AARHUS UNIVERSITY

Data Science 10

Chris Mathys

Master's Degree Programme in Cognitive Science

Spring 2023

Outline

1. What can we forecast?
2. Time series data and random futures
3. Some case studies
4. Assignment 1

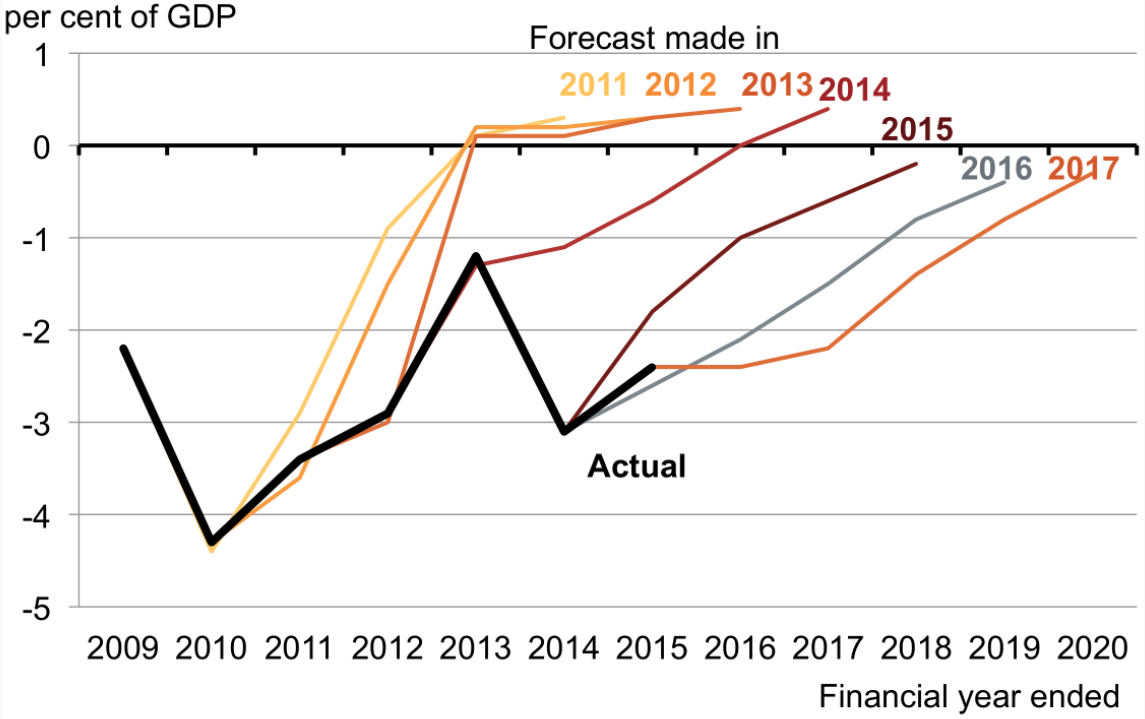
Outline

1. What can we forecast?
2. Time series data and random futures
3. Some case studies
4. Assignment 1

Forecasts that aren't forecasts

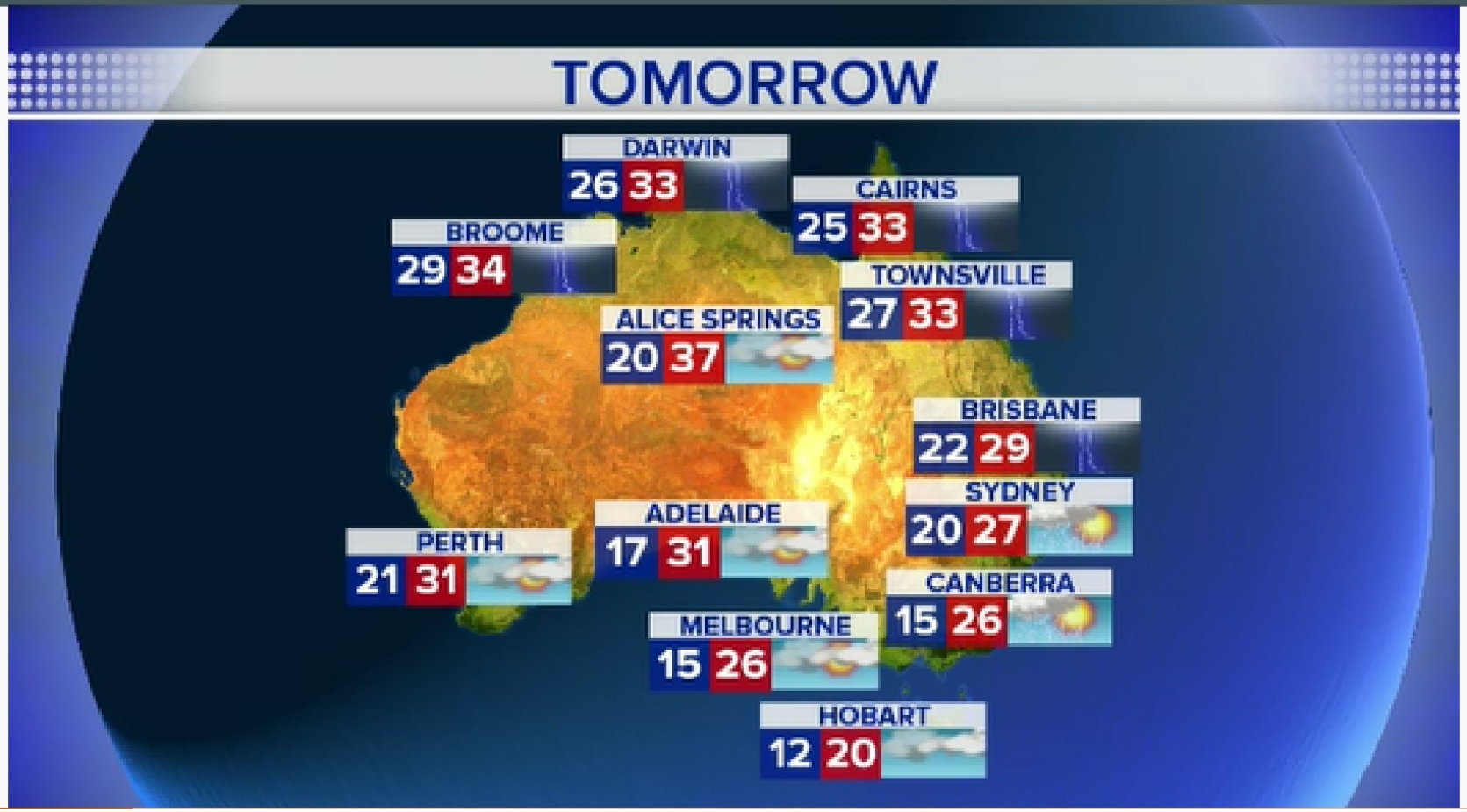
Commonwealth plans to drift back to surplus show the triumph of experience over hope

GRATTAN Institute



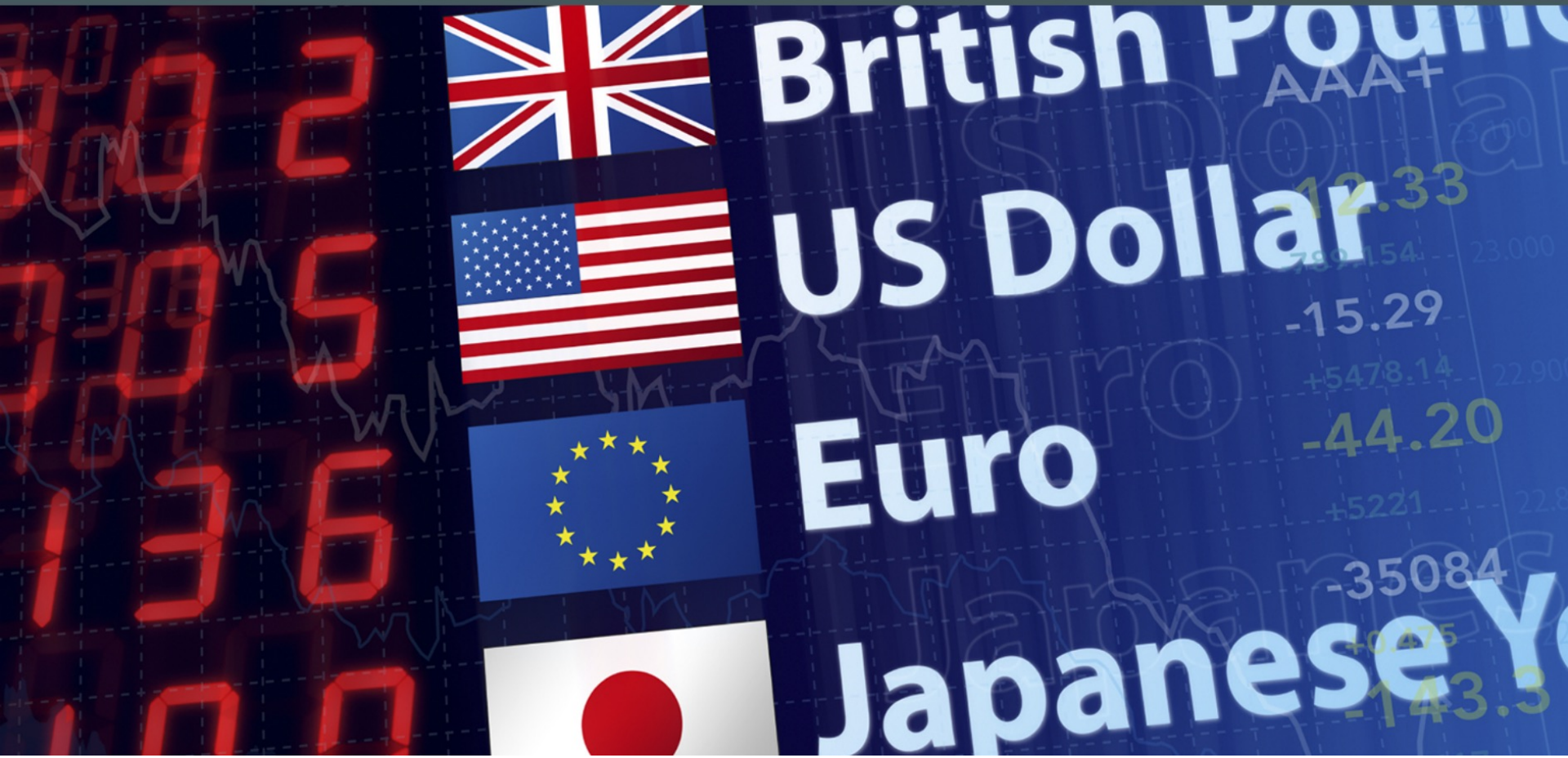
Grattan analysis of Commonwealth Budget Papers

What can we forecast?

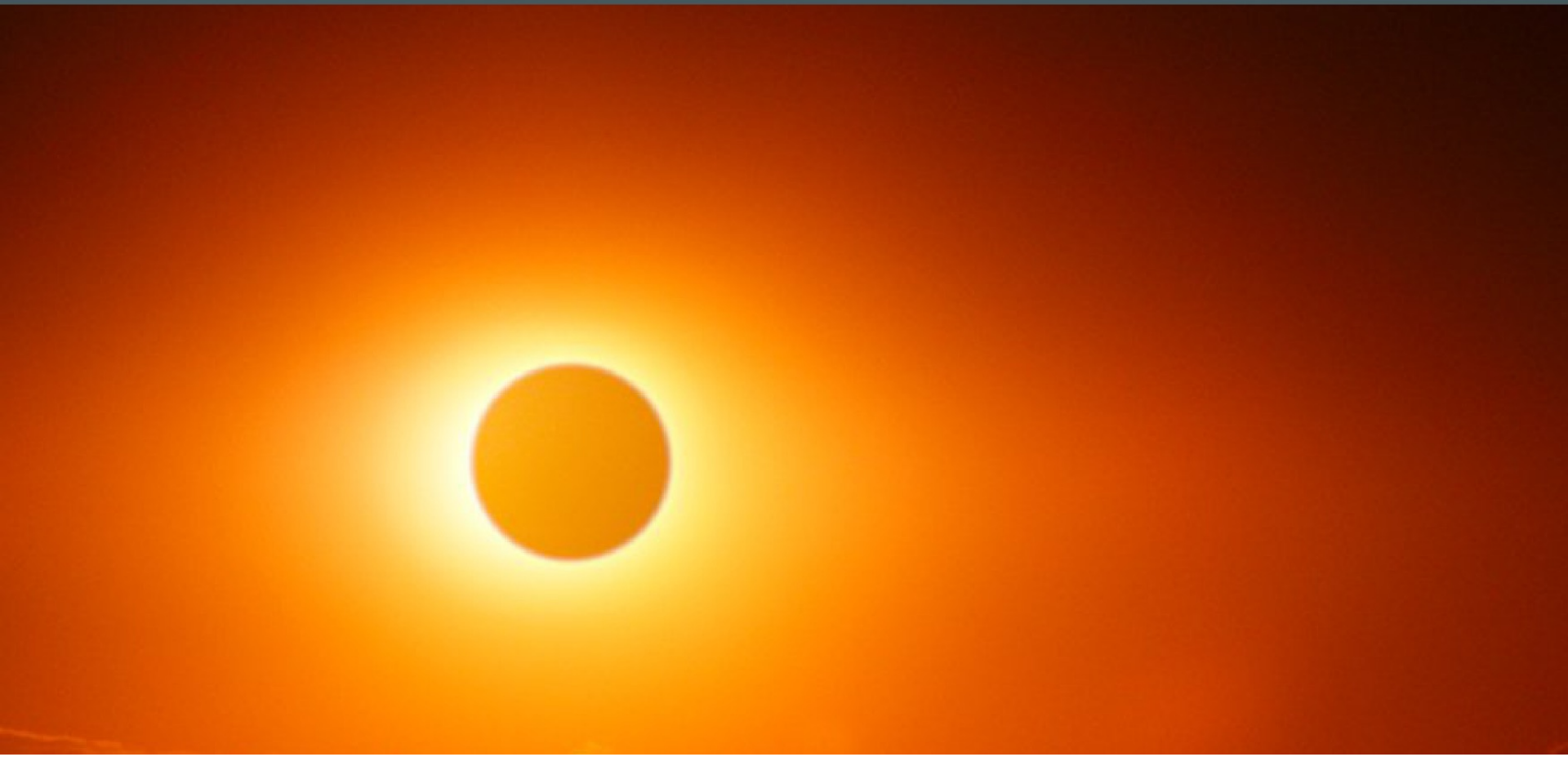


What can we forecast?

What can we forecast?



What can we forecast?

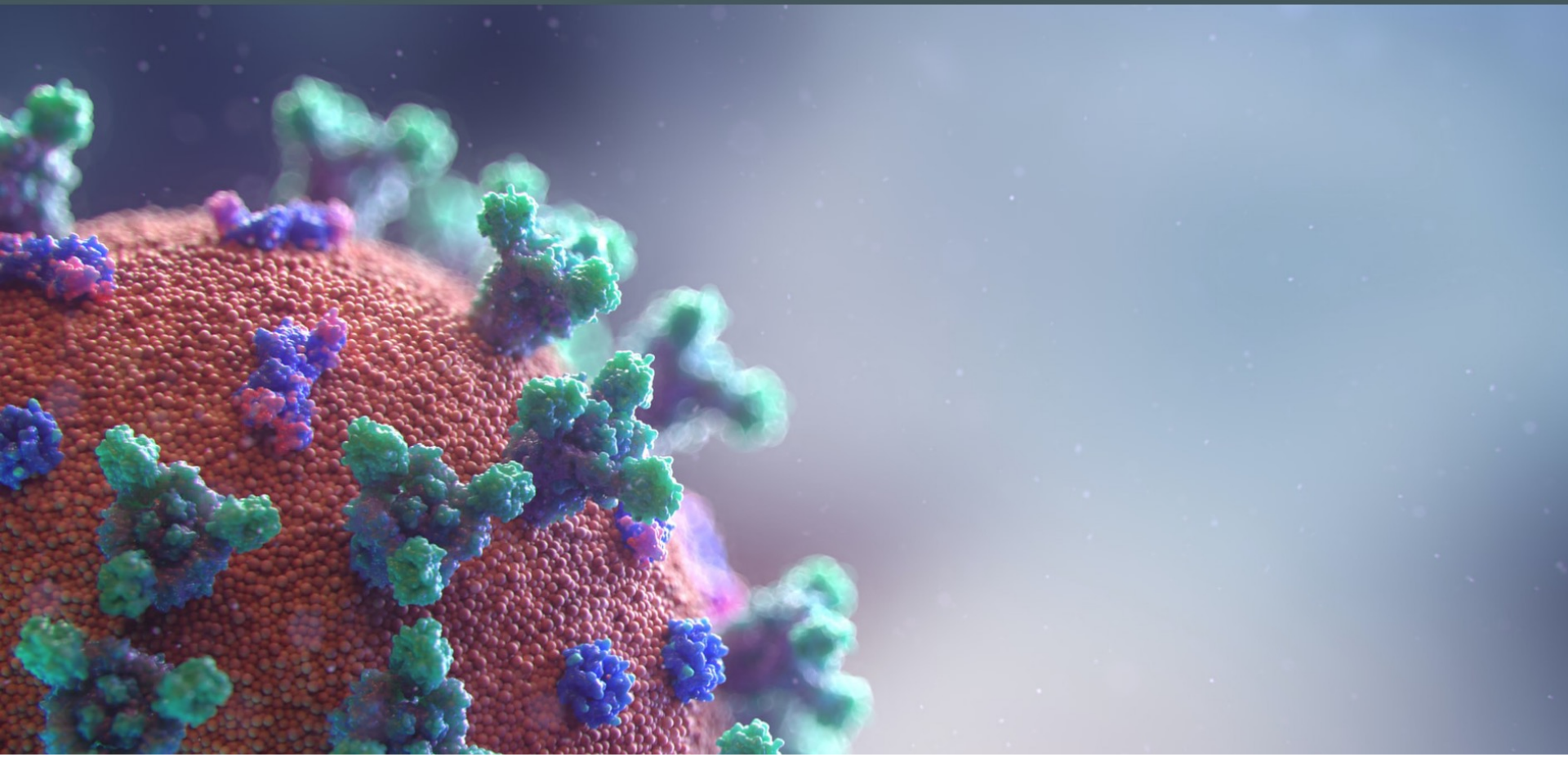


What can we forecast?

What can we forecast?

What can we forecast?

What can we forecast?



Which is easiest to forecast?

- daily electricity demand in 3 days time
- time of sunrise this day next year
- Google stock price tomorrow
- Google stock price in 6 months time
- maximum temperature tomorrow
- exchange rate of $US/AUS next week
- total sales of drugs in Australian pharmacies next month
- timing of next Halley's comet appearance

Which is easiest to forecast?

1. time of sunrise this day next year
2. timing of next Halley's comet appearance
3. maximum temperature tomorrow
4. daily electricity demand in 3 days time
5. total sales of drugs in Australian pharmacies next month
6. Google stock price tomorrow
7. exchange rate of $US/AUS next week
8. Google stock price in 6 months time

Which is easiest to forecast?

1. time of sunrise this day next year
2. timing of next Halley's comet appearance
3. maximum temperature tomorrow
4. daily electricity demand in 3 days time
5. total sales of drugs in Australian pharmacies next month
6. Google stock price tomorrow
7. exchange rate of $US/AUS next week
8. Google stock price in 6 months time

- how do we measure "easiest"?
- what makes something easy/difficult to forecast?

Forecastability factors

Something is easier to forecast if:

1. we have a good understanding of the factors that contribute to it
2. there is lots of data available;
3. the future is somewhat similar to the past
4. the forecasts cannot affect the thing we are trying to forecast.

Outline

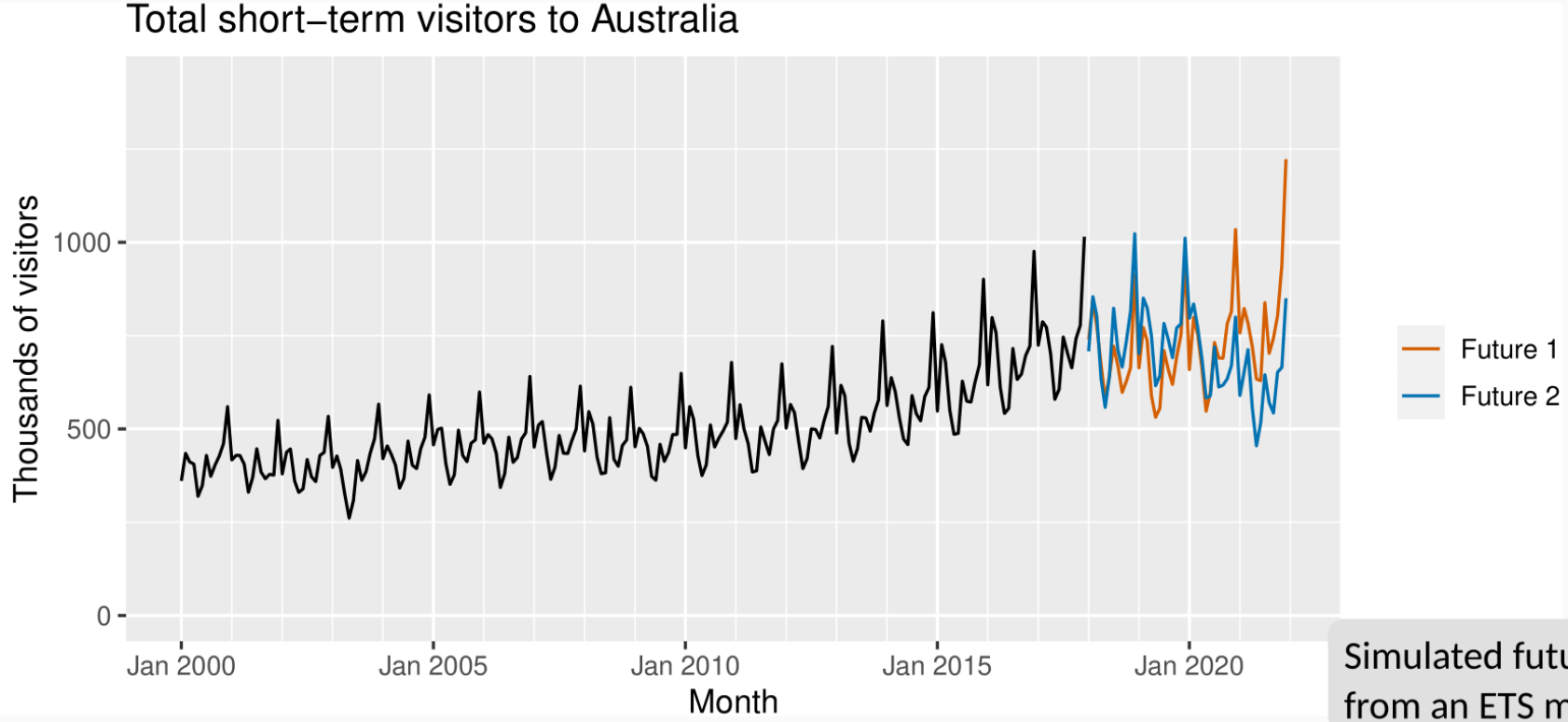
1. What can we forecast?
2. Time series data and random futures
3. Some case studies
4. Assignment 1

Time series data

- Four-yearly Olympic winning times
- Annual Google profits
- Quarterly Australian beer production
- Monthly rainfall
- Weekly retail sales
- Daily IBM stock prices
- Hourly electricity demand
- 5-minute freeway traffic counts
- Time-stamped stock transaction data

Random futures

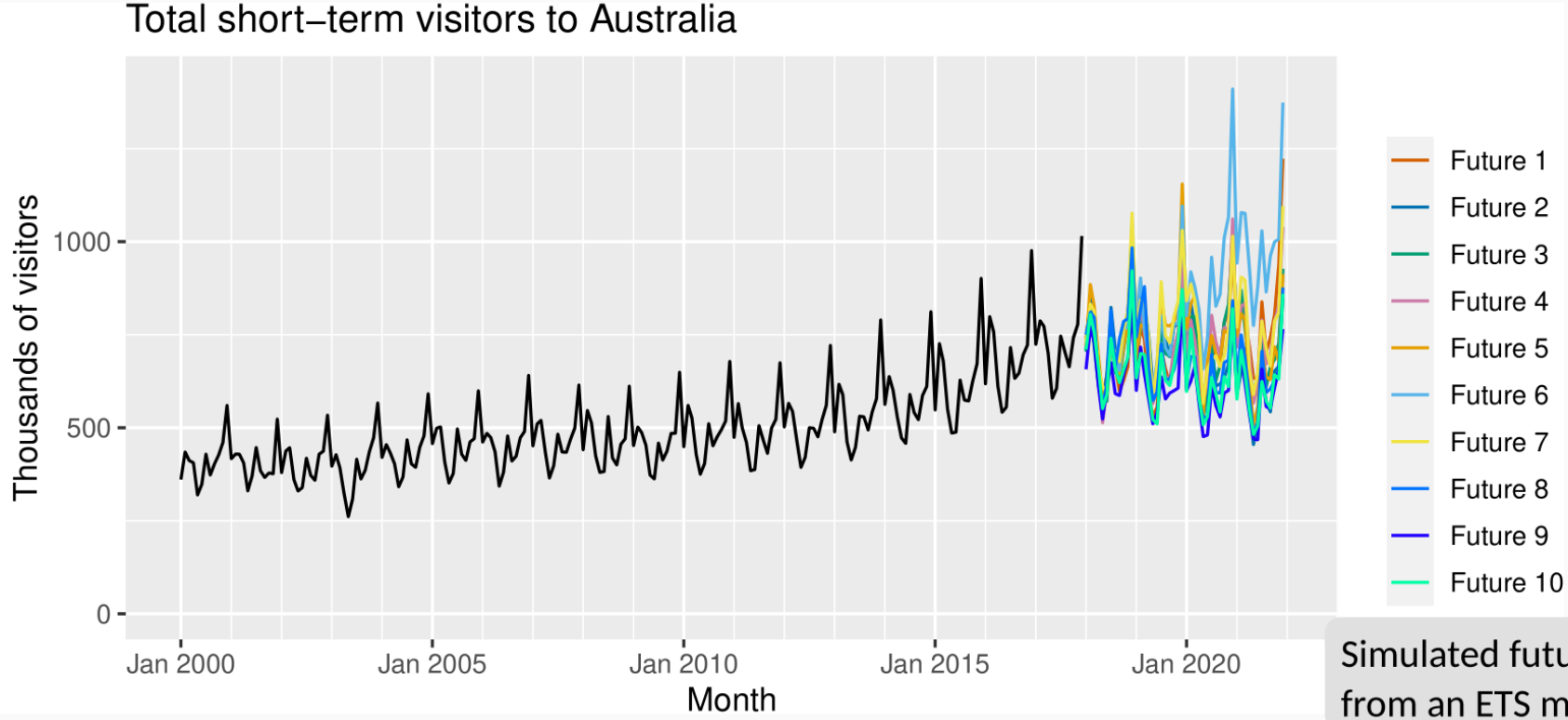
A forecast is an estimate of the probabilities of possible futures.



Simulated futures from an ETS model

Random futures

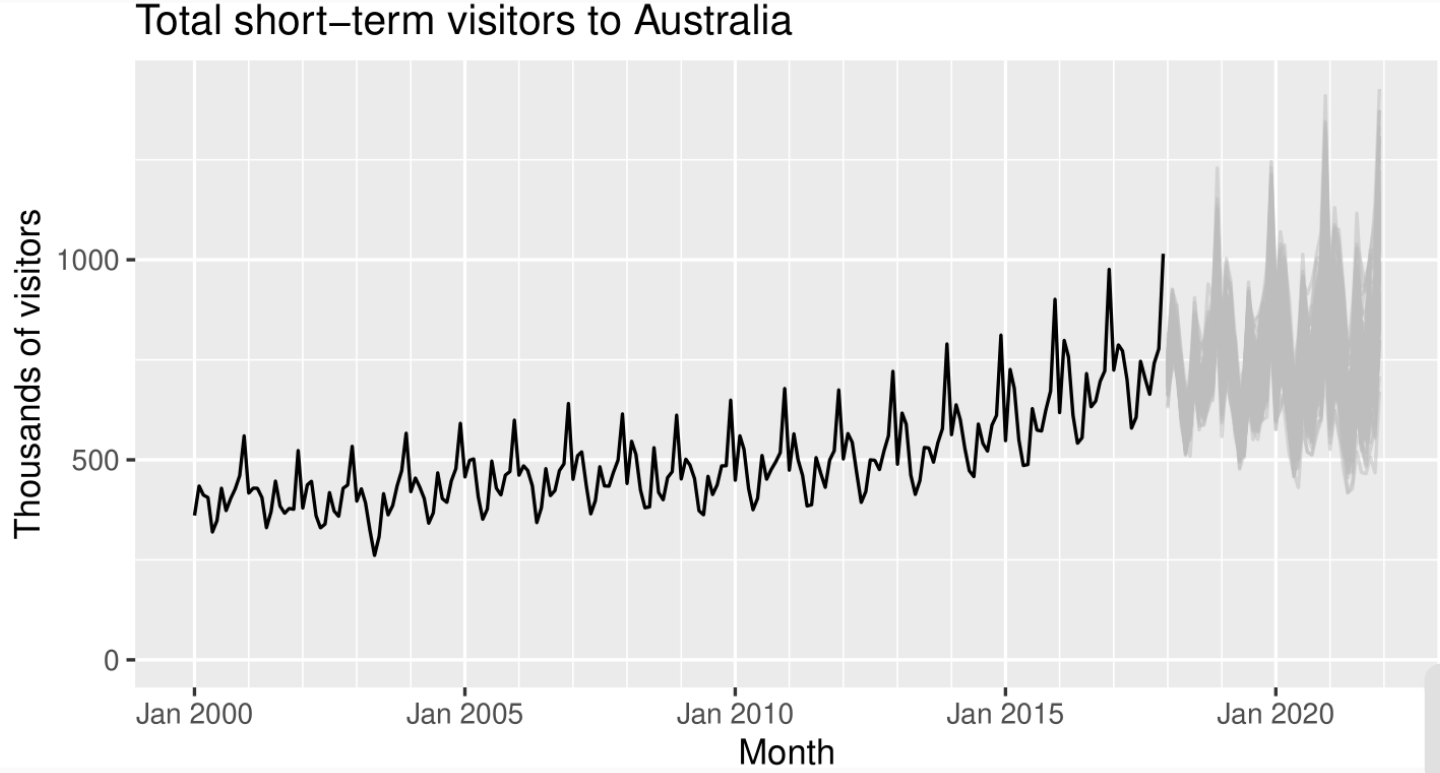
A forecast is an estimate of the probabilities of possible futures.



Simulated futures from an ETS model

Random futures

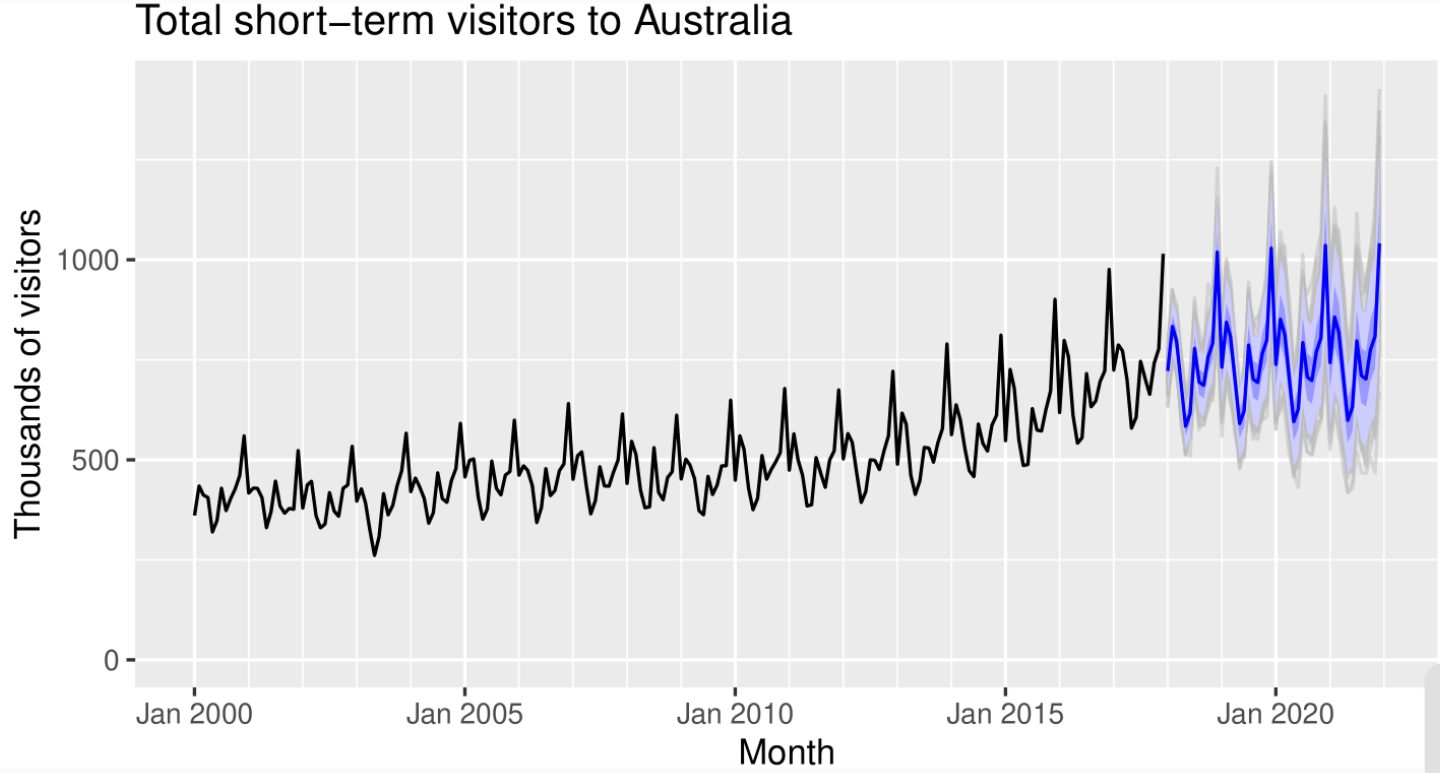
A forecast is an estimate of the probabilities of possible futures.



