Applied Economic Forecasting

5. Time Series Decomposition

Section 1

Simple Data Transformations

Adjusting for Population

Data affected by population change (or size) are better expressed on a per-capita basis.

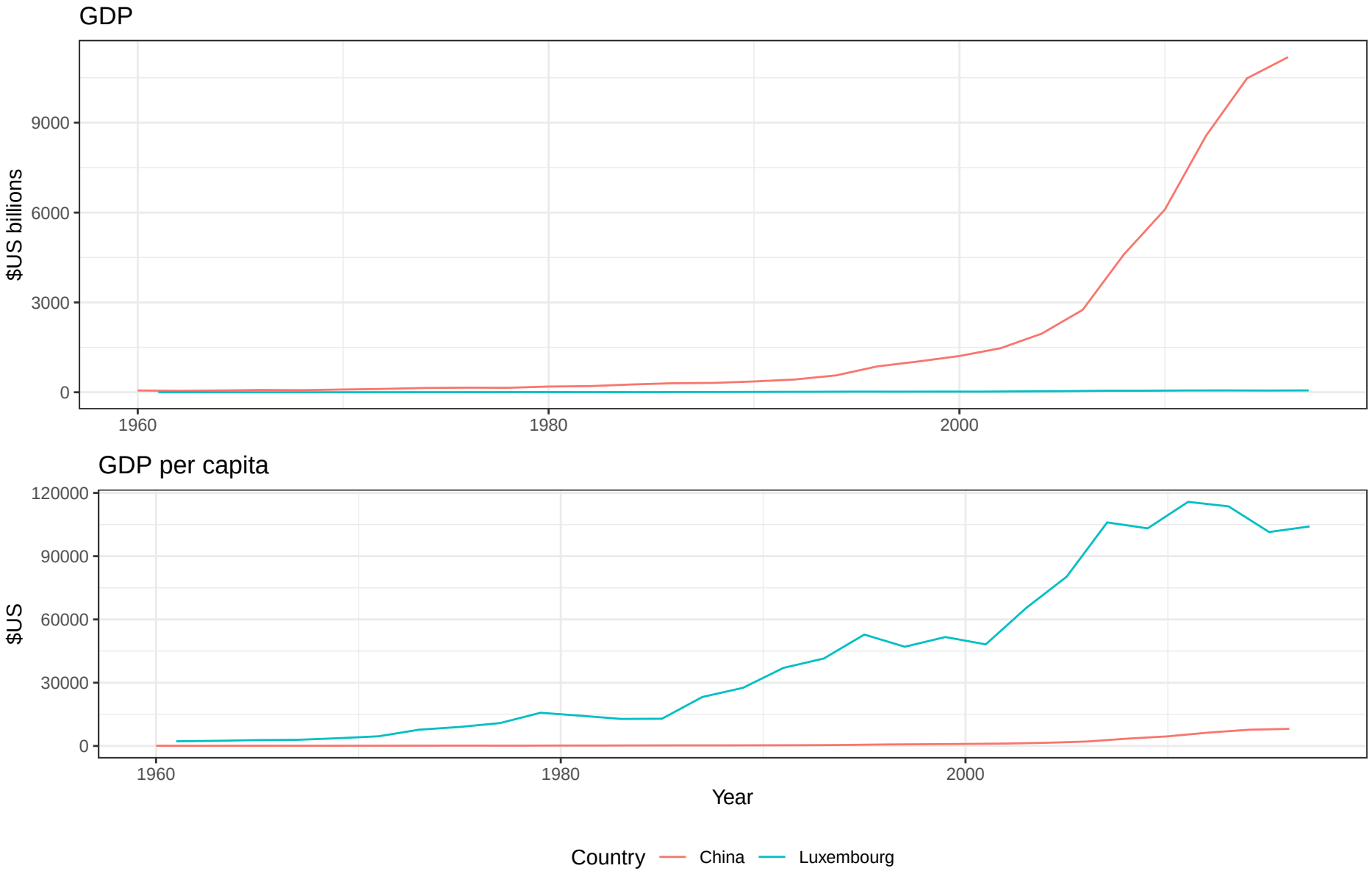
Example of China vs Luxembourg

Suppose you are studying the relationship between income and literacy across two regions, rather than use the total GDP, it might be more constructive to look at *income per person*.

With such a large population, China's total GDP is much larger than Luxembourg's for example. After removing the effect of population size however, Luxembourg (650,000 persons) emerges as the leader (even ahead of the U.S.). (IMF 2023).

Adjusting for Population

```
p1 <- global_economy %>%
  filter(Country == c("China","Luxembourg")) %>%
  ggplot(aes(x = Year)) +
  geom_line(aes(y = GDP/1e9, col = Country)) +
  labs(title= "GDP", y = "$US billions", x = NULL) +
  theme(legend.pos = "none")

p2 <- global_economy %>%
  filter(Country == c("China","Luxembourg")) %>%
  ggplot(aes(x = Year)) +
  geom_line(aes(y = GDP/Population, col = Country)) +
  labs(title= "GDP per capita", y = "$US") +
  theme(legend.pos = "bottom")

gridExtra::grid.arrange(p1,p2, nrow = 2)
```

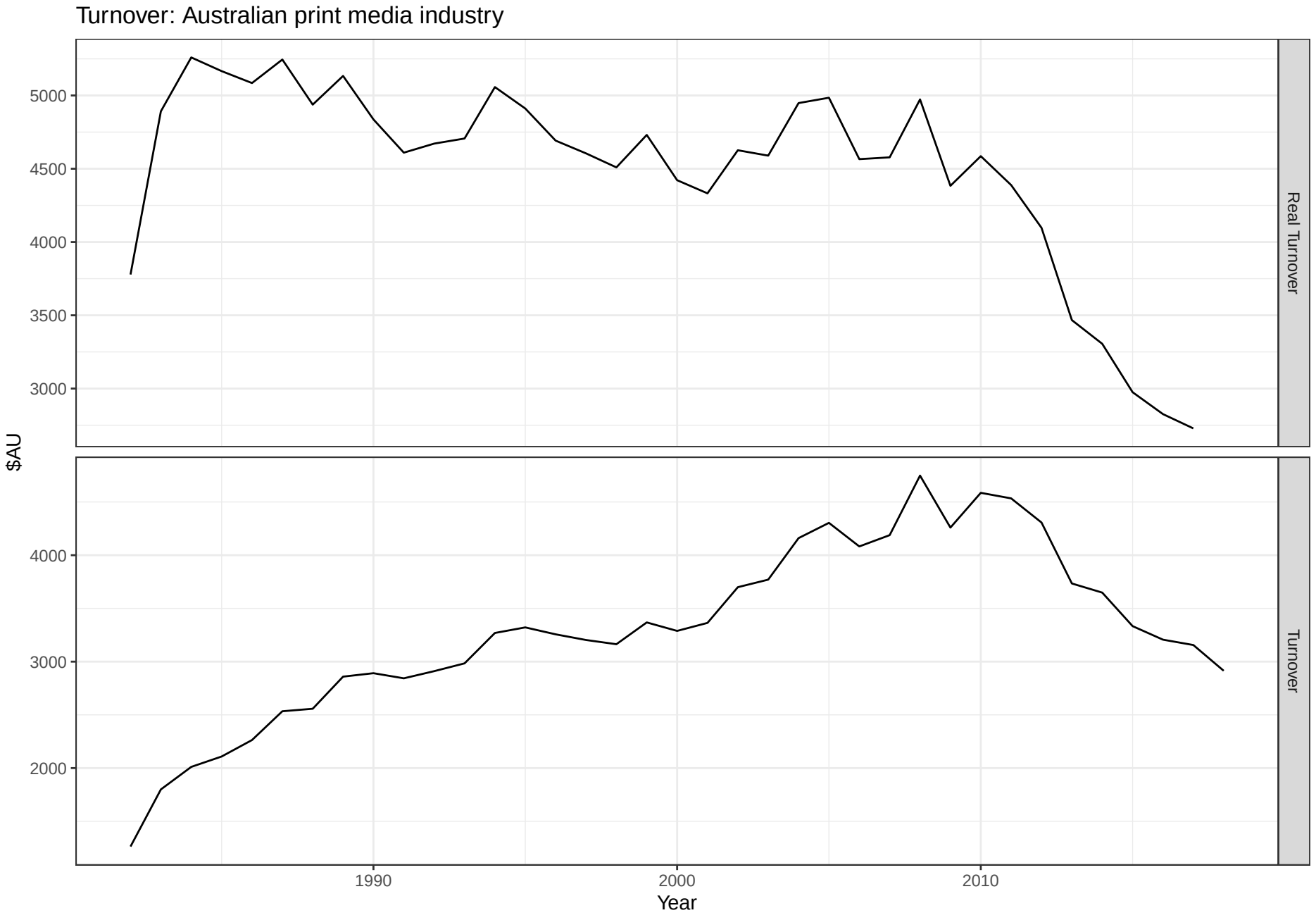

GDP
$US billions
9000
6000
3000
0
1960
1980
2000
GDP per capita
$US
120000
90000
60000
30000
0
1960
1980
2000
Year
Country
China
Luxembourg

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.
- \$1 this year is not the same as \$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 (CPI).
- To convert *current* prices(y_{2023}) to real (or constant) prices (in 2000 dollars - x_{2000}) we can use the transformation:

$$x_{2000} = \frac{y_{2023}}{CPI_{2023}} \times CPI_{2000}$$

```r
# Convert Newspaper sales to Annual
print_retail <- aus_retail  %>%
  filter(Industry == "Newspaper and book retailing")  %>%
  index_by(Year = year(Month))  %>%
  summarise(Turnover = sum(Turnover))

# Get CPI values for Australia
aus_economy <- global_economy  %>%
  filter(Code == "AUS")

# Join both datasets by year index
print_retail %>%
  left_join(aus_economy, by = "Year") %>%
  mutate(`Real Turnover` = Turnover / CPI * 100)  %>%
   pivot_longer(c(Turnover, `Real Turnover`),
                values_to = "Turnover", names_to = "Series") %>%
  ggplot(aes(x = Year, y = Turnover)) +
   geom_line() +
  facet_grid( Series ~ ., scales = "free_y") +
  labs(title = "Turnover: Australian print media industry",
       y = "$AU")
```


Turnover: Australian print media industry
$AU
5000
4500
4000
3500
3000
Real Turnover
4000
3000
2000
Turnover
1990
2000
2010
Year

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 $w_1, \ldots, w_n$.

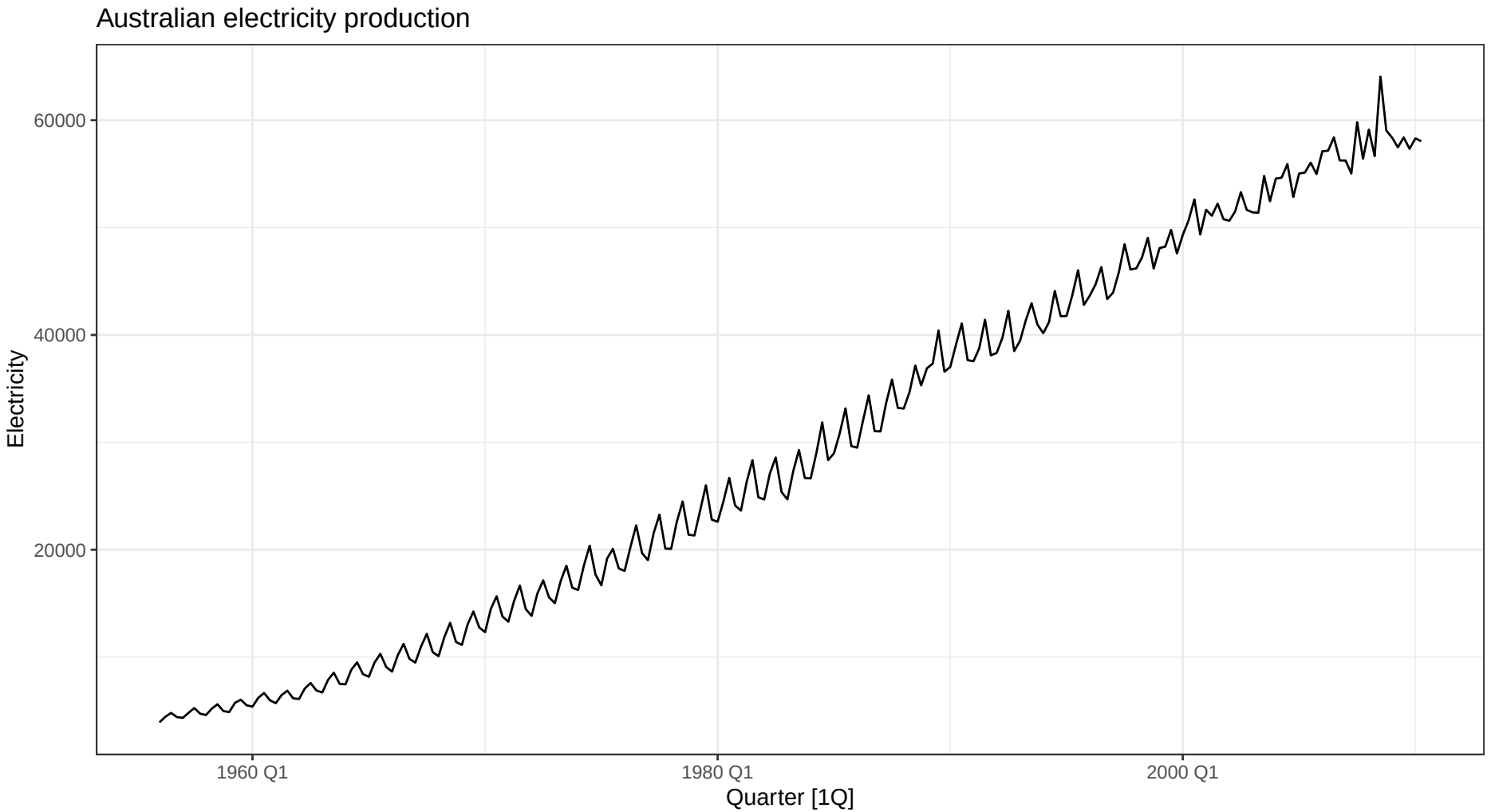
Mathematical transformations for stabilizing variation

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.

