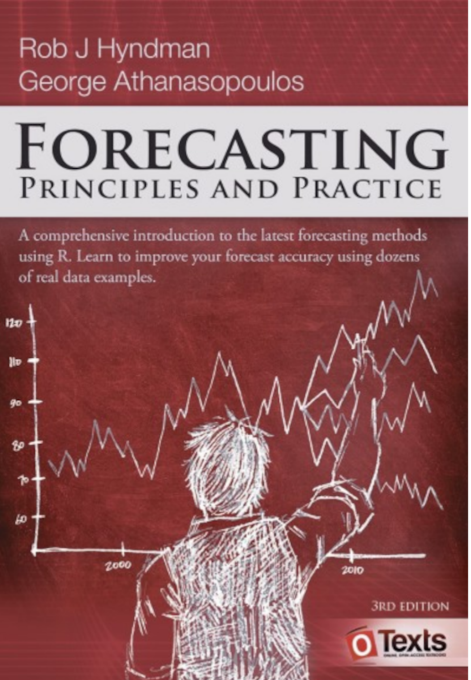


3. Time series decomposition

3.6 STL decomposition

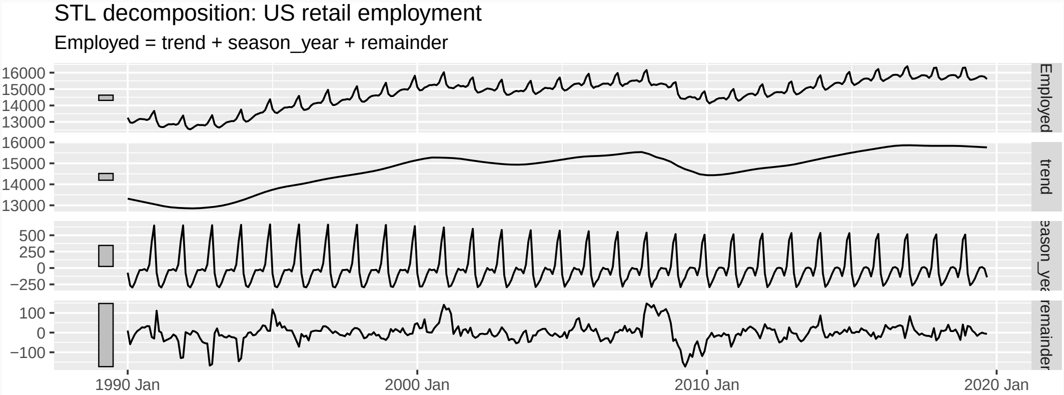
OTexts.org/fpp3/

STL decomposition

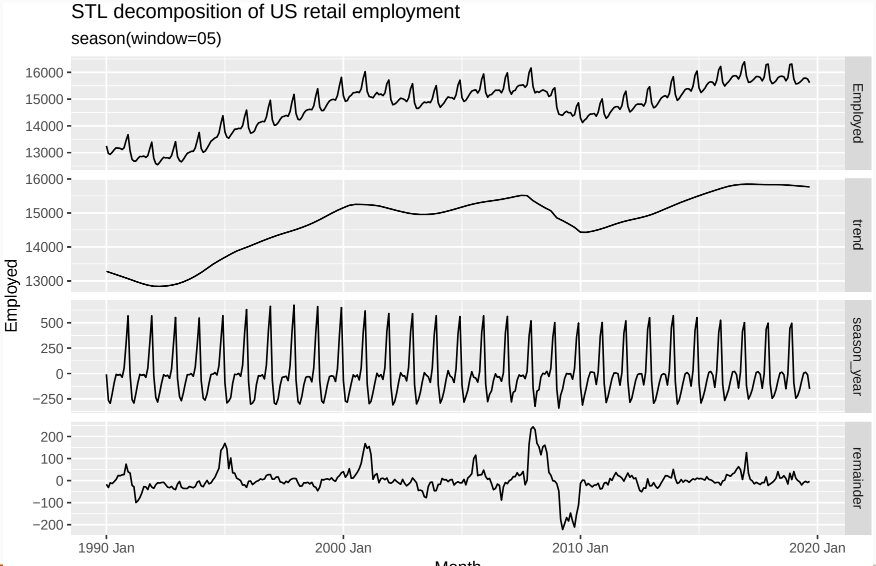
- STL: “Seasonal and Trend decomposition using Loess”
- Very versatile and robust.
- Unlike X-12-ARIMA, STL will handle any type of seasonality.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Robust to outliers
- 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 = 9), robust = TRUE)) |>
  components() |>
  autoplot() + labs(title = "STL decomposition: US retail employment")
```



STL decomposition



STL decomposition

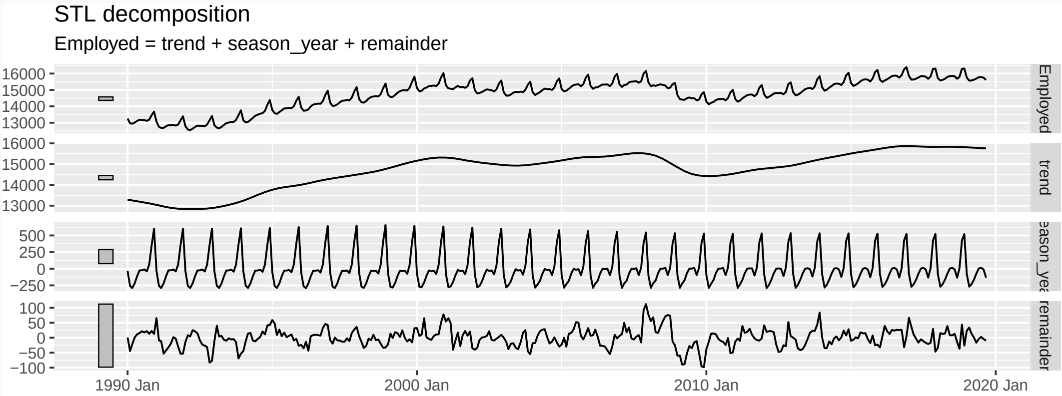
```
