Chapter 2: Time series graphics

Ankit Gupta

30/11/2023

The first thing to do in any data analysis task is to plot the data. Graphs enable many features of the data to be visualised, including patterns, unusual observations, changes over time, and relationships between variables. The features that are seen in plots of the data must then be incorporated, as much as possible, into the forecasting methods to be used. Just as the type of data determines what forecasting method to use, it also determines what graphs are appropriate. But before we produce graphs, we need to set up our time series in R.

tsibble objects

when we load *fpp3* library then some packages get also attached, we don't have to call separately. Also, some datasets are loaded after loading this package.

```
library(fpp3)
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.5 --

## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.3     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
global_economy
```

```
## # A tsibble: 15,150 x 9 [1Y]
## # Key:       Country [263]
##    Country     Code   Year         GDP Growth   CPI Imports Exports Population
##    <fct>       <fct> <dbl>       <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>
##  1 Afghanistan AFG    1960  537777811.     NA    NA    7.02    4.13    8996351
##  2 Afghanistan AFG    1961  548888896.     NA    NA    8.10    4.45    9166764
```

```
##  3 Afghanistan AFG    1962  546666678.     NA    NA    9.35    4.88    9345868
##  4 Afghanistan AFG    1963  751111191.     NA    NA   16.9     9.17    9533954
##  5 Afghanistan AFG    1964  800000044.     NA    NA   18.1     8.89    9731361
##  6 Afghanistan AFG    1965 1006666638.     NA    NA   21.4    11.3     9938414
##  7 Afghanistan AFG    1966 1399999967.     NA    NA   18.6     8.57   10152331
##  8 Afghanistan AFG    1967 1673333418.     NA    NA   14.2     6.77   10372630
##  9 Afghanistan AFG    1968 1373333367.     NA    NA   15.2     8.90   10604346
## 10 Afghanistan AFG    1969 1408888922.     NA    NA   15.0    10.1    10854428
## # i 15,140 more rows
```

Here, we have a tsibble object "global_economy" which has 15,150 rows and 6 columns and frequency of observation is [1Y] i.e. 1 year.

Here, key variable is "Country". Here we have 263 countries in this dataset and it means we have 263 separate time series.

Now, we have variables as "Index" which indexes the time series. "Country" variable determines the unique time series in the data set. Rest all other variables are called "measured variables" or "y variables" which tells about we wish to model.

(1) A tsibble allows storage and manipulation of multiple time series in R.

(2) It contains:

- An index (mandatory): time information about the observation
- Measured variable(s): numbers of interest
- Key variable(s): optional unique identifiers for each series

(3) It works with tidyverse functions. Example: Creation of tsibble object by own:

```
mydata <- tsibble(year = 2015:2019, y=c(123,39,78,52,110),index = year)
mydata
```

```
## # A tsibble: 5 x 2 [1Y]
##    year     y
##   <int> <dbl>
## 1  2015   123
## 2  2016    39
## 3  2017    78
## 4  2018    52
## 5  2019   110
```

In R, we use pipe symbols in 2 ways (in most ways both works in the same way) i.e. %>% and |>

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

Example:

We have a dataset "z" in which Month column is in string form () and so we have to convert it into tsibble object as:

```
#z |> mutate(Month=yearmonth(Month)) |> as_tsibble(index=Month)
```

Common time index variables can be created with these functions:

```
#Frequency --- Functions

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

Read acsv file and convert it into a tsibble object

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

Creating a Quarter variable:

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

Get rid of date variable

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

creating tsibble object

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

key must be there because it works like a primary key (unique combination of columns)

Example: Australian Pharmaseutical Benefits scheme dataset

```
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-payme~ A     Alimenta~ A01   STOMATOL~   18228 67877
##  2 1991 Aug Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   15327 57011
```

```
##  3 1991 Sep Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   14775 55020
##  4 1991 Oct Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   15380 57222
##  5 1991 Nov Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   14371 52120
##  6 1991 Dec Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   15028 54299
##  7 1992 Jan Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   11040 39753
##  8 1992 Feb Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   15165 54405
##  9 1992 Mar Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   16898 61108
## 10 1992 Apr Concessional Co-payme~ A     Alimenta~ A01   STOMATOL~   18141 65356
## # i 67,586 more rows
```

Here, ATC-1 and ATC-2 variable describe about ATC1 and ATC2.

Filtering some some from ATC2 column.

```
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-paym~ A     Alimenta~ A10   ANTIDIAB~   89733 2.09e6
##  2 1991 Aug Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   77101 1.80e6
##  3 1991 Sep Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   76255 1.78e6
##  4 1991 Oct Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   78681 1.85e6
##  5 1991 Nov Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   70554 1.69e6
##  6 1991 Dec Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   75814 1.84e6
##  7 1992 Jan Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   64186 1.56e6
##  8 1992 Feb Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   75899 1.73e6
##  9 1992 Mar Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   89445 2.05e6
## 10 1992 Apr Concessional Co-paym~ A     Alimenta~ A10   ANTIDIAB~   97315 2.23e6
## # i 806 more rows
```

Now we got 4 time series (2 times 2) because 4 unique combination of keys.

Now, we can use 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
```

```
##  9 1992 Mar Concessional Co-payments 2046102
## 10 1992 Apr Concessional Co-payments 2225977
## # i 806 more rows
```

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

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

```
## # A tsibble: 204 x 2 [1M]
##       Month  TotalC
##       <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
##  9 1992 Mar 2985811
## 10 1992 Apr 3204780
## # i 194 more rows
```

We can use mutate() function to create new variables. Here, we create a new variable for total dollar per million by dividing total cost in dollar by 1e6 i.e. 1 million.

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

```
## # A tsibble: 204 x 3 [1M]
##       Month  TotalC  Cost
##       <mth>   <dbl> <dbl>
##  1 1991 Jul 3526591  3.53
##  2 1991 Aug 3180891  3.18
##  3 1991 Sep 3252221  3.25
##  4 1991 Oct 3611003  3.61
##  5 1991 Nov 3565869  3.57
##  6 1991 Dec 4306371  4.31
##  7 1992 Jan 5088335  5.09
##  8 1992 Feb 2814520  2.81
##  9 1992 Mar 2985811  2.99
## 10 1992 Apr 3204780  3.20
## # i 194 more rows
```

Now, assign this filtered data to a10 variable.

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

```
## # A tsibble: 204 x 3 [1M]
##       Month  TotalC  Cost
##       <mth>   <dbl> <dbl>
##  1 1991 Jul 3526591  3.53
##  2 1991 Aug 3180891  3.18
##  3 1991 Sep 3252221  3.25
##  4 1991 Oct 3611003  3.61
##  5 1991 Nov 3565869  3.57
##  6 1991 Dec 4306371  4.31
##  7 1992 Jan 5088335  5.09
##  8 1992 Feb 2814520  2.81
##  9 1992 Mar 2985811  2.99
## 10 1992 Apr 3204780  3.20
## # i 194 more rows
```

A time series can be thought of as a list of numbers (the measurements), along with some information about what times those numbers were recorded (the index). This information can be stored as a tsibble object in R. tsibble objects extend tidy data frames (tibble objects) by introducing temporal structure.

A *tsibble* also allows multiple time series to be stored in a single object. Suppose you are interested in a dataset containing the fastest running times for women's and men's track races at the Olympics, from 100m to 10000m:

```
olympic_running
```

```
## # A tsibble: 312 x 4 [4Y]
## # Key:       Length, Sex [14]
##     Year Length Sex    Time
##    <int>  <int> <chr> <dbl>
##  1  1896    100 men    12
##  2  1900    100 men    11
##  3  1904    100 men    11
##  4  1908    100 men    10.8
##  5  1912    100 men    10.8
##  6  1916    100 men    NA
##  7  1920    100 men    10.8
##  8  1924    100 men    10.6
##  9  1928    100 men    10.8
## 10  1932    100 men    10.3
## # i 302 more rows
```

The summary above shows that this is a tsibble object, which contains 312 rows and 4 columns. Alongside this, "[4Y]" informs us that the interval of these observations is every four years. Below this is the key structure, which informs us that there are 14 separate time series in the tsibble. A preview of the first 10

observations is also shown, in which we can see a missing value occurs in 1916. This is because the Olympics were not held during World War I.

The distinct() function can be used to show the categories of each variable or even combinations of variables:

```
olympic_running |> distinct(Sex)
```

```
## # A tibble: 2 x 1
##   Sex
##   <chr>
## 1 men
## 2 women
```

We can use dplyr functions such as mutate(), filter(), select() and summarise() to work with tsibble objects. To illustrate these, we will use the PBS tsibble containing sales data on pharmaceutical products in Australia.

