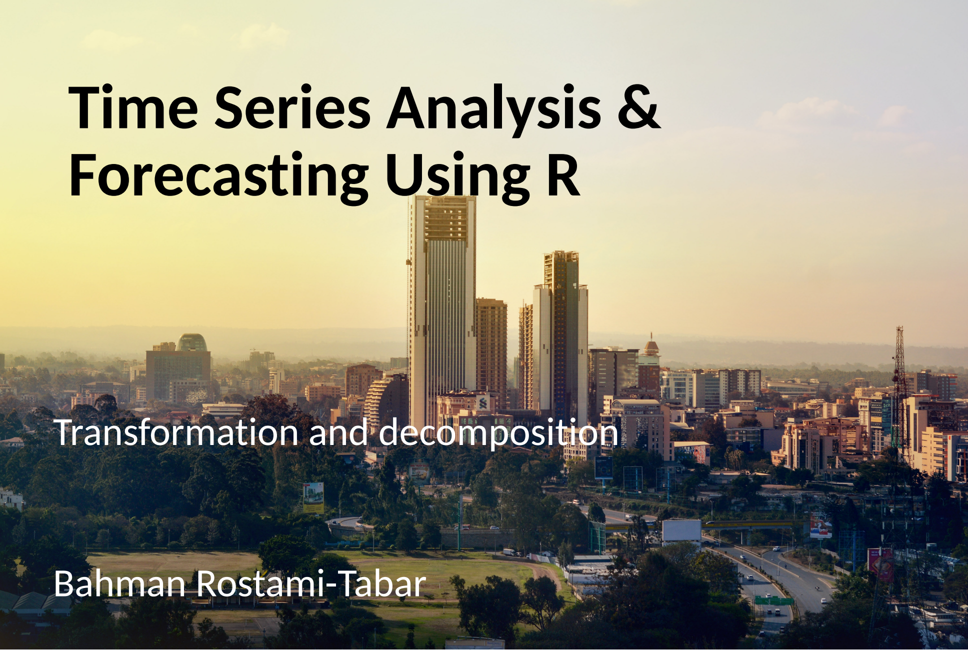


Time Series Analysis & Forecasting Using R

Transformation and decomposition

Bahman Rostami-Tabar

Outline

Outline

Learning outcome

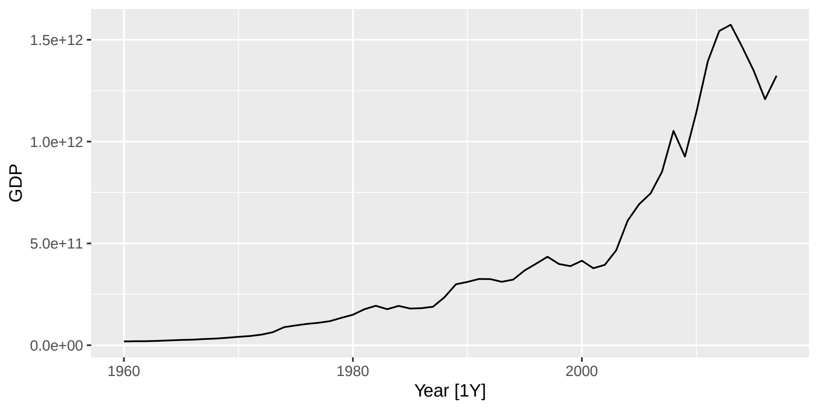
You should be able to:

1. Apply required adjustments on time series data before your analysis
2. Understand when time series decomposition would be helpful
3. Decompose time series using STL method