Simulated futures from an ETS model

Random futures

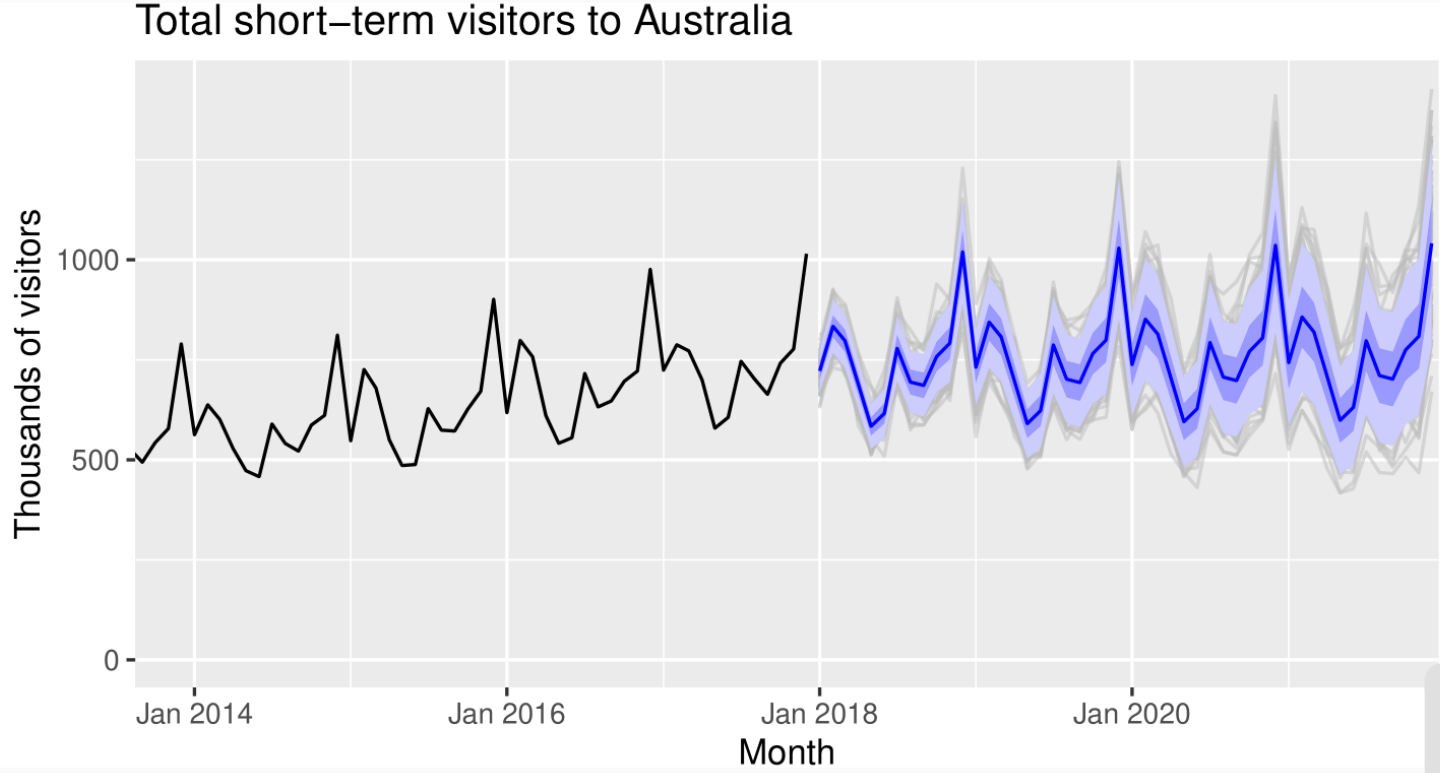
A forecast is an estimate of the probabilities of possible futures.



Simulated futures from an ETS model

Random futures

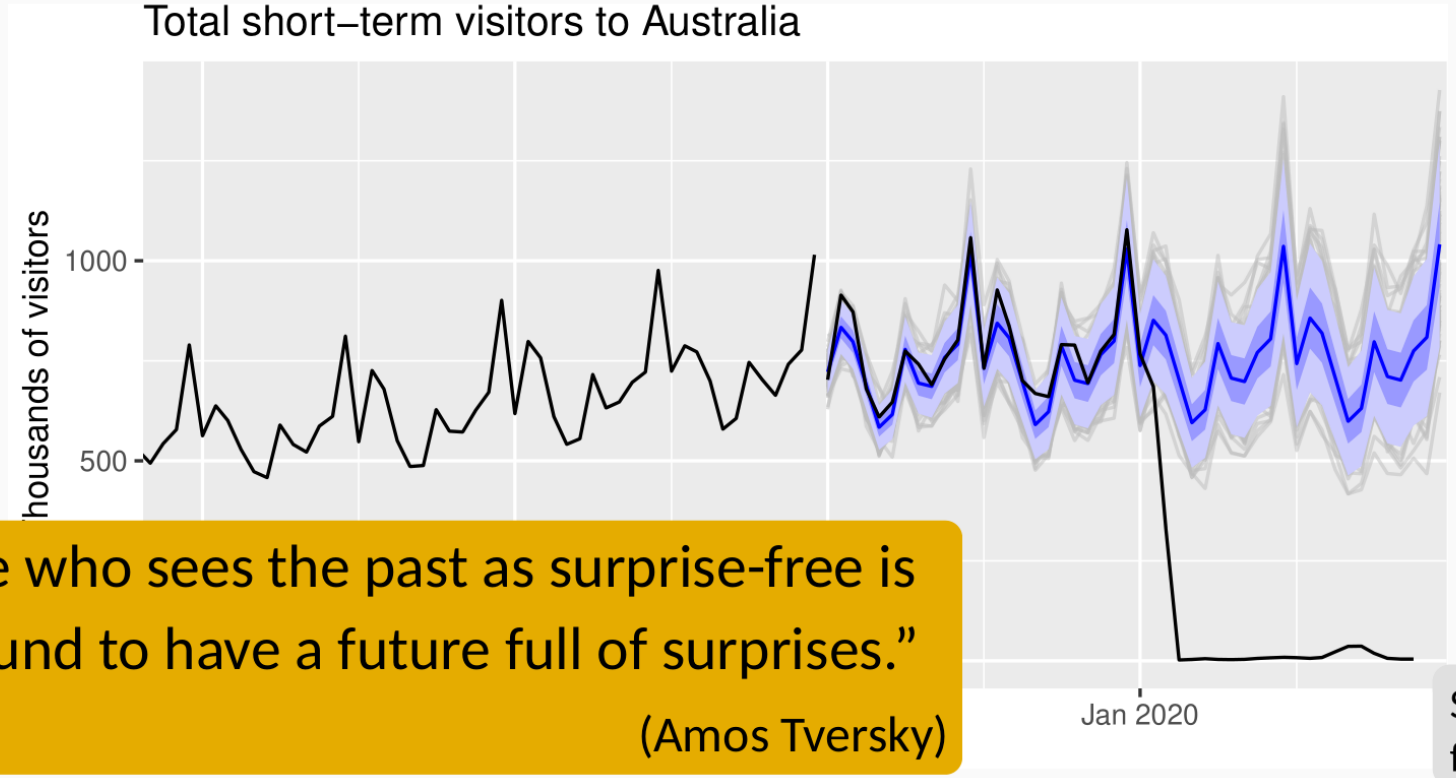
A forecast is an estimate of the probabilities of possible futures.



Simulated futures from an ETS model

Random futures

A forecast is an estimate of the probabilities of possible futures.



"He who sees the past as surprise-free is bound to have a future full of surprises."

(Amos Tversky)

Simulated futures from an ETS model

Statistical forecasting

- Thing to be forecast: a random variable, y_t.
- Forecast distribution: If $\mathcal{I}$ is all observations, then $y_t|\mathcal{I}$ means "the random variable y_t given what we know in $\mathcal{I}$.
- The "point forecast" is the mean (or median) of $y_t|\mathcal{I}$
- The "forecast variance" is $\text{var}[y_t|\mathcal{I}]$
- A prediction interval or "interval forecast" is a range of values of y_t with high probability.
- With time series, $y_{t|t-1} = y_t|\{y_1, y_2, \ldots, y_{t-1}\}$.
- $\hat{y}_{T+h|T} = \text{E}[y_{T+h}|y_1, \ldots, y_T]$ (an h-step forecast taking account of all observations up to time T).

Outline

1. What can we forecast?
2. Time series data and random futures
3. Some case studies
4. Assignment 1

CASE STUDY 1: Paperware company

Problem: Want forecasts of each of hundreds of items. Series can be stationary, trended or seasonal. They currently have a large forecasting program written in-house but it doesn't seem to produce sensible forecasts. They want me to fix it.

Additional information

- Program written in COBOL making numerical calculations limited. It is not possible to do any optimisation.
- Their programmer has little experience in numerical computing.
- They employ no statisticians and want the program to produce forecasts automatically.

CASE STUDY 1: Paperware company

Methods currently used

A 12 month average

C 6 month average

E straight line regression over last 12 months

G straight line regression over last 6 months

H average slope between last year's and this year's values. (Equivalent to differencing at lag 12 and taking mean.)

I Same as H except over 6 months.

K I couldn't understand the explanation.

CASE STUDY 2: PBS

CASE STUDY 2: PBS

The Pharmaceutical Benefits Scheme (PBS) is the Australian government drugs subsidy scheme.

- Many drugs bought from pharmacies are subsidised to allow more equitable access to modern drugs.
- The cost to government is determined by the number and types of drugs purchased. Currently nearly 1% of GDP.
- The total cost is budgeted based on forecasts of drug usage.

CASE STUDY 2: PBS

CASE STUDY 2: PBS

- In 2001: $4.5 billion budget, under-forecasted by $800 million.
- Thousands of products. Seasonal demand.
- Subject to covert marketing, volatile products, uncontrollable expenditure.
- Although monthly data available for 10 years, data are aggregated to annual values, and only the first three years are used in estimating the forecasts.
- All forecasts being done with the `FORECAST` function in MS-Excel!

CASE STUDY 3: Car fleet company

Client: One of Australia's largest car fleet companies

Problem: how to forecast resale value of vehicles? How should this affect leasing and sales policies?

CASE STUDY 3: Car fleet company

Client: One of Australia's largest car fleet companies

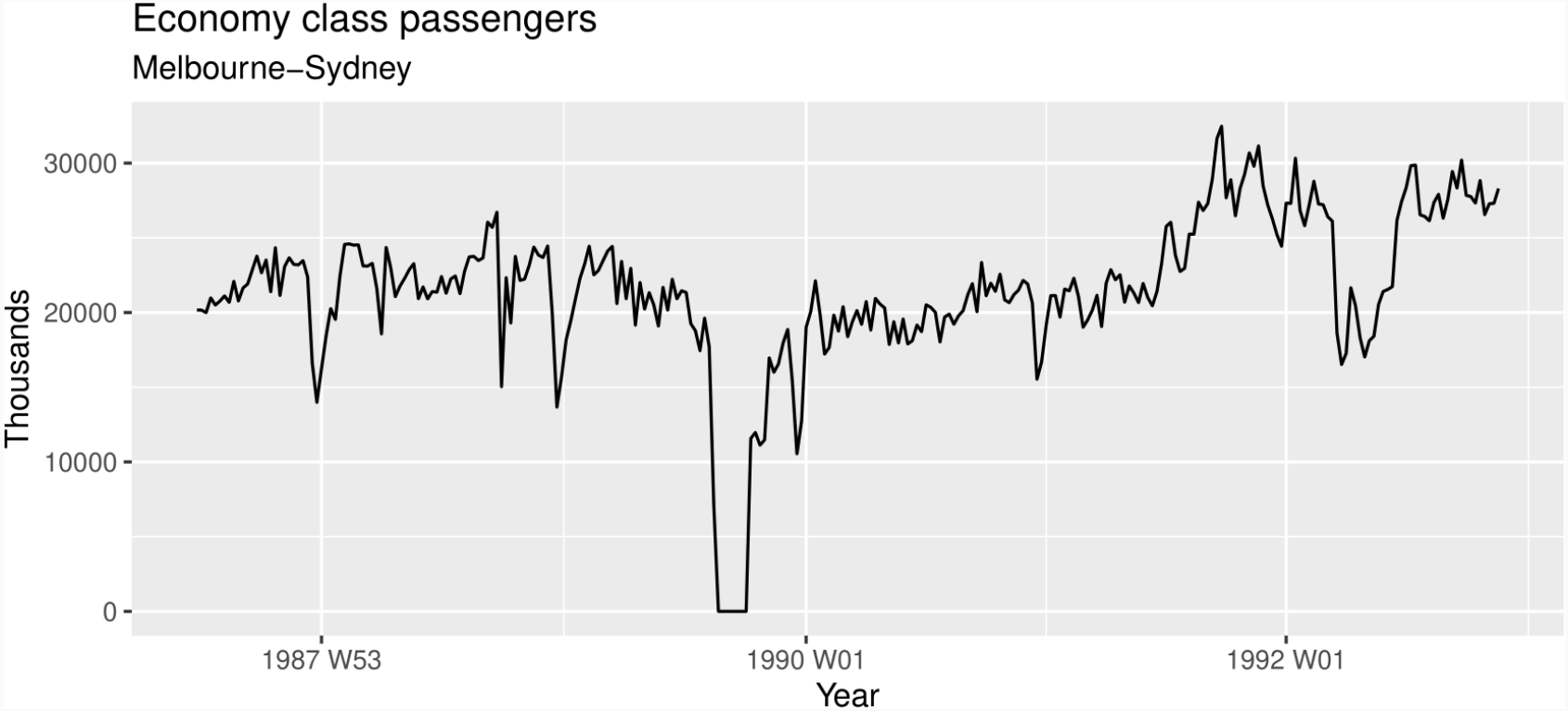
Problem: how to forecast resale value of vehicles? How should this affect leasing and sales policies?

Additional information

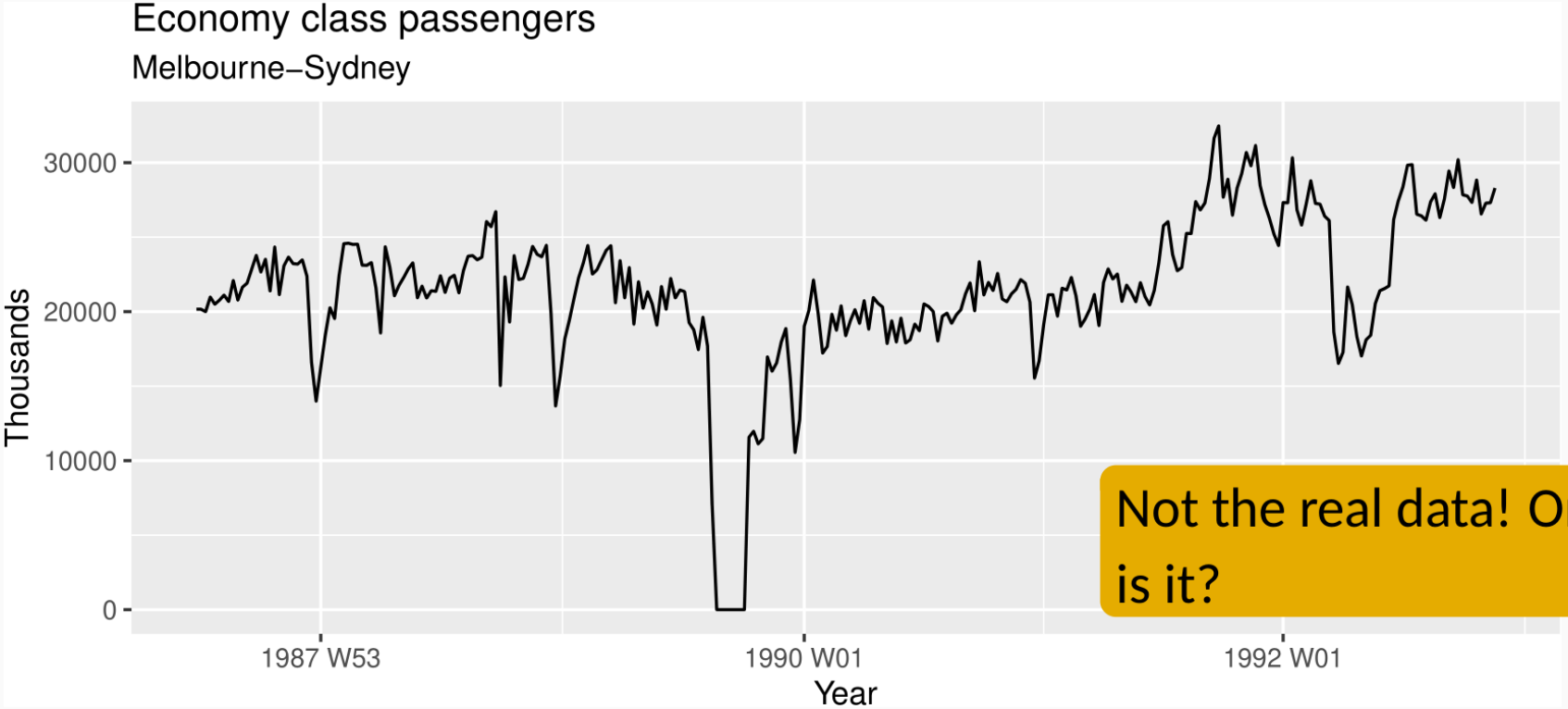
- They can provide a large amount of data on previous vehicles and their eventual resale values.
- The resale values are currently estimated by a group of specialists. They see me as a threat and do not cooperate.

CASE STUDY 4: Airline

CASE STUDY 4: Airline



CASE STUDY 4: Airline



CASE STUDY 4: Airline

Problem: how to forecast passenger traffic on major routes?

Additional information

- They can provide a large amount of data on previous routes.
- Traffic is affected by school holidays, special events such as the Grand Prix, advertising campaigns, competition behaviour, etc.
- They have a highly capable team of people who are able to do most of the computing.

Outline

1. Time series in R
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Time plots
5. Seasonal and subseries plots
6. Lag plots and autocorrelation
7. White noise

Outline

1. Time series in R
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Time plots
5. Seasonal and subseries plots
6. Lag plots and autocorrelation
7. White noise

`tsibble` objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
## # Key:       Country [263]
##     Year Country              GDP Imports Exports Population
##    Index Key                  Measured variables
##  1  1960 Afghanistan  537777811.    7.02    4.13    8996351
##  2  1961 Afghanistan  548888896.    8.10    4.45    9166764
##  3  1962 Afghanistan  546666678.    9.35    4.88    9345868
##  4  1963 Afghanistan  751111191.   16.9     9.17    9533954
##  5  1964 Afghanistan  800000044.   18.1     8.89    9731361
##  6  1965 Afghanistan 1006666638.   21.4    11.3     9938414
##  7  1966 Afghanistan 1399999967.   18.6     8.57   10152331
##  8  1967 Afghanistan 1673333418.   14.2     6.77   10372630
##  9  1968 Afghanistan 1373333367.   15.2     8.90   10604346
## 10  1969 Afghanistan 1408888922.   15.0    10.1    10854428
```

`tsibble` objects

```
tourism
```

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:       Region, State, Purpose [304]
##    Quarter Region   State Purpose  Trips
##      Index    Keys                Measure
##  1 1998 Q1 Adelaide SA    Business  135.
##  2 1998 Q2 Adelaide SA    Business  110.
##  3 1998 Q3 Adelaide SA    Business  166.
##  4 1998 Q4 Adelaide SA    Business  127.
##  5 1999 Q1 Adelaide SA    Business  137.
##  6 1999 Q2 Adelaide SA    Business  200.
##  7 1999 Q3 Adelaide SA    Business  169.
##  8 1999 Q4 Adelaide SA    Business  134.
##  9 2000 Q1 Adelaide SA    Business  154.
## 10 2000 Q2 Adelaide SA    Business  169.
```

Domestic visitor nights in thousands by state/region and purpose.

`tsibble` objects

- A `tsibble` allows storage and manipulation of multiple time series in R.
- It contains:
  - An index: time information about the observation
  - Measured variable(s): numbers of interest
  - Key variable(s): optional unique identifiers for each series
- It works with tidyverse functions.

The `tsibble` index

Example

```
mydata <- tsibble(
    year = 2012:2016,
    y = c(123, 39, 78, 52, 110),
    index = year
)
mydata
```

```
## # A tsibble: 5 x 2 [1Y]
##    year     y
##   <int> <dbl>
## 1  2012   123
## 2  2013    39
## 3  2014    78
## 4  2015    52
## 5  2016   110
```

The `tsibble` index

Example

```
mydata <- tibble(
    year = 2012:2016,
    y = c(123, 39, 78, 52, 110)
  ) %>%
  as_tsibble(index = year)
mydata
```

```
## # A tsibble: 5 x 2 [1Y]
##    year     y
##   <int> <dbl>
## 1  2012   123
## 2  2013    39
## 3  2014    78
## 4  2015    52
## 5  2016   110
```

The `tsibble` index

For observations more frequent than once per year, we need to use a time class function on the index.

```
z
```

```
## # A tibble: 5 x 2
##   Month    Observation
##   <chr>          <dbl>
## 1 2019 Jan          50
## 2 2019 Feb          23
## 3 2019 Mar          34
## 4 2019 Apr          30
## 5 2019 May          25
```

The `tsibble` index

For observations more frequent than once per year, we need to use a time class function on the index.

```
z %>%
  mutate(Month = yearmonth(Month)) %>%
  as_tsibble(index = Month)
```

```
## # A tsibble: 5 x 2 [1M]
##      Month Observation
##      <mth>       <dbl>
## 1 2019 Jan          50
## 2 2019 Feb          23
## 3 2019 Mar          34
## 4 2019 Apr          30
## 5 2019 May          25
```

The `tsibble` index

Common time index variables can be created with these functions:

Frequency	Function
Annual	`start:end`
Quarterly	`yearquarter()`
Monthly	`yearmonth()`
Weekly	`yearweek()`
Daily	`as_date(), ymd()`
Sub-daily	`as_datetime()`

Outline

1. Time series in R
2. Example: Australian prison population
3. Example: Australian pharmaceutical sales
4. Time plots
5. Seasonal and subseries plots
6. Lag plots and autocorrelation
7. White noise

Australian prison population

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv")
```

```
## # A tibble: 3,072 x 6
##    date       state gender legal     indigenous count
##    <date>     <chr> <chr>  <chr>     <chr>      <dbl>
##  1 2005-03-01 ACT   Female Remanded  ATSI           0
##  2 2005-03-01 ACT   Female Remanded  Other          2
##  3 2005-03-01 ACT   Female Sentenced ATSI           0
##  4 2005-03-01 ACT   Female Sentenced Other          0
##  5 2005-03-01 ACT   Male   Remanded  ATSI           7
##  6 2005-03-01 ACT   Male   Remanded  Other         58
##  7 2005-03-01 ACT   Male   Sentenced ATSI           0
##  8 2005-03-01 ACT   Male   Sentenced Other          0
##  9 2005-03-01 NSW   Female Remanded  ATSI          51
## 10 2005-03-01 NSW   Female Remanded  Other        131
## # ... with 3,062 more rows
```

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%
  mutate(Quarter = yearquarter(date))
```

```
## # A tibble: 3,072 x 7
##    date       state gender legal     indigenous count Quarter
##    <date>     <chr> <chr>  <chr>     <chr>       <dbl>   <qtr>
##  1 2005-03-01 ACT   Female Remanded  ATSI            0 2005 Q1
##  2 2005-03-01 ACT   Female Remanded  Other           2 2005 Q1
##  3 2005-03-01 ACT   Female Sentenced ATSI            0 2005 Q1
##  4 2005-03-01 ACT   Female Sentenced Other           0 2005 Q1
##  5 2005-03-01 ACT   Male   Remanded  ATSI            7 2005 Q1
##  6 2005-03-01 ACT   Male   Remanded  Other          58 2005 Q1
##  7 2005-03-01 ACT   Male   Sentenced ATSI            0 2005 Q1
##  8 2005-03-01 ACT   Male   Sentenced Other           0 2005 Q1
##  9 2005-03-01 NSW   Female Remanded  ATSI           51 2005 Q1
## 10 2005-03-01 NSW   Female Remanded  Other         131 2005 Q1
```

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%
  mutate(Quarter = yearquarter(date)) %>%
  select(-date)
```

```
## # A tibble: 3,072 x 6
##    state gender legal     indigenous count Quarter
##    <chr> <chr>  <chr>     <chr>      <dbl>   <qtr>
##  1 ACT   Female Remanded  ATSI           0 2005 Q1
##  2 ACT   Female Remanded  Other          2 2005 Q1
##  3 ACT   Female Sentenced ATSI           0 2005 Q1
##  4 ACT   Female Sentenced Other          0 2005 Q1
##  5 ACT   Male   Remanded  ATSI           7 2005 Q1
##  6 ACT   Male   Remanded  Other         58 2005 Q1
##  7 ACT   Male   Sentenced ATSI           0 2005 Q1
##  8 ACT   Male   Sentenced Other          0 2005 Q1
##  9 NSW   Female Remanded  ATSI          51 2005 Q1
## 10 NSW   Female Remanded  Other        131 2005 Q1
```

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%
  mutate(Quarter = yearquarter(date)) %>%
  select(-date) %>%
  as_tsibble(
    index = Quarter,
    key = c(state, gender, legal, indigenous)
  )
```

```
## # A tsibble: 3,072 x 6 [1Q]
## # Key:       state, gender, legal, indigenous [64]
##    state gender legal    indigenous count Quarter
##    <chr> <chr>  <chr>    <chr>      <dbl>   <qtr>
##  1 ACT   Female Remanded ATSI           0 2005 Q1
##  2 ACT   Female Remanded ATSI           1 2005 Q2
##  3 ACT   Female Remanded ATSI           0 2005 Q3
##  4 ACT   Female Remanded ATSI           0 2005 Q4
##  5 ACT   Female Remanded ATSI           1 2006 Q1
```

Outline

Australian Pharmaceutical Benefits Scheme

Australian Pharmaceutical Benefits Scheme

The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy scheme.

Australian Pharmaceutical Benefits Scheme

The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy scheme.

- Many drugs bought from pharmacies are subsidised to allow more equitable access to modern drugs.
- The cost to government is determined by the number and types of drugs purchased. Currently nearly 1% of GDP.
- The total cost is budgeted based on forecasts of drug usage.
- Costs are disaggregated by drug type (ATC1 x15 / ATC2 84), concession category (x2) and patient type (x2), giving $84 \times 2 \times 2 = 336$ time series.

Working with `tsibble` objects

PBS

```
## # A tsibble: 67,596 x 9 [1M]
## # Key:       Concession, Type, ATC1, ATC2 [336]
##       Month Concession   Type    ATC1  ATC1_desc ATC2  ATC2_desc Scripts  Cost
##       <mth> <chr>        <chr>   <chr> <chr>     <chr> <chr>       <dbl> <dbl>
##  1 1991 Jul Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   18228 67877
##  2 1991 Aug Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   15327 57011
##  3 1991 Sep Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   14775 55020
##  4 1991 Oct Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   15380 57222
##  5 1991 Nov Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   14371 52120
##  6 1991 Dec Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   15028 54299
##  7 1992 Jan Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   11040 39753
##  8 1992 Feb Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   15165 54405
##  9 1992 Mar Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   16898 61108
## 10 1992 Apr Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~   18141 65356
## # ... with 67,586 more rows
```

Working with `tsibble` objects

We can use the `filter()` function to select rows.

```
PBS %>%
  filter(ATC2 == "A10")
```

```
## # A tsibble: 816 x 9 [1M]
## # Key:       Concession, Type, ATC1, ATC2 [4]
##       Month Concession   Type   ATC1  ATC1_desc ATC2  ATC2_desc Scripts   Cost
##       <mth> <chr>        <chr>  <chr> <chr>     <chr> <chr>       <dbl>  <dbl>
##  1 1991 Jul Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   89733 2.09e6
##  2 1991 Aug Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   77101 1.80e6
##  3 1991 Sep Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   76255 1.78e6
##  4 1991 Oct Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   78681 1.85e6
##  5 1991 Nov Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   70554 1.69e6
##  6 1991 Dec Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   75814 1.84e6
##  7 1992 Jan Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   64186 1.56e6
##  8 1992 Feb Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   75899 1.73e6
##  9 1992 Mar Concessional Co-pa~ A     Alimenta~ A10   ANTIDIAB~   89445 2.05e6
```

Working with `tsibble` objects

We can use the `select()` function to select columns.

```
PBS %>%
  filter(ATC2 == "A10") %>%
  select(Month, Concession, Type, Cost)
```

```
## # A tsibble: 816 x 4 [1M]
## # Key:       Concession, Type [4]
##       Month Concession   Type           Cost
##       <mth> <chr>        <chr>         <dbl>
##  1 1991 Jul Concessional Co-payments 2092878
##  2 1991 Aug Concessional Co-payments 1795733
##  3 1991 Sep Concessional Co-payments 1777231
##  4 1991 Oct Concessional Co-payments 1848507
##  5 1991 Nov Concessional Co-payments 1686458
##  6 1991 Dec Concessional Co-payments 1843079
##  7 1992 Jan Concessional Co-payments 1564702
##  8 1992 Feb Concessional Co-payments 1732508
```

Working with `tsibble` objects

We can use the `summarise()` function to summarise over keys.

```
PBS %>%
  filter(ATC2 == "A10") %>%
  select(Month, Concession, Type, Cost) %>%
  summarise(total_cost = sum(Cost))
```

```
## # A tsibble: 204 x 2 [1M]
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul    3526591
##  2 1991 Aug    3180891
##  3 1991 Sep    3252221
##  4 1991 Oct    3611003
##  5 1991 Nov    3565869
##  6 1991 Dec    4306371
##  7 1992 Jan    5088335
##  8 1992 Feb    2814520
```

Working with `tsibble` objects

We can use the `mutate()` function to create new variables.

```
PBS %>%
  filter(ATC2 == "A10") %>%
  select(Month, Concession, Type, Cost) %>%
  summarise(total_cost = sum(Cost)) %>%
  mutate(total_cost = total_cost / 1e6)
```

```
## # A tsibble: 204 x 2 [1M]
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul       3.53
##  2 1991 Aug       3.18
##  3 1991 Sep       3.25
##  4 1991 Oct       3.61
##  5 1991 Nov       3.57
##  6 1991 Dec       4.31
##  7 1992 Jan       5.09
```

Working with `tsibble` objects

We can use the `mutate()` function to create new variables.