Almost all of the data used in this book is already stored as tsibble objects. But most data lives in databases, MS-Excel files or csv files, before it is imported into R. So often the first step in creating a tsibble is to read in the data, and then identify the index and key variables.

For example, suppose we have the following quarterly data stored in a csv file (only the first 10 rows are shown). This data set provides information on the size of the prison population in Australia, disaggregated by state, gender, legal status and indigenous status. (Here, ATSI stands for Aboriginal or Torres Strait Islander.)

We can read it into R, and create a tsibble object, by simply identifying which column contains the time index, and which columns are keys. The remaining columns are values — there can be many value columns, although in this case there is only one (Count). The original csv file stored the dates as individual days, although the data is actually quarterly, so we need to convert the Date variable to quarters.

For a tsibble to be valid, it requires a unique index for each combination of keys. The tsibble() or as_tsibble() function will return an error if this is not true.

The seasonal period

Some graphics and some models will use the seasonal period of the data. The seasonal period is the number of observations before the seasonal pattern repeats. In most cases, this will be automatically detected using the time index variable.

```
#Data        Minute Hour    Day   Week    Year

#Quarters                                    4
#Months                                     12
#Weeks                                      52
#Days                                 7      365.25
#Hours                      24    168       8766
#Minutes                60  1440  10080    525960
#Seconds      60      3600  86400  604800  31557600
```

For quarterly, monthly and weekly data, there is only one seasonal period — the number of observations within each year. Actually, there are not
52 weeks in a year, but $365.25/7 = 52.18$ on average, allowing for a leap year every fourth year. Approximating seasonal periods to integers can be useful as many seasonal terms in models only support integer seasonal periods.

If the data is observed more than once per week, then there is often more than one seasonal pattern in the data. For example, data with daily observations might have weekly (period =7) or annual (period =365.25) seasonal patterns. Similarly, data that are observed every minute might have hourly (period =60), daily (period = 24×60=1440), weekly (period =24×60×7=10080) and annual seasonality (period =24×60×365.25=525960).

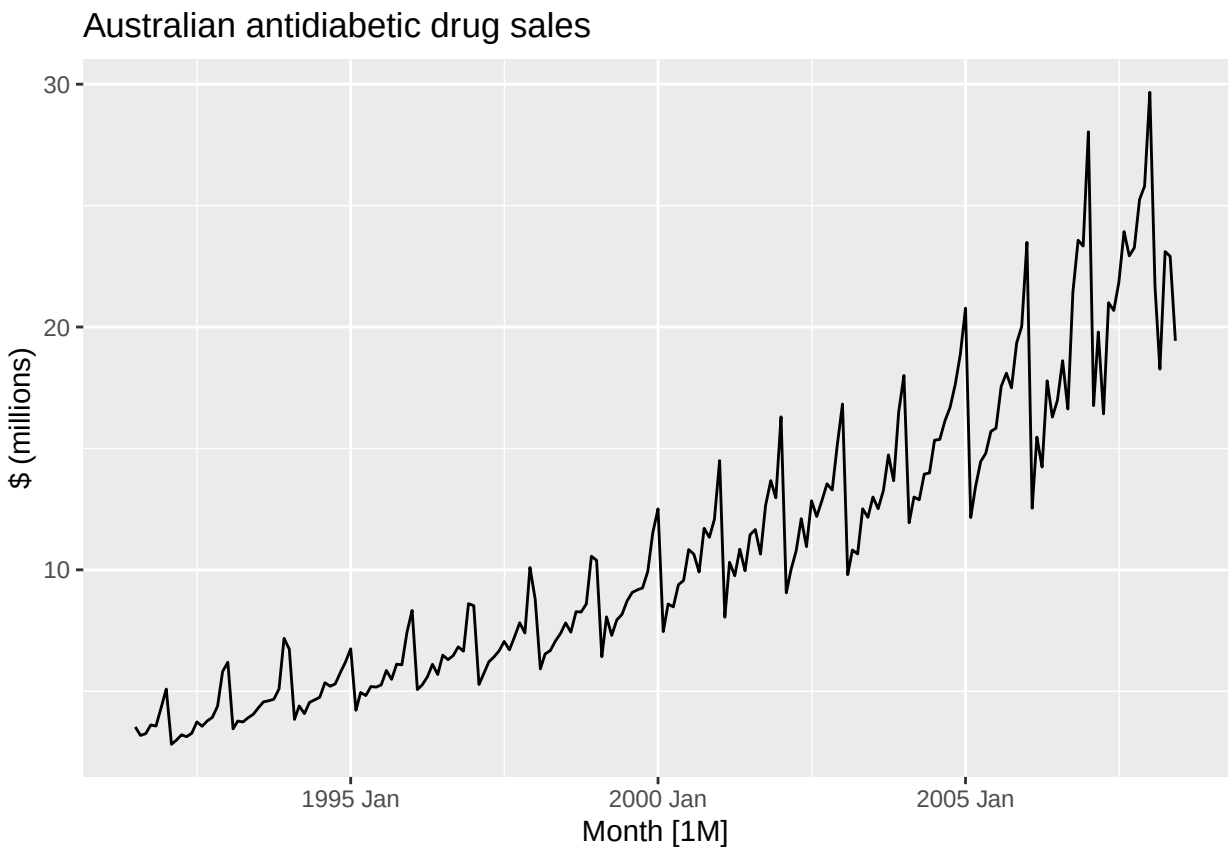
More complicated (and unusual) seasonal patterns can be specified using the period() function in the lubridate package.

2.2 Time plots

Plots allow us to identify

- Patterns
- Unusual Observations
- Changes over time
- Relationship variables

```
a10 |>
  autoplot(Cost) +
  labs(y="$ (millions)",title = "Australian antidiabetic drug sales")
```



Example:

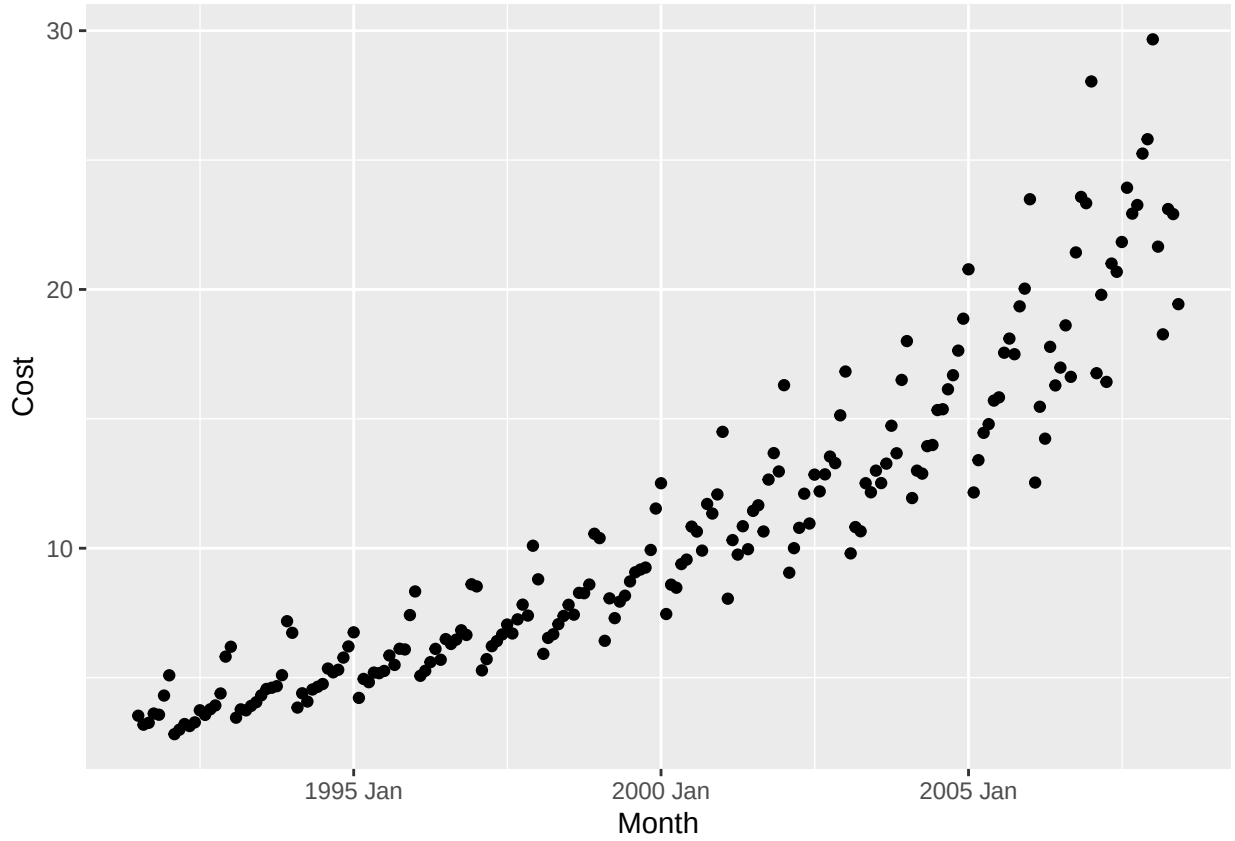
Here, *a10* contains the total cost of 18 drugs per month.

autoplot() is a smart in-built function under *ggplot*. Since we are passing a tsibble object as an argument in this function, so it identifies that it is an time series object and so generate the time plot.If we don't pass any argument then it picks the first variable it sees and here it will be the "totalC".

