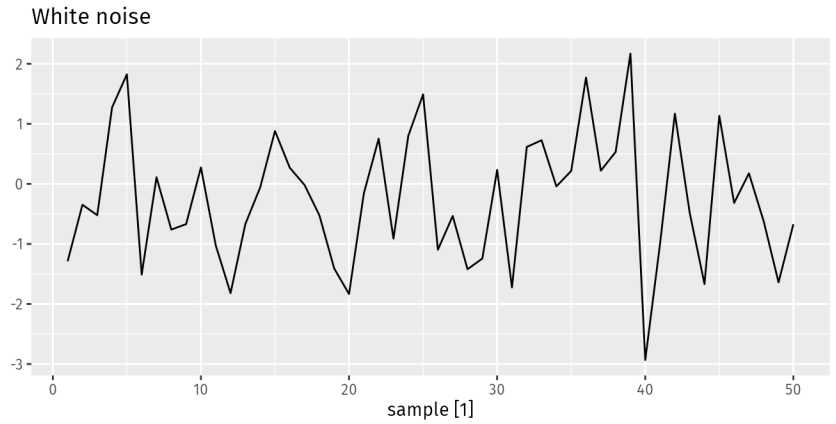
Predictive Analytics

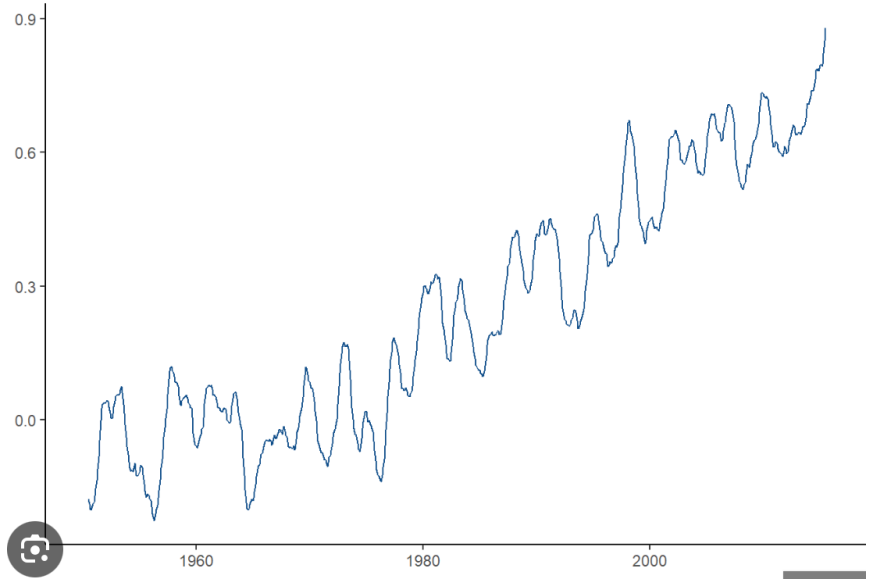
Lecture 2

Herdis Steingrimsdottir

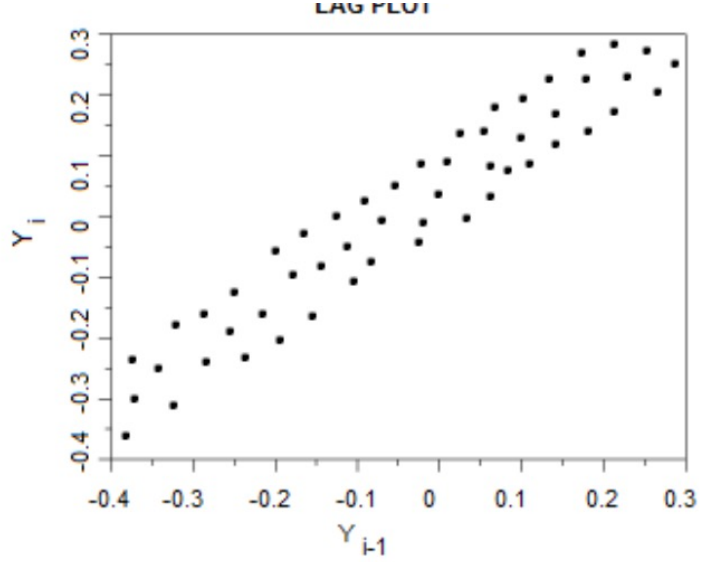
Yesterday

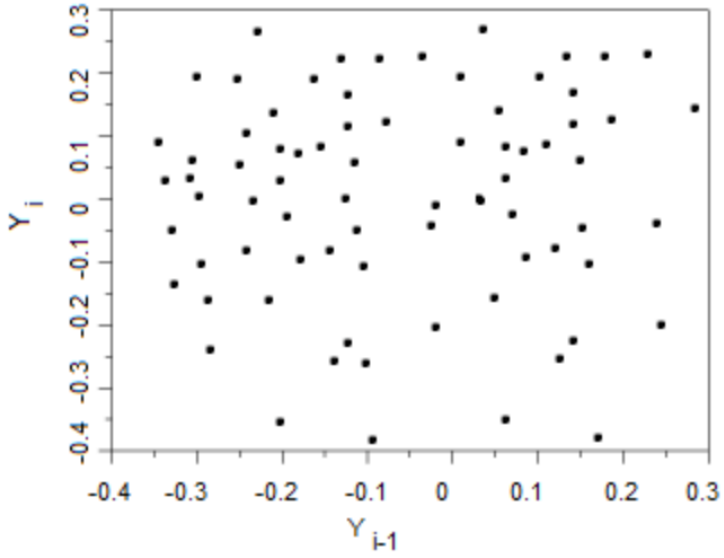
- Chapters 1 & 2
  - What is forecasting and what can we forecast?
  - Plotting time series data
  - Seasonality, Autocorrelation, White noise





Forecasting – how much does the past help?