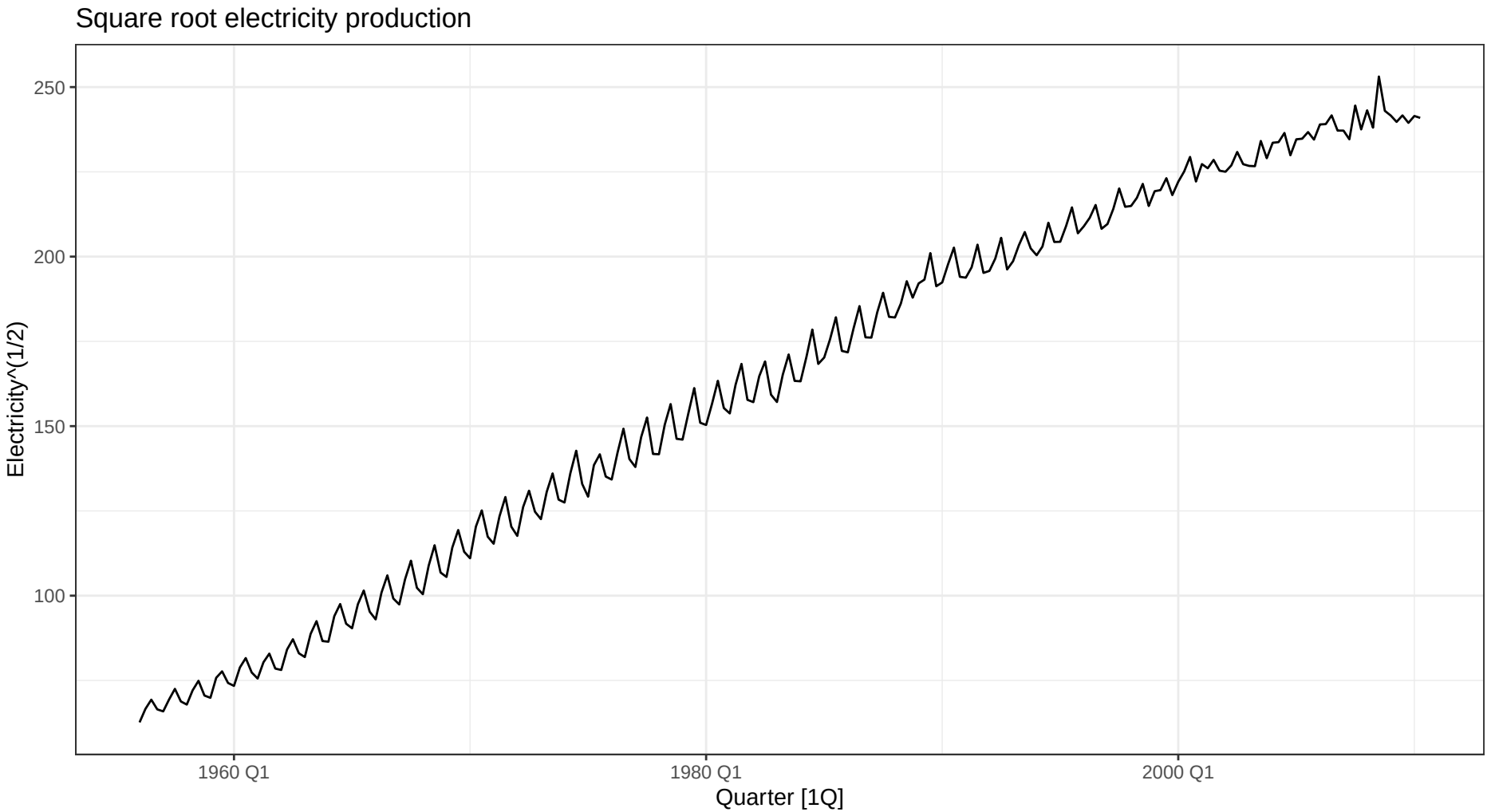
Variance stabilization

```
aus_production %>% autoplot(Electricity) +
  ggtitle("Australian electricity production")
```



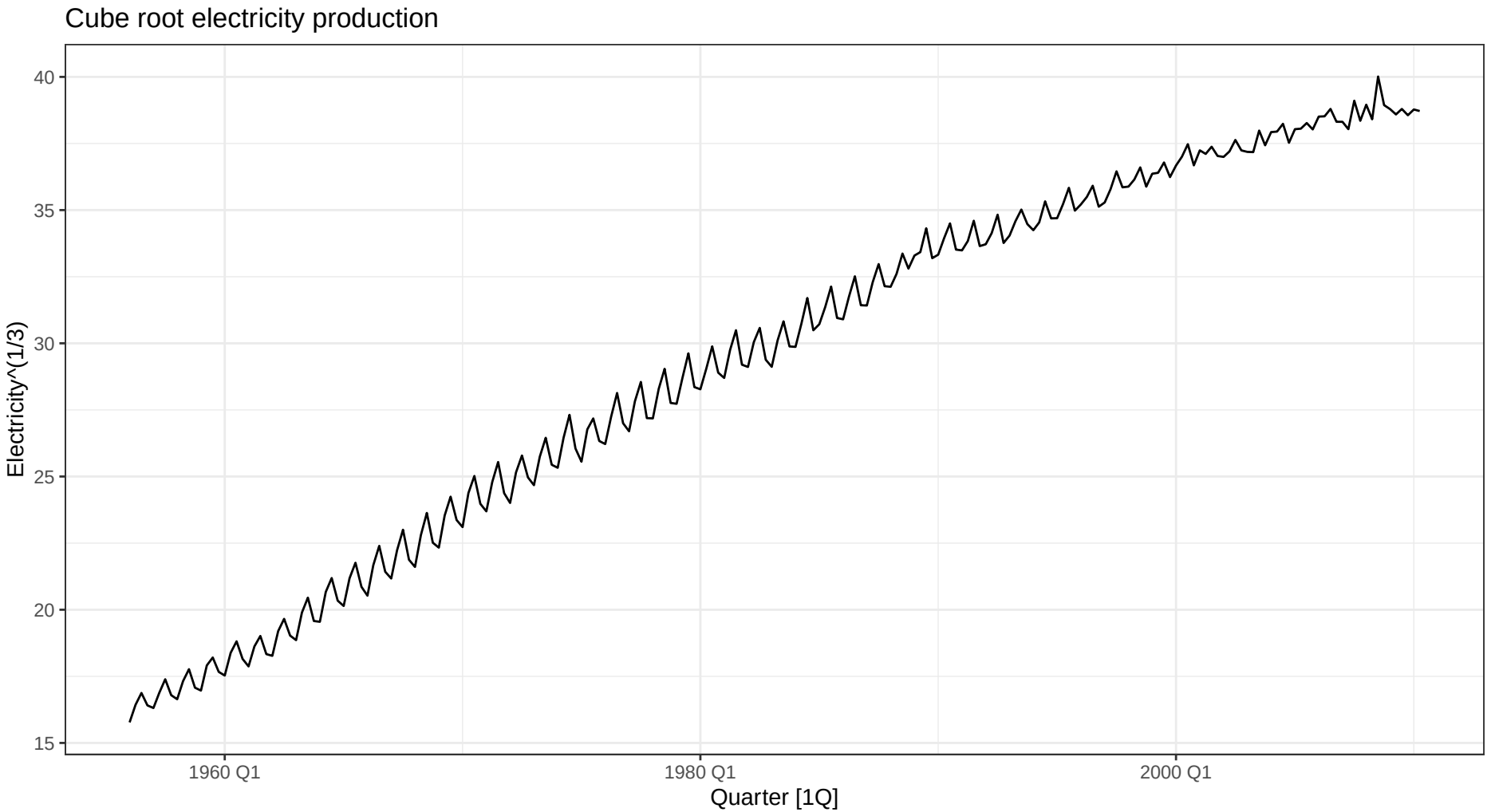
Variance stabilization

```
aus_production %>% autoplot(Electricity^(1/2)) +
ggtitle("Square root electricity production")
```



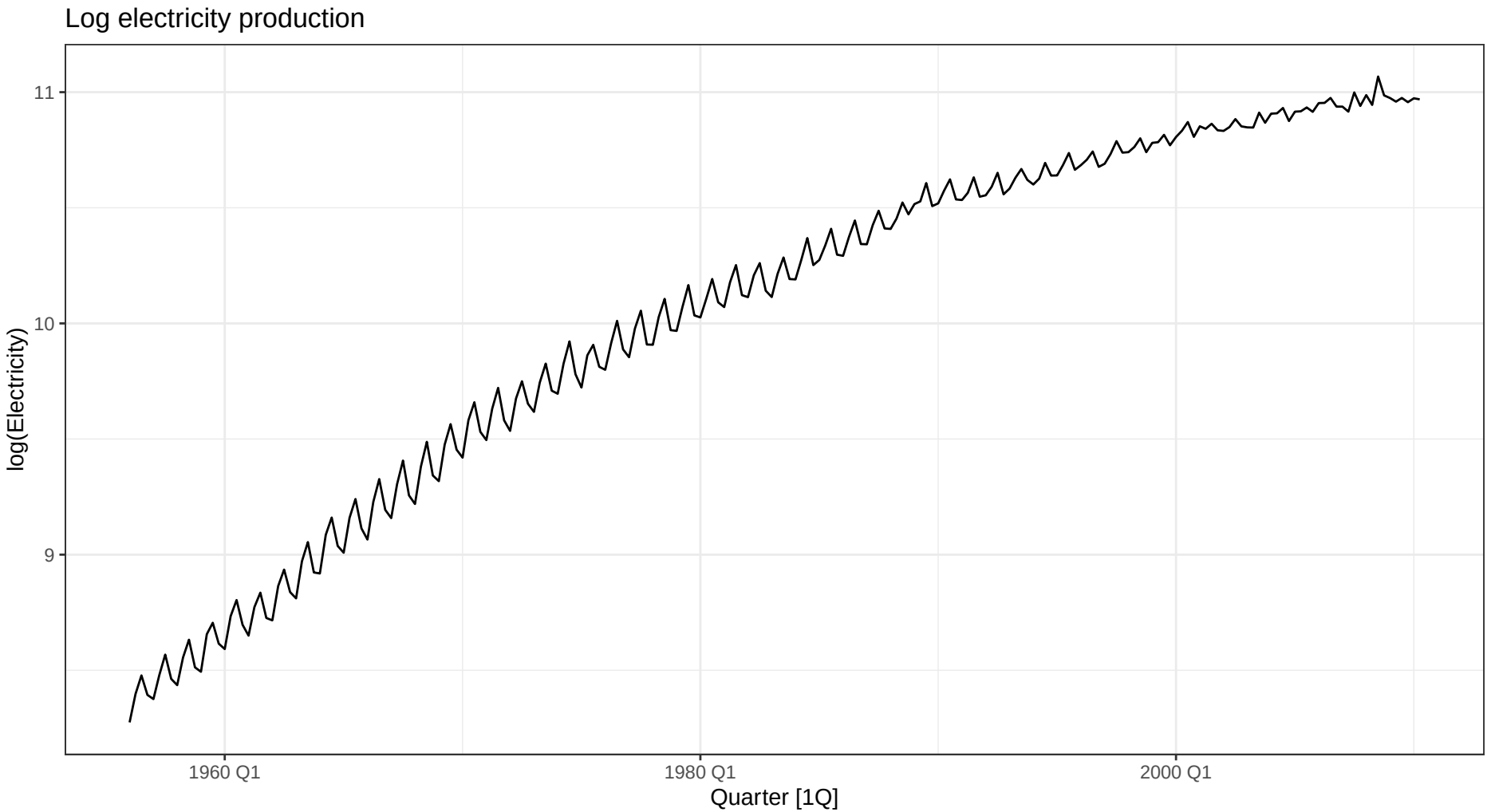
Variance stabilization

```
aus_production %>% autoplot(Electricity**(1/3)) +
  ggtitle("Cube root electricity production")
```



