

# Exploratory time series analysis using R



## 2. Visualising trend and seasonality

# Outline

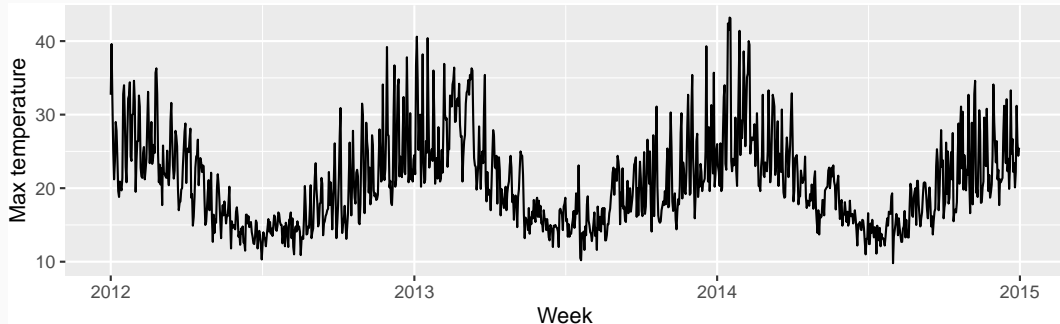
- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

# Outline

- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

# Time plots: autoplot( )

```
maxtemp <- vic_elec ▷  
  index_by(Day = date(Time)) ▷  
  summarise(Temperature = max(Temperature))  
maxtemp ▷  
  autoplot(Temperature) +  
  labs(x = "Week", y = "Max temperature")
```

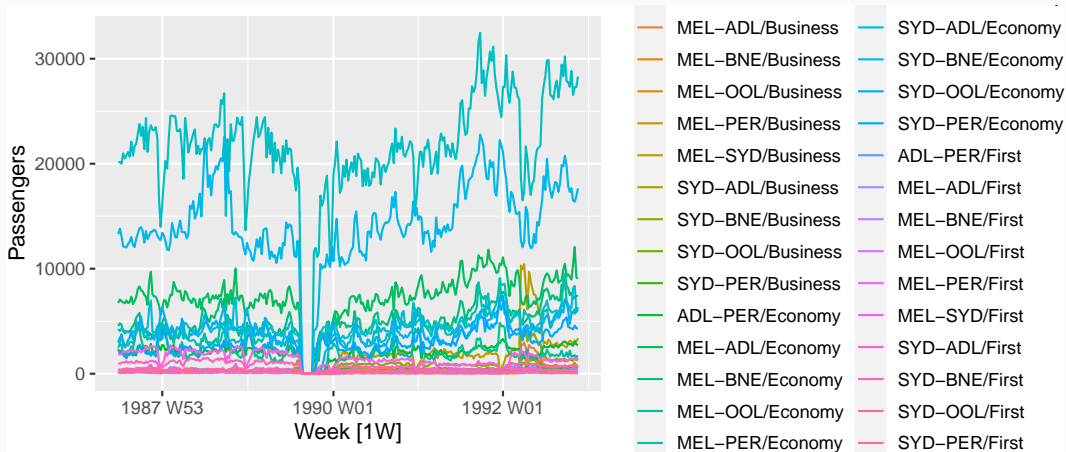


# Ansett airlines



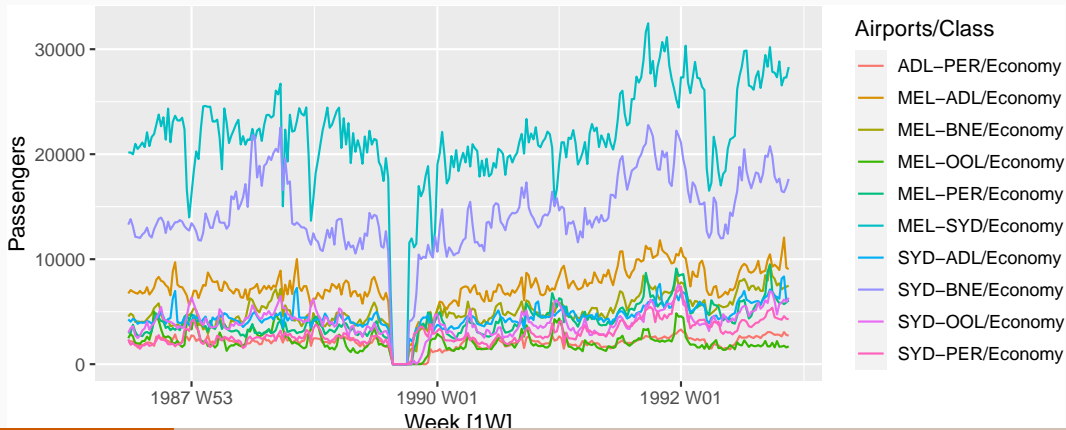
# Ansett airlines

```
ansett ▷  
autoplot(Passengers)
```