Outline

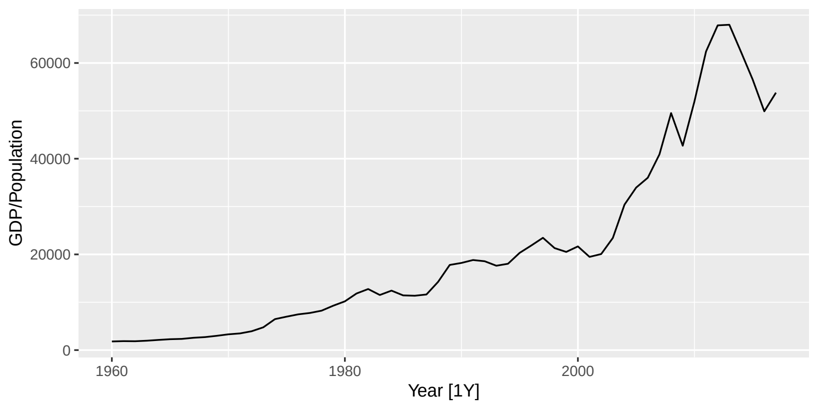
Per capita adjustments

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



Per capita adjustments

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

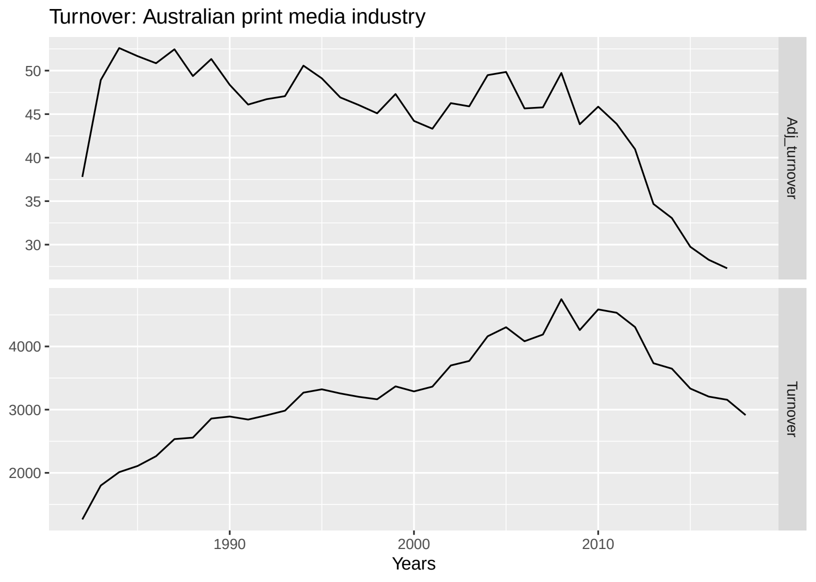


Outline

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 <- filter(global_economy, Code == "AUS")
print_retail |>
  left_join(aus_economy, by = "Year") |>
  mutate(Adj_turnover = Turnover / CPI) |>
  pivot_longer(c(Turnover, Adj_turnover),
    names_to = "Type", values_to = "Turnover"
  ) |>
  ggplot(aes(x = Year, y = Turnover)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(x = "Years", y = NULL,
       title = "Turnover: Australian print media industry")
```

Inflation adjustments