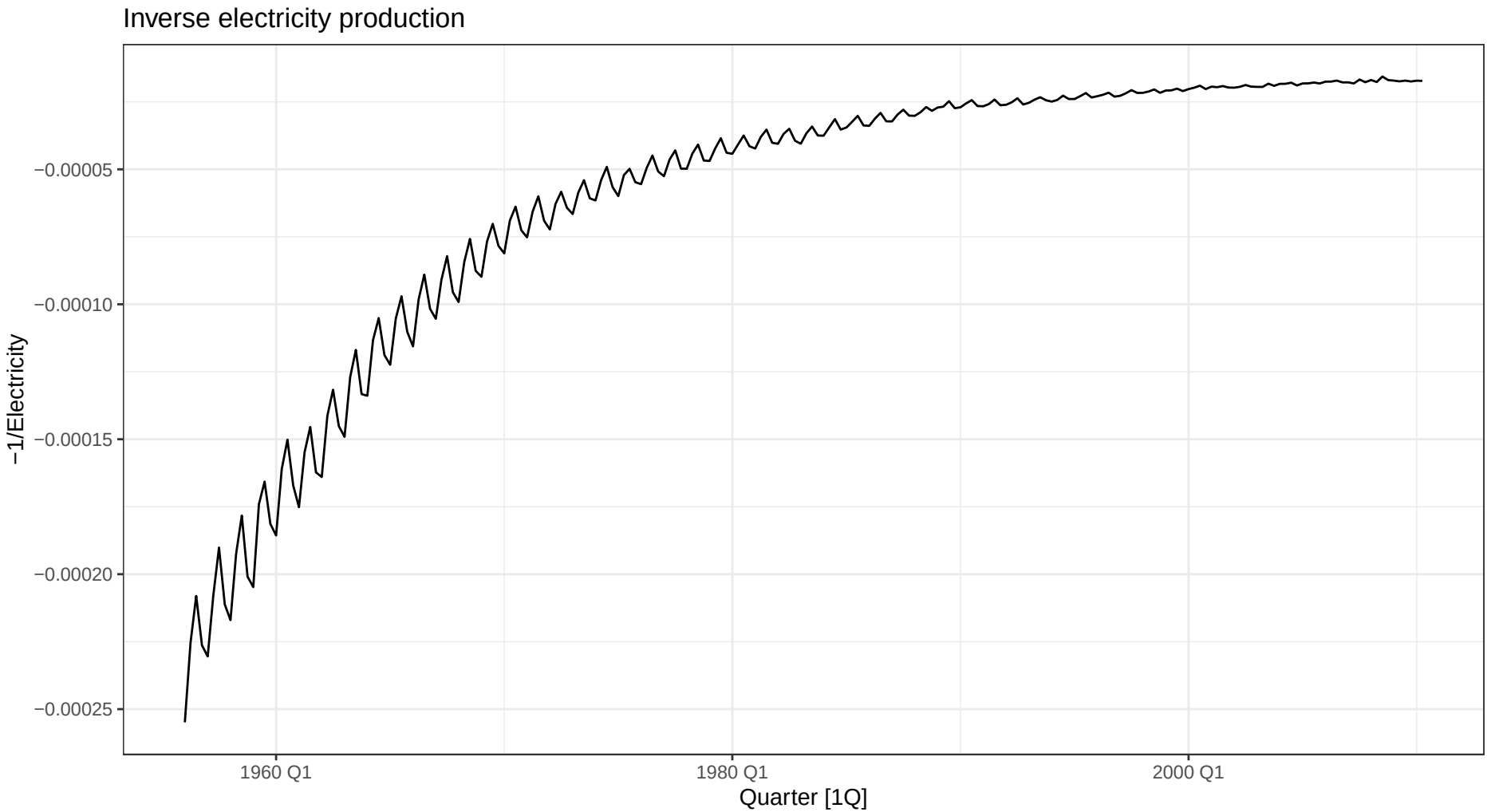
Variance stabilization

```
aus_production %>% autoplot(log(Electricity)) +
  ggtitle("Log electricity production")
```



Variance stabilization

```
aus_production %>% autoplot(-1/Electricity) +
  ggtitle("Inverse electricity production")
```



Box-Cox transformations

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

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

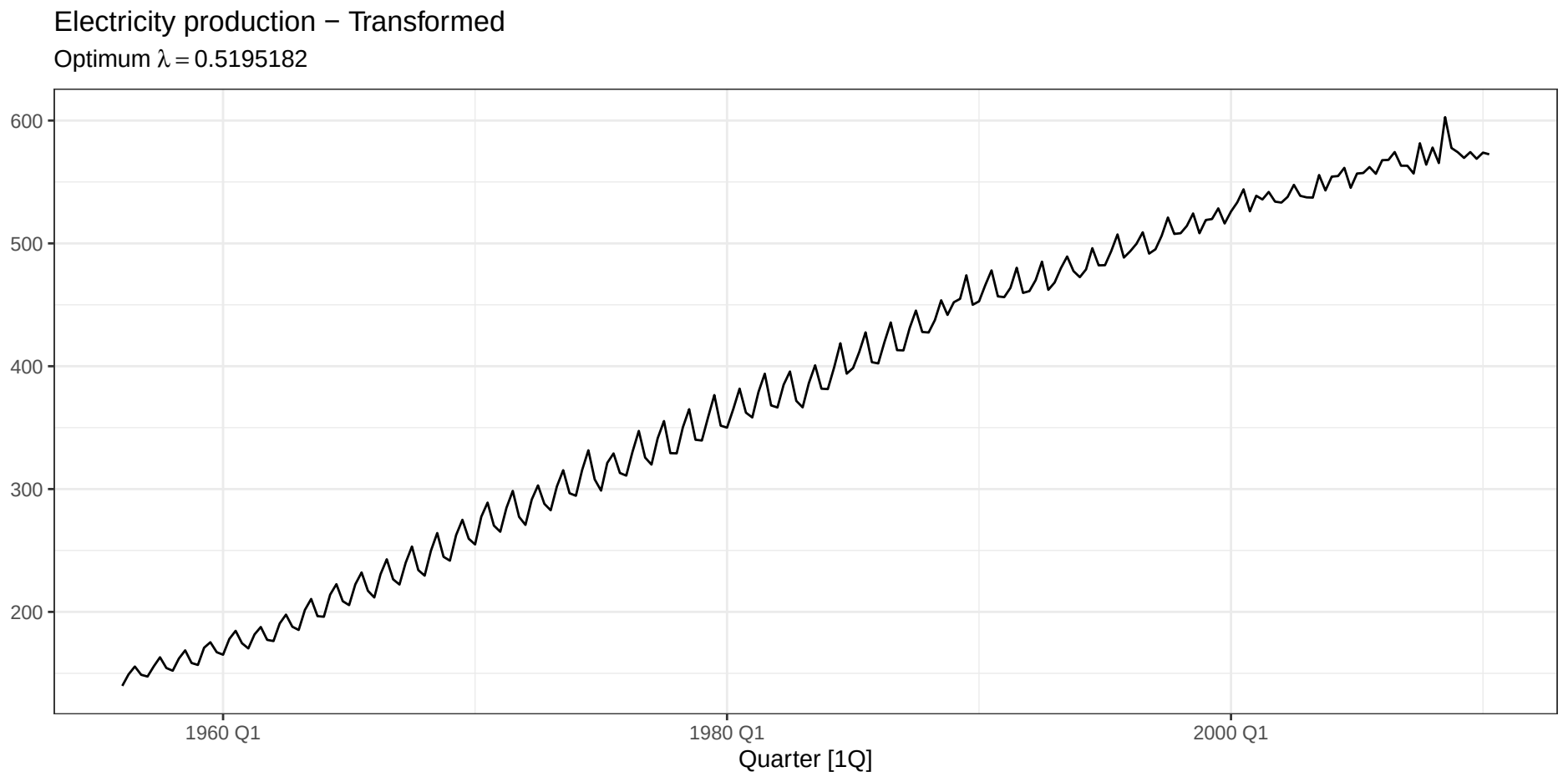
- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

```
(opt.lambda <- aus_production %>%
   features(Electricity, features = guerrero))
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1           0.520
```

Box-Cox transformations

```
aus_production %>% autoplot(box_cox(Electricity,opt.lambda)) +
  labs(title = "Electricity production - Transformed",
       subtitle = bquote("Optimum"~lambda == .(opt.lambda[[1]])),
       y = NULL)
```



Section 2

Time series components

Time series patterns

Recall

Trend pattern exists when there is a long-term increase or decrease in the data.

Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of **at least** 2 years).

Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). It tends to repeat itself at a fixed period.

Irregular pattern consists of unpredictable or random fluctuations.

Time series patterns

We can decompose a time series into several components:

1. Trend component represents the underlying growth (or decline) in a time series. The trend may be produced, for example, by consistent population change, inflation, technological change, and productivity increases.

2. Cyclical component a series of wavelike fluctuations or cycles of more than one year's duration. Changing economic conditions generally produce cycles.
   In practice, cycles are often difficult to identify and are frequently regarded as part of the trend. In this case, the underlying general growth (or decline) component is called the trend-cycle. **We will denote the trend-cycle component as T_t.**

3. Seasonal component typically found in quarterly, monthly, or weekly data. Seasonal variation refers to a more or less stable pattern of change that appears annually and repeats itself year after year. Seasonal patterns occur because of the influence of the weather or because of calendar-related events such as school vacations and national holidays. : **We will denote the seasonal component as S_t.**

4. Remainder (Irregular) component consists of unpredictable or random fluctuations. These fluctuations are the result of a myriad of events that individually may not be particularly important but whose combined effect could be large.
We will denote the trend component as R_t.

Time series decomposition

$$y_t = f(S_t, T_t, R_t)$$

where
$y_t =$ data at period t
$T_t =$ trend-cycle component at period t
$S_t =$ seasonal component at period t
$R_t =$ remainder(Irregular) component at period

Additive decomposition: $y_t = S_t + T_t + R_t$.

Multiplicative decomposition: $y_t = S_t \times T_t \times R_t$.

Mixed components model: $Y_t = T_t \times S_t + I_t$

Time series decomposition

Additive vs Multiplicative model

Additive

- Appropriate if magnitude of seasonal fluctuations does not vary with level.

Multiplicative

- Appropriate if seasonal are proportional to level of series.
- Multiplicative decomposition more prevalent with economic series

Alternative

We could use a Box-Cox transformation, and then use additive decomposition.