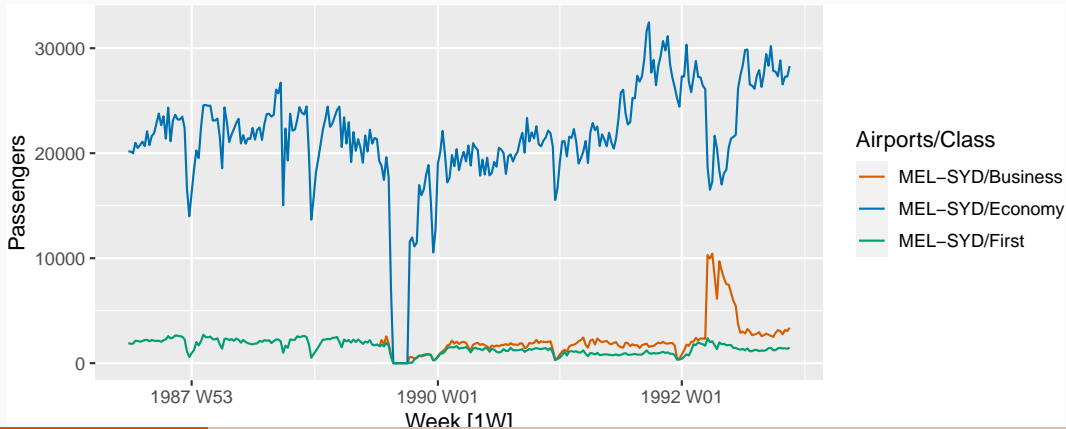
# Ansett airlines

```
ansett ▷  
  filter(Class = "Economy") ▷  
  autoplot(Passengers)
```



# Ansett airlines

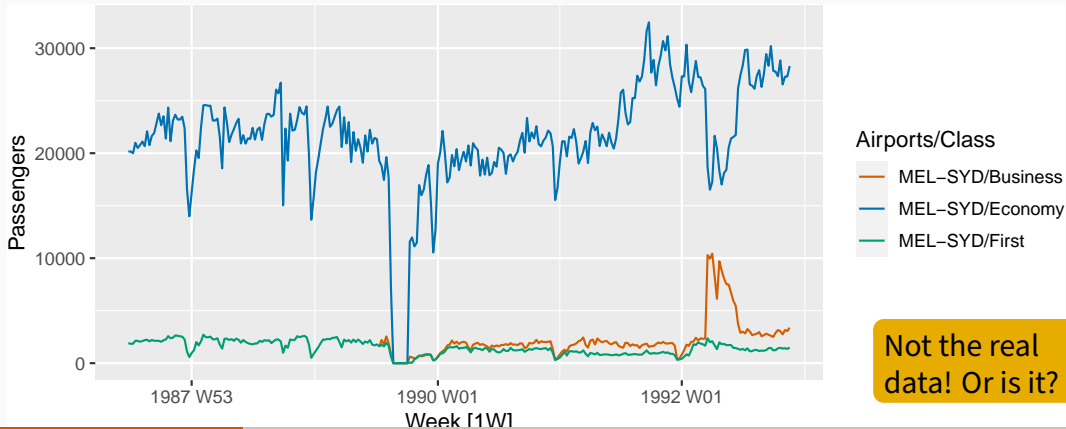
```
ansett ▷  
  filter(Airports = "MEL-SYD") ▷  
  autoplot(Passengers)
```





# Ansett airlines

```
ansett ▷  
  filter(Airports = "MEL-SYD") ▷  
  autoplot(Passengers)
```



Not the real  
data! Or is it?

