November 3rd, 2023

Applied Forecasting in Complex Systems
(Lecture 2)

Erman Acar

Informatics Institute
Institute for Logic, Language and Computation

Today's Roadmap

- Time Series Decomposition
  - Adjustments
  - Transformations
  - Moving Averages
  - Classical Decompositions
  - A Brief Overview of Complex Decompositions

- Time Series Features in R
  - Some Handy Features
  - STL Features

Time Series Decomposition

Introduction

Why bother decomposition?

Time series data often exhibit a variety of patterns, hence it is useful to split it into several components

- to understand the characteristics of each component, and
- the contribution of each component to the original time series.

It can also help improving the forecast accuracy.

Three components

We usually decompose a time series into three component

1. *Trend-cycle component* (i.e., we just call *trend component* for simplicity)
2. *Seasonal component*
3. *Remainder component*

In this part, we will see the most common methods for extracting these components from a time series.

Adjustments & transformations

Motivation

When doing decomposition, it is often useful to check for applicable **adjustment** or **transformation** first, because:

- It can greatly simplify the time series, hence the decomposition process.
- It can remove the known source of variation (exploiting our background/domain knowledge).
- It can make the patterns more consistent across the whole data set.
- Simpler patterns are usually easier to model and lead to more accurate forecasts.

Adjustments

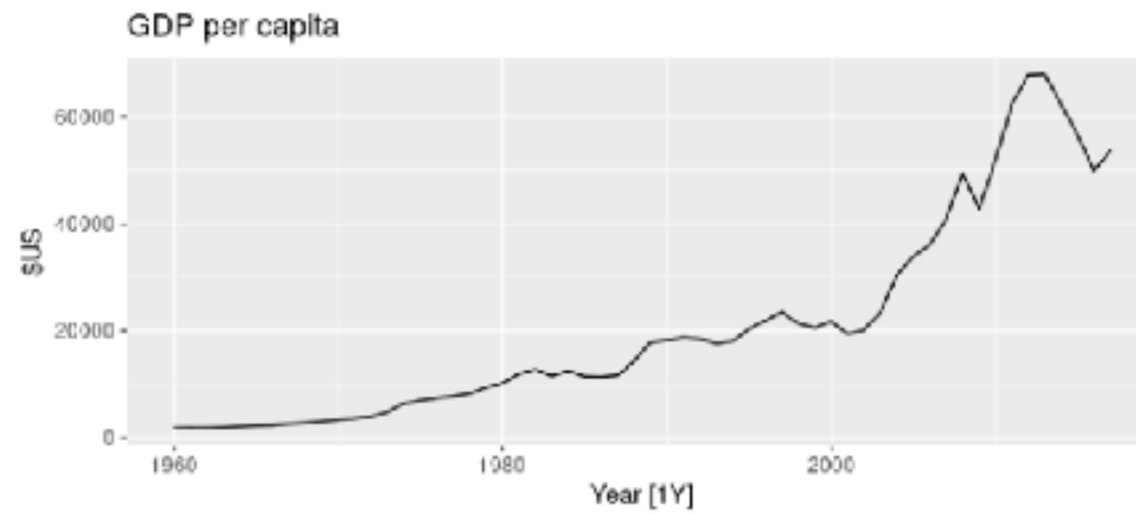
Recall that adjustments exploits our background knowledge about the data. So we list three common examples of that:

Population adjustments

Any data that are affected by population changes can be adjusted by *per capita* (or *per thousand, million*, etc.).

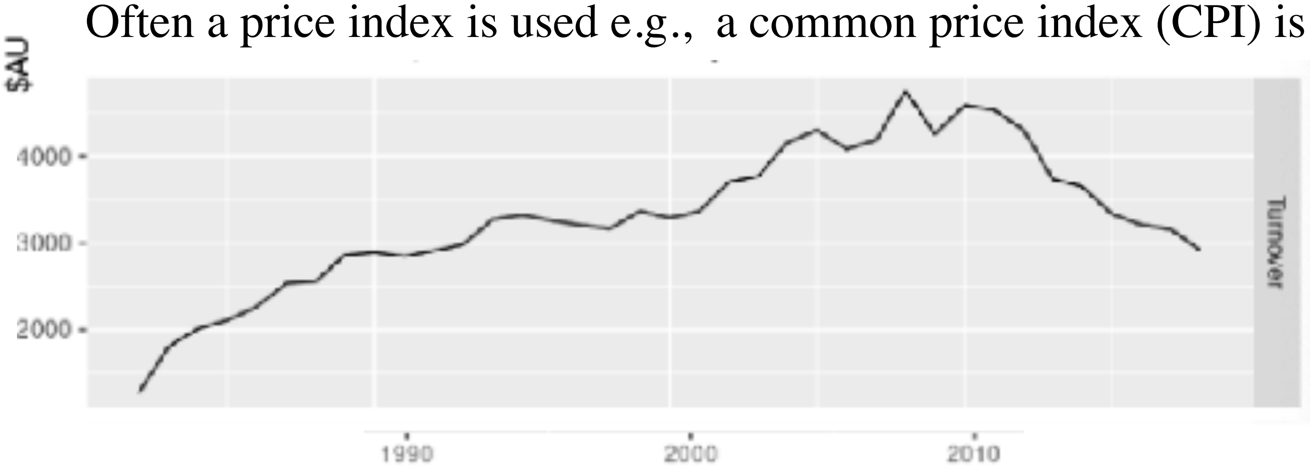
It stabilises/smoothens the series.

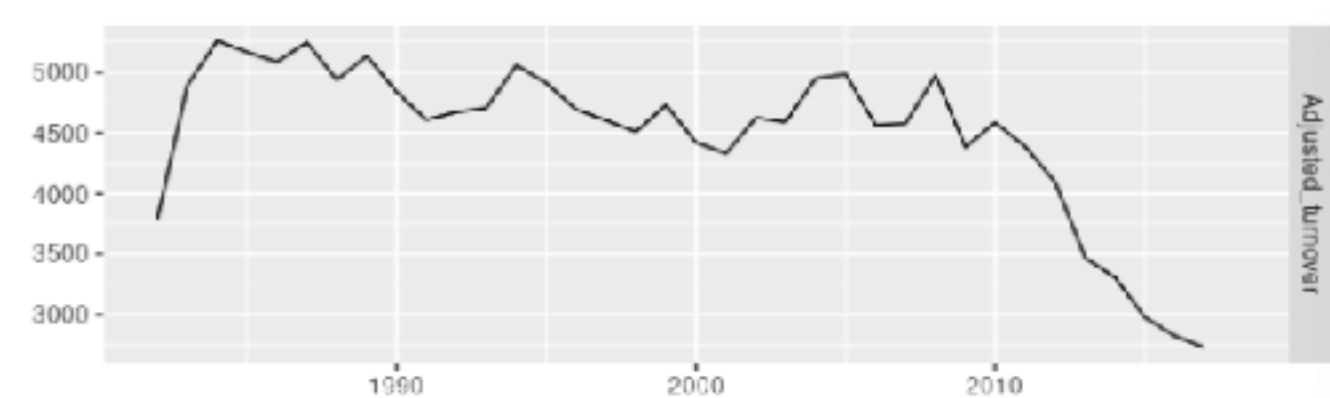
```
global_economy |>
    filter(Country == "Australia") |>
    autoplot(GDP/Population) +
    labs(title= "GDP per capita", y = "$US")
```



Inflation Adjustments

Often a price index is used e.g., a common price index (CPI) is a value/year. *Example*: Adjusted_turnover = Turnover / CPI · 100:





Data for Newspaper and book retail turnover.

Year

Calendar Adjustments

Some of the variation seen in seasonal data may be due to simple calendar effects.

Ex: *average sales per day in each month (adjusted) vs. Total sales in a month* — across the whole year —

Stabilising the Variations

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

- We denote original observations as $y_1, \ldots, y_T$ and transformed observations as $w_1, \ldots, w_T$.

- E.g., for the *logarithmic transformation*: $\log(y_t) = w_t$

Logarithms, in particular, are useful because they are easily interpretable:

- Changes in a log value are relative (percent) changes on the original scale:

  Ex.: For log base 10, an increase of 1 unit on the log scale corresponds to multiplication by 10 on the original scale.

- However, if any value of the original series is zero or negative, then logarithms are not possible.

- Other transformations that are less interpretable are *power transformations*: $w_t = y_t^p$

Mathematical transformations for stabilizing variation
Square root $w_t = \sqrt{y_t}$ when p is 1/2
Cube root $w_t = \sqrt[3]{y_t}$ when p is 1/3
Logarithm $w_t = \log y_t$

Box-Cox transformations
(A More General Family)

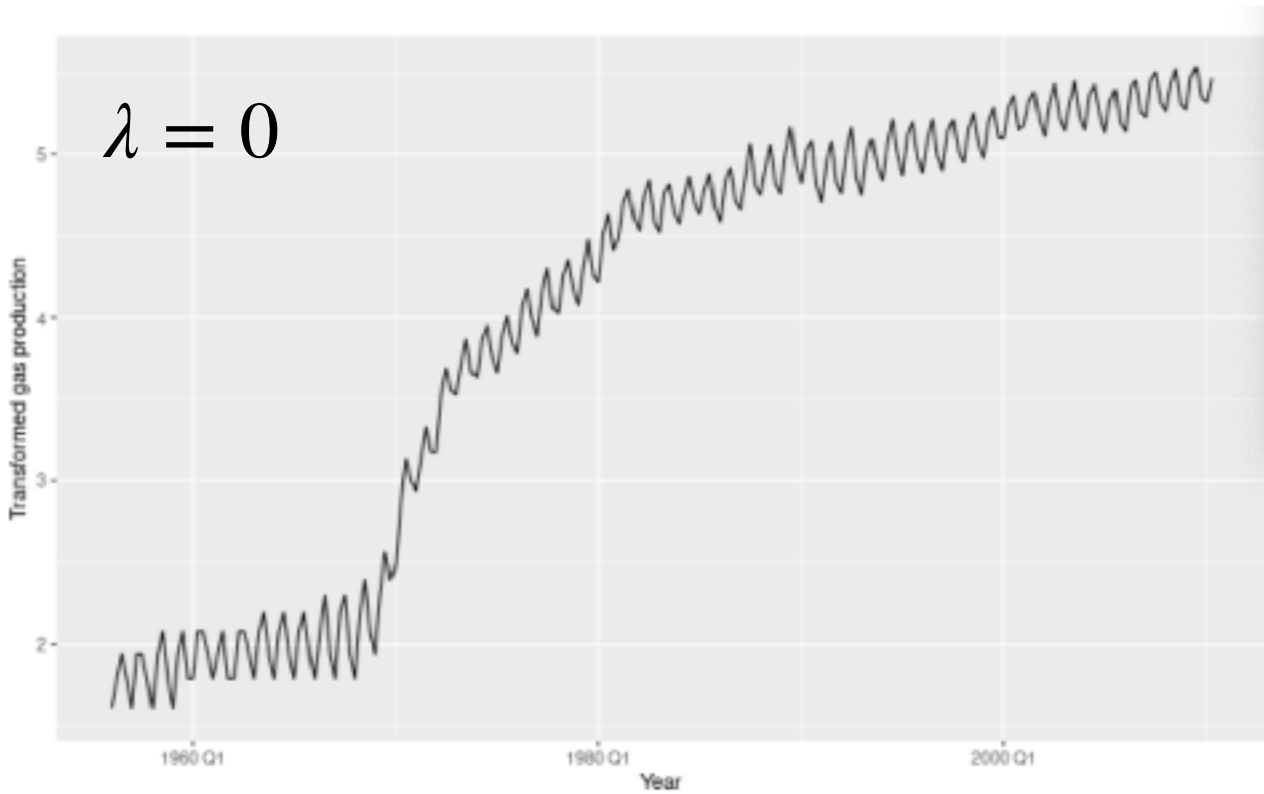
A useful family of transformations that includes <u>both logarithms and power transformations</u>, is the Box-Cox transformations (Box & Cox, 1964).