Also note that here points/observations are joint by lines

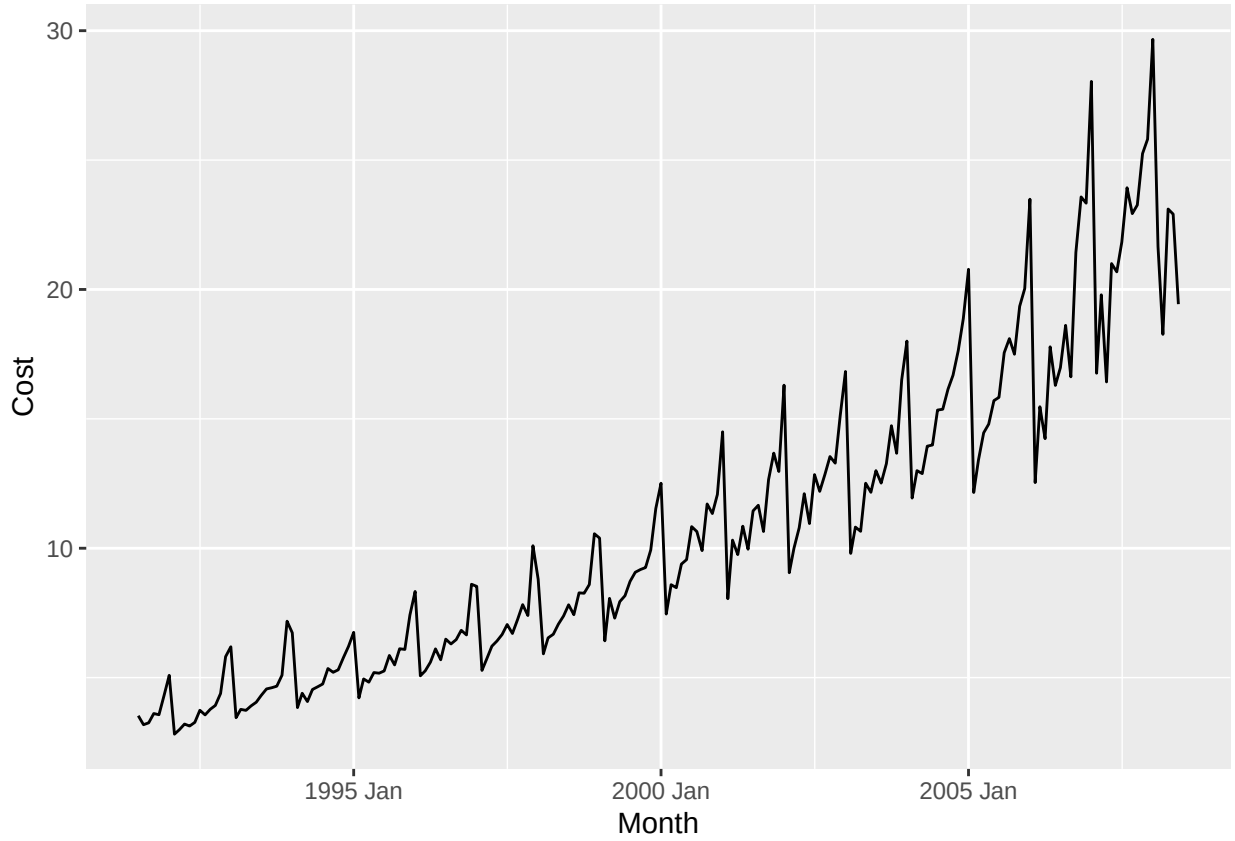
If we only have to plot points, we can write code as:

```
a10 |>
  ggplot(aes(x=Month,y=Cost))+
  geom_point()
```



But it is not useful because we have to see the patterns in the time series. So, If we have to plot line between the observations then we write it as:

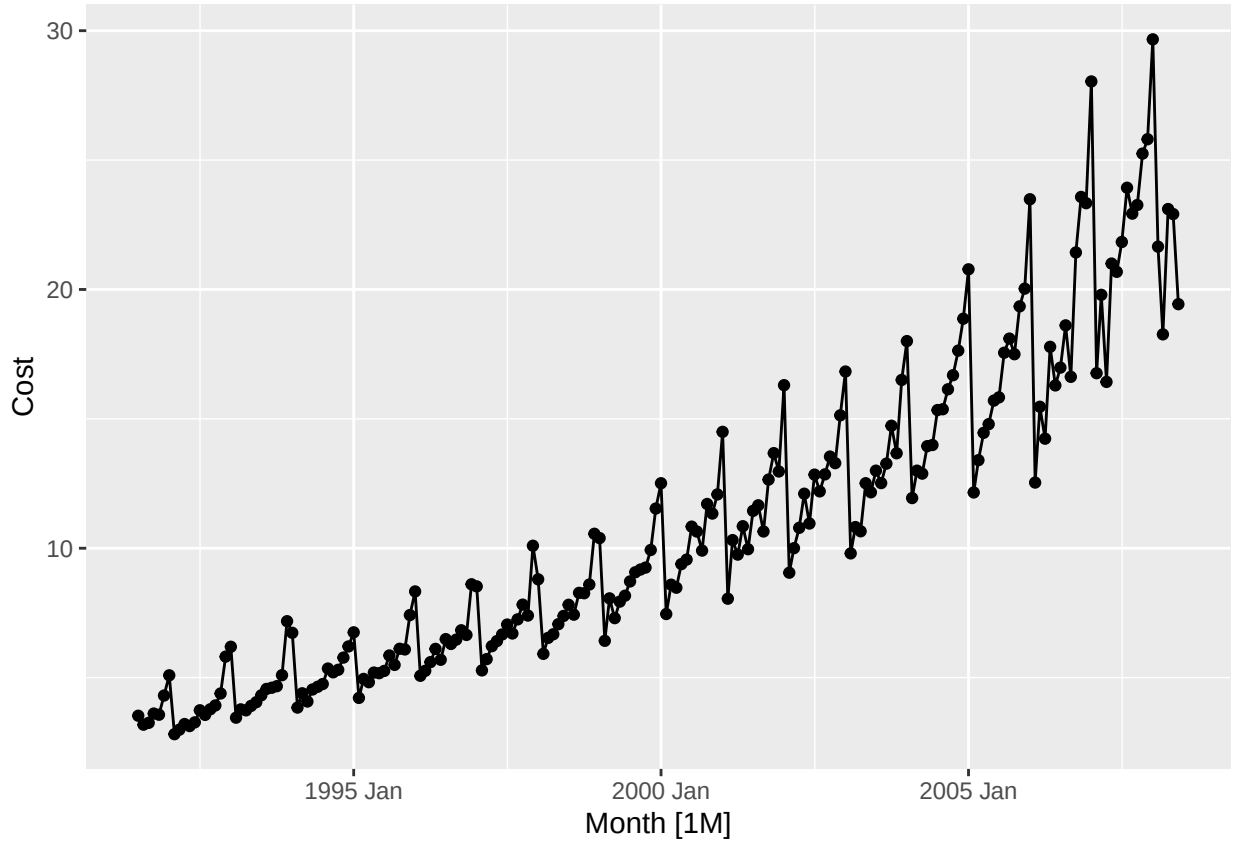
```
a10 |>
  ggplot(aes(x=Month,y=Cost))+
  geom_line()
```



Same we can do with autoplot.

If we have to sraw both lines and points then we do:

```
a10|> autoplot(Cost)+geom_point()
```



This is useful if we have to observe the spikes at particular time.

