MA 5124 Financial Time Series Analysis and Forecasting

Chapter 1: Introduction to time series and forecasting
Lesson 2

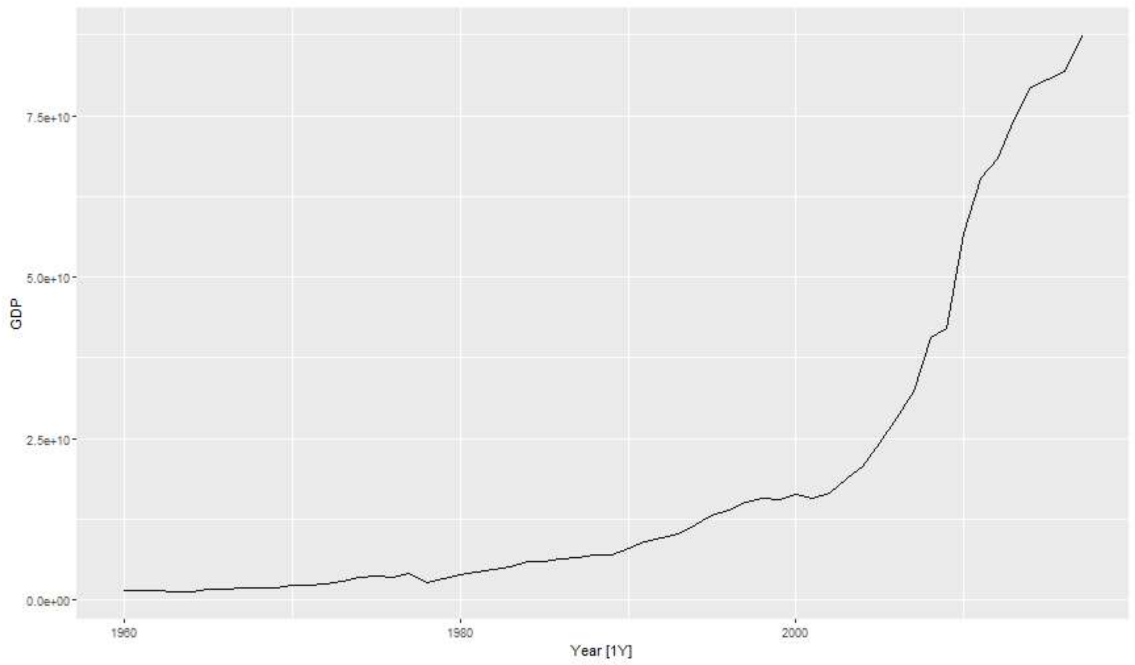
Dr. Priyanga Talagala

30-06-2024

Transformations and adjustments

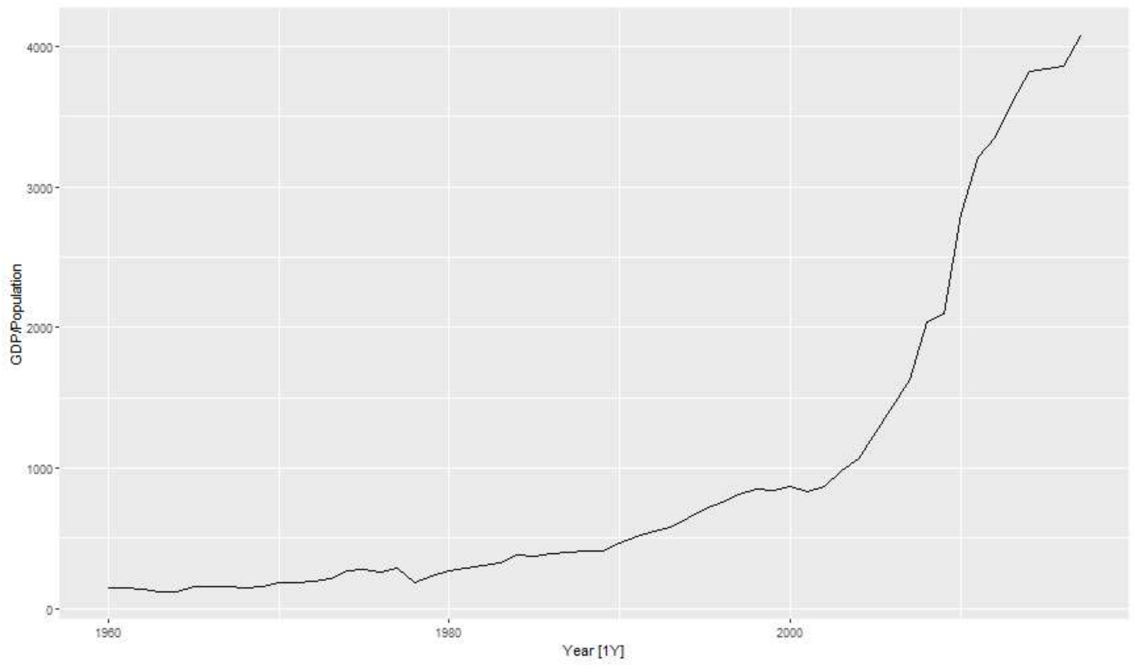
Per capita adjustments

```
global_economy |>
  filter(Country == "Sri Lanka") |>
  autoplot(GDP)
```



Per capita adjustments

```r
global_economy |>
  filter(Country == "Sri Lanka") |>
  autoplot(GDP / Population)
```



Mathematical transformations

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

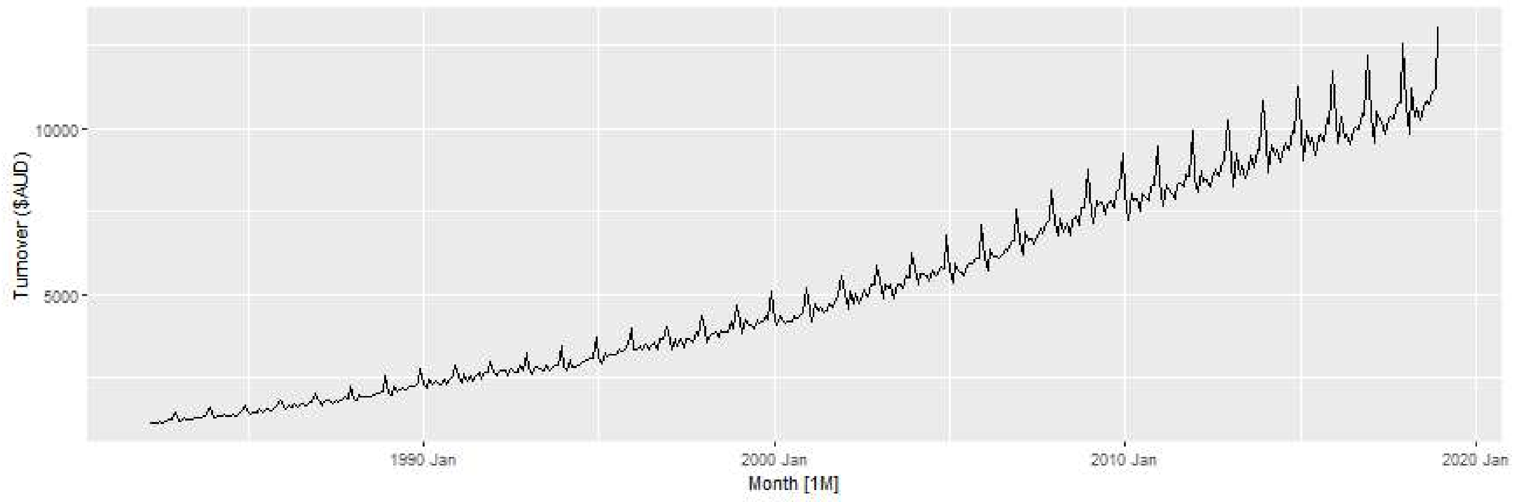
Transformation	**Equation**
Square root	$w_t = \sqrt{y_t}$
Cube root	$w_t = \sqrt[3]{y_t}$
Logarithm	$w_t = \log(y_t)$

- 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

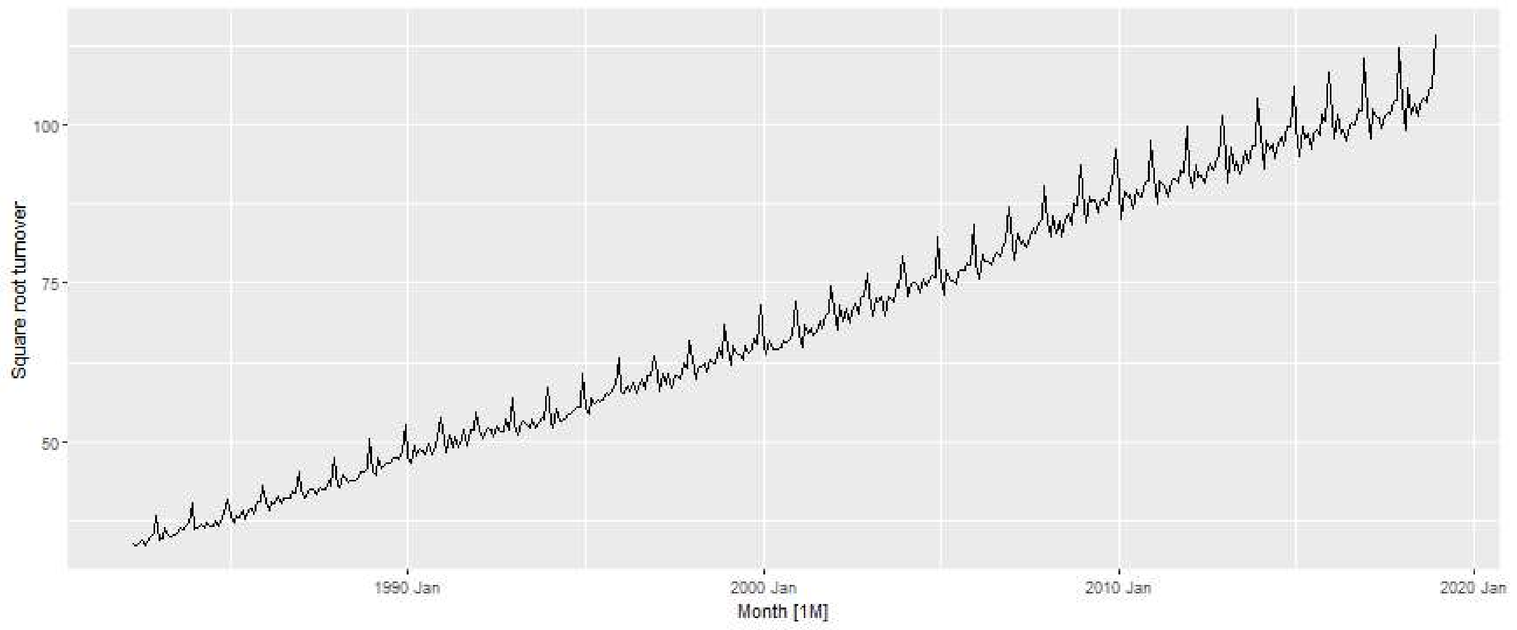
```r
food <- aus_retail |>
  filter(Industry == "Food retailing") |>
  summarise(Turnover = sum(Turnover))
```

```r
food |> autoplot(Turnover) +
  labs(y = "Turnover ($AUD)")
```