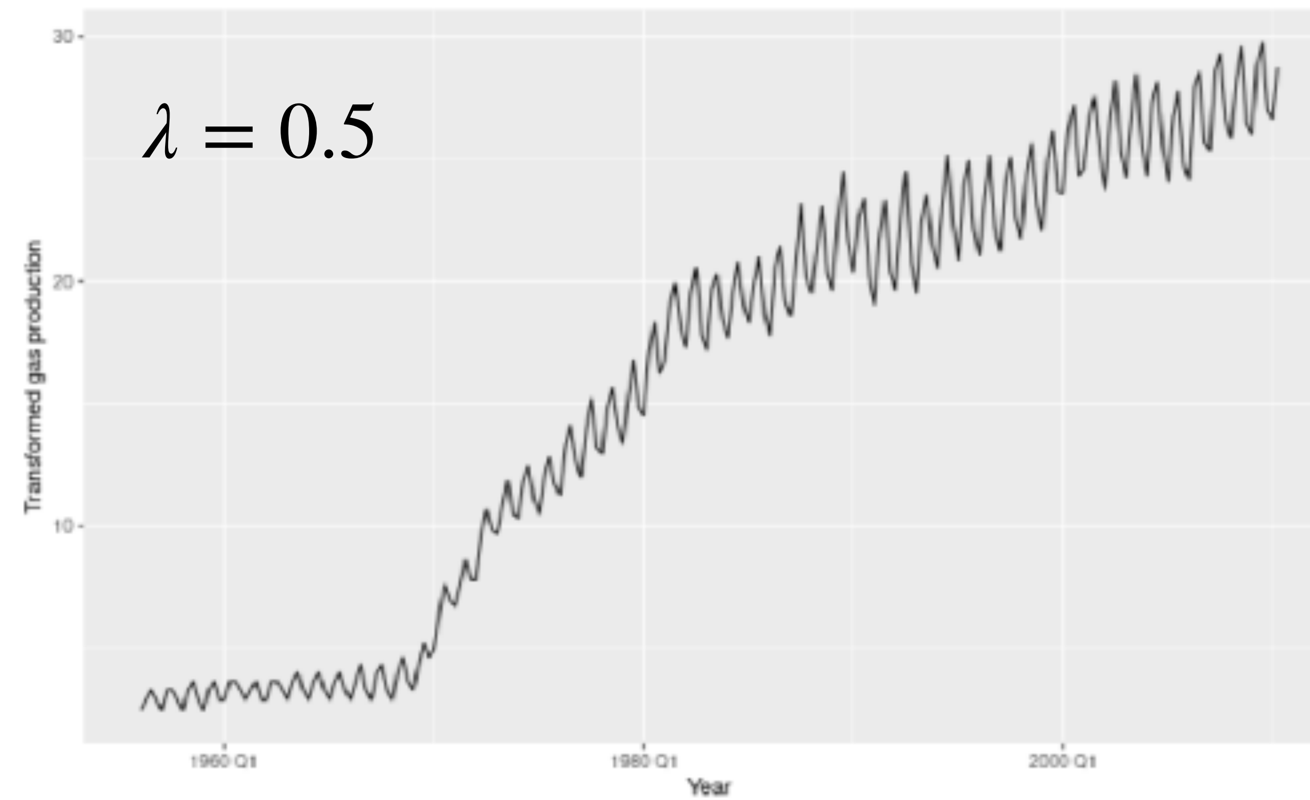
$$w_t = \begin{cases} \log(y_t) & \text{if } \lambda = 0; \\ (\text{sign}(y_t)|y_t|^\lambda - 1)/\lambda & \text{otherwise.} \end{cases}$$

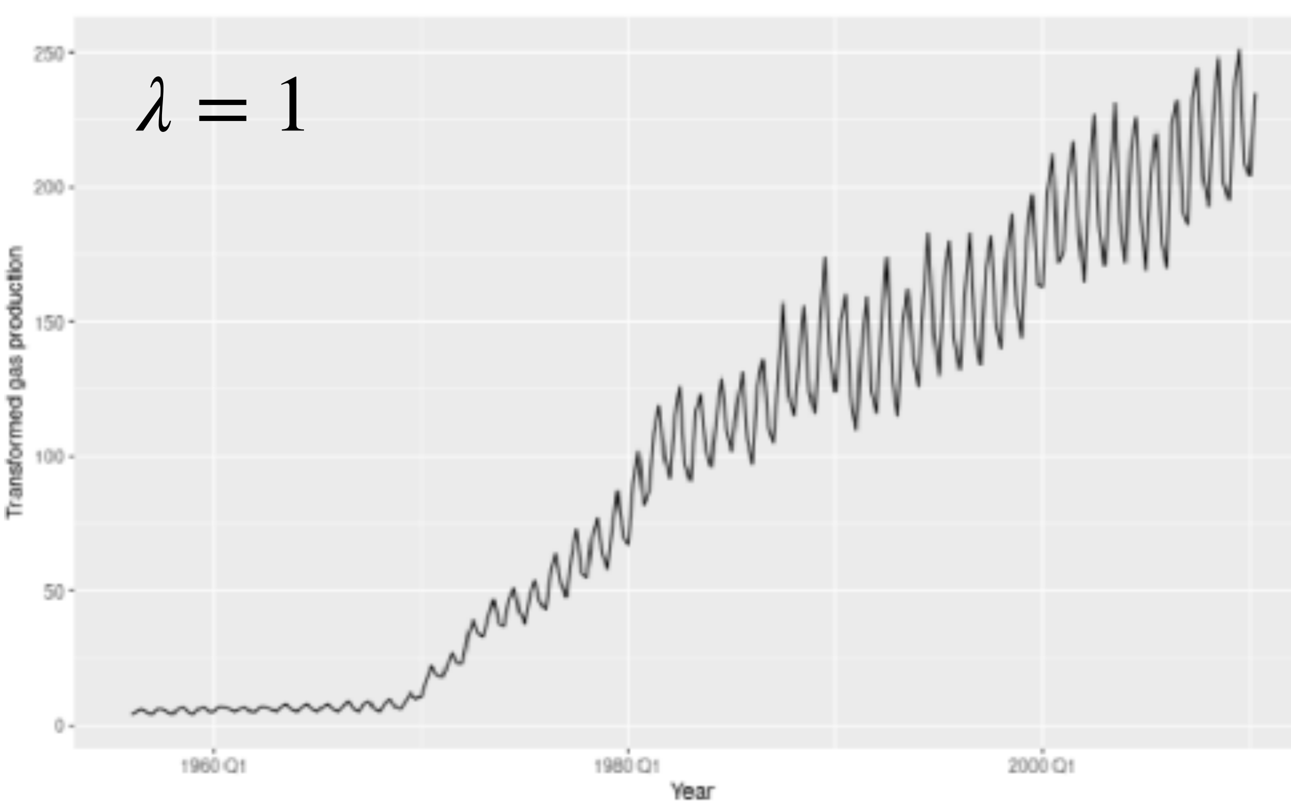
Remark: The logarithm in a Box-Cox transformation is always a natural logarithm (i.e., to base e).

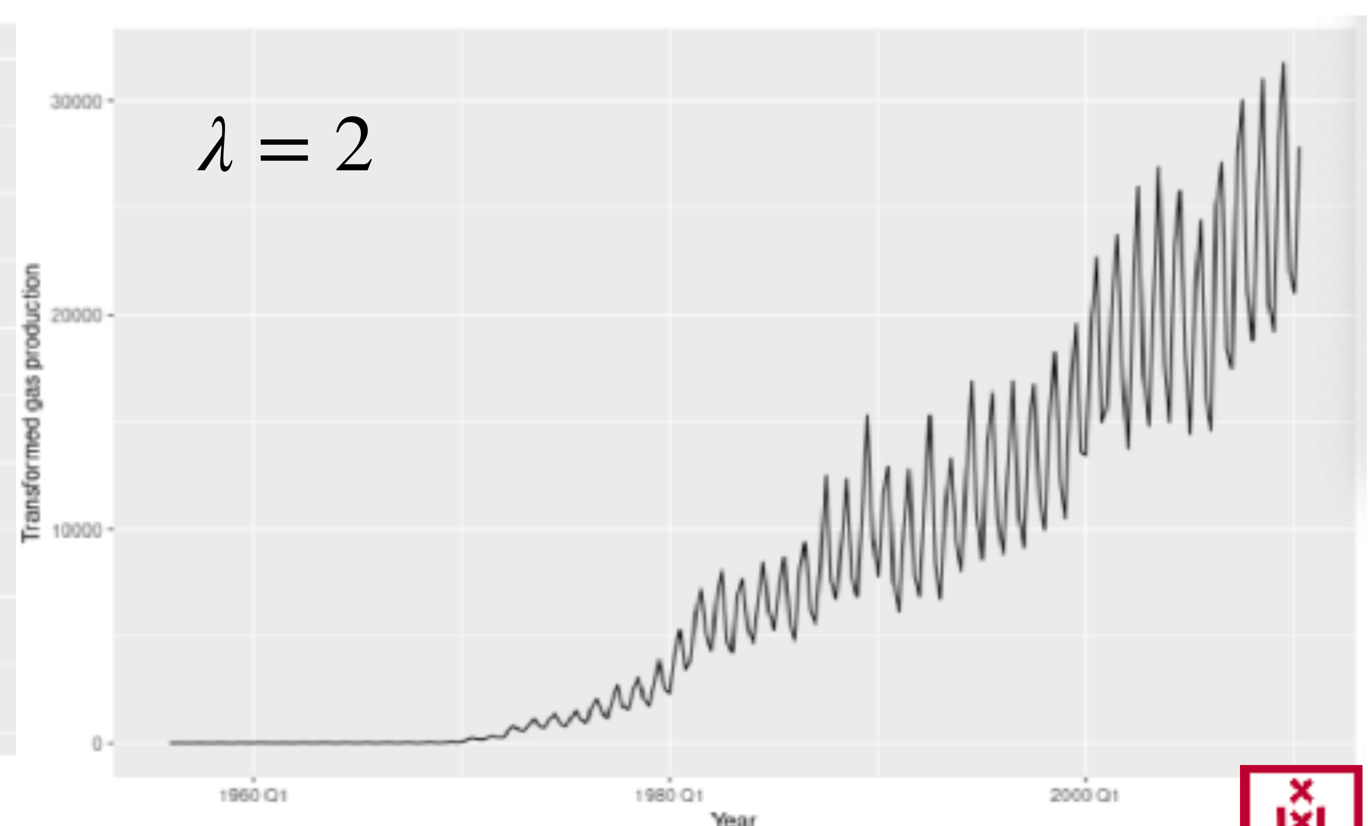
Realize that if $\lambda = 1$, then $w_t = y_t - 1$, meaning the data is shifted downwards without any change in its shape.