us_retail_employment |>
  model(STL(Employed ~ season(window = 5))) |>
  components()

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

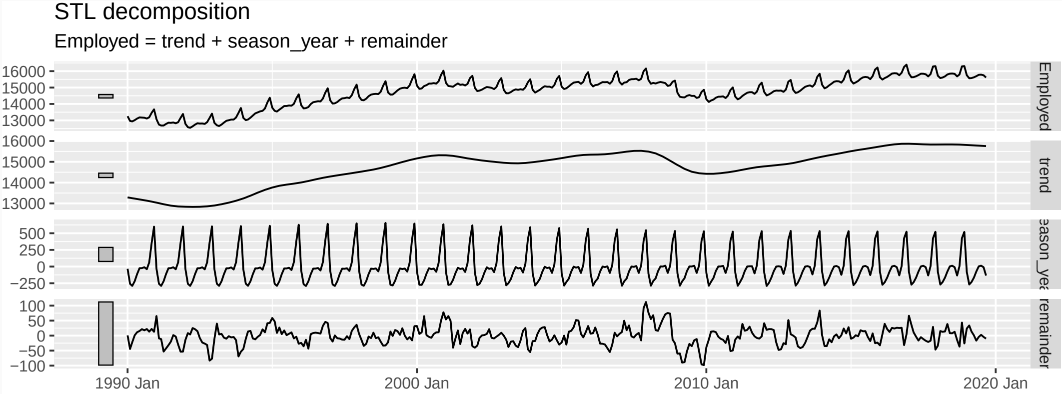
```r
us_retail_employment |>
  model(STL(Employed)) |>
  components() |>
  autoplot()
```



STL decomposition

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

```
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.
- The trend window controls loess bandwidth applied to deasonalised values.
- The season window controls loess bandwidth applied to detrended subseries.
- Robustness weights based on remainder.
- Default season: `window = 13`
- Default trend: `window = nextodd(ceiling((1.5*period)/(1-(1.5/s.window)))`