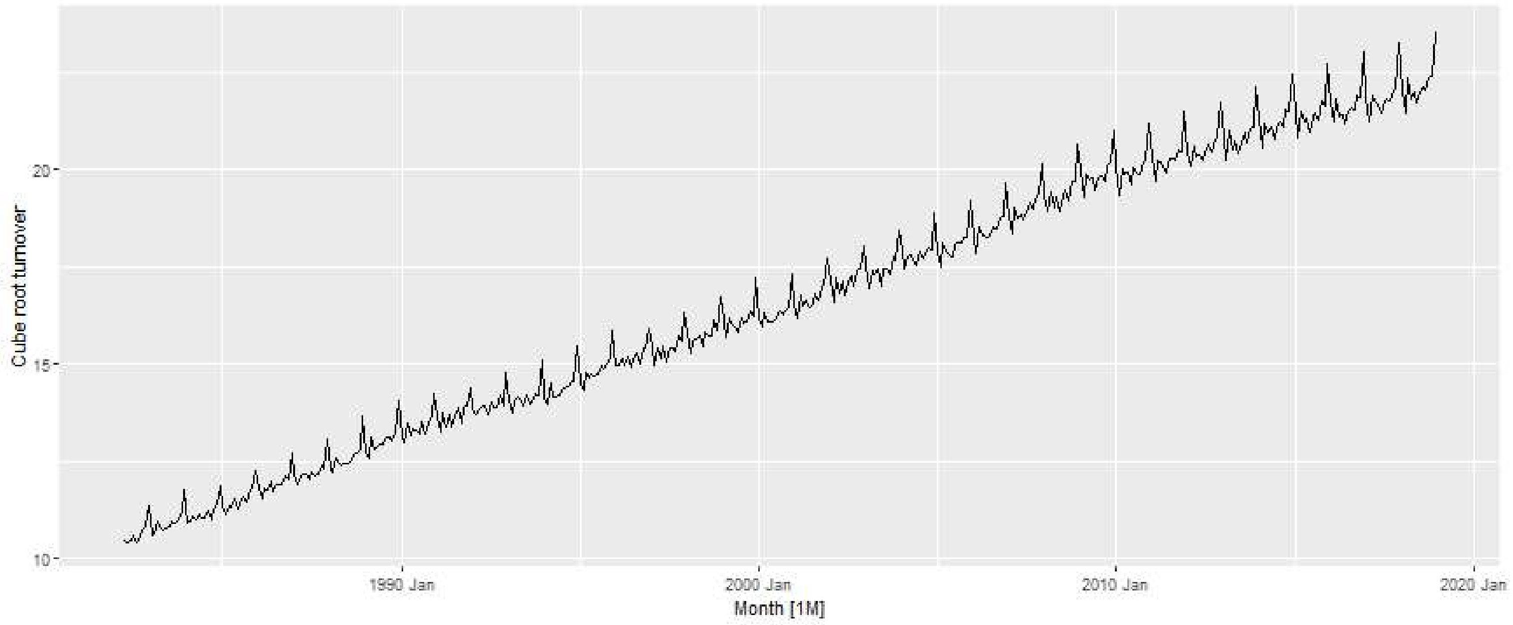
Mathematical transformations

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



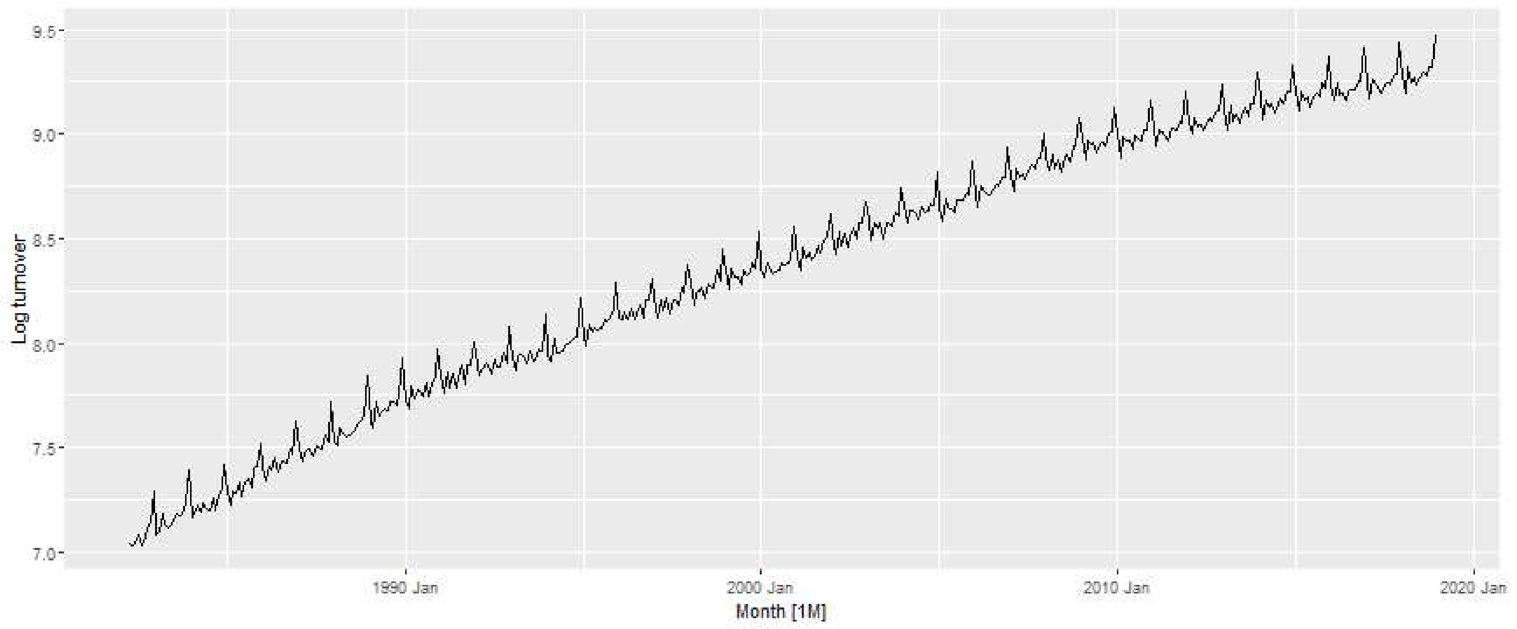
Mathematical transformations

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



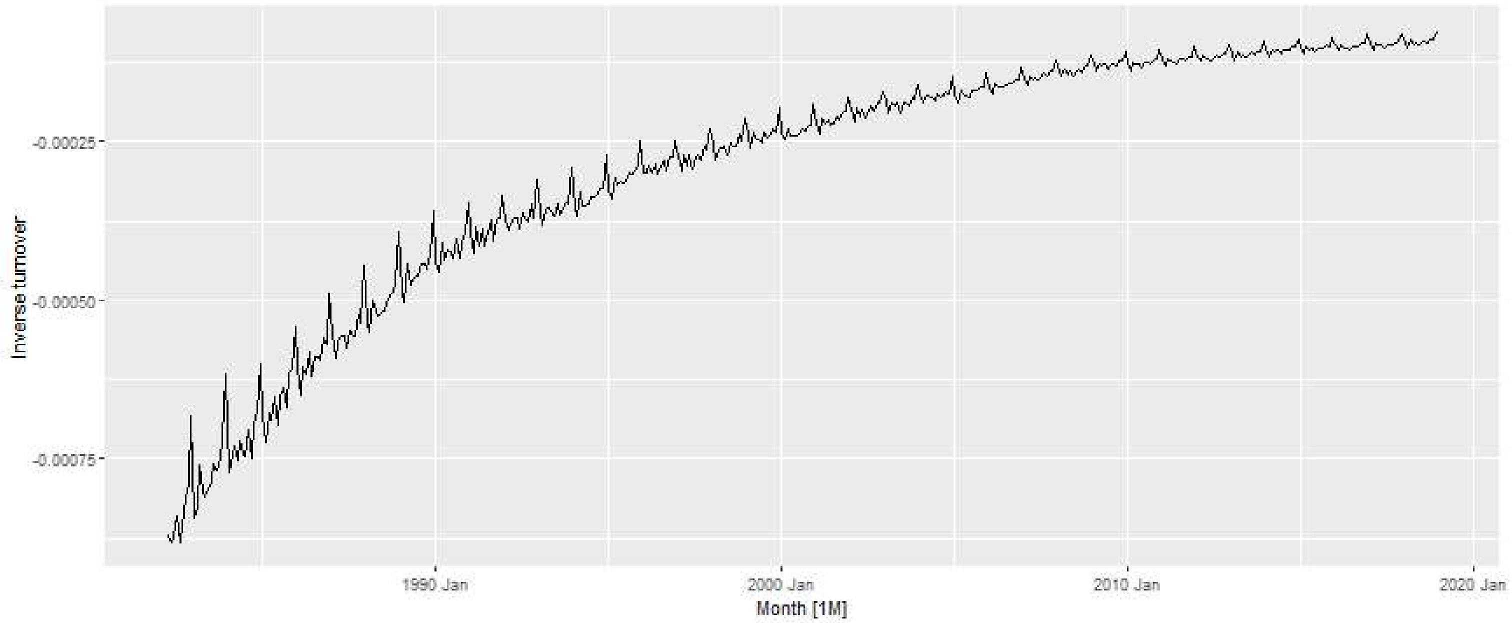
Mathematical transformations

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



Mathematical transformations

```r
food |> autoplot(-1/Turnover) +
  labs(y = "Inverse turnover")
```