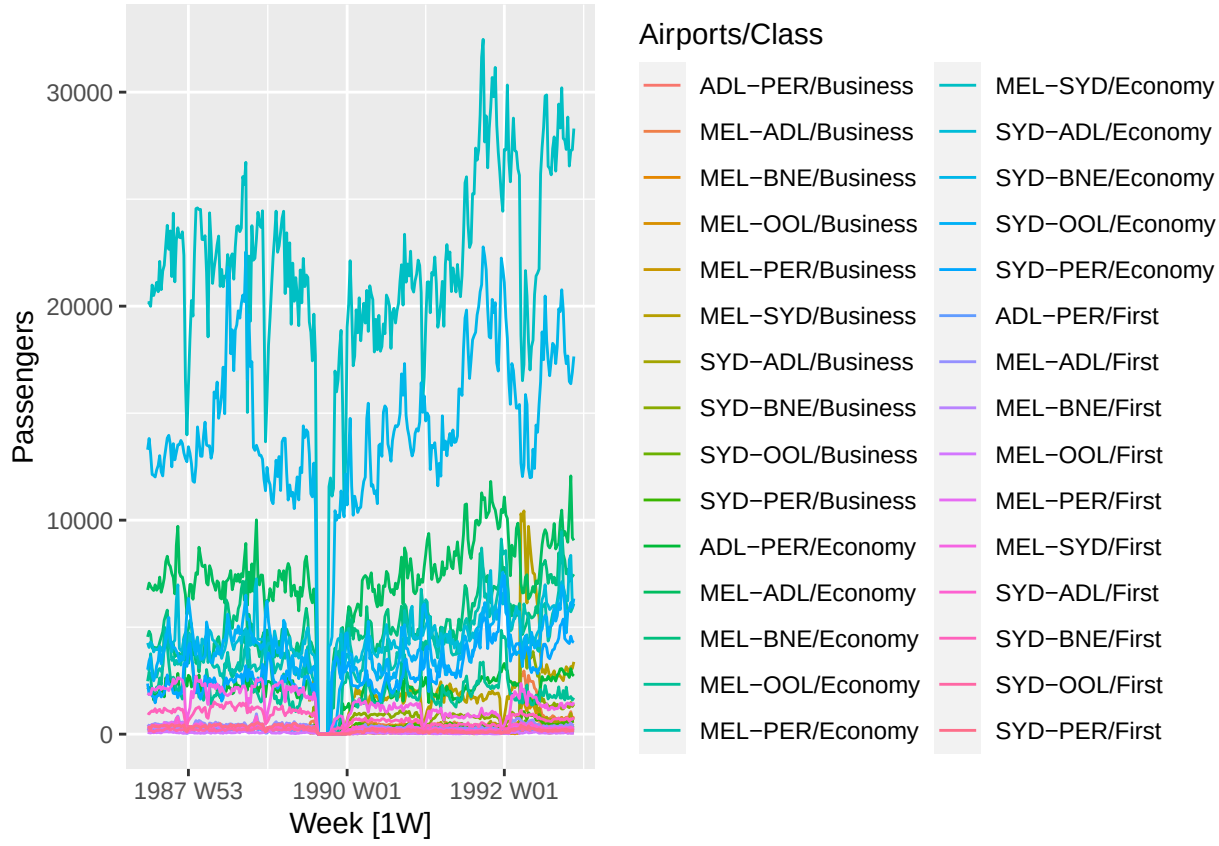
Observed Features

Here we can see the spikes at regular interval which shows the seasonal pattern which happens at some specific month of each year. Each spike is followed by a trough

Also we can see the trend in this data that is it is going upward. Maybe it is a non-linear trend. Also the difference between spikes and trough is increasing as levels of data is increasing. This is called a multiplicative effect because seasonality amplitude increases as the level of data increases

Example:

```
ansett |> autoplot(Passengers)
```




```
ansett
```

```
## # A tsibble: 7,407 x 4 [1W]
## # Key:       Airports, Class [30]
##         Week Airports Class    Passengers
##       <week> <chr>    <chr>         <dbl>
##  1 1989 W28 ADL-PER  Business        193
##  2 1989 W29 ADL-PER  Business        254
##  3 1989 W30 ADL-PER  Business        185
##  4 1989 W31 ADL-PER  Business        254
##  5 1989 W32 ADL-PER  Business        191
##  6 1989 W33 ADL-PER  Business        136
##  7 1989 W34 ADL-PER  Business          0
##  8 1989 W35 ADL-PER  Business          0
##  9 1989 W36 ADL-PER  Business          0
## 10 1989 W37 ADL-PER  Business          0
## # i 7,397 more rows
```

Here, we have 30 time series which we are plotting using autoplot().

```
ansett|>distinct(Class)
```

```
## # A tibble: 3 x 1
##   Class
##   <chr>
```

```
## 1 Business
## 2 Economy
## 3 First
```

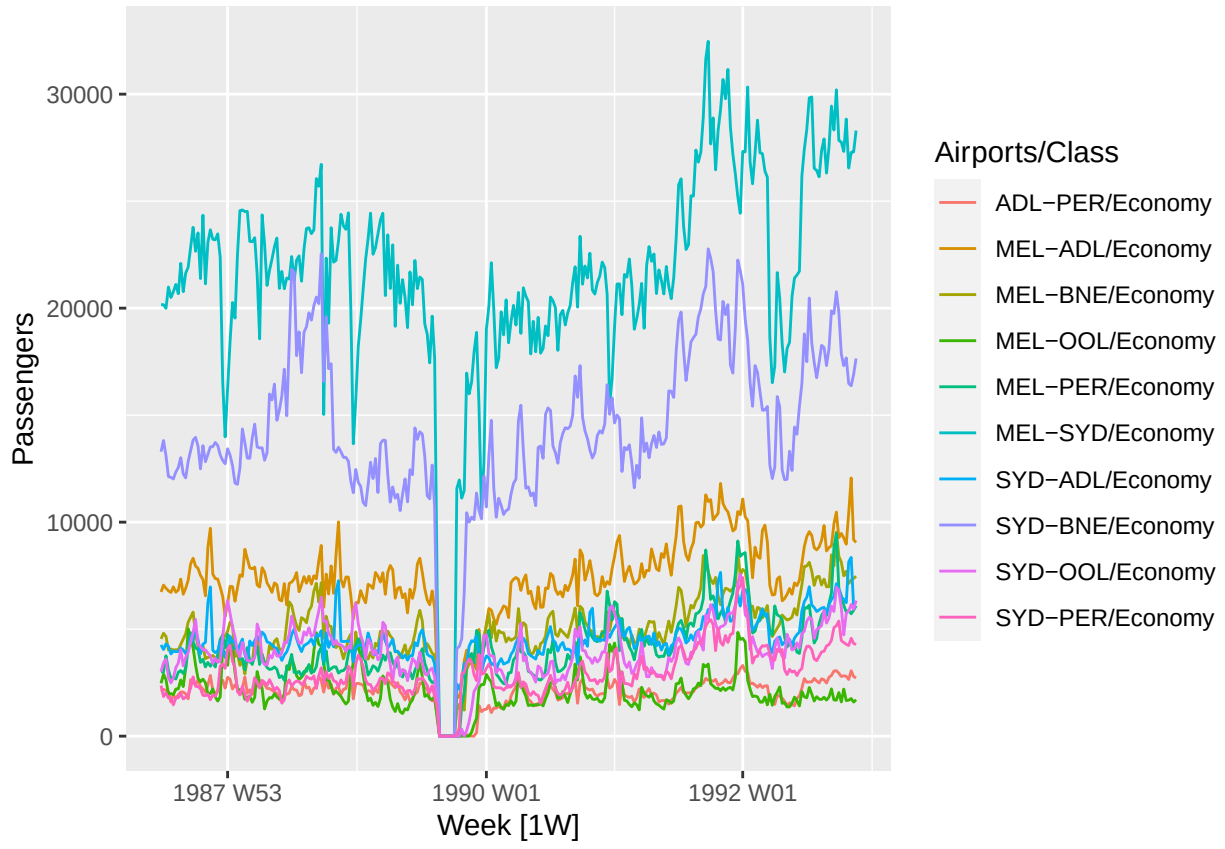
```
ansett|>distinct(Airports)
```

```
## # A tibble: 10 x 1
##    Airports
##    <chr>
##  1 ADL-PER
##  2 MEL-ADL
##  3 MEL-BNE
##  4 MEL-OOL
##  5 MEL-PER
##  6 MEL-SYD
##  7 SYD-ADL
##  8 SYD-BNE
##  9 SYD-OOL
## 10 SYD-PER
```

So, total 3*10=30 times series are possible.

```
ansett |> filter(Class=="Economy")|>autoplot()
```