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 $w_1, \dots, w_n$.

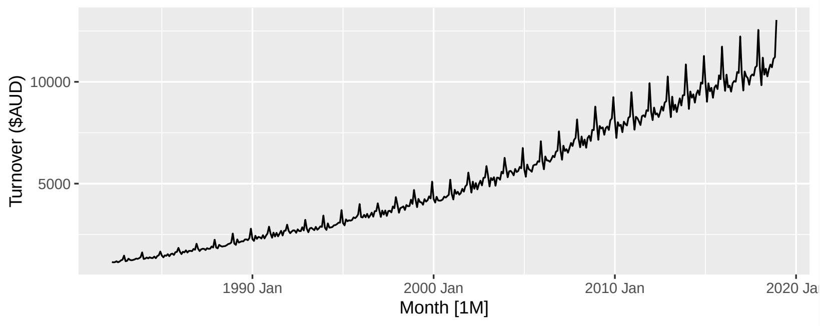
Mathematical transformations for stabilizing variation

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**.

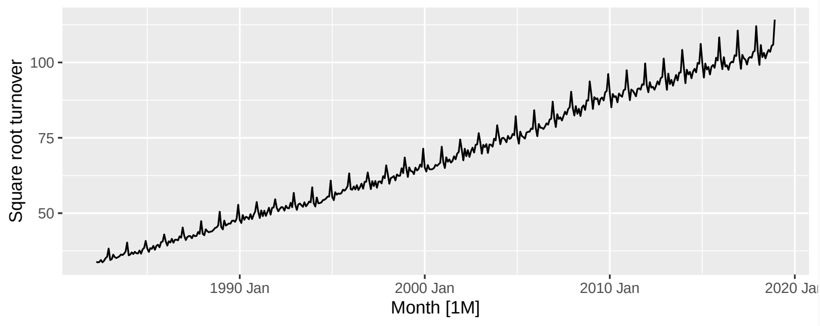
Variance stabilization

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



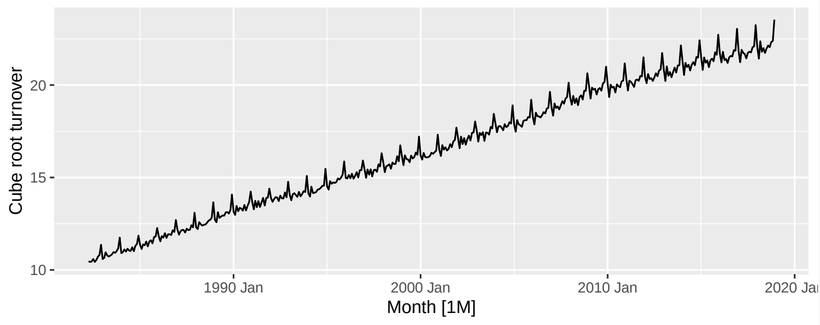
Variance stabilization

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



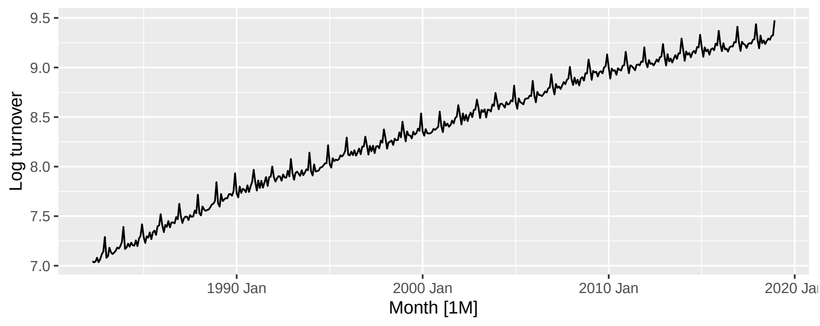
Variance stabilization

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



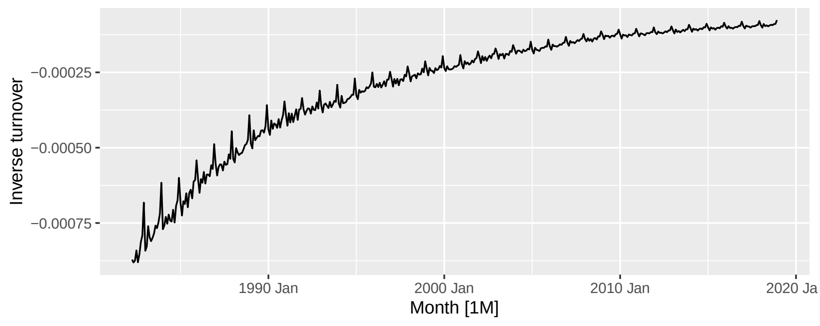
Variance stabilization

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



Variance stabilization

```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**:

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

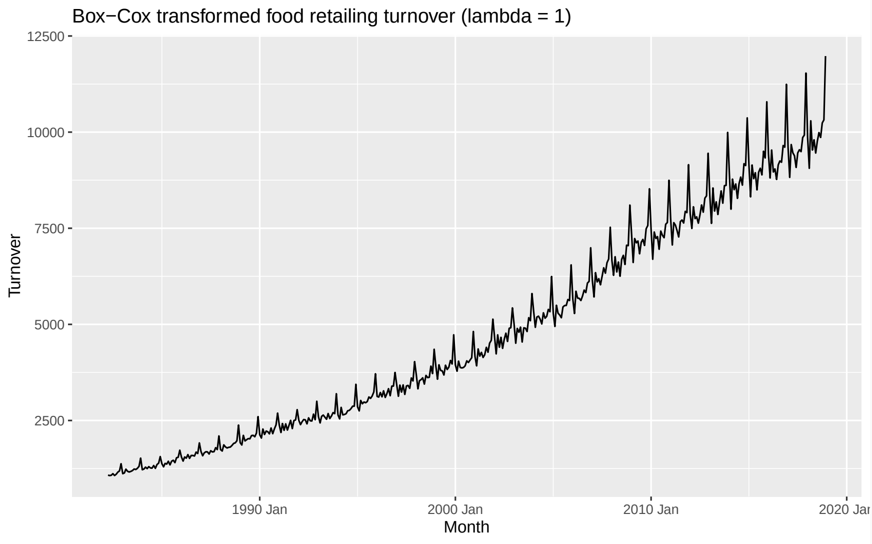
Box-Cox transformations

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

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

- 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

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

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

Box-Cox transformations

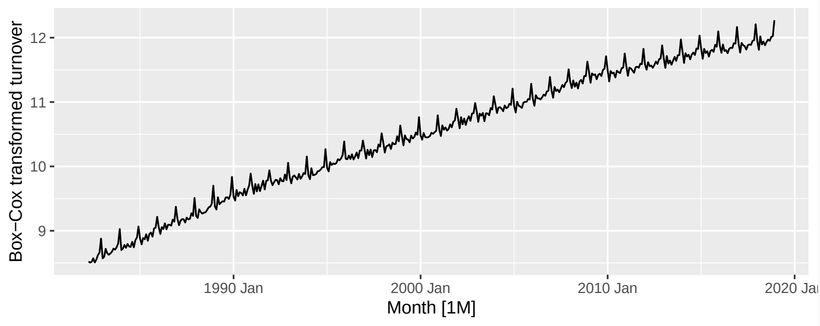
```
food |>
  features(Turnover, features = guerrero)
```

```
## # 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

```
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`.)

Outline

Time series decomposition

Trend-Cycle aperiodic changes in level over time.

Seasonal (almost) periodic changes in level due to seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Additive decomposition

$$y_t = S_t + T_t + R_t$$

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

STL decomposition

- STL: “Seasonal and Trend decomposition using Loess”
- Very versatile and robust.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Optionally robust to outliers
- No trading day or calendar adjustments.
- Only additive.
- Take logs to get multiplicative decomposition.
- Use Box-Cox transformations to get other decompositions.

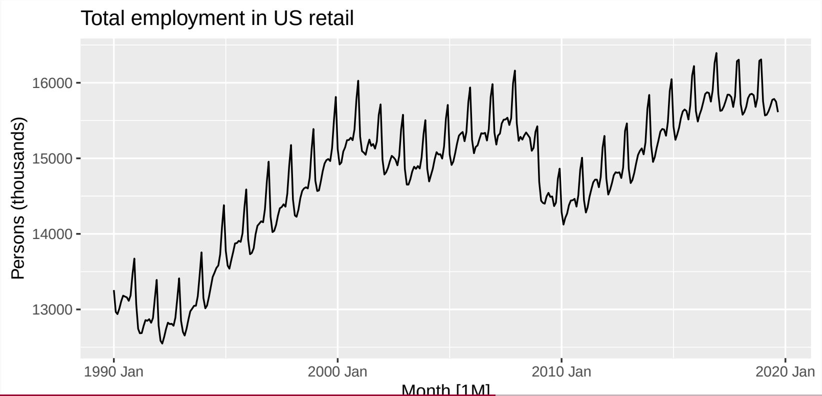
US Retail Employment

```r
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.
```

US Retail Employment

```
us_retail_employment |>
  autoplot(Employed) +
  labs(y = "Persons (thousands)", title = "Tot
```