# Outline

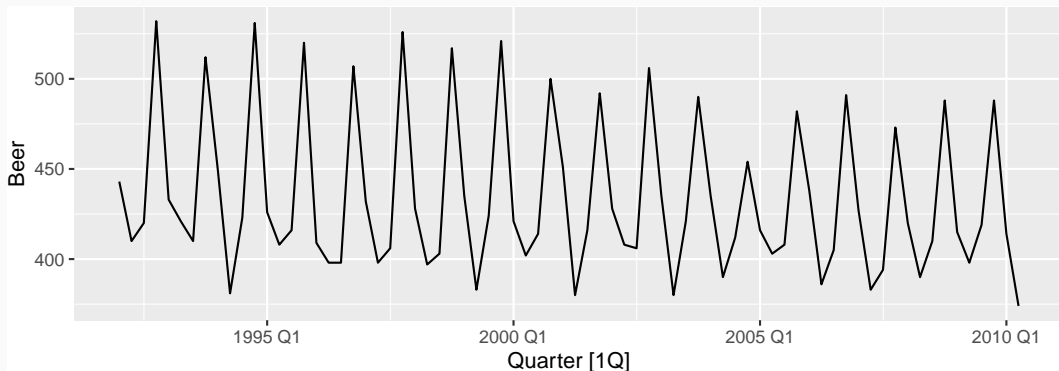
- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

## Seasonal plots: `gg_season()`

- Data plotted against the individual “seasons” in which the data were observed.
- Something like a time plot except that the data from each season are overlapped.
- Enables the underlying seasonal pattern to be seen more clearly, and also allows any substantial departures from the seasonal pattern to be easily identified.

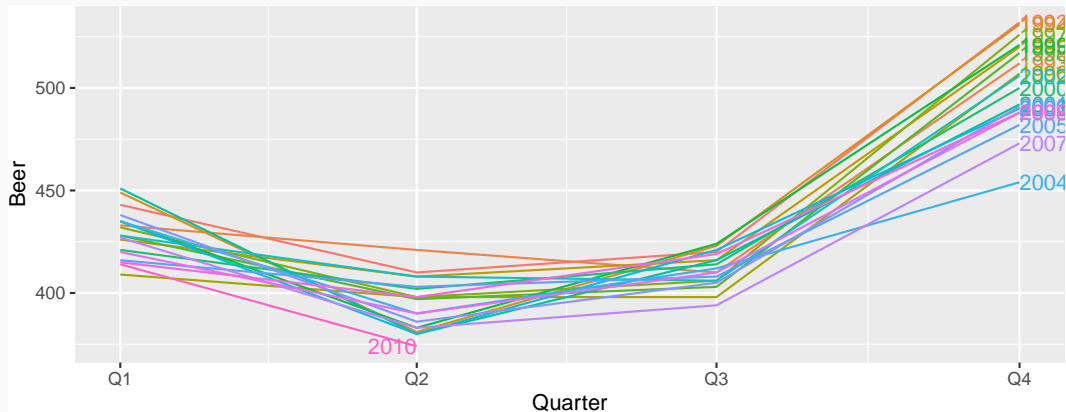
# Quarterly Australian Beer Production

```
beer <- aus_production ▷  
  select(Quarter, Beer) ▷  
  filter(year(Quarter) ≥ 1992)  
beer ▷ autoplot(Beer)
```