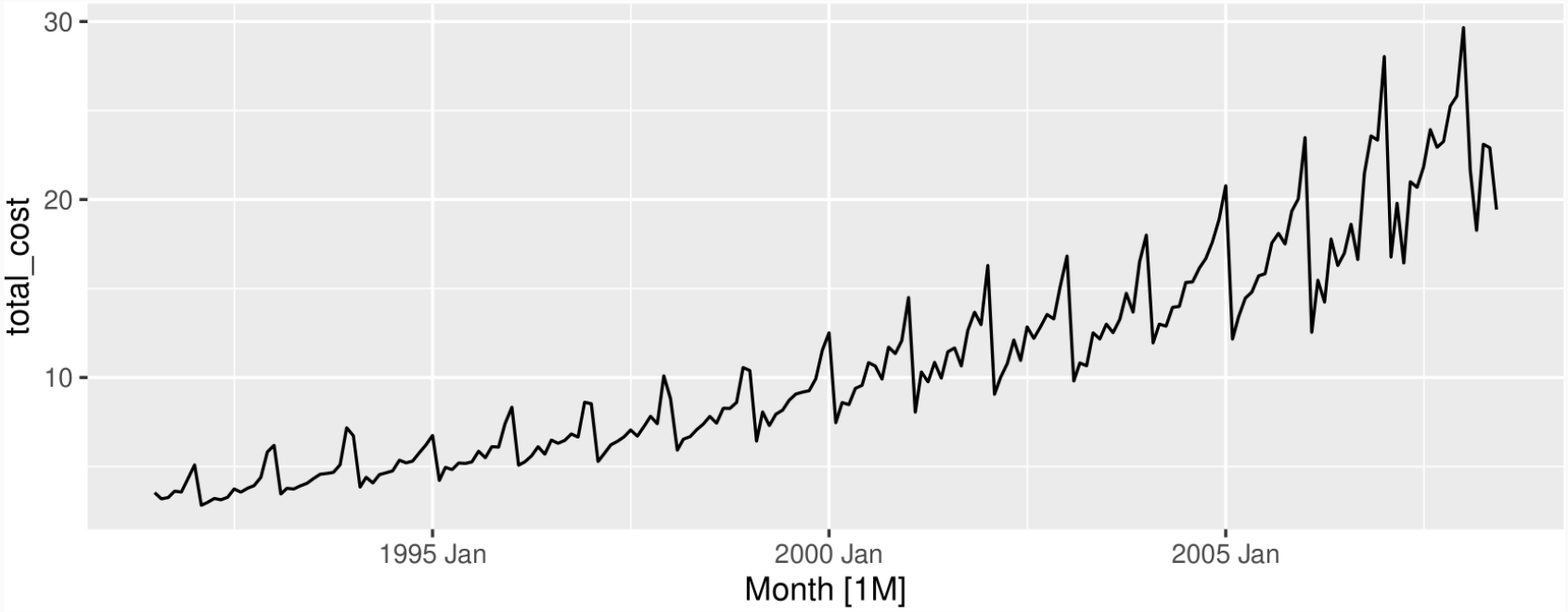
```
PBS %>%
  filter(ATC2 == "A10") %>%
  select(Month, Concession, Type, Cost) %>%
  summarise(total_cost = sum(Cost)) %>%
  mutate(total_cost = total_cost / 1e6) -> a10
```

```
## # A tsibble: 204 x 2 [1M]
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul       3.53
##  2 1991 Aug       3.18
##  3 1991 Sep       3.25
##  4 1991 Oct       3.61
##  5 1991 Nov       3.57
##  6 1991 Dec       4.31
##  7 1992 Jan       5.09
```

Outline

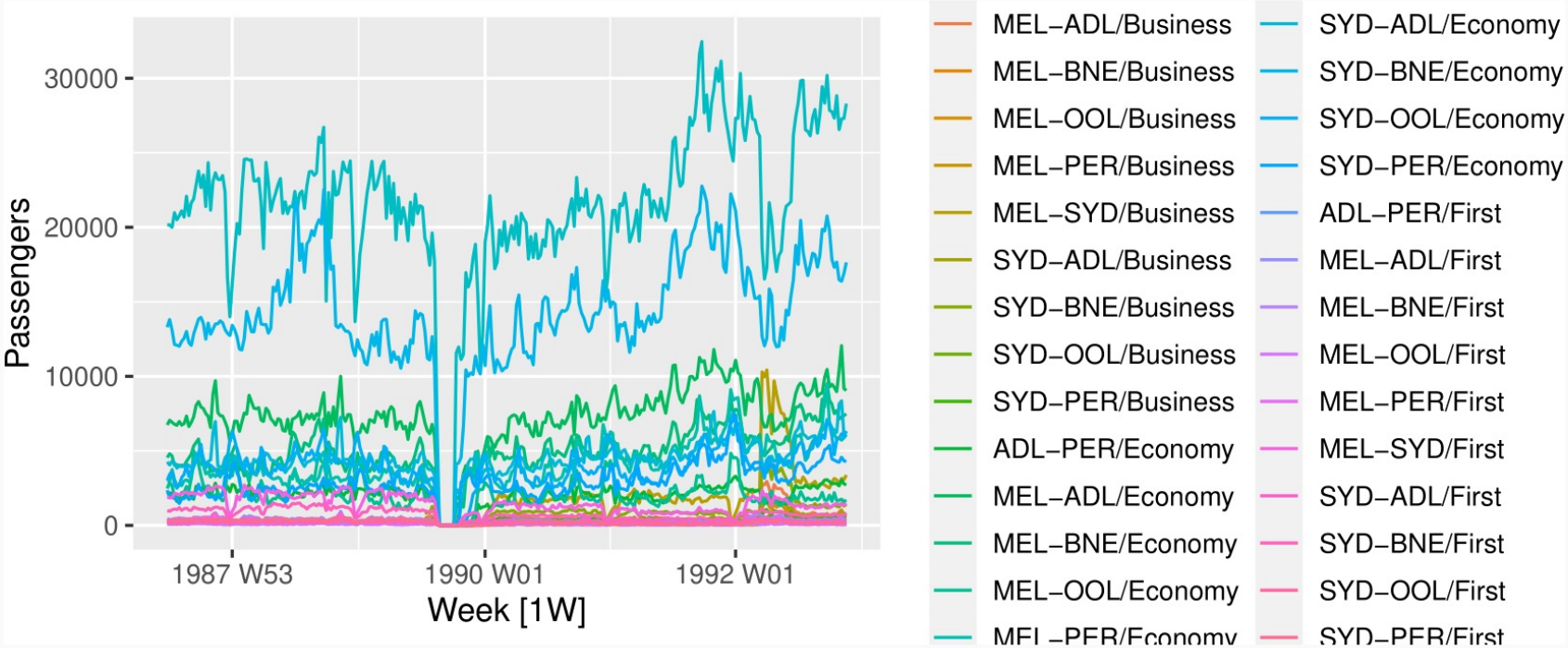
Time plots

```
a10 %>%
  autoplot(total_cost)
```



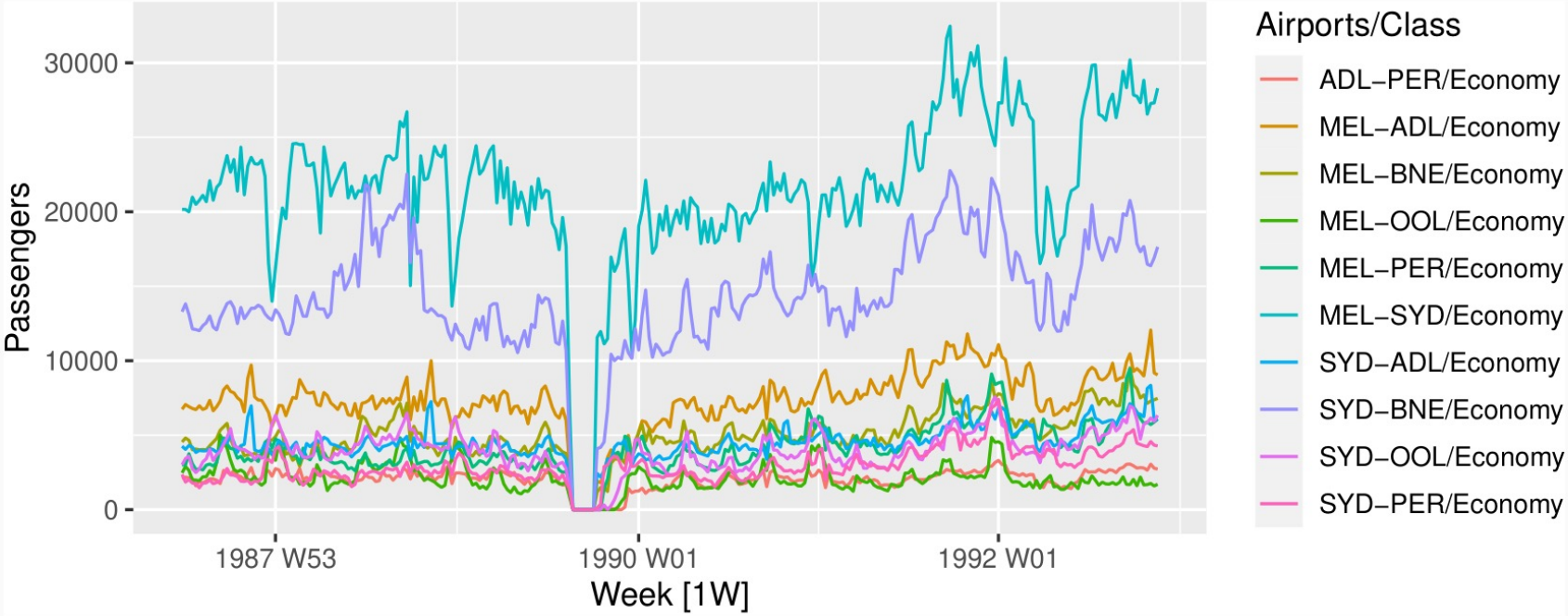
Ansett airlines

```
ansett %>%
  autoplot(Passengers)
```



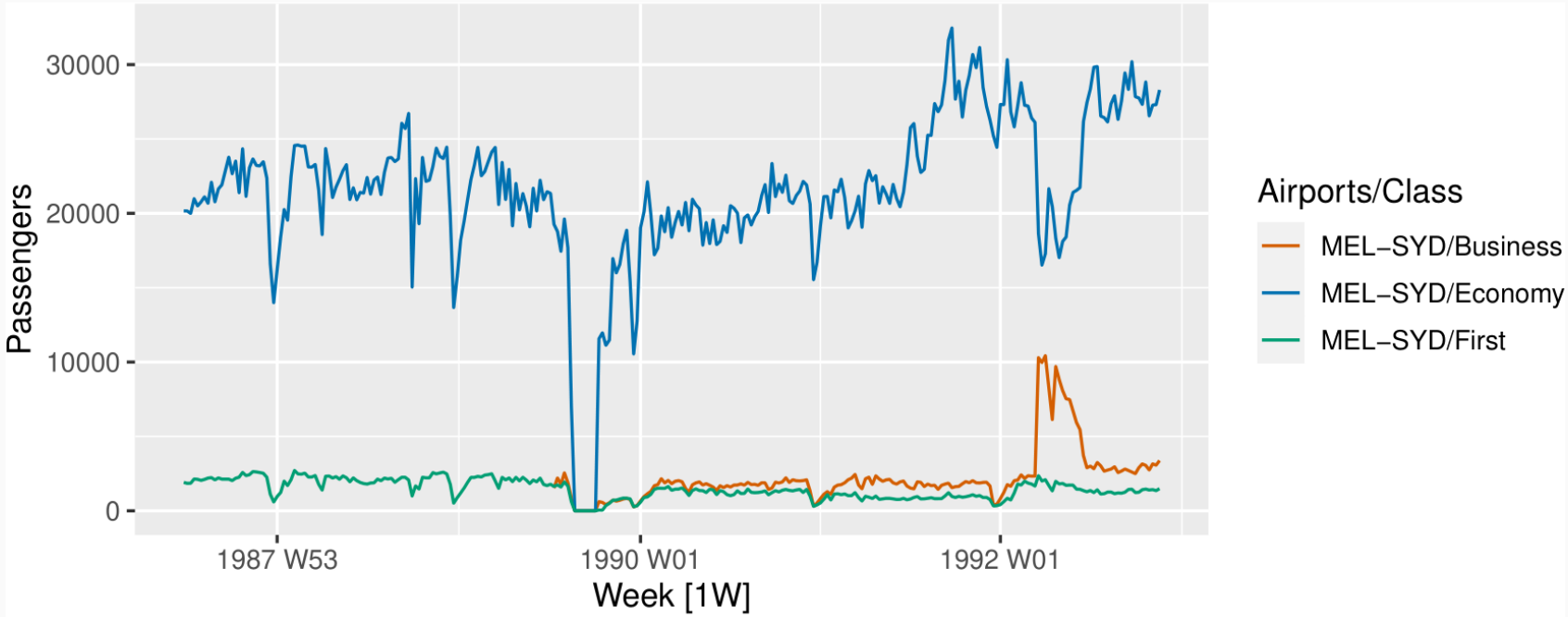
Ansett airlines

```
ansett %>%
  filter(Class == "Economy") %>%
  autoplot(Passengers)
```



Ansett airlines

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



Time series patterns

Trend pattern exists when there is a long-term increase or decrease in the data.

Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).

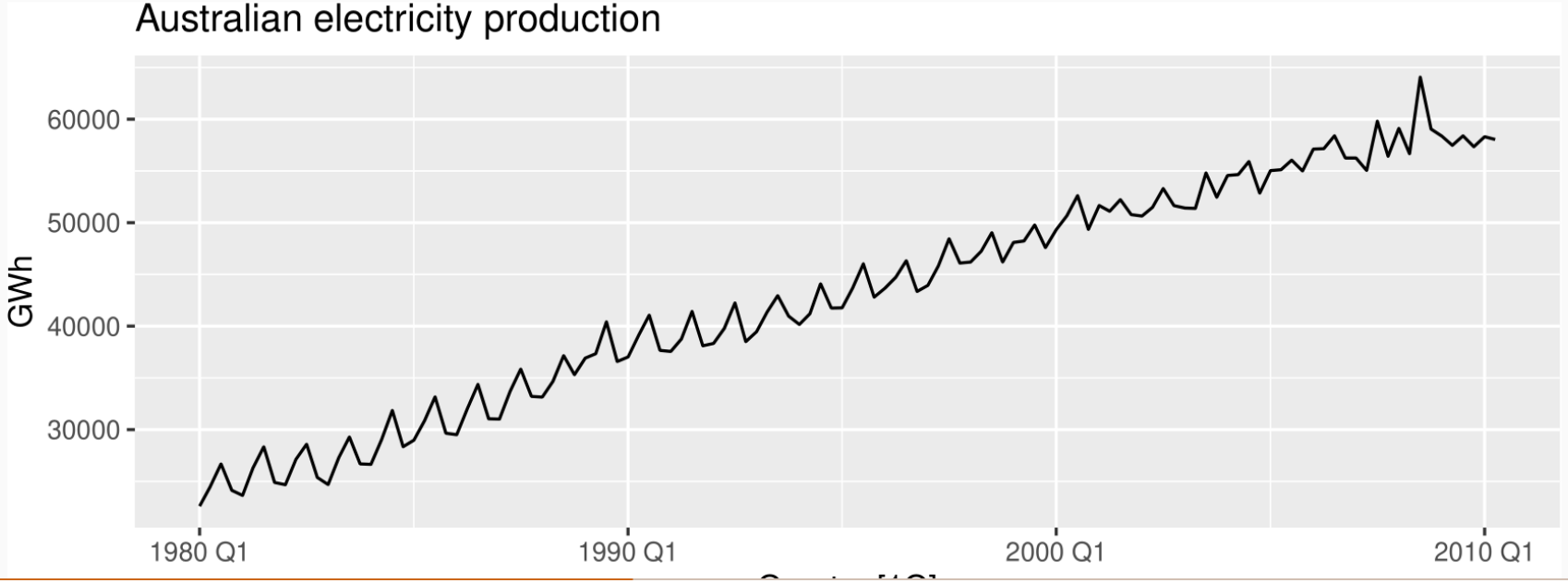
Time series components

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

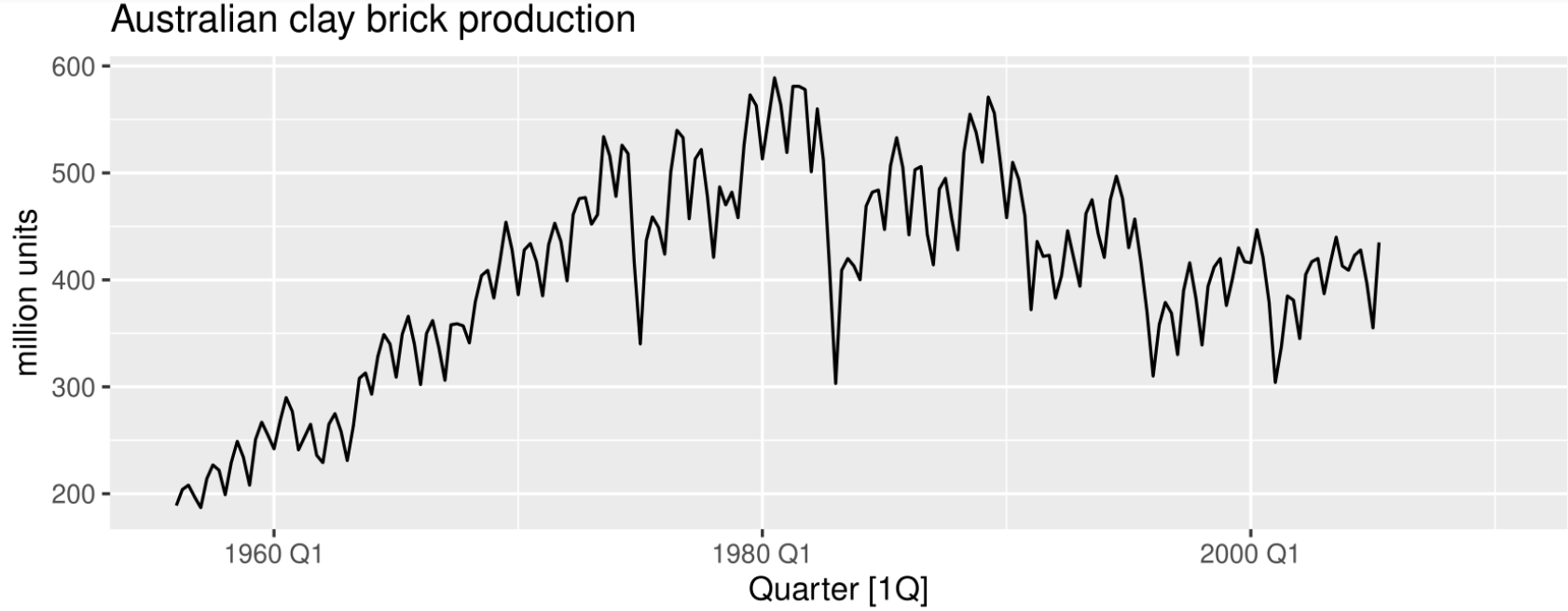
Time series patterns

```
aus_production %>%
  filter(year(Quarter) >= 1980) %>%
  autoplot(Electricity) +
  labs(y = "GWh", title = "Australian electricity production")
```



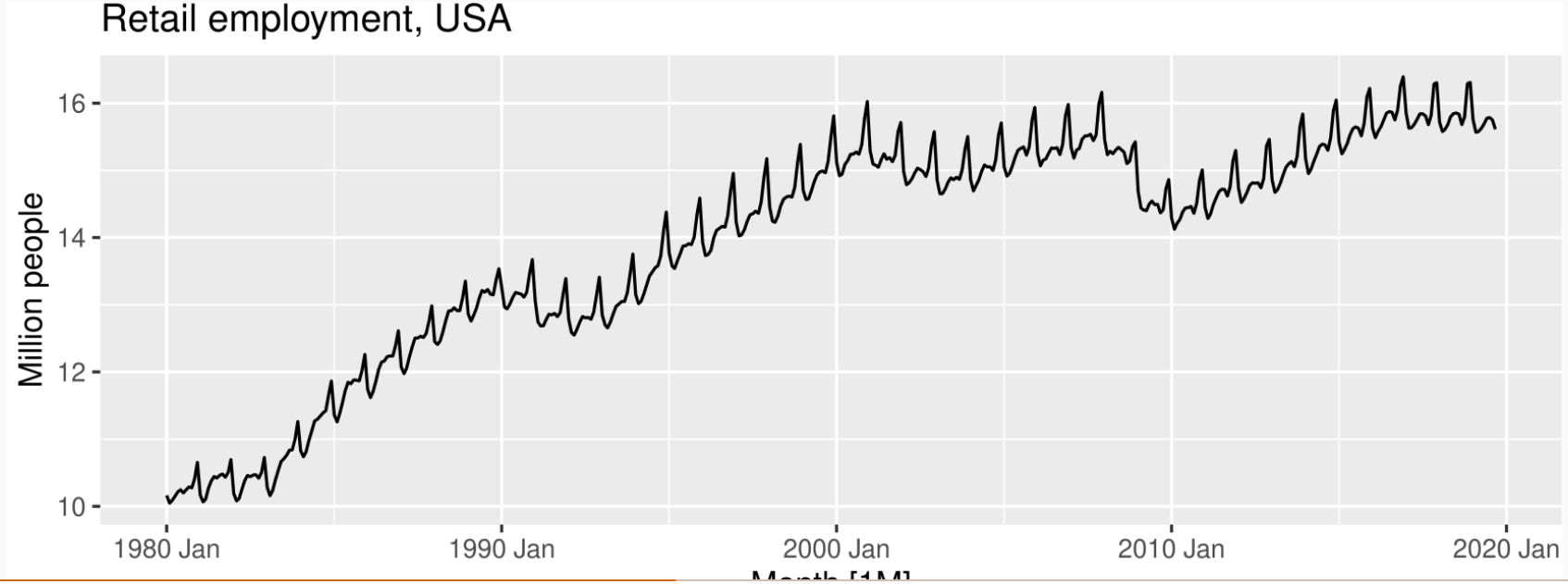
Time series patterns

```
aus_production %>%
  autoplot(Bricks) +
  labs(y = "million units", title = "Australian clay brick production")
```



Time series patterns

```
us_employment %>%
  filter(Title == "Retail Trade", year(Month) >= 1980) %>%
  autoplot(Employed / 1e3) +
  labs(y = "Million people", title = "Retail employment, USA")
```

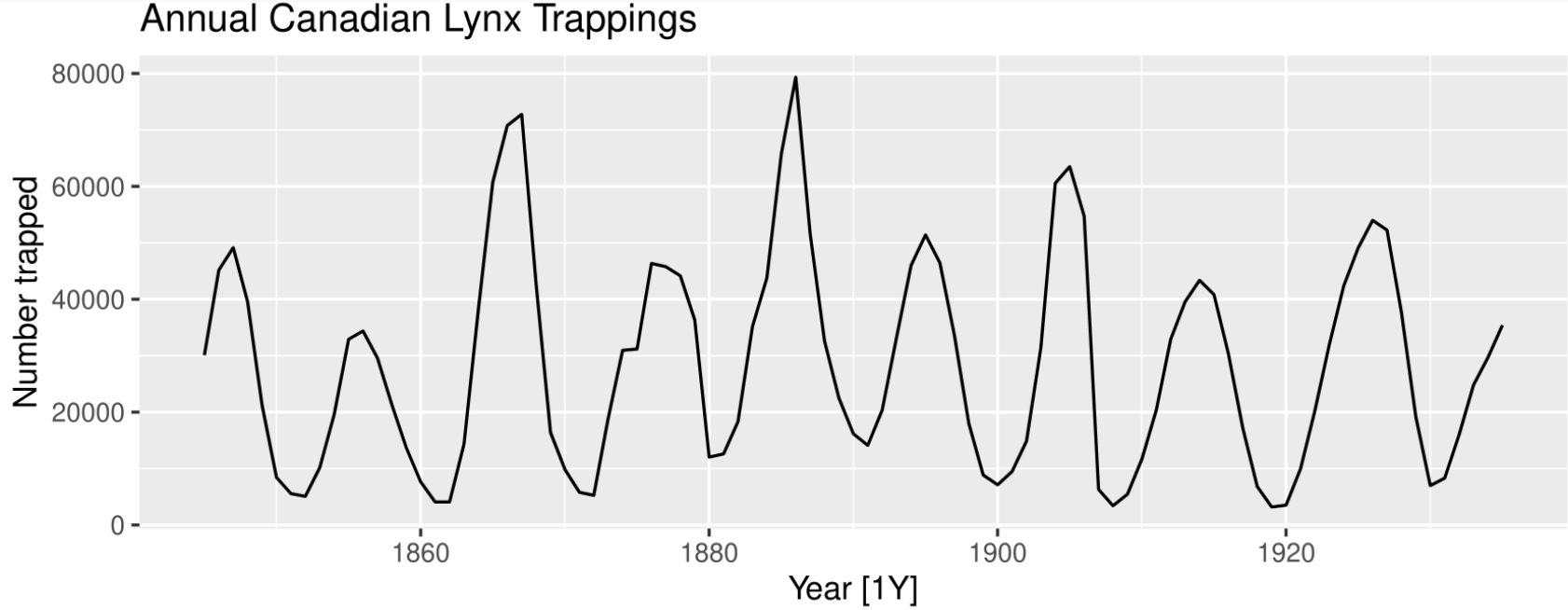


Time series patterns

```
gafa_stock %>%
  filter(Symbol == "AMZN", year(Date) >= 2018) %>%
  autoplot(Close) +
  labs(y = "$US", title = "Amazon closing stock price")
```

Time series patterns

```
pelt %>%
  autoplot(Lynx) +
  labs(y="Number trapped", title = "Annual Canadian Lynx Trappings")
```



Seasonal or cyclic?

Differences between seasonal and cyclic patterns:

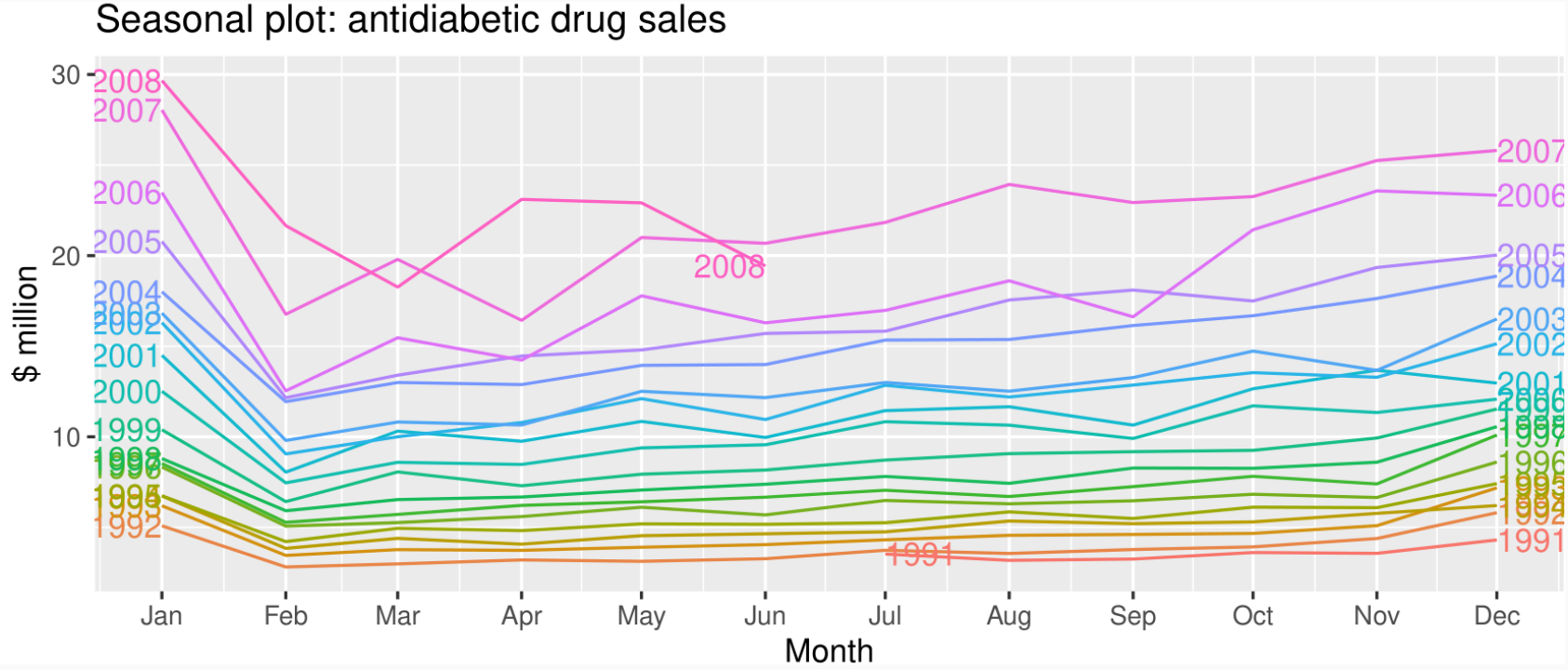
- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

The timing of peaks and troughs is predictable with seasonal data, but unpredictable in the long term with cyclic data.

Outline

Seasonal plots

```
a10 %>% gg_season(total_cost, labels = "both") +
  labs(y = "$ million", title = "Seasonal plot: antidiabetic drug sales")
```

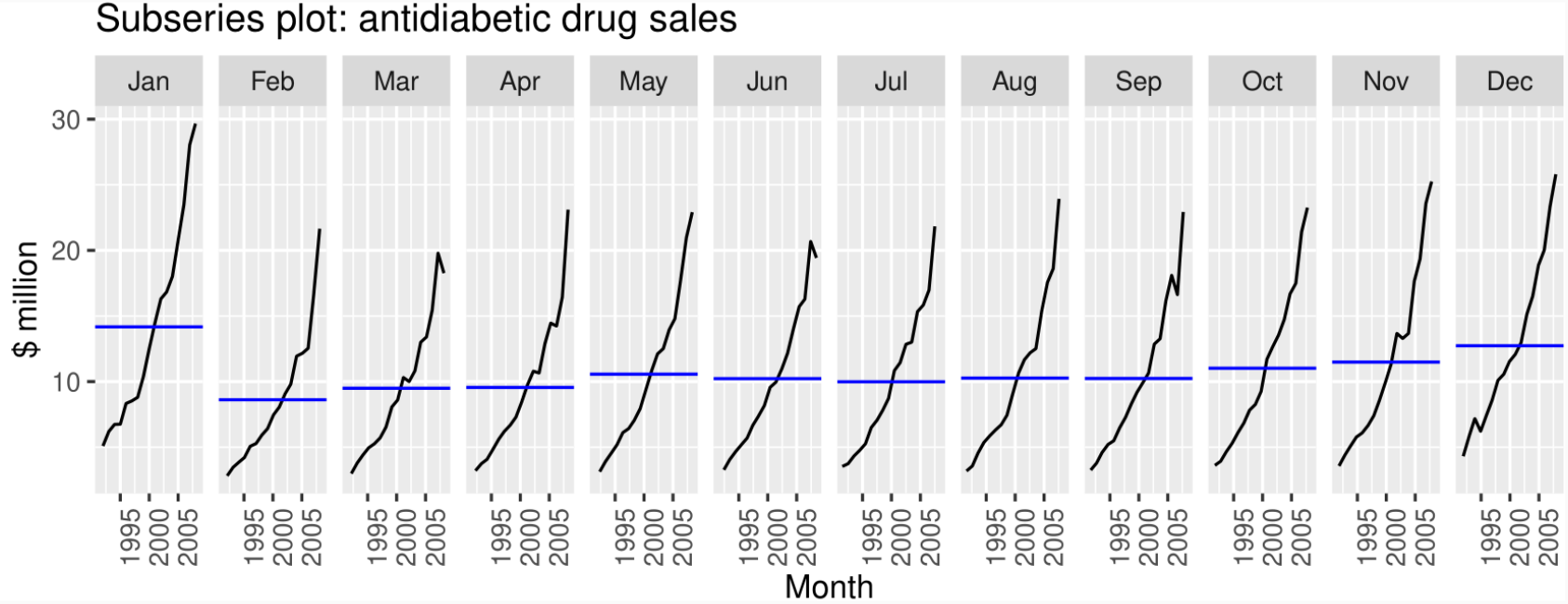


Seasonal plots

- Data plotted against the individual “seasons” in which the data were observed. (In this case a “season” is a month.)
- 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.
- In R: `gg_season()`

Seasonal subseries plots

```
a10 %>%
  gg_subseries(total_cost) +
  labs(y = "$ million", title = "Subseries plot: antidiabetic drug sales")
```

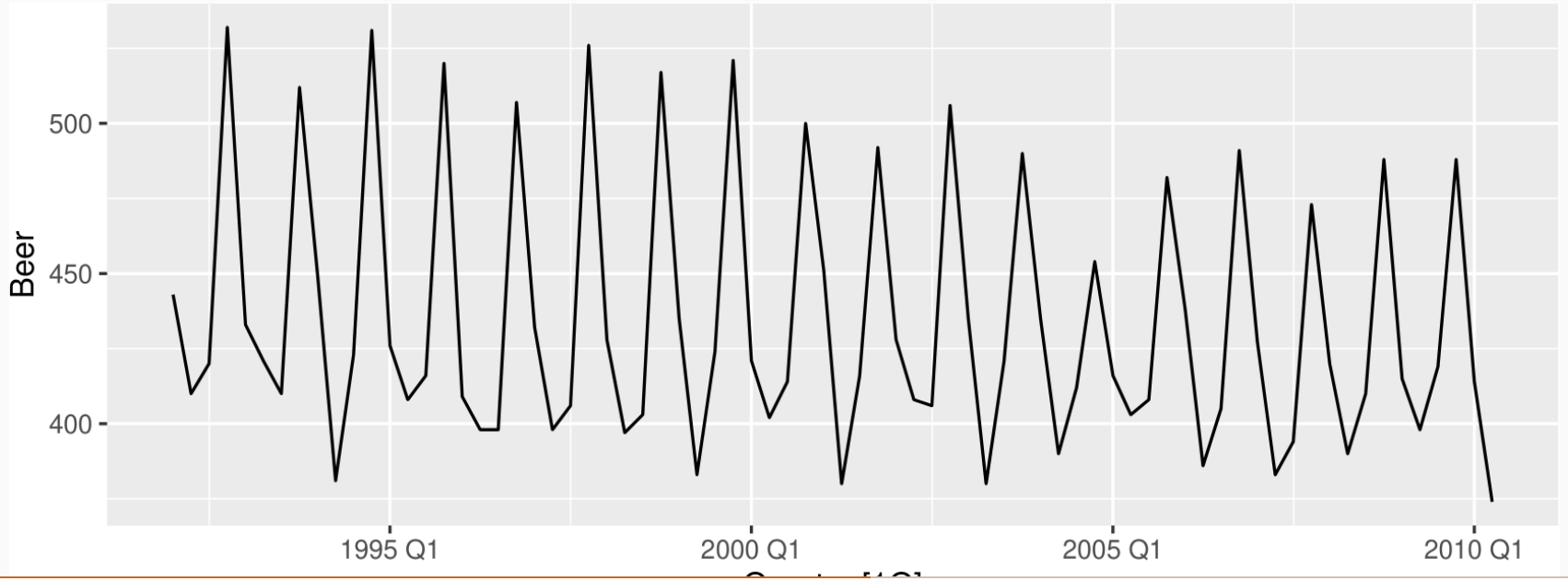


Seasonal subseries plots

- 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.
- In R: `gg_subseries()`

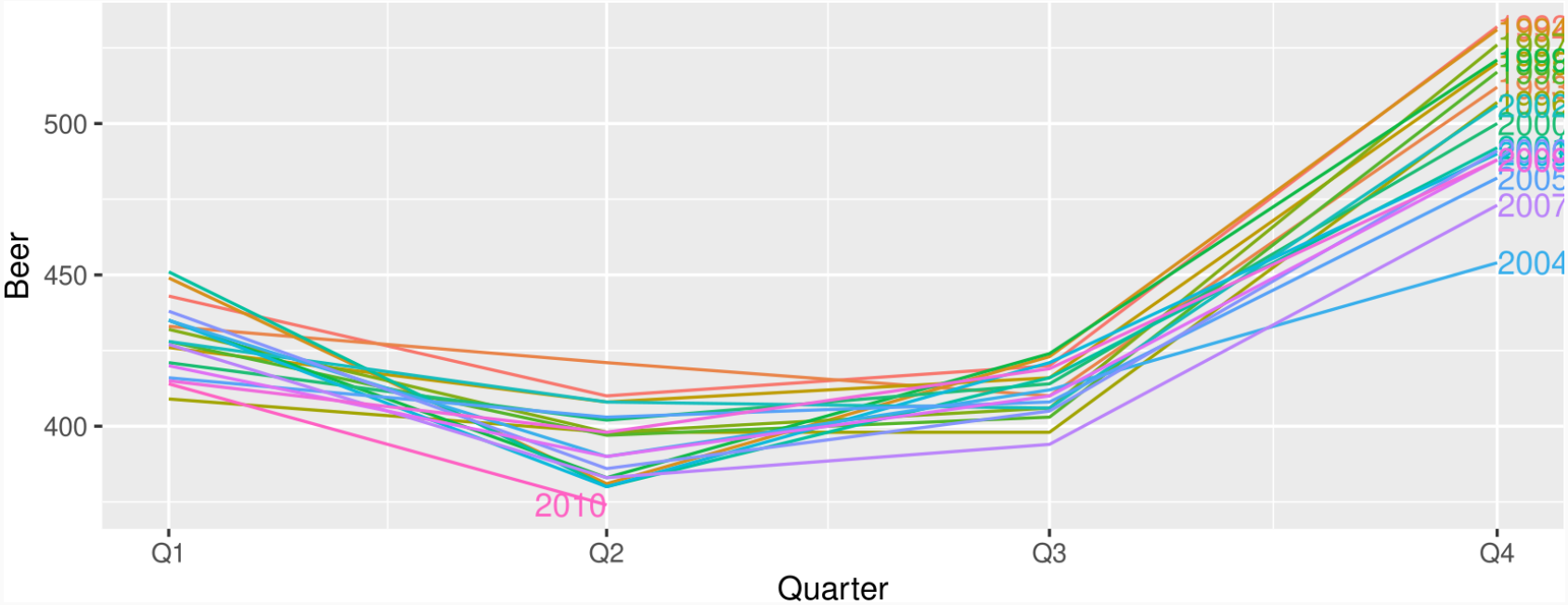
Quarterly Australian Beer Production

```
beer <- aus_production %>%
  select(Quarter, Beer) %>%
  filter(year(Quarter) >= 1992)
beer %>% autoplot(Beer)
```