US Retail Employment

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

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

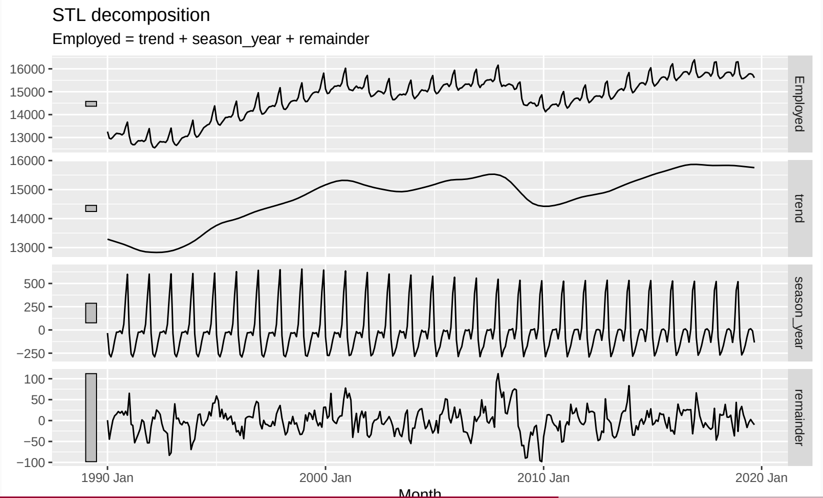
US Retail Employment

```
components(dcmp)
```

```
## # A dable: 357 x 7 [1M]
## # Key:       .model [1]
## # :          Employed = trend + season_year +
## #   remainder
##     .model     Month Employed  trend seaso~1 remai~2
##     <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
##   8 stl     1990 Aug   13160. 13151.    -9.52  17.8
##   9 stl     1990 Sep   13113. 13131.   -39.5   22.0
```

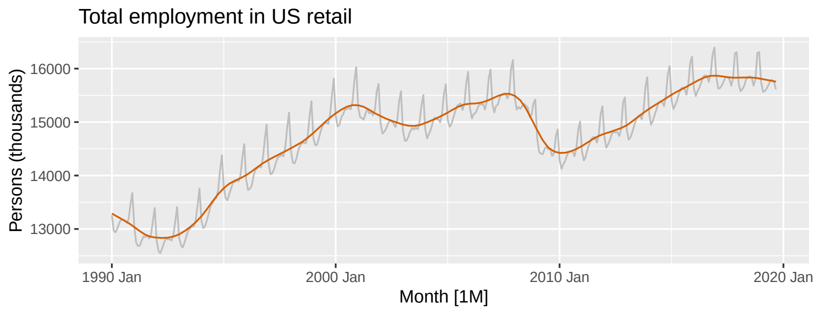
US Retail Employment

```
components(dcmp) |> autoplot()
```



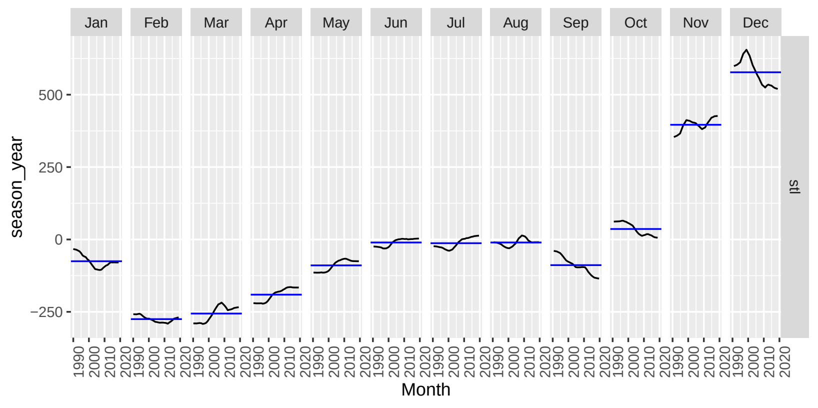
US Retail Employment

```
us_retail_employment |>
  autoplot(Employed, color = "gray") +
  autolayer(components(dcmp), trend, color = "
  labs(y = "Persons (thousands)", title = "Tot
```



