

# Predictive Analytics

Ch2. Time series graphics

Prof. Dr. Benjamin Buchwitz

Wir geben Impulse



- 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
##   <dbl> <fct>          <dbl> <dbl> <dbl>      <dbl>
## 1  1960 Afghanistan 5377777811.  7.02  4.13  8996351
## 2  1961 Afghanistan 5488888896.  8.10  4.45  9166764
## 3  1962 Afghanistan 5466666678.  9.35  4.88  9345868
## 4  1963 Afghanistan 7511111191. 16.9   9.17  9533954
## 5  1964 Afghanistan 8000000044. 18.1   8.89  9731361
## 6  1965 Afghanistan 10066666638. 21.4  11.3  9938414
## 7  1966 Afghanistan 13999999967. 18.6   8.57 10152331
## 8  1967 Afghanistan 16733333418. 14.2   6.77 10372630
## 9  1968 Afghanistan 13733333367. 15.2   8.90 10604346
## 10 1969 Afghanistan 14088888922. 15.0  10.1 10854428
```

# tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:      Country [263]
```

```
##   Year Country      GDP Imports Exports Population
##   <dbl> <fct>          <dbl> <dbl>   <dbl>       <dbl>
## 1  Index afgghanistan 5377777811.  7.02   4.13   8996351
## 2  1961 Afghanistan 5488888896.  8.10   4.45   9166764
## 3  1962 Afghanistan 5466666678.  9.35   4.88   9345868
## 4  1963 Afghanistan 7511111191. 16.9    9.17   9533954
## 5  1964 Afghanistan 8000000044. 18.1    8.89   9731361
## 6  1965 Afghanistan 10066666638. 21.4   11.3   9938414
## 7  1966 Afghanistan 13999999967. 18.6    8.57  10152331
## 8  1967 Afghanistan 16733333418. 14.2    6.77  10372630
## 9  1968 Afghanistan 13733333367. 15.2    8.90  10604346
## 10 1969 Afghanistan 14088888922. 15.0   10.1  10854428
```

# tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:      Country [263]
```

```
##   Year Country      GDP Imports Exports Population
##   <dbl> <fct>          <dbl> <dbl> <dbl>      <dbl>
## 1  Index Key      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

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:      Country [263]
```

```
##   Year Country      GDP Imports Exports Population
##   <dbl> <fct>      <dbl>  <dbl>  <dbl>      <dbl>
```

```
## 1   Index Key      5 Measured variables
```

```
## 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
##   <qtr> <chr>    <chr> <chr>    <dbl>
## 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.
```

# tsibble objects

```
tourism
```

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:           Region, State, Purpose [304]
##   Quarter Region  State Purpose  Trips
##   <qtr> <chr>    <chr> <chr>    <dbl>
## 1 1 Index 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.
```



# tsibble objects

```
tourism
```

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:           Region, State, Purpose [304]
##   Quarter Region  State Purpose  Trips
##   <qtr> <chr>    <chr> <chr>    <dbl>
## 1 1 Index     Keys     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.
```

# tsibble objects

```
tourism
```

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:           Region, State, Purpose [304]
##   Quarter Region  State Purpose  Trips
##   <qtr> <chr>    <chr> <chr>    <dbl>
## 1 1 Index      Keys      Measure
## 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.
```

# tsibble objects

```
tourism
```

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:      Region, State, Purpose [304]
##   Quarter Region  State Purpose  Trips
##   <qtr> <chr>    <chr> <chr>    <dbl>
## 1 1 Index      Keys      Measure
## 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.

- 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 Mär         34  
## 4 2019 Apr         30  
## 5 2019 Mai         25
```



Common time index variables can be created with these functions:

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

- 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 tibble

```
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 tibble

```
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
```

## 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
```

- 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 Pharmaceutical Benefits Scheme





The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy 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
##   <moth> <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 Okt  Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~ 15380 57222
## 5 1991 Nov  Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~ 14371 52120
## 6 1991 Dez  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 Mär  Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~ 16898 61108
## 10 1992 Apr  Concessional Co-pay~ A     Alimenta~ A01   STOMATOL~ 18141 65356
```

## 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 Okt  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 Dez  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
```

## 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 Okt  Concessional Co-payments 1848507  
## 5 1991 Nov  Concessional Co-payments 1686458  
## 6 1991 Dez  Concessional Co-payments 1843079  
## 7 1992 Jan  Concessional Co-payments 1564702
```

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 Okt      3611003  
## 5 1991 Nov      3565869  
## 6 1991 Dez      4306371  
## 7 1992 Jan      5088335
```

## 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 Okt         3.61  
## 5 1991 Nov         3.57  
## 6 1991 Dez         4.31
```