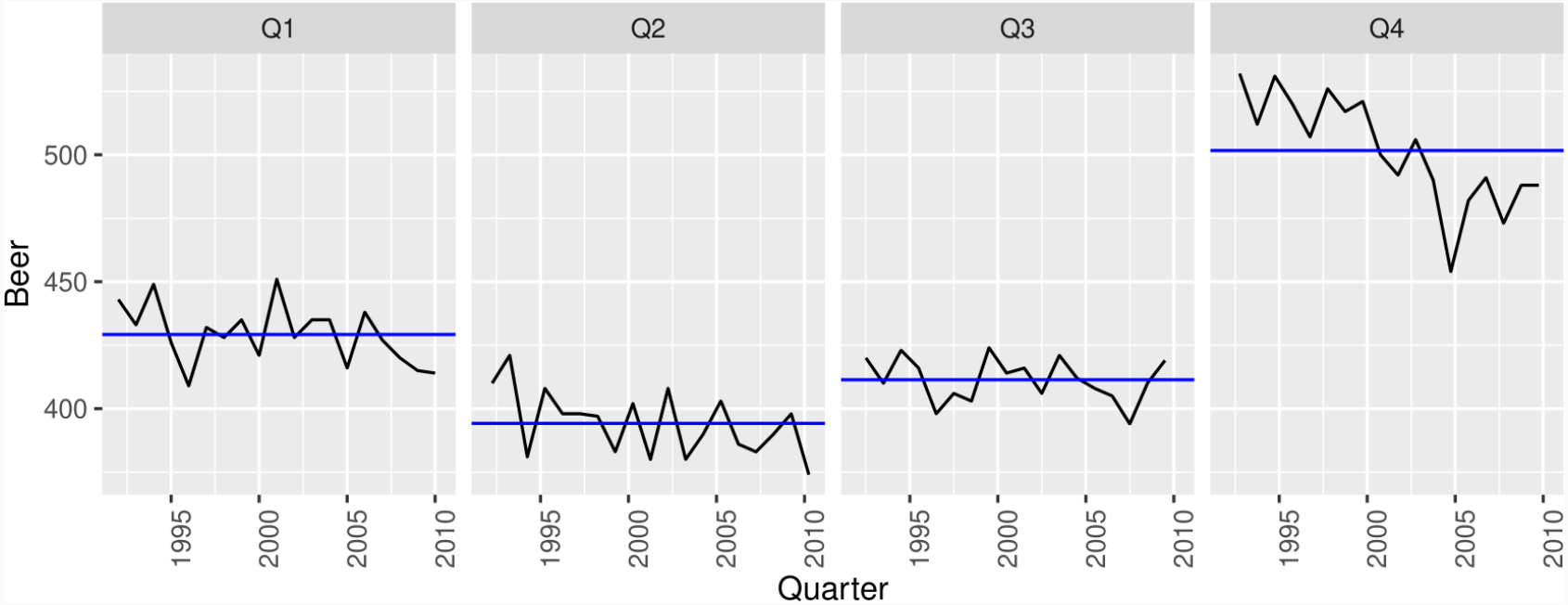
Quarterly Australian Beer Production

```
beer %>% gg_season(Beer, labels="right")
```



Quarterly Australian Beer Production

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



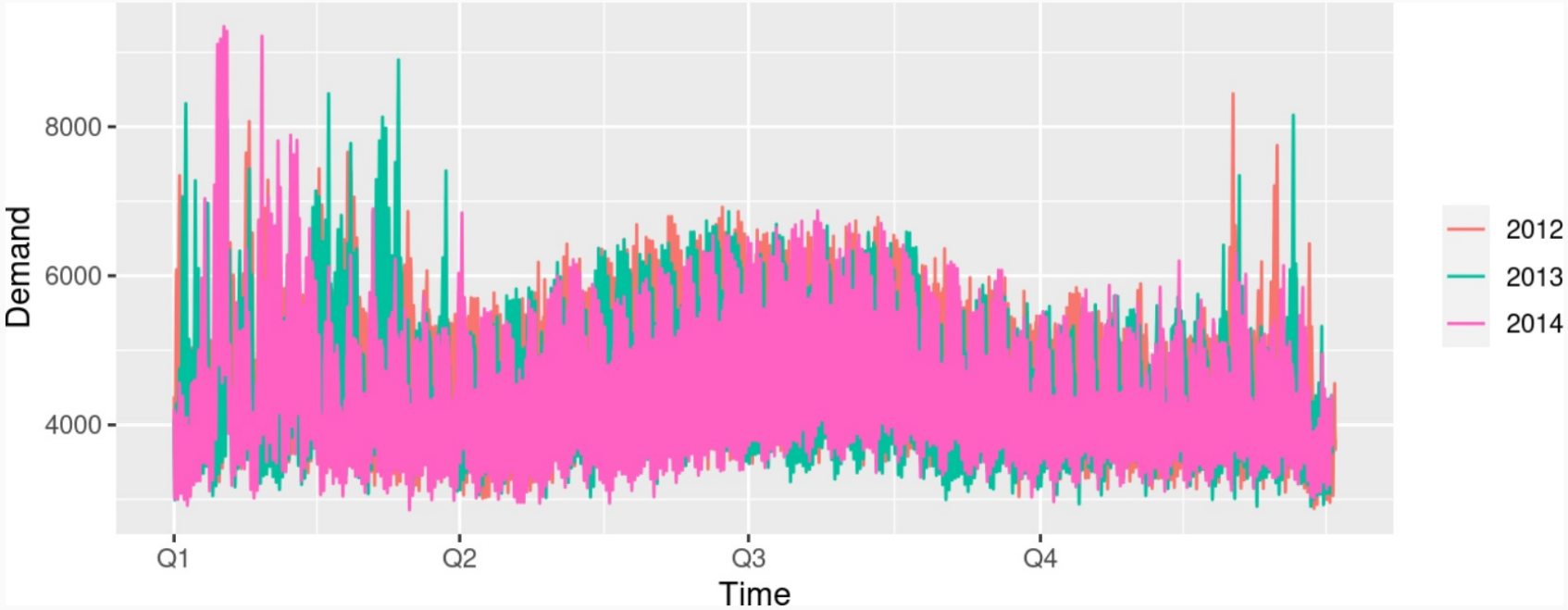
Multiple seasonal periods

```
vic_elec
```

```
## # A tsibble: 52,608 x 5 [30m] <Australia/Melbourne>
##    Time                Demand Temperature Date       Holiday
##    <dttm>               <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
```

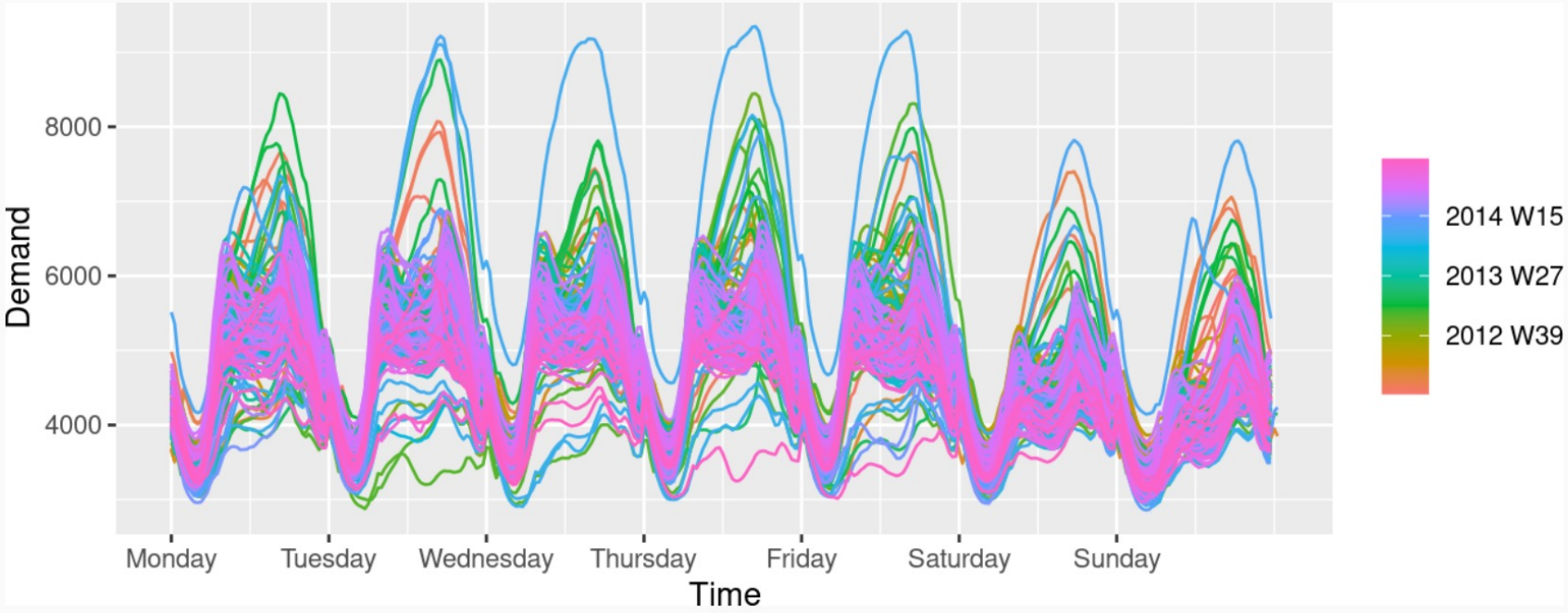
Multiple seasonal periods

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



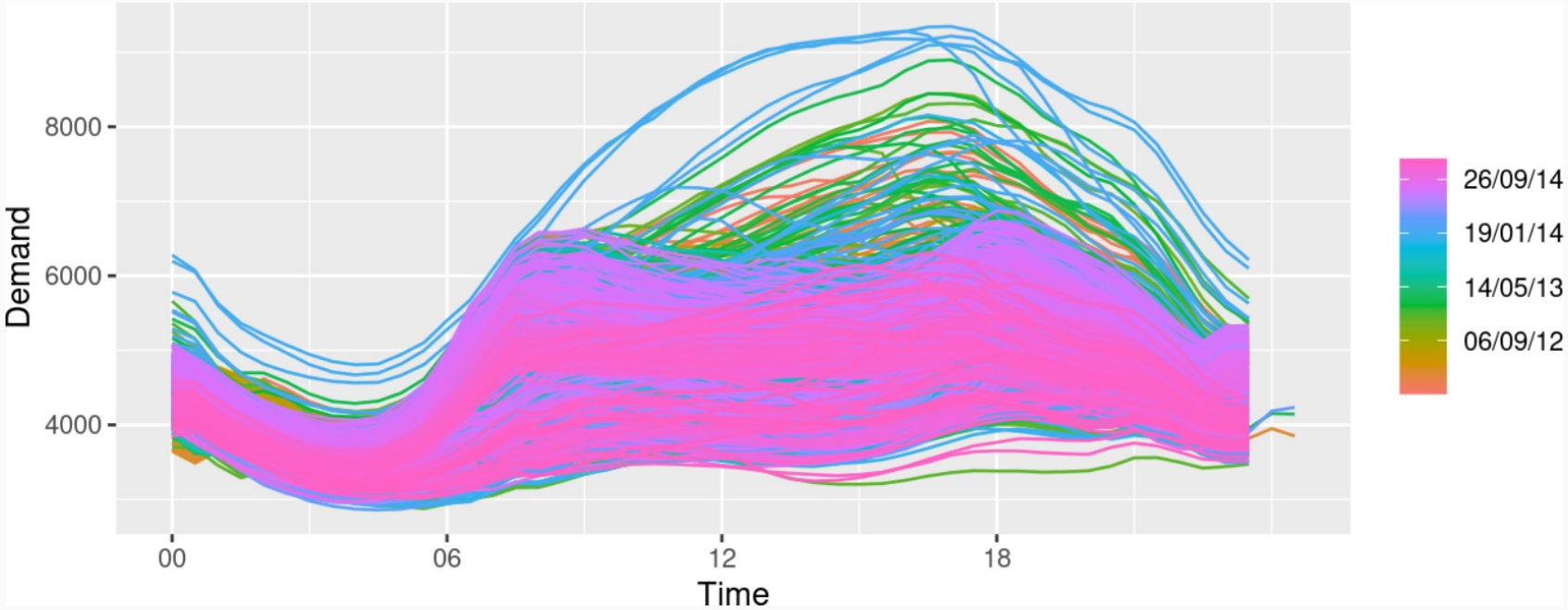
Multiple seasonal periods

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



Multiple seasonal periods

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



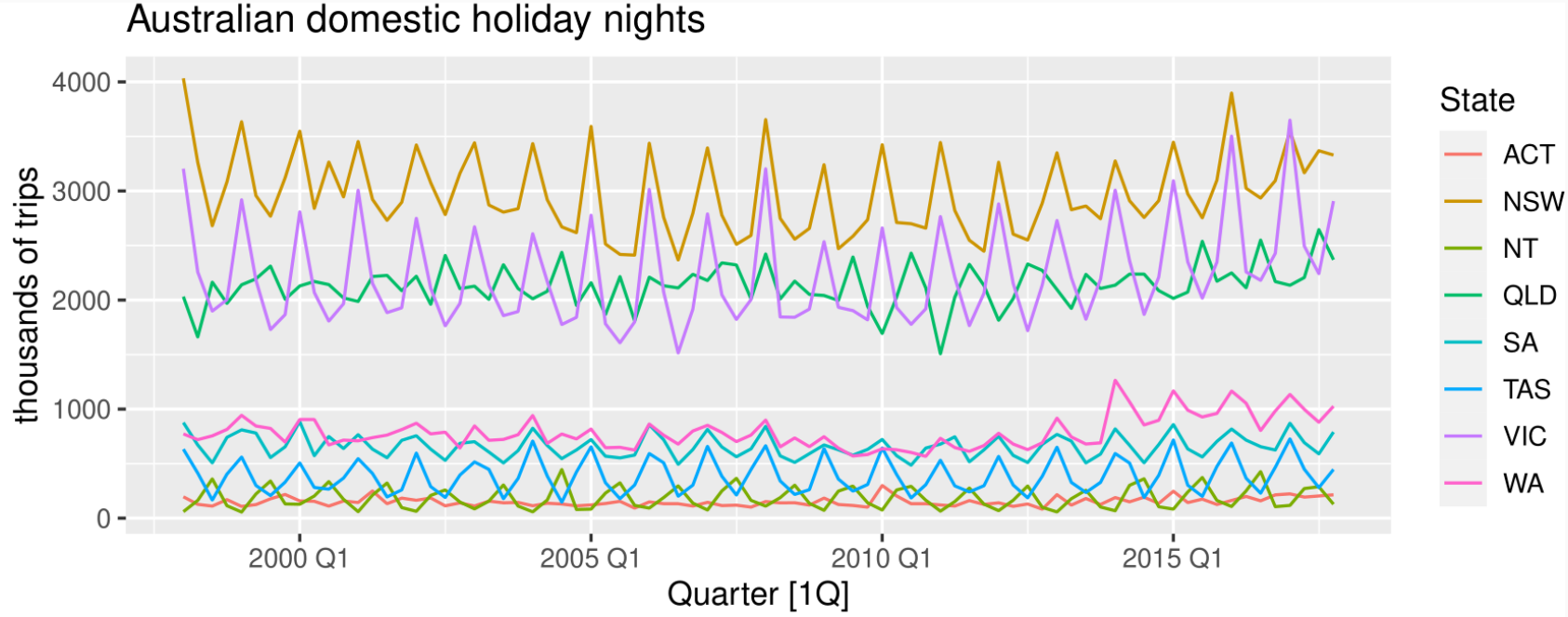
Australian holidays

```
holidays <- tourism %>%
  filter(Purpose == "Holiday") %>%
  group_by(State) %>%
  summarise(Trips = sum(Trips))
```

```
## # A tsibble: 640 x 3 [1Q]
## # Key:       State [8]
##    State Quarter Trips
##    <chr>   <qtr> <dbl>
##  1 ACT   1998 Q1  196.
##  2 ACT   1998 Q2  127.
##  3 ACT   1998 Q3  111.
##  4 ACT   1998 Q4  170.
##  5 ACT   1999 Q1  108.
##  6 ACT   1999 Q2  125.
##  7 ACT   1999 Q3  178.
##  8 ACT   1999 Q4  218.
##  9 ACT   2000 Q1  158.
```

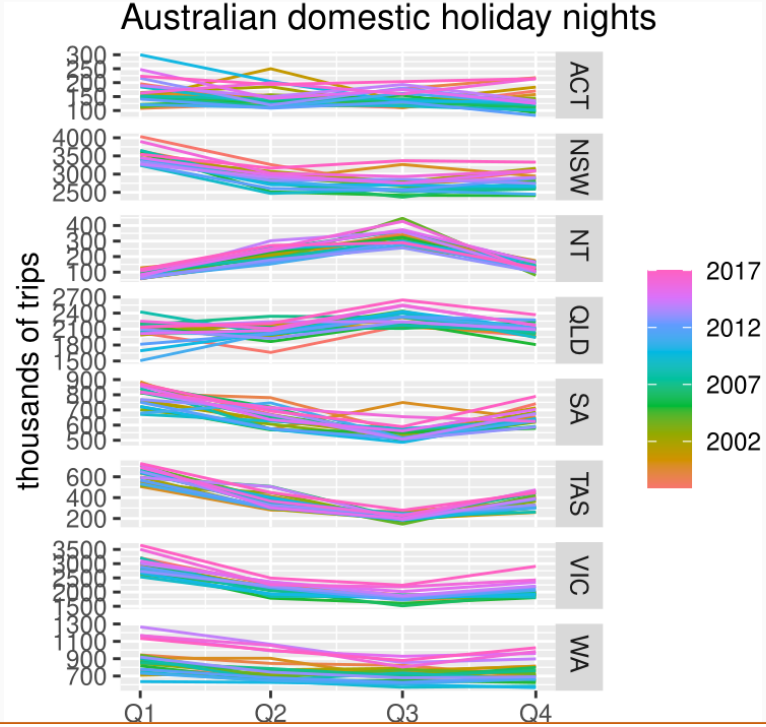
Australian holidays

```
holidays %>% autoplot(Trips) +
  labs(y = "thousands of trips", title = "Australian domestic holiday nights")
```



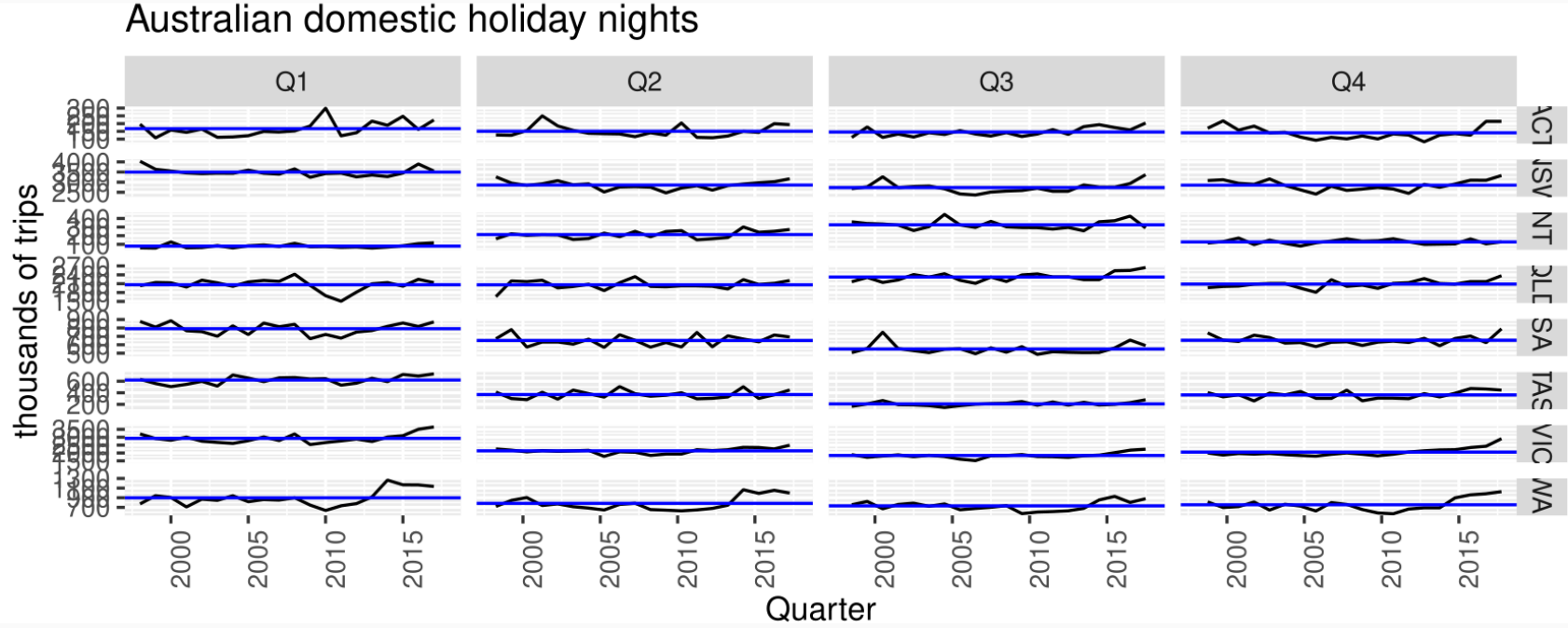
Seasonal plots

```
holidays %>% gg_season(Trips) +
  labs(y = "thousands of trips", title = "Australian domestic holiday nights")
```



Seasonal subseries plots

```
holidays %>%
  gg_subseries(Trips) +
  labs(y = "thousands of trips", title = "Australian domestic holiday nights")
```



Outline

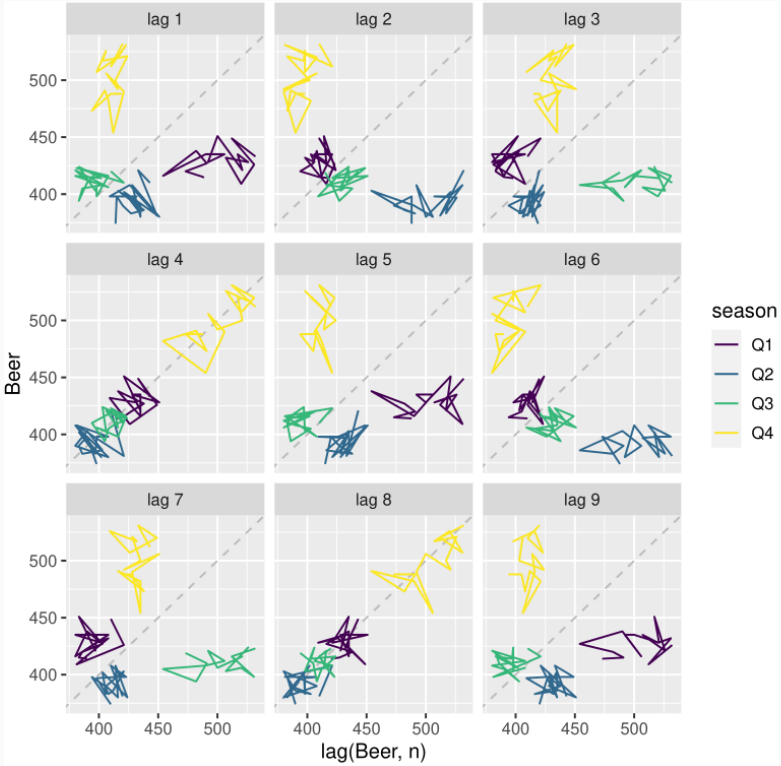
Example: Beer production

```
new_production <- aus_production %>%
  filter(year(Quarter) >= 1992)
new_production
```

```
## # A tsibble: 74 x 7 [1Q]
##    Quarter  Beer Tobacco Bricks Cement Electricity   Gas
##      <qtr> <dbl>   <dbl>  <dbl>  <dbl>       <dbl> <dbl>
##  1 1992 Q1   443    5777    383   1289       38332   117
##  2 1992 Q2   410    5853    404   1501       39774   151
##  3 1992 Q3   420    6416    446   1539       42246   175
##  4 1992 Q4   532    5825    420   1568       38498   129
##  5 1993 Q1   433    5724    394   1450       39460   116
##  6 1993 Q2   421    6036    462   1668       41356   149
##  7 1993 Q3   410    6570    475   1648       42949   163
##  8 1993 Q4   512    5675    443   1863       40974   138
##  9 1994 Q1   449    5311    421   1468       40162   127
## 10 1994 Q2   381    5717    475   1755       41199   159
## # ... with 64 more rows
```

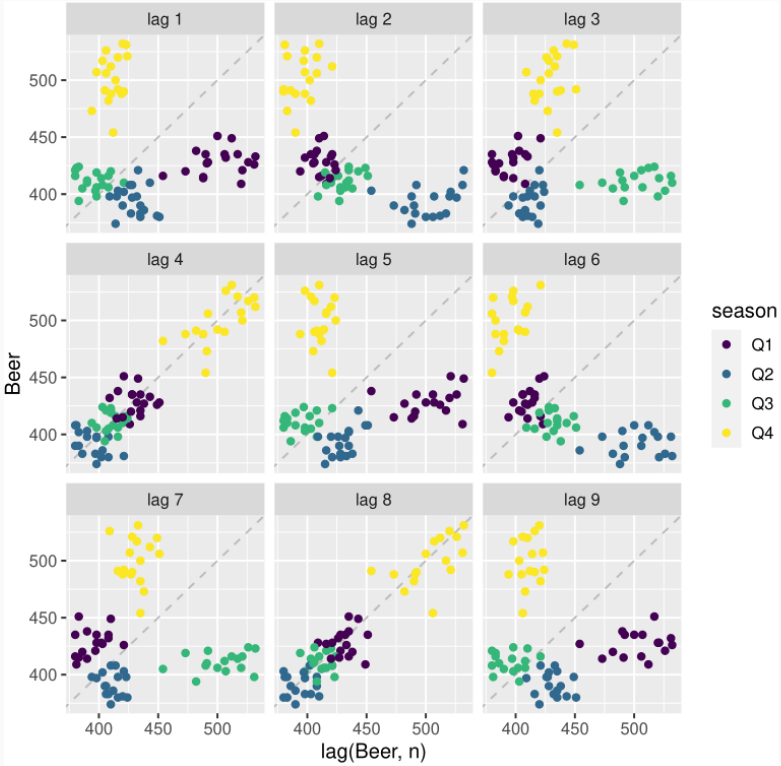
Example: Beer production

```
new_production %>% gg_lag(Beer)
```



Example: Beer production

```
new_production %>% gg_lag(Beer, geom='point')
```



Lagged scatterplots

- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.
- ACF (autocorrelation function):
  - r_1 = Correlation(y_t, y_{t-1})
  - r_2 = Correlation(y_t, y_{t-2})
  - r_3 = Correlation(y_t, y_{t-3})
  - etc.

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocovariance and **autocorrelation**: measure linear relationship between **lagged values** of a time series y.

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocovariance and **autocorrelation**: measure linear relationship between **lagged values** of a time series y.

We measure the relationship between:

- y_t and y_{t-1}
- y_t and y_{t-2}
- y_t and y_{t-3}
- etc.

Autocorrelation

We denote the sample autocovariance at lag k by c_k and the sample autocorrelation at lag k by r_k. Then define

$$c_k = \frac{1}{T}\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})$$

and $$r_k = c_k/c_0$$

- r_1 indicates how successive values of y relate to each other
- r_2 indicates how y values two periods apart relate to each other
- r_k is *almost* the same as the sample correlation between y_t and y_{t-k}.

Autocorrelation

Results for first 9 lags for beer data:

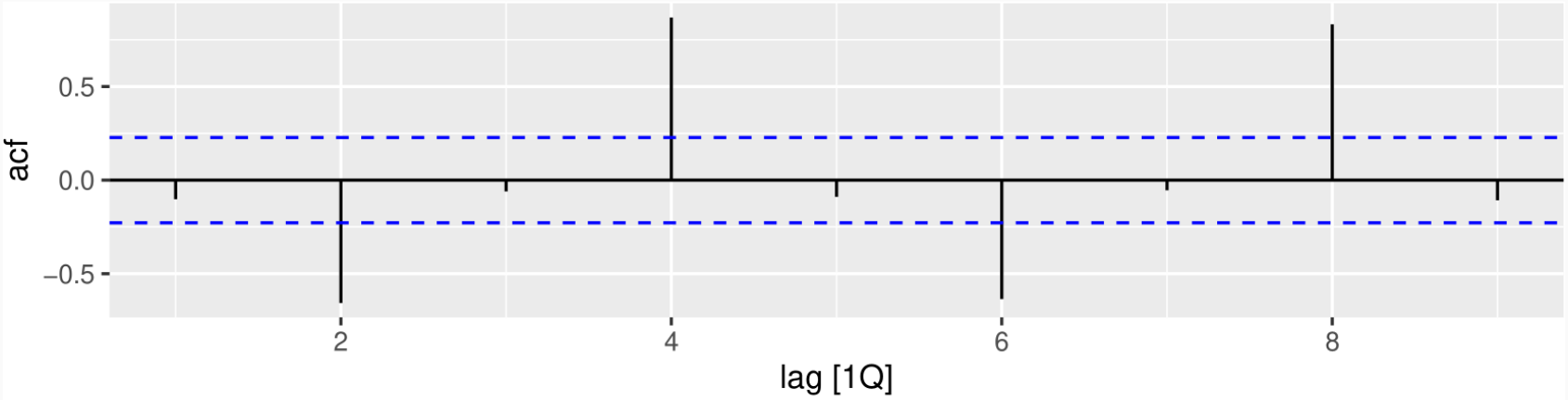
```
new_production %>% ACF(Beer, lag_max = 9)
```

```
## # A tsibble: 9 x 2 [1Q]
##     lag     acf
##   <lag>   <dbl>
## 1    1Q -0.102
## 2    2Q -0.657
## 3    3Q -0.0603
## 4    4Q  0.869
## 5    5Q -0.0892
## 6    6Q -0.635
## 7    7Q -0.0542
```

Autocorrelation

Results for first 9 lags for beer data:

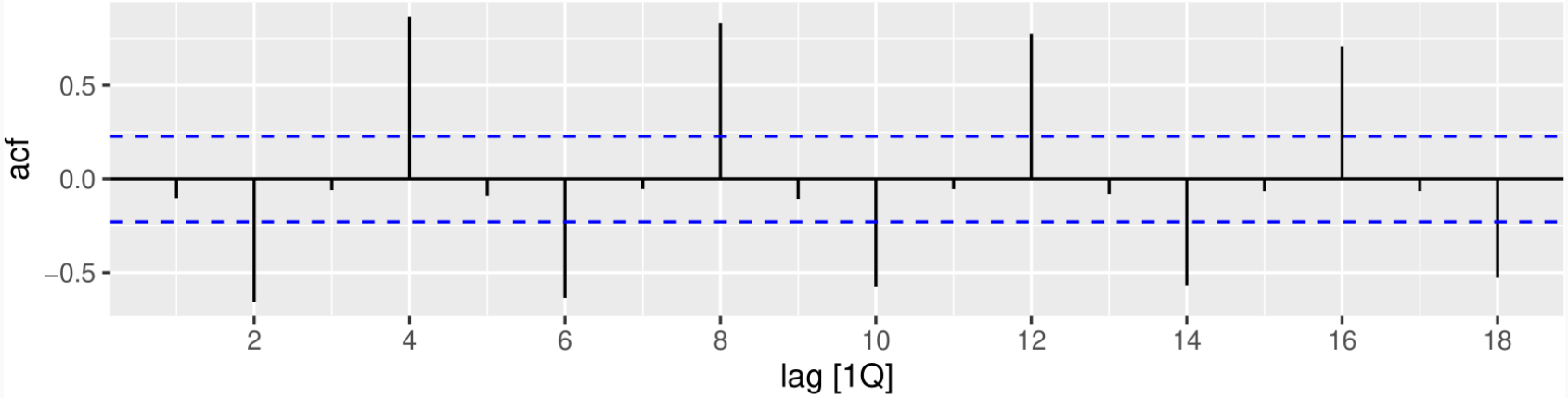
```
new_production %>% ACF(Beer, lag_max = 9) %>% autoplot()
```



- Together, the autocorrelations at lags 1, 2, . . . , make up the *autocorrelation* or ACF.
- The plot is known as a **correlogram**

Autocorrelation

```
new_production %>% ACF(Beer) %>% autoplot()
```

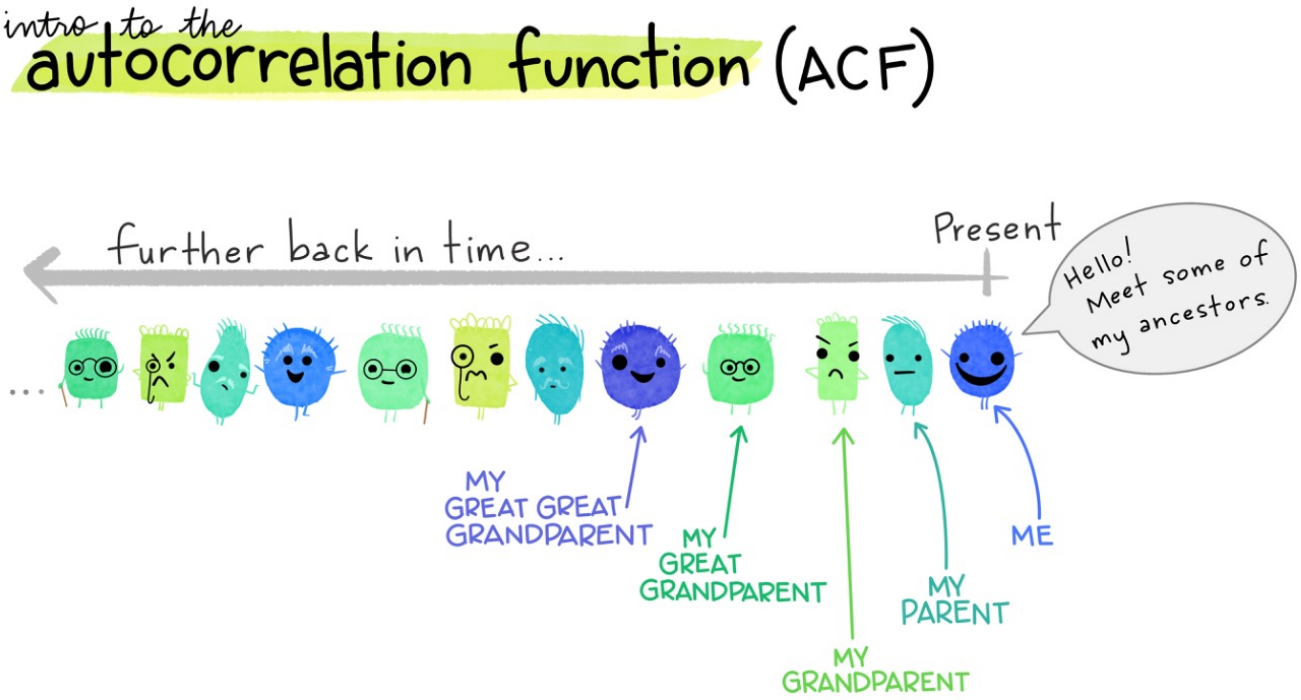


- r_4 higher than for the other lags. This is due to **the seasonal pattern in the data**: the peaks tend to be **4 quarters** apart and the troughs tend to be **2 quarters** apart.
- r_2 is more negative than for the other lags because troughs tend to be 2 quarters behind peaks.