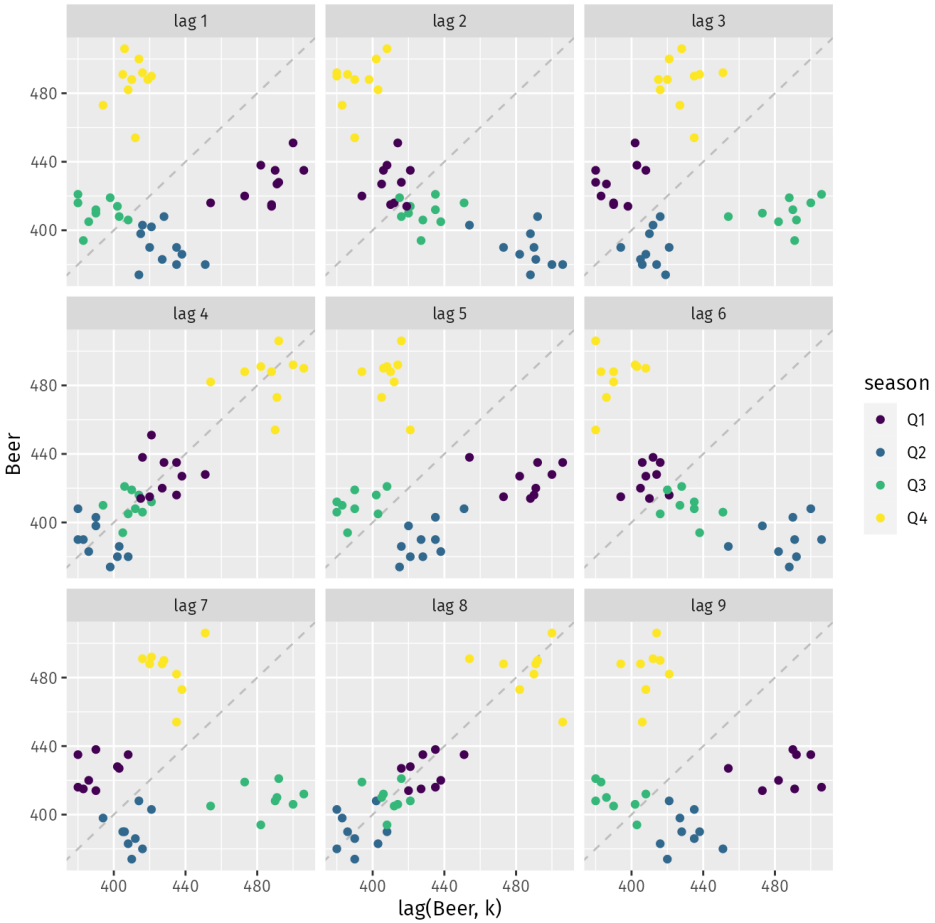
Lag plots

T	Y	Y, LAG = 1	Y, LAG = 4
1	3		
2	4	3	
3	1	4	
4	2	1	
5	7	2	3
6	2	7	4
7	0	2	1

Autocorrelation

To measure autocorrelation, we compare a time series to a lagged version of itself

A "lag" refers to the time shift in the variable



Lag plots

Plot the value of y against lagged value of y

Autocorrelation – how much do our time series remember the past?

- We just looked at some lag plots which gave us an idea of how our time series values are correlated with past values of the variable
- **Autocorrelation** measures how correlated a time series value is with its own past value. E.g., how correlated is the temperature today with the temperature yesterday?

Autocorrelation

- If past values strongly predict future values, autocorrelation is high
- If past values do very little to help predict future values, autocorrelation is low
- Autocorrelation is important factor in helping us determine whether past patterns can be used to predict future values.

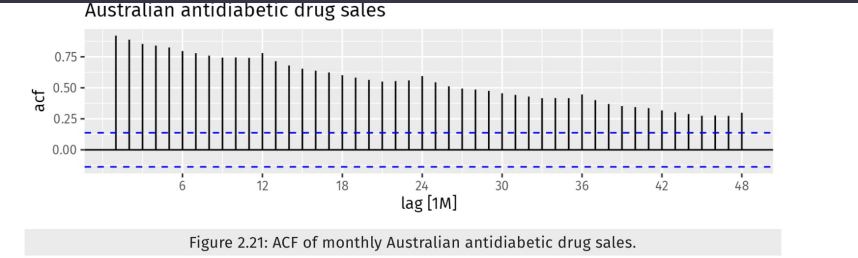


Figure 2.21: ACF of monthly Australian antidiabetic drug sales.

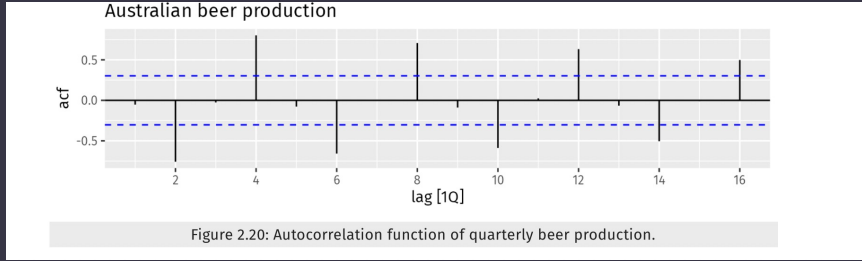


Figure 2.20: Autocorrelation function of quarterly beer production.

Autocorrelation function (ACF)

- The autocorrelation coefficients make up the **autocorrelation function** or the **ACF**
- ACF can help us visualize the autocorrelation patterns in our time series
- ACF shows us how strongly each value in a time series is related to its past values at different lags

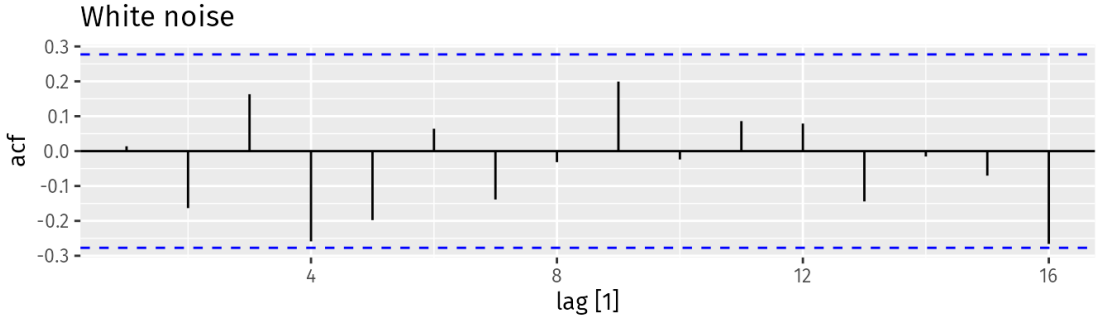
White noise

With white noise we expect each autocorrelation to be close to zero

But each will not be exactly equal to zero because of random variation

We expect 95% of the spikes in the ACF to lie within the confidence interval

We use the dotted lines in the ACF graphs to notate the boundaries of the confidence interval, if substantially more than 5% of the spikes are outside the bounds, the series is probably not **white noise**



Discuss

- Can you think of something in your own life that follows a time pattern?

- What kind of time series would it be? Would it be easy for you to forecast the future values based on the past? Why/Why not?

Visualizing the time series data

Time series graphics

We already saw some time plots and lag plots. There are various other ways to visualise the data, depending on what you want to highlight.