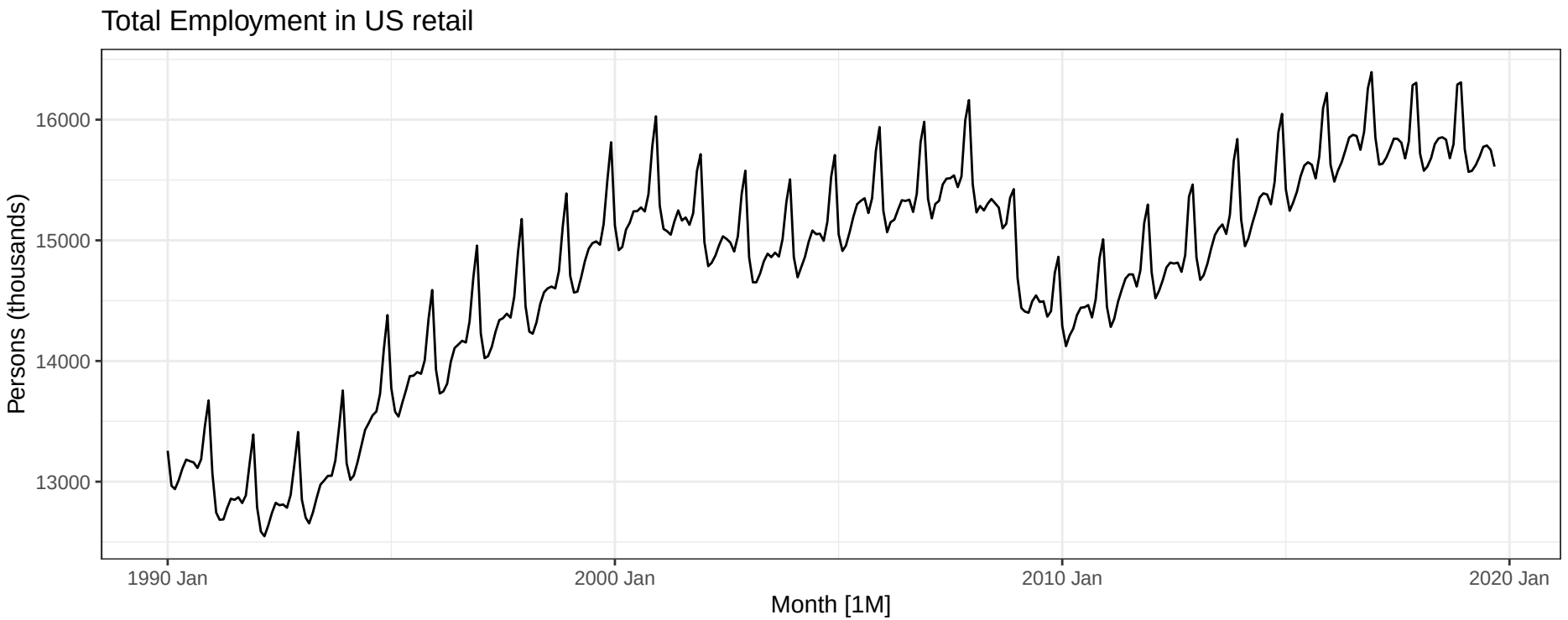
- Logs turn multiplicative relationship into an additive relationship:

$$y_t = S_t \times T_t \times R_t \quad \Rightarrow \quad \log y_t = \log S_t + \log T_t + \log R_t.$$

US Retail Sector Employment

What can you observe?

```
us_retail_employment <- us_employment %>%
  filter_index("1990 Jan"~.) %>%
  filter(Title == "Retail Trade")  %>% select(-Series_ID)

us_retail_employment %>%  autoplot(Employed) +
  labs(y = "Persons (thousands)", title = "Total Employment in US retail")
```



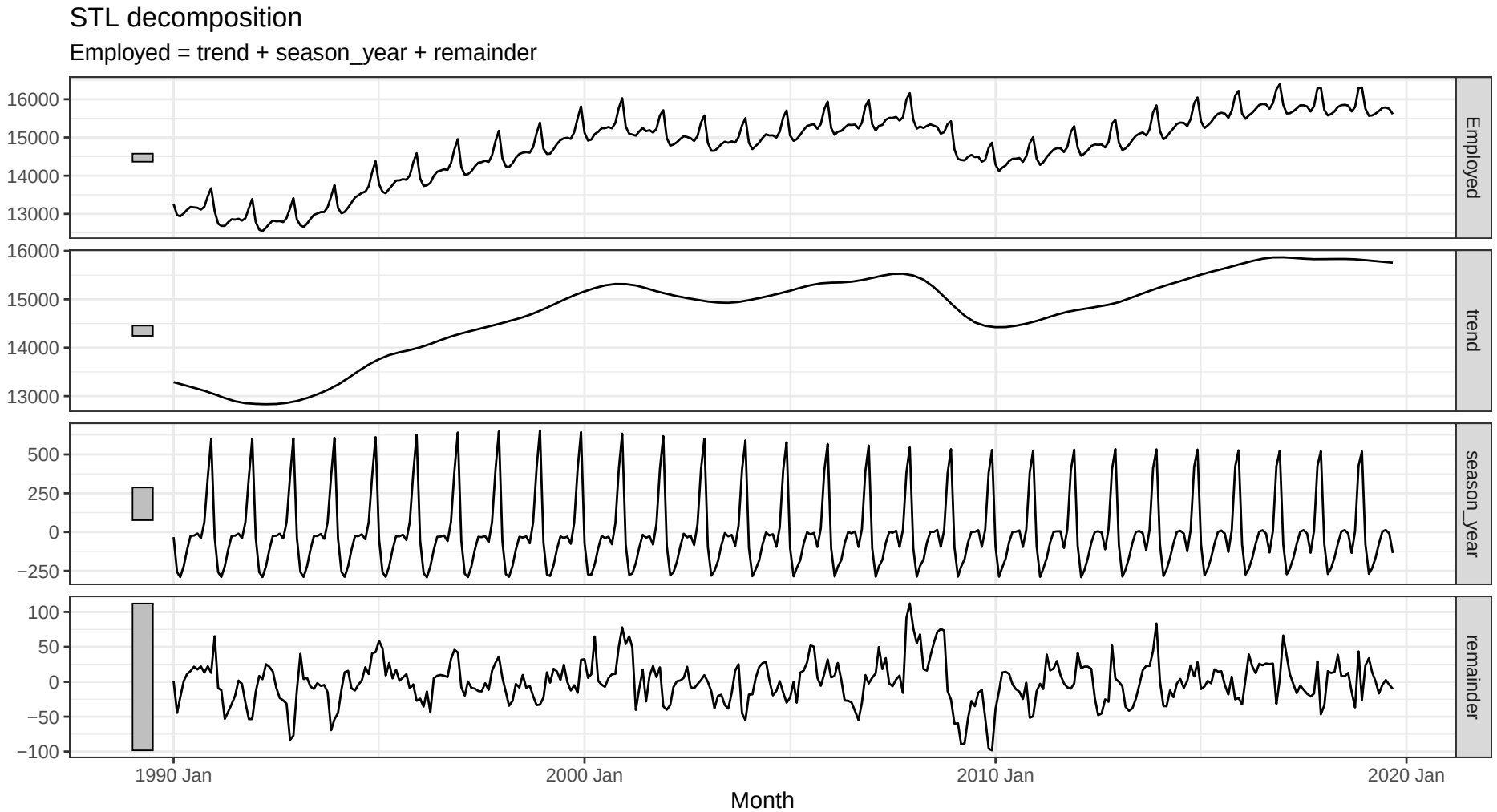
Decomposition in practice

```
dcomp <- us_retail_employment %>% model(
  stl = STL(Employed))

dcomp %>% components()
```

```
## # A dable: 357 x 7 [1M]
## # Key:     .model [1]
## # :        Employed = trend + season_year + remainder
##    .model    Month Employed  trend season_year remainder se
##    <chr>     <mth>    <dbl>  <dbl>       <dbl>     <dbl>
##  1 stl    1990 Jan   13256. 13288.      -33.0     0.836
##  2 stl    1990 Feb   12966. 13269.     -258.     -44.6
##  3 stl    1990 Mar   12938. 13250.     -290.     -22.1
##  4 stl    1990 Apr   13012. 13231.     -220.      1.05
##  5 stl    1990 May   13108. 13211.     -114.     11.3
##  6 stl    1990 Jun   13183. 13192.      -24.3    15.5
##  7 stl    1990 Jul   13170. 13172.      -23.2    21.6
```

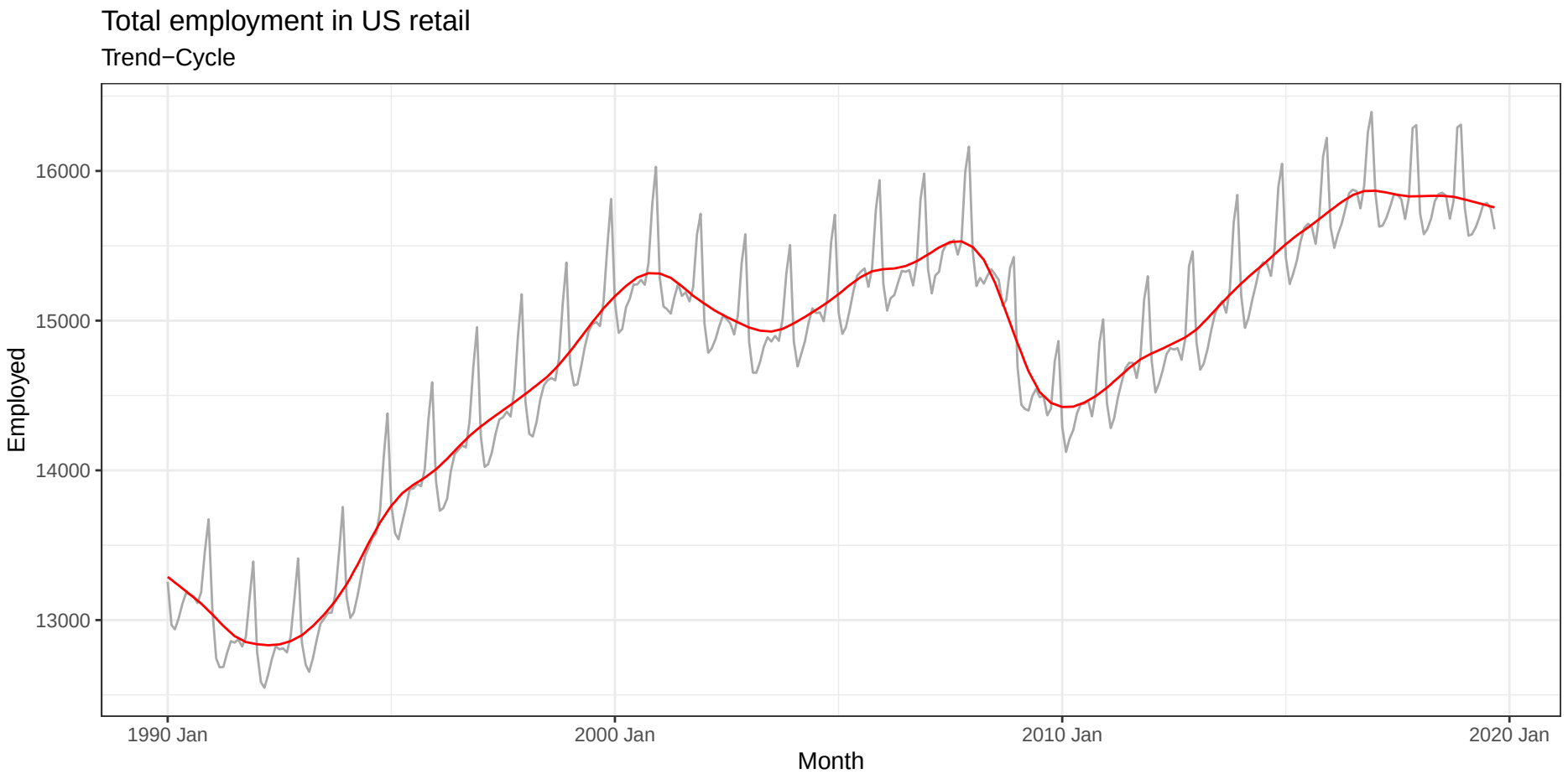
Decomposition in practice

```
dcomp %>% components %>% autoplot()
```