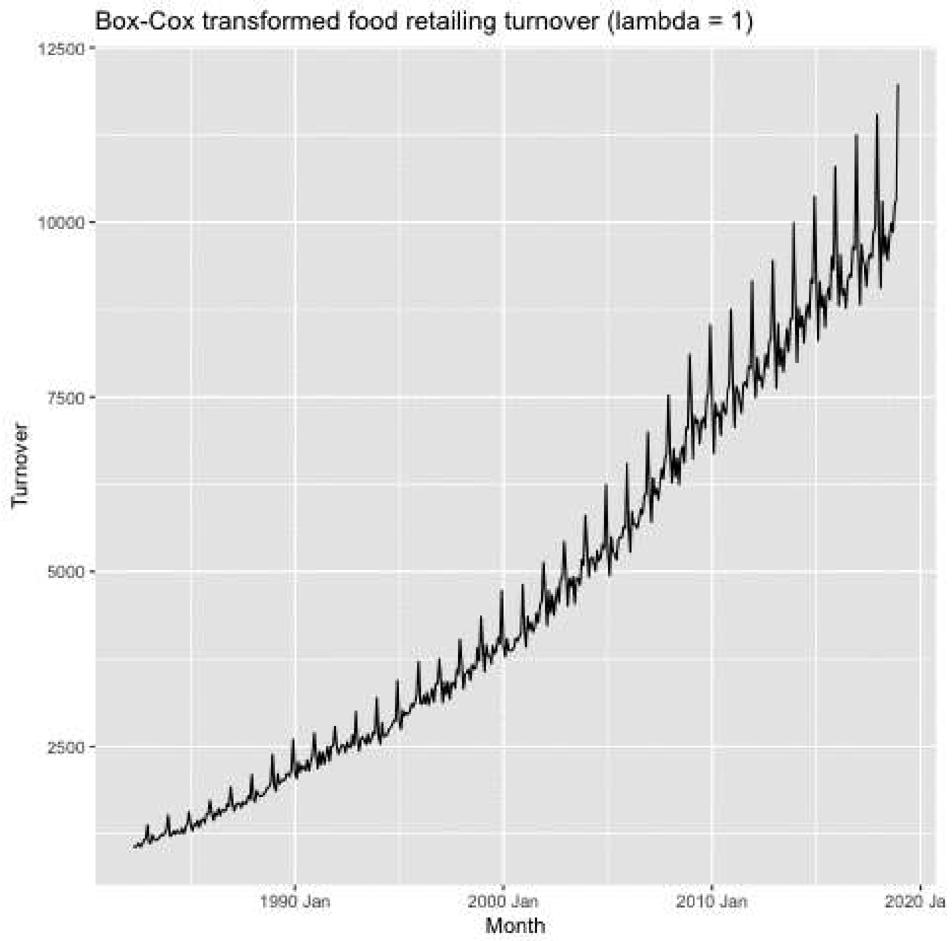
Box-Cox transformations

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

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

- $\lambda = 1$: (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

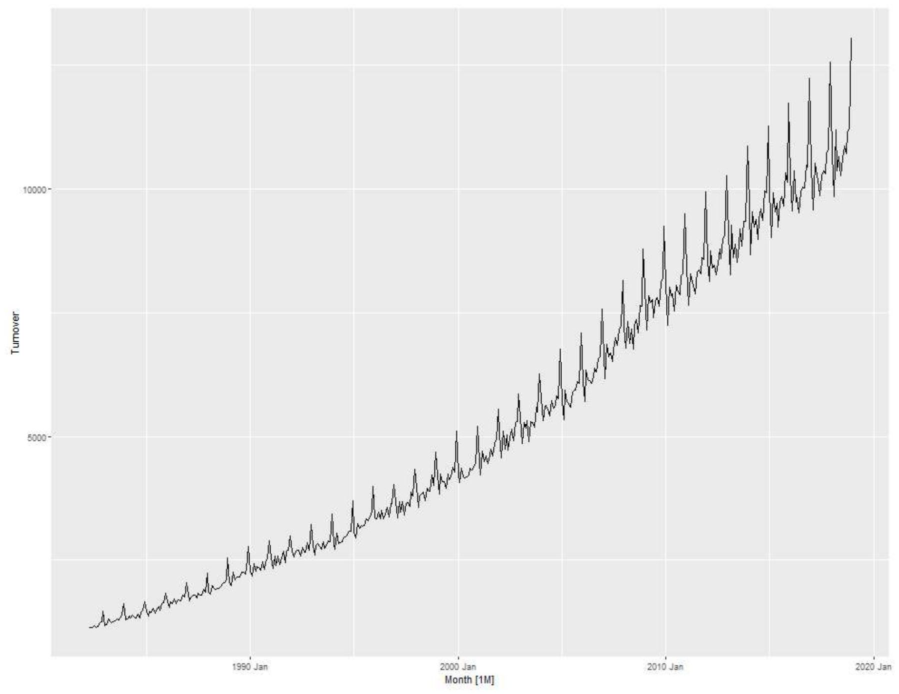
```
food |>
  features(Turnover, features = guerrero)
```

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

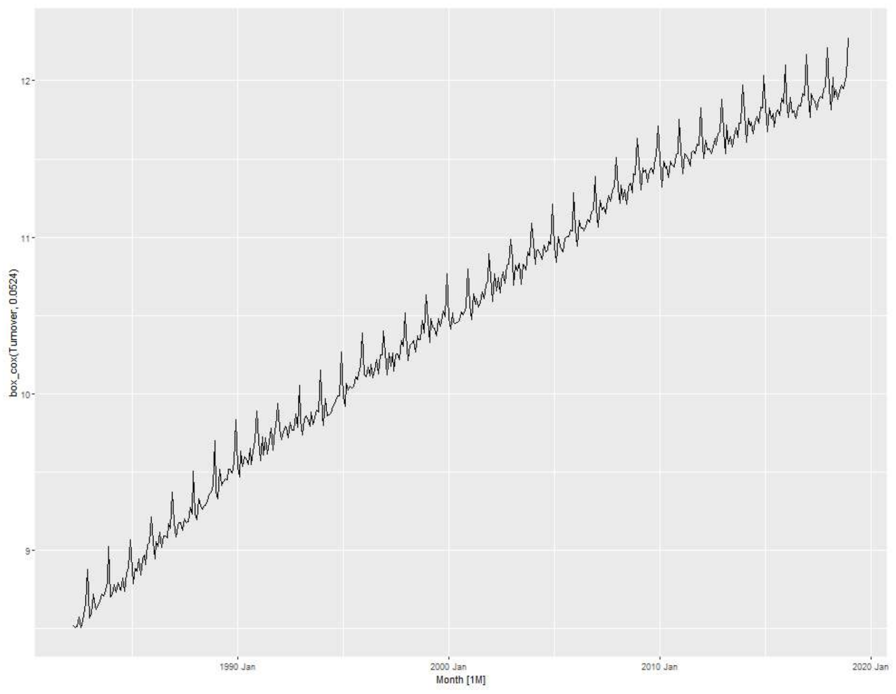
- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

Box-Cox transformations

```r
food |>
  autoplot(Turnover)
```



```r
food |>
autoplot(box_cox(Turnover, 0.0524))
```



Transformations

- Often no transformation needed.
- Simple transformations are easier to explain and work well enough.
- Transformations can have very large effect on PI.
- If the data contains zeros, then don't take logs.
- `log1p()` can be useful for data with zeros.
- If some data are negative, no power transformation is possible unless a constant is added to all values.
- 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`.)

Time series components

RECALL: Time series patterns

- **Trend** pattern exists when there is a long-term increase or decrease in the data.
- **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).
- **Cyclic** pattern exists when data exhibit rises and falls that are not of fixed frequency (duration usually of at least 2 years).

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 component at period t

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

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

Time series decomposition

- Additive model appropriate if magnitude of seasonal fluctuations does not vary with level.
- If seasonal are proportional to level of series, then multiplicative model appropriate.
- Multiplicative decomposition more prevalent with economic series
- Alternative: use a Box-Cox transformation, and then use additive decomposition.
- Logs turn multiplicative relationship into an additive relationship:

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

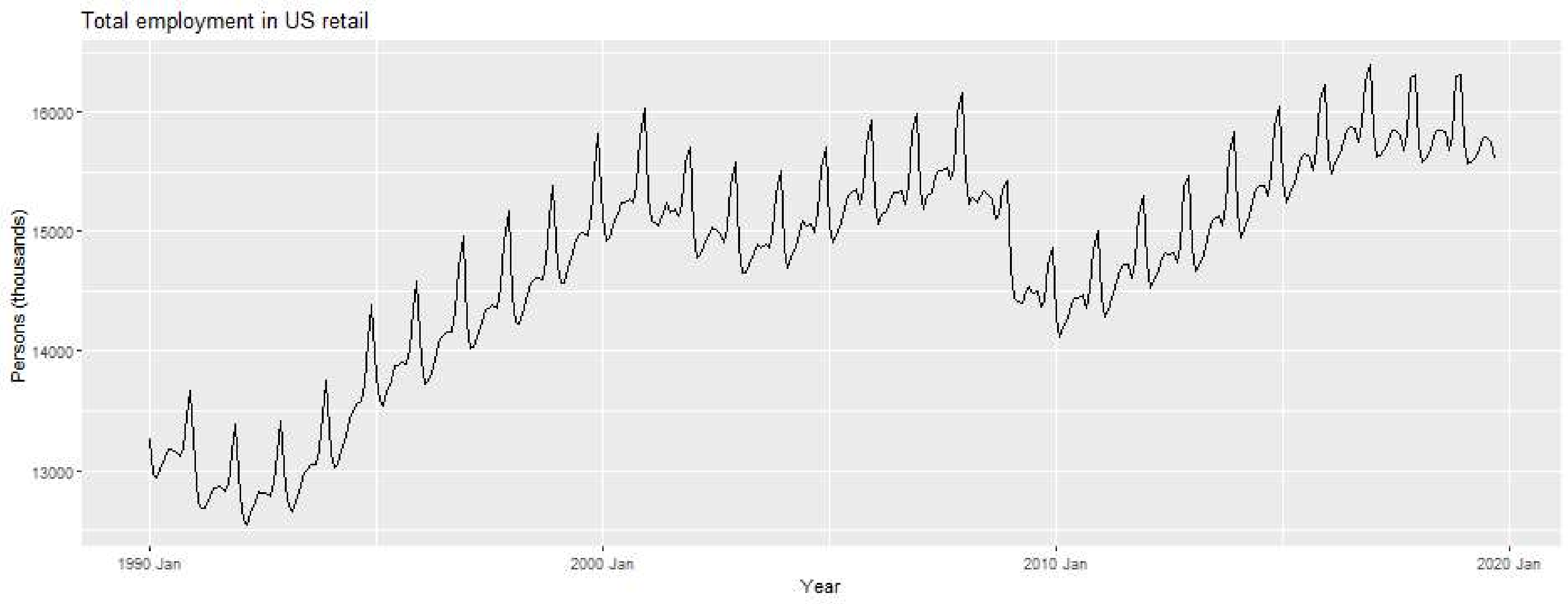
US Retail Employment

```r
library(fpp3)
us_retail_employment <- us_employment |>
  filter(year(Month) >= 1990, Title == "Retail Trade") |>
  select(-Series_ID)
us_retail_employment
```

```
## # A tsibble: 357 x 3 [1M]
##        Month Title        Employed
##        <mth> <chr>           <dbl>
##  1 1990 Jan Retail Trade   13256.
##  2 1990 Feb Retail Trade   12966.
##  3 1990 Mar Retail Trade   12938.
##  4 1990 Apr Retail Trade   13012.
##  5 1990 May Retail Trade   13108.
##  6 1990 Jun Retail Trade   13183.
##  7 1990 Jul Retail Trade   13170.
##  8 1990 Aug Retail Trade   13160.
##  9 1990 Sep Retail Trade   13113.
## 10 1990 Oct Retail Trade   13185.
## # ℹ 347 more rows
```

US Retail Employment

```
us_retail_employment |>
  autoplot(Employed) +
  xlab("Year") + ylab("Persons (thousands)") +
  ggtitle("Total employment in US retail")
```



US Retail Employment

```r
us_retail_employment |>
  model(stl = STL(Employed))
```

```
## # A mable: 1 x 1
##       stl
##   <model>
## 1   <STL>
```

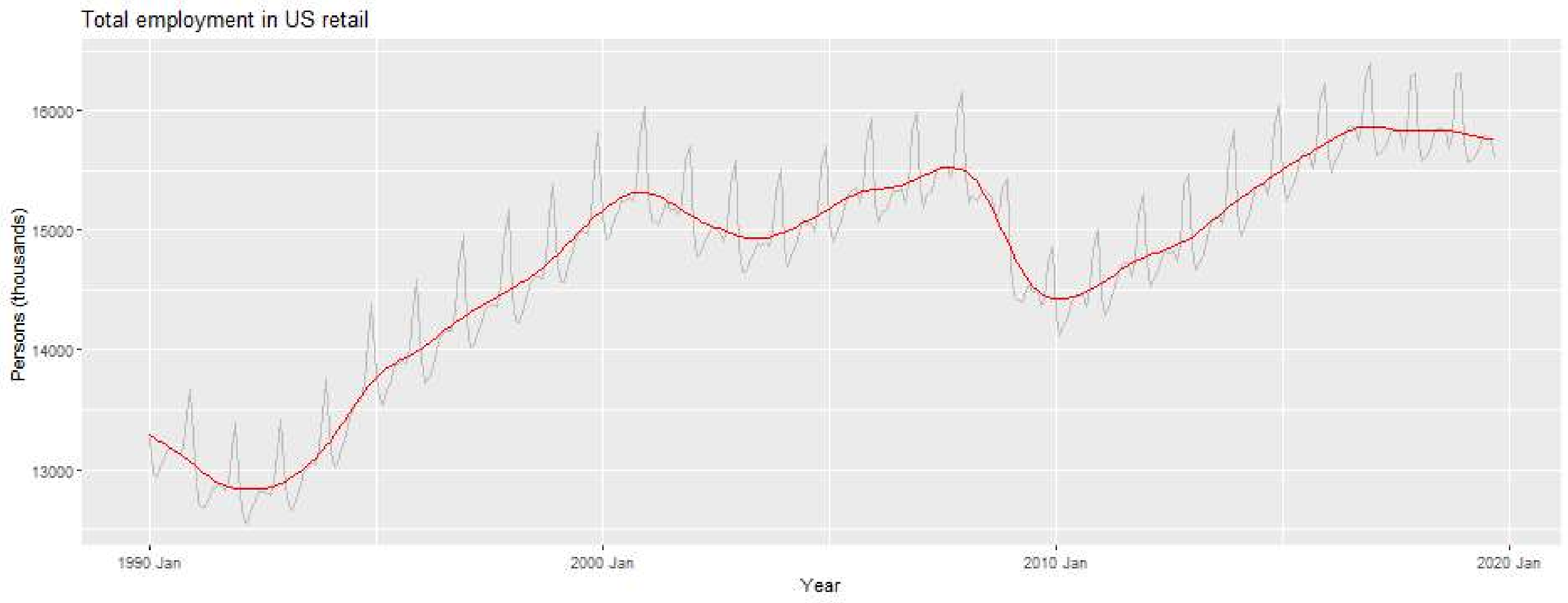
US Retail Employment

```r
dcmp <- us_retail_employment |>
  model(stl = STL(Employed))
components(dcmp)
```

```
## # A dable: 357 x 7 [1M]
## # Key:     .model [1]
## # :        Employed = trend + season_year + remainder
##    .model    Month Employed  trend season_year remainder season_adjust
##    <chr>     <mth>    <dbl>  <dbl>       <dbl>     <dbl>         <dbl>
##  1 stl    1990 Jan   13256. 13288.      -33.0      0.836        13289.
##  2 stl    1990 Feb   12966. 13269.     -258.     -44.6          13224.
##  3 stl    1990 Mar   12938. 13250.     -290.     -22.1          13228.
##  4 stl    1990 Apr   13012. 13231.     -220.       1.05         13232.
##  5 stl    1990 May   13108. 13211.     -114.      11.3          13223.
##  6 stl    1990 Jun   13183. 13192.      -24.3     15.5          13207.
##  7 stl    1990 Jul   13170. 13172.      -23.2     21.6          13193.
##  8 stl    1990 Aug   13160. 13151.       -9.52    17.8          13169.
##  9 stl    1990 Sep   13113. 13131.      -39.5     22.0          13153.
## 10 stl    1990 Oct   13185. 13110.       61.6     13.2          13124.
## # ℹ 347 more rows
```

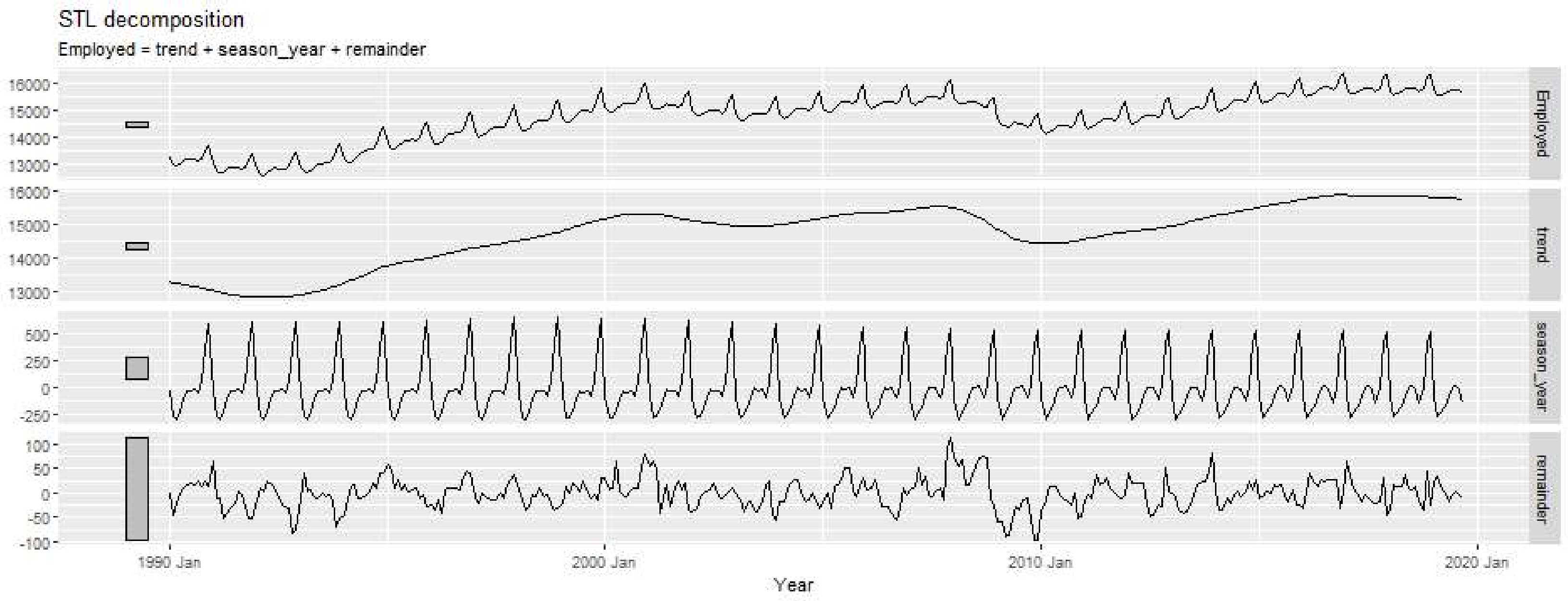
US Retail Employment

```
us_retail_employment |>
  autoplot(Employed, color='gray') +
  autolayer(components(dcmp), trend, color='red') +
  xlab("Year") + ylab("Persons (thousands)") +
  ggtitle("Total employment in US retail")
```



