MONASH University

MONASH BUSINESS SCHOOL

ETC3550 Applied forecasting for business and economics

Ch3. The forecasters' toolbox

OTexts.org/fpp3/

Outline

Outline

A tidy forecasting workflow

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

1. Data preparation
2. Visualise
3. Specify a model
4. Estimating the model
5. Evaluate model
6. Producing forecasts

A tidy forecasting workflow

Outline

Some simple forecasting methods

How would you forecast these time

Some simple forecasting methods

How would you forecast these time

Some simple forecasting methods

How would you forecast these time

Some simple forecasting methods

Average method

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

Some simple forecasting methods

Average method

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

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: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$

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

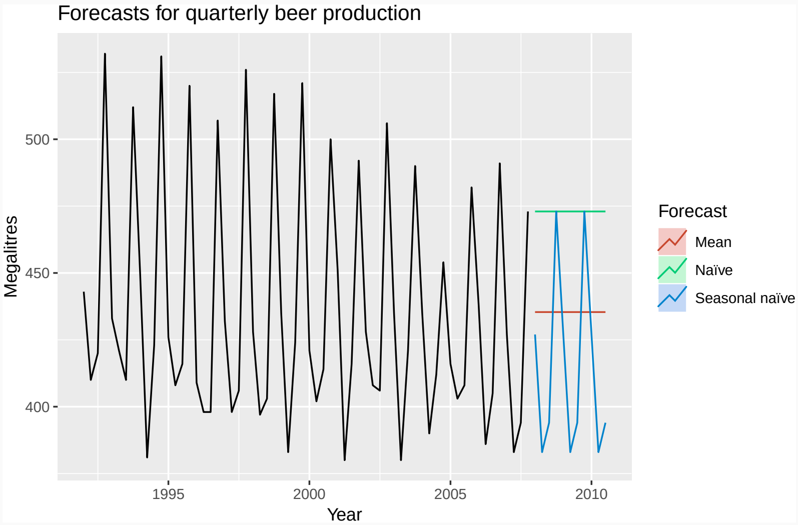
Drift method

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

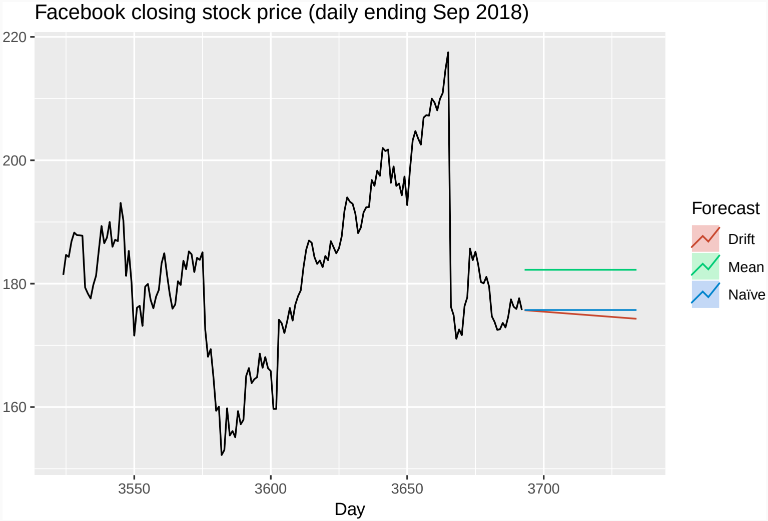
$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})$$
$$= y_T + \frac{h}{T-1}(y_T - y_1).$$

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

Some simple forecasting methods



Some simple forecasting methods



Some simple forecasting methods

Code for previous graph

```
fb_stock <- gafa_stock %>%
  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")))

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"))
```

Some simple forecasting methods

- Mean: `MEAN(y)`
- Naïve: `NAIVE(y)`
- Seasonal naïve: `SNAIVE(y)`
- Drift: `RW(y ~ drift())`

Some simple forecasting methods

- Mean: `MEAN(y)`
- Naïve: `NAIVE(y)`
- Seasonal naïve: `SNAIVE(y)`
- Drift: `RW(y ~ drift())`

Your turn

- Use these four functions to produce forecasts for Facebook closing price (`gafa_stock`) and Australian takeaway food turnover (`aus_retail`).
- Plot the results using `autoplot()`.

Outline

Specifying a model

```
model_fn(t(LHS) ~ specials, extras)
```

Outline

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	$w_t = \sqrt{y_t}$	$\downarrow$
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.