Trend and seasonality in ACF plots

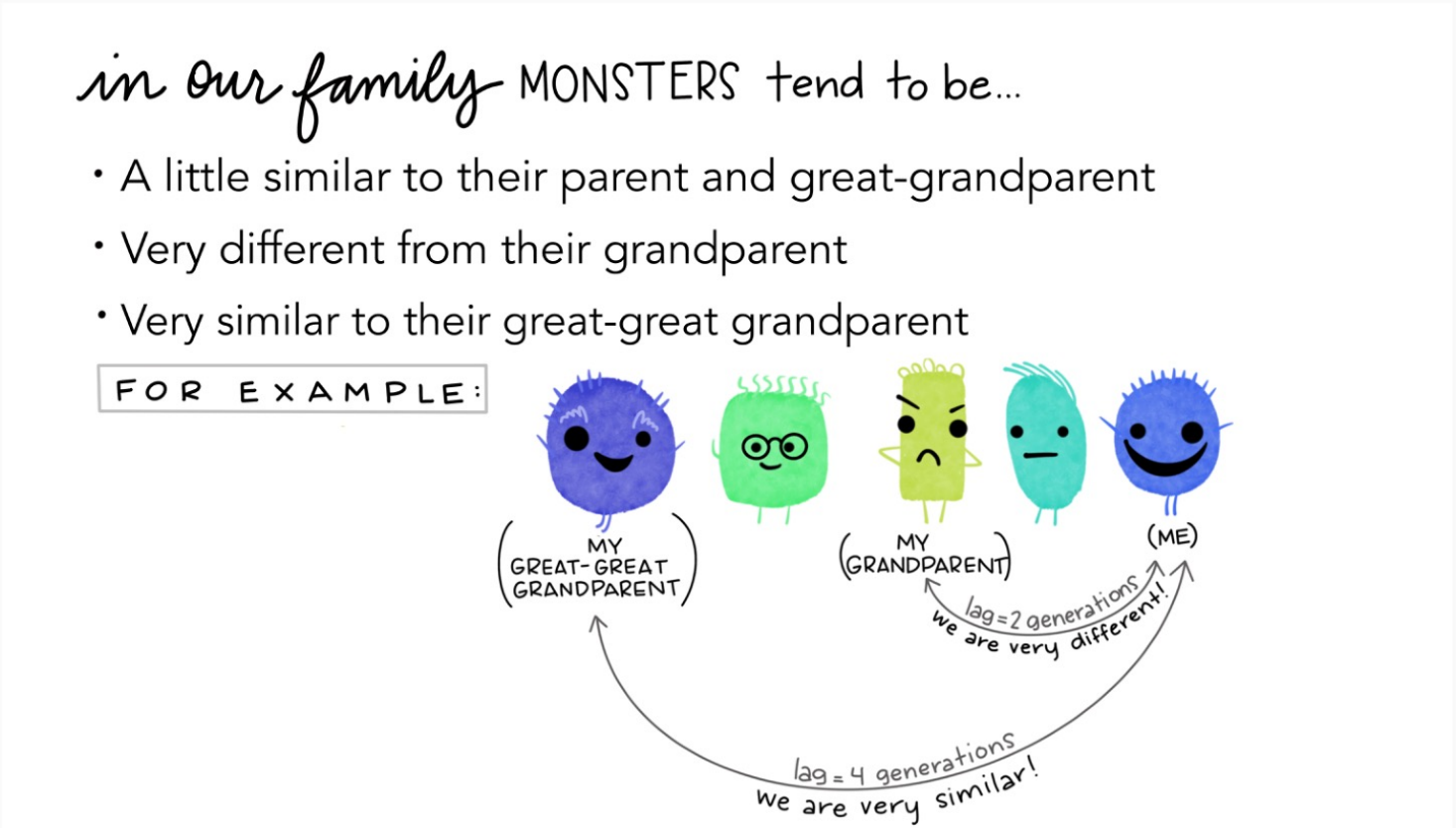
- When data have a trend, the autocorrelations for small lags tend to be large and positive.
- When data are seasonal, the autocorrelations will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency)
- When data are trended and seasonal, you see a combination of these effects.

Autocorrelation functions



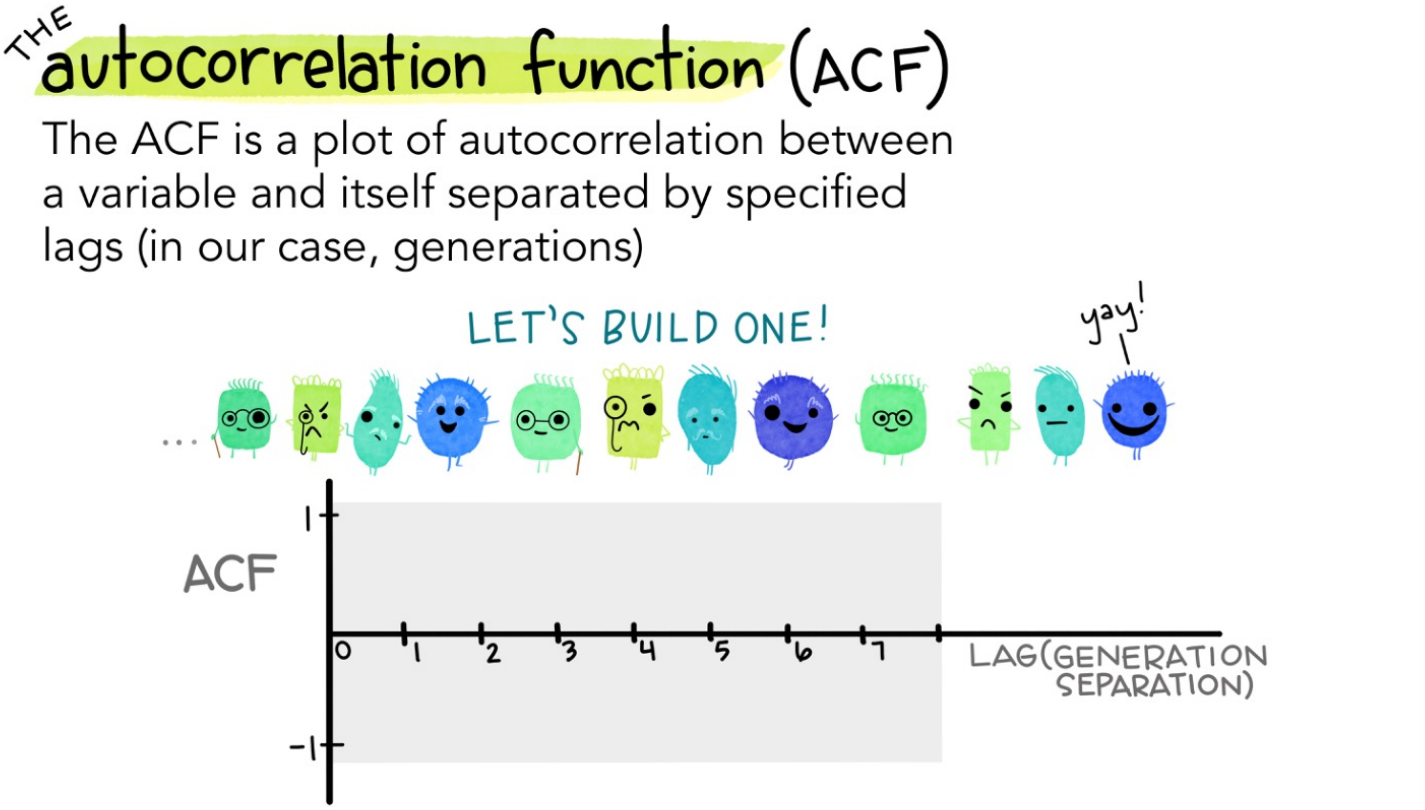
Artwork by @allison_horst

Autocorrelation functions



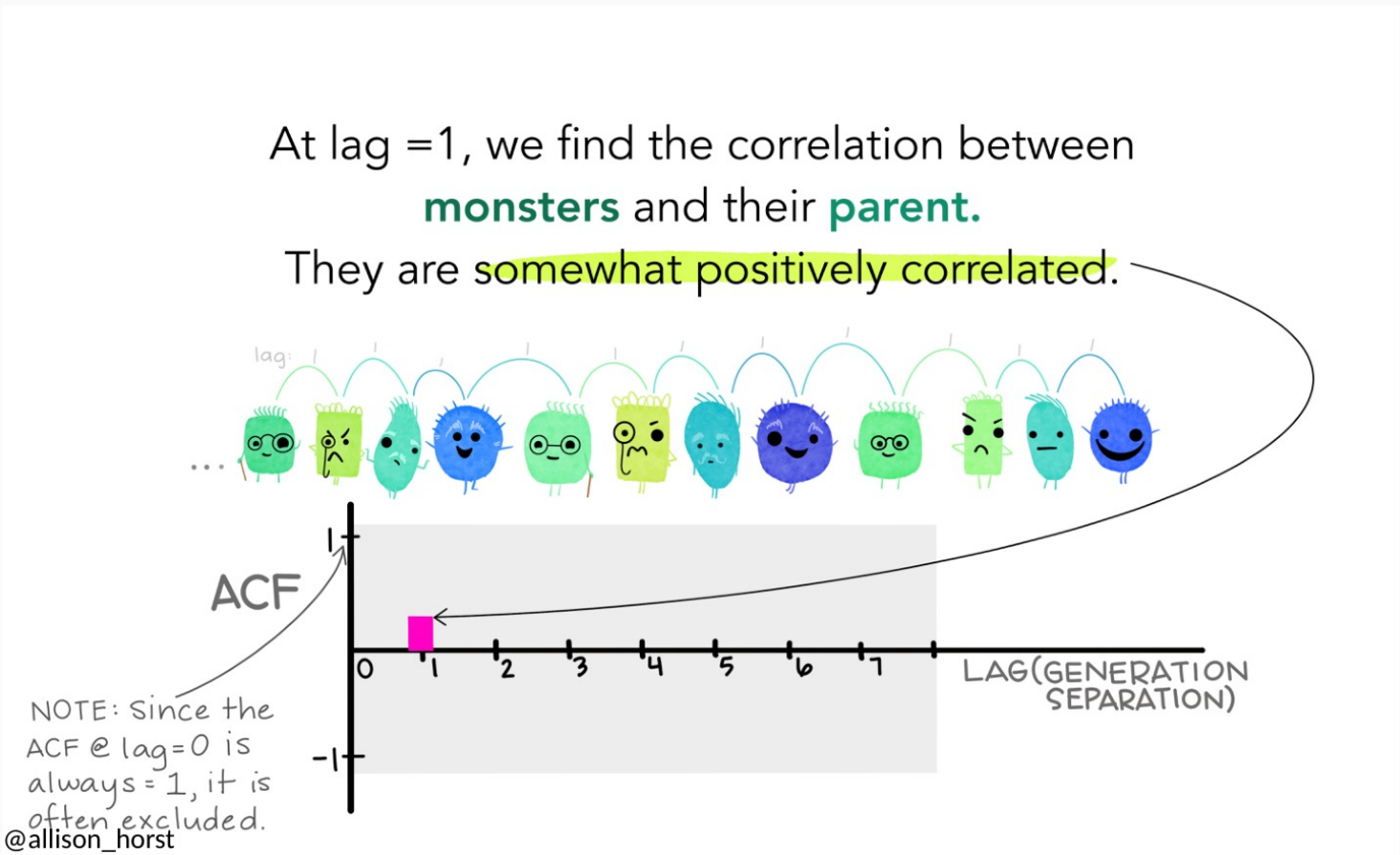
Artwork by @allison_horst

Autocorrelation functions



Artwork by @allison_horst

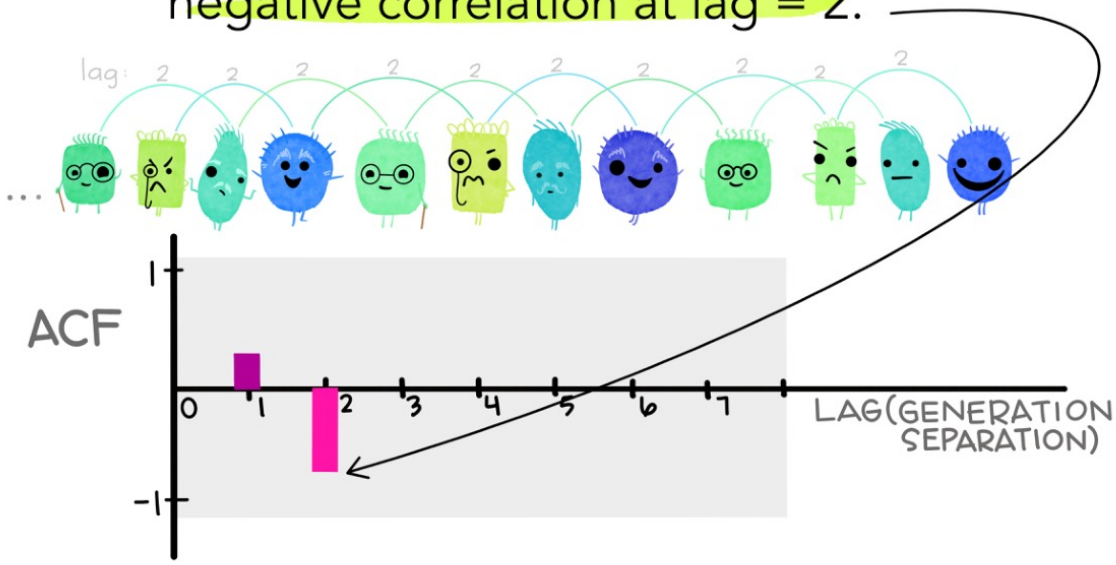
Autocorrelation functions



Artwork by @allison_horst

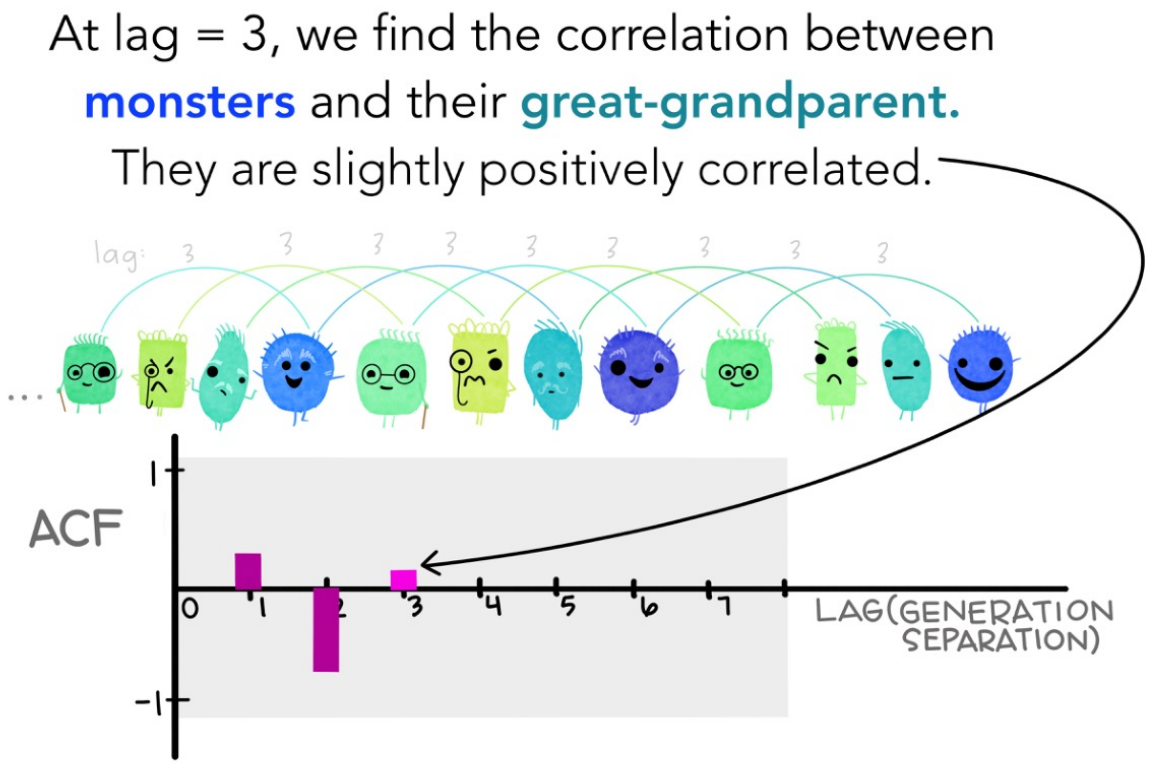
Autocorrelation functions

At lag = 2, we find the correlation between **monsters** and their **grandparent**.
Since they tend to be very different, we find a negative correlation at lag = 2.



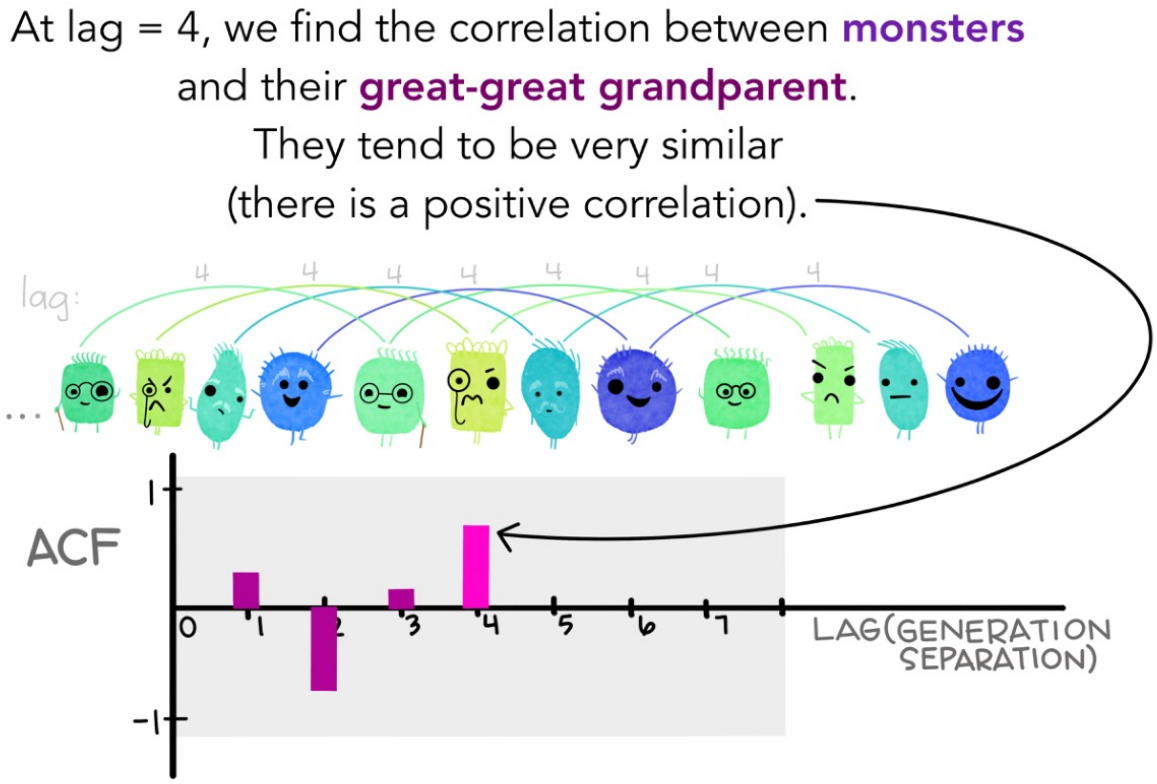
Artwork by @allison_horst

Autocorrelation functions



Artwork by @allison_horst

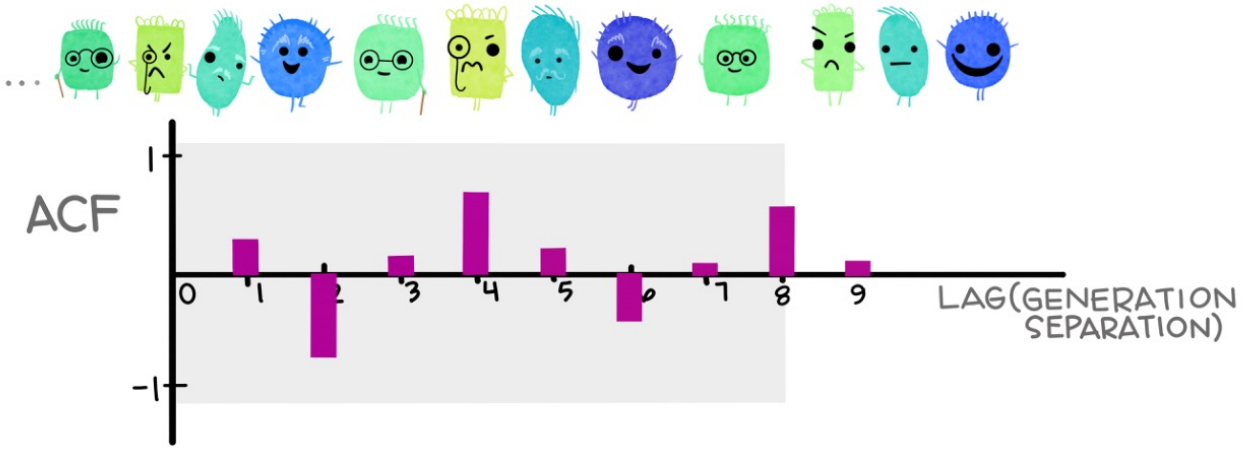
Autocorrelation functions



Artwork by @allison_horst

Autocorrelation functions

...and we continue finding the correlations as we increase the lag (generations) between the monsters...

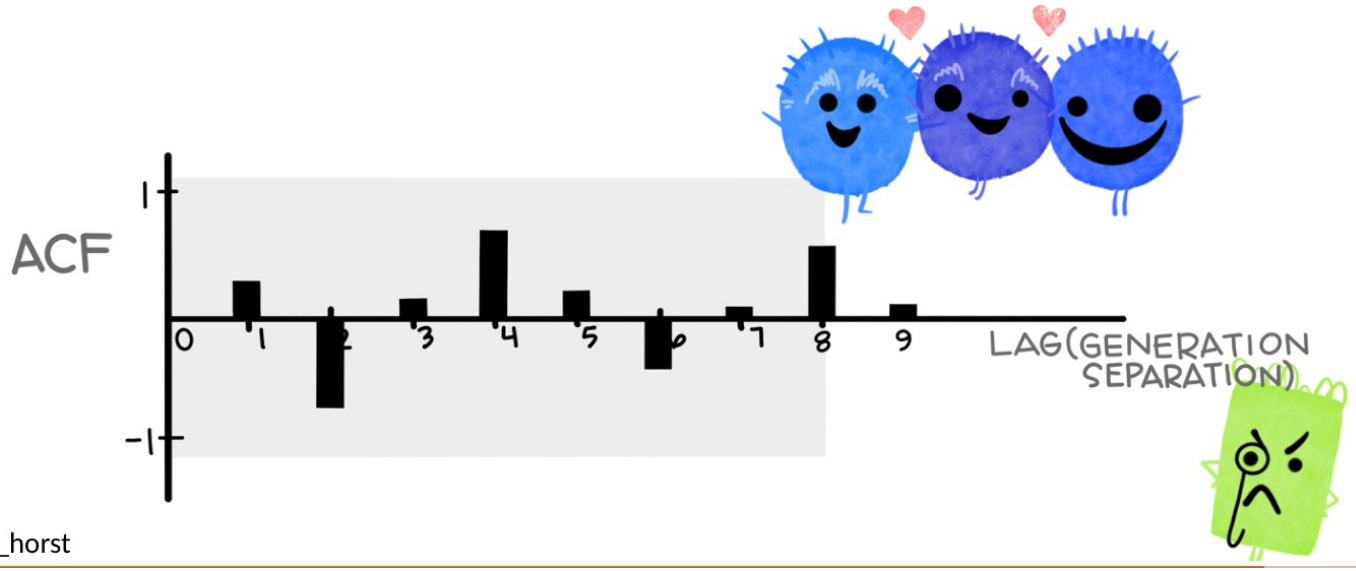


Artwork by @allison_horst

Autocorrelation functions

in Summary:

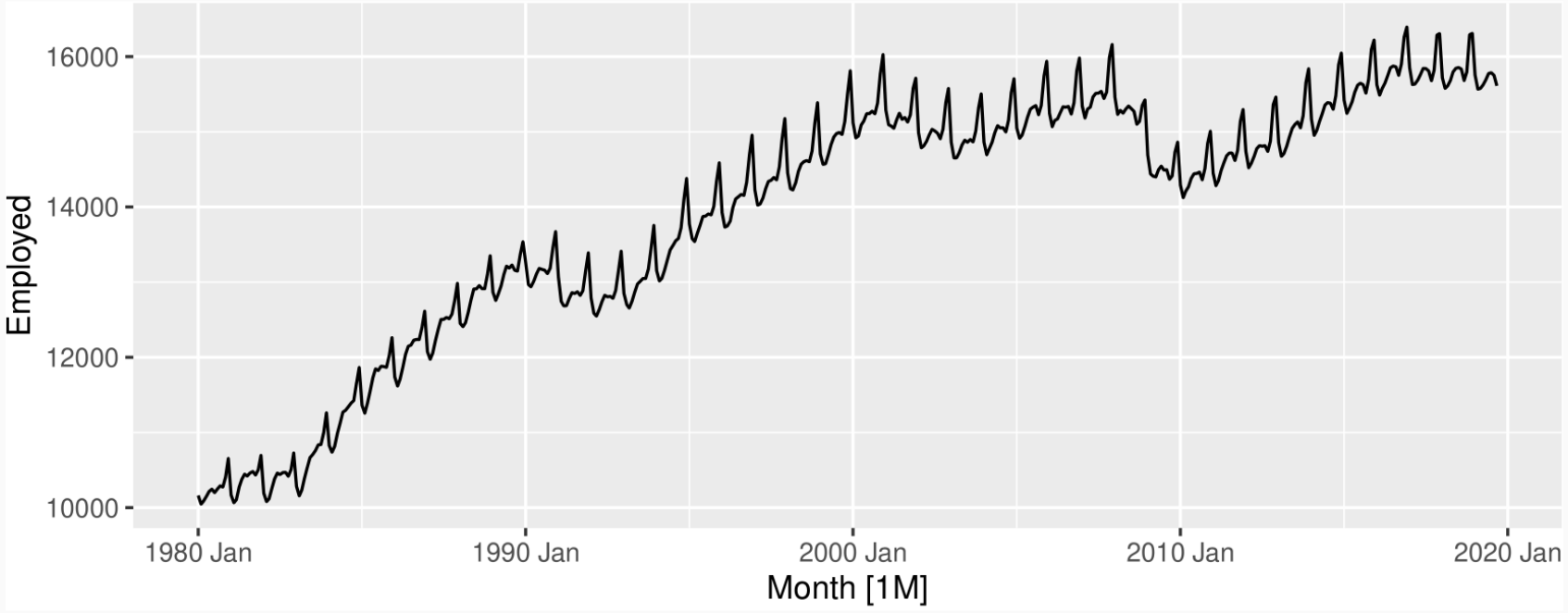
The autocorrelation function (ACF) tells us the correlation between observations and those that came before them, separated by different lags (here, monster generations)!



Artwork by @allison_horst

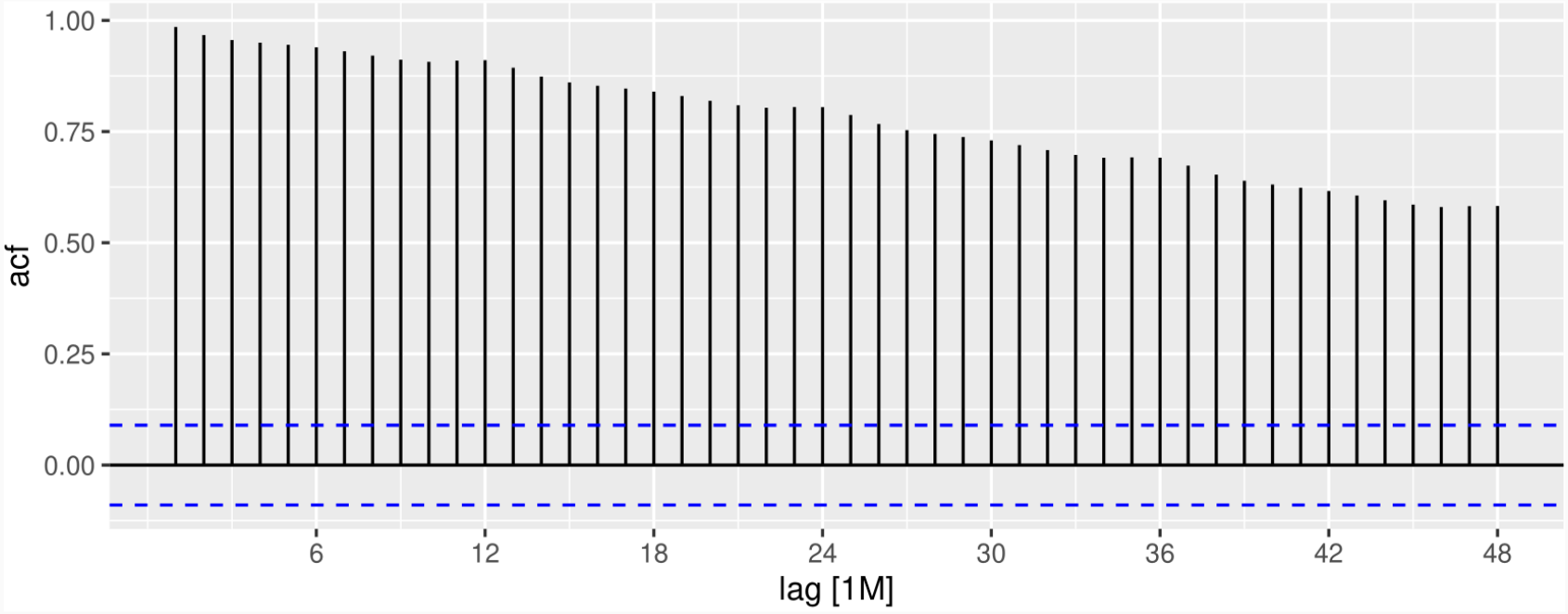
US retail trade employment

```
retail <- us_employment %>%
  filter(Title == "Retail Trade", year(Month) >= 1980)
retail %>% autoplot(Employed)
```



US retail trade employment

```
retail %>%
  ACF(Employed, lag_max = 48) %>%
  autoplot()
```



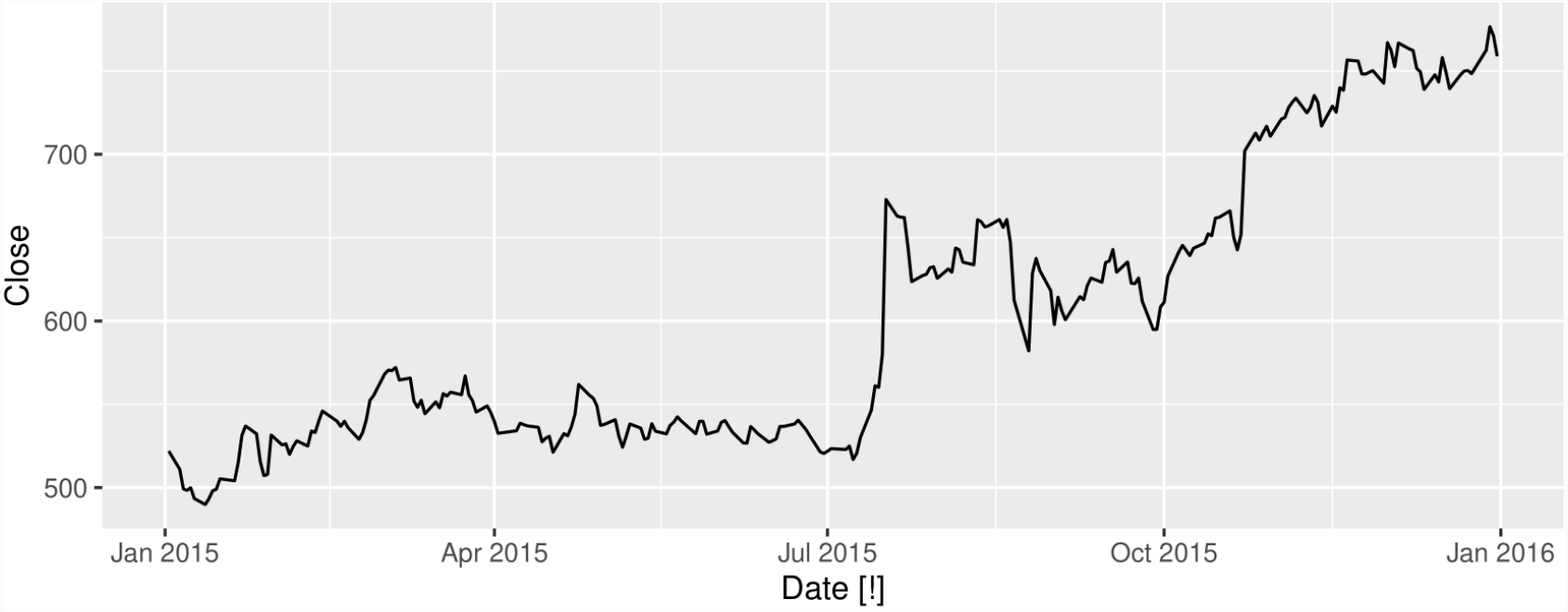
Google stock price

```
google_2015 <- gafa_stock %>%
  filter(Symbol == "GOOG", year(Date) == 2015) %>%
  select(Date, Close)
google_2015
```

```
## # A tsibble: 252 x 2 [!]
##    Date       Close
##    <date>     <dbl>
##  1 2015-01-02  522.
##  2 2015-01-05  511.
##  3 2015-01-06  499.
##  4 2015-01-07  498.
##  5 2015-01-08  500.
##  6 2015-01-09  493.
##  7 2015-01-12  490.
##  8 2015-01-13  493.
##  9 2015-01-14  498.
## 10 2015-01-15  499.
```

Google stock price

```
google_2015 %>% autoplot(Close)
```



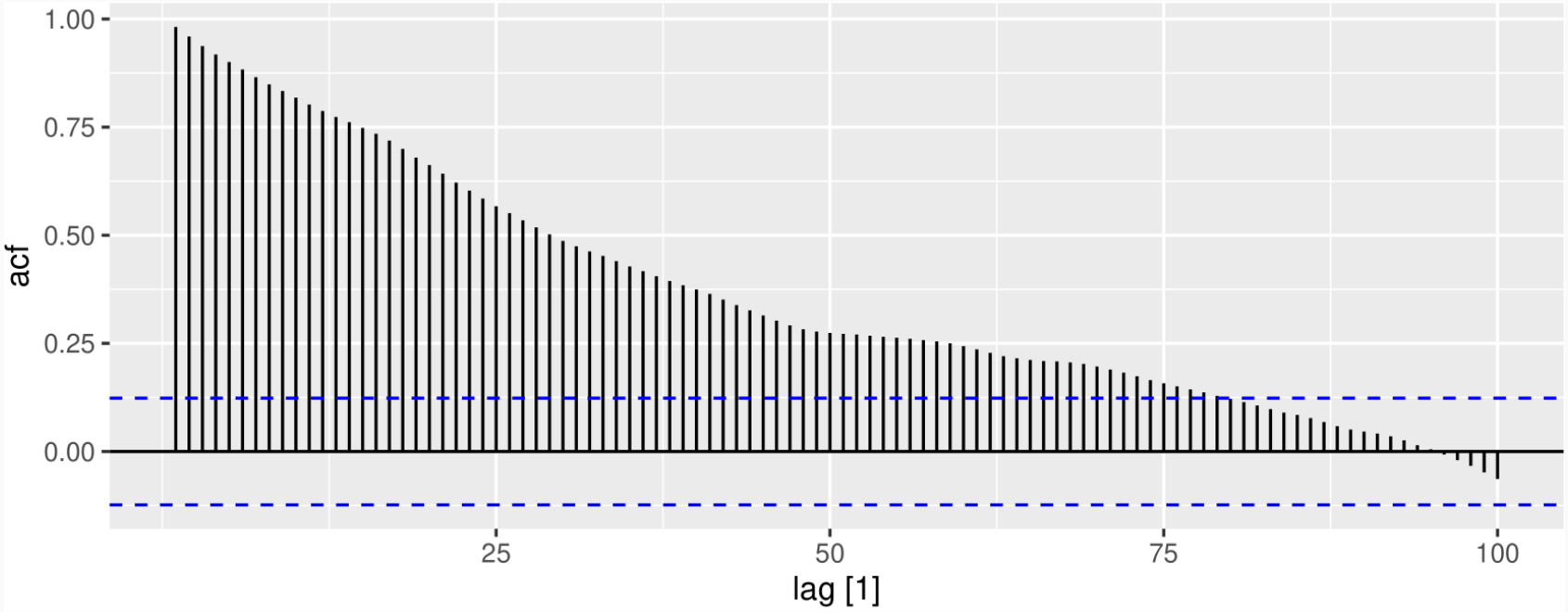
Google stock price

```
google_2015 %>%
  ACF(Close, lag_max=100)
```