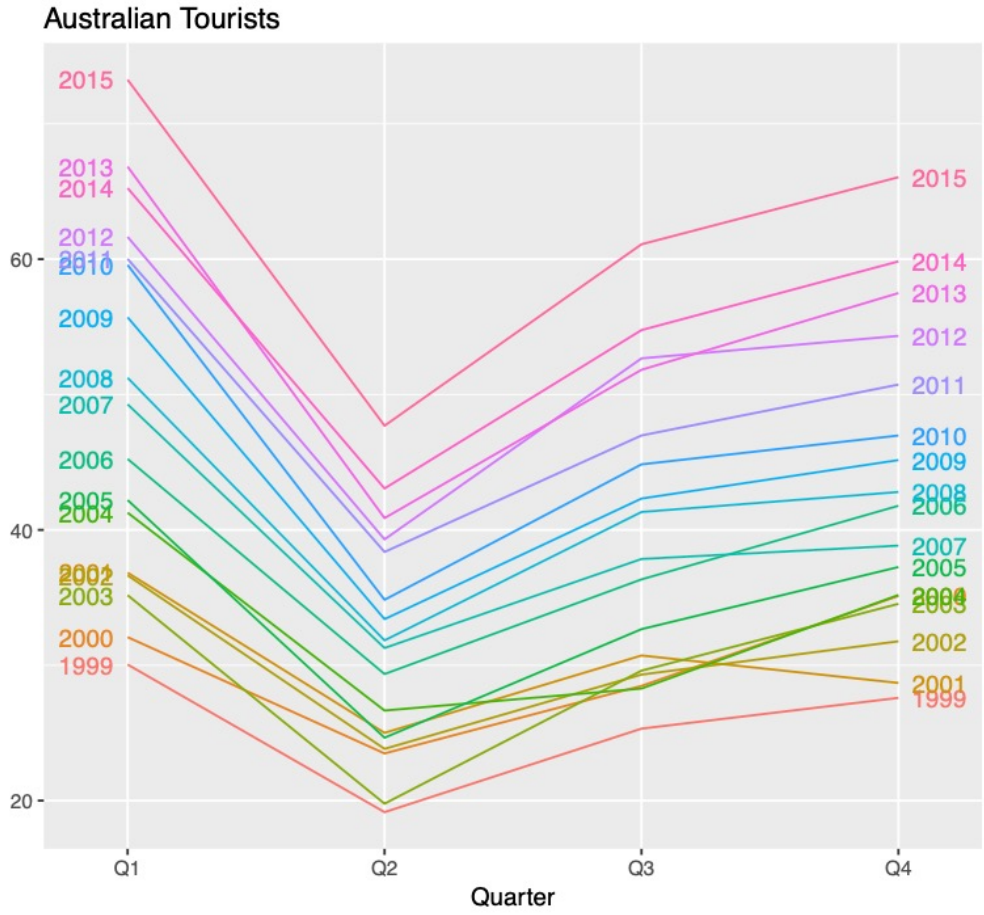
If we are interested in learning about the seasonality we may e.g. Look at seasonal plots or seasonal subseries plots

Seasonal plots

We now have the season on the x axis and overlap data from each year. Why would this be helpful?

Is there anything that we can see here that was not clear in the previous graph (i.e. the time plot)?

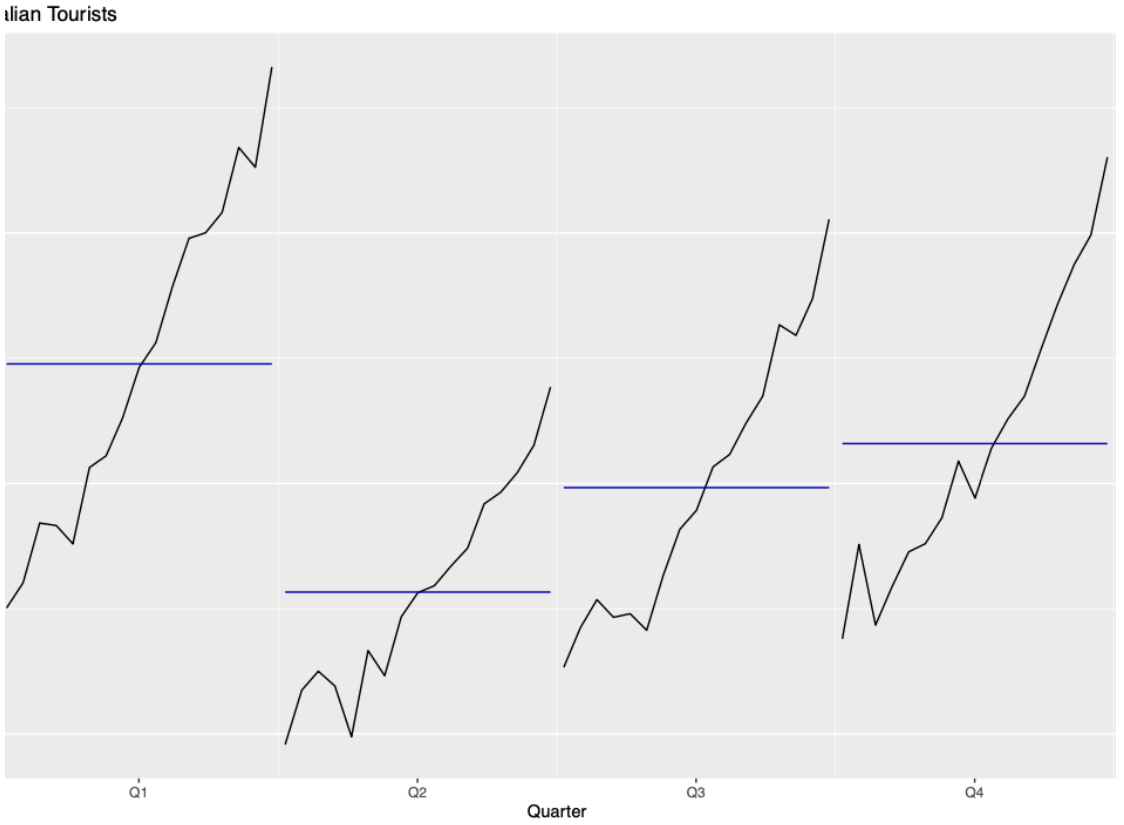
In general it can allow us to see seasonal patterns more clearly, and identify years in which the pattern changes



Seasonal subseries plots

An alternative way to emphasize seasonal patterns is to collect the data for each season in a separate mini time plot.