Box-Cox transformations
(Example: Australian Quarterly Gas Production)









Which one is the best?

Box-Cox transformations
(Example: Australian Quarterly Gas Production)

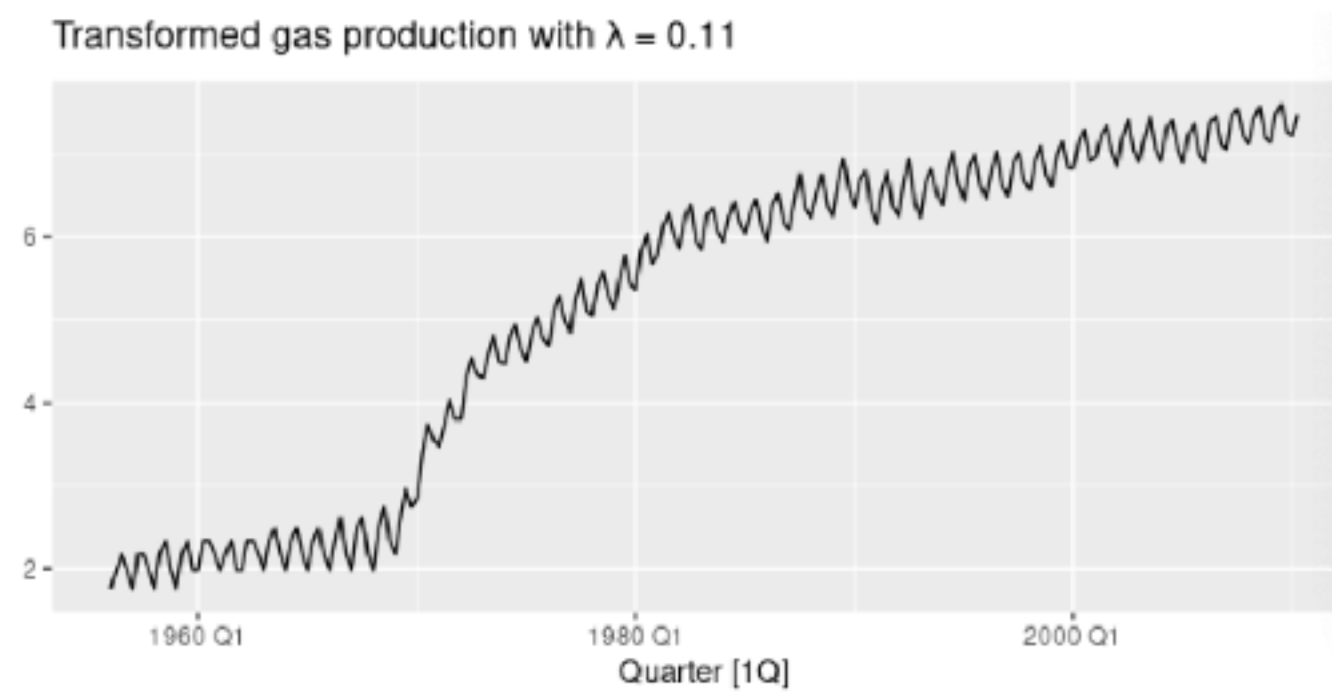
Which one is the best?

The one which makes the size of the seasonal variation about the same across the whole series.
(Since that makes the forecasting model simpler.)

Calculating Optimal λ

A *feature* that is called Guerrero can be used to choose the value.

```
lambda <- aus_production |>
    features(Gas, features = guerrero) |>
    pull(lambda_guerrero)
aus_production |>
    autoplot(box_cox(Gas, lambda)) +
    labs(y = "",
         title = latex2exp::TeX(paste0(
             "Transformed gas production with $\\lambda$ = ",
             round(lambda,2))))
```



Time Series Components

Two Main Types of Decomposition:

Additive Decomposition: $y_t = S_t + T_t + R_t$

preferred when the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series.

Multiplicative Decomposition: $y_t = S_t \times T_t \times R_t$

preferred when the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series. Ex.: economic time series

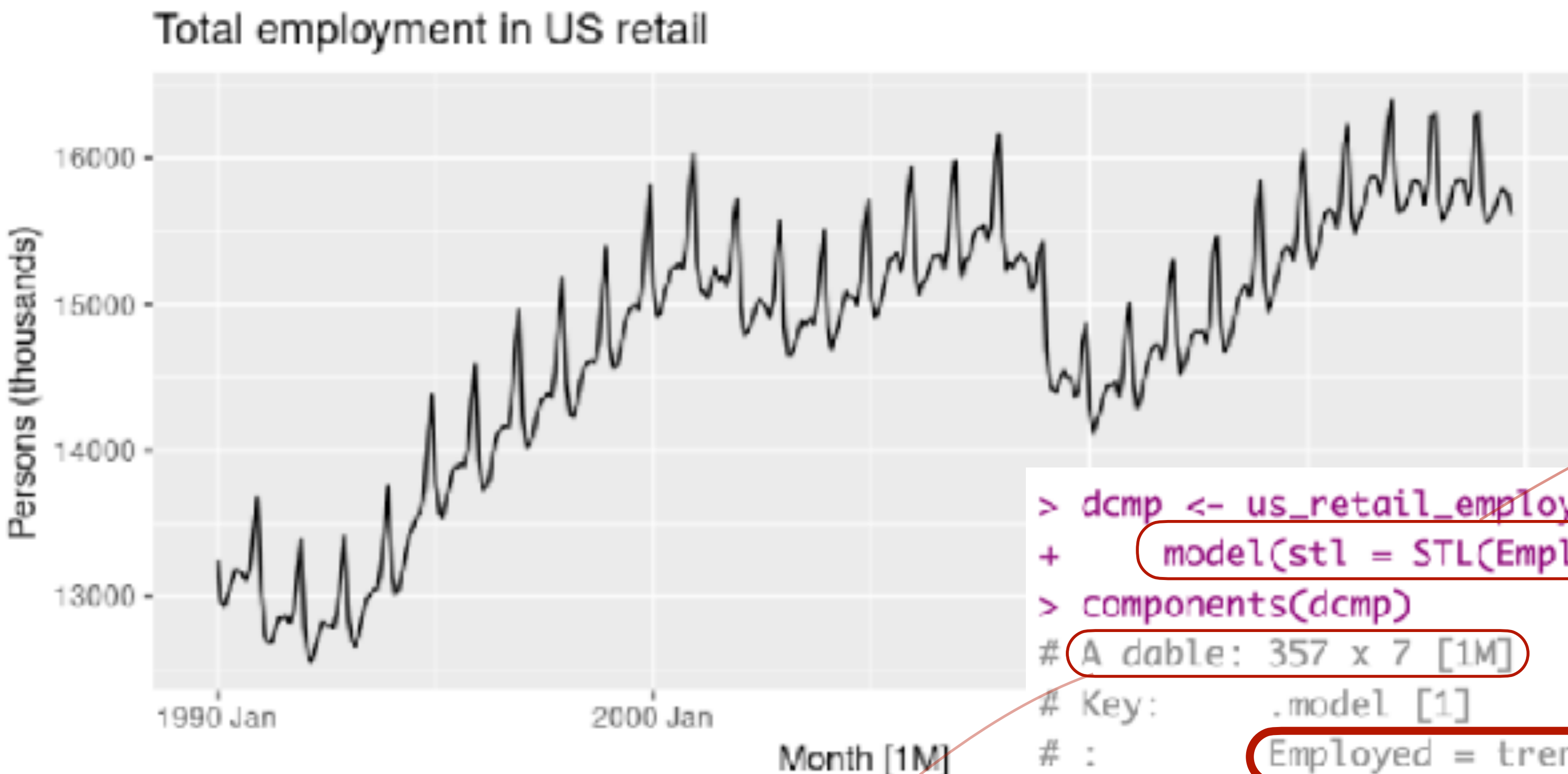
where y_t is the data at time t, and

- S_t is the seasonal component,
- T_t is the trend-cycle component,
- R_t is the remainder component.

Observe that a log transformation is equivalent to using a multiplicative decomposition:

$y_t = S_t \times T_t \times R_t$ is equivalent to $\log y_t = \log S_t + \log T_t + \log R_t$.

Decomposition
(Example)



STL function

(Seasonal and Trend decomposition using Loess)

```
> 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_adj…¹
  <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.
# … with 347 more rows, and abbreviated variable name ¹season_adjust
```