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

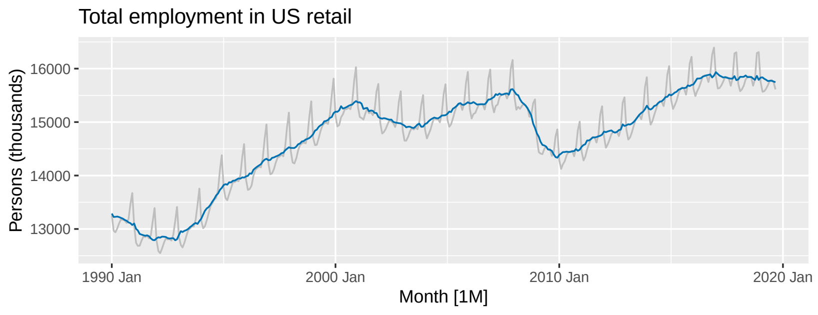
$$y_t - S_t = T_t + R_t$$

- Multiplicative decomposition: seasonally adjusted data given by

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

US Retail Employment

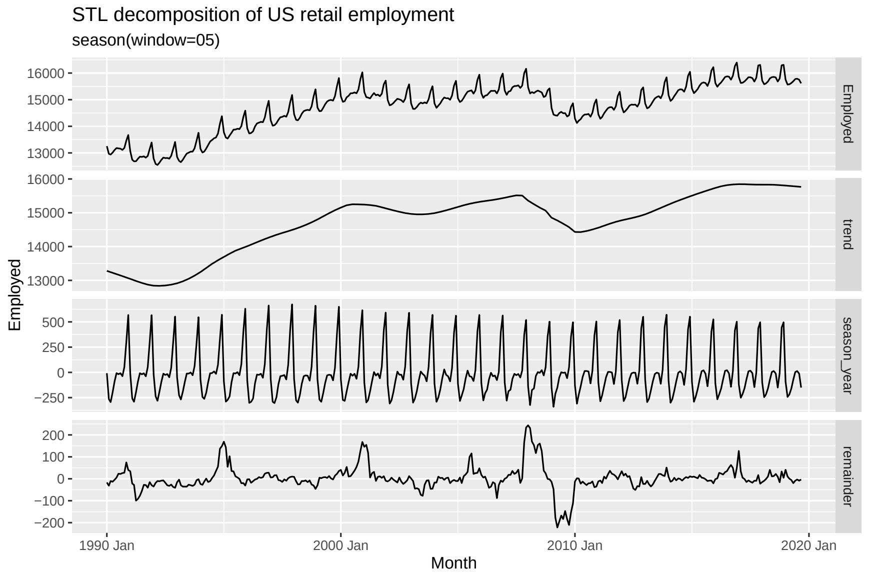
```
us_retail_employment |>
  autoplot(Employed, color = "gray") +
  autolayer(components(dcmp), season_adjust, c
  labs(y = "Persons (thousands)", title = "Tot
```



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.