It can be useful e.g. to see changes in seasonality over time



To sum up

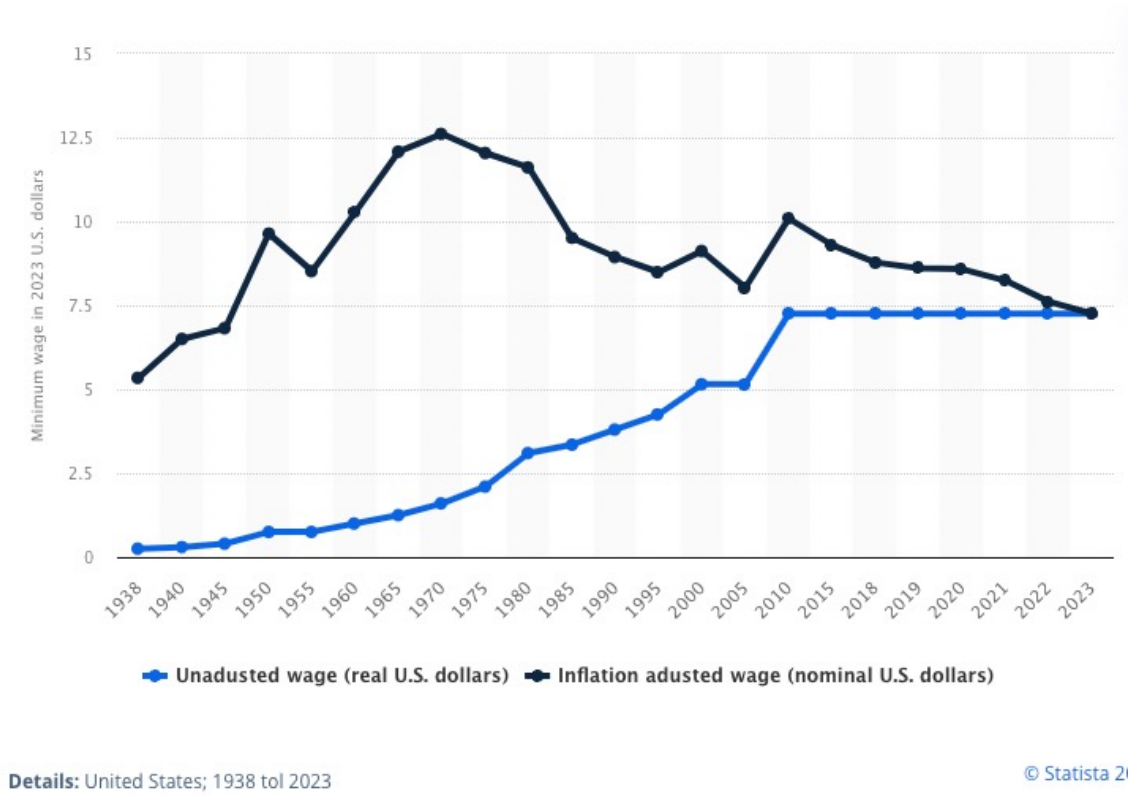
- Forecasting means we use data and models to predict what the future will look like
- There is always some uncertainty in our forecasts
- We often rely on time series data to do our forecasts
- The first thing to do in any data analysis is to **plot the data**
- We look for the obvious patterns such as trends and seasonality – this will guide us in how we model the data. We also want to check whether there are some unusual periods/observations in our data.
- **Autocorrelation** tells us how correlated the observations are over time, i.e. how predictive the past is for the future
- **White noise** is a time series where there is no systematic structure and no autocorrelation

NOW

- Chapters 3 & 4
- **Raw data → Adjust → Decompose → Summarise**
- Today we want to go from raw data to structure
  - Why can raw time series be misleading and how can we adjust the data?
  - How can we separate signal from noise?
  - How can we summarise the information in our time series?

Adjusting our data

Raw time series data can sometimes be misleading



Why do we adjust time series data

- We want to remove the variation in our data that we do not care about
- Common examples:
- Price changes/inflation
- Calendar effects
- Population

Comparing Prices over time

- In the US a big mac meal sold for $2.59 in 1985
- In 2023 the price was $8.64
- These prices are *nominal*, i.e. they tell us how much things cost at the time they are sold/bought
- To make a meaningful comparison we often want to look at *real prices*, i.e. prices that take into account that the price levels have changed
- To convert nominal values to real values we multiply the nominal value by the change in the price index:

$$P_{BASE} = P_t \times \frac{CPI_{BASE}}{CPI_t}$$

Real vs nominal values

- E.g. to compare the price of a big mac meal in 1985 and 2023 we can adjust the 1985 price for inflation using the CPI:

- $P_{1985} \times \frac{CPI_{2023}}{CPI_{1985}} = 2.59 \times \frac{299.2}{105.5} = \$7{,}35$

- Since the actual price in 2023 was $8.64 it means that a big mac meal is more expensive than in 1985, even after adjusting for inflation

Inflation adjustments

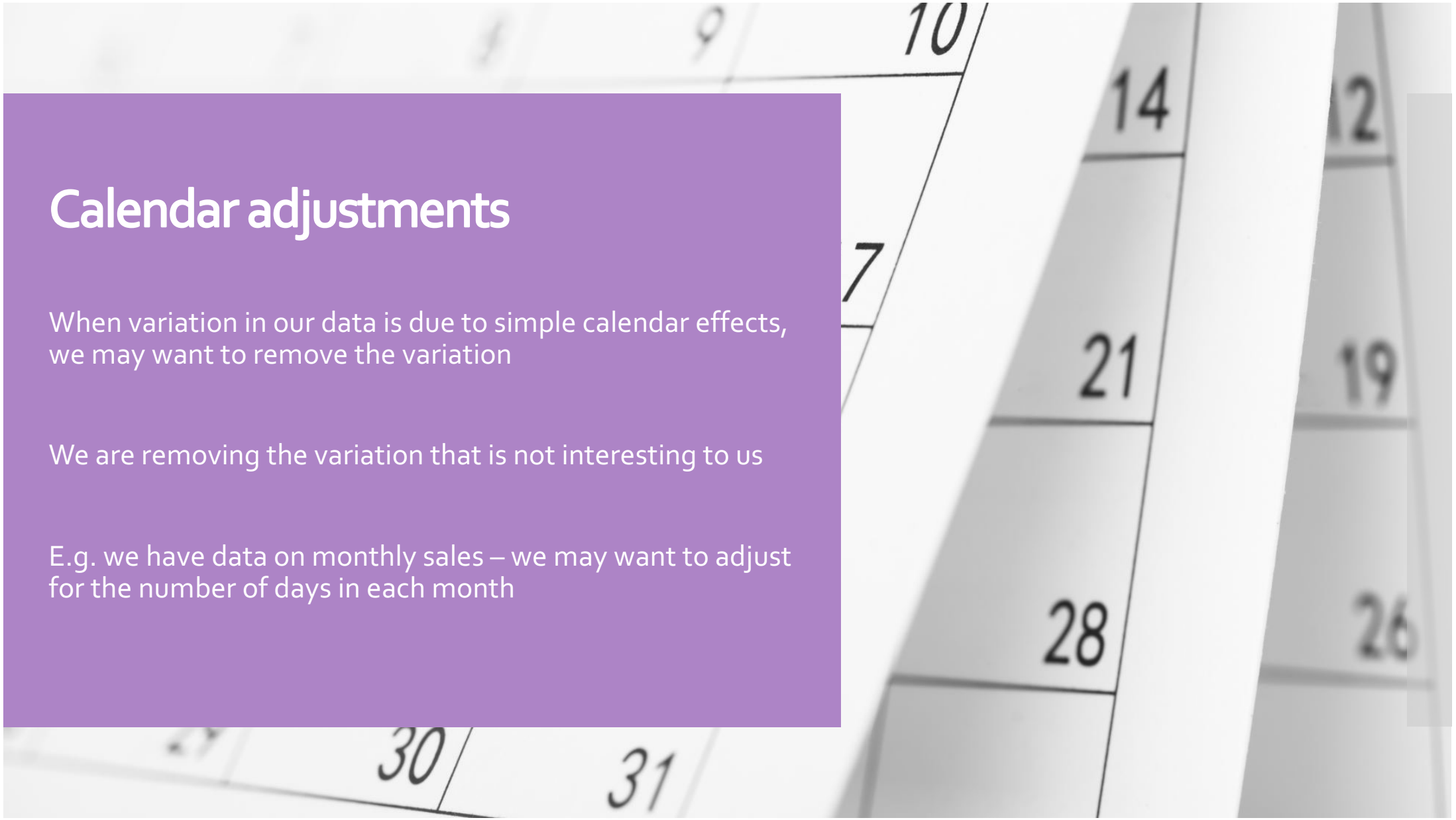
- Financial time series are usually adjusted such that all values are stated in a particular currency from a particular year, e.g. all amounts are in 2016 USD, or 2020 DKK etc.…
- To make these adjustments we use so called price indexes, e.g. the Consumer Price Index (CPI)
- E.g. lets say you have a time series for health care spending in Denmark. To adjust for price changes you want to have all expenditures, E_t, in 2020 values.
- $E_{2020} = E_t \times \frac{I_{2020}}{I_t}$
- **Ware trying to isolate the change that we are interested in – by getting rid of the change that is due to inflation**