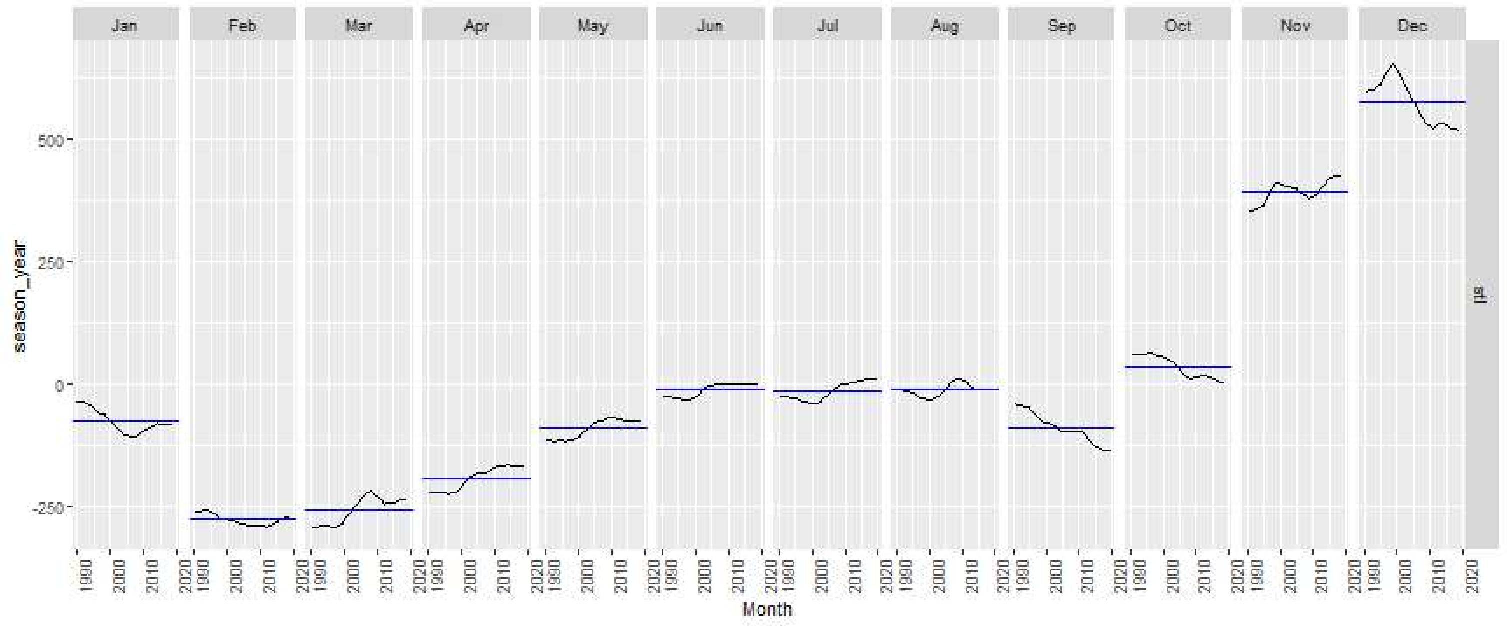
US Retail Employment

```r
components(dcmp) |> autoplot() + xlab("Year")
```



US Retail Employment

```
components(dcmp) |> gg_subseries(season_year)
```



Seasonal adjustment

- Useful by-product of decomposition: an easy way to calculate seasonally adjusted data.
- Additive decomposition: seasonally adjusted data given by

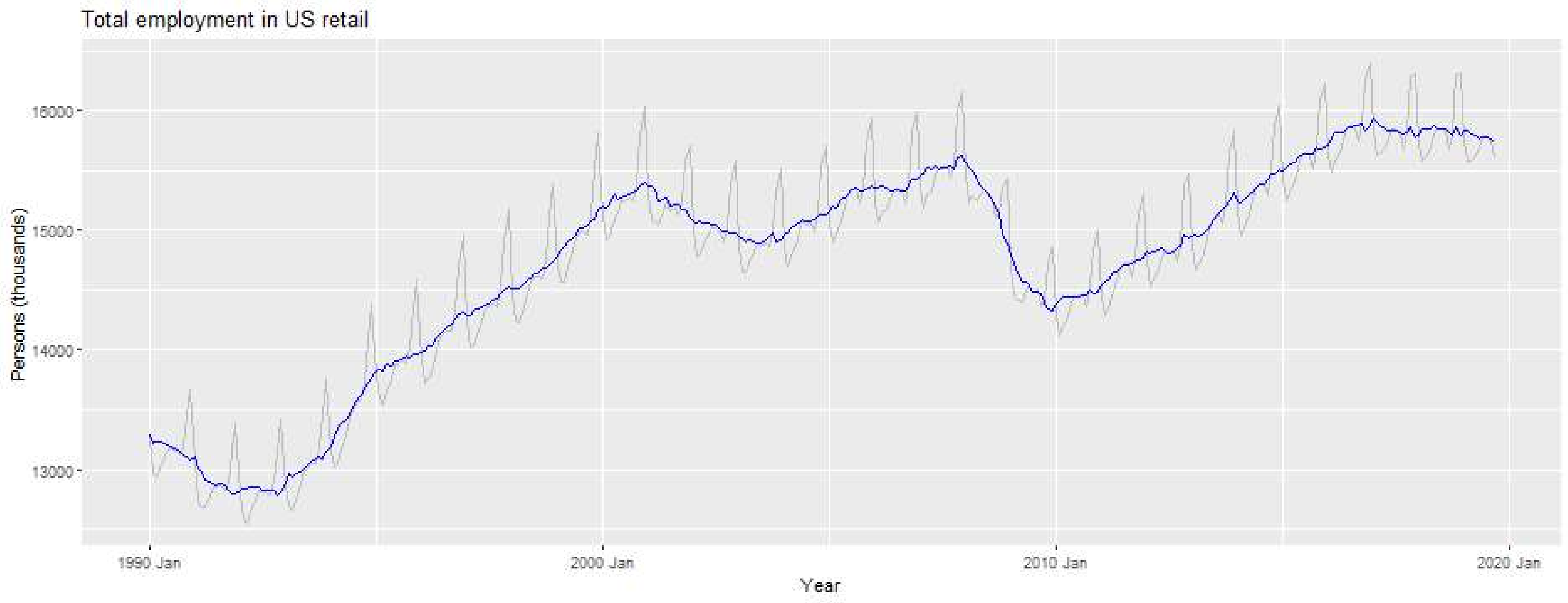
$$y_t - S_t = T_t + R_t$$

- Multiplicative decomposition: seasonally adjusted data given by

$$y_t / S_t = T_t \times R_t$$

US Retail Employment

```
us_retail_employment |>
  autoplot(Employed, color='gray') +
  autolayer(components(dcmp), season_adjust, color='blue') +
  xlab("Year") + ylab("Persons (thousands)") +
  ggtitle("Total employment in US retail")
```



Seasonal adjustment

- We use estimates of S based on past values to seasonally adjust a current value.
- Seasonally adjusted series reflect **remainders** as well as **trend**. Therefore they are not "smooth" and "downturns" or "upturns" can be misleading.
- It is better to use the trend-cycle component to look for turning points.

Moving Averages

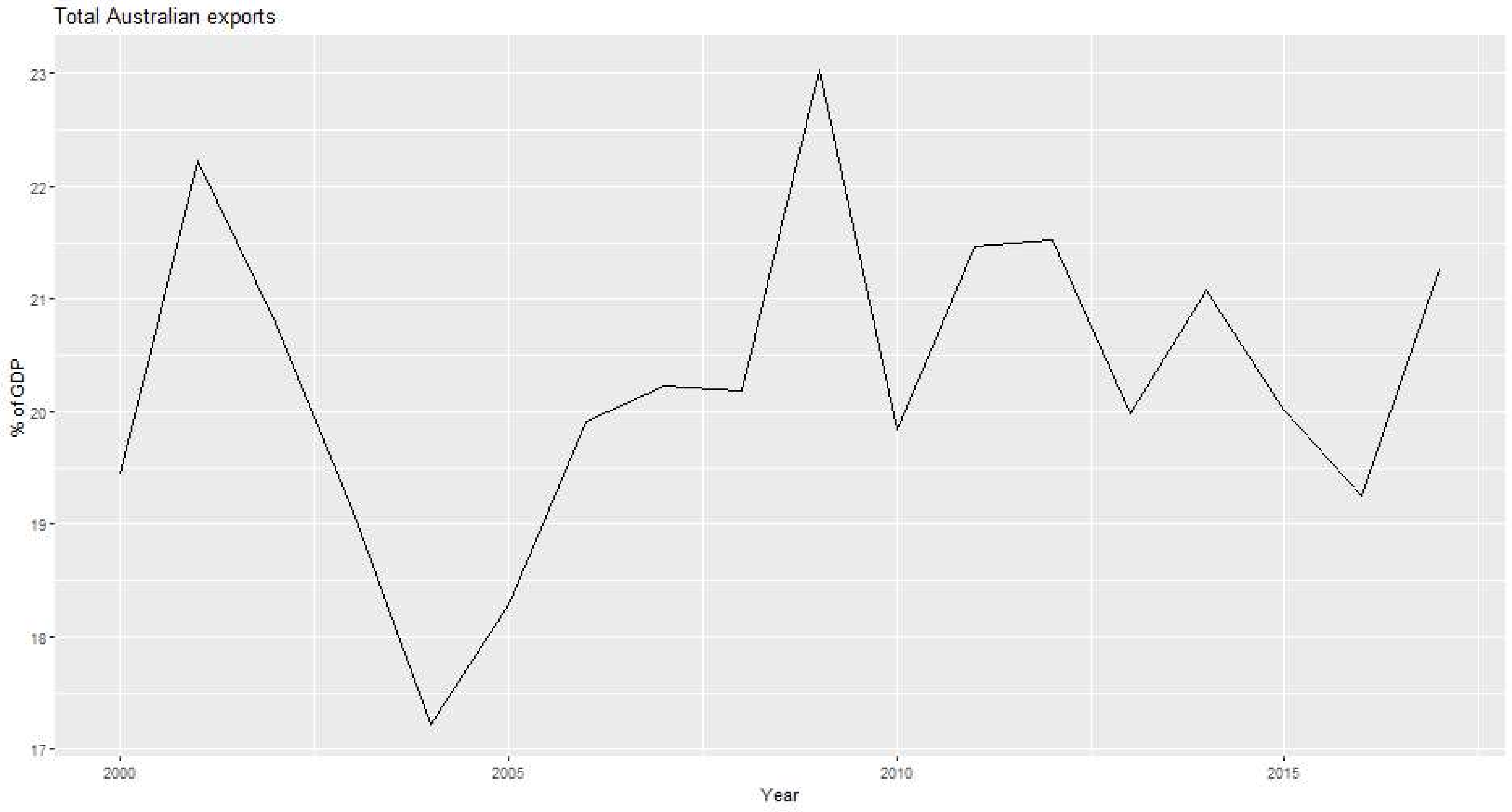
- The simplest estimate of the trend-cycle uses **moving averages**.