## 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
```

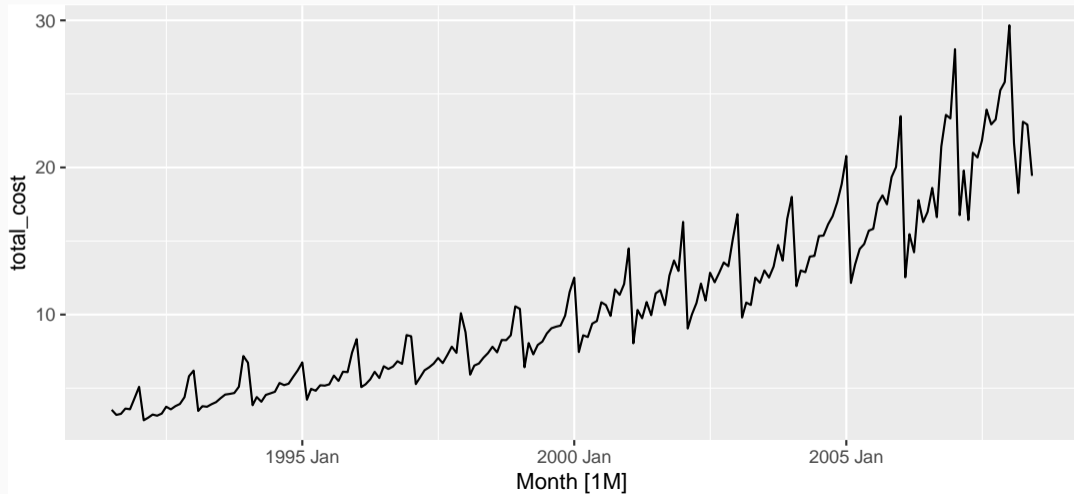
```
## # A tsibble: 204 x 2 [1M]  
##       Month total_cost  
##       <mth>      <dbl>  
## 1 1991 Jul         3.53  
## 2 1991 Aug         3.18  
## 3 1991 Sep         3.25  
## 4 1991 Okt         3.61  
## 5 1991 Nov         3.57  
## 6 1991 Dez         4.31
```



- 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

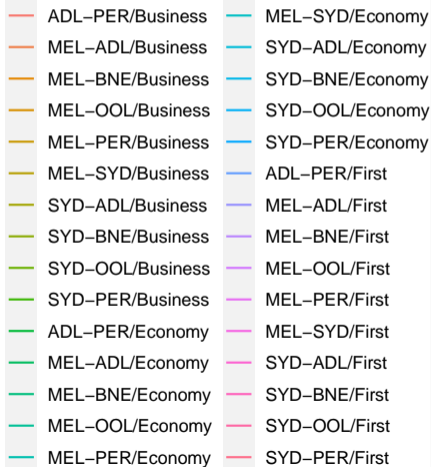
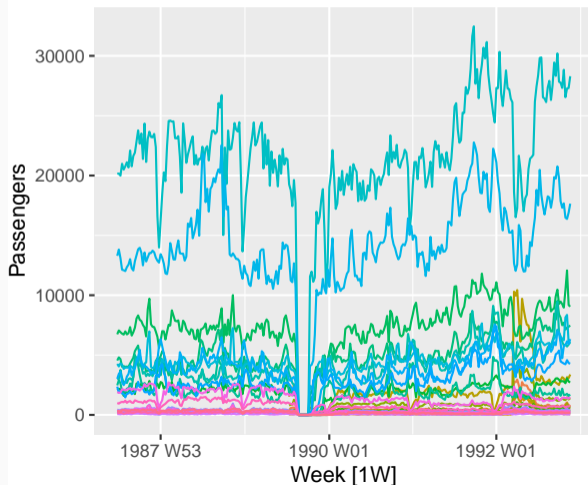
# Time plots

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



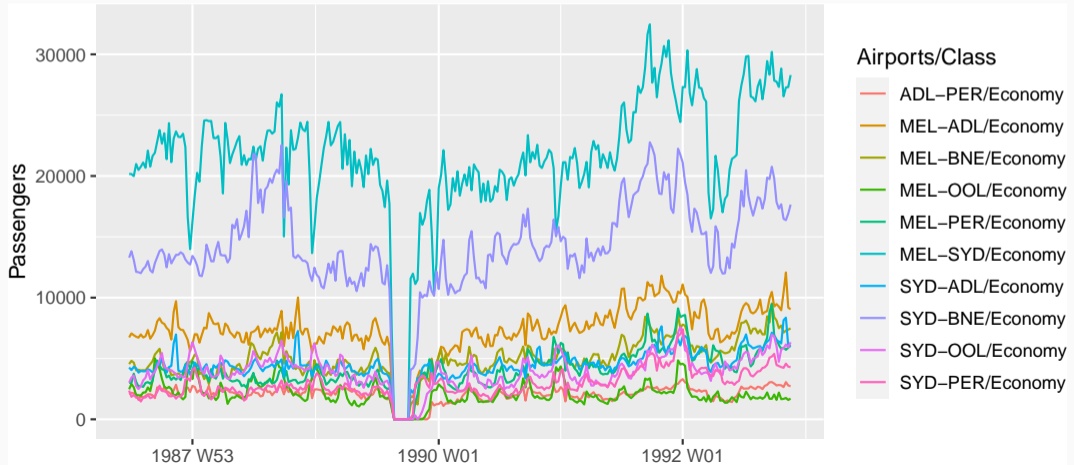
# Ansett airlines

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



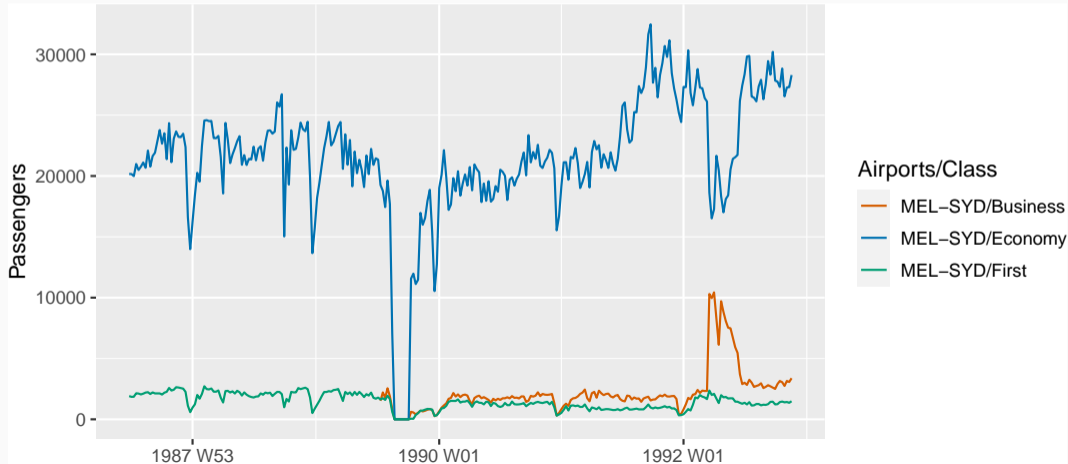
# Ansett airlines

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



# Ansett airlines

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



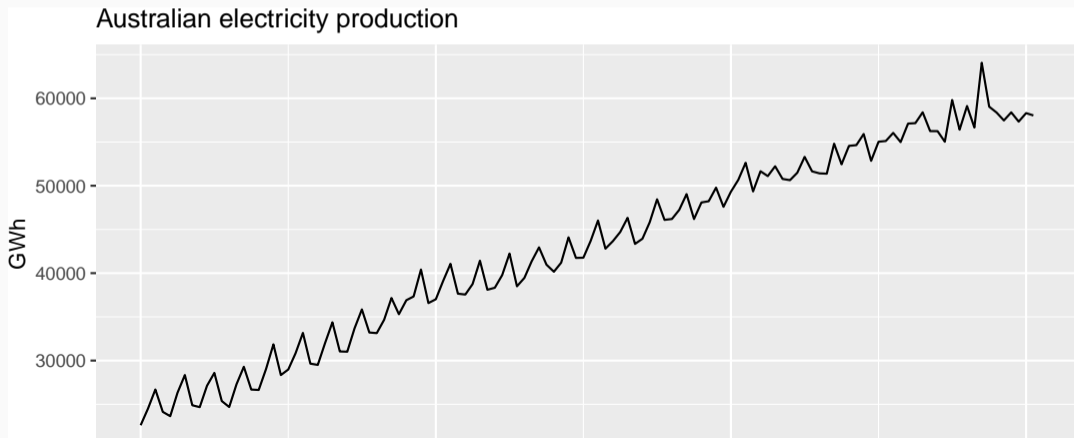
- Trend** pattern exists when there is a long-term increase or decrease in the data.
- Seasonal** pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).
- Cyclic** pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).

### 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

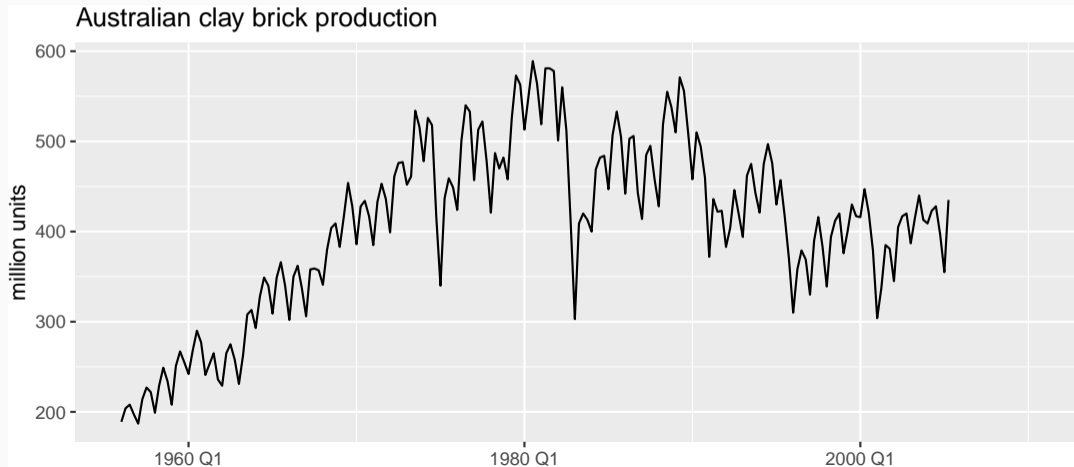
```
aus_production %>%  
  filter(year(Quarter) >= 1980) %>%  
  autoplot(Electricity) +  
  labs(y = "GWh", title = "Australian electricity production")
```





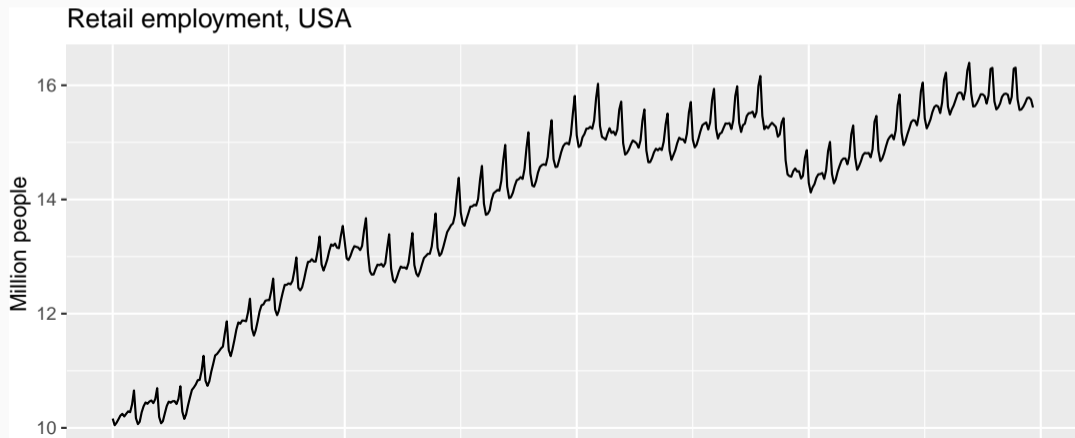
# Time series patterns

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



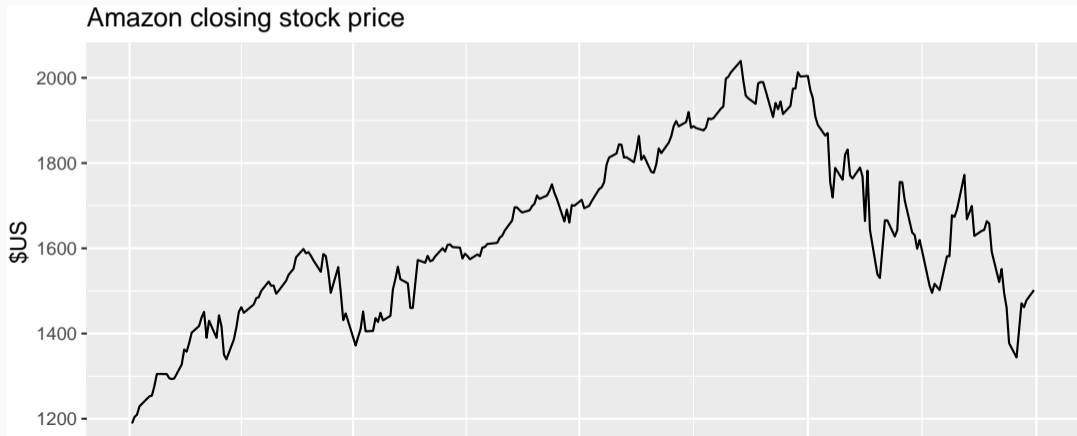
# Time series patterns

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



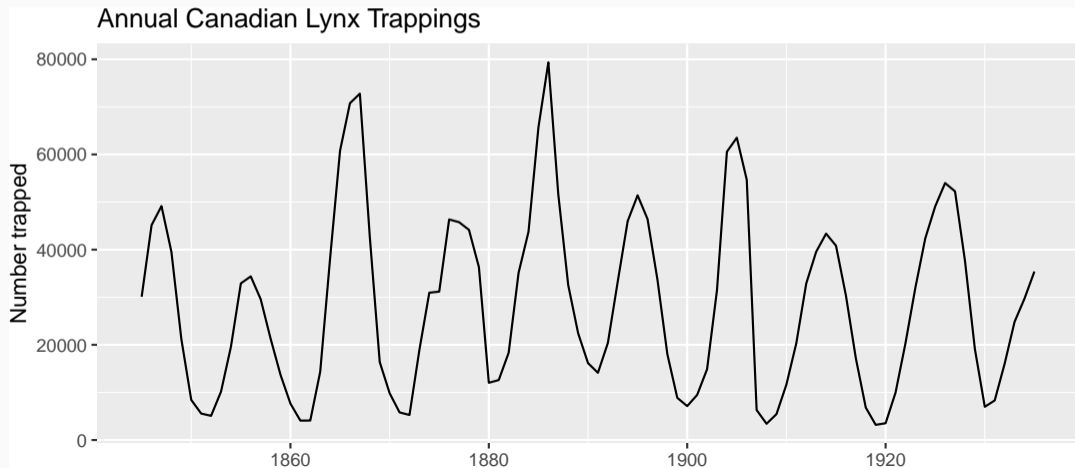
# Time series patterns