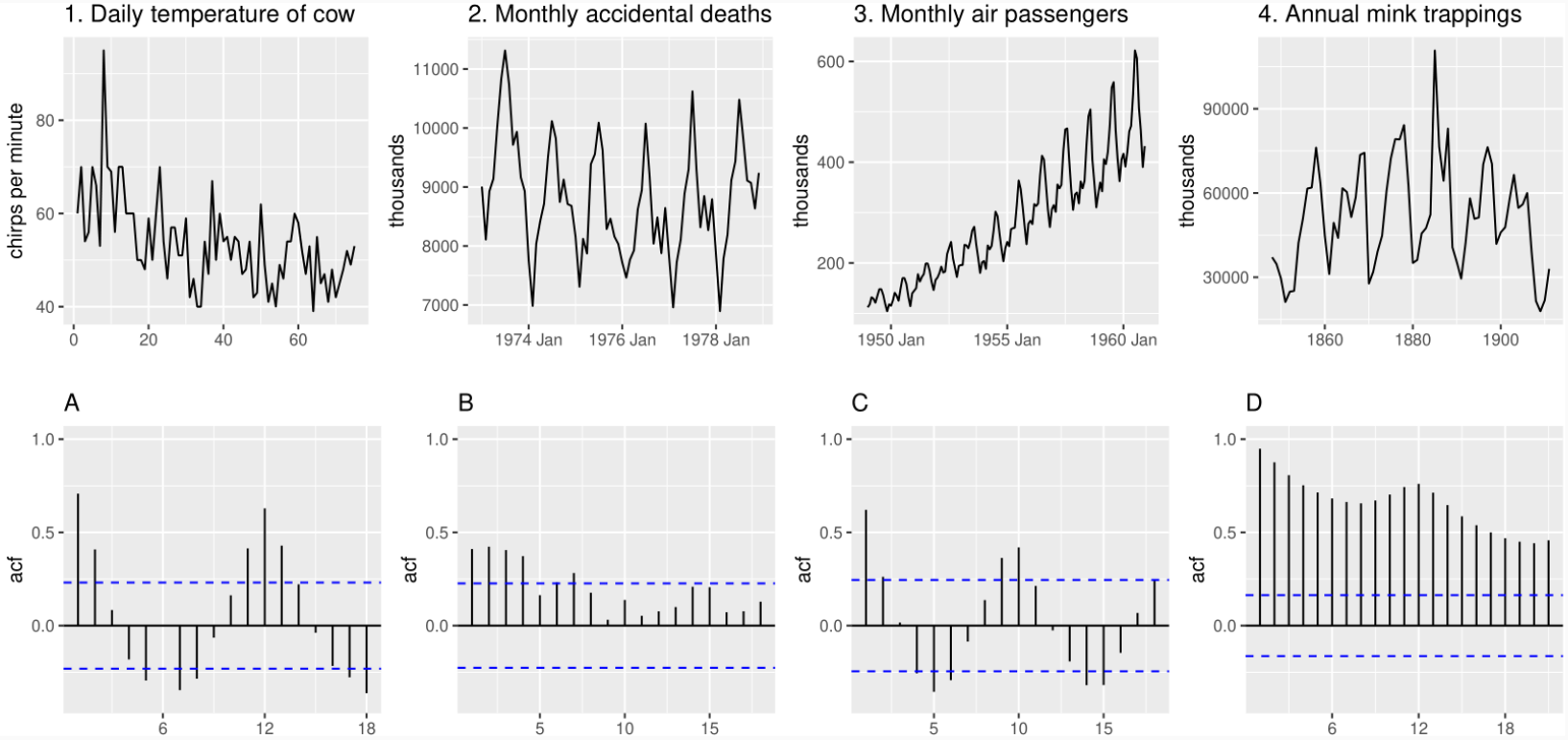
```
## # A tsibble: 100 x 2 [1]
##      lag   acf
##    <lag> <dbl>
##  1     1 0.982
##  2     2 0.959
##  3     3 0.937
##  4     4 0.918
##  5     5 0.901
##  6     6 0.883
##  7     7 0.865
##  8     8 0.849
##  9     9 0.834
## 10    10 0.818
## # ... with 90 more rows
```

Google stock price

```
google_2015 %>%
  ACF(Close, lag_max = 100) %>%
  autoplot()
```



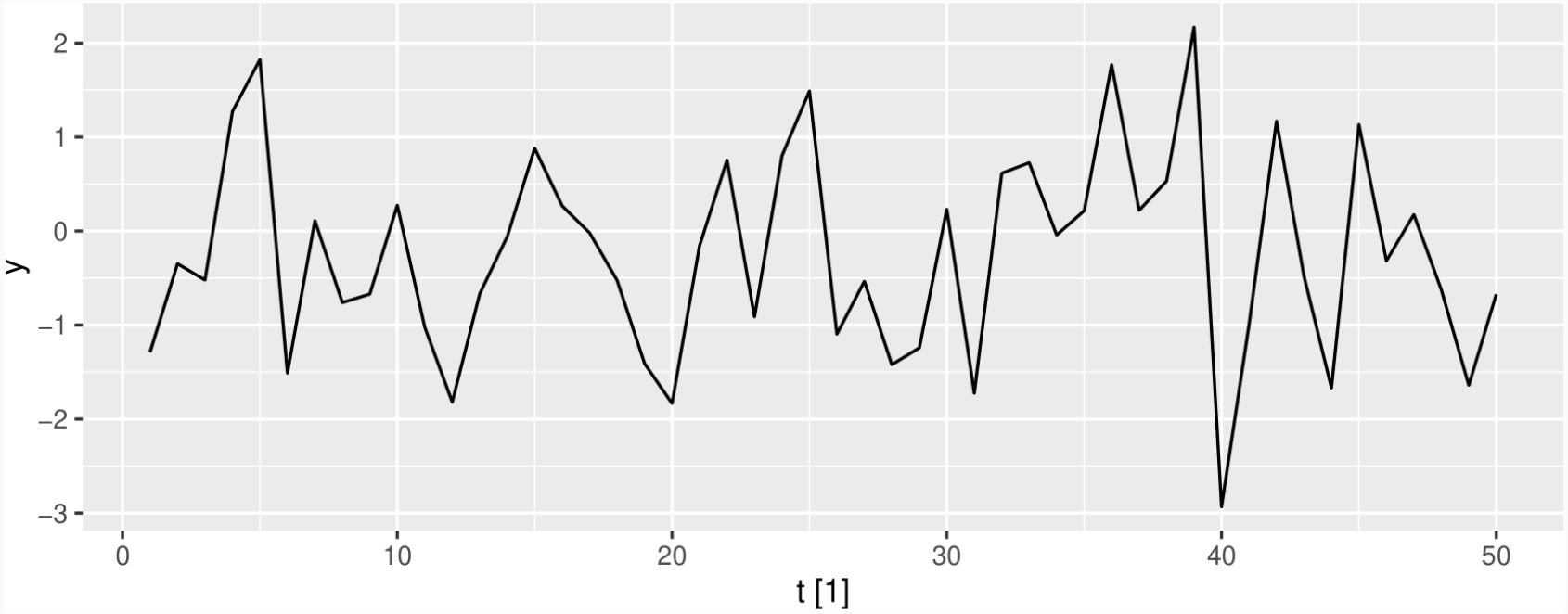
Which is which?



Outline

Example: White noise

```
set.seed(30)
wn <- tsibble(t = 1:50, y = rnorm(50), index = t)
wn %>% autoplot(y)
```



Example: White noise

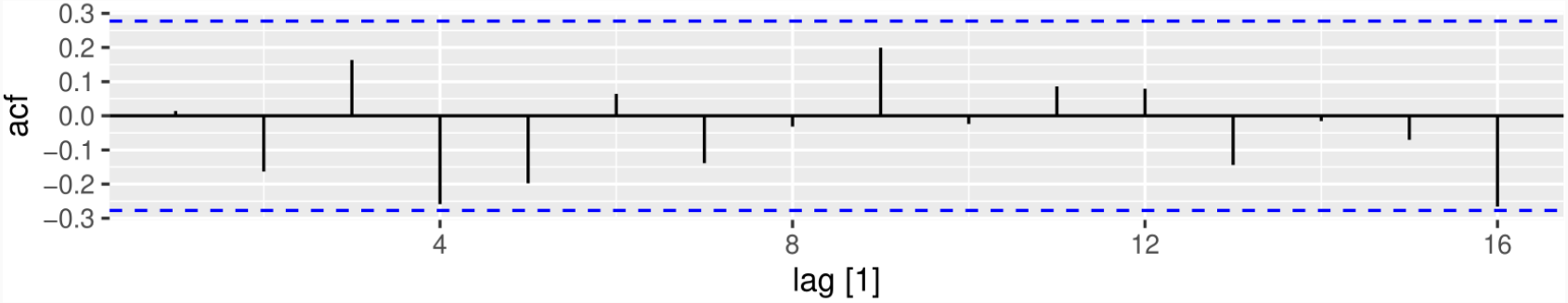
```
set.seed(30)
wn <- tsibble(t = 1:50, y = rnorm(50), index = t)
wn %>% autoplot(y)
```

White noise data is uncorrelated across time with zero mean and constant variance.
(Technically, we require independence as well.)

Example: White noise

```
wn %>% ACF(y)
```

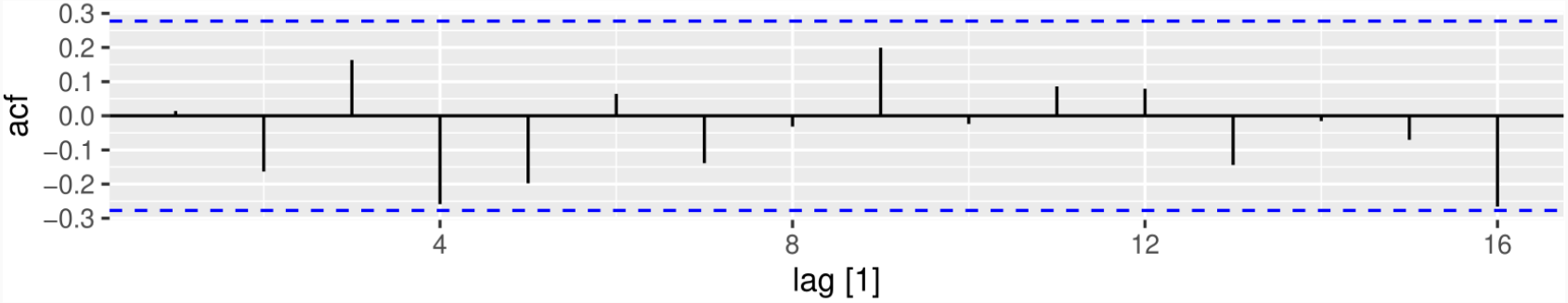
r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}
0.014	-0.163	0.163	-0.259	-0.198	0.064	-0.139	-0.032	0.199	-0.024



Example: White noise

```
wn %>% ACF(y)
```

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}
0.014	-0.163	0.163	-0.259	-0.198	0.064	-0.139	-0.032	0.199	-0.024



- Sample autocorrelations for white noise series.
- Expect each autocorrelation to be close to zero.
- Blue lines show 95% critical values.

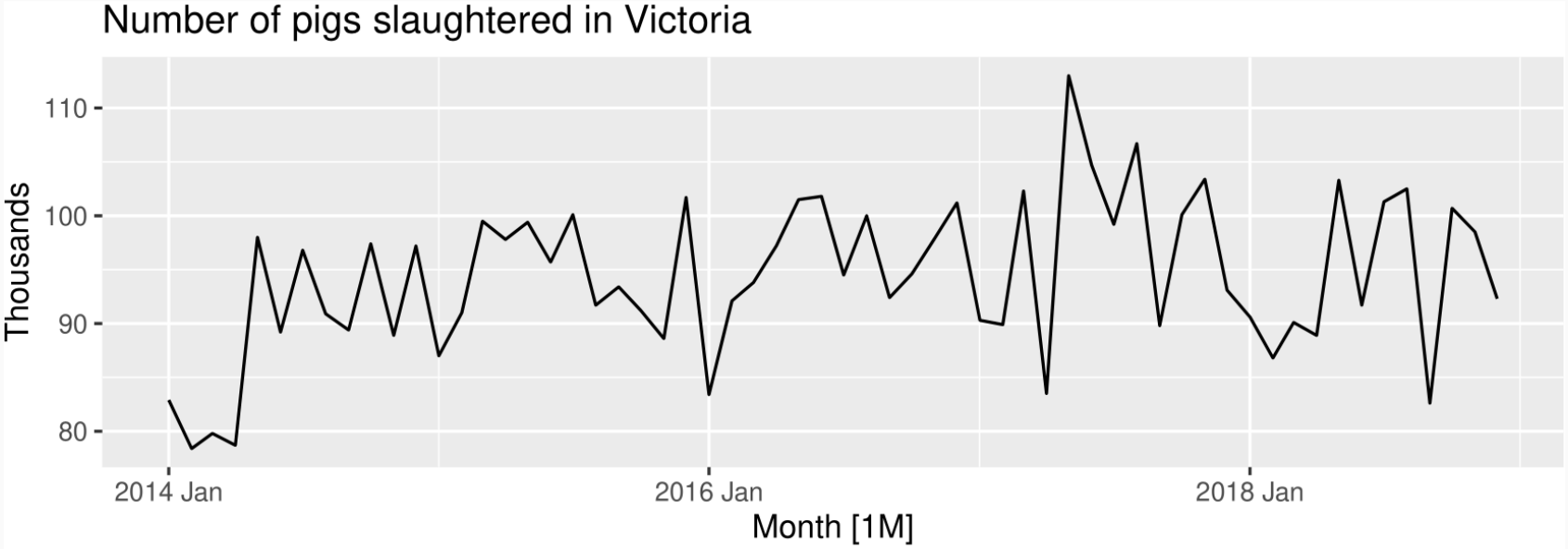
Sampling distribution of autocorrelations

Sampling distribution of r_k for white noise data is asymptotically $\text{N}(0,1/T)$.

- 95% of all r_k for white noise must lie within $\pm 1.96/\sqrt{T}$.
- If this is not the case, the series is probably not WN.
- Common to plot lines at $\pm 1.96/\sqrt{T}$ when plotting ACF. These are the **critical values**.

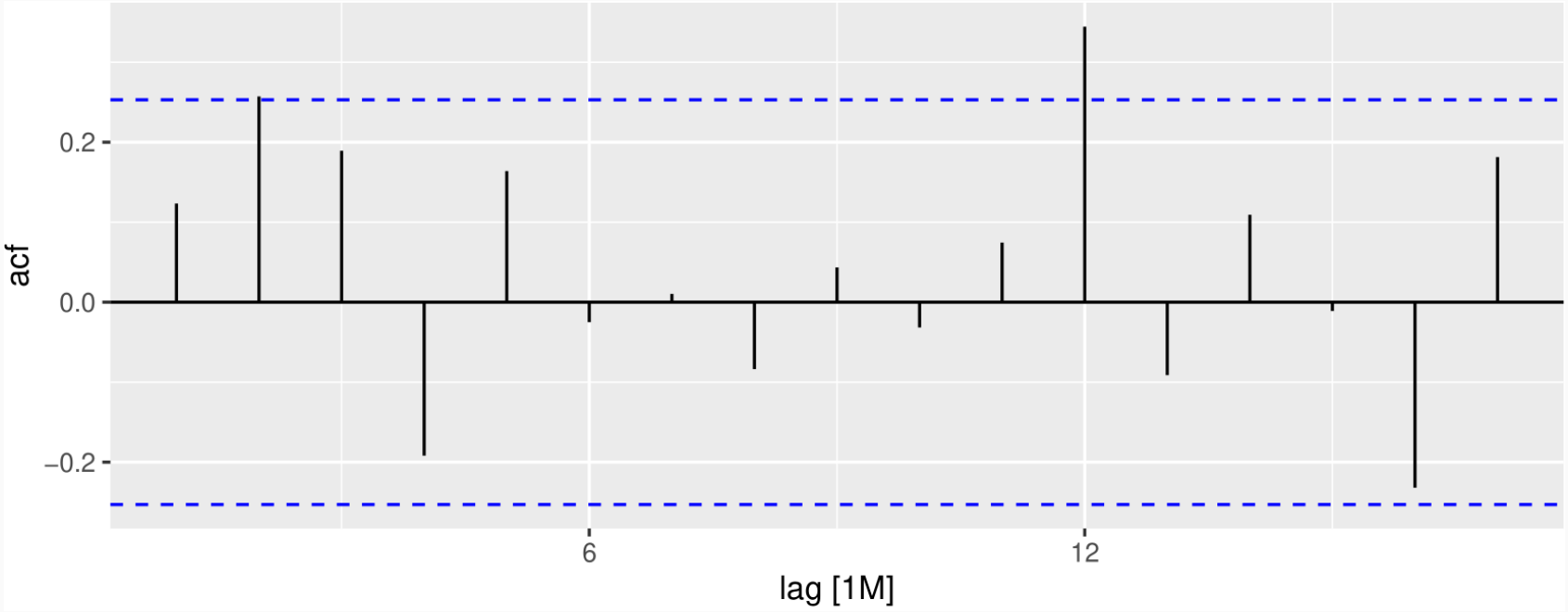
Example: Pigs slaughtered

```
pigs <- aus_livestock %>%
  filter(State == "Victoria", Animal == "Pigs", year(Month) >= 2014)
pigs %>% autoplot(Count/1e3) +
  labs(y = "Thousands", title = "Number of pigs slaughtered in Victoria")
```



Example: Pigs slaughtered

```
pigs %>% ACF(Count) %>% autoplot()
```



Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows significant autocorrelation for lag 2 and 12.
- Indicate some slight seasonality.

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 2014 through December 2018 (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows significant autocorrelation for lag 2 and 12.
- Indicate some slight seasonality.

These show the series is **not a white noise series**.

Your turn

You can compute the daily changes in the Google stock price in 2018 using

```
dgoog <- gafa_stock %>%
  filter(Symbol == "GOOG", year(Date) >= 2018) %>%
  mutate(diff = difference(Close))
```

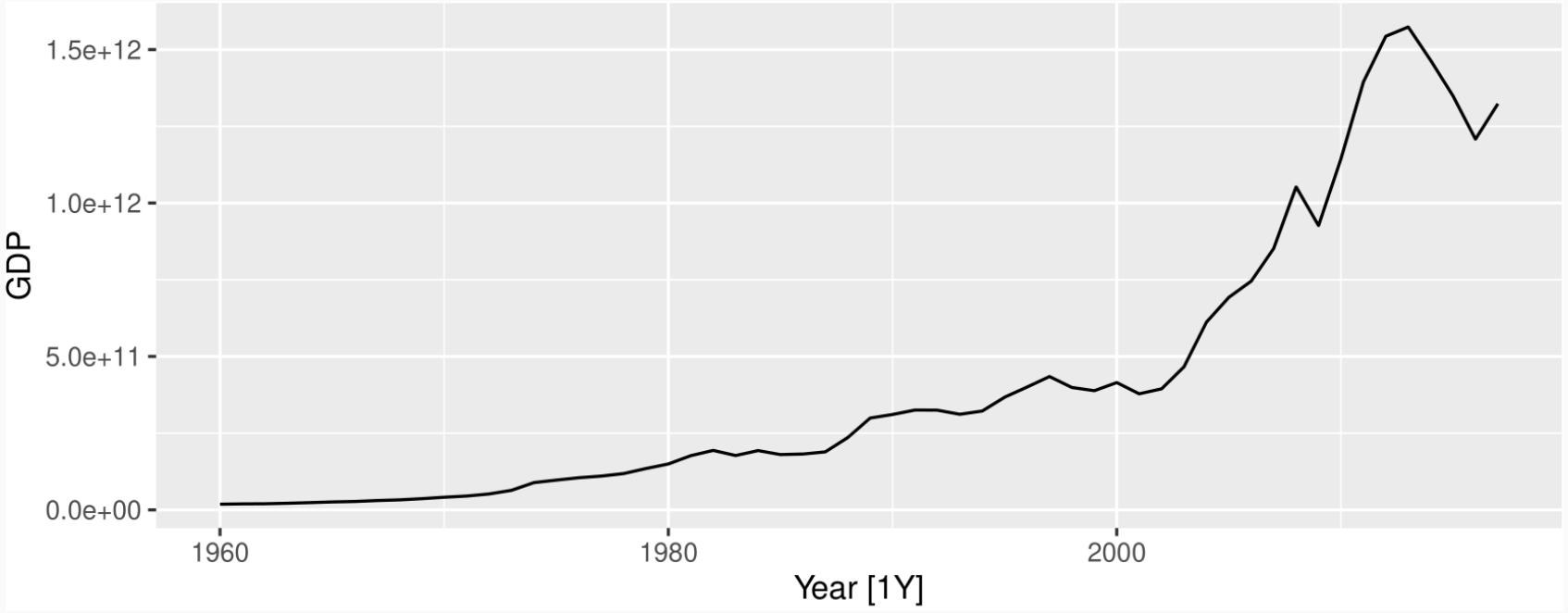
Does `diff` look like white noise?

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

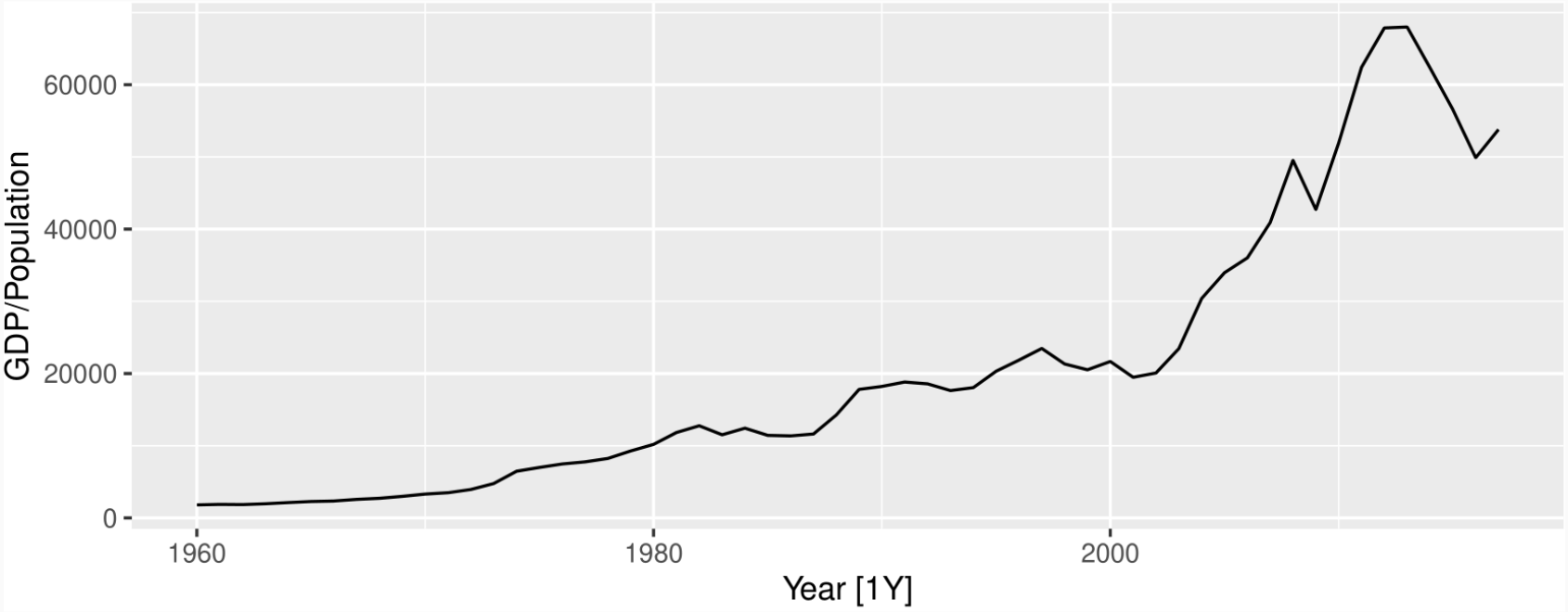
Per capita adjustments

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



Per capita adjustments

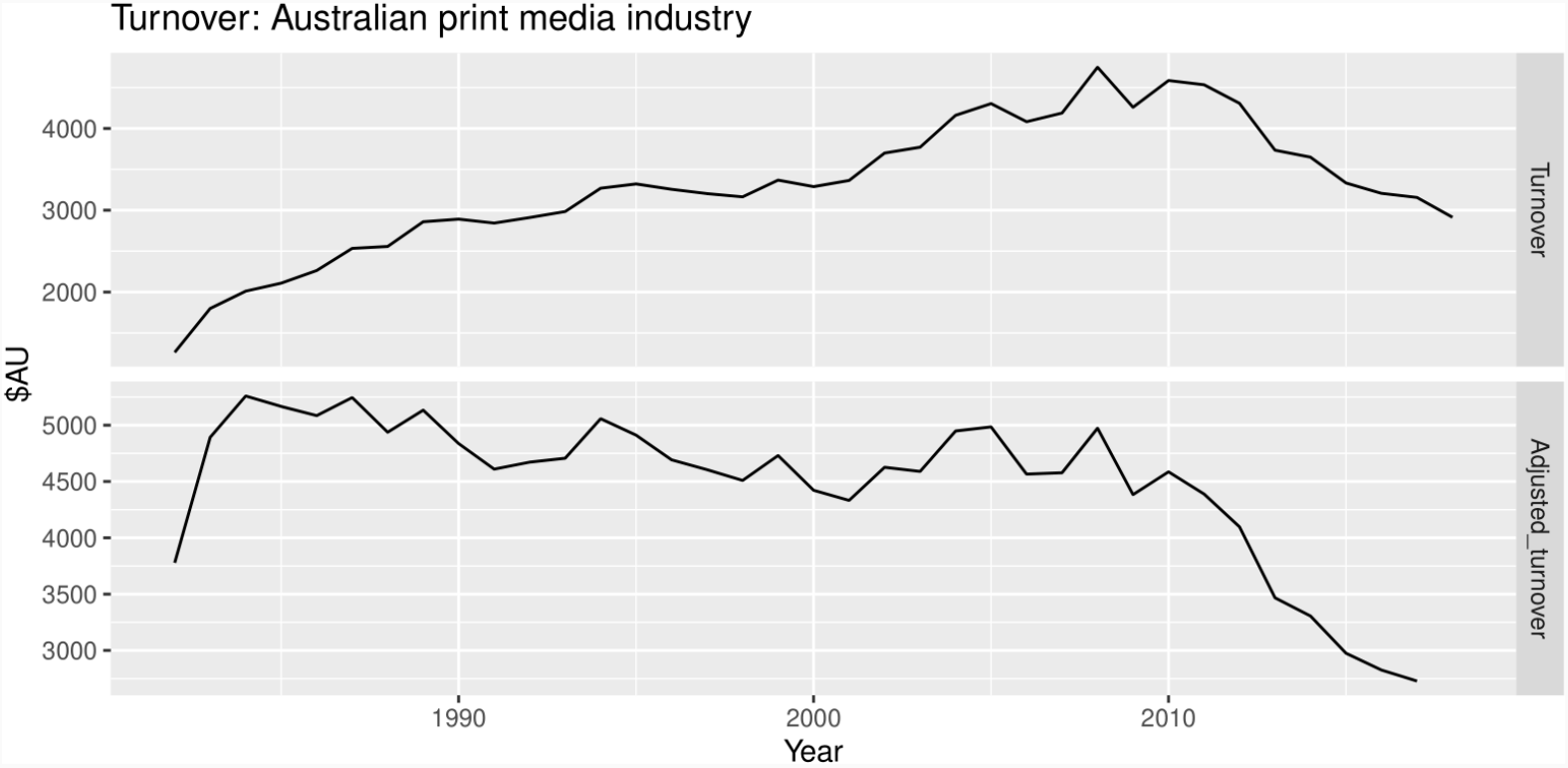
```
global_economy %>%
  filter(Country == "Australia") %>%
  autoplot(GDP / Population)
```



Inflation adjustments

```
print_retail <- aus_retail %>%
  filter(Industry == "Newspaper and book retailing") %>%
  group_by(Industry) %>%
  index_by(Year = year(Month)) %>%
  summarise(Turnover = sum(Turnover))
aus_economy <- global_economy %>%
  filter(Code == "AUS")
print_retail %>%
  left_join(aus_economy, by = "Year") %>%
  mutate(Adjusted_turnover = Turnover / CPI * 100) %>%
  pivot_longer(c(Turnover, Adjusted_turnover), values_to = "Turnover") %>%
  mutate(name = factor(name, levels=c("Turnover","Adjusted_turnover"))) %>%
  ggplot(aes(x = Year, y = Turnover)) +
  geom_line() +
  facet_grid(name ~ ., scales = "free_y") +
  labs(title = "Turnover: Australian print media industry", y = "$AU")
```

Inflation adjustments



Mathematical transformations

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

Denote original observations as $y_1, \ldots, y_n$ and transformed observations as $w_1, \ldots, w_n$.

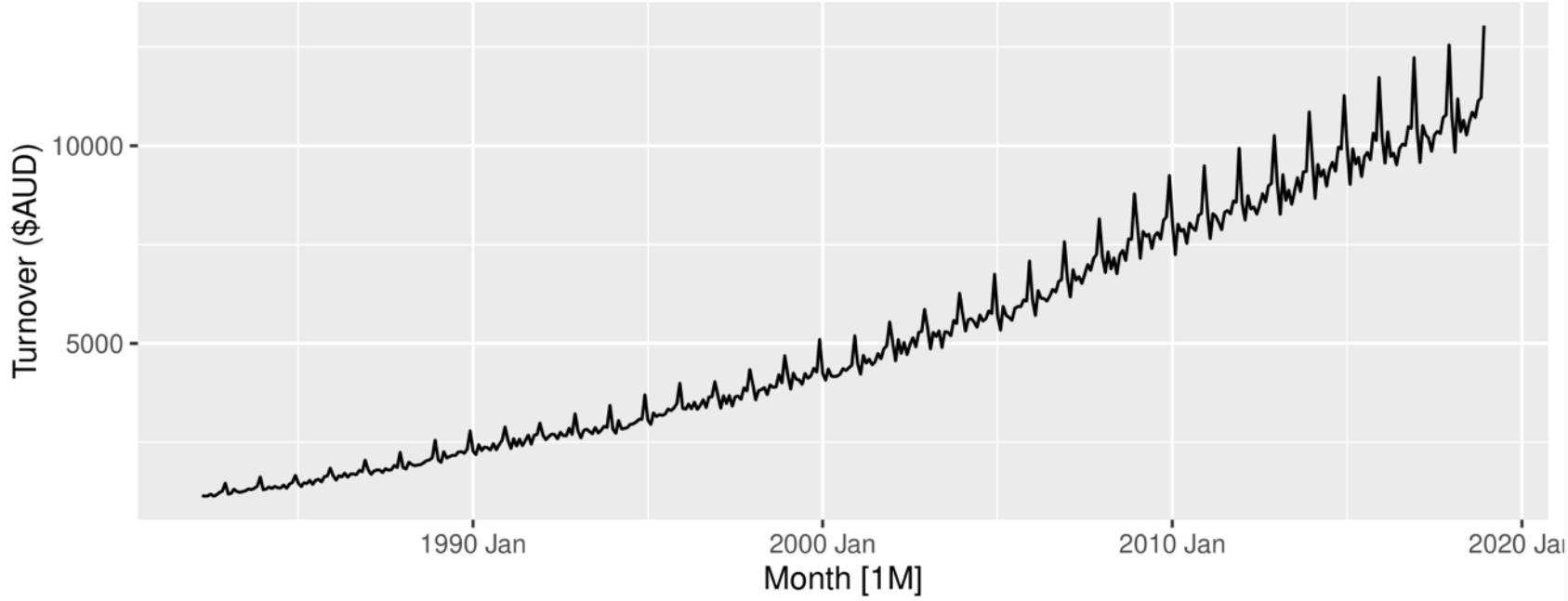
Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	↓
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**.