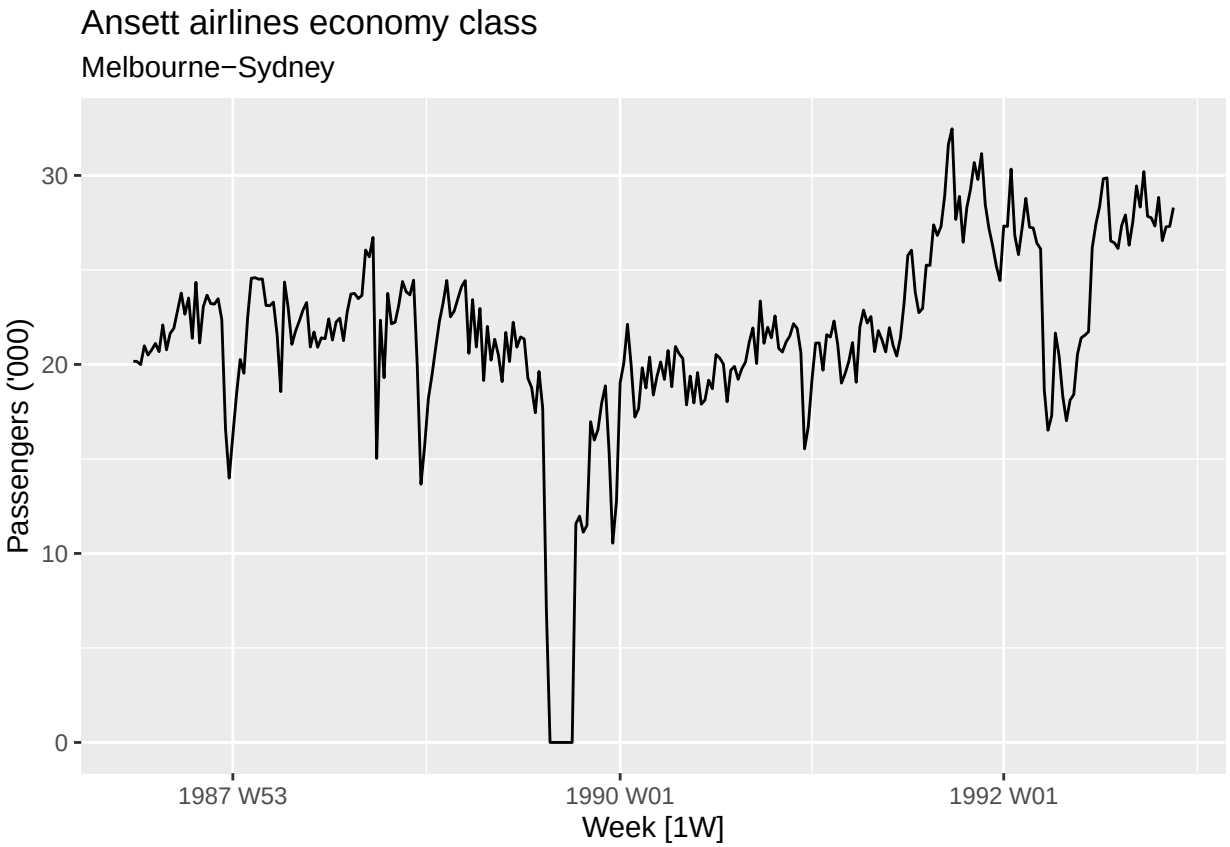
```
## Plot variable not specified, automatically selected `.vars = Passengers`
```



```
melsyd_economy <- ansett |> filter(Airports=='MEL-SYD') |> select(-Airports)
melsyd_economy
```

```
## # A tsibble: 740 x 3 [1W]
## # Key:       Class [3]
##         Week Class    Passengers
##       <week> <chr>         <dbl>
##  1 1989 W28 Business       1524
##  2 1989 W29 Business       2212
##  3 1989 W30 Business       1777
##  4 1989 W31 Business       2552
##  5 1989 W32 Business       1889
##  6 1989 W33 Business        851
##  7 1989 W34 Business          0
##  8 1989 W35 Business          0
##  9 1989 W36 Business          0
## 10 1989 W37 Business          0
## # i 730 more rows
```

```
melsyd_economy |> filter(Class=='Economy') |> mutate(Passengers=Passengers/1000)|>
  autoplot(Passengers)+
  labs(title="Ansett airlines economy class",subtitle="Melbourne-Sydney",y="Passengers ('000)")
```

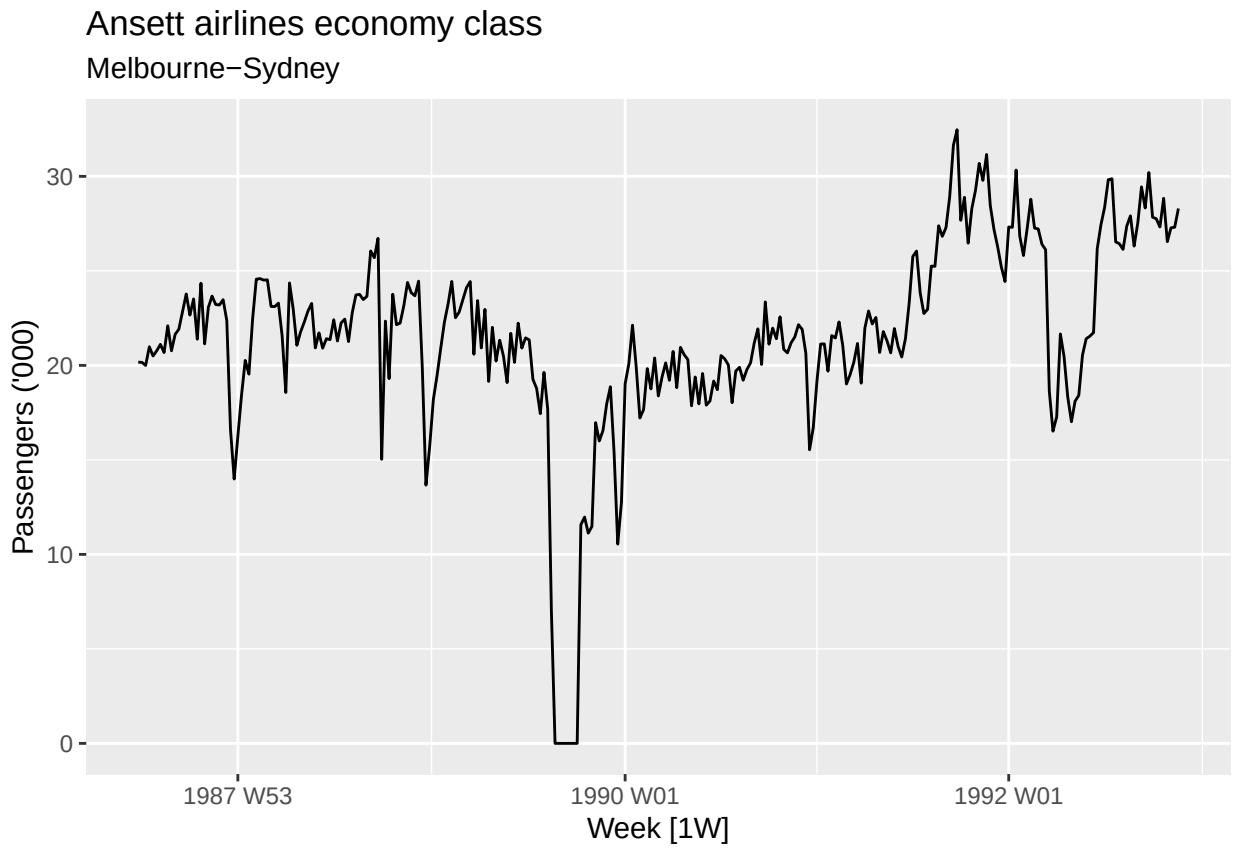


Theory

For time series data, the obvious graph to start with is a time plot. That is, the observations are plotted against the time of observation, with consecutive observations joined by straight lines.