```
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")
```



### 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

### Differences between seasonal and cyclic patterns:

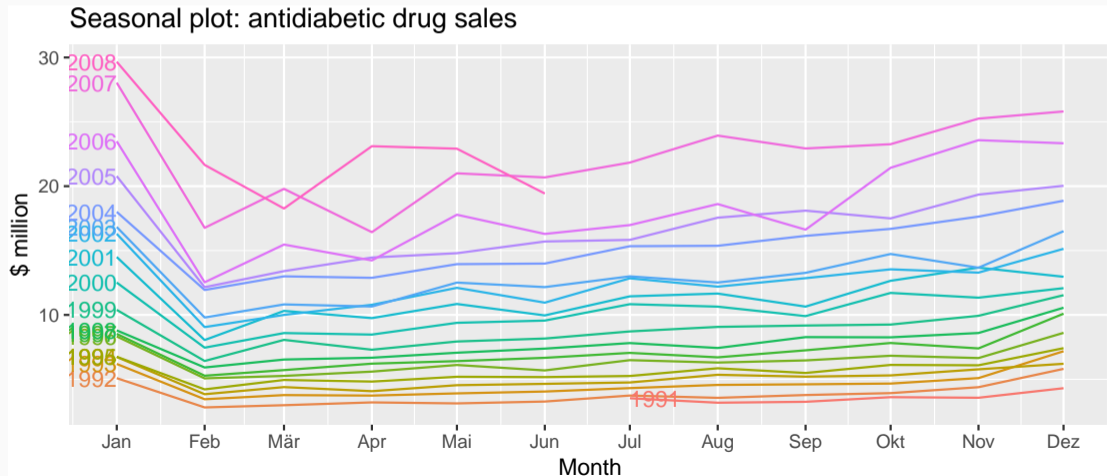
- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

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

- 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

# Seasonal plots

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





- 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()`

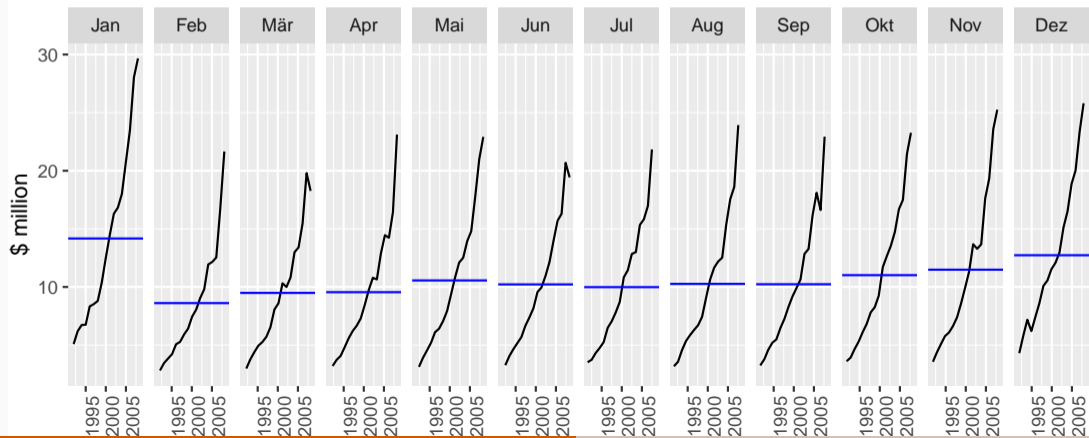
# Seasonal subseries plots