Calendar adjustments

When variation in our data is due to simple calendar effects, we may want to remove the variation

We are removing the variation that is not interesting to us

E.g. we have data on monthly sales – we may want to adjust for the number of days in each month



Population adjustments

In a similar way we may have time series that are affected by population changes. E.g. GDP is partly growing over time because the population is growing – same with health care expenditures

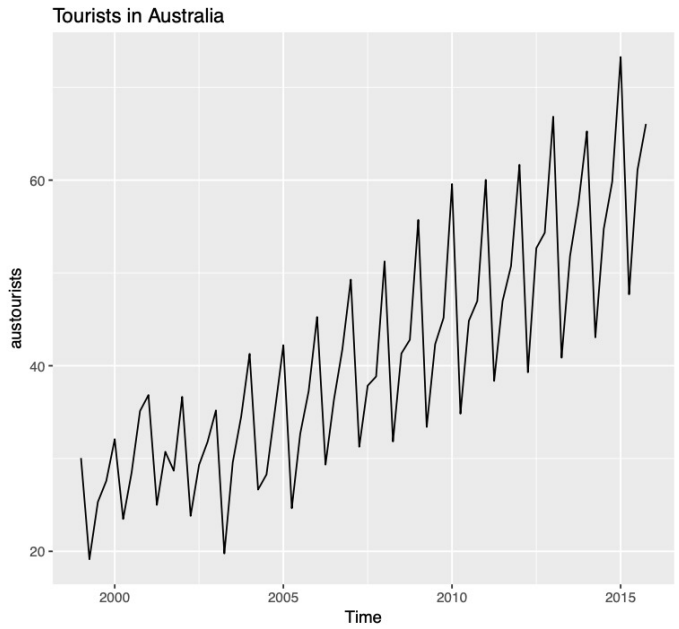
A solution is to adjust the data by the population and look at measures per-capita

Again the idea is to focus on, or isolating, the variation that we are interested in forecasting

Adjusted data

- The adjustments we discussed remove external distortions in our time series (inflation, population growth etc.)
- Now we start looking at the internal structure, i.e., we open the black box and start separating the underlying components

Decomposition: separating signal from noise



Decomposition

Decomposition

- After we have transformed our time series (if needed) we can start exploring the underlying components of the time series, such as trends, seasonality and random fluctuations
- **Decomposition** allows us to separate these components to better understand and model the data
- After the decomposition, we can analyze each component individually which helps us to apply forecasting methods more effectively

Time Series Decomposition

- In last lecture we discussed three types of time series patterns: (1) trend (2) seasonality and (3) cycles
- We now want to look at how we can decompose our time series into (1) trend-cycle component (2) a seasonal component and (3) a remainder component
- Time series decomposition
  - Improves our understanding of the time series
  - Can improve our forecasting models

Time series components

- We think of the time series as Trend+Seasonality+Remainder

- Trend-Cycle component, T_t
- Seasonal component, S_t
- Remainder component, R_t

- Additive decomposition: $y_t = S_t + T_t + R_t,$

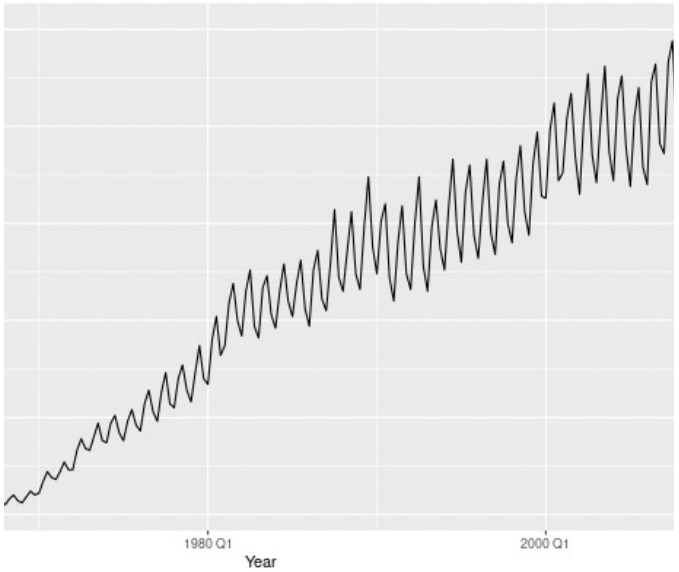
Decomposition: Additive or multiplicative?

- If magnitude of seasonal fluctuations depends on the level of the series, e.g., seasonality gets bigger with time → multiplicative decomposition

Additive	Multiplicative
Seasonal effect constant size	Seasonal effect grows with level
$Y = T + S + R$	$Y = T \times S \times R$

Additive or multiplicative

- When decomposing our TS we need to decide whether to use an **additive** or **multiplicative** model
- The key difference is how the components (trend, seasonality, and residuals) combine
- **Additive decomposition:** when each component contributes a fixed amount. This works well for stable seasonal patterns that do not grow larger as the trend increases.
- **Multiplicative decomposition:** when seasonal fluctuations grow larger when the trend increases
- Additive decomposition is simpler. We can use log transformation to convert a multiplicative relationship into and additive one, making the decomposition and analysis easier.