# Quarterly Australian Beer Production

```
beer ▷ gg_season(Beer, labels = "right")
```



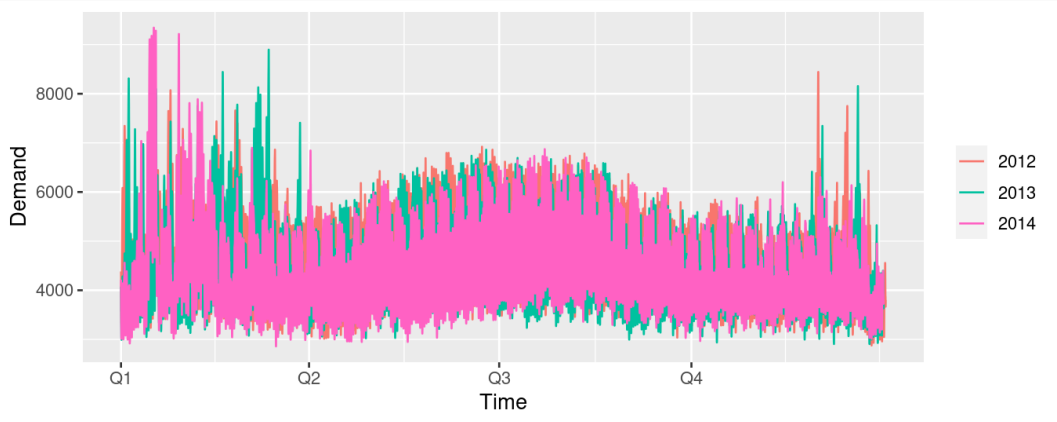
# Multiple seasonal periods

```
vic_elec
```

```
## # A tsibble: 52,608 x 5 [30m] <Australia/Melbourne>
##   Time                Demand Temperature Date        Holiday
##   <dtm>                <dbl>         <dbl> <date>      <lgl>
## 1 2012-01-01 00:00:00  4383.         21.4 2012-01-01 TRUE
## 2 2012-01-01 00:30:00  4263.         21.0 2012-01-01 TRUE
## 3 2012-01-01 01:00:00  4049.         20.7 2012-01-01 TRUE
## 4 2012-01-01 01:30:00  3878.         20.6 2012-01-01 TRUE
## 5 2012-01-01 02:00:00  4036.         20.4 2012-01-01 TRUE
## 6 2012-01-01 02:30:00  3866.         20.2 2012-01-01 TRUE
## 7 2012-01-01 03:00:00  3694.         20.1 2012-01-01 TRUE
## 8 2012-01-01 03:30:00  3562.         19.6 2012-01-01 TRUE
## 9 2012-01-01 04:00:00  3433.         19.1 2012-01-01 TRUE
## 10 2012-01-01 04:30:00  3359.         19.0 2012-01-01 TRUE
## # ... with 52,598 more rows
```

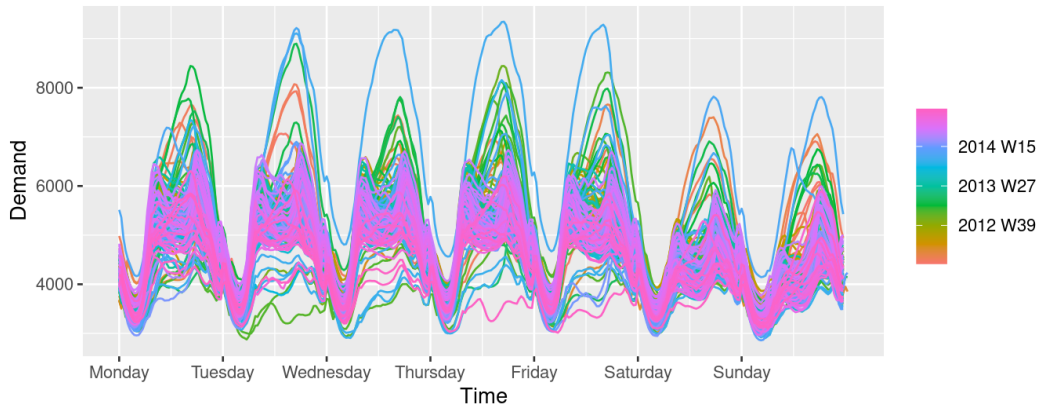
# Multiple seasonal periods

```
vic_elec > gg_season(Demand)
```