m-MA

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

where $m = 2k + 1$

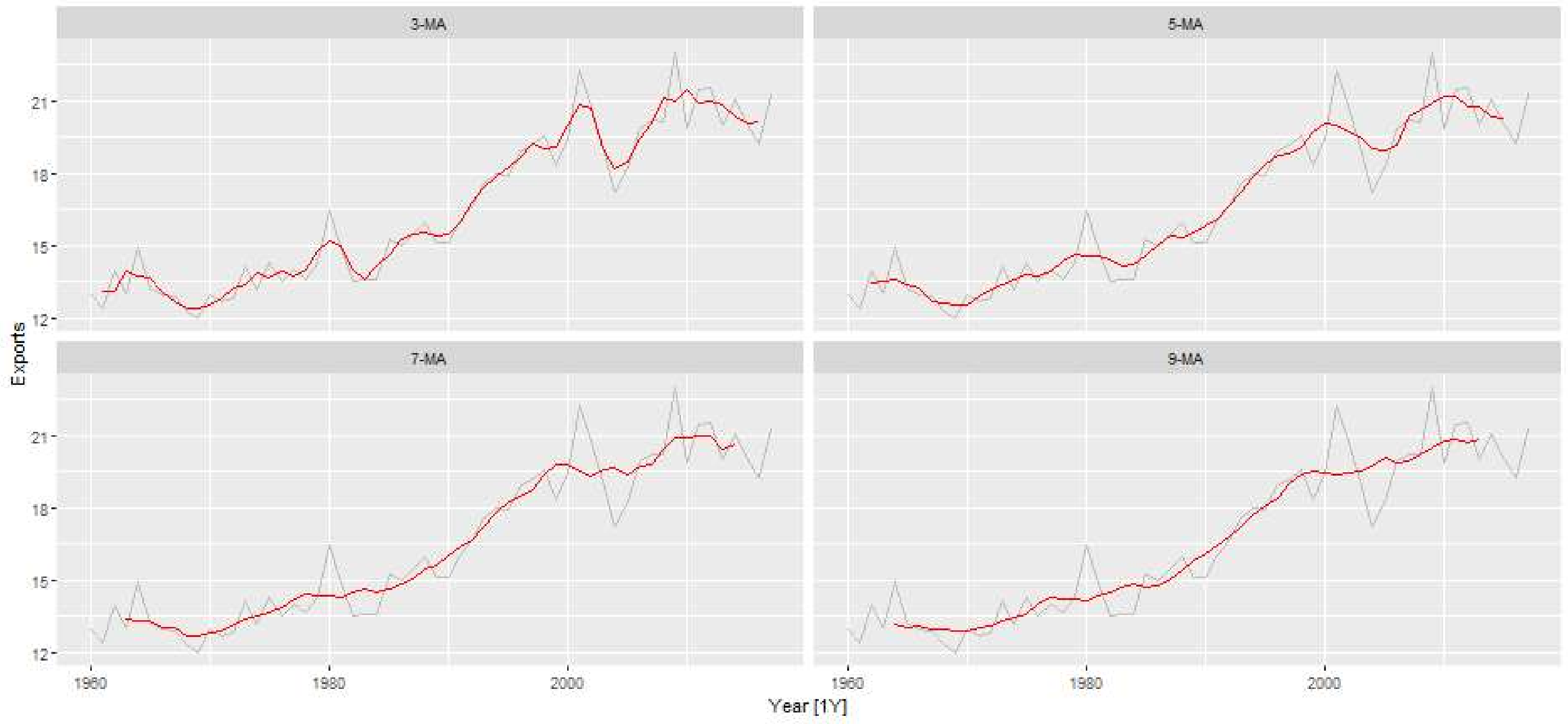
Moving averages



Moving averages

```
## # A tsibble: 18 x 5 [1Y]
##     Year Exports `3-MA` `5-MA` `7-MA`
##    <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
##  1  2000    19.4   NA     NA     NA
##  2  2001    22.2   20.8   NA     NA
##  3  2002    20.8   20.7   19.8   NA
##  4  2003    19.1   19.0   19.5   19.6
##  5  2004    17.2   18.2   19.1   19.7
##  6  2005    18.3   18.5   18.9   19.4
##  7  2006    19.9   19.5   19.2   19.7
##  8  2007    20.2   20.1   20.3   19.8
##  9  2008    20.2   21.2   20.6   20.4
## 10  2009    23.0   21.0   21.0   20.9
## 11  2010    19.8   21.5   21.2   20.9
## 12  2011    21.5   20.9   21.2   21.0
## 13  2012    21.5   21.0   20.8   21.0
## 14  2013    20.0   20.9   20.8   20.5
## 15  2014    21.1   20.4   20.4   20.7
## 16  2015    20.0   20.1   20.3   NA
## 17  2016    19.3   20.2   NA     NA
## 18  2017    21.3   NA     NA     NA
```

Moving averages



Moving averages

- So a moving average is an **average of nearby points**
- observations nearby in time are also likely to be **close in value**.
- average eliminates some **randomness** in the data, leaving a **smooth trend-cycle** component.

$$3 - MA : \hat{T}_t = \frac{(y_{t-1} + y_t + y_{t+1})}{3}$$

$$5 - MA : \hat{T}_t = \frac{(y_{t-2} + y_{t-1} + y_t + y_{t+1} + y_{t+2})}{5}$$

- each average computed by dropping **oldest** observation and including **next** observation.
- averaging **moves** through time series until trend-cycle computed at each observation possible

Endpoints

Why is there no estimate at ends

- For a 3-MA, there cannot be estimates at time 1 or time n because the observations at time 0 and $n+1$ are not available.
- Generally: there cannot be estimates at times near the endpoints.

The order of the MA

- larger order means smoother, flatter curve
- larger order means more points lost at ends

Moving averages of moving averages

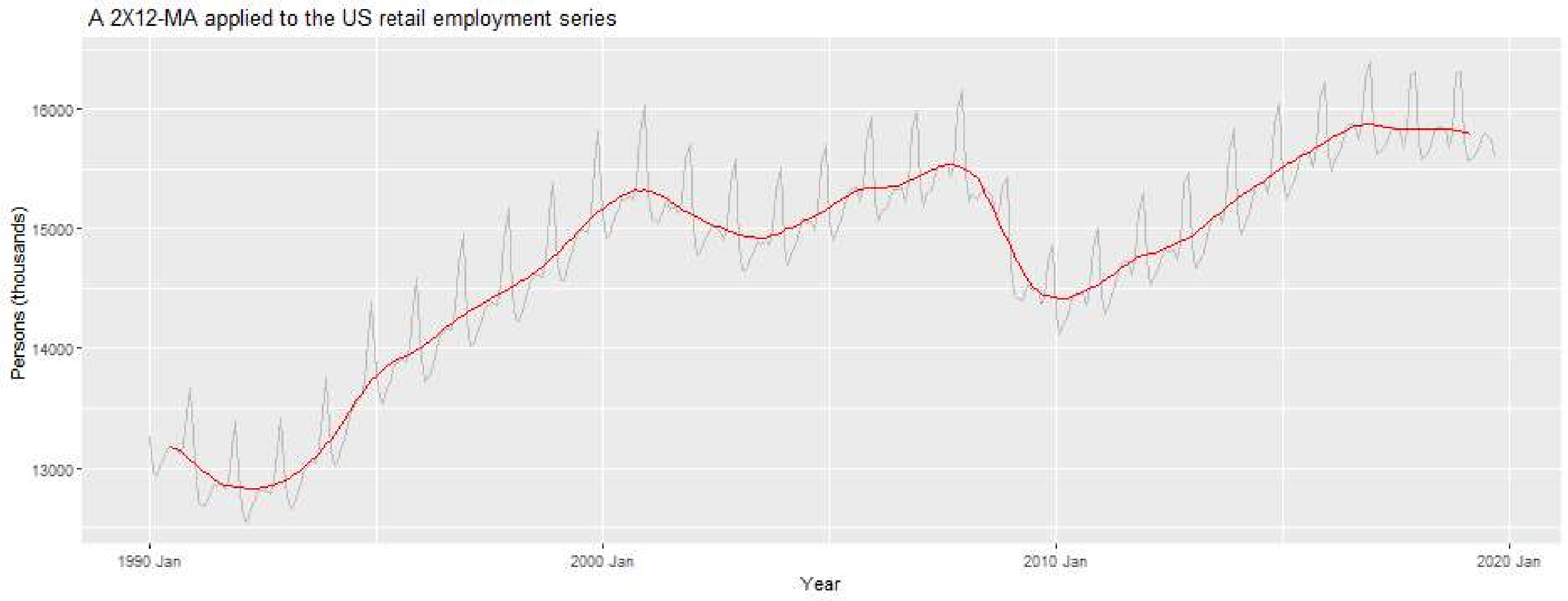
- Centered MA

Moving averages

```
## # A tsibble: 18 x 4 [1Y]
##     Year Exports `4-MA` `2*4-MA`
##    <dbl>   <dbl>  <dbl>    <dbl>
##  1  2000    19.4   NA       NA
##  2  2001    22.2   20.4     NA
##  3  2002    20.8   19.8     20.1
##  4  2003    19.1   18.8     19.3
##  5  2004    17.2   18.6     18.7
##  6  2005    18.3   18.9     18.8
##  7  2006    19.9   19.7     19.3
##  8  2007    20.2   20.8     20.2
##  9  2008    20.2   20.8     20.8
## 10  2009    23.0   21.1     21.0
## 11  2010    19.8   21.5     21.3
## 12  2011    21.5   20.7     21.1
## 13  2012    21.5   21.0     20.9
## 14  2013    20.0   20.6     20.8
```

Estimating the trend-cycle with seasonal data

- A moving average of the same length as the season removes the seasonal pattern.
- For quarterly data: use a $2 \times 4MA$
- For monthly data: use a $2 \times 12MA$