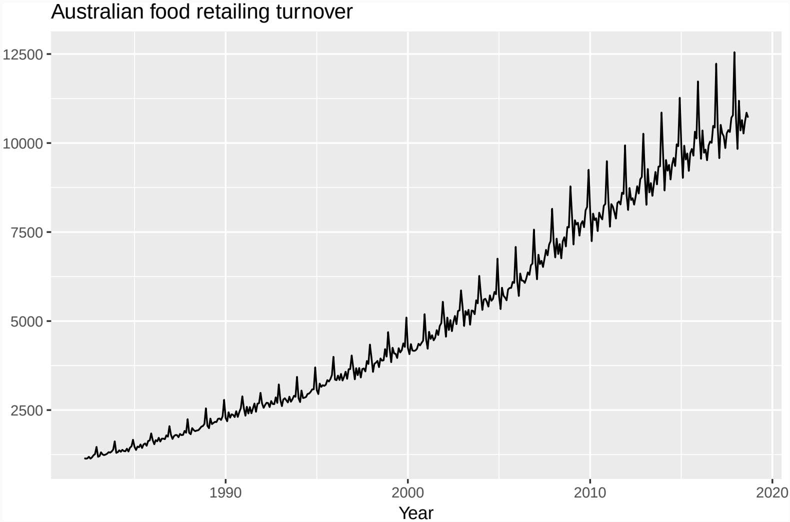
Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Mathematical transformations for stabilizing variation

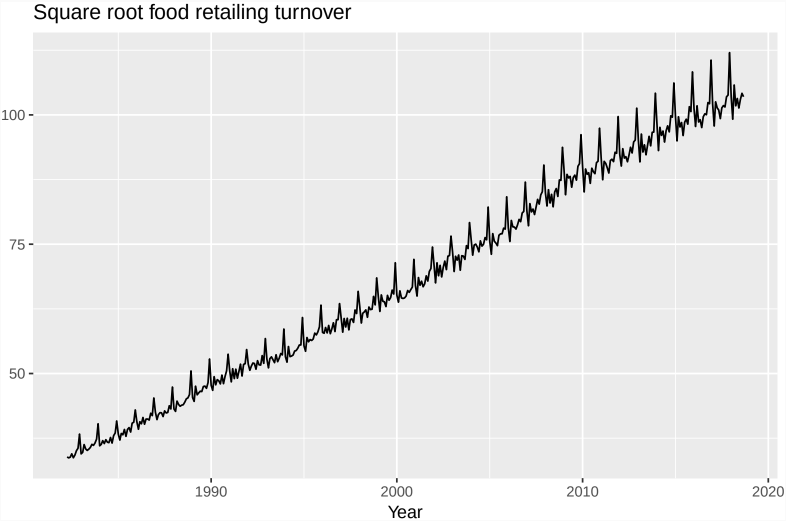
Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

