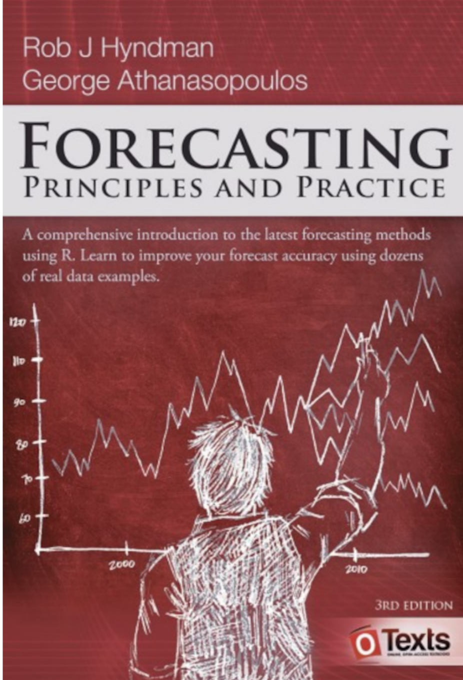
Ch3. Time series decomposition

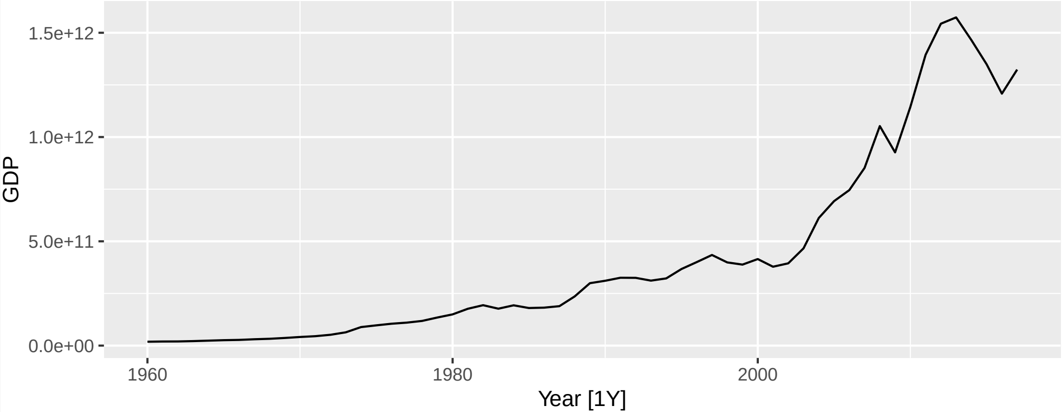
3.1 Transformations and adjustments

OTexts.org/fpp3/



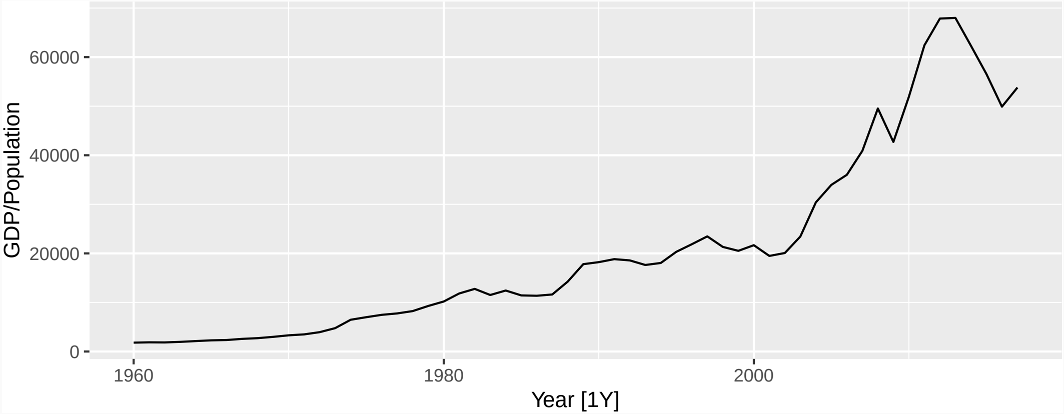
Per capita adjustments

```
global_economy %>%
  filter(Country == "Australia") %>%
  autoplot(GDP)
```