We will use the autoplot() command frequently. It automatically produces an appropriate plot of whatever you pass to it in the first argument.

```
melsyd_economy <- ansett |>
  filter(Airports == "MEL-SYD", Class == "Economy") |>
  mutate(Passengers = Passengers/1000)
autoplot(melsyd_economy, Passengers) +
  labs(title = "Ansett airlines economy class",
       subtitle = "Melbourne-Sydney",
       y = "Passengers ('000)")
```



Features:

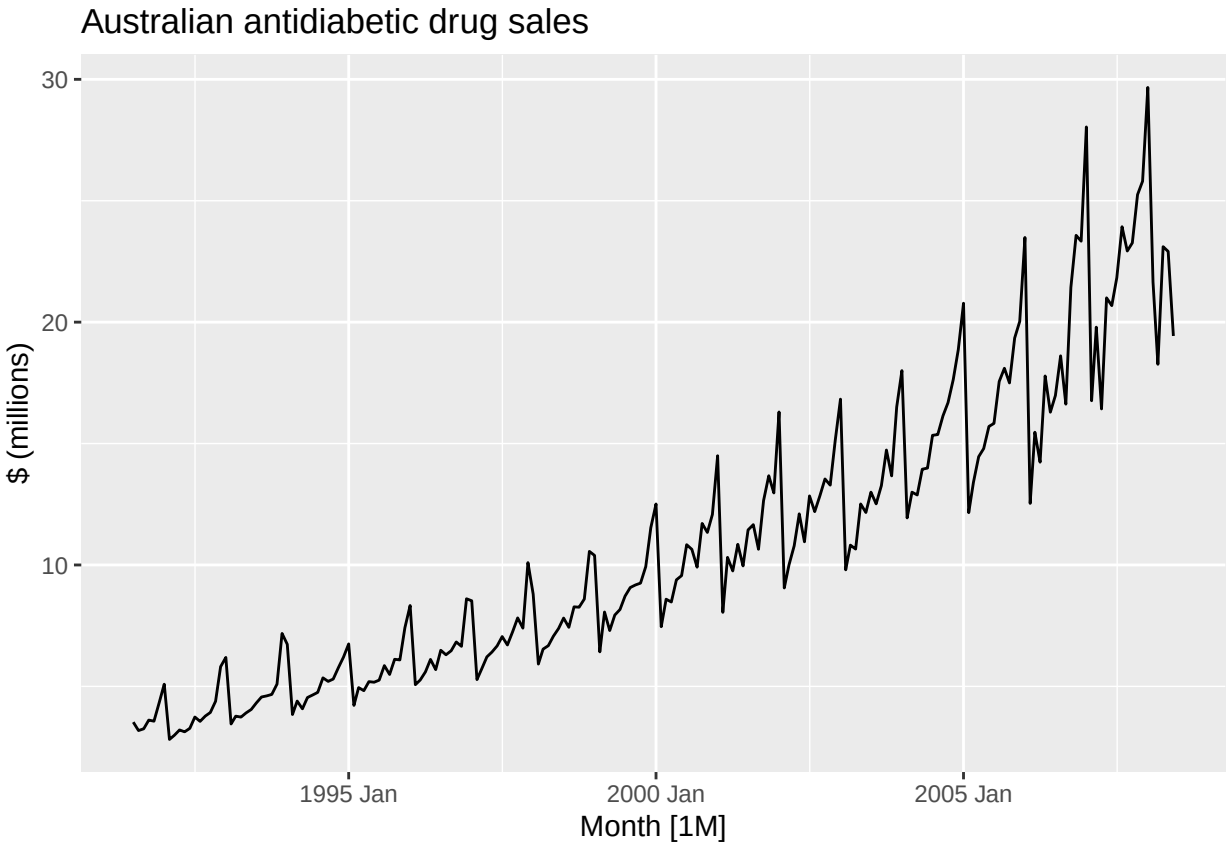
- There was a period in 1989 when no passengers were carried — this was due to an industrial dispute.
- There was a period of reduced load in 1992. This was due to a trial in which some economy class seats were replaced by business class seats.
- A large increase in passenger load occurred in the second half of 1991. There are some large dips in load around the start of each year. These are due to holiday effects.

- There is a long-term fluctuation in the level of the series which increases during 1987, decreases in 1989, and increases again through 1990 and 1991.

Any model will need to take all these features into account in order to effectively forecast the passenger load into the future.

Example:

```
autoplot(a10, Cost) +
  labs(y = "$ (millions)",
       title = "Australian antidiabetic drug sales")
```



Features: Here, there is a clear and increasing trend. There is also a strong seasonal pattern that increases in size as the level of the series increases. The sudden drop at the start of each year is caused by a government subsidisation scheme that makes it cost-effective for patients to stockpile drugs at the end of the calendar year. Any forecasts of this series would need to capture the seasonal pattern, and the fact that the trend is changing slowly.

2.3 Time series patterns

Trend

Pattern increase when there is a long-term increase or decrease in the data

Seasonal

Pattern increase when a series is influenced by seasonal factors (e.g. the quarter of the year, the month, or day of the week). Example in Christmas, month factor is involved and for electricity, day factor is involved like in offices, electricity consumption is low at weekends.

cyclic

Pattern increase when data exhibits rises and falls that are not of fixed period (minimum duration of these cycles are usually of 2 years). Example, business cycle is of 2 to 7 years.

A season is something that repeats at regular interval like every year or every summer quarter and winter quarter like ice cream consumption in summer quarter is different than the winter quarter. Retail sales in Christmas increase year after year after year. Ice cream consumption on weekends is higher than weekdays.

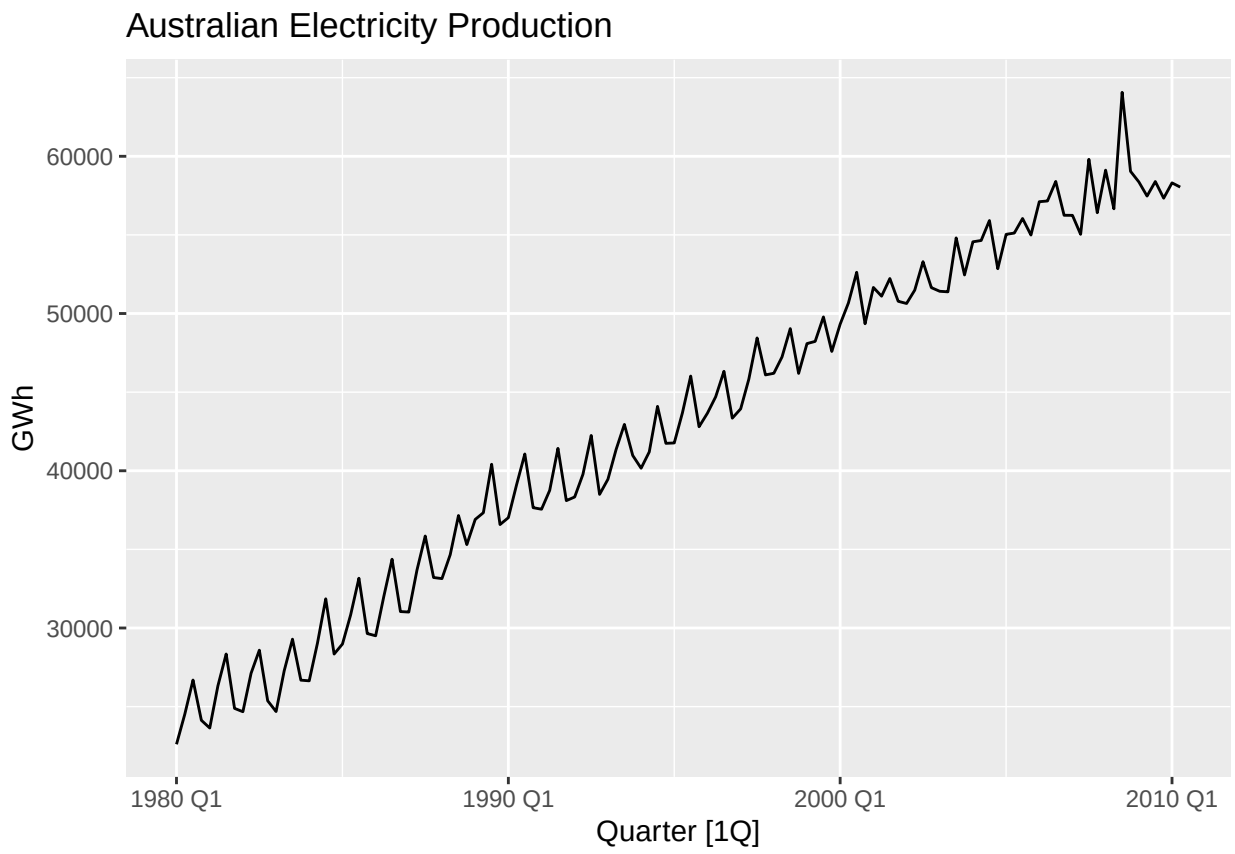
Cycles don't have these regular occurrence.

Difference between seasonal and cyclic patterns

- Seasonal Patterns are of constant length whereas cyclic patterns are of variable length.
- Average length of the seasonal pattern(like 12 months,4 quarters etc) is longer than cyclic pattern (2 to 7 years).
- Magnitude of cycle more variable than magnitude of seasonal pattern.