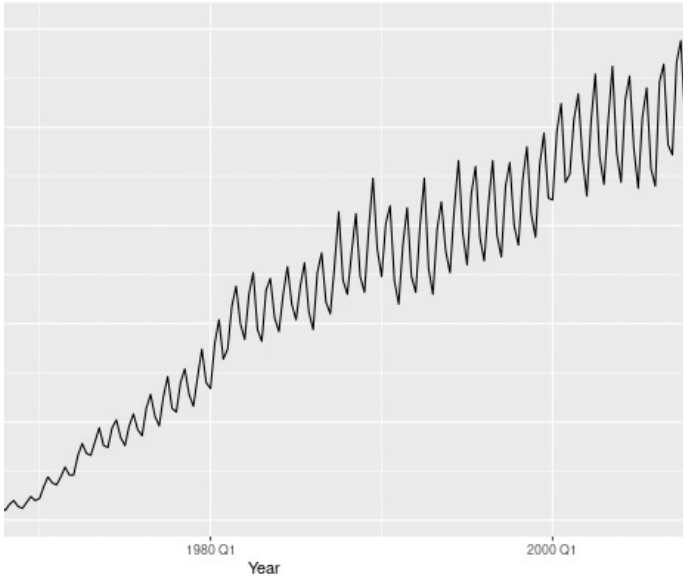
Time series where seasonal amplitude is increasing

Log transformation

For this time series it would be more appropriate to use a multiplicative model: $Y_t = T_t \times S_t \times R_t$

We can take log on both sides. Since $\log(a \cdot b) = \log(a) + \log(b)$ our model becomes:

$$\log(Y_t) = \log(T_t) + \log(S_t) + \log(R_t)$$



Box-Cox transformation

- When series grow over time, its fluctuations often grow as well
- However, we do not want patterns to change just because the scale changes
- We want fluctuations to be comparable across time
- We saw that taking logs can help making proportional changes look constant
- The Box-Cox transformation generalizes this idea
- The Box-Cox uses the data to find the optimal power transformation which makes the scale of the fluctuations more stable

The Box-Cox transformation

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

λ describes how strong the transformation is

A good value of λ is one which makes the size of seasonal variation about the same across the whole series – which makes the forecasting model simpler

Note for later: In R the *guerrero* feature can be used to choose the optimal lambda (λ)

Box-Cox transformation

The idea of the transformation is to make variation more homogenous/even across the data series

https://otexts.com/fpp3/transformations.html

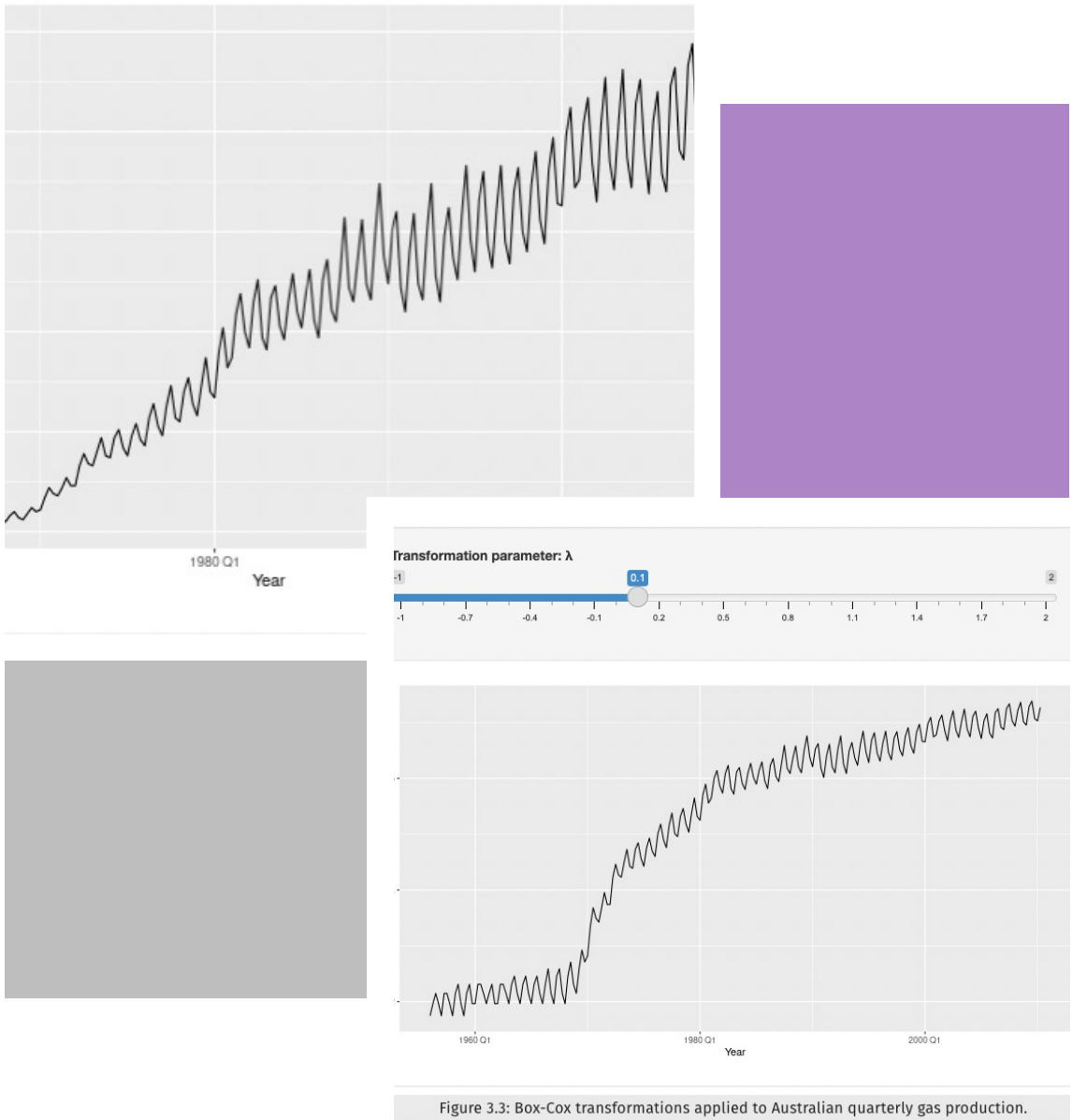


Figure 3.3: Box-Cox transformations applied to Australian quarterly gas production.

Classical Additive Decomposition

Decomposition

- Now we have adjusted the time series (taken out the irrelevant variation) and stabilized the variance (so we can use the additive decomposition)
- Now we can move to the decomposition:
- Step 1: Estimate trend
- Step 2: Remove trend
- Step 3: Estimate seasonality
- Step 4: Compute remainder

Step 1: How do we estimate the trend?