additive decomposition

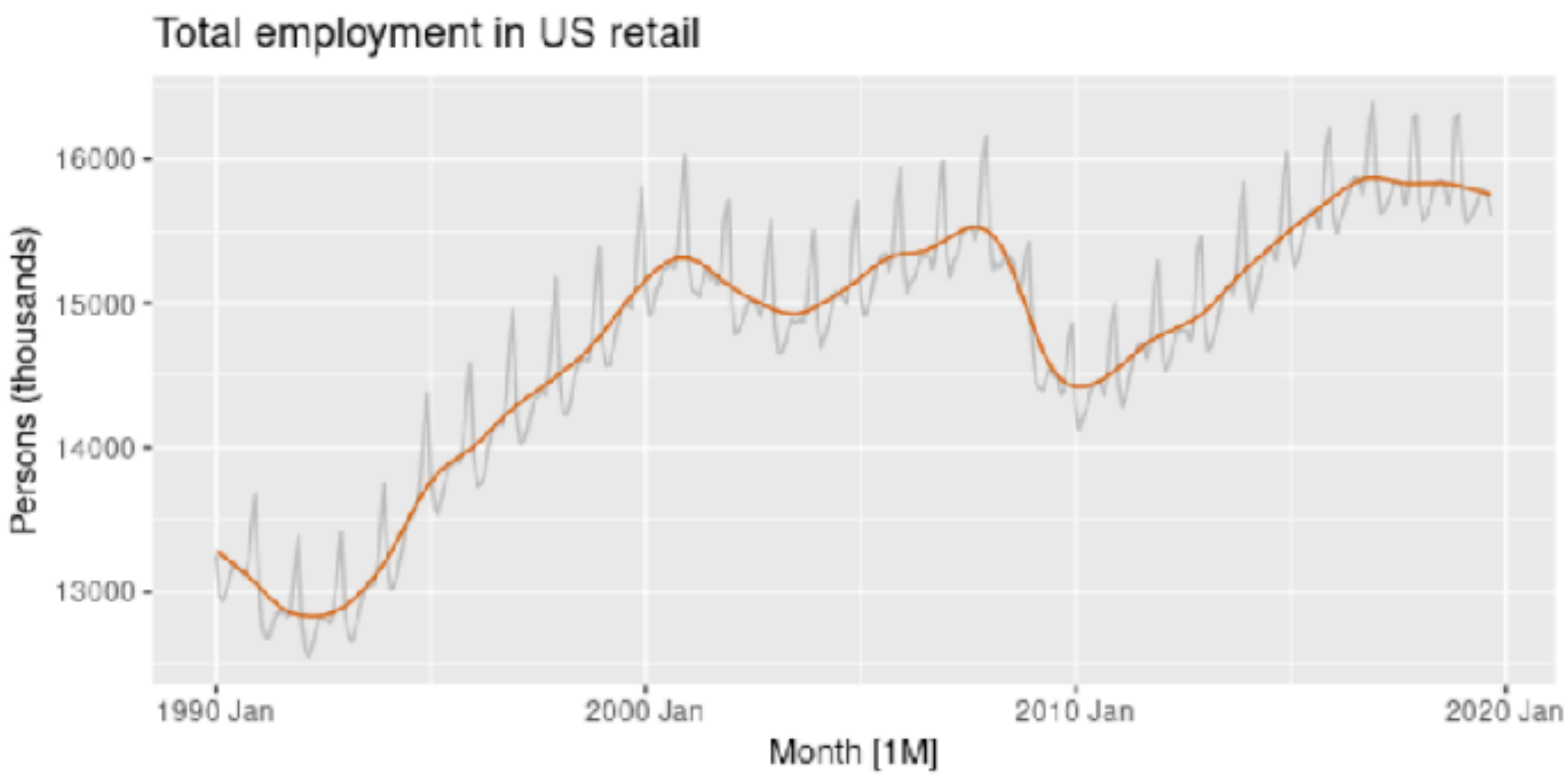
dable: decomposition table

Original data value

Decomposition
(Example Cont'd)

```
components(dcmp) |>
    as_tsibble() |>
    autoplot(Employed, colour="gray") +
    geom_line(aes(y=trend), colour = "#D55E00") +
    labs(
        y = "Persons (thousands)",
        title = "Total employment in US retail"
    )
```

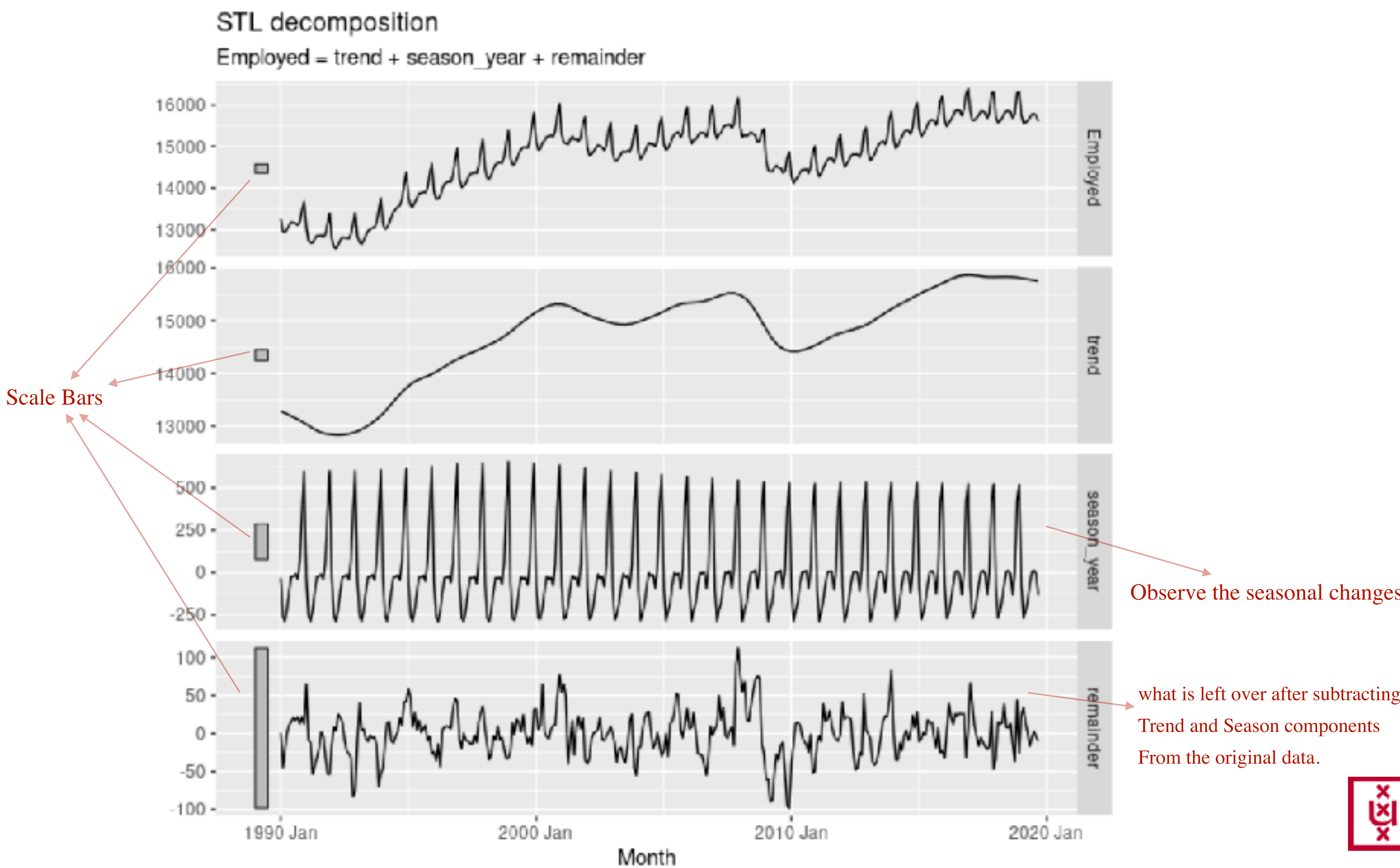
Observe below that the trend component follows the overall movement of the series, ignoring any seasonality and random fluctuations.



Decomposition (Example Cont'd)

One can visualise separate components altogether next to original data.

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



Seasonally Adjusted Data

Seasonal Adjustment

If the seasonal component is removed from the original data, the resulting values are the *seasonally adjusted* data: $y_t - S_t$ (for additive decomp.) y_t / S_t (for multiplicative decomp.)

Seasonal adjustment is useful when you are interested in non-seasonal variation e.g., unemployment due to economic recession.

Remark: To study *turn points*, seasonally adjusted data can be misleading (since it has remainder component) i.e., better use trend-cycle component alone.

```
components(dcmp) |>
    as_tsibble() |>
    autoplot(Employed, colour = "gray") +
    geom_line(aes(y=season_adjust), colour = "#9800C3") +
    labs(y = "Persons (thousands)",
         title = "Total employment in US retail")
```

Classical Decomposition

Moving Average Smoothing

The first step in a *classical decomposition* is to use a moving average method to estimate the trend-cycle,

Moving average of order *m* (i.e., *m*-MA)

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

where $m = 2k + 1$.

Intuition: The estimate of the trend-cycle at time t which is obtained by averaging values of the time series within k periods of t.

Purpose: Obtaining a smooth trend-cycle component.

```
# A tsibble: 58 x 10 [1Y]
# Key:       Country [1]
   Country   Code   Year         GDP Growth   CPI Imports Exports Population `5-MA`
   <fct>     <fct> <dbl>       <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>  <dbl>
 1 Australia AUS    1960 18573188487.  NA     7.96    14.1    13.0   10276477   NA
 2 Australia AUS    1961 19648336880.   2.49  8.14    15.0    12.4   10483000   NA
 3 Australia AUS    1962 19888005376.   1.30  8.12    12.6    13.9   10742000   13.5
 4 Australia AUS    1963 21501847911.   6.21  8.17    13.8    13.0   10950000   13.5
 5 Australia AUS    1964 23758539590.   6.98  8.40    13.8    14.9   11167000   13.6
 6 Australia AUS    1965 25931235301.   5.98  8.69    15.3    13.2   11388000   13.4
 7 Australia AUS    1966 27261731437.   2.38  8.98    15.1    12.9   11651000   13.3
 8 Australia AUS    1967 30389741292.   6.30  9.29    13.9    12.9   11799000   12.7
 9 Australia AUS    1968 32657632434.   5.10  9.52    14.5    12.3   12009000   12.6
10 Australia AUS    1969 36620002240.   7.04  9.83    13.3    12.0   12263000   12.6
# … with 48 more rows
# ℹ Use `print(n = ...)` to see more rows
```

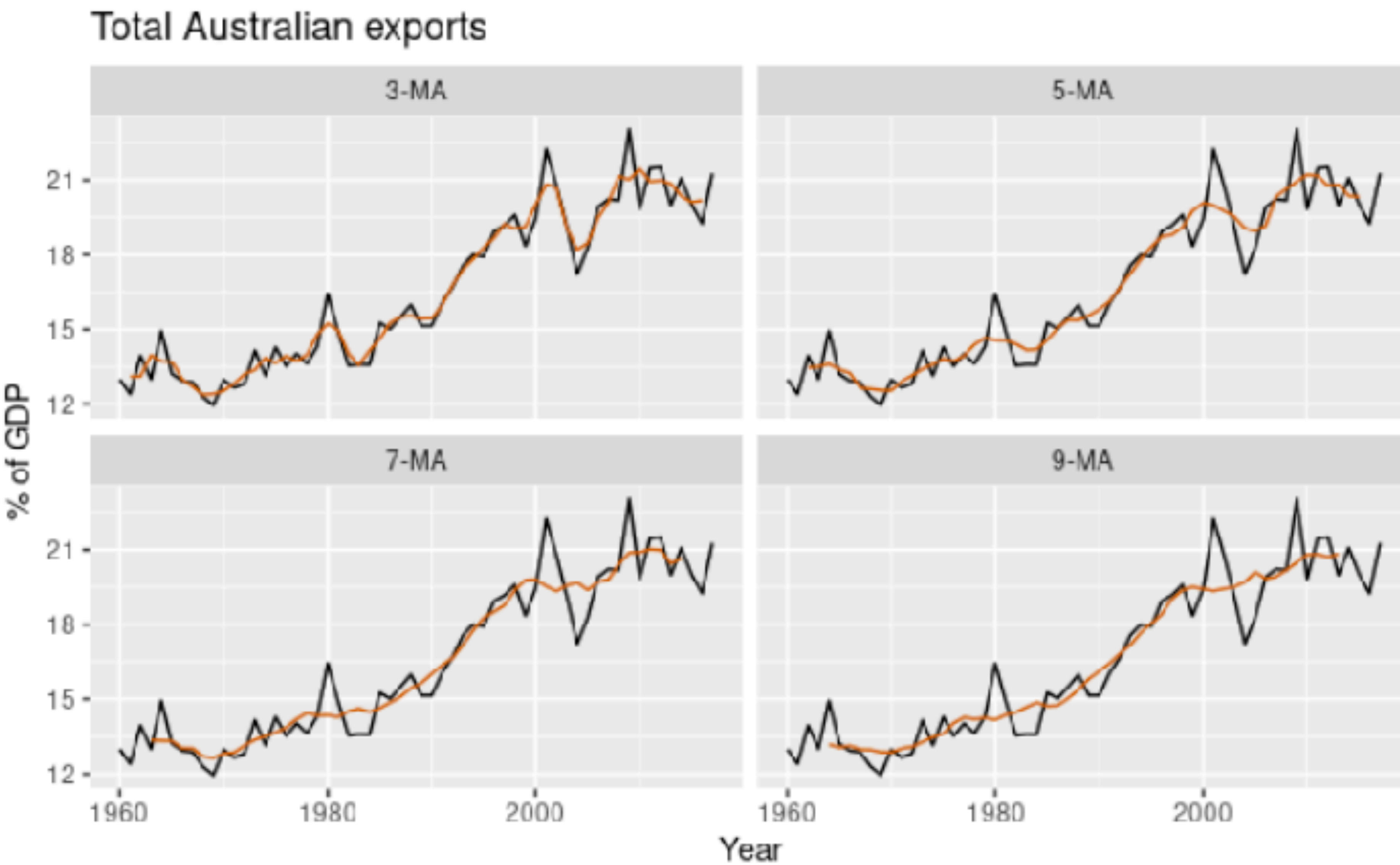
$k = 2$

$m = 2k+1 = 5$

Annual Australian exports of goods and services starting from 1960

Moving Average Smoothing

Larger the order smoother the trend-cycle estimate.