Example 1

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



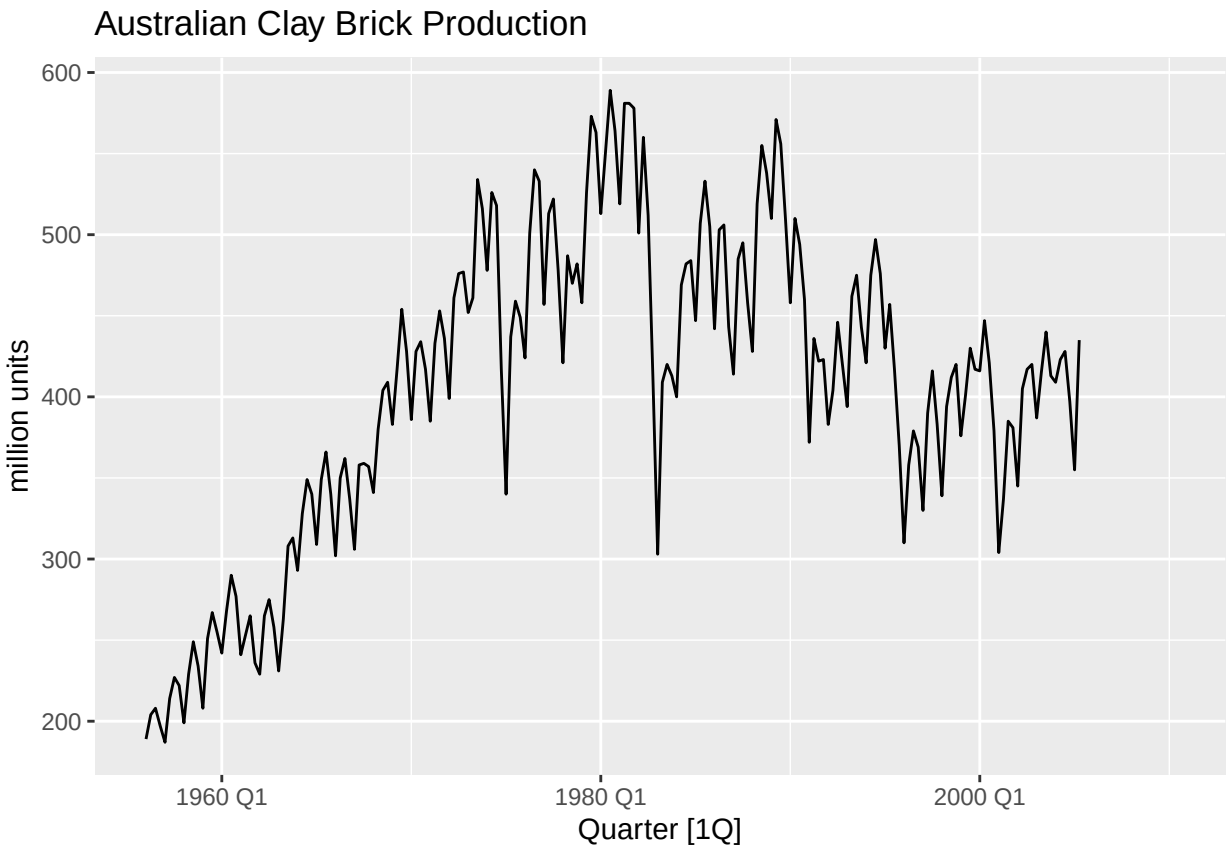
Features

- Most obvious feature is we have a trend here i.e. a long term increase in the data
- We have a seasonal component i.e. regular peaks at every 4 observations(points)
- This peak changes through the time series with electricity demand/production/consumption as air conditioners enter our life.

Example 2

```
aus_production |>
  autoplot(Bricks) +
  labs(y=' million units ', title = "Australian Clay Brick Production")
```

```
## Warning: Removed 20 rows containing missing values (`geom_line()`).
```

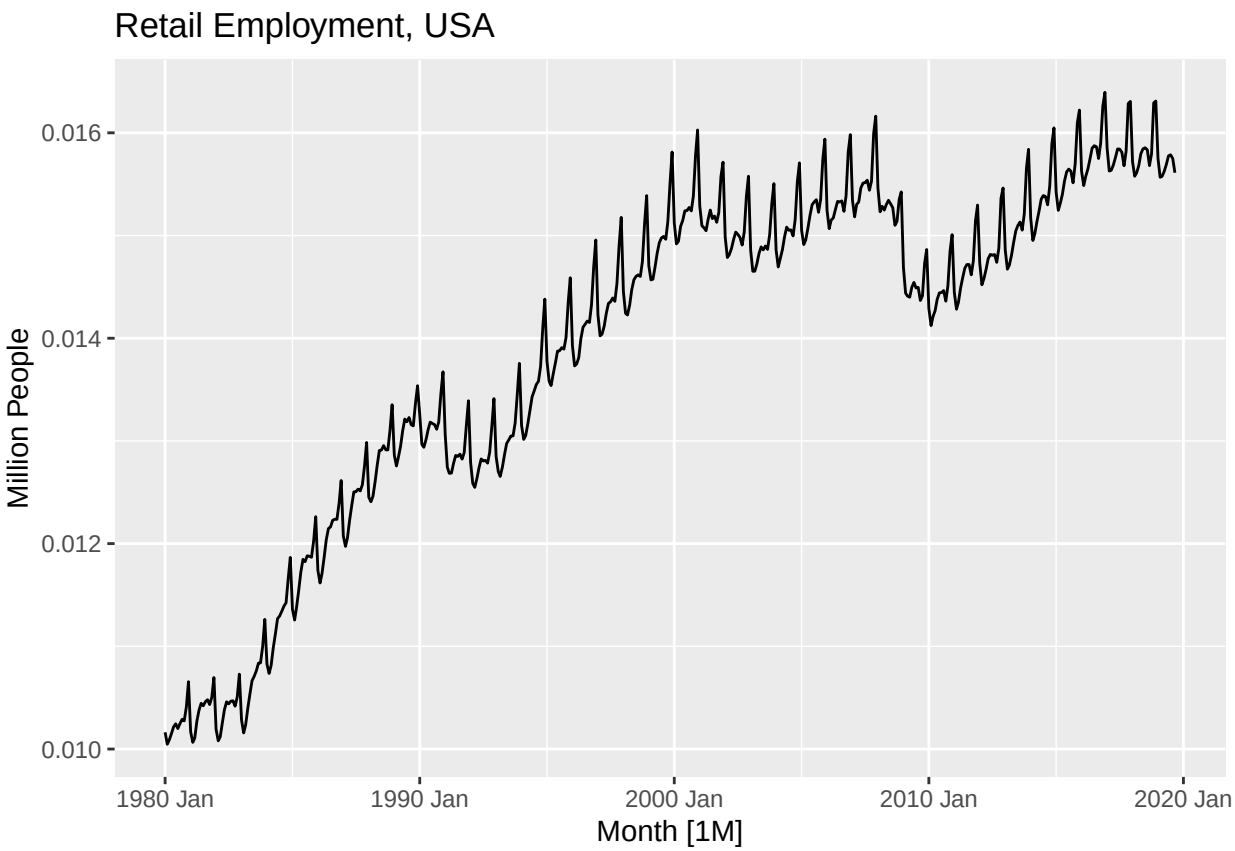


Features

- For this quarterly data, there is a sharp increase from mid 1950 to mid 1960. Lots of bricks produced and trend goes upwards.
- Then we have a big bust i.e. a recession or a couple of recessions in mid 1970s and early 1980.
- After early 1980, there is a increase in bust and then decrease and it goes on cyclically
- More bricks are needed in summer because it's drier and people build more houses. So summer quarter has spikes.