- Answer: By smoothing our time series
- One way to do that is to use moving averages (MA)

Moving averages

- **Moving Averages** is a simple method to smooth out short-term fluctuations and highlight the underlying trend by averaging nearby values.
- Instead of looking at raw data points, a moving average calculates the **average of a fixed number of past observations** at each point in time.
- E.g. MA(4) means that we take the average of our current observation and the three previous ones

Moving averages

Time Series	MA(3)	MA(5)
10	NaN	NaN
12	NaN	NaN
14	12.00	NaN
18	14.67	NaN
20	17.33	14.80
22	20.00	17.20
24	22.00	19.60
26	24.00	22.00
28	26.00	24.00
30	28.00	26.00

Moving Averages

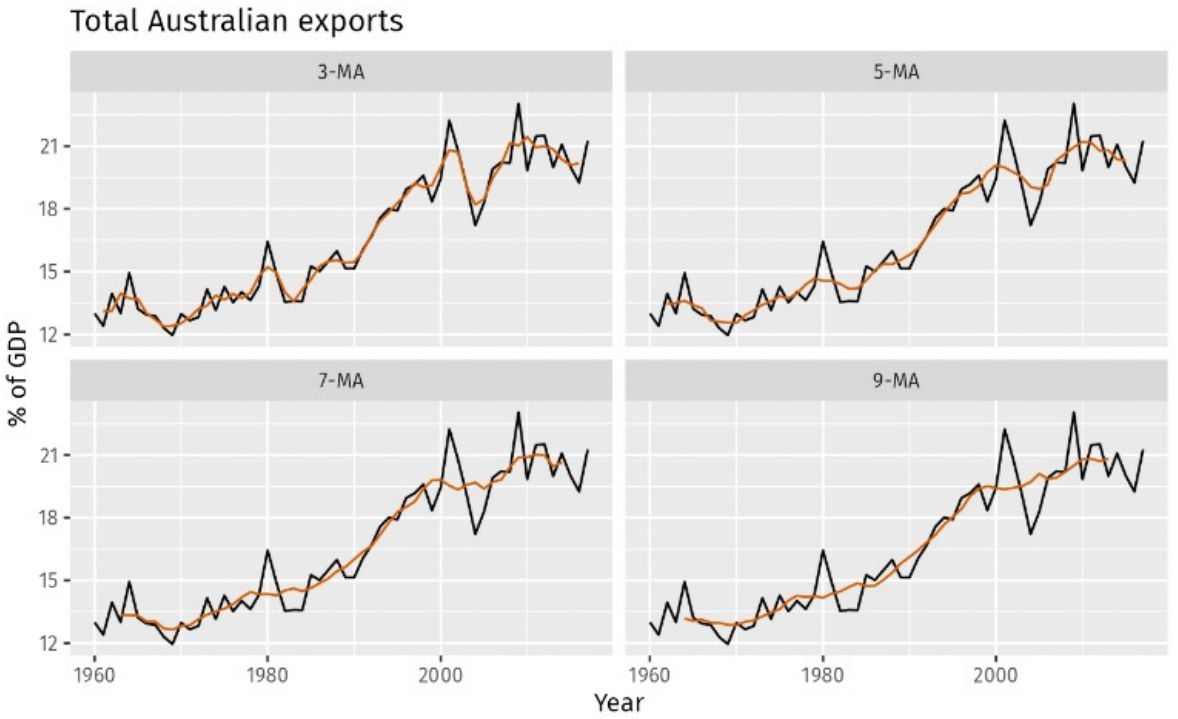


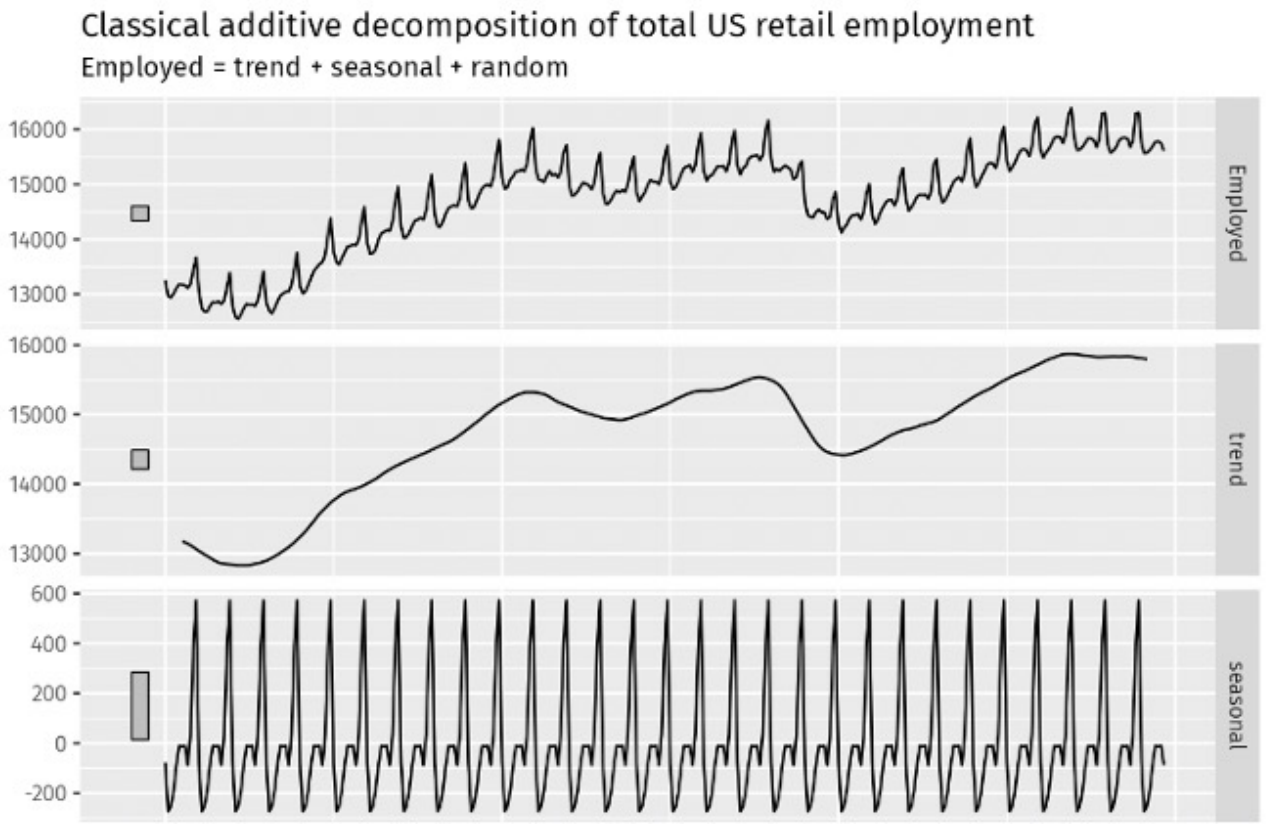
Figure 3.11: Different moving averages applied to the Australian exports data.

Moving averages

- The higher degree of MA, the more smoothing
- But how do we choose the degree of the moving averages?
- Lower degree: captures some of the rapid fluctuations, but may still leave noise in the data
- Higher degree: produces smoother trend, but risk of **oversmoothing** i.e., risk of hiding some important changes in the data. With higher degree of smoothing our time series also becomes significantly shorter
- When we have seasonal data, it is common to use the degree that matches the seasonal cycle, e.g. MA-12 if we have monthly data, MA-4 if we have quarterly data, etc.