STL decomposition



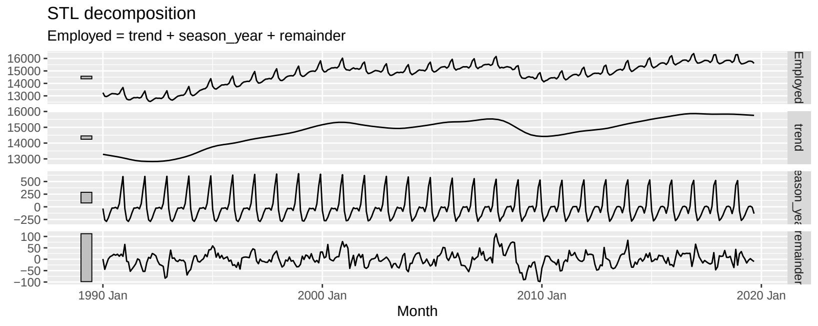
STL decomposition

```
us_retail_employment |>
  model(STL(Employed ~ trend(window = 15) + se
    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

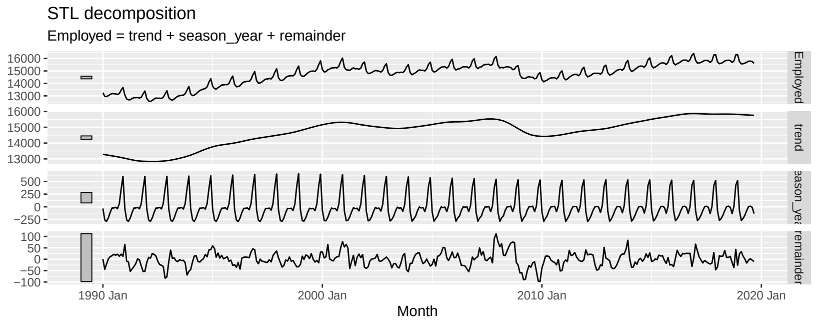
```
us_retail_employment |>
  model(STL(Employed)) |>
  components() |>
  autoplot()
```



STL decomposition

- `STL()` chooses `season(`
  by default
- Can include transformati

```
us_retail_employment |>
  model(STL(Employed)) |>
  components() |>
  autoplot()
```



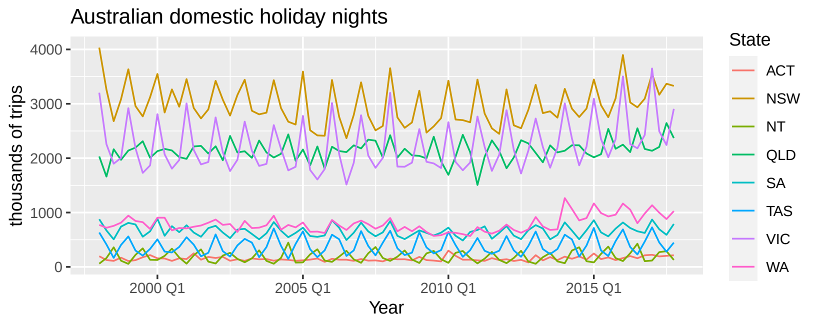
STL decomposition

- 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.
- trend window controls loess bandwidth on deasonalised values.
- season window controls loess bandwidth on detrended subseries.
- Robustness weights based on remainder.
- Default season: `window = 13`
- Default trend:

```
window = nextodd(ceiling((1.5*period)/(1-
```

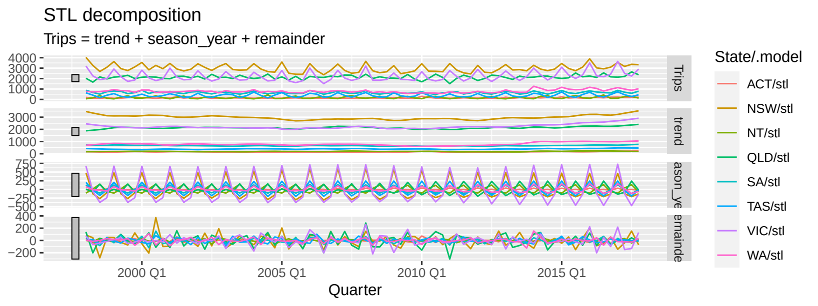
Australian holidays

```
holidays |> autoplot(Trips) +
  labs(y = "thousands of trips", x = "Year",
       title = "Australian domestic holiday ni
```



Australian holidays