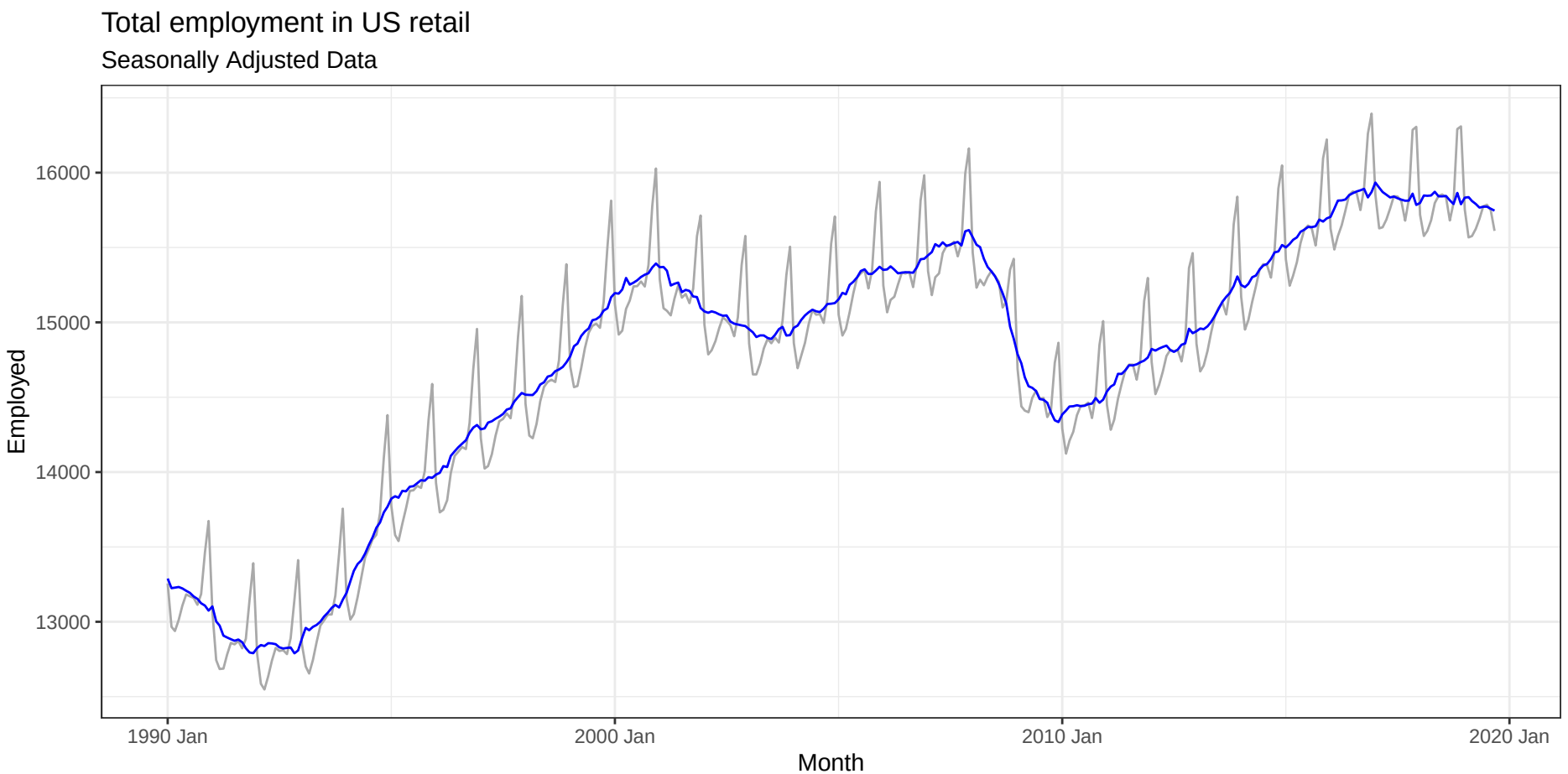
Decomposition in practice

```
dcomp %>% components() %>%
  ggplot(aes(x = Month)) +
  geom_line(aes(y = Employed), col = "darkgrey") +
  geom_line(aes(y = trend), col= "red") +
  labs(title = "Total employment in US retail", subtitle = "Trend-Cycle")
```



Decomposition in practice

```
dcomp %>% components() %>%
  ggplot(aes(x = Month)) +
  geom_line(aes(y = Employed), col = "darkgrey") +
  geom_line(aes(y = season_adjust), col= "blue") +
  labs(title = "Total employment in US retail",
       subtitle = "Seasonally Adjusted Data")
```



Moving Averages

A classical method for extracting the trend-cycle is via a moving average.

m-MA

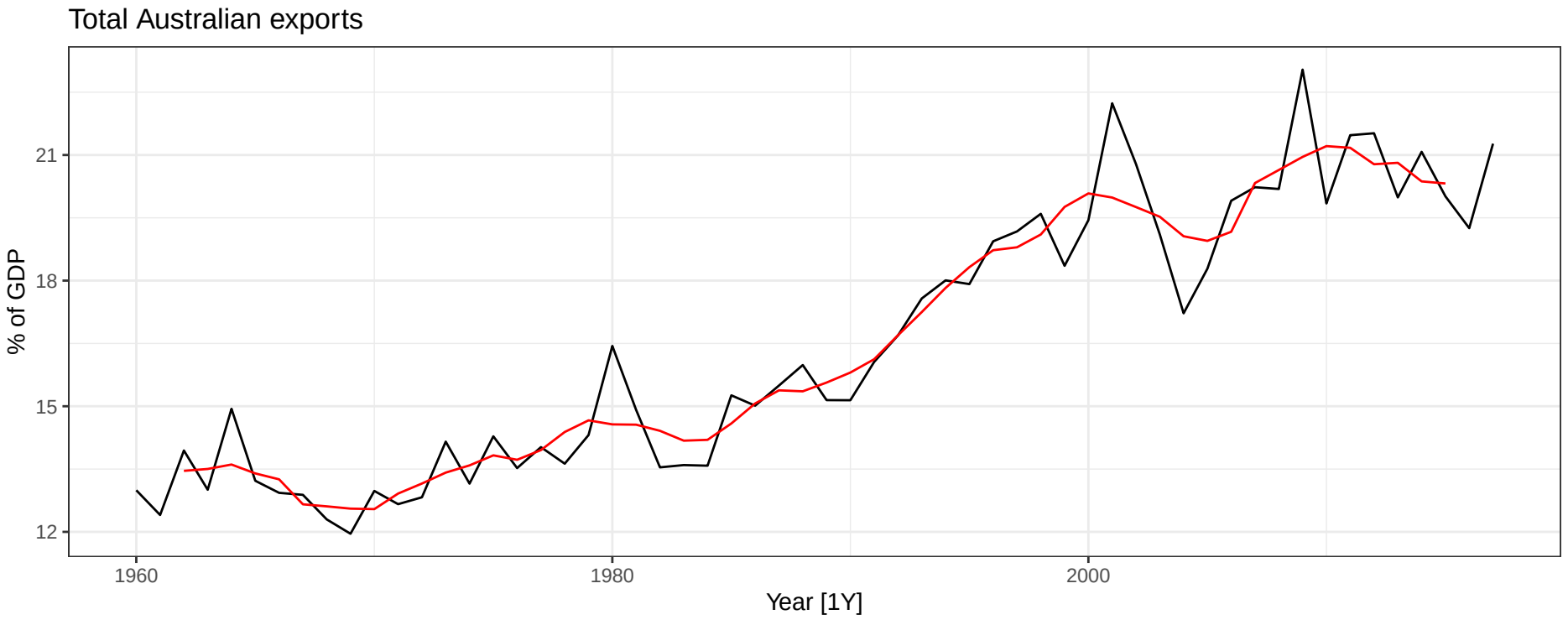
$$\hat{\mathrm{T}}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}$$

where $k = \frac{m-1}{2}$

Here,

- the estimate of the trend-cycle at time, t is obtained by averaging values of the time series within k periods of t.
- observations that are nearby in time are also likely to be close in value.
- the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component.

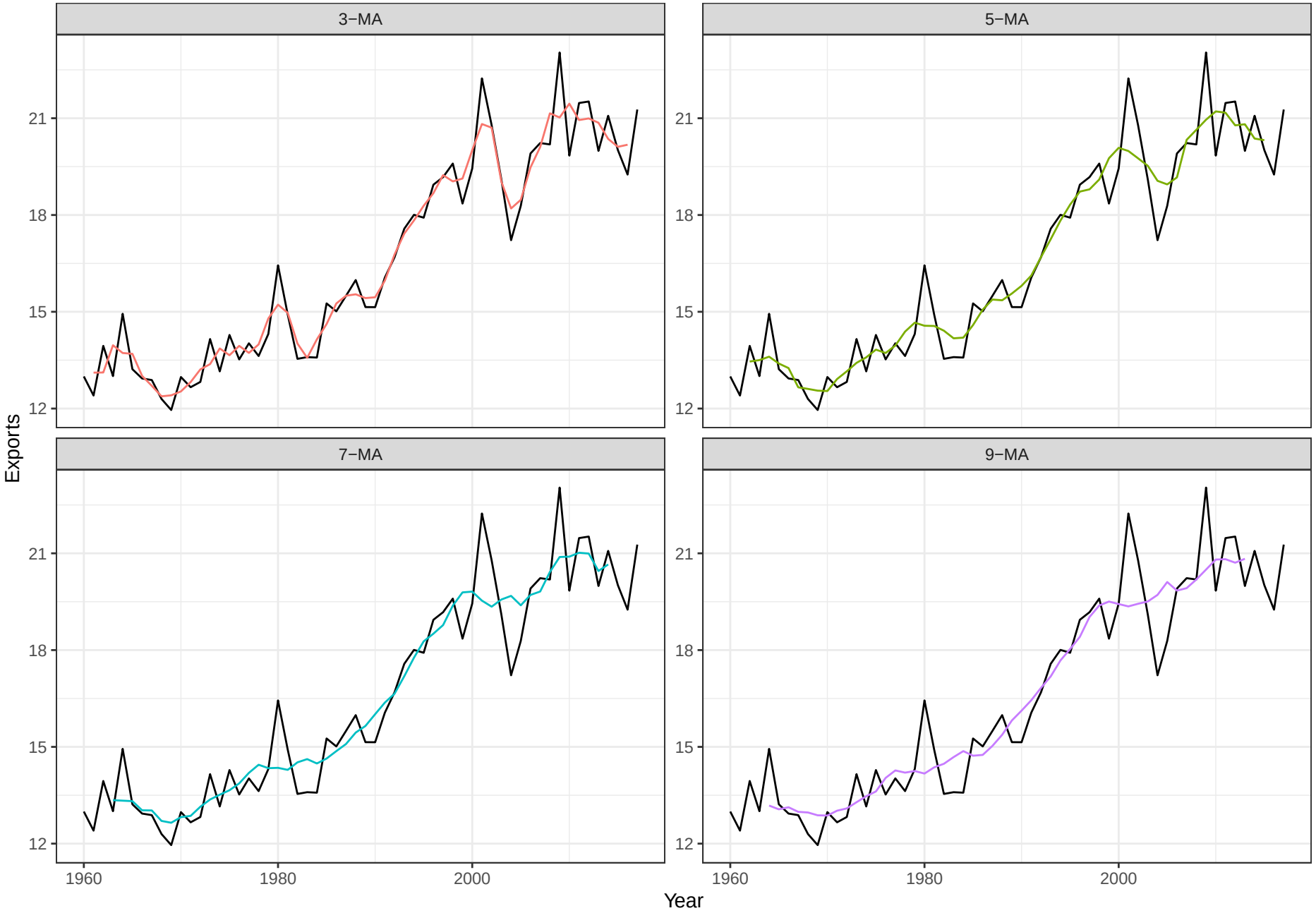
```r
aus_exports <- global_economy %>%
  filter(Country == "Australia") %>%
  mutate(`5-MA` = slider::slide_dbl(Exports, mean,
                .before = 2, .after = 2, .complete = TRUE)
  )

aus_exports%>% autoplot(Exports) +
  geom_line(aes(x= Year, y = `5-MA`), col = "red") +
  labs(y = "% of GDP", title = "Total Australian exports")
```



Moving Averages

We usually specify the order as odd numbers (e.g. $k = 3, 5, 7,$ etc.) to ensure that they are symmetric in m.

```
aus_exports %>% mutate(
  `3-MA` = slider::slide_dbl(Exports, mean, .before = 1,
                             .after = 1, .complete = TRUE),
  `7-MA` = slider::slide_dbl(Exports, mean, .before = 3,
                             .after = 3, .complete = TRUE),
  `9-MA` = slider::slide_dbl(Exports, mean, .before = 4,
                             .after = 4, .complete = TRUE)
) %>%
  pivot_longer(ends_with("-MA"), names_to = "MA") %>%
  ggplot(aes(x = Year)) + geom_line(aes(y = Exports)) +
  geom_line(aes(y = value, col = MA)) +
  facet_wrap(MA ~. , scales = "free_y") +
  theme(legend.pos = "none")
```


3−MA
5−MA
7−MA
9−MA
Exports
Year
12
15
18
21
1960
1980
2000

Moving Averages of Moving Averages

It is possible to apply a moving average to a moving average. One reason for doing this is to make an even-order moving average symmetric.

```
beer <- aus_production  %>%
  filter(year(Quarter) >= 1992)  %>%
  select(Quarter, Beer)

beer_ma <- beer  %>%
  mutate(
  #Compute a 4-MA
    `4-MA` = slider::slide_dbl(Beer, mean,
                .before = 1, .after = 2, .complete = TRUE),
  # compute a 4-MA followed by a 2-MA
    `2x4-MA` = slider::slide_dbl(`4-MA`, mean,
                .before = 1, .after = 0, .complete = TRUE)
  )
```

```
## # A tsibble: 74 x 4 [1Q]
##    Quarter  Beer `4-MA` `2x4-MA`
##      <qtr> <dbl>  <dbl>    <dbl>
##  1 1992 Q1   443     NA       NA
##  2 1992 Q2   410   451.       NA
##  3 1992 Q3   420   449.      450
##  4 1992 Q4   532   452.      450.
##  5 1993 Q1   433   449       450.
##  6 1993 Q2   421   444       446.
##  7 1993 Q3   410   448       446
##  8 1993 Q4   512   438       443
##  9 1994 Q1   449   441.      440.
## 10 1994 Q2   381   446       444.
## # ... with 64 more rows
```

Moving Averages of Moving Averages

Final thoughts:

- Other combinations of moving averages are also possible. For example, a 3×3-MA is often used, and consists of a moving average of order 3 followed by another moving average of order 3.
- In general, **an even order MA should be followed by an even order MA to make it symmetric**.
- Similarly, **an odd order MA should be followed by an odd order MA**.