```
a10 %>%
```

```
gg_subseries(total_cost) +
```

```
labs(y = "$ million", title = "Subseries plot: antidiabetic drug sales")
```

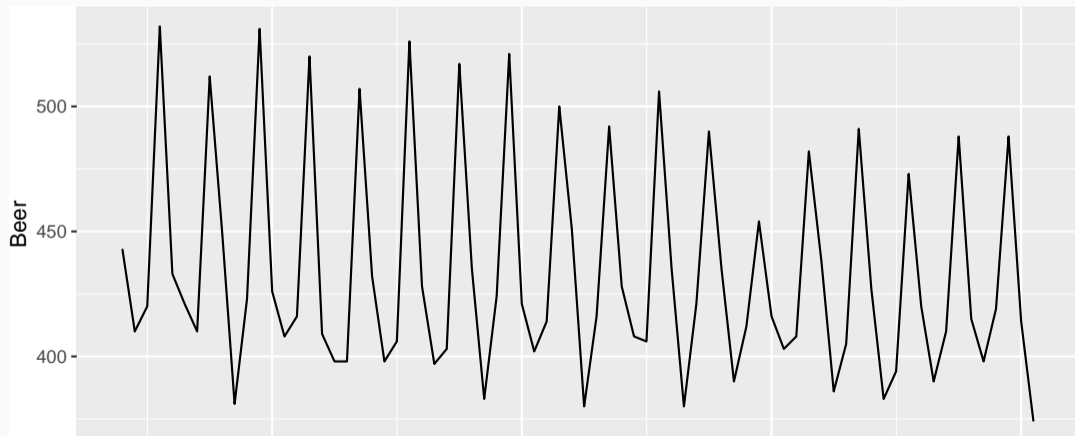
Subseries plot: antidiabetic drug sales



- 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()`

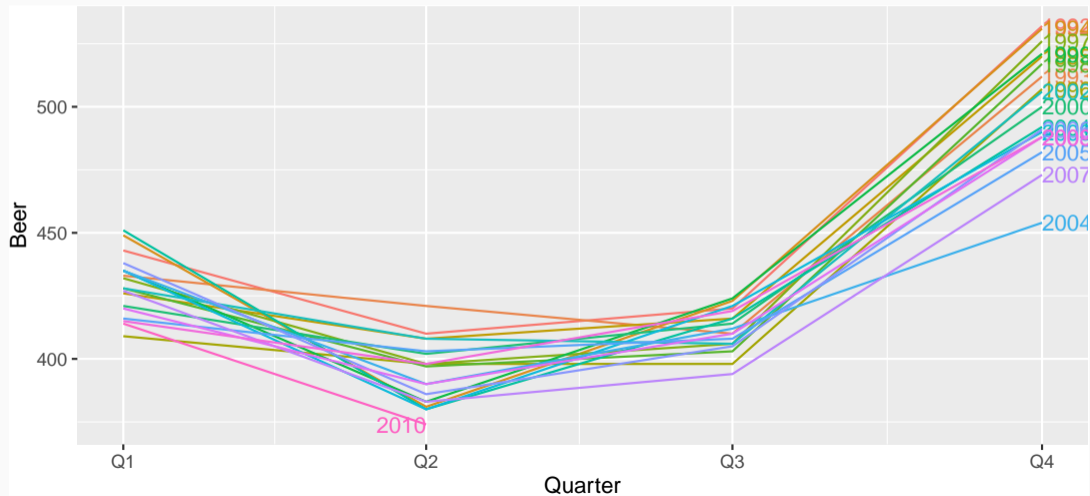
# Quarterly Australian Beer Production

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



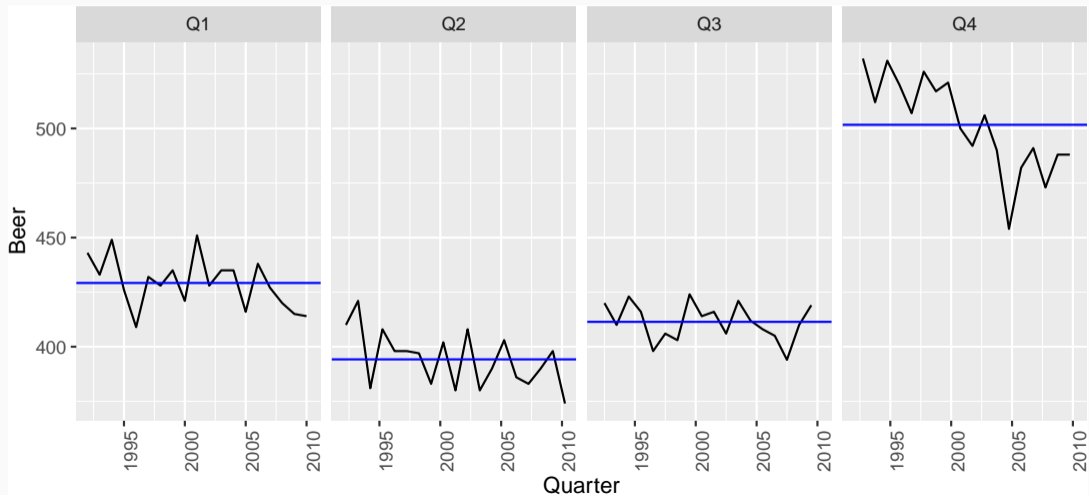
# Quarterly Australian Beer Production

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



# Quarterly Australian Beer Production

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



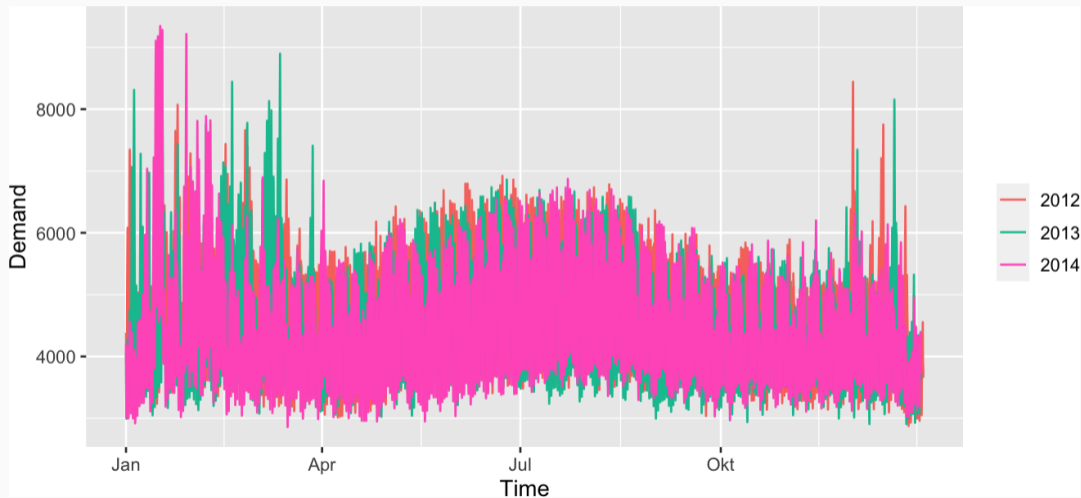
## Multiple seasonal periods

```
vic_elec
```

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

# Multiple seasonal periods

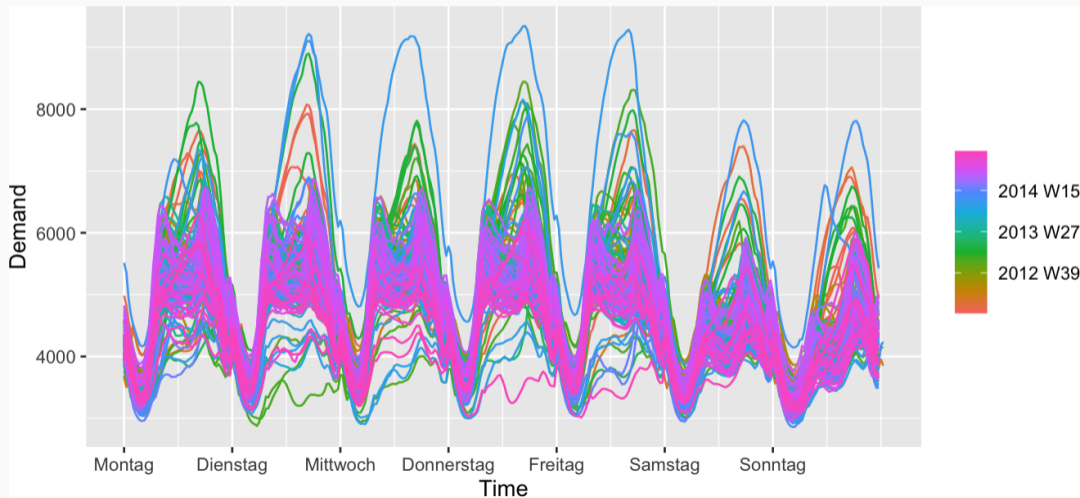
```
vic_elec %>% gg_season(Demand)
```





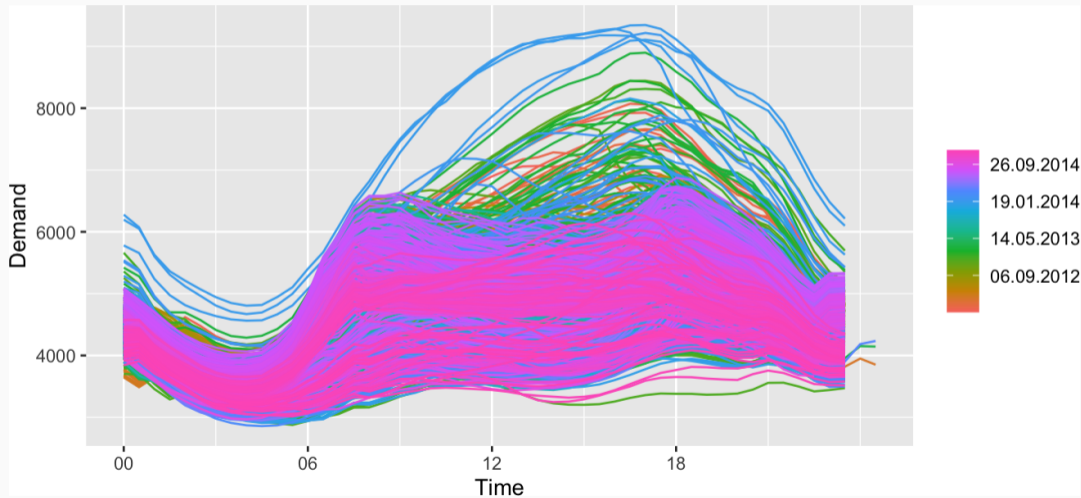
# Multiple seasonal periods

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



# Multiple seasonal periods

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



# Australian holidays

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

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

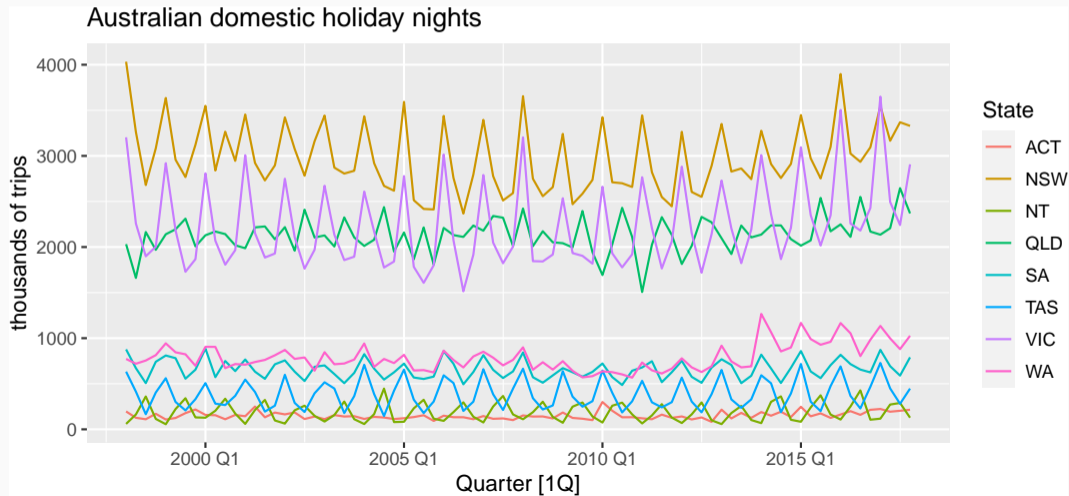
```
## 6 ACT    1999 Q2  125.
```