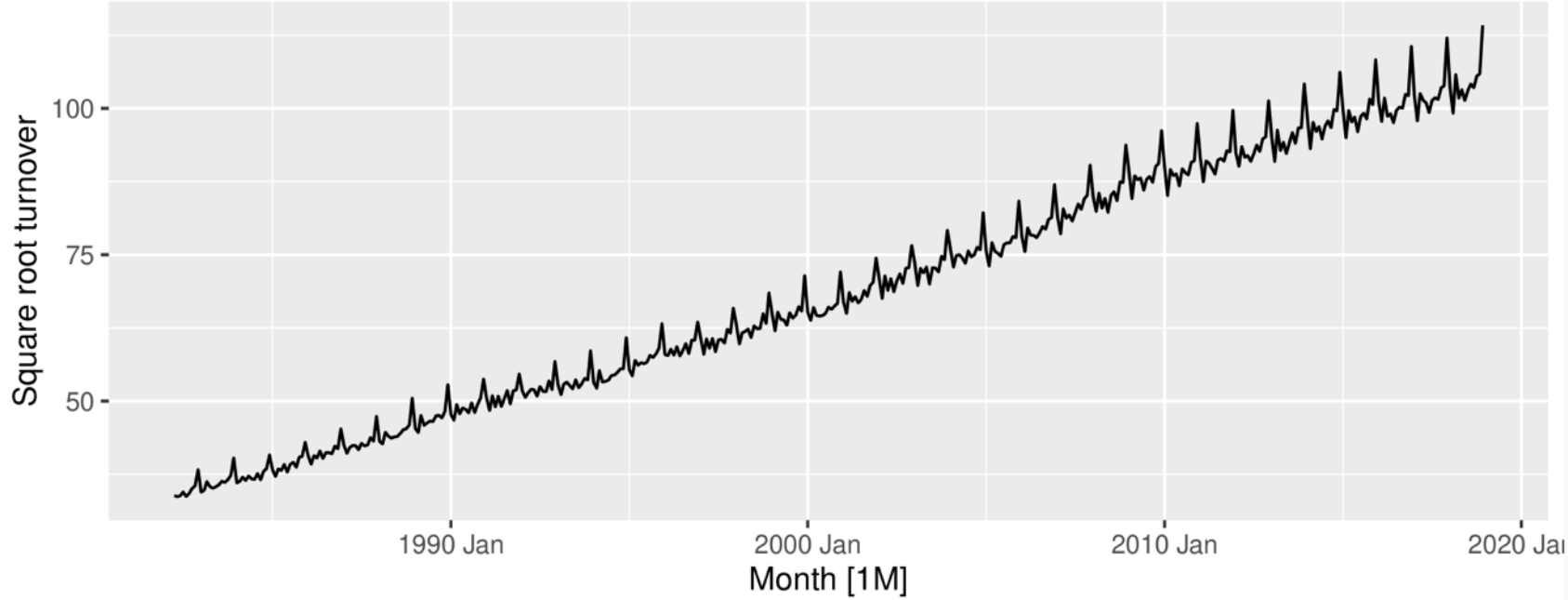
Mathematical transformations

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



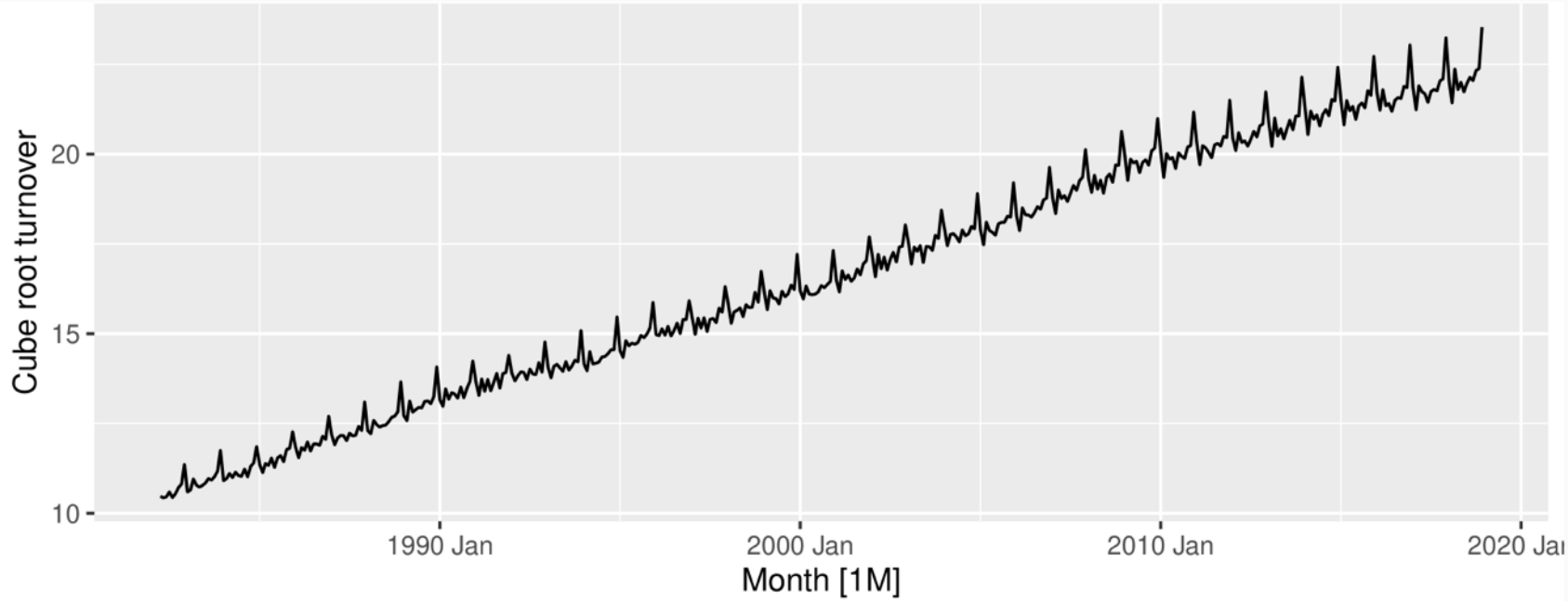
Mathematical transformations

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



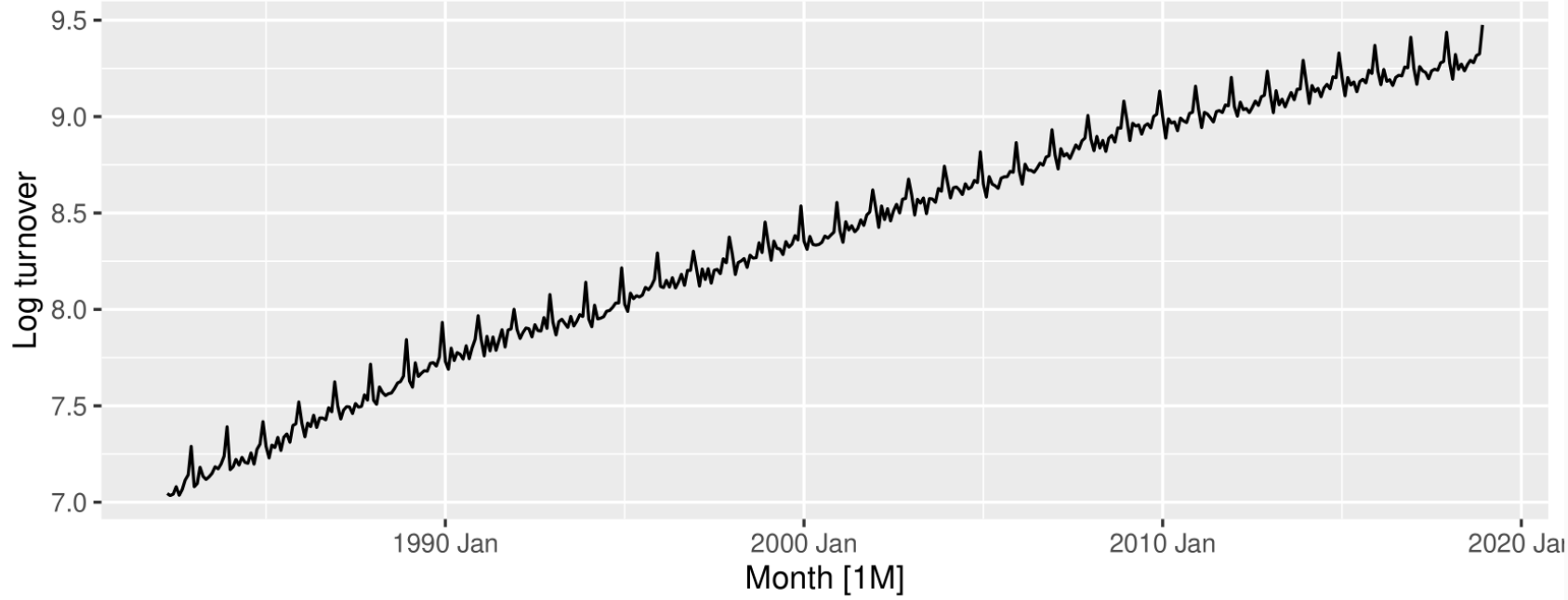
Mathematical transformations

```
food %>% autoplot(Turnoverˆ(1/3)) +
  labs(y = "Cube root turnover")
```



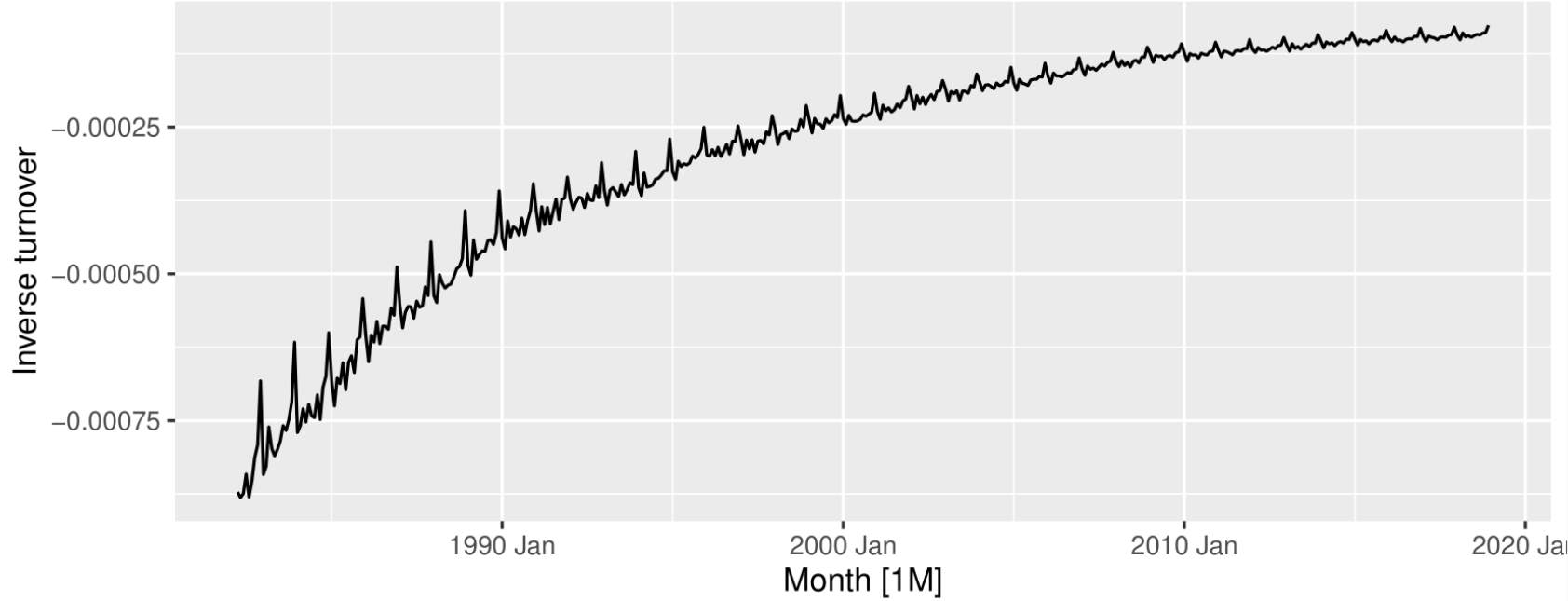
Mathematical transformations

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



Mathematical transformations

```
food %>% autoplot(-1/Turnover) +
  labs(y = "Inverse turnover")
```



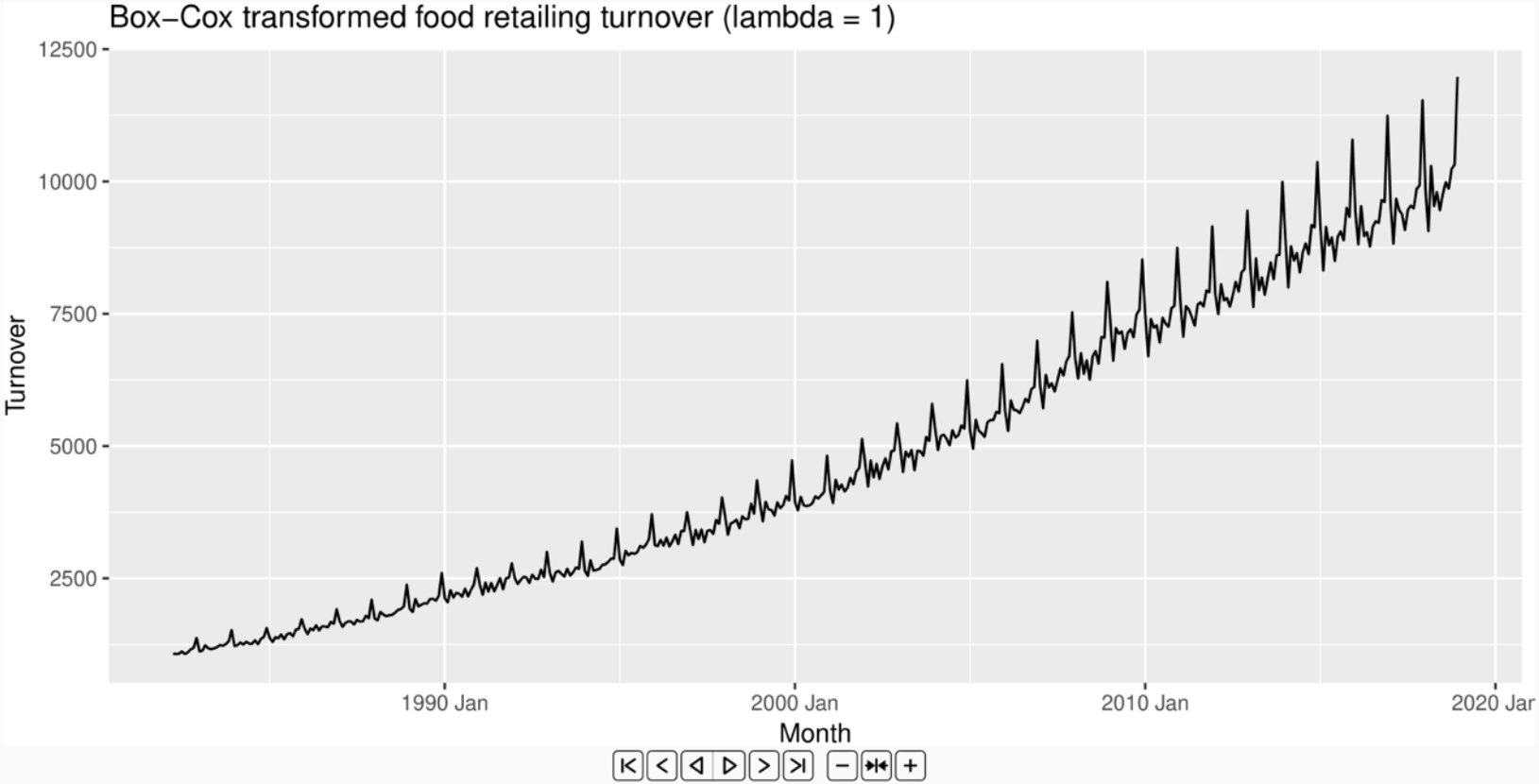
Box-Cox transformations

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

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

- 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

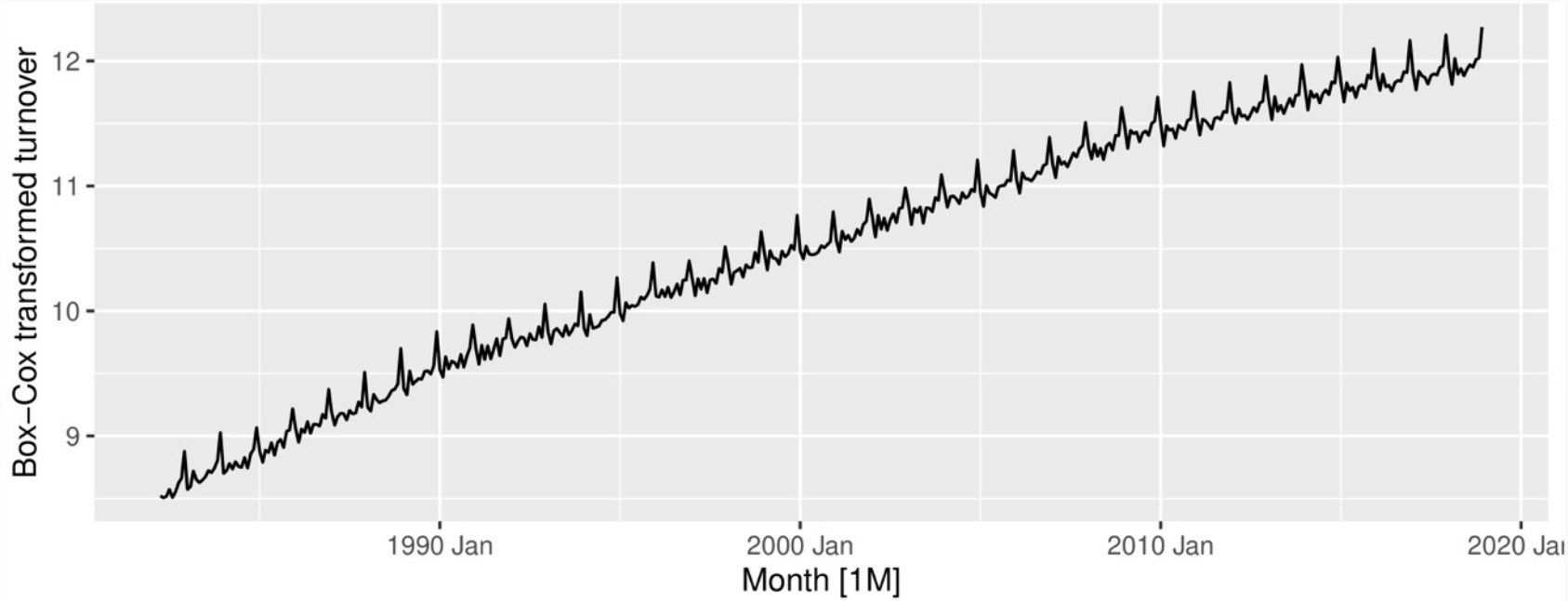
```
food %>%
  features(Turnover, features = guerrero)
```

```
## # A tibble: 1 x 1
##   lambda_guerrero
##             <dbl>
## 1          0.0524
```

- 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

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

Time series patterns

Recall

Trend pattern exists when there is a long-term increase or decrease in the data.

Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).

Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Time series decomposition

$$y_t = f(S_t, T_t, R_t)$$

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

Time series decomposition

$$y_t = f(S_t, T_t, R_t)$$

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

Additive decomposition: $y_t = S_t + T_t + R_t$.

Multiplicative decomposition: $y_t = S_t \times T_t \times R_t$.

Time series decomposition

- Additive model appropriate if magnitude of seasonal fluctuations does not vary with level.
- If seasonal are proportional to level of series, then multiplicative model appropriate.
- Multiplicative decomposition more prevalent with economic series
- Alternative: use a Box-Cox transformation, and then use additive decomposition.
- Logs turn multiplicative relationship into an additive relationship:

$$y_t = S_t \times T_t \times R_t \quad \Rightarrow \quad \log y_t = \log S_t + \log T_t + \log R_t.$$

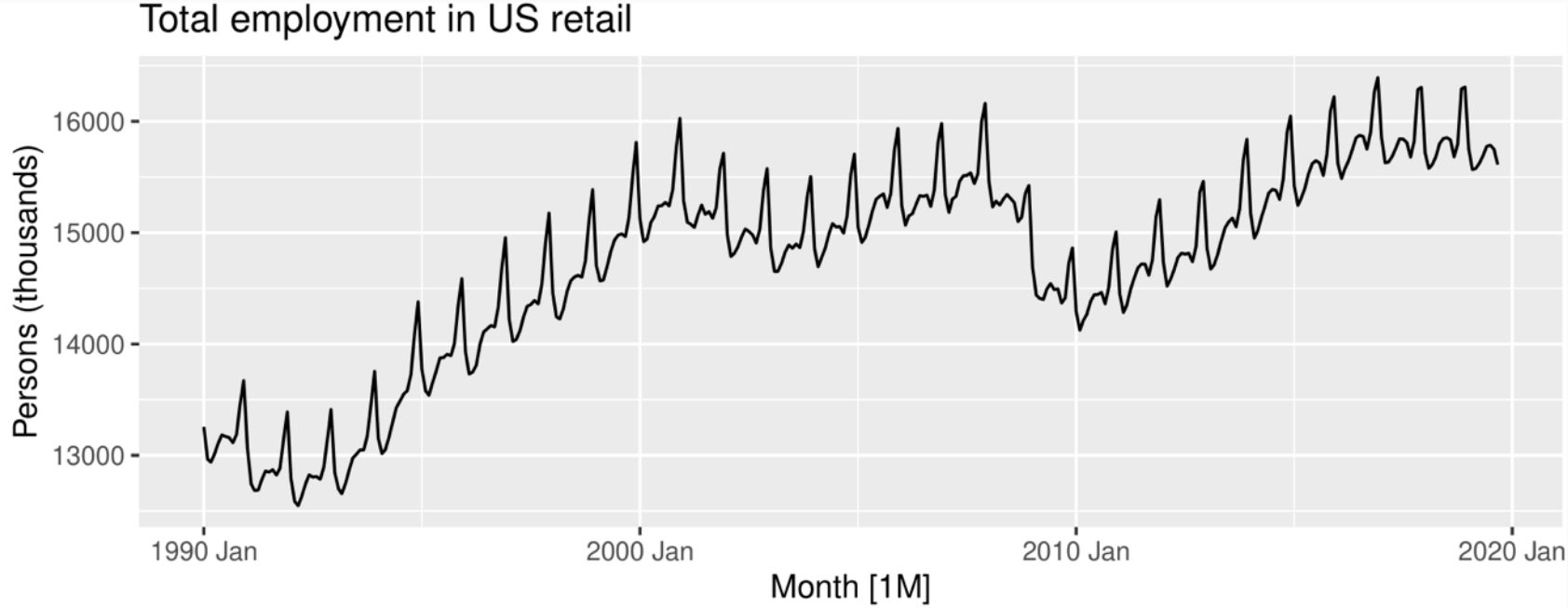
US Retail Employment

```
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
```

US Retail Employment

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



US Retail Employment

```
us_retail_employment %>%
  model(stl = STL(Employed))
```

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

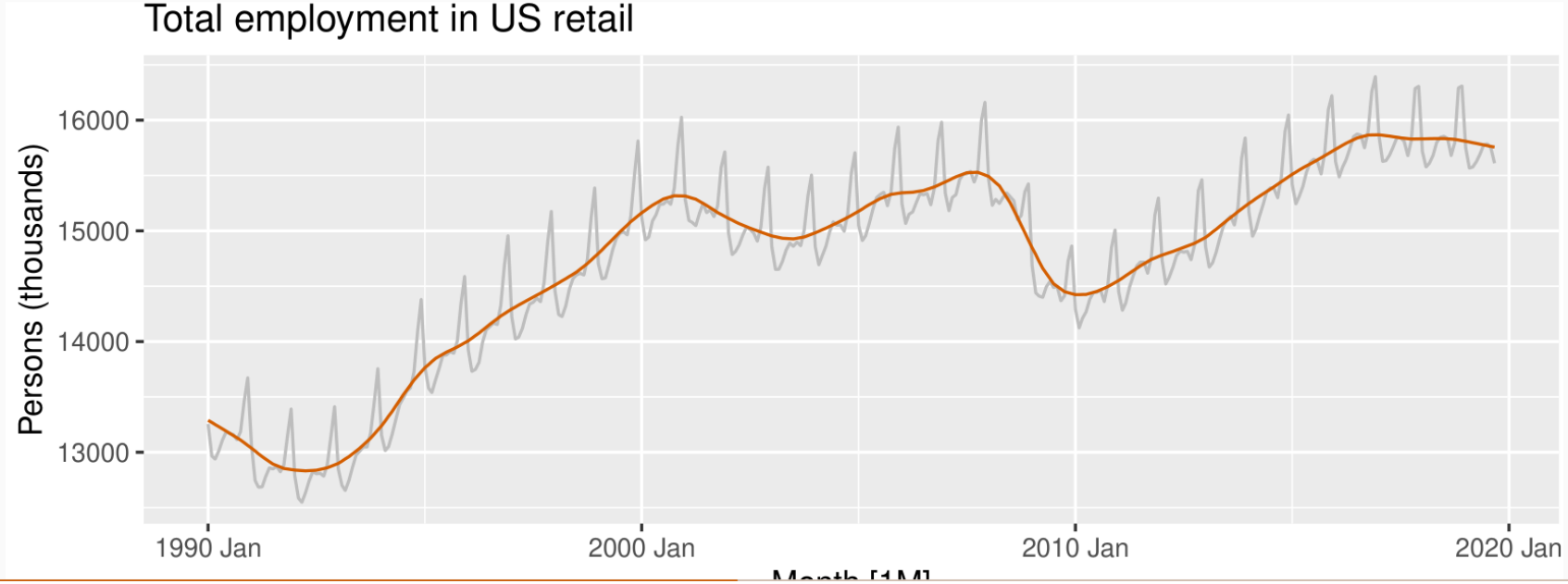
US Retail Employment

```
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_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.       -0.53    17.8           13160.
```

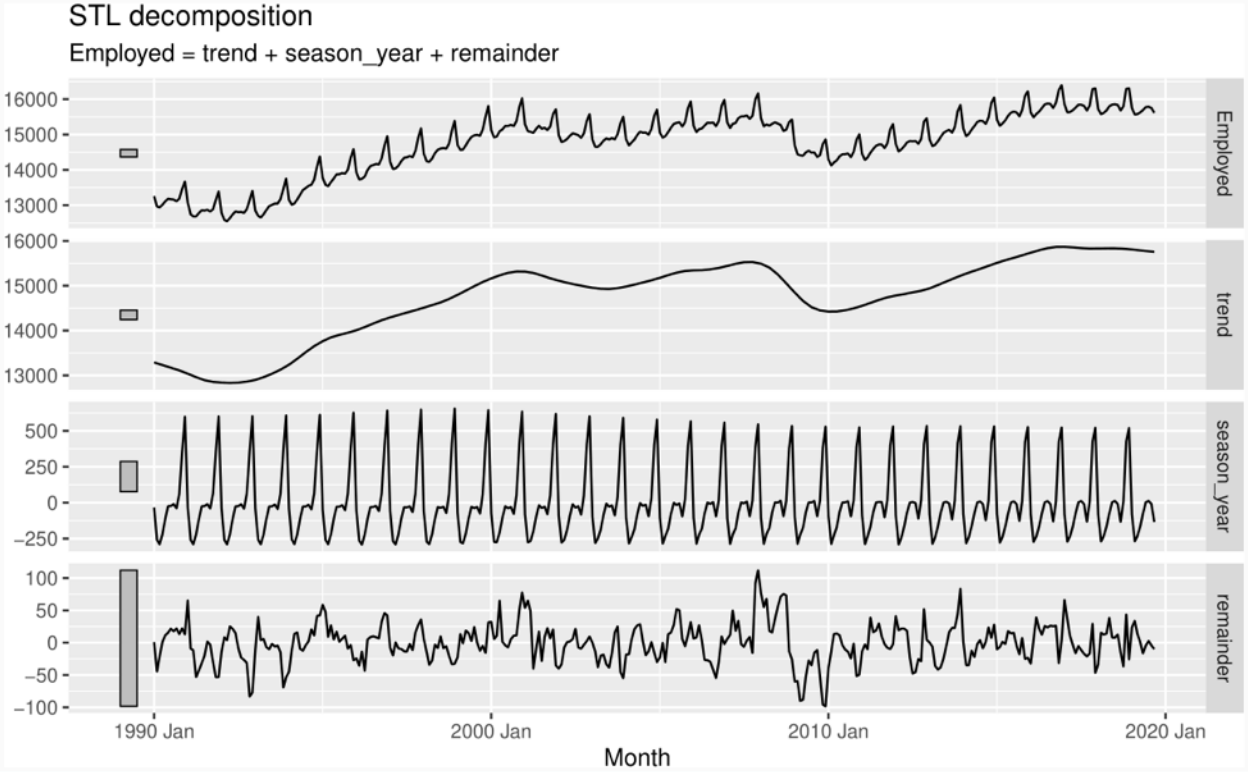
US Retail Employment

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



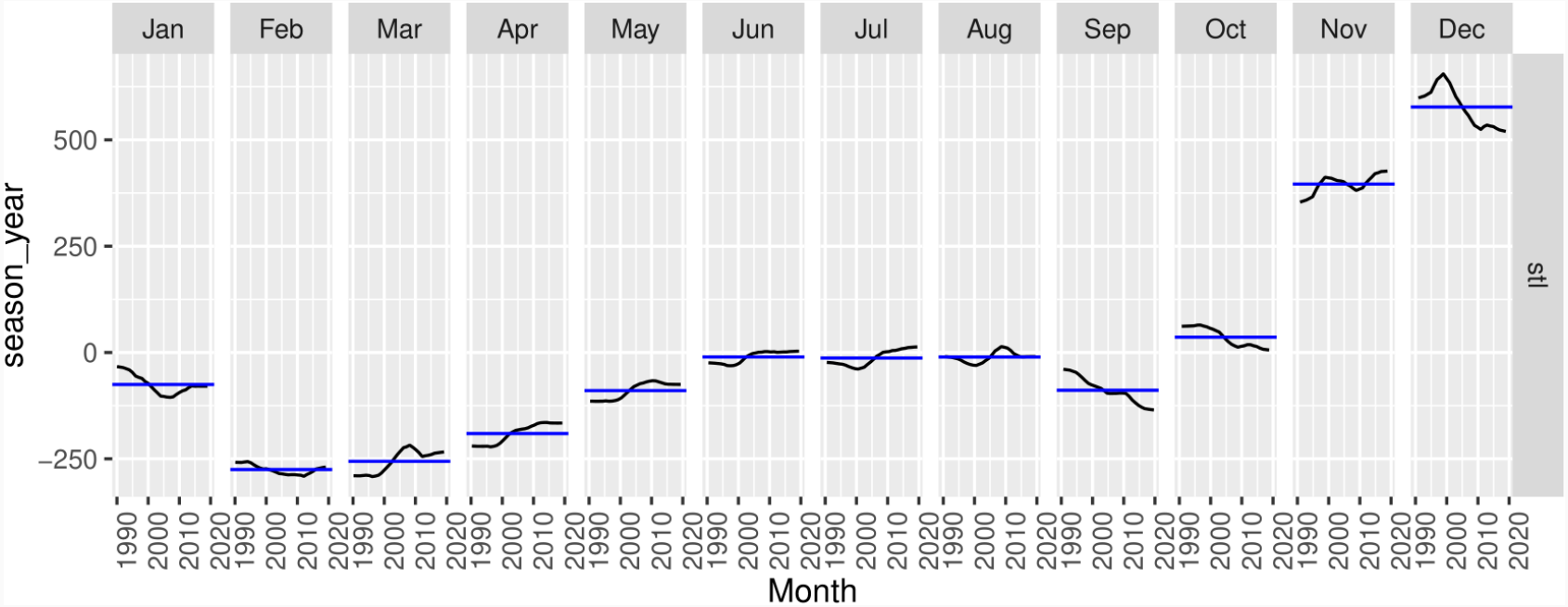
US Retail Employment

```
components(dcmp) %>% autoplot()
```