# Multiple seasonal periods

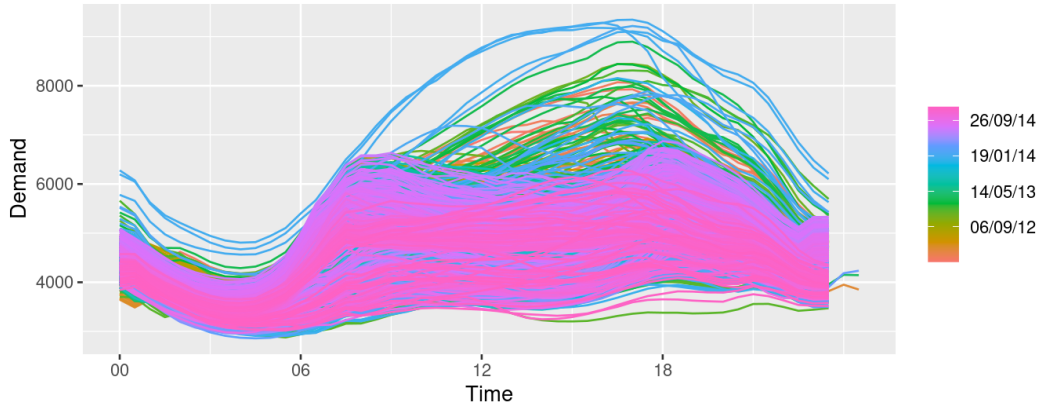
```
vic_elec > gg_season(Demand, period = "week")
```





# Multiple seasonal periods

```
vic_elec > gg_season(Demand, period = "day")
```

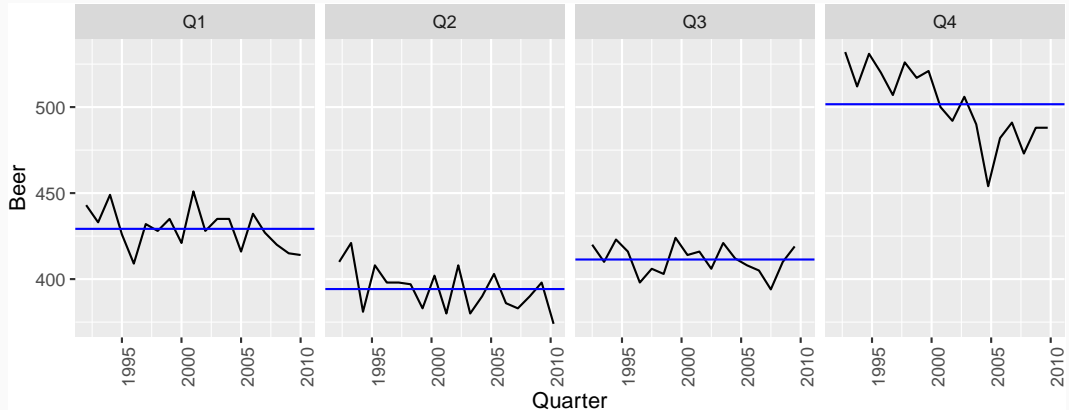


## Seasonal subseries plots: `gg_subseries()`

- Data for each season collected together in time plot as separate time series.
- Enables the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.

# Quarterly Australian Beer Production

```
beer > gg_subseries(Beer)
```



# Outline

- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

## Lab Session 2

Look at the quarterly tourism data for the Snowy Mountains

```
snowy <- tourism %>%  
  filter(Region == "Snowy Mountains")
```

- Use `autoplot( )`, `gg_season( )` and `gg_subseries( )` to explore the data.
- What do you learn?

# Outline

- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

# 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  $t$

# US Retail Employment

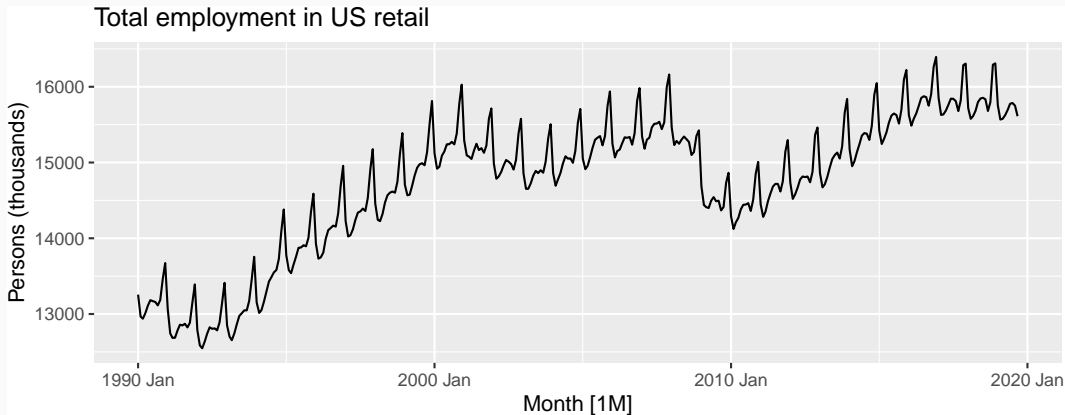
```
us_retail_employment <- us_employment %>%  
  filter(year(Month) ≥ 1990, Title == "Retail Trade") %>%  
  select(-Series_ID)  
us_retail_employment
```

```
## # A tibble: 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 13112.
```