What is *k*?

Question: How to do an even *m*?

Moving Averages[2]

How to do an even *m*?

One can do a second order moving averages (i.e., applying it twice or moving averages of moving averages)

And it can get even smoother (i.e., not necessarily always desirable.)

Moving average of order *m* (i.e., *m*-MA)

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

where $m = 2k + 1$.

Quarter	Beer	4 - MA	2 x 4 - MA
1992 Q1	443.00		
1992 Q2	410.00	451.25	
1992 Q3	420.00	448.75	450.00
1992 Q4	532.00	451.50	450.12
1993 Q1	433.00	449.00	450.25
1993 Q2	421.00	444.00	446.50
...	...	...	...
2009 Q1	415.00	430.00	428.88
2009 Q2	398.00	430.00	430.00
2009 Q3	419.00	429.75	429.88
2009 Q4	488.00	423.75	426.75
2010 Q1	414.00		
2010 Q2	374.00		

451.25=(443+410+420+532)/4

448.75=(410+420+532+433)/4

Meaning: a 4-MA followed by a 2-MA.

$$\hat{T}_t = \frac{1}{2}\left[\frac{1}{4}(y_{t-2} + y_{t-1} + y_t + y_{t+1}) + \frac{1}{4}(y_{t-1} + y_t + y_{t+1} + y_{t+2})\right]$$
$$= \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}.$$

Also called *centred moving averages* due to symmetry

450.00=(451.25+448.75)/2

Other examples are also possible e.g., 3 x 3 MA is often used.

In general, an even (odd) order should be followed by an even (odd) order to make it symmetric.

Weighted Moving Averages

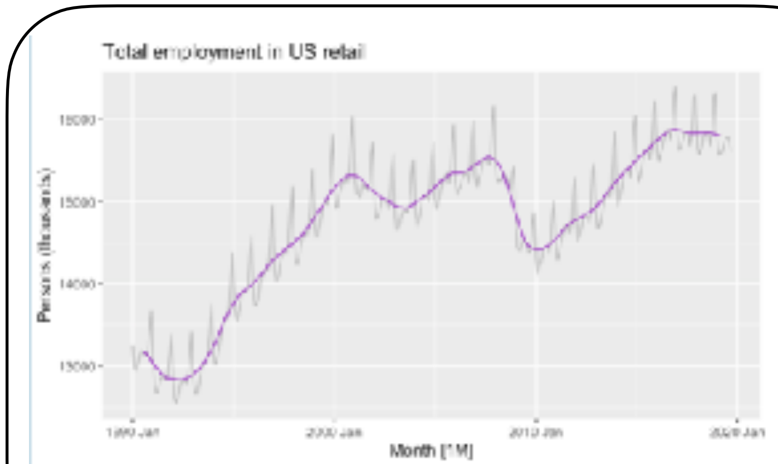
You can also estimate trend-cycle using seasonal data

Recall the previous 2 x 4 MA: $\hat{T}_t = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$

The Idea: In case of quarterly data, each quarter of the year is given equal weight as the first and last terms correspond to consecutive years. Hence, the seasonal variation will be averaged out. (Similar effect when 2 x 8 or 2 x 12 MA).

In general: When estimating trend cycle for seasonal period of order *m*, we can use

- 2 x *m*- MA if *m* is even
  - Example: 2 x 12-MA for monthly data with annual seasonality
- *m* - MA if *m* is odd
  - Example :7-MA can for daily data with a weekly seasonality



Trend cycle obtained with 2 x 12-MA

This can be generalised into:

Weighted Moving Averages:

$$\hat{T}_t = \sum_{j=-k}^{k} a_j y_{t+j},$$

where $k = (m-1)/2$ and a_j is the j-th weight

Example: 2 x 4-MA is equivalent to $\left[\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right]$

m-MA is a special case where all of the weights are equal to $1/m$

Advantage: ability to have smoother estimate of the trend-cycle.

Idea: weights slowly increase and then slowly decrease, resulting in a smoother curve.

Classical Decomposition

The classical decomposition method originated in the 1920s, and is relatively simple. It is the basis of other complex methods of decomposition.

General Idea:

Additive Decomposition:

Assumption: Seasonal component is constant.

Step 1: Compute the Trend-cycle $\hat{T}_t$ (i.e., previous slide)

Step 2: Calculate the detrended series: $y_t - \hat{T}_t$

Step 3: Estimate the seasonal component $\hat{S}_t$:
Simply average the detrended values for each season and string them together.

Step 4: Calculate the remainder by: $\hat{R}_t = y_t - \hat{T}_t - \hat{S}_t$

Multiplicative Decomposition:

Step 1: Compute the Trend-cycle $\hat{T}_t$ (i.e., previous slide)

Step 2: Calculate the detrended series: $y_t / \hat{T}_t$

Step 3: Estimate the seasonal component $\hat{S}_t$:
Simply average the detrended values for each season and string them together.

Step 4: Calculate the remainder by: $\hat{R}_t = y_t / (\hat{T}_t \hat{S}_t)$

Problems with Classical Decomposition:

Although classical decomposition is still widely used, it is not recommended, and there are better methods.

- It is not robust to unusual values/events (e.g., strikes in airport total working hours).
- The estimate of the trend-cycle is unavailable for the first few and last few observations.
- It assumes that the seasonal component repeats from year to year (e.g., changes in decades).
- Over-smoothing of rapid rises and falls.

A Brief Overview of Complex Decomposition

A Brief Info

Official statistics agencies (such as Centraal Bureau Statistiek or U.S. Census Burau) are responsible for a large number of official economic and social time series.

Most of them have their decomposition procedures used for seasonal adjustments, which are variants of the methods such as X-11, or the SEATS, or a combination of the two.

They are specifically designed to work with quarterly and monthly data, and not such as daily data, or hourly data.

In addition to those, there is also a commonly used STL (Seasonal and Trend decomposition using Loess) developed by [R. B. Cleveland et al. 1990]

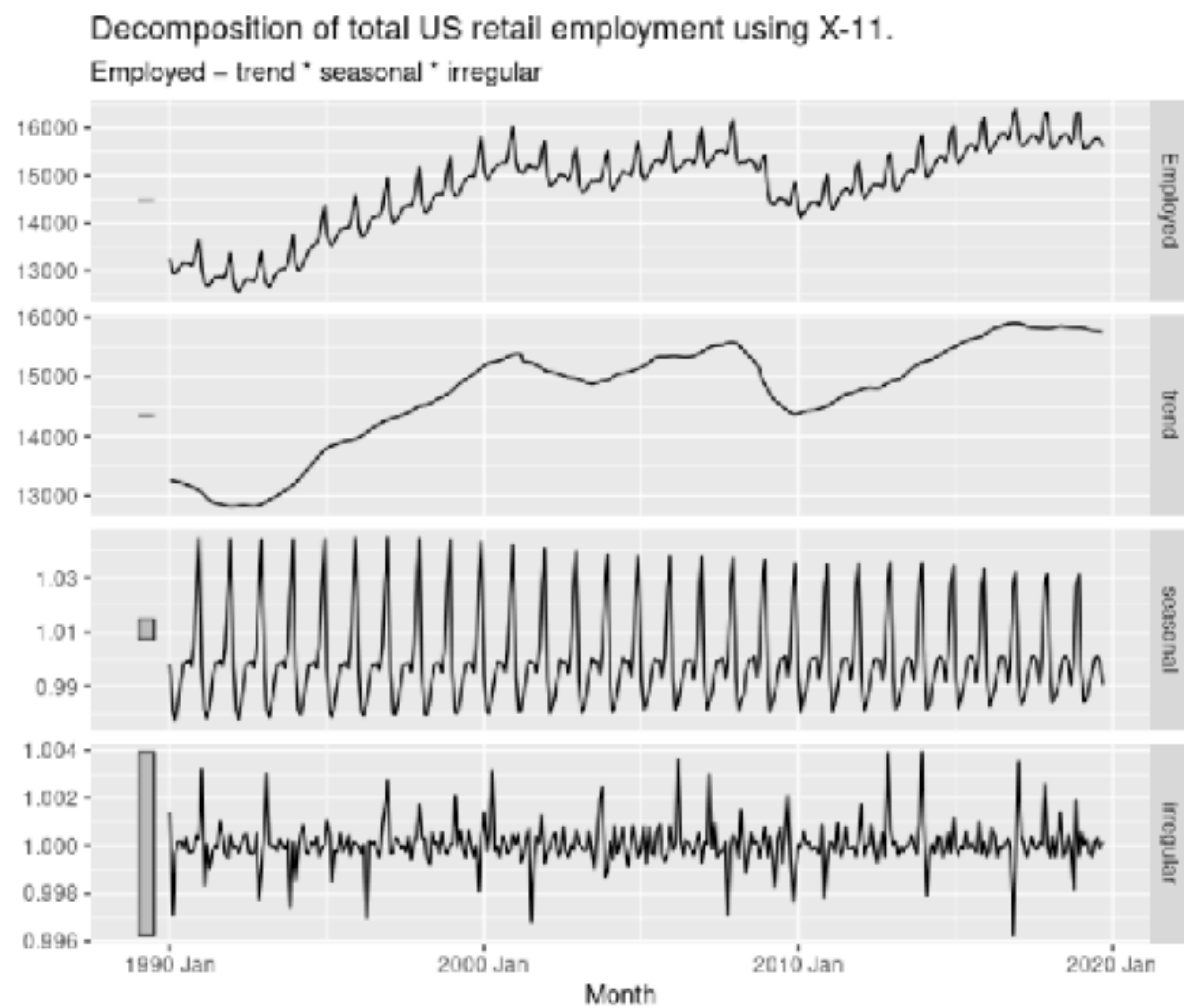
X-11 Method

Based on classical decomposition, X-11 comes with more advanced techniques (see [Dagum, E. B., & Bianconcini, S. , 2016] for detail) that overcomes the problems of classical decomposition has, including:

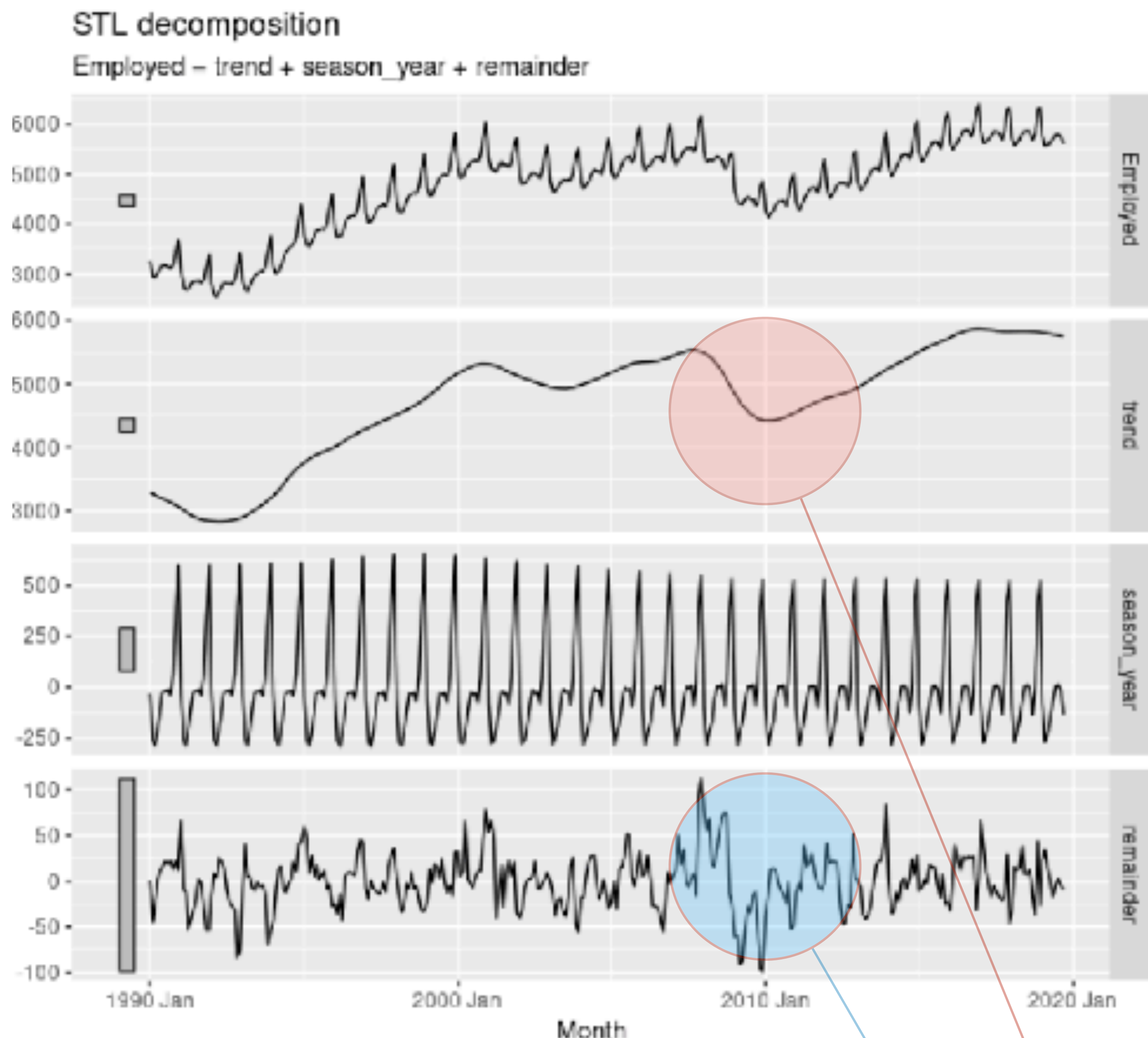
- trend-cycle estimates are available for all observations including the end points.
- the seasonal component is allowed to vary slowly over time.
- It handles trading day variation, holiday effects and the effects of known predictors
- It supports both additive and multiplicative decomposition, and highly robust against level shits and outliers.

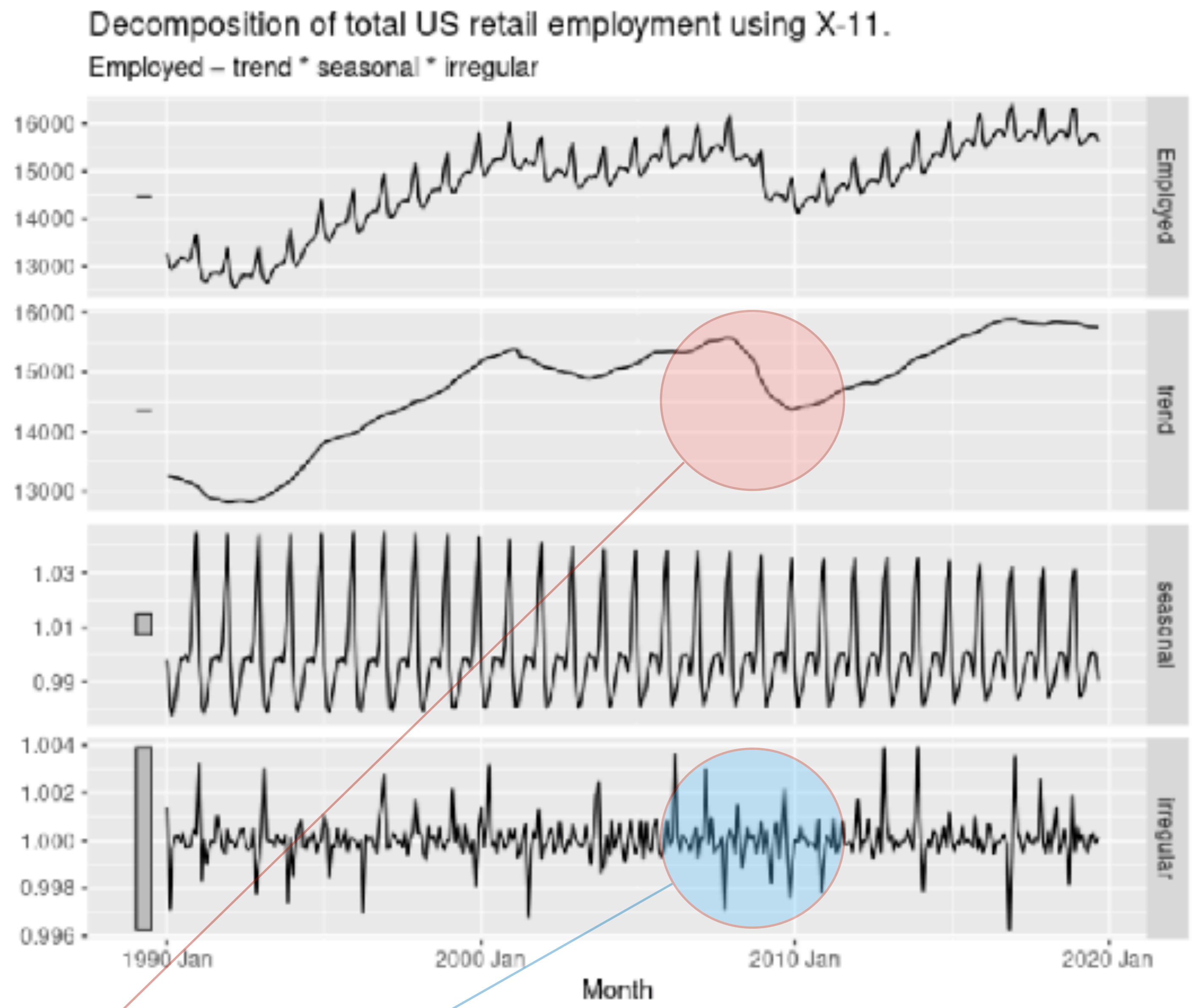
```
x11_dcmp <- us_retail_employment |>
    model(x11 = X_13ARIMA_SEATS(Employed ~ x11())) |>
    components()
autoplot(x11_dcmp) +
    labs(title =
             "Decomposition of total US retail employment using X-11.")
```

X_13_ARIMA_SEATS is a latest X-11 implementation which is multiplicative in default.



STL vs. X-11 Method





Observations: The X-11 has captured the sudden fall in the data due to the 2007–2008 global financial crisis.

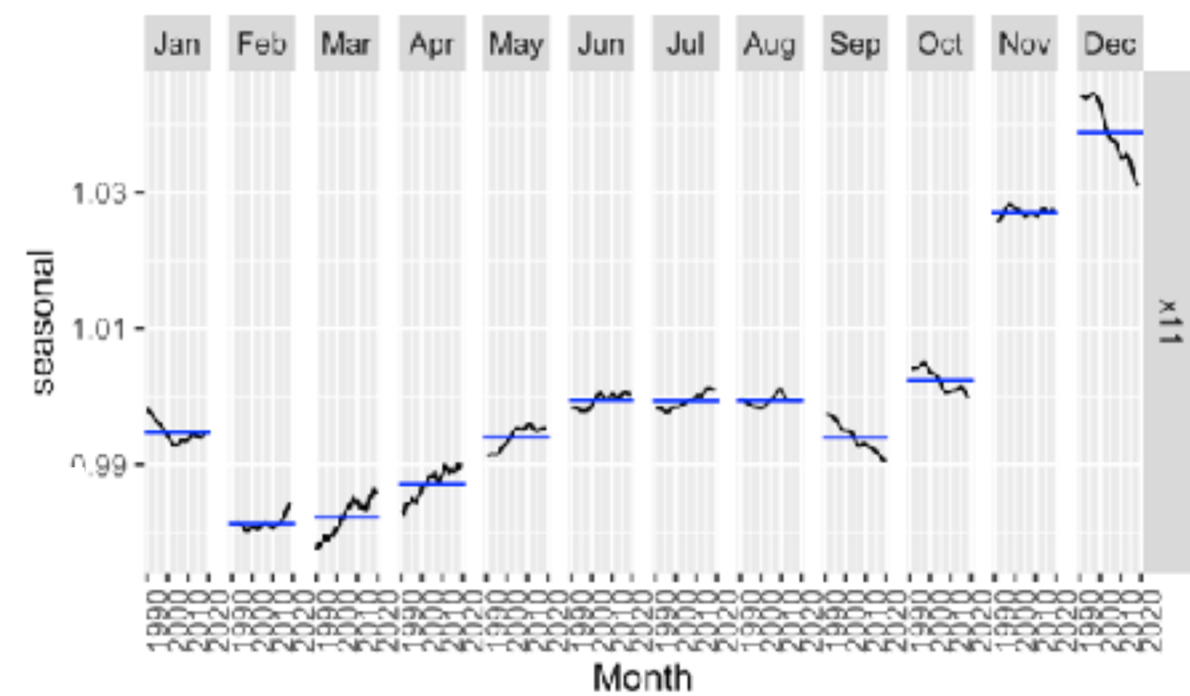
Effects of this crisis somehow leaked into the remainder component.