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

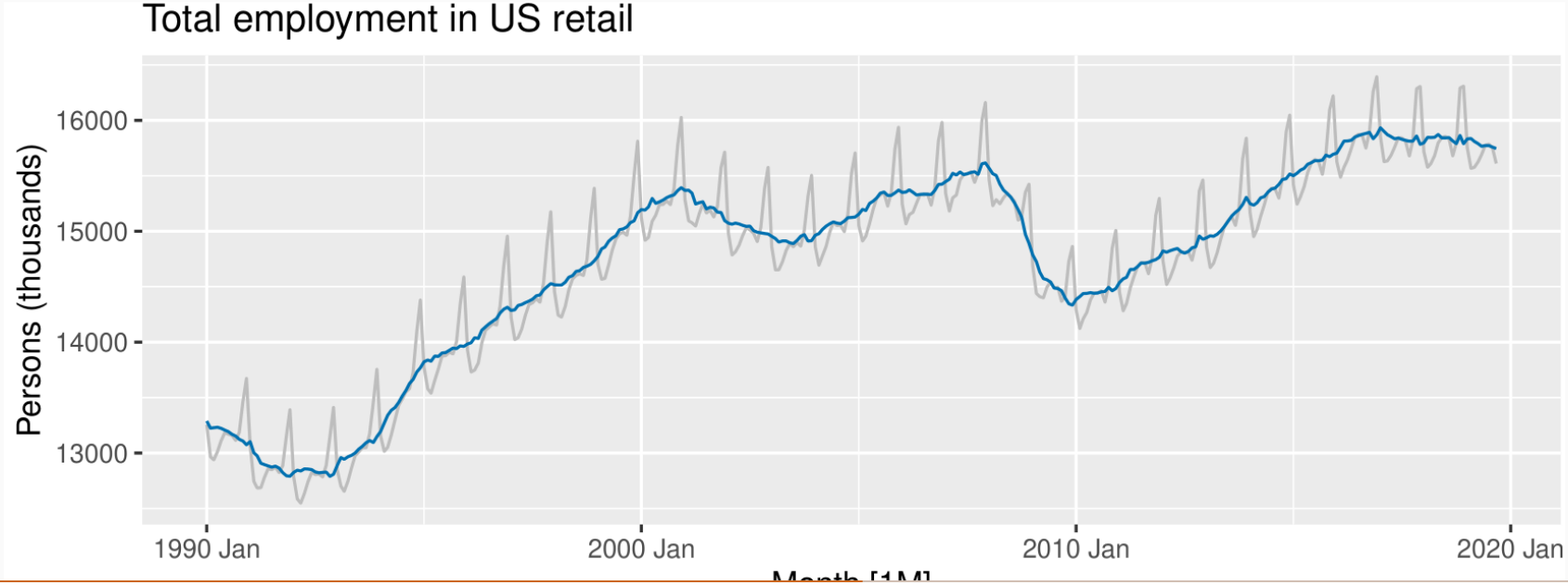
$$y_t - S_t = T_t + R_t$$

- 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, color='#0072B2') +
  labs(y="Persons (thousands)", title="Total employment in US retail")
```



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.

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

History of time series decomposition

- Classical method originated in 1920s.
- Census II method introduced in 1957. Basis for X-11 method and variants (including X-12-ARIMA, X-13-ARIMA)
- STL method introduced in 1983
- TRAMO/SEATS introduced in 1990s.

National Statistics Offices

- ABS uses X-12-ARIMA
- US Census Bureau uses X-13ARIMA-SEATS
- Statistics Canada uses X-12-ARIMA
- ONS (UK) uses X-12-ARIMA
- EuroStat use X-13ARIMA-SEATS

X-11 decomposition

Advantages

- Relatively robust to outliers
- Completely automated choices for trend and seasonal changes
- Very widely tested on economic data over a long period of time.

Disadvantages

- No prediction/confidence intervals
- Ad hoc method with no underlying model
- Only developed for quarterly and monthly data

Extensions: X-12-ARIMA and X-13-ARIMA

- The X-11, X-12-ARIMA and X-13-ARIMA methods are based on Census II decomposition.
- These allow adjustments for trading days and other explanatory variables.
- Known outliers can be omitted.
- Level shifts and ramp effects can be modelled.
- Missing values estimated and replaced.
- Holiday factors (e.g., Easter, Labour Day) can be estimated.

X-13ARIMA-SEATS

Advantages

- Model-based
- Smooth trend estimate
- Allows estimates at end points
- Allows changing seasonality
- Developed for economic data

Disadvantages

- Only developed for quarterly and monthly data

Outline

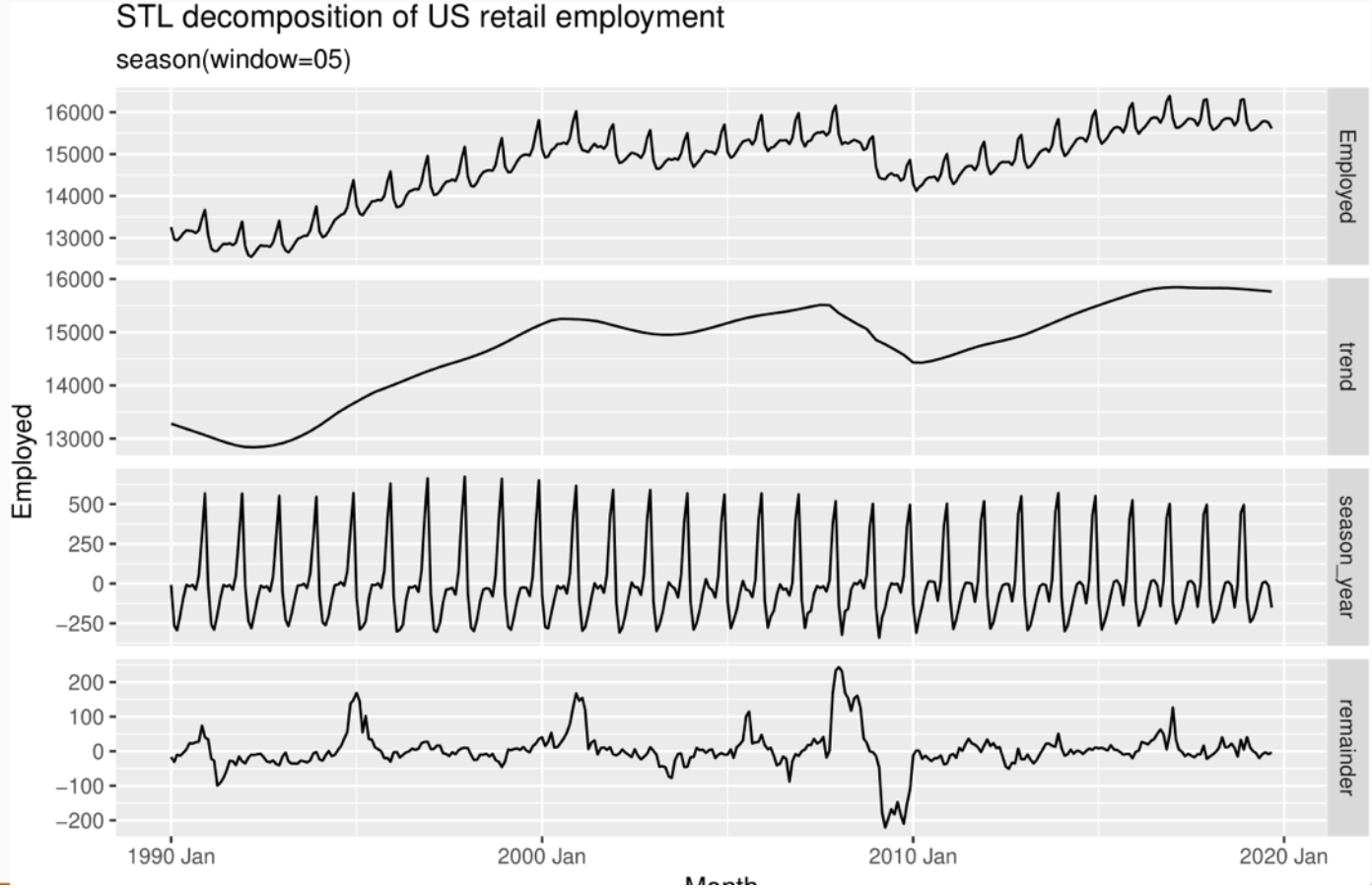
STL decomposition

- 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

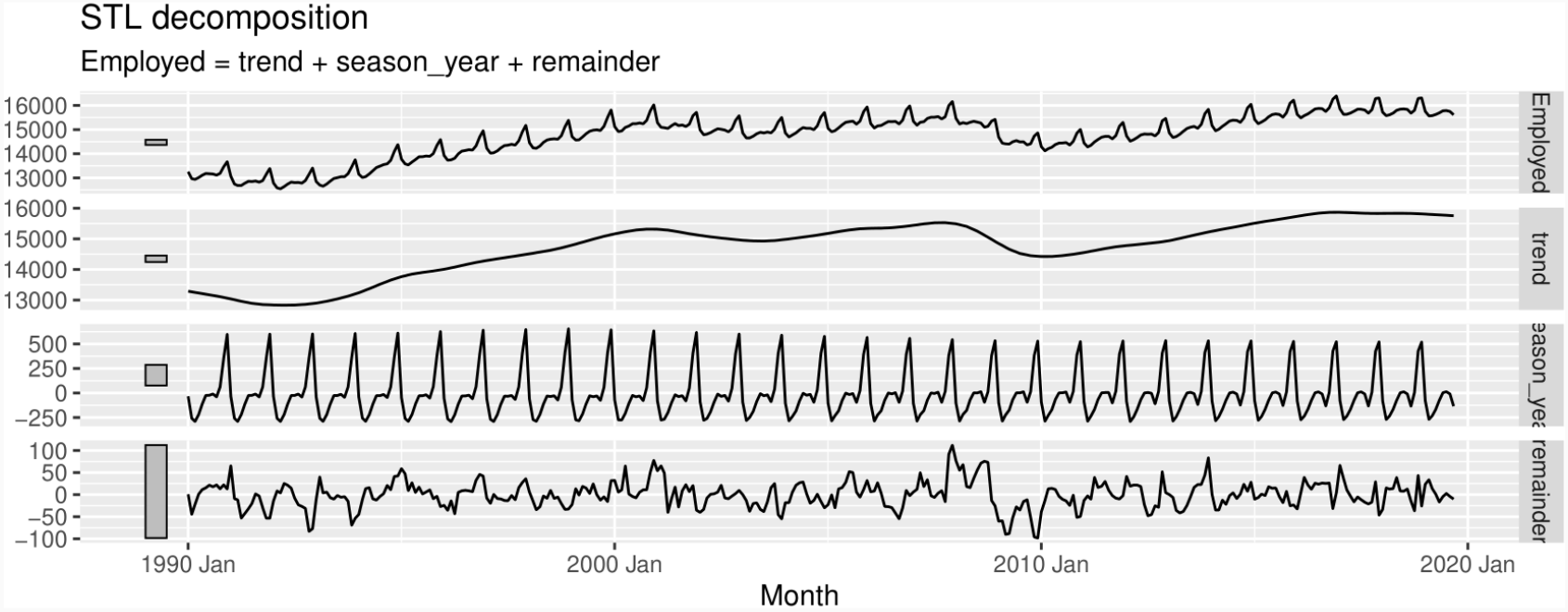
```
us_retail_employment %>%
  model(STL(Employed ~ season(window=5))) %>%
  components()

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

- `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)))`

Outline

1. Transformations and adjustments
2. Time series components
3. History of time series decomposition
4. STL decomposition
5. When things go wrong

The ABS stuff-up

NEWS

LOCATION: Clayton, Vic Change

Just In | Australia | World | Business | Sport | Analysis & Opinion | Fact Check | Programs

BREAKING NEWS Police arrest man in connection with stabbing death of 17-year-old Masa Vukotic in M

Print | Email | Facebook | Twitter | More

Treasurer Joe Hockey calls for answers over Australian Bureau of Statistics jobs data

By Michael Vincent and Simon Frazer

Updated 9 Oct 2014, 12:17pm

Federal Treasurer Joe Hockey says he wants answers to the problems the Australian Bureau of Statistics (ABS) has had with unemployment figures.

Mr Hockey, who is in the US to discuss Australia's G20 agenda, said last month's unemployment figures were "extraordinary".

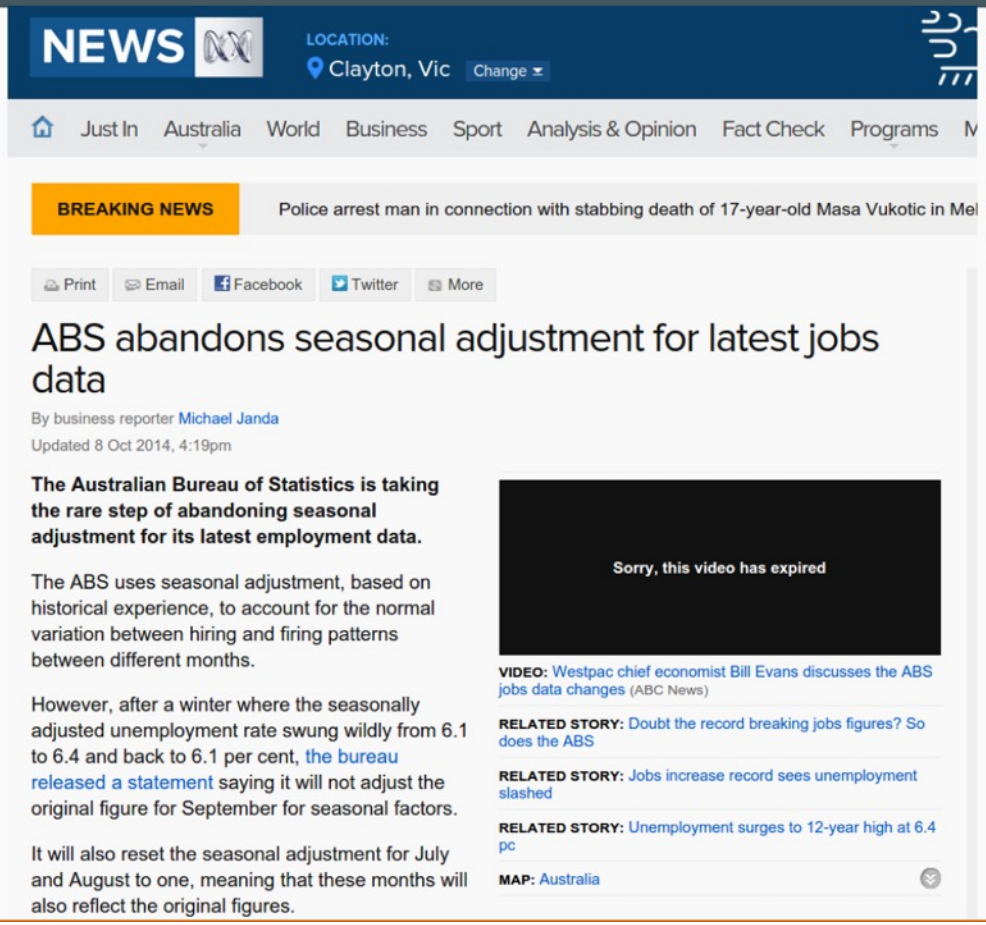
The rate was 6.1 per cent after jumping to a 12-year high of 6.4 per cent the previous month.

The ABS has now taken the rare step of abandoning seasonal adjustment for its latest employment data.

PHOTO: Joe Hockey says he is unhappy with the volatility of ABS unemployment figures. (AAP: Alan Porritt)

RELATED STORY: ABS abandons seasonal adjustment for latest jobs data

The ABS stuff-up



NEWS

LOCATION: Clayton, Vic Change

Just In | Australia | World | Business | Sport | Analysis & Opinion | Fact Check | Programs

BREAKING NEWS Police arrest man in connection with stabbing death of 17-year-old Masa Vukotic in Mel

Print | Email | Facebook | Twitter | More

ABS abandons seasonal adjustment for latest jobs data

By business reporter Michael Janda

Updated 8 Oct 2014, 4:19pm

The Australian Bureau of Statistics is taking the rare step of abandoning seasonal adjustment for its latest employment data.

The ABS uses seasonal adjustment, based on historical experience, to account for the normal variation between hiring and firing patterns between different months.

However, after a winter where the seasonally adjusted unemployment rate swung wildly from 6.1 to 6.4 and back to 6.1 per cent, the bureau released a statement saying it will not adjust the original figure for September for seasonal factors.

It will also reset the seasonal adjustment for July and August to one, meaning that these months will also reflect the original figures.

Sorry, this video has expired

VIDEO: Westpac chief economist Bill Evans discusses the ABS jobs data changes (ABC News)

RELATED STORY: Doubt the record breaking jobs figures? So does the ABS

RELATED STORY: Jobs increase record sees unemployment slashed

RELATED STORY: Unemployment surges to 12-year high at 6.4 pc

MAP: Australia

The ABS stuff-up

ABS jobs and unemployment figures – key questions answered by an expert

A professor of statistics at Monash University explains exactly what is seasonal adjustment, why it matters and what went wrong in the July and August figures

School leavers come on to the jobs market at the same time, causing a seasonal fluctuation. Photograph: Brian Snyder/Reuters

The Australian Bureau of Statistics has retracted its seasonally adjusted employment data for July and August, which recorded huge swings in the jobless rate. The ABS is also planning to review the methods it uses for seasonal adjustment to ensure its figures are as accurate as possible. Rob Hyndman, a professor of statistics at Monash University and member of the bureau's methodology advisory board, answers our questions:

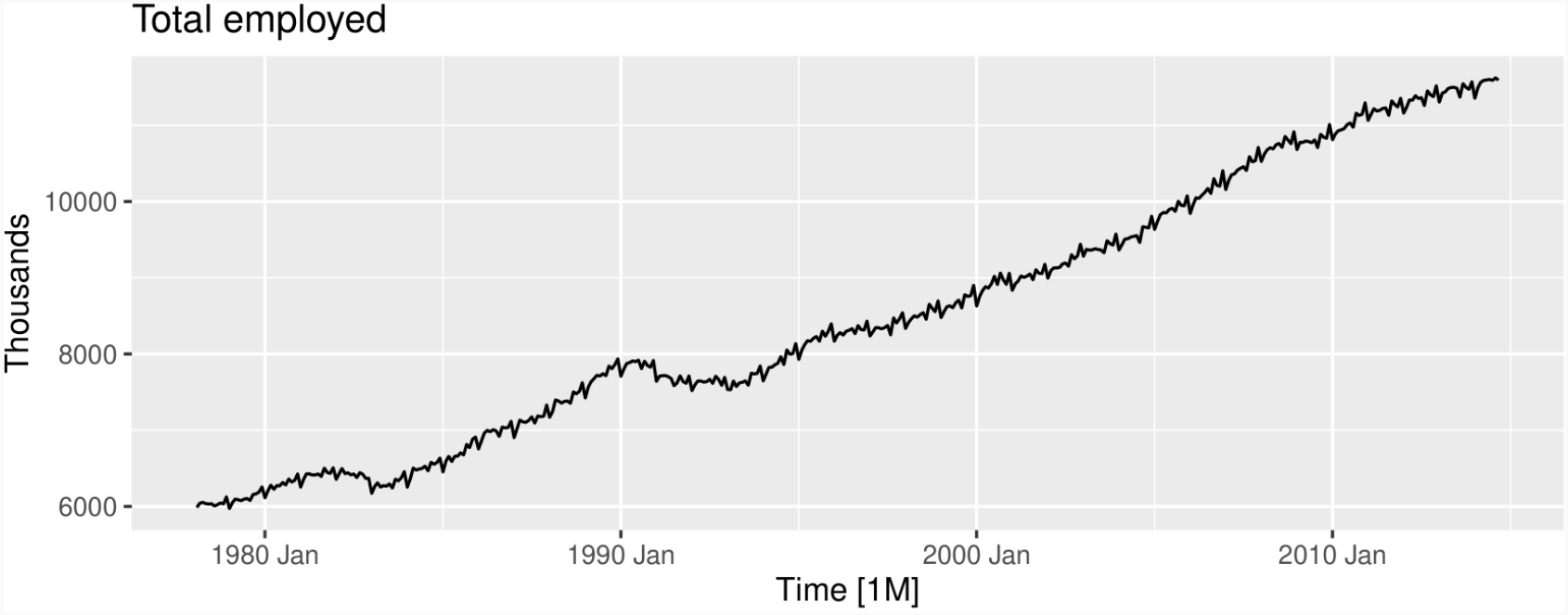
The ABS stuff-up

```
employed
```

```
## # A tsibble: 440 x 4 [1M]
##         Time Month  Year Employed
##        <mth> <ord> <dbl>    <dbl>
##  1 1978 Feb Feb    1978    5986.
##  2 1978 Mar Mar    1978    6041.
##  3 1978 Apr Apr    1978    6054.
##  4 1978 May May    1978    6038.
##  5 1978 Jun Jun    1978    6031.
##  6 1978 Jul Jul    1978    6036.
##  7 1978 Aug Aug    1978    6005.
##  8 1978 Sep Sep    1978    6024.
##  9 1978 Oct Oct    1978    6046.
## 10 1978 Nov Nov    1978    6034.
## # ... with 430 more rows
```

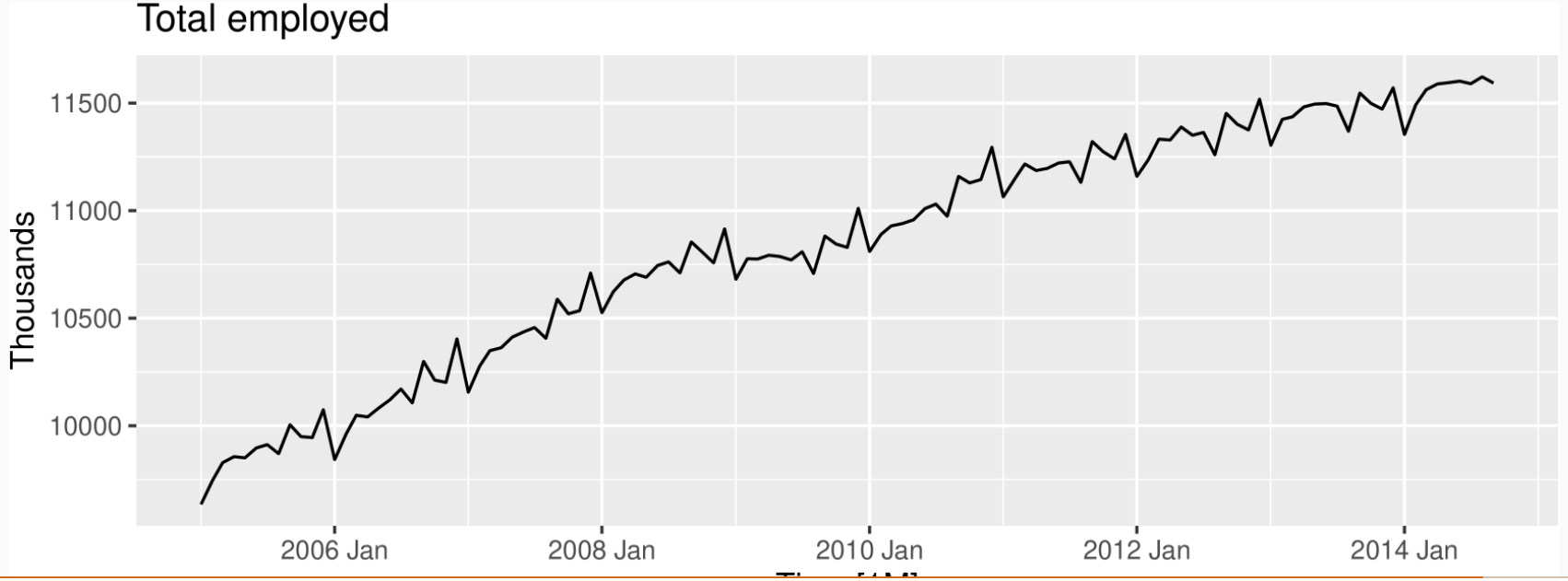
The ABS stuff-up

```
employed %>%
  autoplot(Employed) +
  labs(title = "Total employed", y = "Thousands")
```



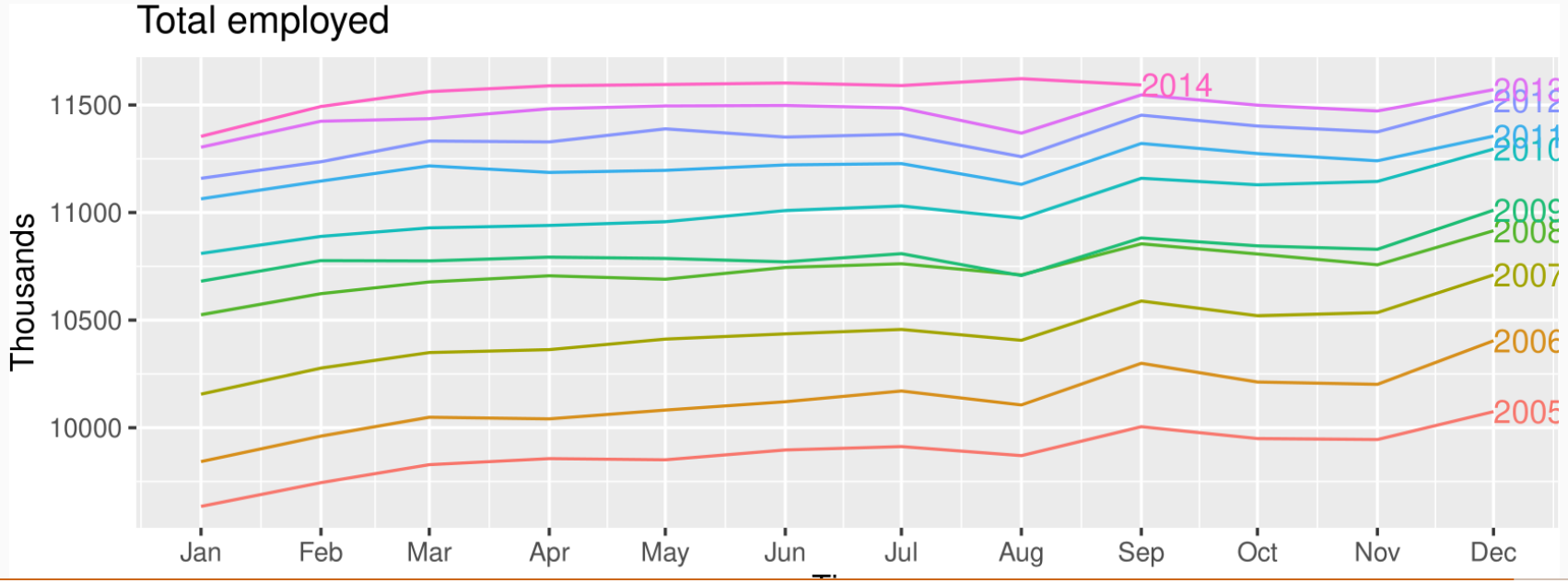
The ABS stuff-up

```
employed %>%
  filter(Year >= 2005) %>%
  autoplot(Employed) +
  labs(title = "Total employed", y = "Thousands")
```



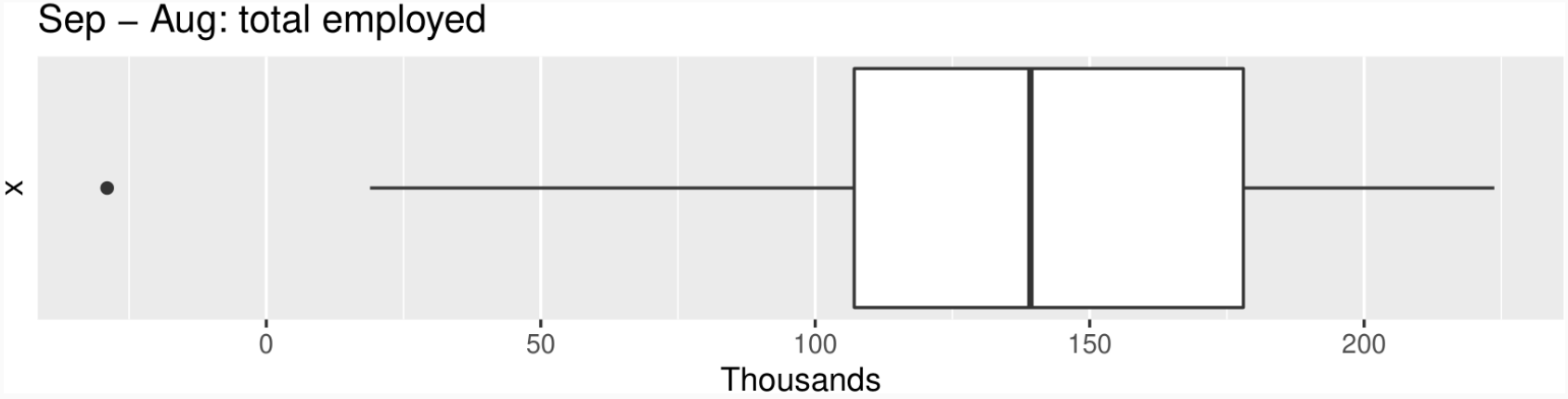
The ABS stuff-up

```
employed %>%
  filter(Year >= 2005) %>%
  gg_season(Employed, label = "right") +
  labs(title = "Total employed", y = "Thousands")
```



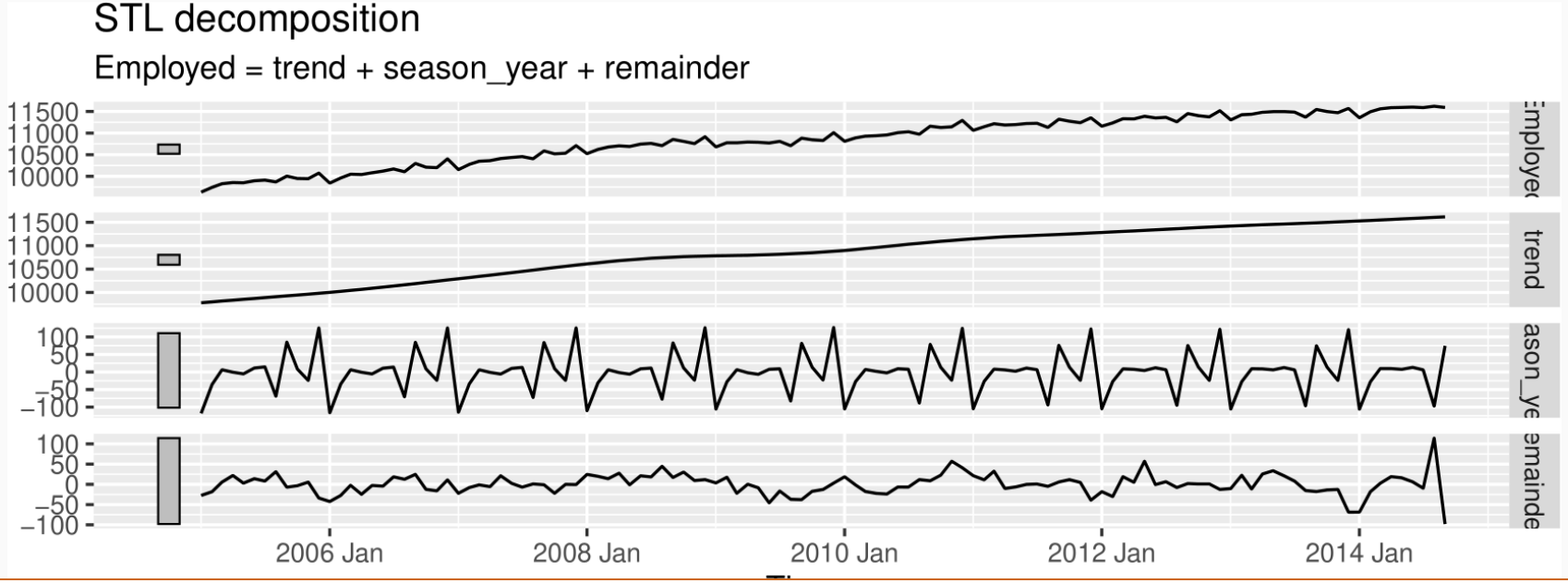
The ABS stuff-up

```
employed %>%
  mutate(diff = difference(Employed)) %>%
  filter(Month == "Sep") %>%
  ggplot(aes(y = diff, x = 1)) +
  geom_boxplot() + coord_flip() +
  labs(title = "Sep - Aug: total employed", y = "Thousands") +
  scale_x_continuous(breaks = NULL, labels = NULL)
```



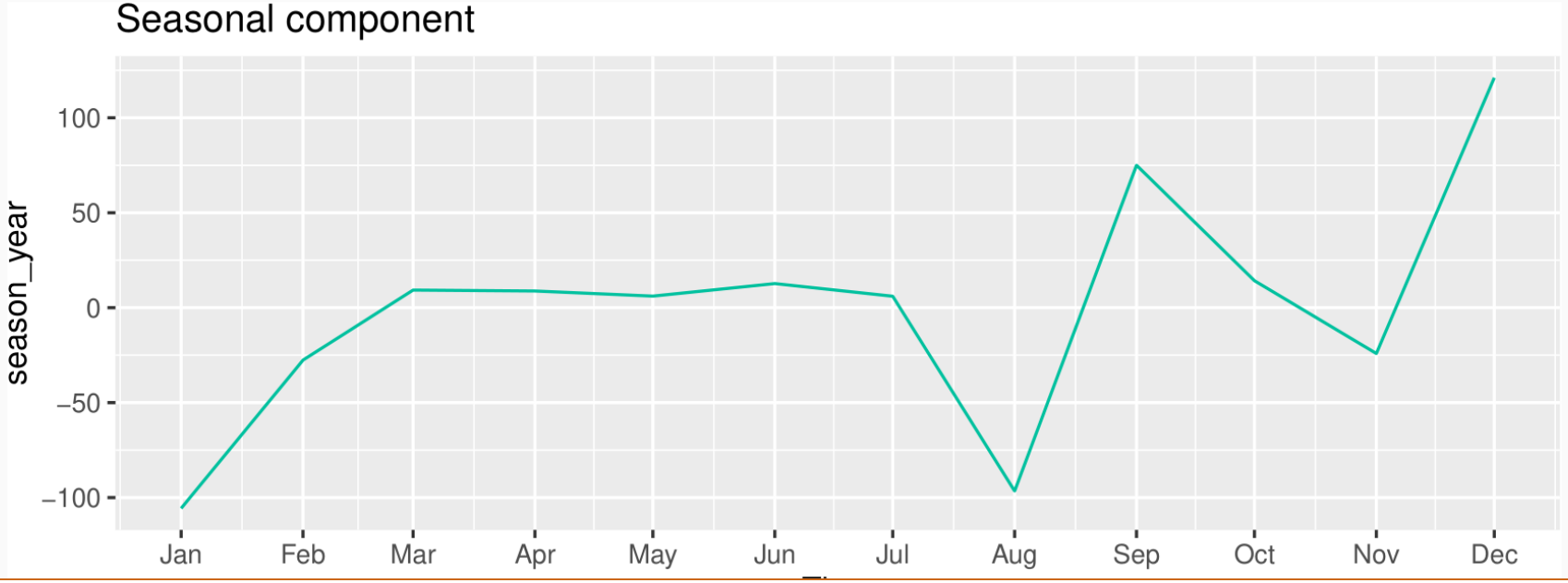
The ABS stuff-up

```
dcmp <- employed %>%
  filter(Year >= 2005) %>%
  model(stl = STL(Employed ~ season(window = 11), robust = TRUE))
components(dcmp) %>% autoplot()
```



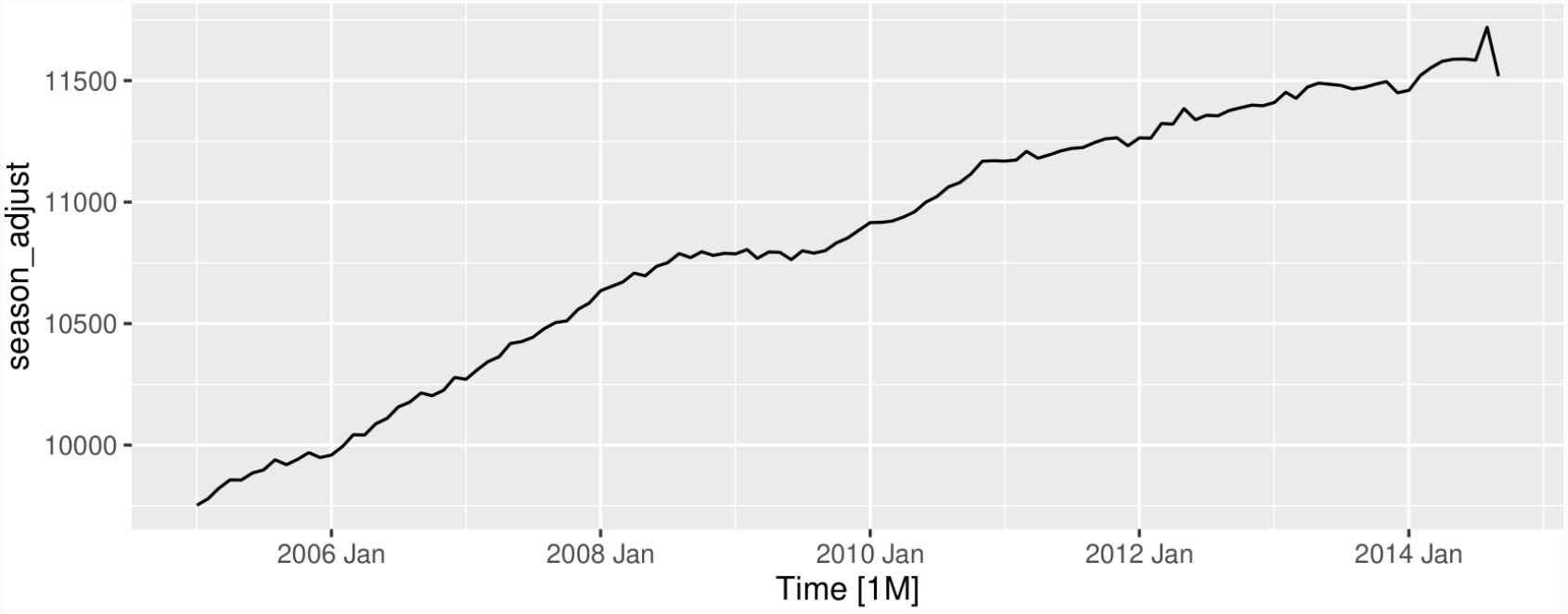
The ABS stuff-up

```
components(dcmp) %>%
  filter(year(Time) == 2013) %>%
  gg_season(season_year) +
  labs(title = "Seasonal component") + guides(colour = "none")
```



The ABS stuff-up

```
components(dcmp) %>%
  as_tsibble() %>%
  autoplot(season_adjust)
```



The ABS stuff-up

- August 2014 employment numbers higher than expected.
- Supplementary survey usually conducted in August for employed people.
- Most likely, some employed people were claiming to be unemployed in August to avoid supplementary questions.
- Supplementary survey not run in 2014, so no motivation to lie about employment.
- In previous years, seasonal adjustment fixed the problem.
- The ABS has now adopted a new method to avoid the bias.