History of time series decomposition

- Classical method originated in 1920s.
- Census II method introduced in 1957. Basis for X-11 method and variants (including X-12-ARIMA, X-13-ARIMA)
- STL method introduced in 1983
- TRAMO/SEATS introduced in 1990s.

National Statistics Offices

- X-12-ARIMA:
  - Created by the U.S. Census Bureau
  - ABS
  - Statistics Canada
  - ONS (UK)
  - U.S. Bureau of Labor Statistics
- X-13-ARIMA-SEATS:
  - US Census Bureau
  - EuroStat

Classical Decomposition

- There are two forms of classical decomposition:
  - an additive decomposition and
  - a multiplicative decomposition.
- In the classical decomposition, we assume that the seasonal component is constant from year to year.
- For multiplicative seasonality, the m values that form the seasonal component are sometimes called the "seasonal indices."

Please see the R notebook file for the In-class exercise here!

Drawbacks of the classical decomposition

1. The estimate of the trend-cycle is unavailable for the first few and last few observations. Consequently, there is also no estimate of the remainder component for the same time periods.
2. The trend-cycle estimate tends to over-smooth rapid rises and falls in the data.
3. Assume that the seasonal component repeats from year to year. For many series, this is a reasonable assumption, but for some longer series it is not.
4. Occasionally, the values of the time series in a small number of periods may be particularly unusual (think outliers). The classical method is not robust to these kinds of unusual values.

Section 3

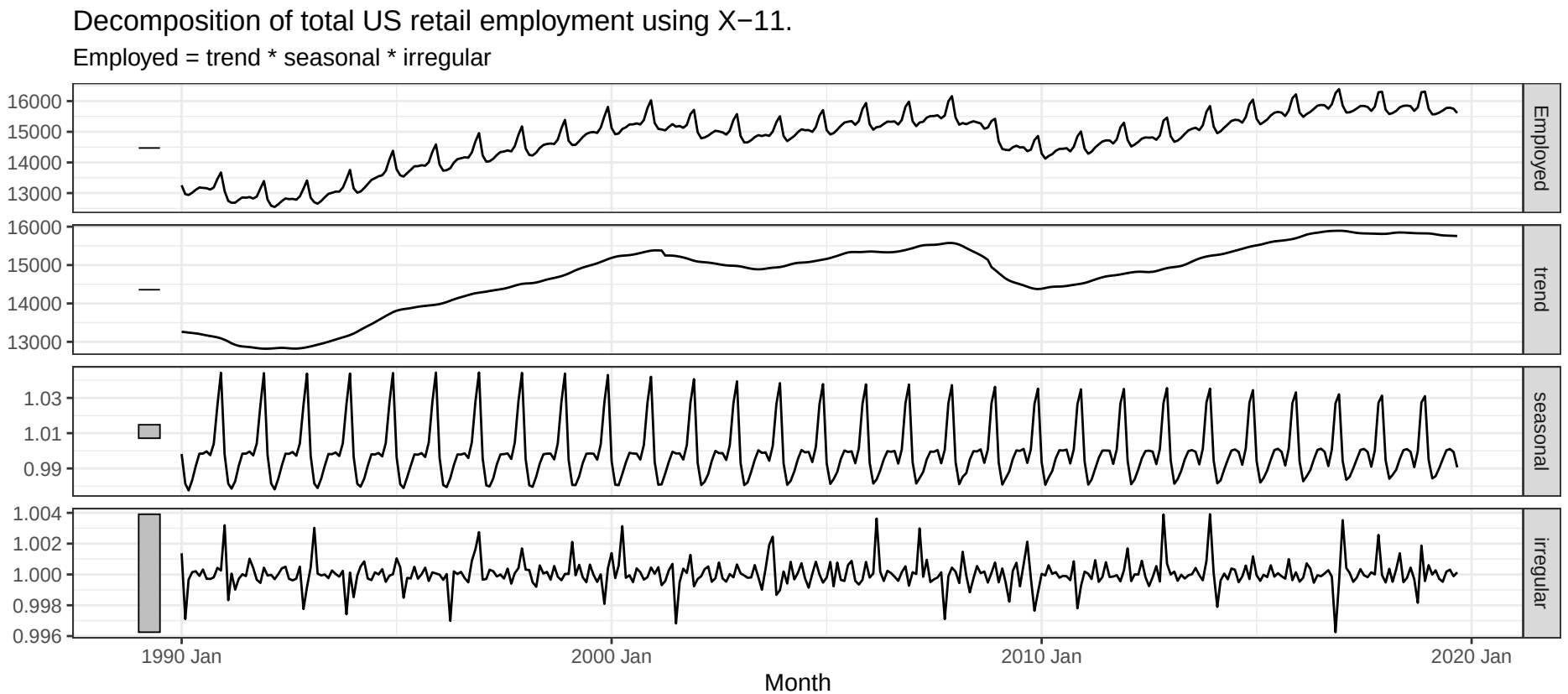
X-11 decomposition

X-11 decomposition

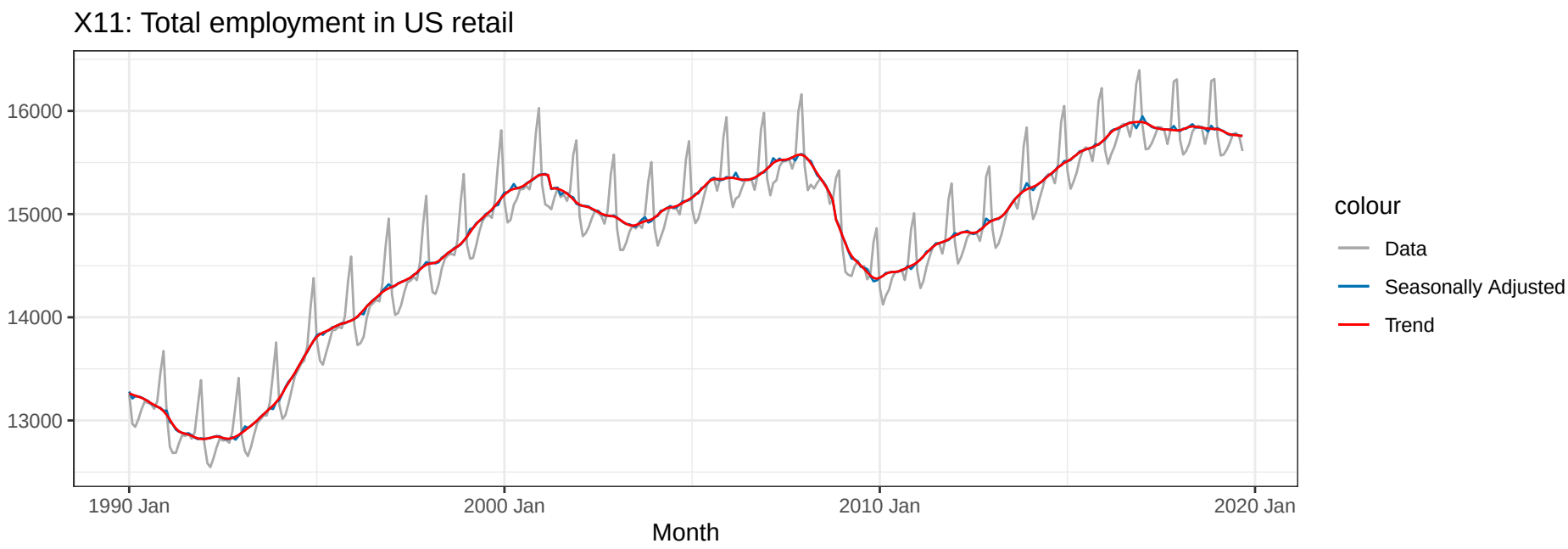
- This method is based on classical decomposition, but includes many extra steps and features to overcome the drawbacks of classical decomposition (See Slide 38).
  - Trend-cycle estimates are available for all observations including the end points.
  - Seasonal component is allowed to vary slowly over time.
- X11 also includes methods for handling trading day variation, holiday effects and the effects of known predictors.
- The process is entirely automatic and tends to be highly robust to outliers and level shifts in the time series.

X-11: US Retail Employment

```
#require(seasonal)
x11_dcmp <- us_retail_employment %>%
  model(x11 = X_13ARIMA_SEATS(Employed ~ x11()))  %>%
  components()

autoplot(x11_dcmp) +
  labs(title = "Decomposition of total US retail employment using X-11.")
```



```
x11_dcmp %>%
  ggplot(aes(x = Month)) +
  geom_line(aes(y = Employed, colour = "Data")) +
  geom_line(aes(y = season_adjust,
                colour = "Seasonally Adjusted")) +
  geom_line(aes(y = trend, colour = "Trend")) +
  labs(y = NULL, title = "X11: Total employment in US retail") +
  scale_colour_manual(
    values = c("darkgray", "#0072B2", "red"),
    breaks = c("Data", "Seasonally Adjusted", "Trend")
  )
```



(Dis)advantages of X-11

Advantages

- Relatively robust to outliers
- Completely automated choices for trend and seasonal changes
- Very widely tested on economic data over a long period of time.

Disadvantages

- No prediction/confidence intervals
- Ad hoc method with no underlying model
- **Only developed for quarterly and monthly data**

Extensions: X-12-ARIMA and X-13-ARIMA

- The X-11, X-12-ARIMA and X-13-ARIMA methods are based on Census decomposition.
- These allow adjustments for trading days and other explanatory variables.
- Known outliers can be omitted.
- Level shifts and ramp effects can be modeled.
- Missing values estimated and replaced.
- Holiday factors (e.g., Easter, Labour Day) can be estimated.