```
## 7 ACT    1999 Q3  178.
```

```
## 8 ACT    1999 Q4  218.
```

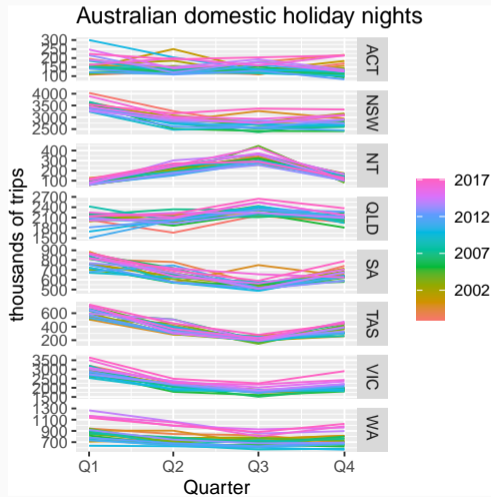
# Australian holidays

```
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")
```

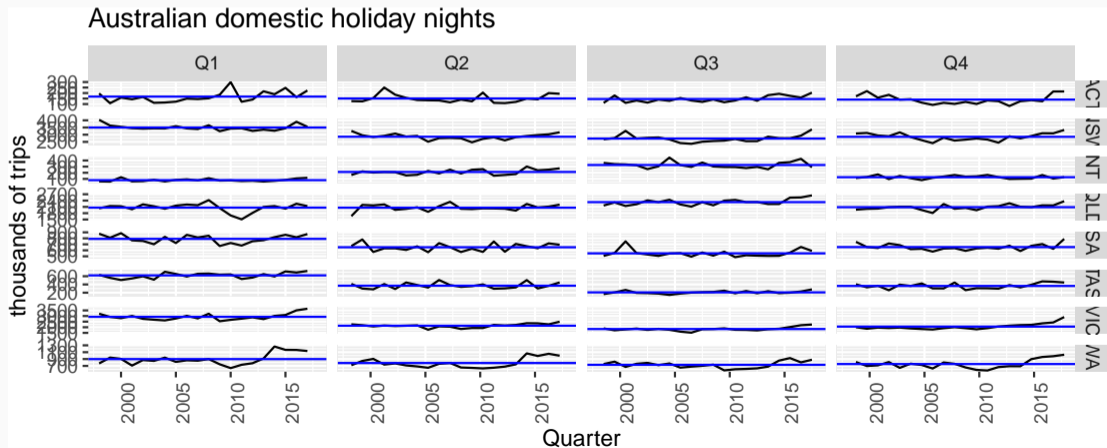


# Seasonal subseries plots

```
holidays %>%
```

```
gg_subseries(Trips) +
```

```
labs(y = "thousands of trips", title = "Australian domestic holiday nights")
```



- 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

## Example: Beer production

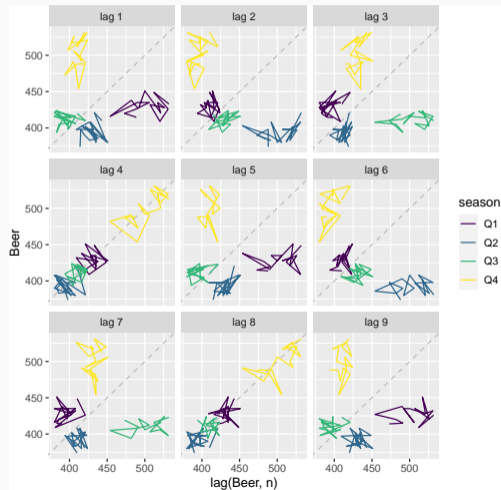
```
new_production <- aus_production %>%  
  filter(year(Quarter) >= 1992)  
new_production
```

```
## # 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 5317 475 1755 41100 150
```



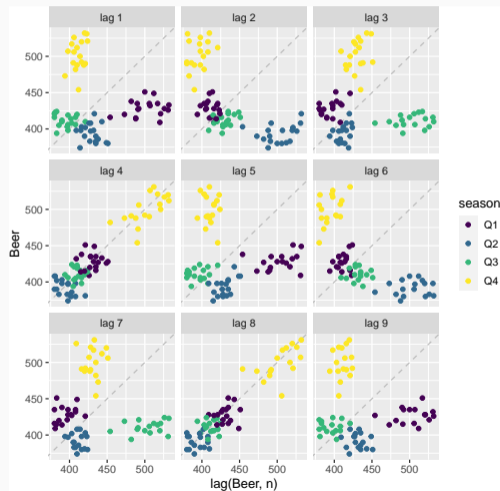
# Example: Beer production

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



# Example: Beer production

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



- 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):
  - ▶  $r_1 = \text{Correlation}(y_t, y_{t-1})$
  - ▶  $r_2 = \text{Correlation}(y_t, y_{t-2})$
  - ▶  $r_3 = \text{Correlation}(y_t, y_{t-3})$
  - ▶ etc.

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

**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  $y$ .

**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.

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  $r_k = c_k/c_0$

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:

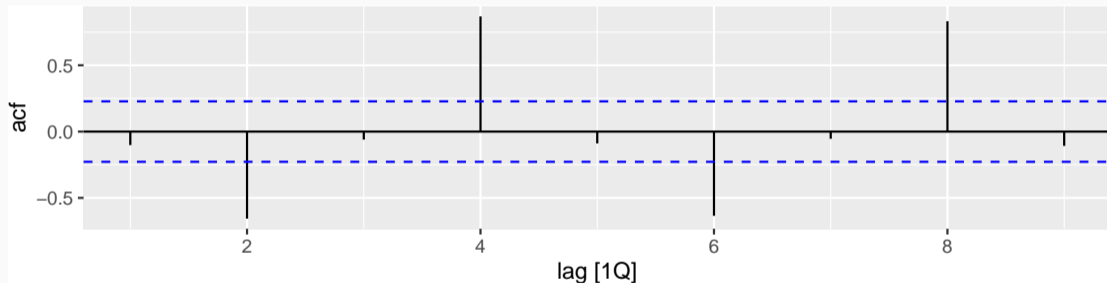
```
new_production %>% ACF(Beer, lag_max = 9)
```