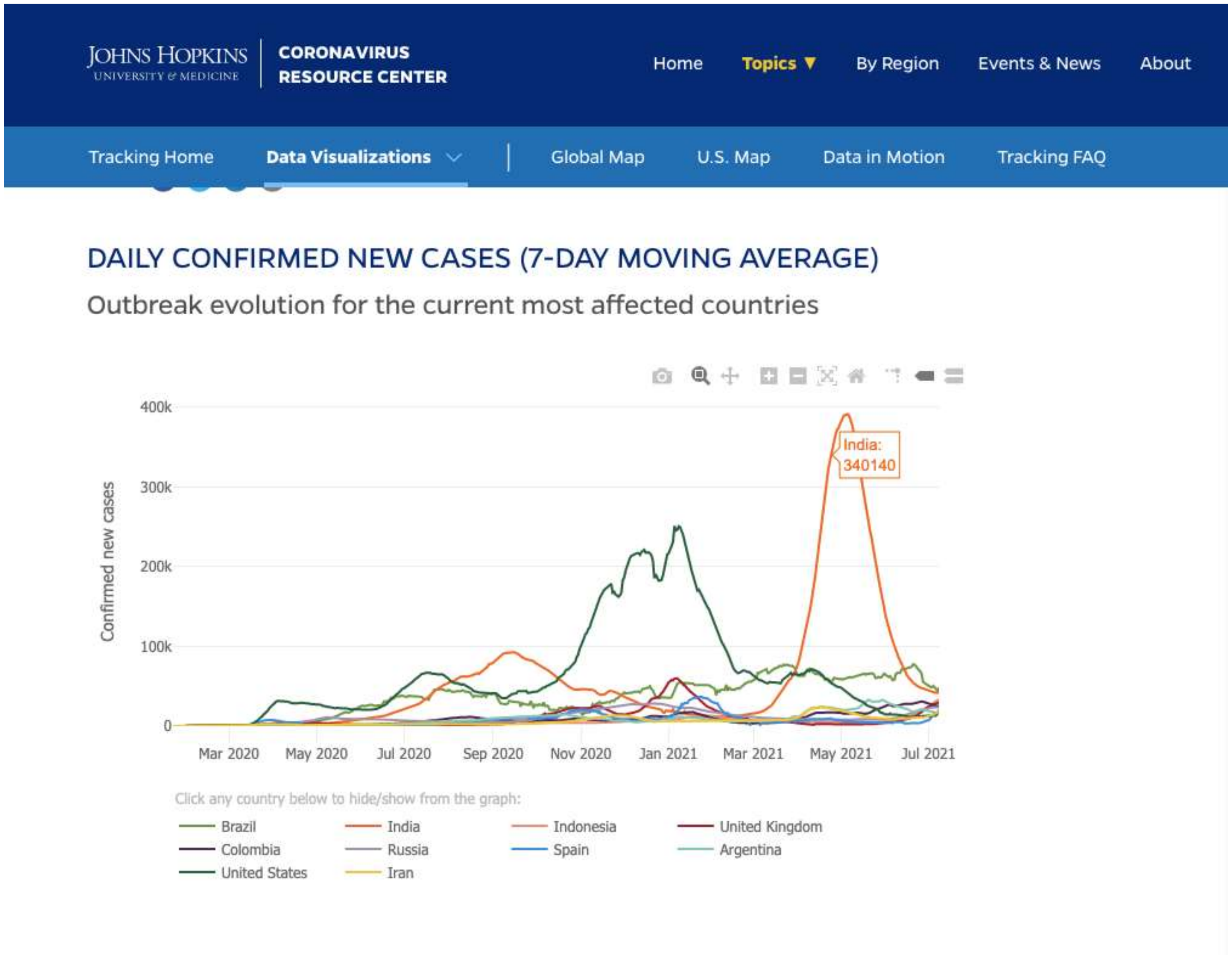
Weighted moving averages

Weighted MA

$$T_t = \sum_{j=-k}^{k} a_j y_{t+j}$$

where $k = (m-1)/2$ is the half-width and the weights are denoted by $[a_{-k}, \dots, a_k]$.

- Simple m-MA: all weights equal to $1/m$.
- Require sum of $a_j = 1$ and $a_j = a_{-j}$.
- Weighted MA are smoother.



https://coronavirus.jhu.edu/data/new-cases

Trend-cycle

- Multiplicative decomposition: $y_t = T_t \times S_t \times R_t = \hat{T}_t \times \hat{S}_t \times \hat{R}_t$
- Additive decomposition: $y_t = T_t + S_t + R_t = \hat{T}_t + \hat{S}_t + \hat{R}_t$
- Estimate $\hat{T}$ using $(2 \times m)$-MA if m is even. Otherwise, estimate $\hat{T}$ using m-MA
- Compute de-trended series
  - Multiplicative decomposition: $y_t / \hat{T}_t$
  - Additive decomposition: $y_t - \hat{T}_t$

De-trending

Remove smoothed series $\hat{T}_t$ from y_t to leave S_t and R_t.

- Multiplicative model: $\frac{y_t}{\hat{T}_t} = \frac{\hat{T}_t \times \hat{S}_t \times \hat{R}_t}{\hat{T}_t} = \hat{S}_t \times \hat{R}_t$
- Additive model: $y_t - \hat{T}_t = (\hat{T}_t + \hat{S}_t + \hat{R}_t) - \hat{T}_t = \hat{S}_t + \hat{R}_t$

Classical decomposition

- Choose additive or multiplicative depending on which gives the most stable components.
- Estimate of trend is unavailable for first few and last few observations.
- Seasonal component repeats from year to year. May not be realistic.
- Not robust to outliers.
- Newer methods designed to overcome these problems.

History of time series decomposition

History of time series decomposition

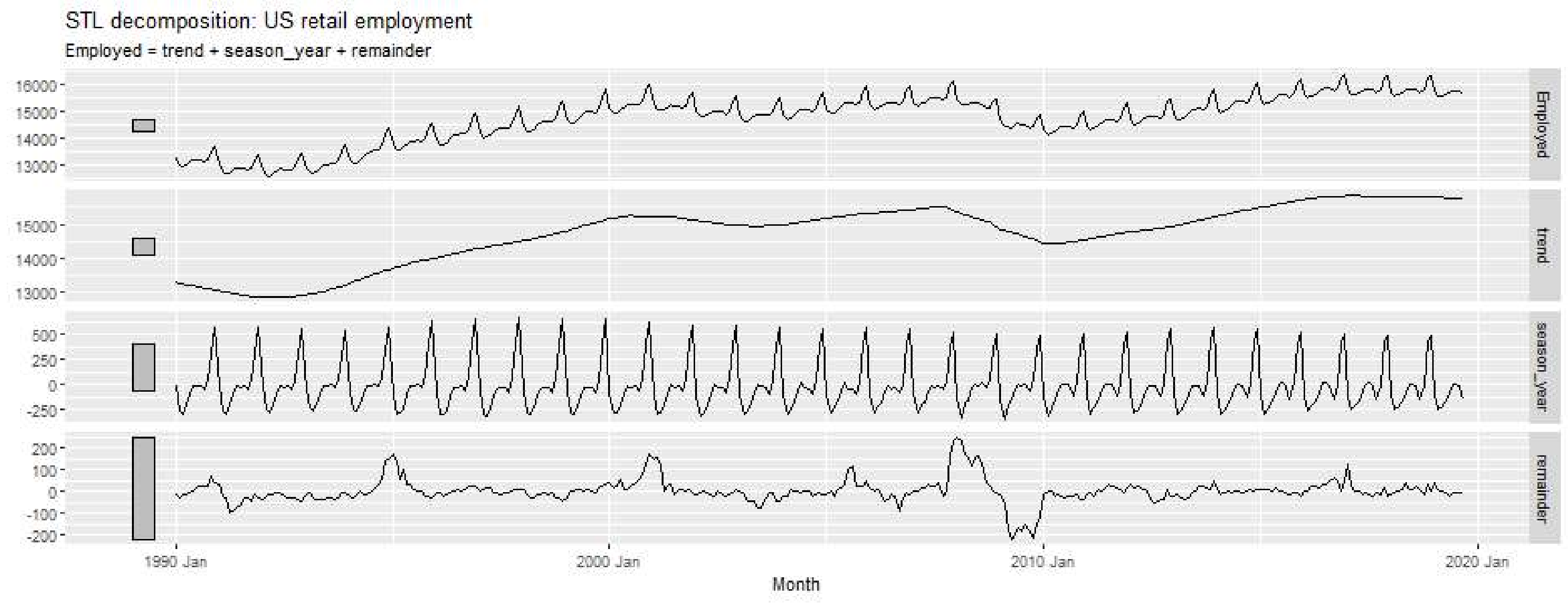
- Classical method originated in 1920s.
- Census II method introduced in 1957. Basis for X-11 method and variants (including X-12-ARIMA, X-13-ARIMA)
- STL method introduced in 1983
- TRAMO/SEATS introduced in 1990s.

STL decomposition

- STL: "Seasonal and Trend decomposition using Loess"
- Very versatile and robust.
- Unlike X-12-ARIMA, STL will handle any type of seasonality.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Robust to outliers
- Not trading day or calendar adjustments.
- Only additive.
- Take logs to get multiplicative decomposition.
- Use Box-Cox transformations to get other decompositions.