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

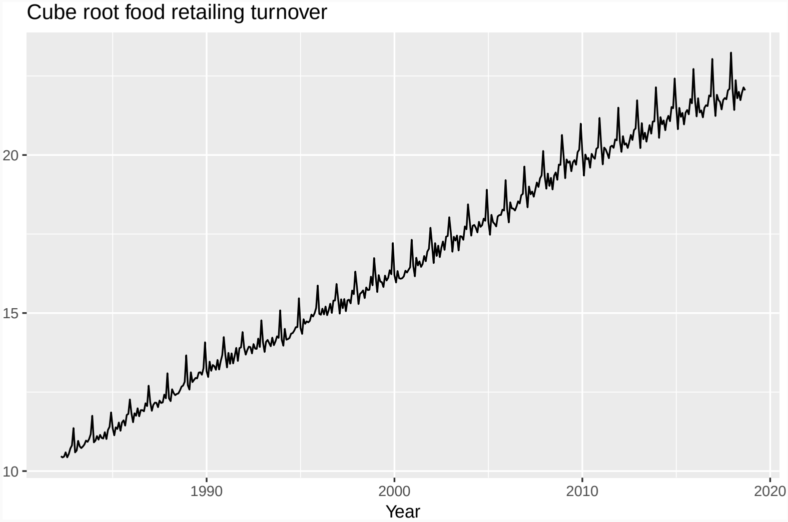
Variance stabilization



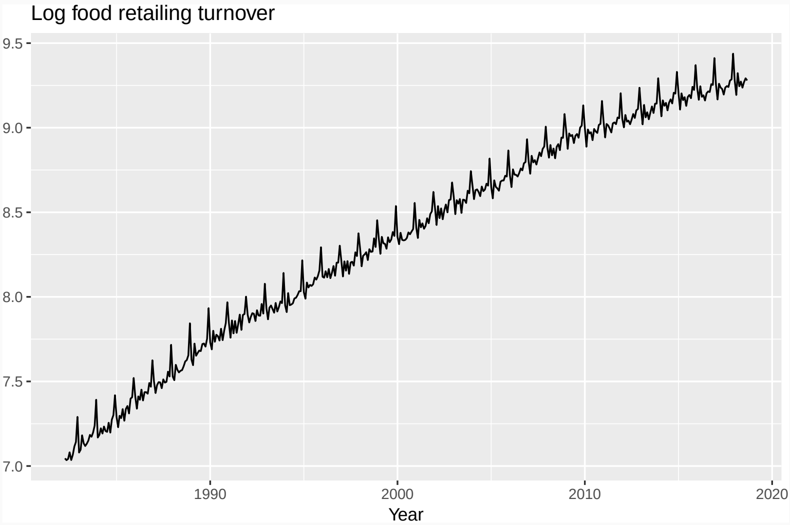
Variance stabilization



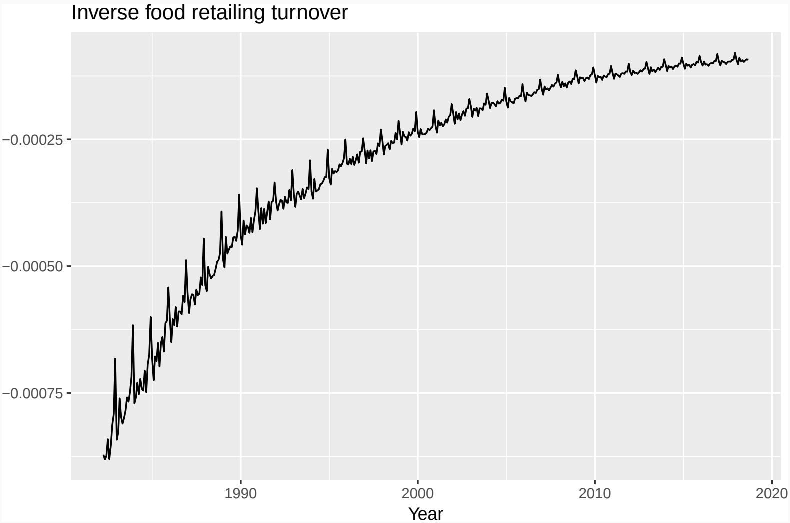
Variance stabilization



Variance stabilization



Variance stabilization



Box-Cox transformations

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

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

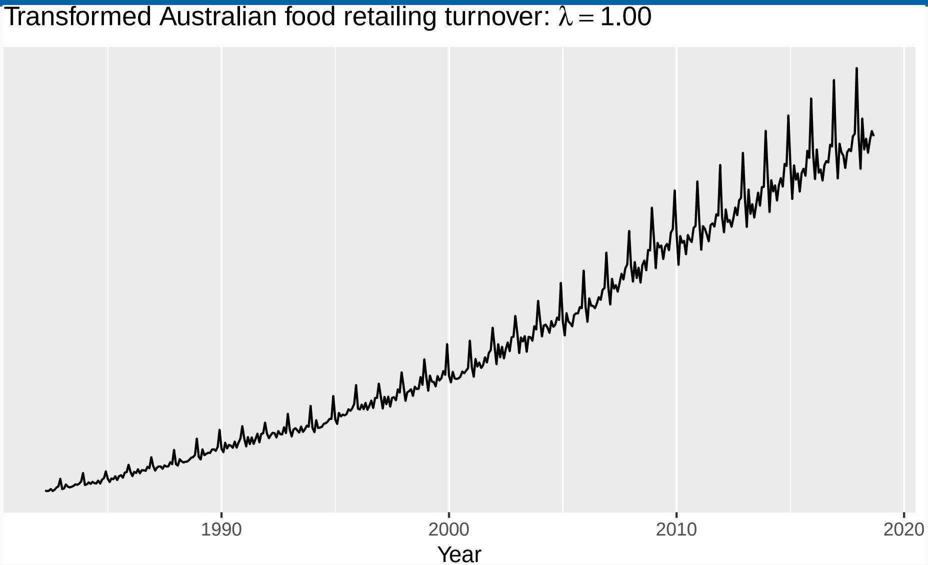
Box-Cox transformations

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

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

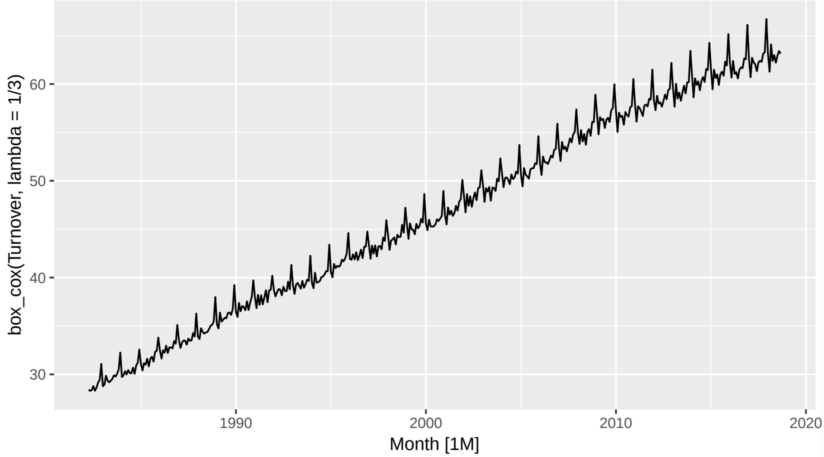
- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations



Box-Cox transformations

```r
food %>% autoplot(box_cox(Turnover,lambda=1/3))
```



Box-Cox transformations

- y_t^λ for λ 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 λ are easier to explain.
- Results are relatively insensitive to λ.
- 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 = guerrer
```

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

Box-Cox transformations

```
food %>% features(Turnover, features = guerrer
```

```
## # 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 λ can give extremely large

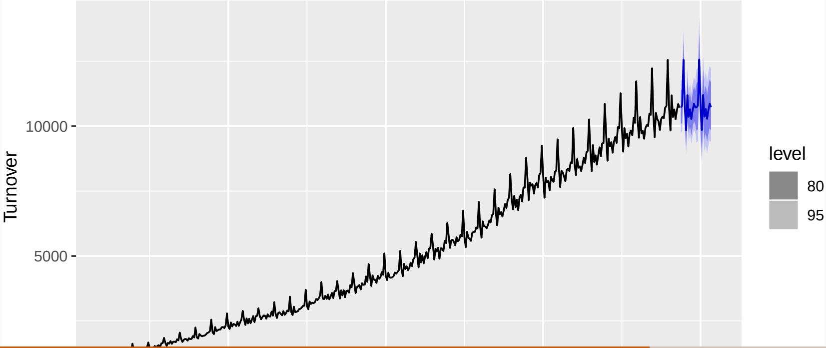
Back-transformation

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

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

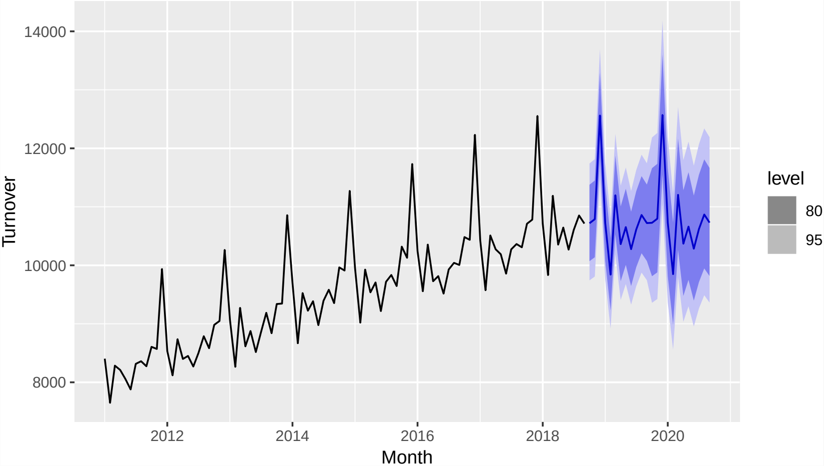
Back-transformation

```
fc <- food %>%
  model(SNAIVE(box_cox(Turnover, lambda=1/3))) %>%
  forecast()
fc %>% autoplot(food)
```



Back-transformation

```
fc %>% autoplot(filter(food, year(Month) > 201
```



Your turn

Find a Box-Cox 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 μ and variance σ^2.

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

Taylor series expansion about μ:

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

Bias adjustment

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

Back-transformed means

Let X be have mean μ and variance σ^2.

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

Taylor series expansion about μ:

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

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

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:

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

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

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

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

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
  autolayer(fc, series="Bias adjusted", level = NULL) +
  guides(colour=guide_legend(title="Forecast"))
```

Outline

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:

```
fit <- fb_stock %>% model(NAIVE(Close))
res_sd <- sd(augment(fit)$.resid, na.rm = TRUE)
last(fb_stock$Close) + 1.96 * res_sd * c(-1,1)
```

```
## [1] 166.6929 184.7670
```

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

```
## # A fable: 1 x 6 [?]
## # Key:     Symbol, .model [1]
##   Symbol .model trading_day Close
##   <fct>  <chr>        <int> <dbl>
```

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 from the forecast distribution.
- Use `level` argument to control coverage.
- Check residual assumptions before believing them.
- Usually too narrow due to unaccounted uncertainty.