# US Retail Employment

```
us_retail_employment ▷  
  autoplot(Employed) +  
  labs(y = "Persons (thousands)", title = "Total employment in US retail")
```



# US Retail Employment

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

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

# US Retail Employment

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

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

- STL: “Seasonal and Trend decomposition using Loess”

# US Retail Employment

```
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.
```

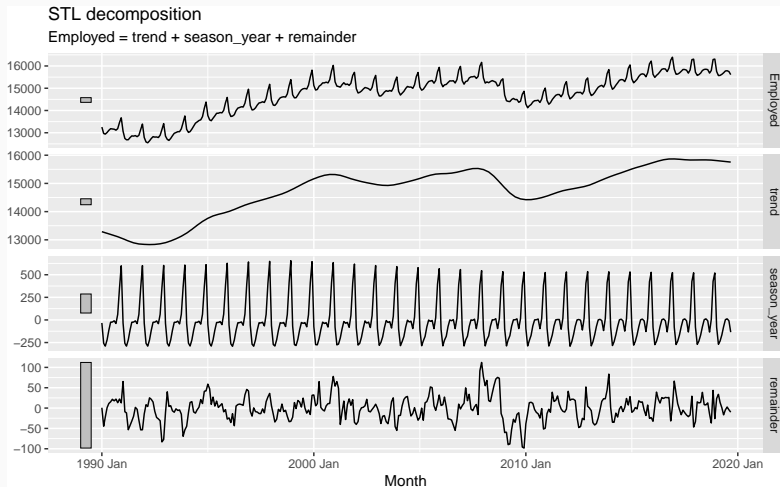
```
## 7 stl      1990 Jul      13170. 13172.      -23.2      21.6      13193.
```

```
## 8 stl      1990 Aug      13160. 13151.      -9.52      17.8      13169.
```

```
## 9 stl      1990 Sep      13113. 13131.      -39.5      22.0      13153.27
```

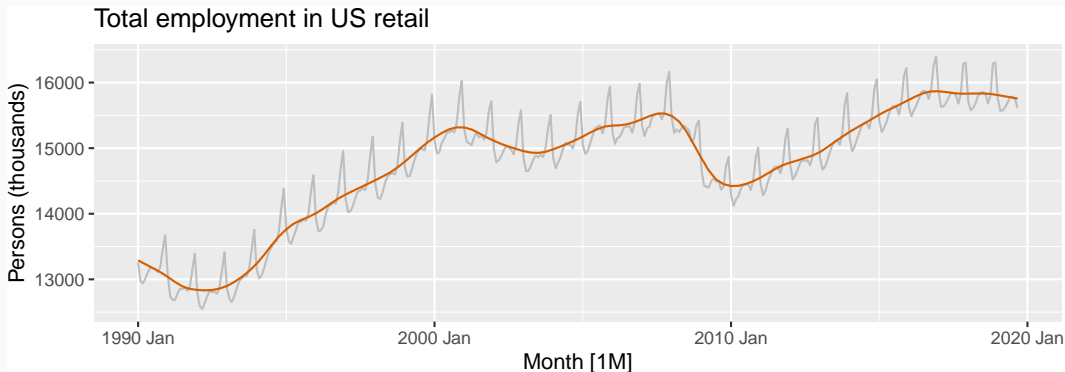
# US Retail Employment

```
components(dcmp) ▷ autoplot()
```