Step 2: Removing the trend

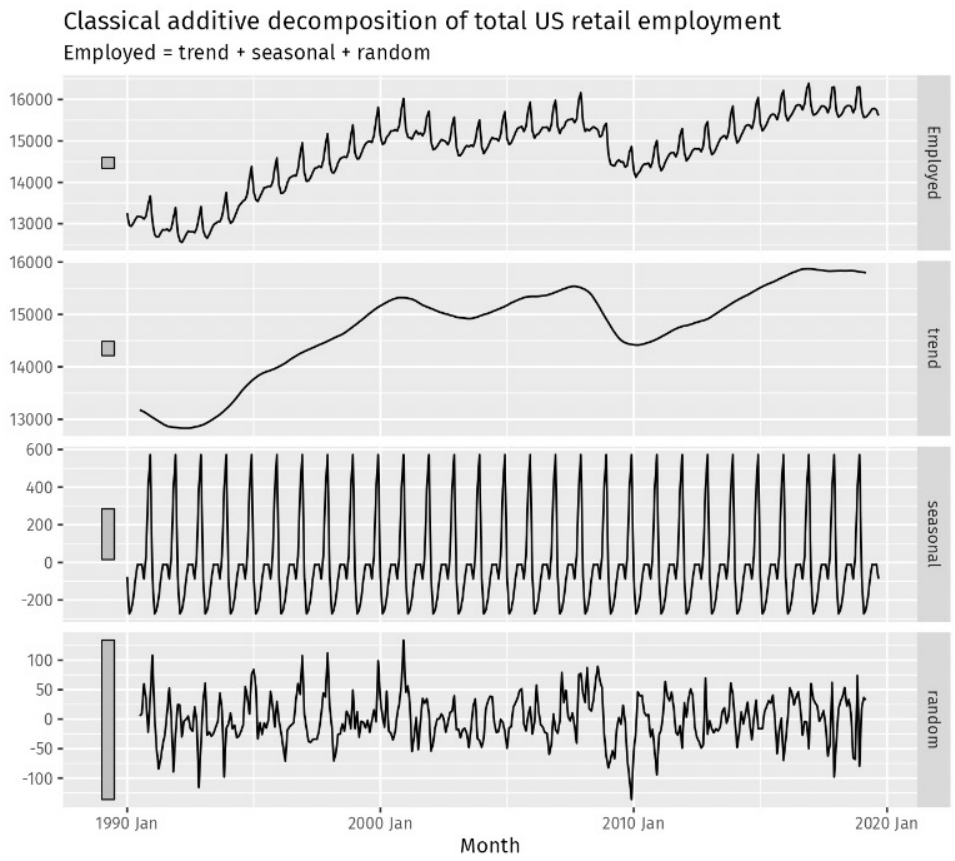
- We used MA to smooth the time series
- This gives us an estimate of T_t
- Removing the trend $Y_t - T_t = S_t + R_t$
- What remains is the seasonality and random fluctuations



Step 3: Estimating the Seasonal Component

- We take the detrended values of the time series and group it by season
- We take the average within each season
- That gives the seasonal effects S_t

Season	Detrended values	Average (Seasonal Component)
Spring	8, 12, 10	**10**
Summer	18, 22, 20	**20**
Fall	-3, -7, -5	**-5**
Winter	-20, -30, -25	**-25**



Step 4: Remove the seasonality to find the remainder

$$Y_t - T_t - S_t = R_t$$

Step 4: The Remainder

- After removing the trend and the seasonality, whatever remains is the irregular component
- Ideally, the remainder is **white noise**
- If the remainder still has structure, then our decomposition has missed something!

STL decomposition

Limitations of the classical decomposition

- Classical decomposition assumes stable patterns:
  - Assume stable seasonality
  - Assumes smooth trend
  - Is very sensitive to outliers
- Sometimes we would like to use more flexible methods

STL decomposition

- STL decomposition (Seasonal and Trend decomposition using Loess)
  - Flexible trend
  - Flexible seasonality
  - Robust to outliers

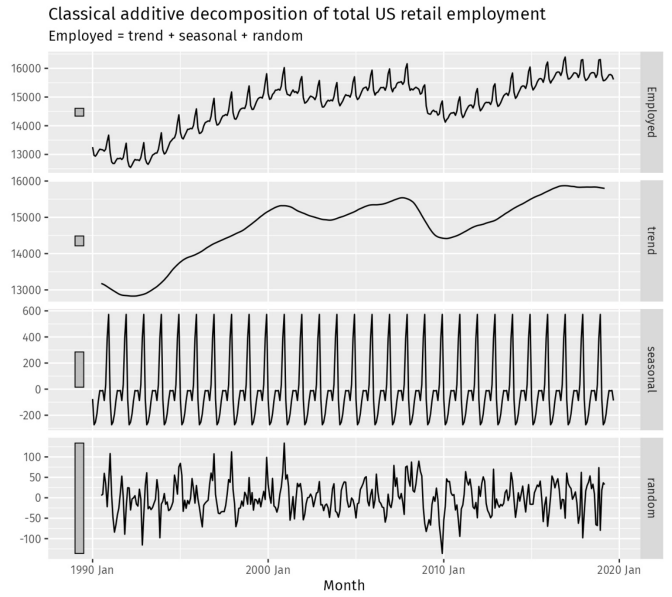


Figure 3.13: A classical additive decomposition of US retail employment.

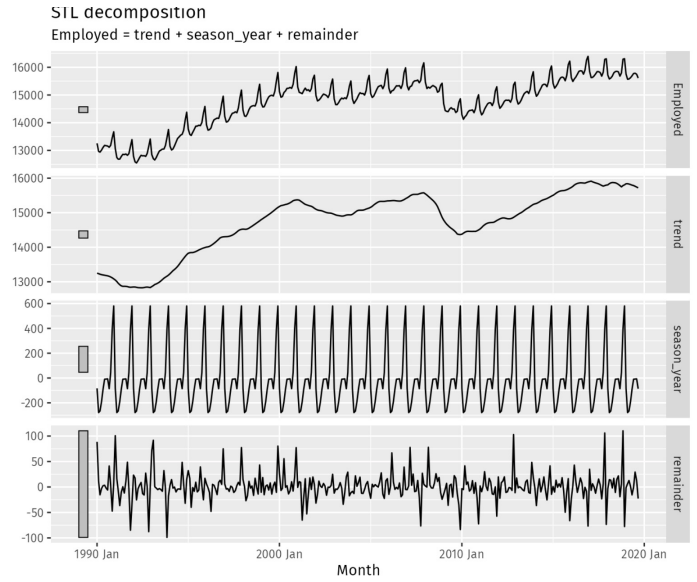


Figure 3.18: Total US retail employment (top) and its three additive components obtained from a robust STL decomposition with flexible trend-cycle and fixed seasonality.

Decomposition (Cleveland, 1985)

- Before we started decomposing, we plotted the data
- Cleveland showed that humans are best at judging position along a common scale (e.g. it is easier for us to read bar chart than pie charts) - > this is why time plots are powerful visualizations
- However, we are worse at judging slopes and angles, which is why smoothing is helpful -> it reveals the structure of the time series
- STL is built on this idea: it is a flexible smoothing method which makes the patterns in our data visible

Today: From Raw Data to Structure

- Today we learned how to:
- Adjust external distortions (inflation, population, etc.)
- Transform the data to stabilize fluctuations
- Decompose the time series to trend, seasonality and remainder
- Classical decomposition and STL decomposition
- Now we understand more about the structure of time series data
- **Next: we start discussing how we model it**