Per capita adjustments

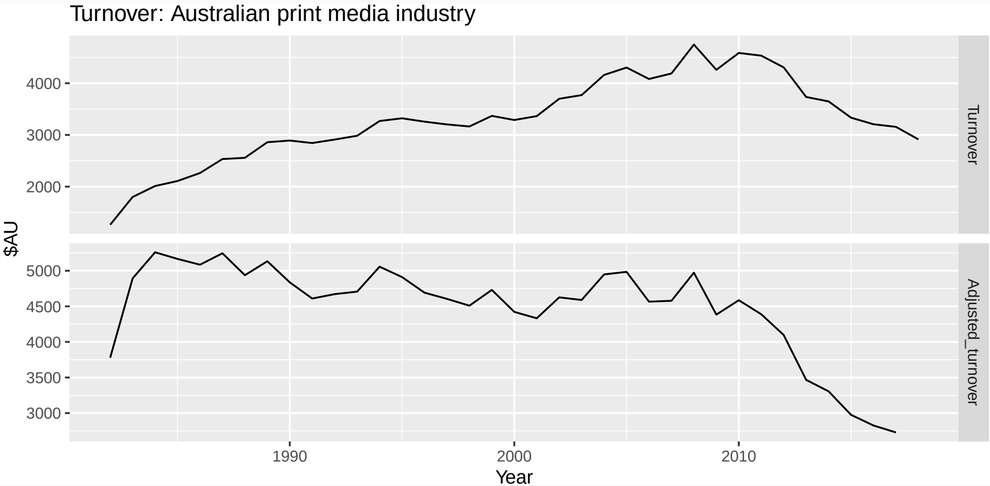
```
global_economy %>%
  filter(Country == "Australia") %>%
  autoplot(GDP / Population)
```



Inflation adjustments

```r
print_retail <- aus_retail %>%
  filter(Industry == "Newspaper and book retailing") %>%
  group_by(Industry) %>%
  index_by(Year = year(Month)) %>%
  summarise(Turnover = sum(Turnover))
aus_economy <- global_economy %>%
  filter(Code == "AUS")
print_retail %>%
  left_join(aus_economy, by = "Year") %>%
  mutate(Adjusted_turnover = Turnover / CPI * 100) %>%
  pivot_longer(c(Turnover, Adjusted_turnover), values_to = "Turnover") %>%
  mutate(name = factor(name, levels=c("Turnover","Adjusted_turnover"))) %>%
  ggplot(aes(x = Year, y = Turnover)) +
  geom_line() +
  facet_grid(name ~ ., scales = "free_y") +
  labs(title = "Turnover: Australian print media industry", y = "$AU")
```

Inflation adjustments



Mathematical transformations

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

Mathematical transformations

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

Denote original observations as $y_1, \dots, y_T$ and transformed observations as $w_1, \dots, w_T$.

Mathematical transformations

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

Denote original observations as $y_1, \dots, y_T$ and transformed observations as $w_1, \dots, w_T$.

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

Mathematical transformations

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

Denote original observations as $y_1, \dots, y_T$ and transformed observations as $w_1, \dots, w_T$.

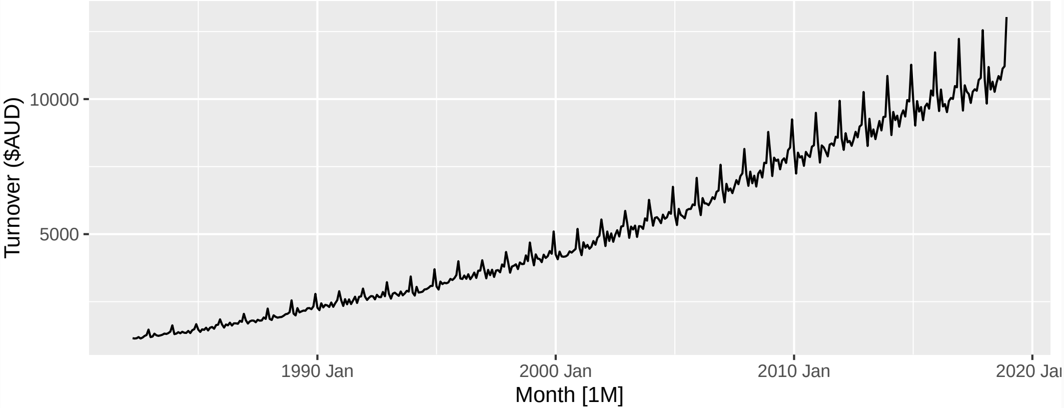
Mathematical transformations for stabilizing variation

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**.