```
## # A tibble: 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
## 8 8Q 0.832
## 9 9Q -0.108
```

# Autocorrelation

Results for first 9 lags for beer data:

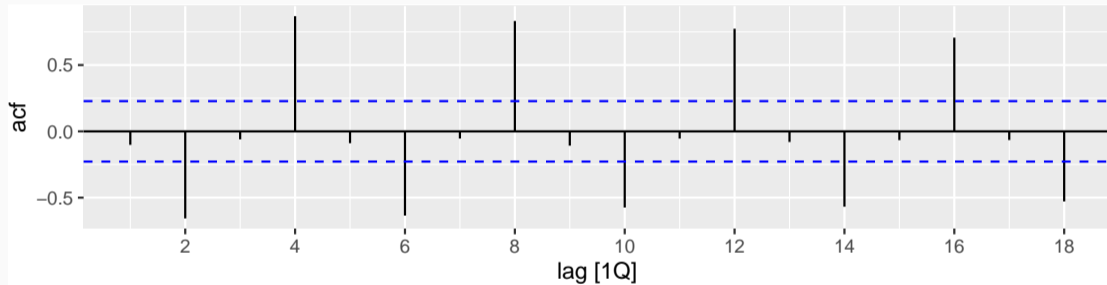
```
new_production %>% ACF(Beer, lag_max = 9) %>% autoplot()
```



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

# Autocorrelation

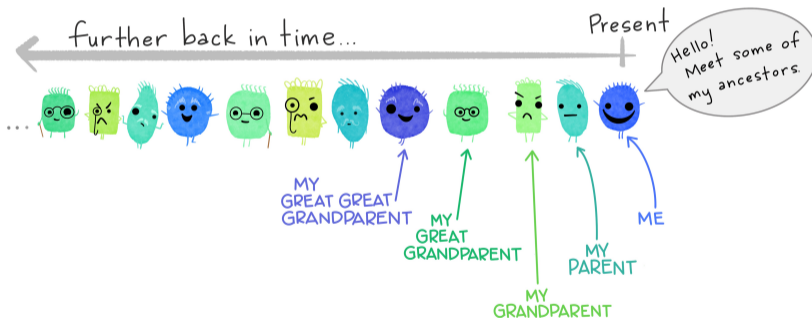
```
new_production %>% ACF(Beer) %>% autoplot()
```



- $r_4$  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.

- 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.

## intro to the autocorrelation function (ACF)

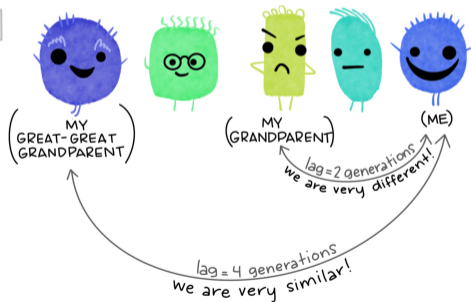


# Autocorrelation functions

*in our family* MONSTERS tend to be...

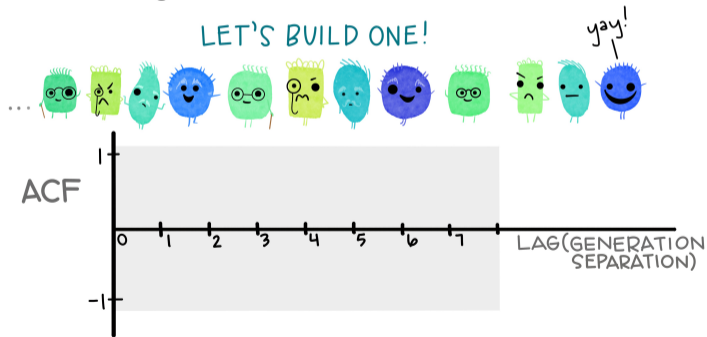
- A little similar to their parent and great-grandparent
- Very different from their grandparent
- Very similar to their great-great grandparent

FOR EXAMPLE:



## THE autocorrelation function (ACF)

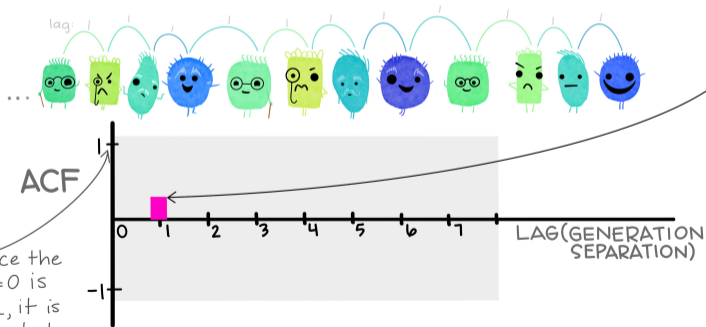
The ACF is a plot of autocorrelation between a variable and itself separated by specified lags (in our case, generations)



# Autocorrelation functions

At lag = 1, we find the correlation between **monsters** and their **parent**.

They are **somewhat positively correlated**.

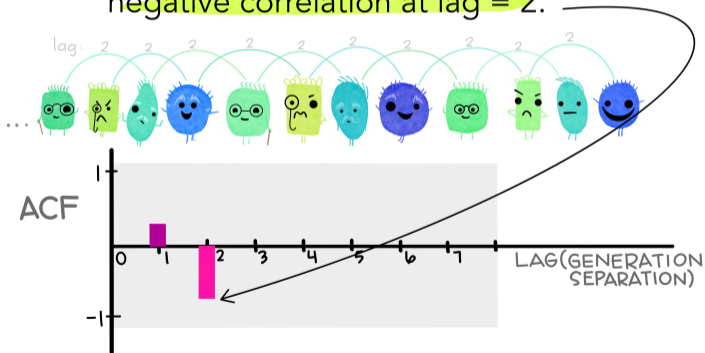




# Autocorrelation functions

At lag = 2, we find the correlation between **monsters** and their **grandparent**.

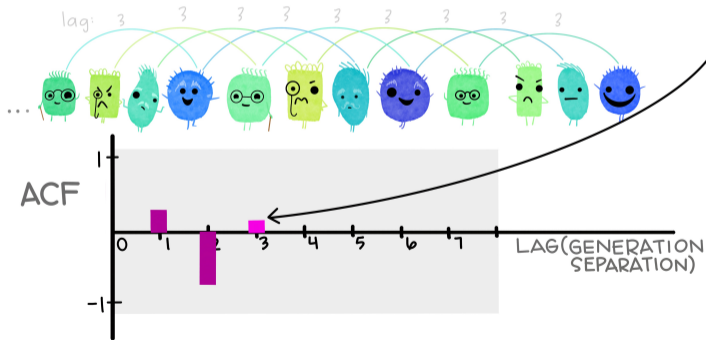
Since they tend to be very different, we find a **negative correlation at lag = 2**.



# Autocorrelation functions

At lag = 3, we find the correlation between **monsters** and their **great-grandparent**.

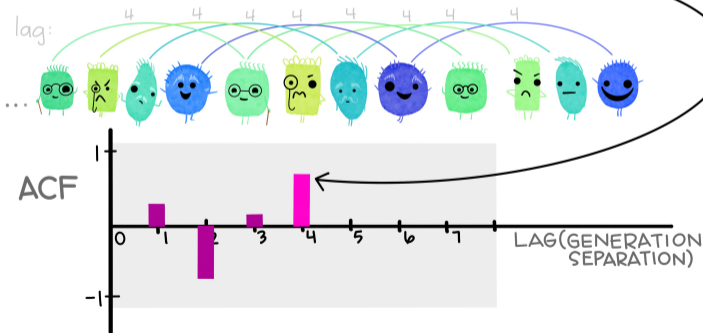
They are slightly positively correlated.



# Autocorrelation functions

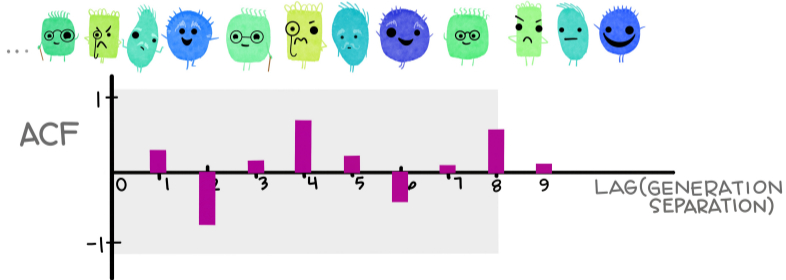
At lag = 4, we find the correlation between **monsters** and their **great-great grandparent**.

They tend to be very similar  
(there is a positive correlation).



# Autocorrelation functions

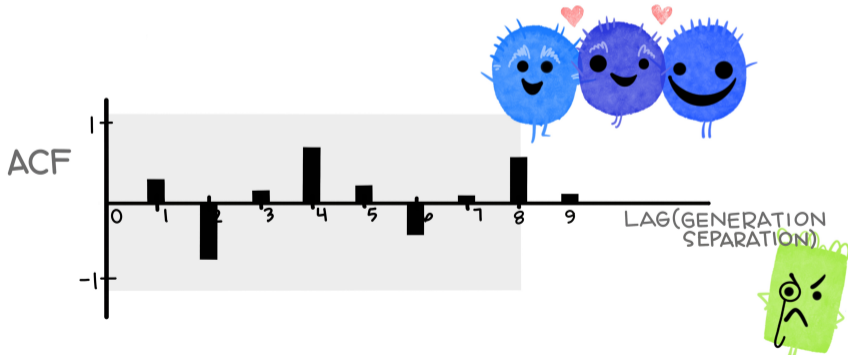
...and we continue finding the correlations as we increase the lag (generations) between the monsters...



# Autocorrelation functions

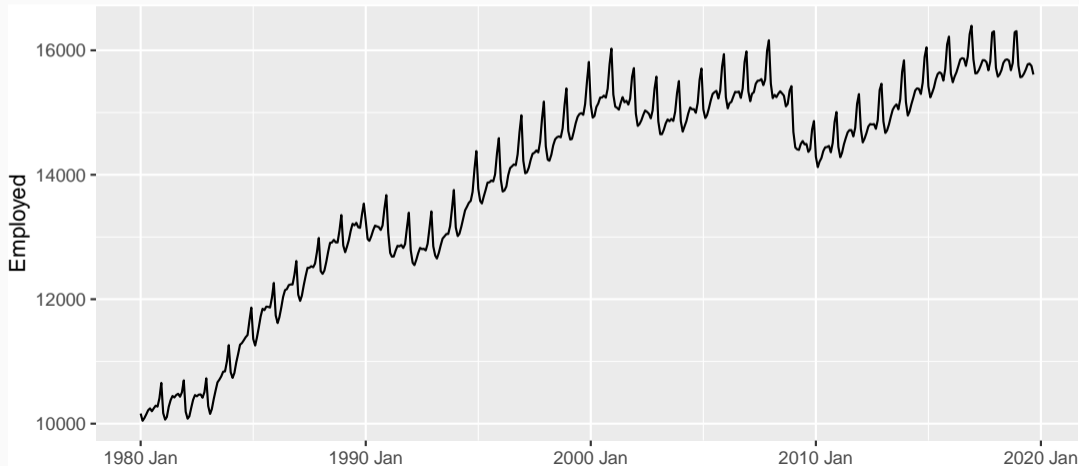
*in summary:*

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



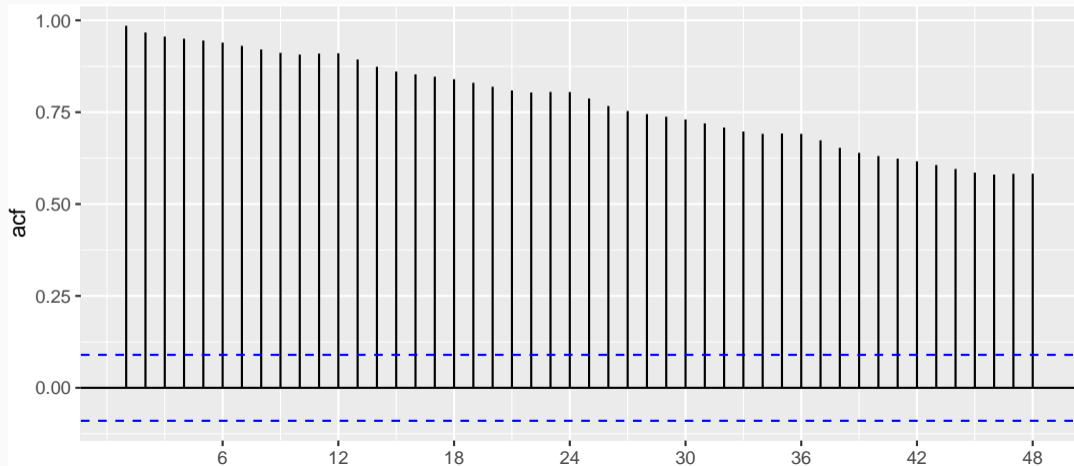
# US retail trade employment