X-11 Seasonal component

hard to see.

Subseries is useful to see the seasonal variation through time.

```
x11_dcmp |>
    gg_subseries(seasonal)
```



SEATS Method

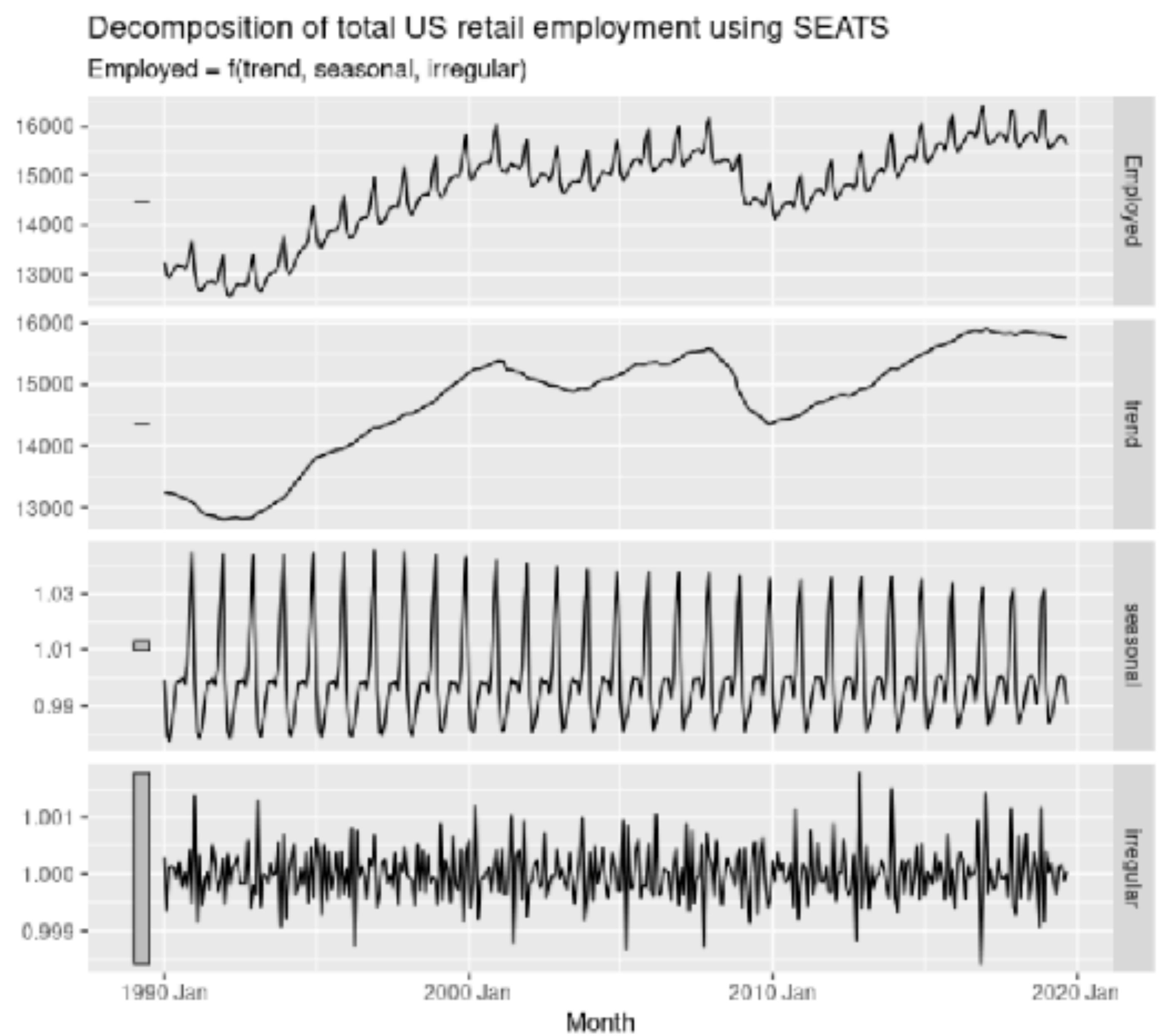
SEATS (Seasonal Extraction in ARIMA Time Series) was developed at the Bank of Spain, and is now widely used by government agencies around the world.

ARIMA models will be discussed in later weeks.

```
seats_dcmp <- us_retail_employment |>
    model(seats = X_13ARIMA_SEATS(Employed ~ seats())) |>
    components()
autoplot(seats_dcmp) +
    labs(title =
           "Decomposition of total US retail employment using SEATS")
```

See Dagum & Bianconcini (2016) for a detailed discussion.

Underlying technical details will be outside the scope of this course, but you will be using it in labs.



STL Decomposition

STL (Seasonal and Trend decomposition using Loess) is a versatile and robust method for decomposing time series developed by R. B. Cleveland et al. (1990).

STL has several advantages over classical decomposition, and the SEATS and X-11 methods:

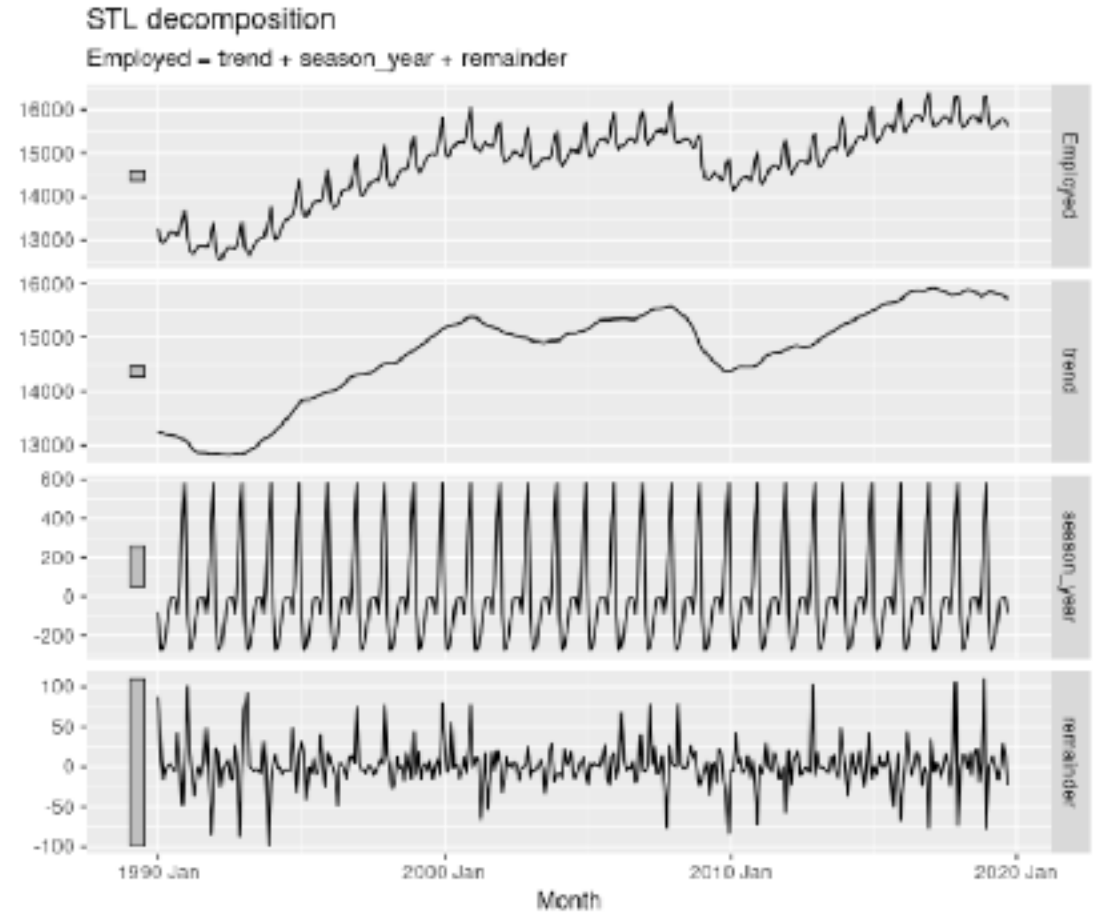
- STL can handle any type of seasonality (such as daily data, or hourly data, or weekly data) in contrast to SEATS and X-11.
- User can specify
  - The rate of change in the seasonal component.
  - The smoothness of the trend-cycle
  - A robust decomposition, so rare unusual observations will not affect the estimates of the trend-cycle and seasonal components, except the remainder component.

As disadvantages, it does not handle trading day or calendar variation automatically, and it only provides facilities for additive decompositions

Smaller values allow for more rapid changes.

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

Periodic means identical across years i.e., infinite.



Some Handy Time Series Features

Times Series Features in R

- The `feats` package includes functions for computing **FE**atures **A**nd **S**tatistics from **T**ime **S**eries
- We have already seen some time series features
    - Autocorrelations
    - Guerrero estimate of the Box-Cox
- In this part of the lecture, we will briefly mention few useful features that is helpful for investigation of the time series.

Some Handy `features()`

Any numerical summary computed from a time series is a feature of that time series e.g., the *mean*, *minimum* or *maximum*.

The `features` function can be used for computing many such features.

Means of all the time series in *tourism* data:

```
> tourism |>
+     features(Trips, list(mean = mean)) |>
+     arrange(mean)
# A tibble: 304 x 4
   Region          State              Purpose   mean
   <chr>           <chr>              <chr>    <dbl>
 1 Kangaroo Island South Australia    Other    0.340
 2 MacDonnell      Northern Territory Other    0.449
 3 Wilderness West Tasmania           Other    0.478
 4 Barkly          Northern Territory Other    0.632
 5 Clare Valley    South Australia    Other    0.898
 6 Barossa         South Australia    Other    1.02
 7 Kakadu Arnhem   Northern Territory Other    1.04
 8 Lasseter        Northern Territory Other    1.14
 9 Wimmera         Victoria           Other    1.15
10 MacDonnell      Northern Territory Visiting 1.18
# … with 294 more rows
```

Observe that the series with least average number of visits was “Other” visits to Kangaroo Island in South Australia.