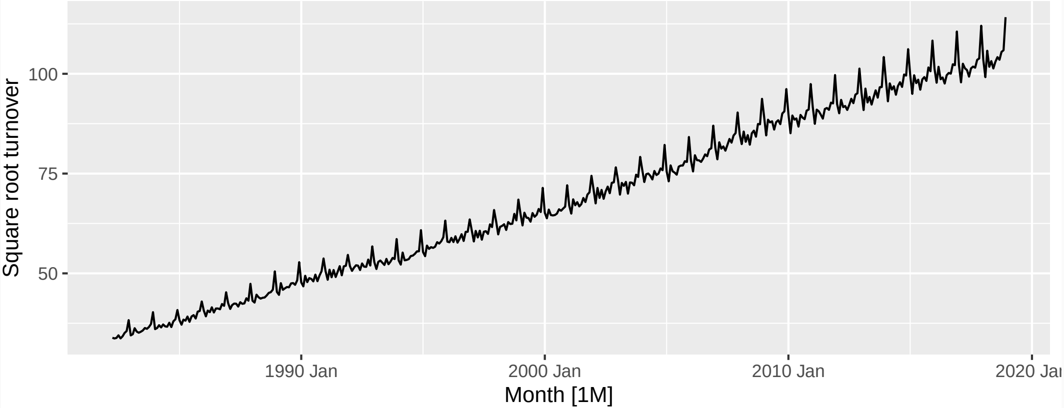
Mathematical transformations

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



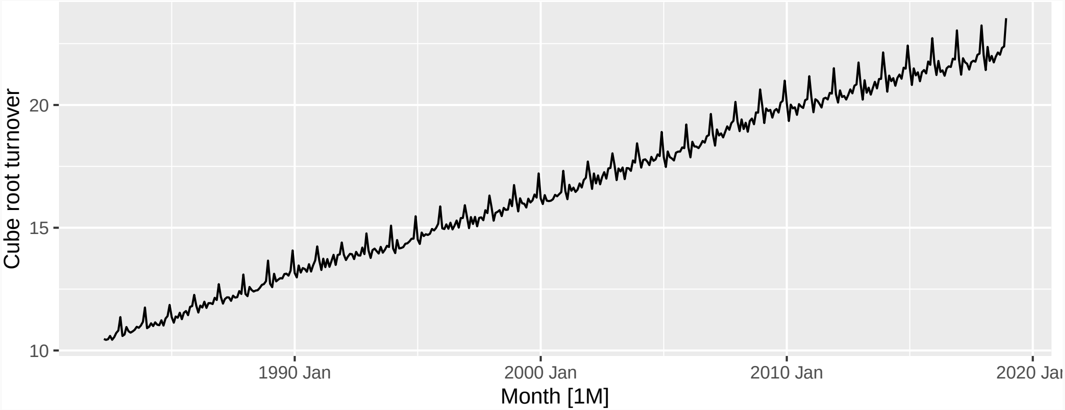
Mathematical transformations

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



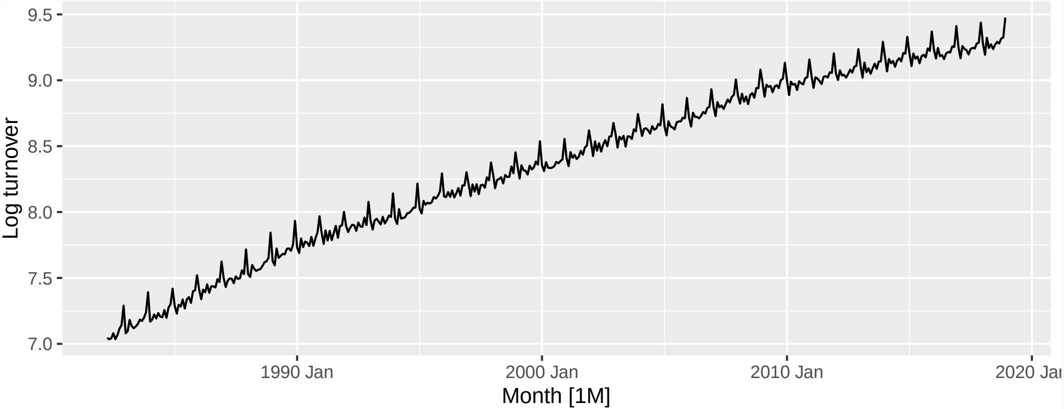
Mathematical transformations

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



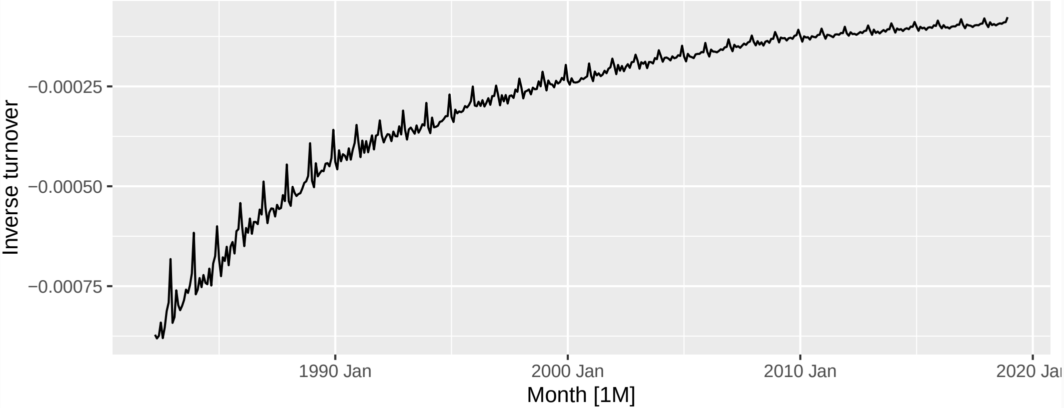
Mathematical transformations

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



Mathematical transformations

```
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**:

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

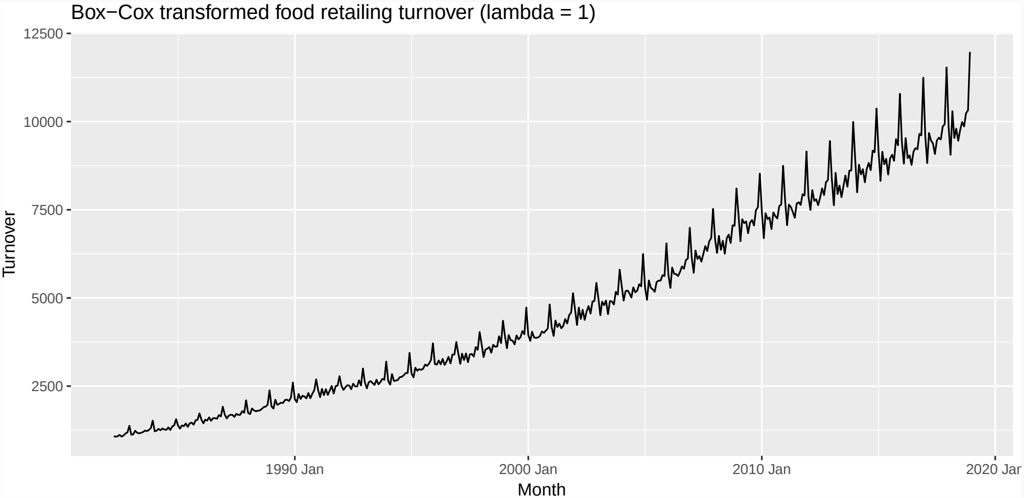
Box-Cox transformations

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

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

- Actually the Bickel-Doksum transformation (allowing for $y_t < 0$)