```
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 tibble: 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.
```

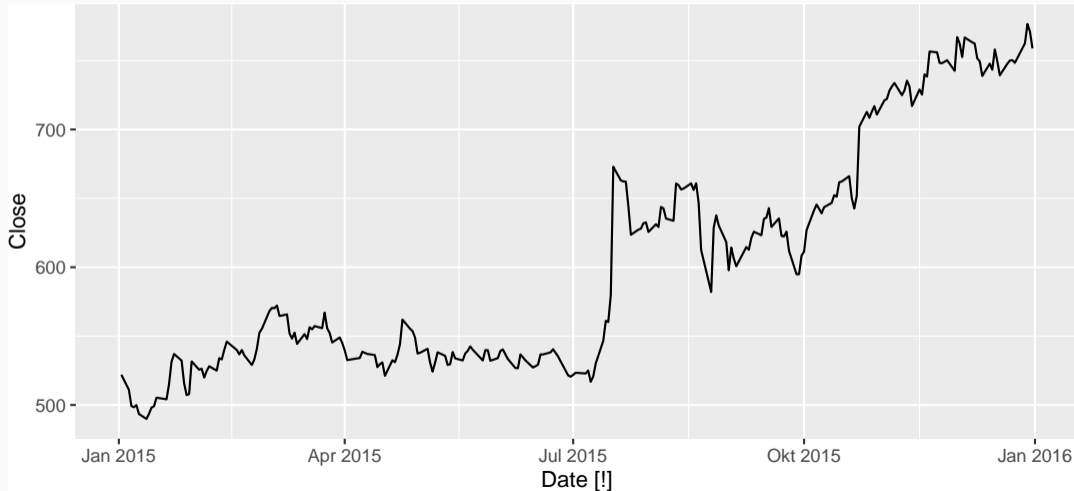
```
## 8 2015-01-13  493.
```

```
## 9 2015-01-14  498.
```



# Google stock price

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



# Google stock price

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

```
## # A tibble: 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
```

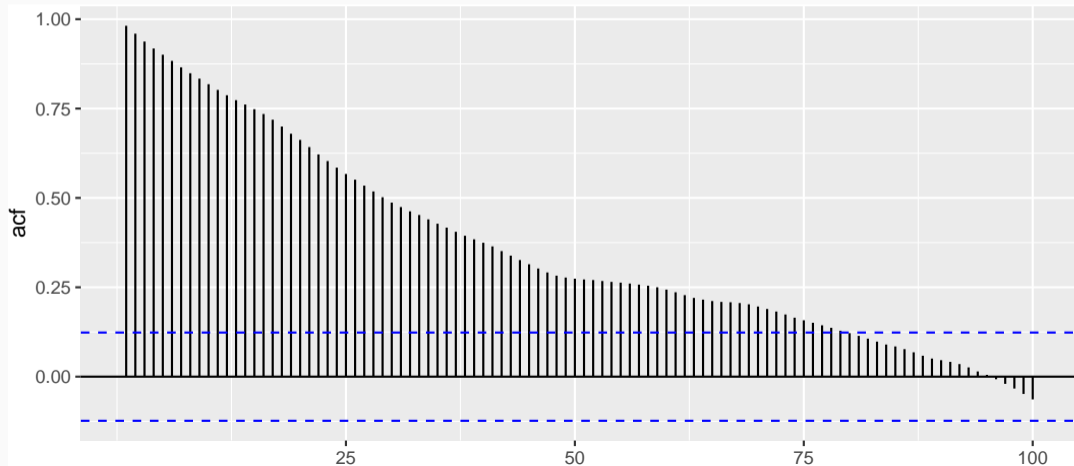
```
## 8     8 0.849
```