Some Handy `features()`

One can also use `quantile()`, to get a simple five number/summary statistics:

> the *minimum*, first *quartile*, *median*, *third quartile* and *maximum*. These divide the data into four equal-size sections, each containing 25% of the data.

```
> tourism |> features(Trips, quantile)
# A tibble: 304 × 8
   Region         State              Purpose    `0%`  `25%`  `50%`  `75%` `100%`
   <chr>          <chr>              <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1 Adelaide       South Australia    Business  68.7   134.   153.   177.   242.
 2 Adelaide       South Australia    Holiday  108.    135.   154.   172.   224.
 3 Adelaide       South Australia    Other     25.9    43.9   53.8   62.5  107.
 4 Adelaide       South Australia    Visiting 137.    179.   206.   229.   270.
 5 Adelaide Hills South Australia    Business   0       0      1.26   3.92  28.6
 6 Adelaide Hills South Australia    Holiday    0       5.77   8.52  14.1   35.8
 7 Adelaide Hills South Australia    Other      0       0      0.908  2.09   8.95
 8 Adelaide Hills South Australia    Visiting   0.778   8.91  12.2   16.8   81.1
 9 Alice Springs  Northern Territory Business   1.01    9.13  13.3   18.5   34.1
10 Alice Springs  Northern Territory Holiday    2.81   16.9   31.5   44.8   76.5
# … with 294 more rows
```

Some Handy `features()`
Autocorrelations

`feat_acf()` function computes a selection of the autocorrelations and create new time series of them.

Some examples are as follows:

- First ten squared autocorrelations can be useful to understand the amount of autocorrelation in the series, regardless of the lags.

- Autocorrelations of the differences in the series between periods i.e., a new time series consisting of the differences between consecutive observations.

- Autocorrelations of these differences.

- One can re-apply "*difference*"-ing again i.e., the differences of differences.

- And take a look at the autocorrelations of this double-differenced series.

- Similar can be done to seasonal data e.g., differencing between consecutive Januaries or other months.

Some Handy `features()`

Autocorrelations

`feat_acf()` function computes a selection of the aforementioned autocorrelations

```
> tourism |> features(Trips, feat_acf)
# A tibble: 304 × 10
   Region         State              Purpose      acf1 acf10 diff1_¹ diff1…² diff2…³ diff2…⁴ seaso_⁵
   <chr>          <chr>              <chr>       <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 Adelaide       South Australia    Busine…   0.0333  0.131  -0.520   0.463  -0.676   0.741  0.201
 2 Adelaide       South Australia    Holiday   0.0456  0.372  -0.343   0.614  -0.487   0.558  0.351
 3 Adelaide       South Australia    Other     0.517   1.15   -0.409   0.383  -0.675   0.792  0.342
 4 Adelaide       South Australia    Visiti…   0.0684  0.294  -0.394   0.452  -0.518   0.447  0.345
 5 Adelaide Hills South Australia    Busine…   0.0709  0.134  -0.580   0.415  -0.750   0.746 -0.0628
 6 Adelaide Hills South Australia    Holiday   0.131   0.313  -0.536   0.500  -0.716   0.906  0.208
 7 Adelaide Hills South Australia    Other     0.261   0.330  -0.253   0.317  -0.457   0.392  0.0745
 8 Adelaide Hills South Australia    Visiti…   0.139   0.117  -0.472   0.239  -0.626   0.408  0.170
 9 Alice Springs  Northern Territ…   Busine…   0.217   0.367  -0.500   0.381  -0.658   0.587  0.315
10 Alice Springs  Northern Territ…   Holiday  -0.00660 2.11   -0.153   2.11   -0.274   1.55   0.729
# … with 294 more rows, and abbreviated variable names ¹diff1_acf1, ²diff1_acf10, ³diff2_acf1,
#   ⁴diff2_acf10, ⁵season_acf1
# ℹ Use `print(n = ...)` to see more rows
```

From left to right:

- the first autocorrelation coefficient
- the sum of squares of the first ten autocorr. coeff.
- the first autocorr. coef. from the differenced data
- the sum of sq. of the first ten autocorr. coeff. from the diff. data
- the first autocorr. coef.t from the twice diff. data
- the sum of sq. of the first ten autocor. coeff. from the 2 x diff. data
- the autocorr. coeff. at the first seasonal lag (for seasonal data).

STL Features

Motivation

A time series decomposition can also be used to measure the strength of trend and seasonality in a time series.

Basic Intuition

Assume a(n additive) decomposition $y_t = T_t + S_t + R_t$

For strongly trended data, the seasonally adjusted data should have much more variation than the remainder component:

$Var(R_t)/Var(T_t + R_t)$ should be relatively small.

For data with little or no trend, the two variances should be approximately the same.

Strength of Trend

$$F_T = \max\left(0, 1 - \frac{\mathrm{Var}(R_t)}{\mathrm{Var}(T_t + R_t)}\right)$$

Strength of Seasonality

$$F_S = \max\left(0, 1 - \frac{\mathrm{Var}(R_t)}{\mathrm{Var}(S_t + R_t)}\right)$$

Especially useful when you need to find the series with the most trend or the most seasonality from a large collection of time series.

STL Features

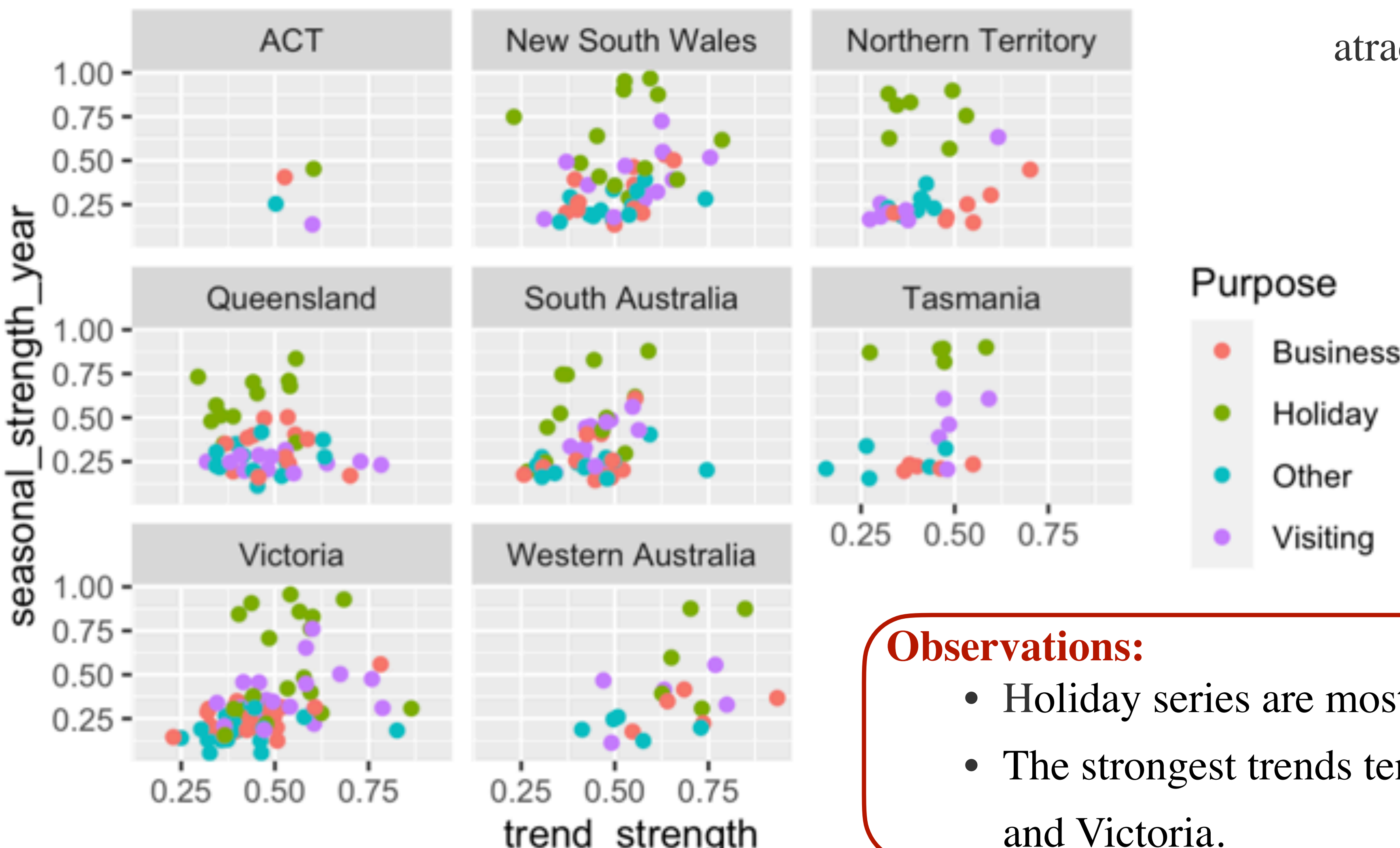
Such STL based features can be computed using the `feat_stl()` function.

```
> tourism |>
+     features(Trips, feat_stl)
# A tibble: 304 × 12
   Region        State Purpose trend…¹ seaso…² seaso…³ seaso…⁴ spiki…⁵ linea…⁶ curva…⁷ stl_e…⁸
   <chr>         <chr> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 Adelaide      Sout… Busine…   0.464   0.407       3       1 1.58e+2  -5.31    71.6    -0.532
 2 Adelaide      Sout… Holiday   0.554   0.619       1       2 9.17e+0  49.0     78.7    -0.510
 3 Adelaide      Sout… Other     0.746   0.202       2       1 2.10e+0  95.1     43.4    -0.351
 4 Adelaide      Sout… Visiti…   0.435   0.452       1       3 5.61e+1  34.6     71.4    -0.501
 5 Adelaide Hil… Sout… Busine…   0.464   0.179       3       0 1.03e-1   0.968   -3.22   -0.600
 6 Adelaide Hil… Sout… Holiday   0.528   0.296       2       1 1.77e-1  10.5     24.0    -0.481
 7 Adelaide Hil… Sout… Other     0.593   0.404       2       2 4.44e-4   4.28     3.19   -0.298
 8 Adelaide Hil… Sout… Visiti…   0.488   0.254       0       3 6.50e+0  34.2     -0.529  -0.472
 9 Alice Springs Nort… Busine…   0.534   0.251       0       1 1.69e-1  23.8     19.5    -0.492
10 Alice Springs Nort… Holiday   0.381   0.832       3       1 7.39e-1 -19.6     10.5    -0.522
# … with 294 more rows, 1 more variable: stl_e_acf10 <dbl>, and abbreviated variable names
#   ¹trend_strength, ²seasonal_strength_year, ³seasonal_peak_year, ⁴seasonal_trough_year,
#   ⁵spikiness, ⁶linearity, ⁷curvature, ⁸stl_e_acf1
```

STL Features: `feat_stl`

We can then use these features in plots to identify the heavily trended or strongly seasonal:

```
tourism |>
    features(Trips, feat_stl) |>
    ggplot(aes(x = trend_strength, y = seasonal_strength_year,
               col = Purpose)) +
    geom_point() +
    facet_wrap(vars(State))
```



`feat_stl` has many more atractive features.

Read the Book!

Observations:

- Holiday series are most seasonal which is unsurprising
- The strongest trends tend to be in Western Australia and Victoria.

Bibliography

Hyndman, Rob J., and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. (Chapters 3 & 4)

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–33.

Dagum, E. B., & Bianconcini, S. (2016). *Seasonal adjustment methods and real time trend-cycle estimation*. Springer.