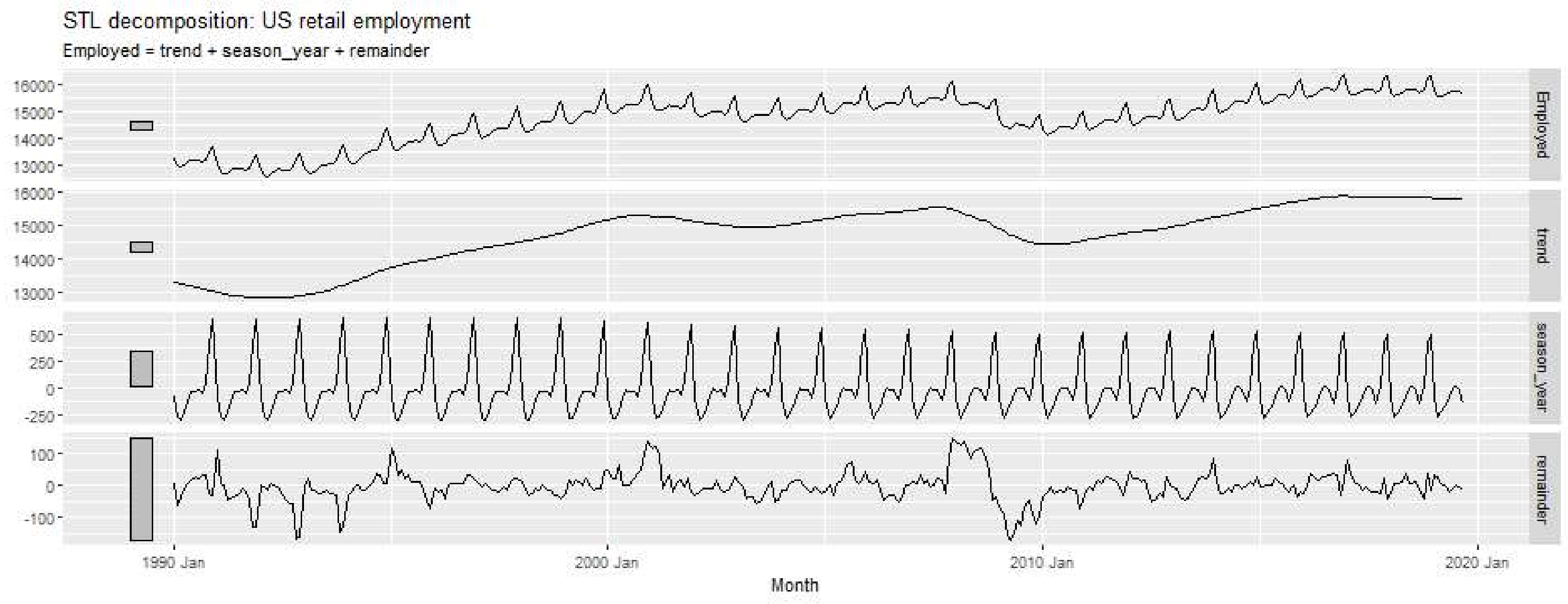
STL decomposition

```
us_retail_employment |>
  model(STL(Employed ~ season(window=5), robust=TRUE)) |>
  components() |> autoplot() +
    ggtitle("STL decomposition: US retail employment")
```



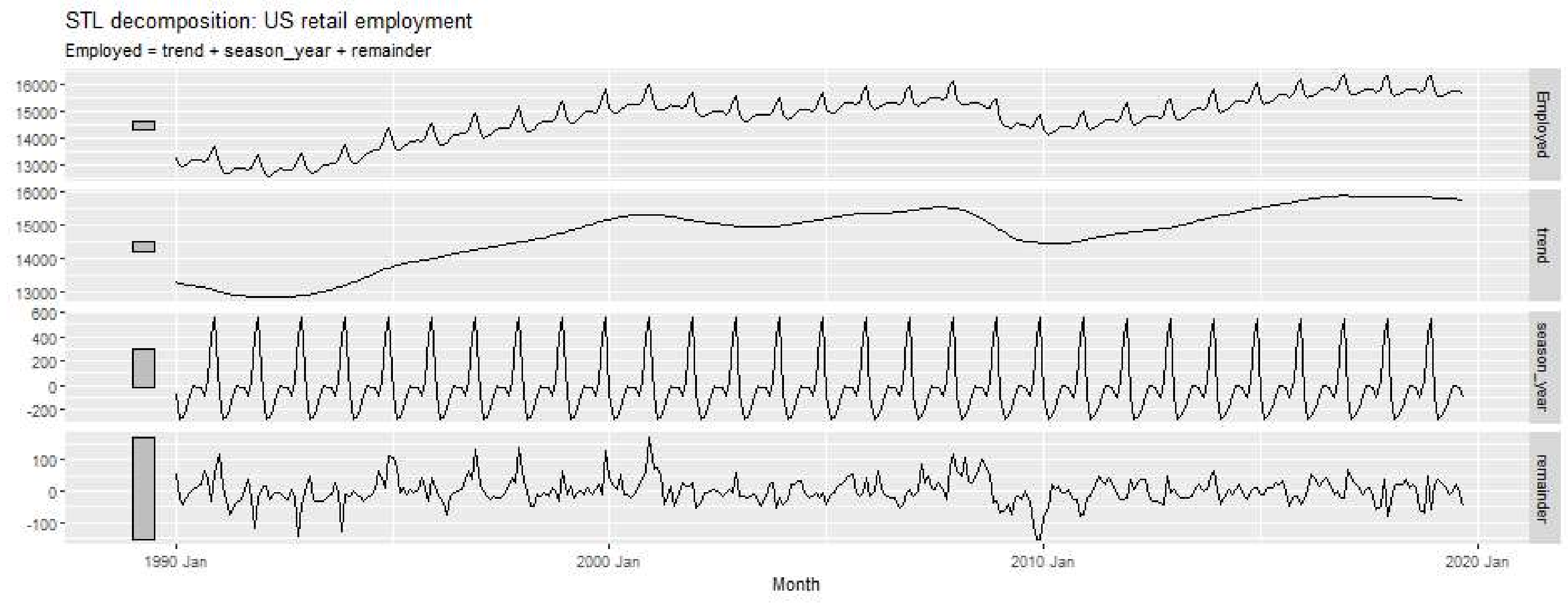
STL decomposition

```
us_retail_employment |>
  model(STL(Employed ~ season(window=9), robust=TRUE)) |>
  components() |> autoplot() +
    ggtitle("STL decomposition: US retail employment")
```



STL decomposition

```
us_retail_employment |>
  model(STL(Employed ~ season(window=55), robust=TRUE)) |>
  components() |> autoplot() +
    ggtitle("STL decomposition: US retail employment")
```



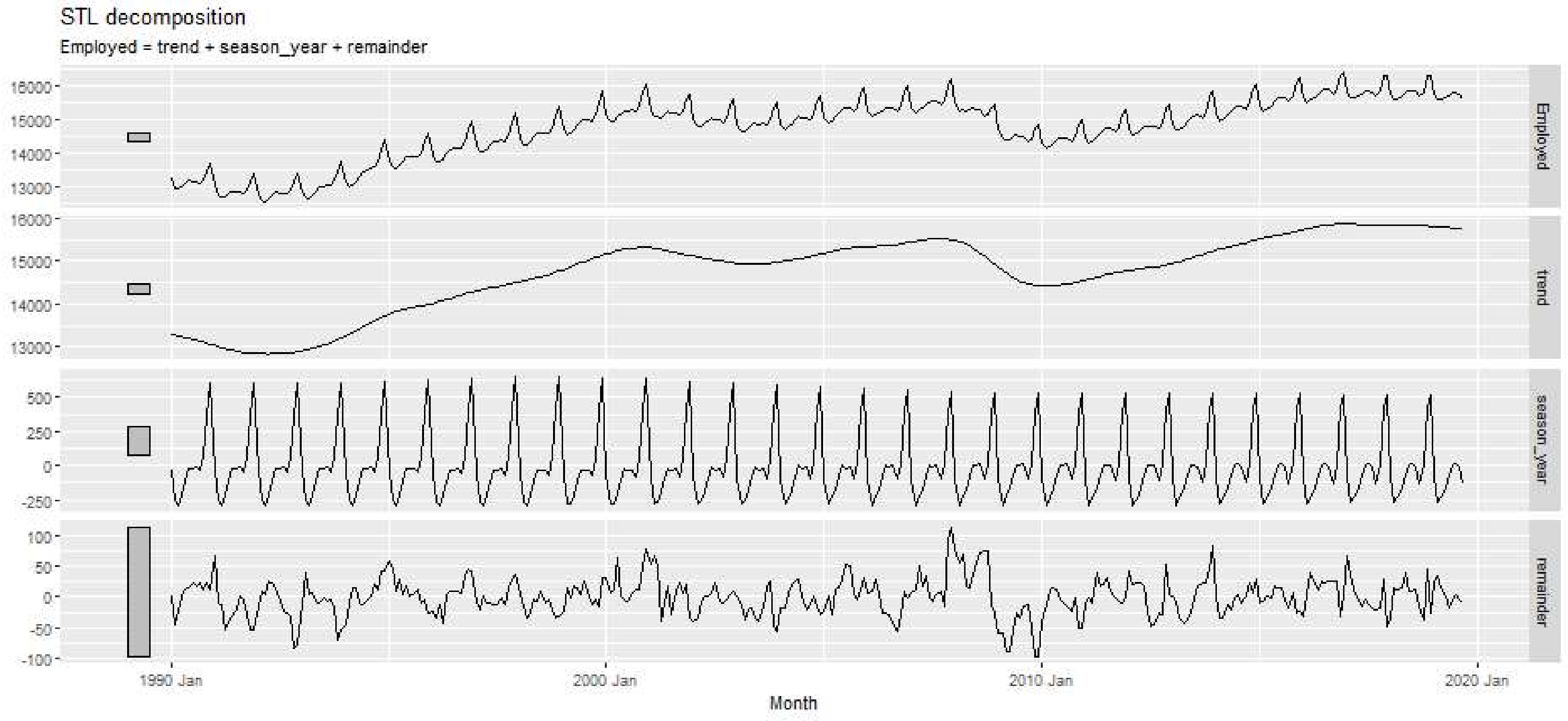
STL decomposition

```r
us_retail_employment |>
  model(STL(Employed ~ season(window=5))) |>
  components()

us_retail_employment |>
  model(STL(Employed ~ trend(window=15) +
                       season(window="periodic"),
            robust = TRUE)
  ) |> components()
```

- `trend(window = ?)` controls wiggliness of trend component.
- `season(window = ?)` controls variation on seasonal component.
- `season(window = 'periodic')` is equivalent to an infinite window.

STL decomposition



- `STL()` chooses `season(window=13)` by default
- Can include transformations.

STL decomposition

- Algorithm that updates trend and seasonal components iteratively.
- Starts with $\hat{T}_t = 0$
- Uses a mixture of loess and moving averages to successively refine the trend and seasonal estimates.
- The trend window controls loess bandwidth applied to deasonalised values.
- The season window controls loess bandwidth applied to detrended subseries.
- Default season `window = 13`
- Default trend `window = nextodd( ceiling((1.5*period)/(1-(1.5/s.window)))`

References

- Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.