```
## 9     9 0.834
```

```
## 10    10 0.818
```

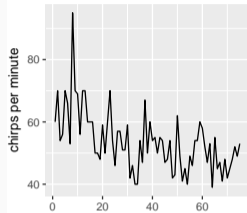
# Google stock price

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

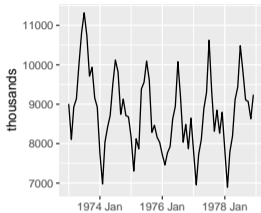


# Which is which?

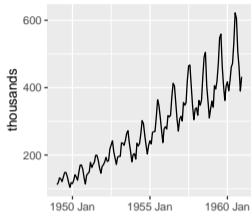
1. Daily temperature of cow



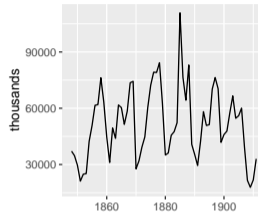
2. Monthly accidental deaths



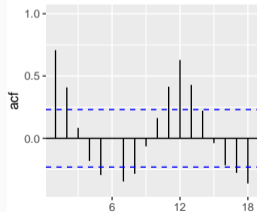
3. Monthly air passengers



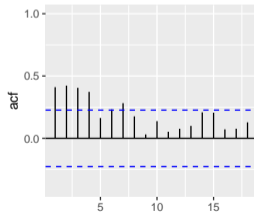
4. Annual mink trappings



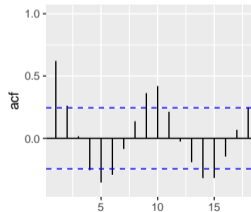
A



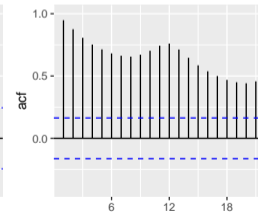
B



C



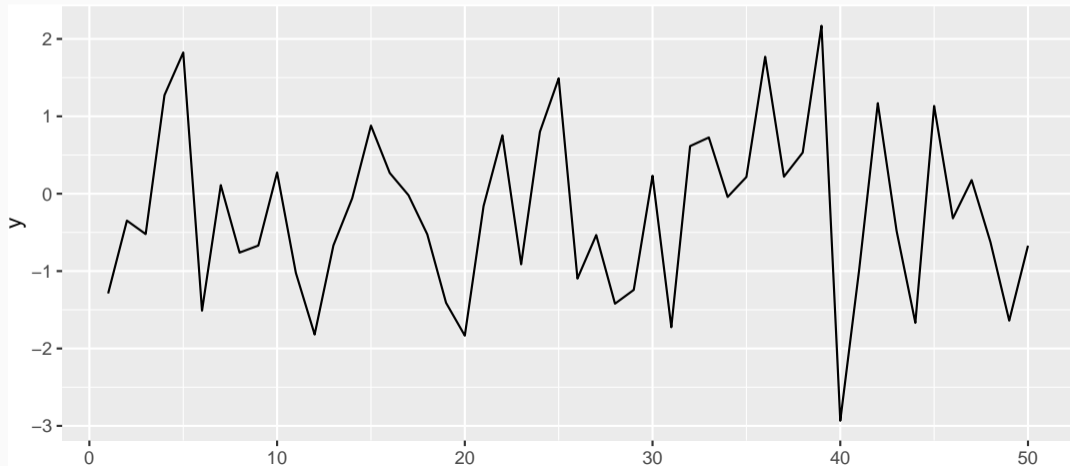
D



- 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

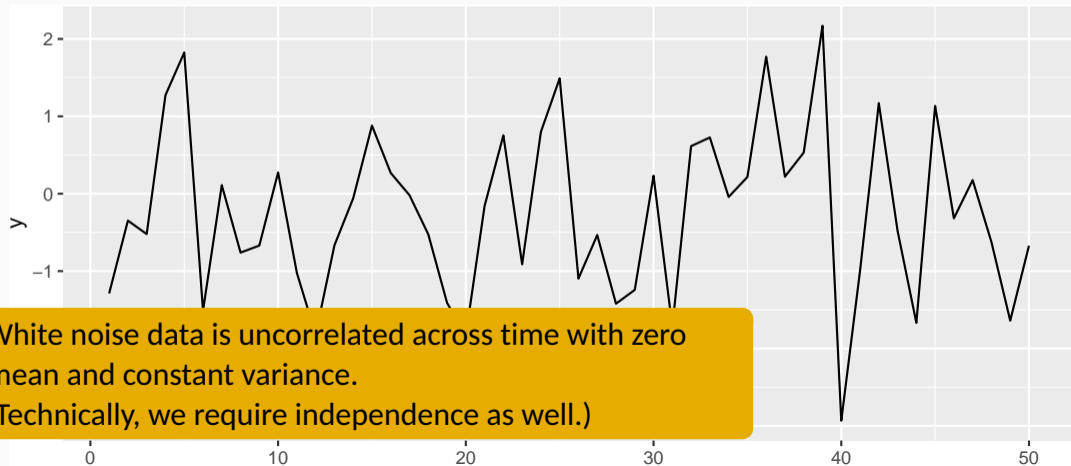
## Example: White noise

```
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)
```

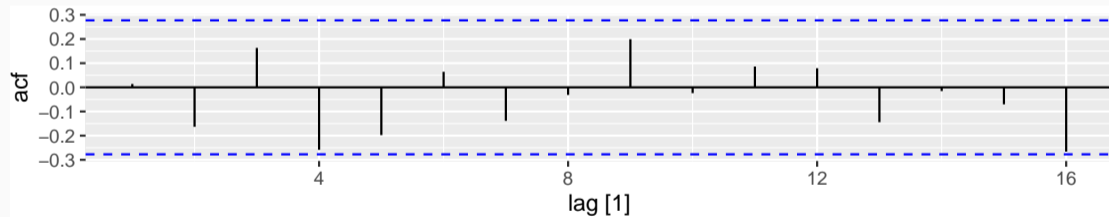


White noise data is uncorrelated across time with zero mean and constant variance.  
(Technically, we require independence as well.)

## 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

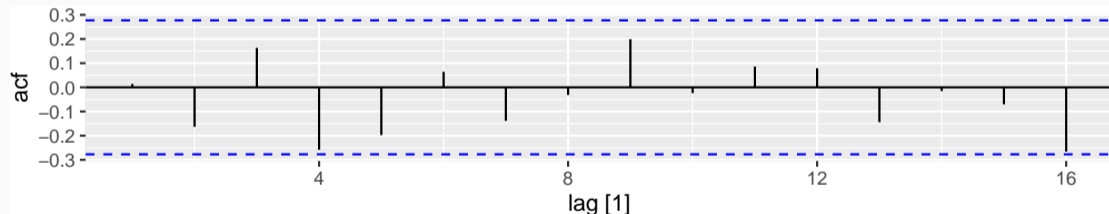




## Example: White noise

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.

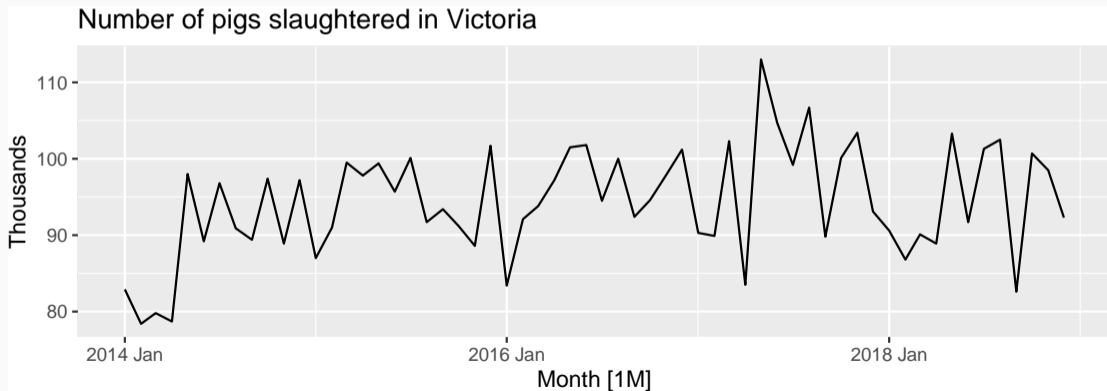
Sampling distribution of  $r_k$  for white noise data is asymptotically  $N(0, 1/T)$ .

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

- 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**.

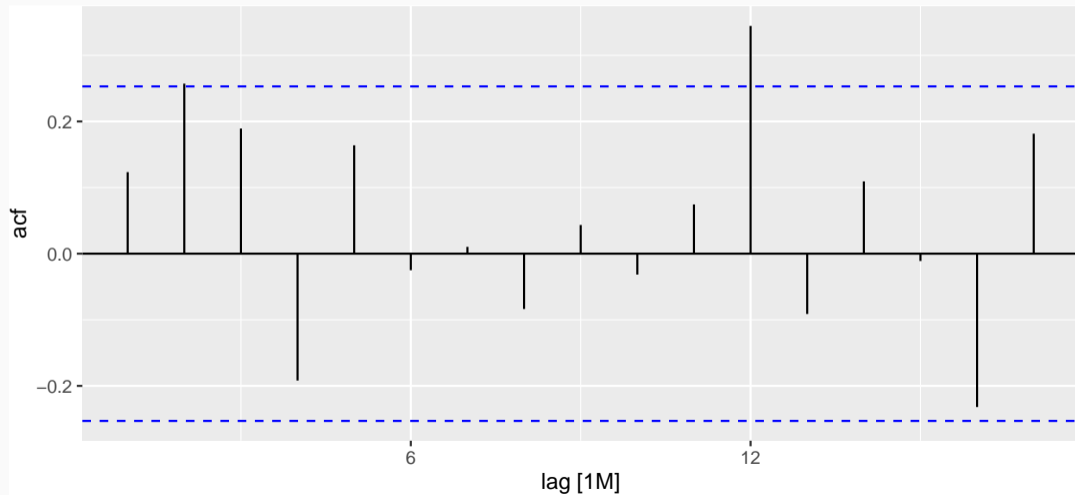
## Example: Pigs slaughtered

```
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**.



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?