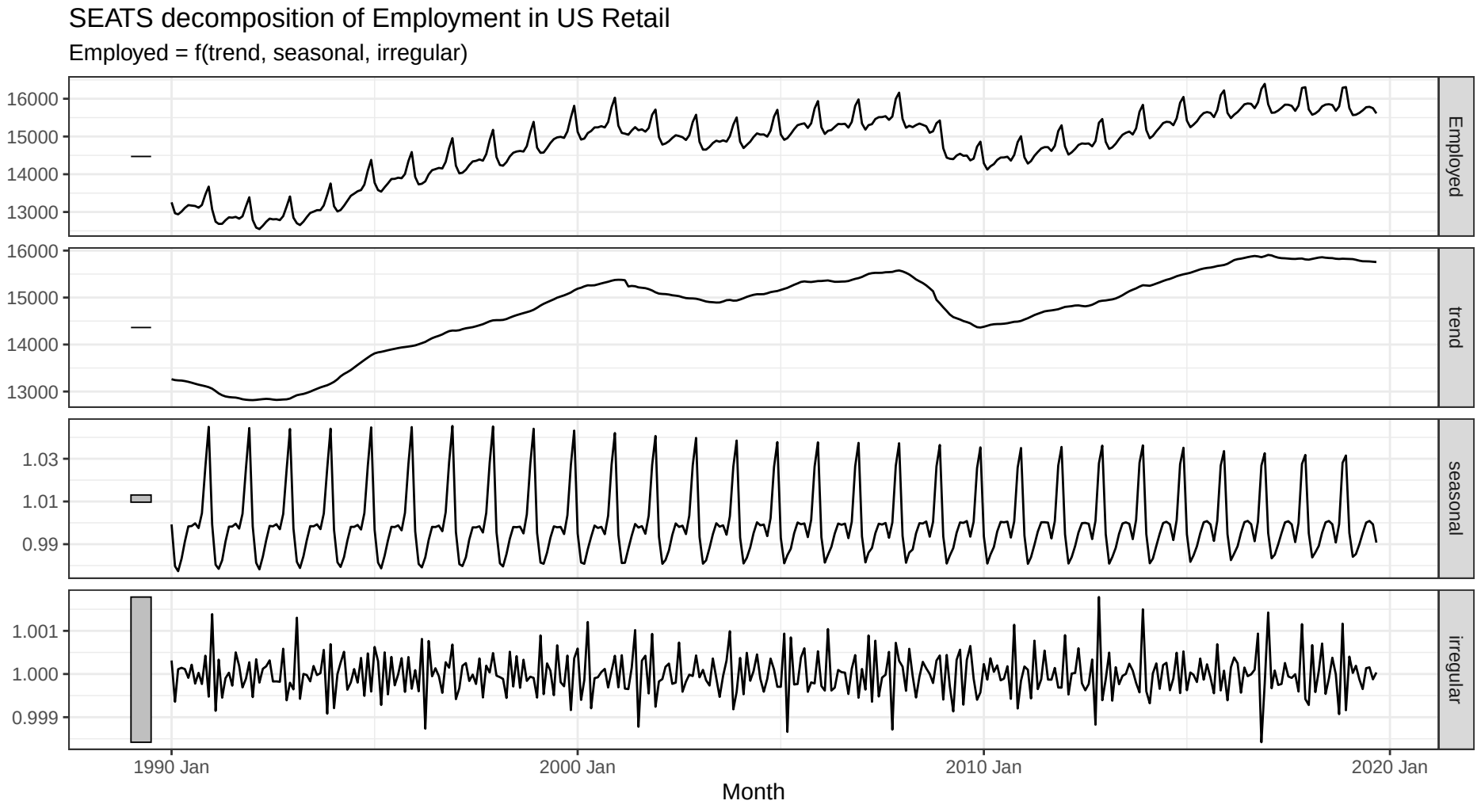
Section 4

Seasonal Extraction in ARIMA Time Series (SEATS) decomposition

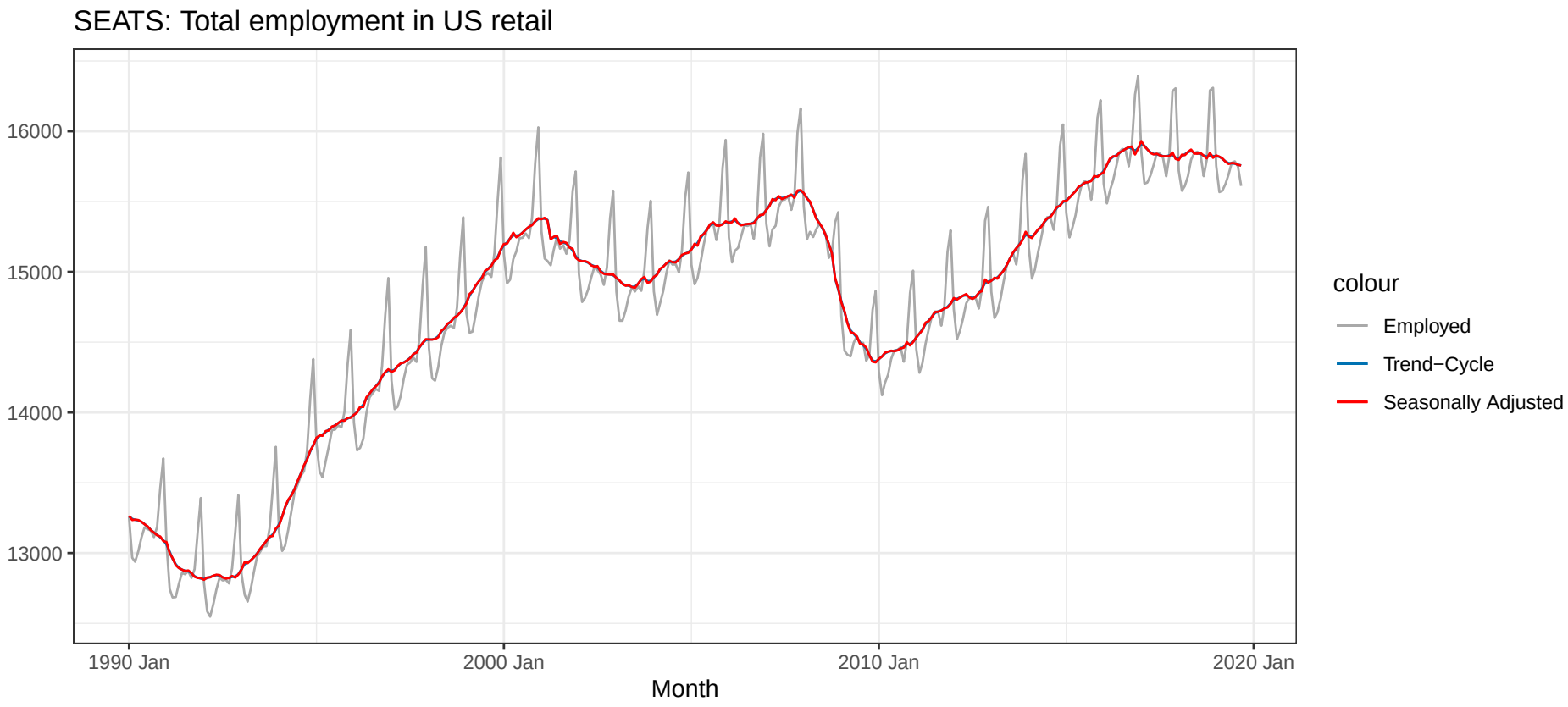
SEATS decomposition

- Developed at the Bank of Spain, and is now widely used by government agencies around the world.
- The procedure works **only with quarterly and monthly data**.
  - So seasonality of other kinds, such as daily data, or hourly data, or weekly data, require an alternative approach.

```
seats_dcmp <- us_retail_employment %>%
  model(seas = X_13ARIMA_SEATS(Employed ~ seats())) %>%
  components()
seats_dcmp %>% autoplot() +
  ggtitle("SEATS decomposition of Employment in US Retail")
```



```r
seats_dcmp %>% ggplot(aes(x = Month)) +
  geom_line(aes(y = Employed, colour="Employed")) +
  geom_line(aes(y = trend, col = "Trend-Cycle")) +
  geom_line(aes(y = season_adjust, col = "Seasonally Adjusted")) +
  labs(y = NULL, title = "SEATS: Total employment in US retail") +
  scale_colour_manual(
    values = c("darkgray", "#0072B2", "red"),
    breaks = c("Employed", "Trend-Cycle", "Seasonally Adjusted")
  )
```



(Dis)advantages of SEATS

Advantages

- Model-based
- Smooth trend estimate
- Allows estimates at end points
- Allows changing seasonality
- Developed for economic data

Disadvantages

- Only developed for quarterly and monthly data

Section 5

Seasonal and Trend decomposition using Loess (STL) Method

(Dis)advantages of STL

Advantages

- Loess is a method for estimating nonlinear relationships.
  - Very versatile and robust.
- Unlike SEATS and X11, **STL will handle any type of seasonality, not only monthly and quarterly data.**
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle can be controlled by user.
- Robust to outliers (i.e., user can specify a robust decomposition), so that occasional unusual observations will not affect the estimates of the trend-cycle and seasonal components. They will, however, affect the remainder component.

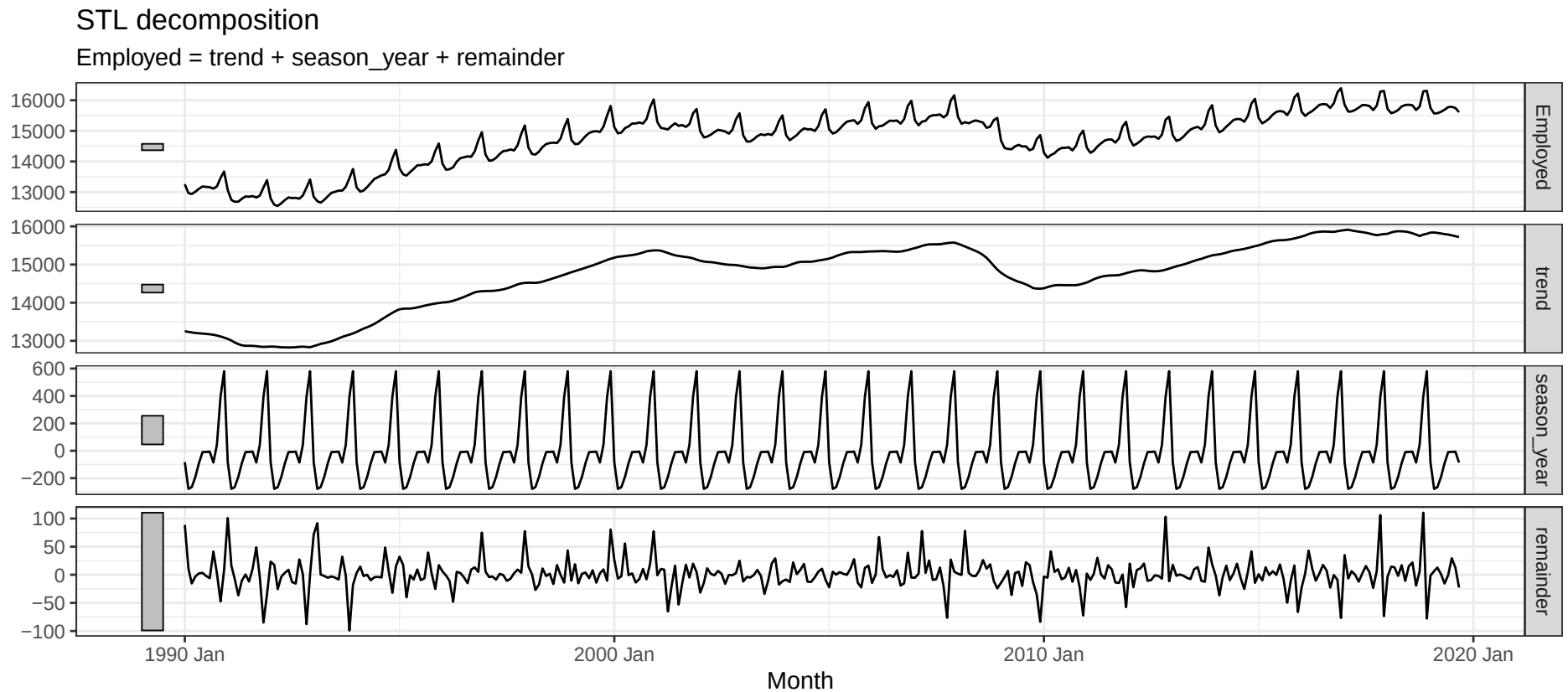
(Dis)advantages of STL

Disadvantages

- Not able to handle trading day or calendar adjustments.
- **Only allows for additive models**.
  - We can obtain an additive decomposition by first taking logs, then back transforming the components.
  - We can use the Box-Cox transformations to get other decompositions.

STL decomposition: US Employment in Retail

```
stl_dcmp <-  us_retail_employment %>%
  model(STL(Employed ~ trend(window = 7) +
              season(window = "periodic"), robust = TRUE)) %>% components()
stl_dcmp  %>% autoplot()
```



STL decomposition

- `trend(window = ?)` and `season(window = ?)` control how rapidly the trend-cycle and seasonal components can change.
- Both should be odd numbers:
  - `trend(window = ?)` is the number of consecutive observations to be used when estimating the trend-cycle. (By default `trend(window=21)`)
  - `season(window = ?)` is the number of consecutive years to be used in estimating each value in the seasonal component. (By default, `season(window=13)`)
  - Setting the seasonal window to be infinite is equivalent to forcing the seasonal component to be **periodic**: `season(window='periodic')` (i.e., identical across years).
- Smaller values allow for more rapid changes.

What happens as you change `season(window = ?)` and `trend(window = ?)`?

Section 6

Forecasting with Decomposition

Forecasting with Decomposition

Assuming an additive decomposition, the decomposed time series can be written as

$$y_t = \hat{S}_t + \hat{A}_t,$$

where $\hat{A}_t = \hat{T}_t + \hat{R}_t$ is the seasonally adjusted component.

For a multiplicative decomposition, we can write:

$$y_t = \hat{S}_t \cdot \hat{A}_t,$$

where $\hat{A}_t = \hat{T}_t \cdot \hat{R}_t$

To forecast a decomposed time series, we forecast the seasonal component, $\hat{S}_t$ and the seasonally adjusted component, $\hat{A}_t$, separately.