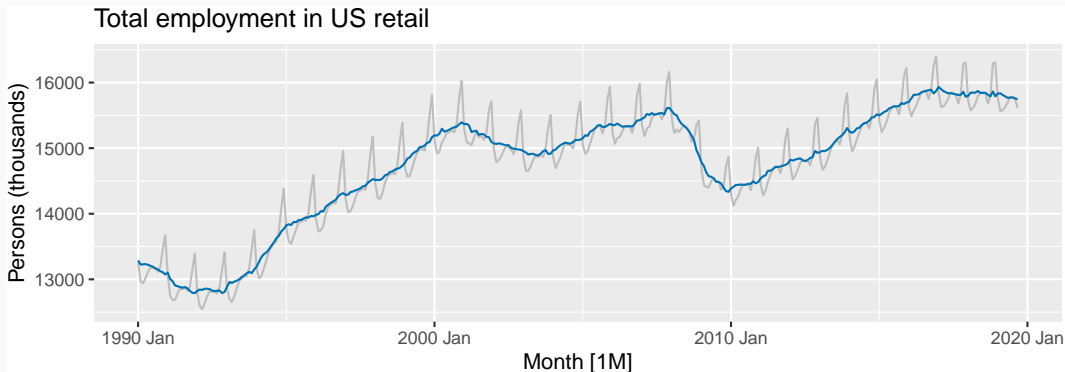
# US Retail Employment

```
us_retail_employment ▷  
  autoplot(Employed, color = "gray") +  
  autolayer(components(dcmp), trend, color = "#D55E00") +  
  labs(y = "Persons (thousands)", title = "Total employment in US retail")
```



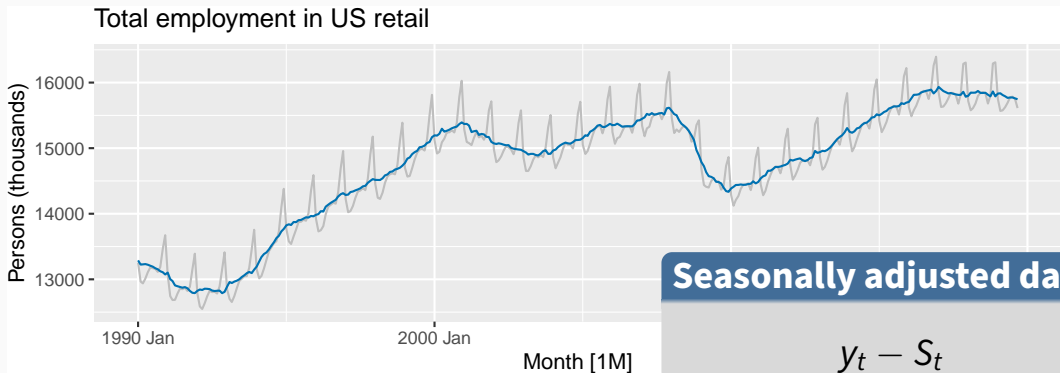
# US Retail Employment

```
us_retail_employment ▷  
  autoplot(Employed, color = "gray") +  
  autolayer(components(dcmp), season_adjust, color = "#0072B2") +  
  labs(y = "Persons (thousands)", title = "Total employment in US retail")
```



# US Retail Employment

```
us_retail_employment ▷  
  autoplot(Employed, color = "gray") +  
  autolayer(components(dcmp), season_adjust, color = "#0072B2") +  
  labs(y = "Persons (thousands)", title = "Total employment in US retail")
```





# STL decomposition

# STL decomposition

```
us_retail_employment ▷  
  model(STL(Employed ~ trend(window = 21) + season(window = 13),  
    robust = TRUE  
  )) ▷  
  components()
```

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

# 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 deseasonalised values.
- season window controls loess bandwidth on detrended subseries.
- Robustness weights based on remainder.
- Default season: window = 13
- Default trend: window =  
 $\text{nextodd}(\text{ceiling}((1.5 * \text{period}) / (1 - (1.5 / s.\text{window}))))$
- window values should be odd numbers for symmetry.

# Outline

- 1 Time plots
- 2 Seasonal plots
- 3 Lab Session 2
- 4 Time series decompositions
- 5 Lab Session 3

# Lab Session 3

- 1 Produce an STL decomposition of the Snowy Mountains data.
- 2 Experiment with different values of the two window arguments.
- 3 Plot the seasonally adjusted series.