- $\lambda = 1$: (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 %>%
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1          0.0895
```

Box-Cox transformations

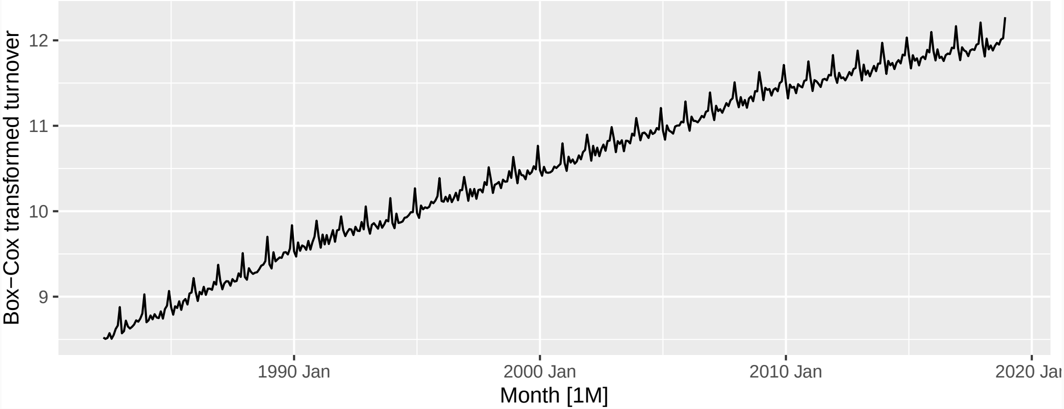
```
food %>%
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1          0.0895
```

- 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 prediction intervals.

Box-Cox transformations

```r
food %>% autoplot(box_cox(Turnover, 0.0524)) +
  labs(y = "Box-Cox transformed turnover")
```



Transformations

- Often no transformation needed.
- Simple transformations are easier to explain and work well enough.
- Transformations can have very large effect on PI.
- If some data are zero or negative, then use $\lambda > 0$.
- `log1p()` can also be useful for data with zeros.
- Choosing logs is a simple way to force forecasts to be positive
- Transformations must be reversed to obtain forecasts on the original scale. (Handled automatically by `fable`.)