Forecasting with Decomposition

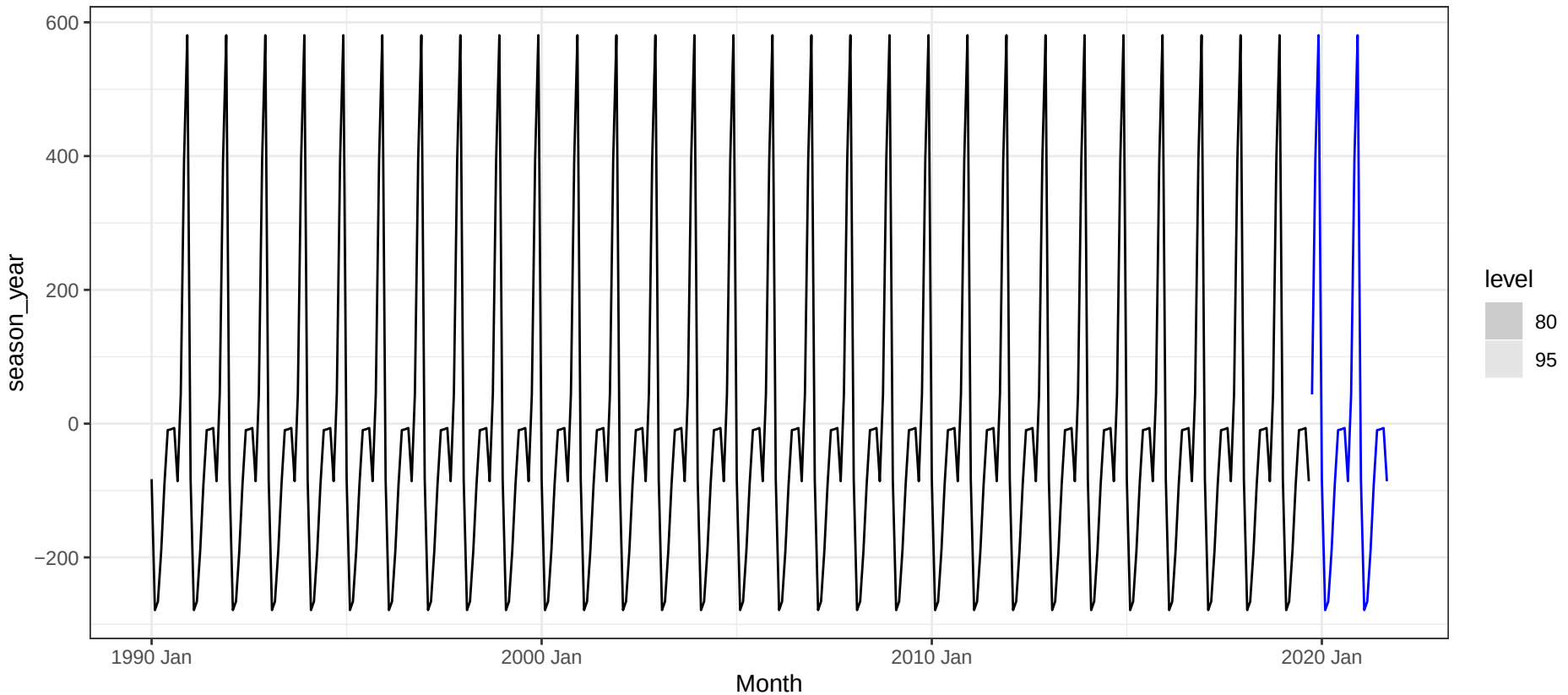
- It is usually assumed that the seasonal component is unchanging, or changing extremely slowly, so it is forecast by simply taking the last year of the estimated component (think seasonal naïve) as the forecasted value for the **seasonal component**.
- **In short, forecast seasonal component by repeating the last year.**
- To forecast the **seasonally adjusted component**, any *non-seasonal* forecasting method may be used.
  - For example, the drift method, or Holt's method (to be discussed), or a non-seasonal ARIMA model (also to be discussed), may be used.
- At the end, we will combine the forecasts of seasonal component with forecasts of seasonally adjusted data to get forecasts of original data. This is referred to as "reseasonalizing" the data.

Forecasting Employment in US Retail

```
#Forecasting Seasonally Adjusted Portion
stl_dcmp2 <-  stl_dcmp %>% select(-.model)
fore.seas.adj <- stl_dcmp2 %>% model(NAIVE(season_adjust))  %>%
  forecast()
fore.seas.adj %>% autoplot(stl_dcmp2) +
  labs(title = "Seasonally Adjusted Employment in Retail",
       subtitle = "Naive Forecast")
```

Forecasting Employment in US Retail

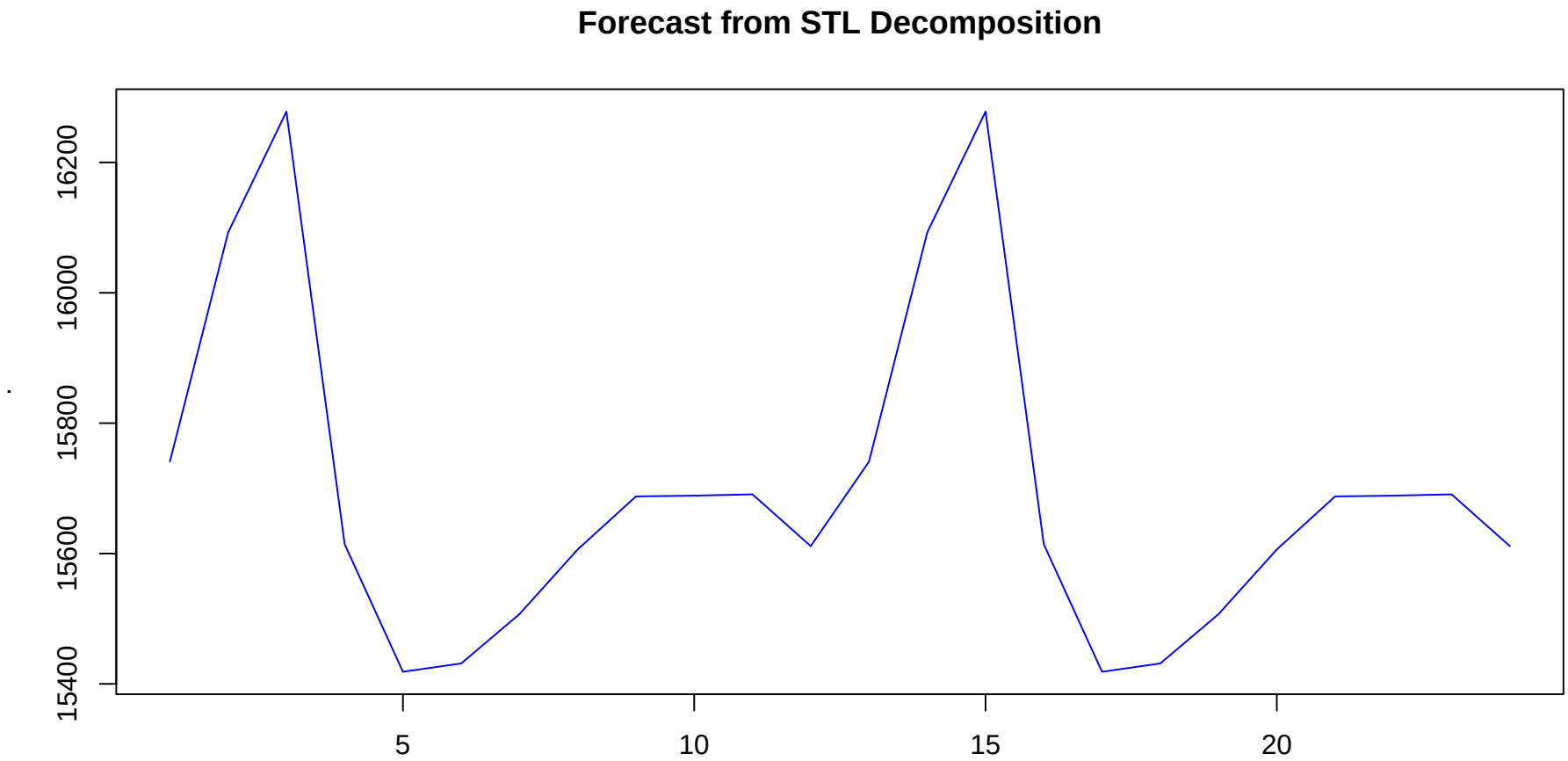
```
#Forecast Seasonal Component
fore.season <- stl_dcmp2 %>% model(SNAIVE(season_year)) %>% forecast()
fore.season %>% autoplot(stl_dcmp2)
```



Reseasonalizing Employment in US Retail

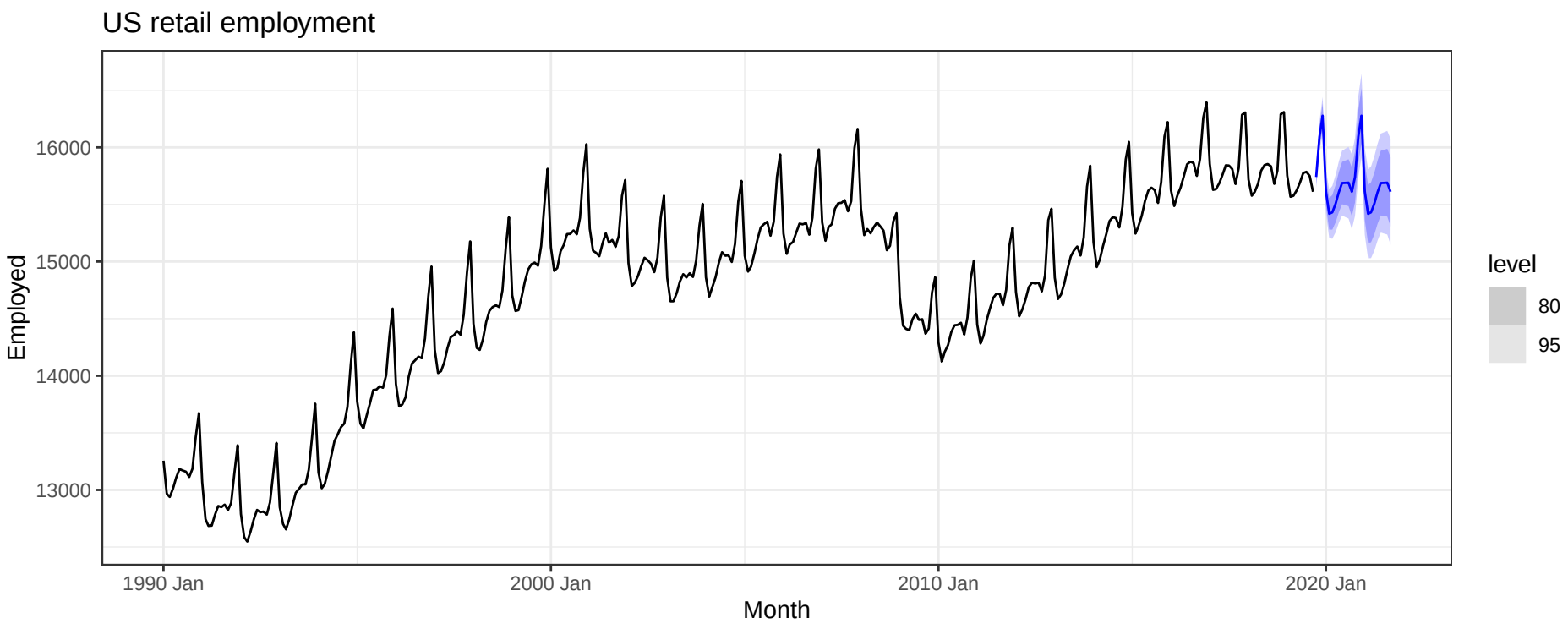
Manually

```
reseasoned <- rowSums(cbind(fore.seas.adj$.mean, fore.season$.mean))
reseasoned %>%  plot(type = "l", col = "blue",
                     main = "Forecast from STL Decomposition")
```



Using the decomposition_model() function

```
fit_dcmp <- us_retail_employment %>%
  model(stlf = decomposition_model(
    STL(Employed ~ trend(window = 7) + season(window = "periodic"),
        robust = TRUE), NAIVE(season_adjust)
  ))

fit_dcmp %>% forecast()  %>%
  autoplot(us_retail_employment)+ labs(title = "US retail employment")
```



Diagnostic Tests

```
fit_dcmp |> gg_tsresiduals()
```