(beer)
autoplot(fbeer)
```

Test if the residuals are white noise.

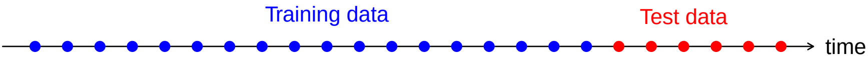
```
checkresiduals(fbeer)
```

What do you conclude?

Section 4

Evaluating forecast accuracy

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

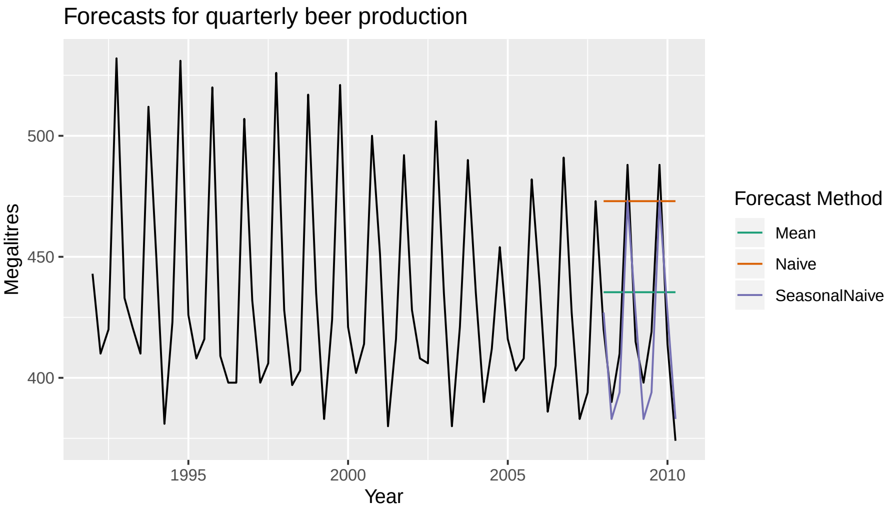
Forecast "error": the difference between an observed value and its forecast.

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

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

$$\begin{aligned} y_{T+h} = &\quad (T+h)\text{th observation, } h = 1, \dots, H \\ \hat{y}_{T+h|T} = &\quad \text{its forecast based on data up to time } T. \\ e_{T+h} = &\quad y_{T+h} - \hat{y}_{T+h|T} \end{aligned}$$

$$\begin{aligned} \text{MAE} &= \text{mean}(|e_{T+h}|) \\ \text{MSE} &= \text{mean}(e_{T+h}^2) \\ \text{RMSE} &= \sqrt{\text{mean}(e_{T+h}^2)} \\ \text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|) \end{aligned}$$

- 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.

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 $\{y_t\}$.

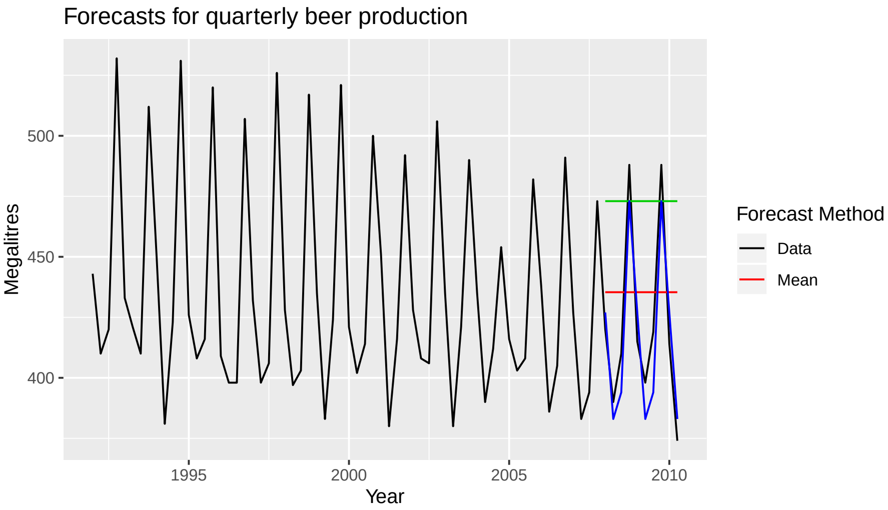
Proposed by Hyndman and Koehler (IJF, 2006).

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

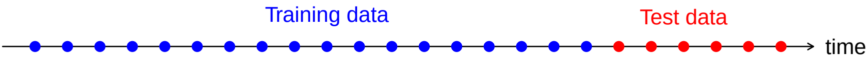


Measures of forecast accuracy

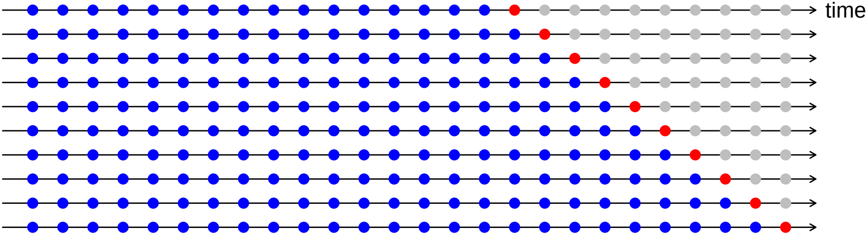
```r
beer2 <- window(ausbeer, start=1992, end=c(2007,4))
beer3 <- window(ausbeer, start=2008)
beerfit1 <- meanf(beer2, h=10)
beerfit2 <- rwf(beer2, h=10)
beerfit3 <- snaive(beer2, h=10)
accuracy(beerfit1, beer3)
accuracy(beerfit2, beer3)
accuracy(beerfit3, beer3)
```

	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

Traditional evaluation



Time series cross-validation



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))
```

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

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

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

Better with a pipe:

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

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

```
res <- goog200 %>% rwf(drift = TRUE) %>% residuals
res^2 %>% mean(na.rm = TRUE) %>% sqrt
```

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

Section 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] 519.3872 543.5694
```

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

```
##     Point Forecast    Lo 95    Hi 95
## 201       531.4783 519.3105 543.6460
## 202       531.4783 514.2705 548.6861
## 203       531.4783 510.4031 552.5534
## 204       531.4783 507.1428 555.8138
## 205       531.4783 504.2704 558.6862
## 206       531.4783 501.6735 561.2830
## 207       531.4783 499.2854 563.6711
## 208       531.4783 497.0627 565.8939
## 209       531.4783 494.9750 567.9815
## 210       531.4783 493.0005 569.9561
```

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, 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.