```
holidays |>
  model(stl = STL(Trips)) |>
  components() |>
  autoplot()
```



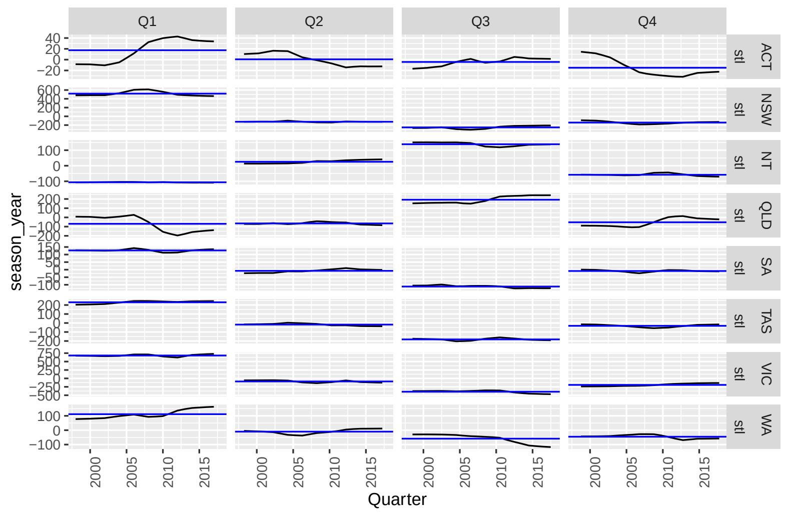
Holidays decomposition

```
dcmp <- holidays |>
  model(stl = STL(Trips)) |>
  components()
dcmp
```

```
## # A dable: 640 x 8 [1Q]
## # Key:     State, .model [8]
## # :        Trips = trend + season_year +
## #   remainder
##    State .model Quarter Trips trend season_year
##    <chr> <chr>    <qtr> <dbl> <dbl>       <dbl>
##  1 ACT   stl    1998 Q1  196.  172.       -8.48
##  2 ACT   stl    1998 Q2  127.  157.       10.3
##  3 ACT   stl    1998 Q3  111.  142.      -16.8
##  4 ACT   stl    1998 Q4  170.  130.       14.6
##  5 ACT   stl    1999 Q1  108.  135.       -8.63
##  6 ACT   stl    1999 Q2  125.  148.       11.0
##  7 ACT   stl    1999 Q3  178.  166.      -16.0
##  8 ACT   stl    1999 Q4  218.  177.       13.2
```

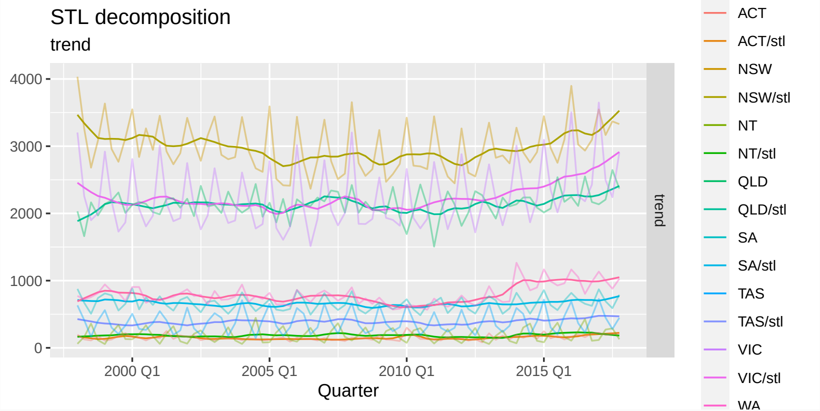
Holidays decomposition

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



Holidays decomposition

```r
autoplot(dcmp, trend, scale_bars = FALSE) +
  autolayer(holidays, alpha = 0.4)
```



Lab Session -transformation

per capita

Consider the GDP information in `global_economy`. Plot the GDP per capita for each country over time. Which country has the highest GDP per capita? How has this changed over time?

transformation

1. For the following series, find an appropriate transformation in order to stabilise the variance.
   - United States GDP from `global_economy`
   - Slaughter of Victorian "Bulls, bullocks and steers" in `aus_livestock`
   - Victorian Electricity Demand from `vic_elec`.

Lab Session- decomposition

1. Produce the following decomposition

```
canadian_gas |>
  model(STL(Volume ~ season(window=7) + trend(window=11))) |>
  components() |>
  autoplot()
```

2. What happens as you change the values of the two `window` arguments?
3. How does the seasonal shape change over time? [Hint: Try plotting the seasonal component using `gg_season`.]
4. Can you produce a plausible seasonally adjusted series? [Hint: `season_adjust` is one of the variables returned by `STL`.]