Example 3

```
us_employment |>
  filter(Title=="Retail Trade",year(Month)>=1980) |>
  autoplot(Employed/1e6)+
  labs(y="Million People", title='Retail Employment, USA')
```



Features

- Overall we see an increasing trend
- Strong Seasonal Patterns like spikes in Jun, July ..
- Cyclic pattern exist because increase then decrease

Example 4

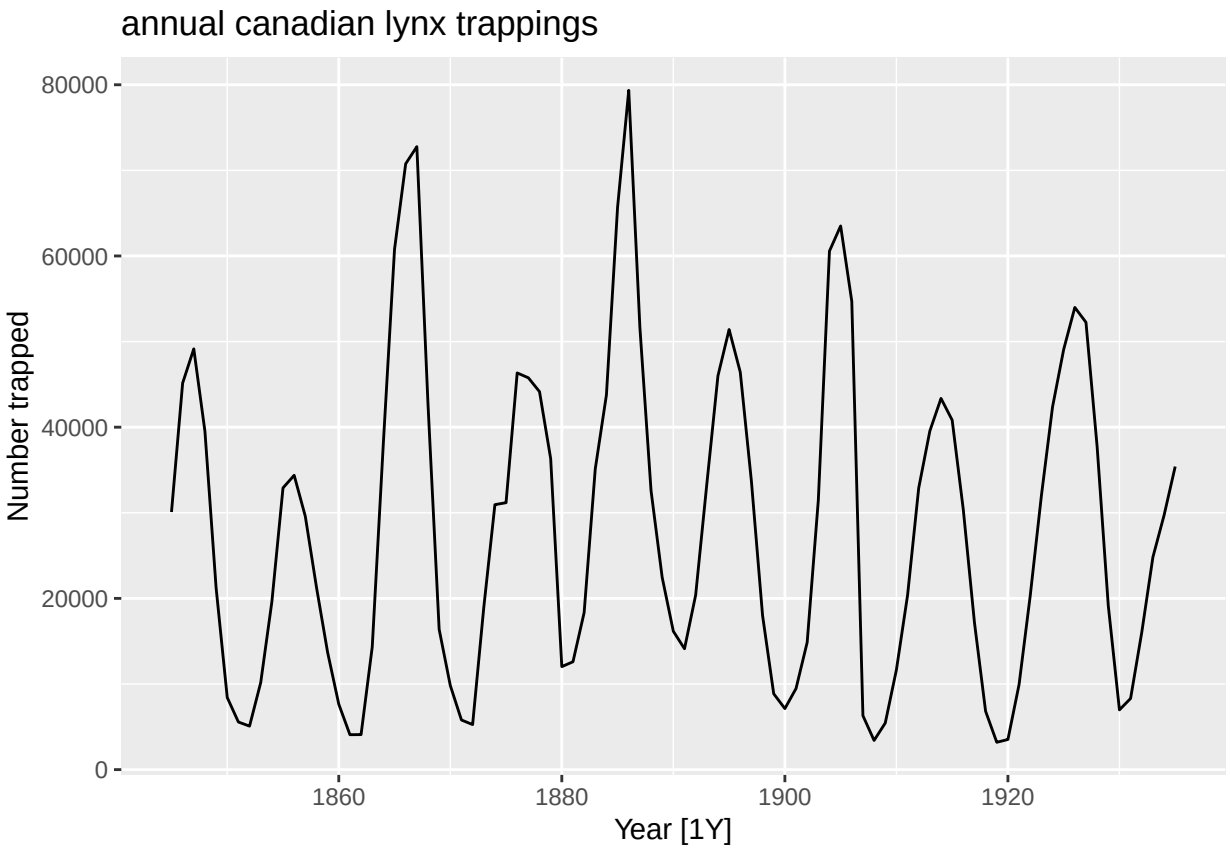
```
gafa_stock |> filter(Symbol=='AMZN', year(Date)>=2018) |>
  autoplot(Close) +
  labs(y="$US",title="Amazon closing stock price")
```

Features

- Increase and decrease – Random walk is present

Example 5

```
pelt |>
  autoplot(Lynx)+
  labs(y='Number trapped', title="annual canadian lynx trappings")
```



Features

- This data is annual
- It looks the pattern is seasonal but it is seasonal, it is cyclic.

The timing of peaks and troughs is predictable with seasonal data like we know when summer comes , we know when christmas comes or ice cream consumption make spikes in summer..but it is unpredictable in long term with cycle data because we don't know when recession is going to come.

Theory

Trend

A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend as "changing direction", when it might go from an increasing trend to a decreasing trend. There is a trend in the antidiabetic drug sales data shown in Figure 2.2.

Seasonal

A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known period. The monthly sales of antidiabetic drugs (Figure 2.2) shows seasonality which is induced partly by the change in the cost of the drugs at the end of the calendar year.

Cyclic

A cycle occurs when the data exhibit rises and falls that are not of a fixed frequency. These fluctuations are usually due to economic conditions, and are often related to the "business cycle". The duration of these fluctuations is usually at least 2 years.

Many people confuse cyclic behaviour with seasonal behaviour, but they are really quite different. If the fluctuations are not of a fixed frequency then they are cyclic; if the frequency is unchanging and associated with some aspect of the calendar, then the pattern is seasonal. In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitudes of cycles tend to be more variable than the magnitudes of seasonal patterns.

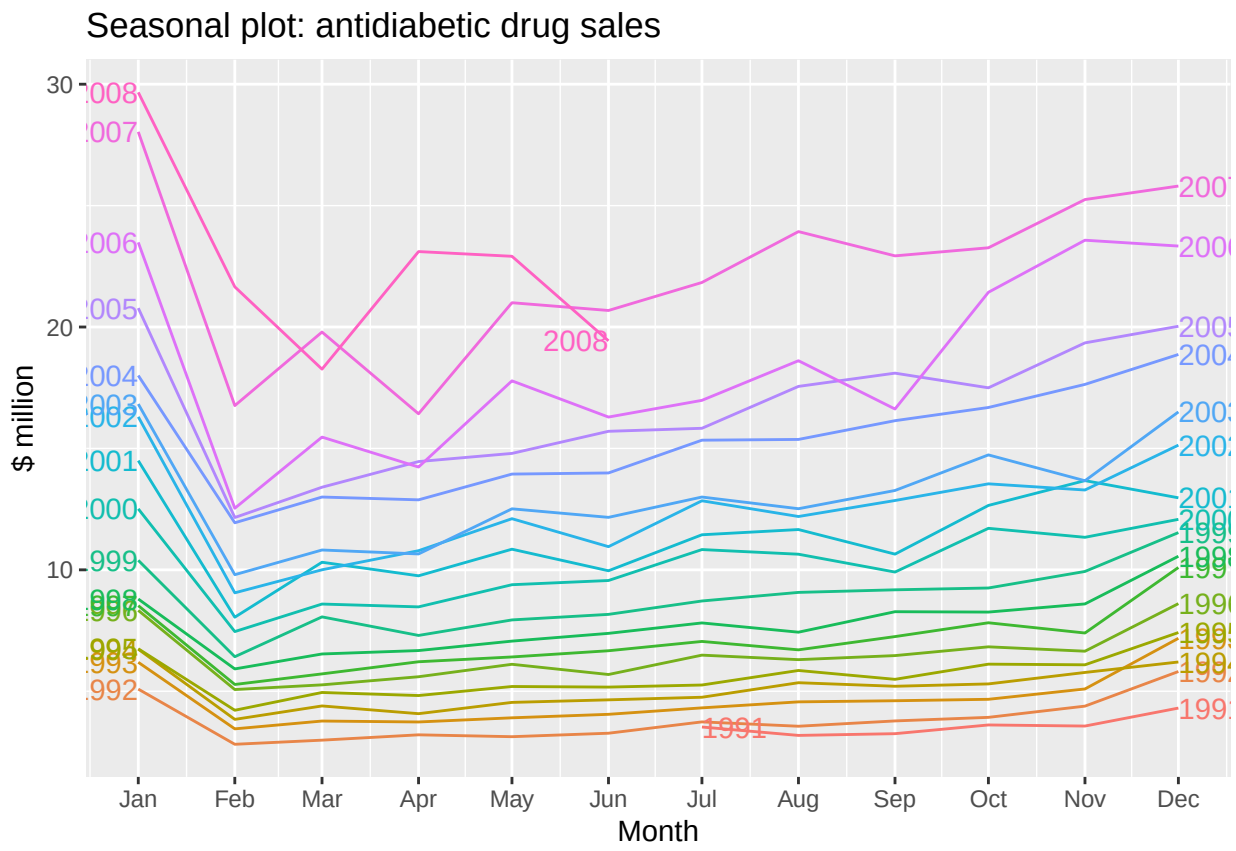
Many time series include trend, cycles and seasonality. When choosing a forecasting method, we will first need to identify the time series patterns in the data, and then choose a method that is able to capture the patterns properly.

2.4 Seasonal plots

A seasonal plot is a plot where the observations are plotted against the season at which they're observed

Example

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



Here the season is "Month". Here years are plotted on top of each other. Basically data wraps around per year. There are couple of features which we can identify by looking at this seasonal plot:

Features

- Here, we can see that for each year spike is at January month.
- There is a drop in feb but there is an increase between feb and mar because feb has 28 days and mar has 31 days so generally we see little difference when number of days are changing between months
- Also we can see the years in the plot are in in ascending order because it is trending data.

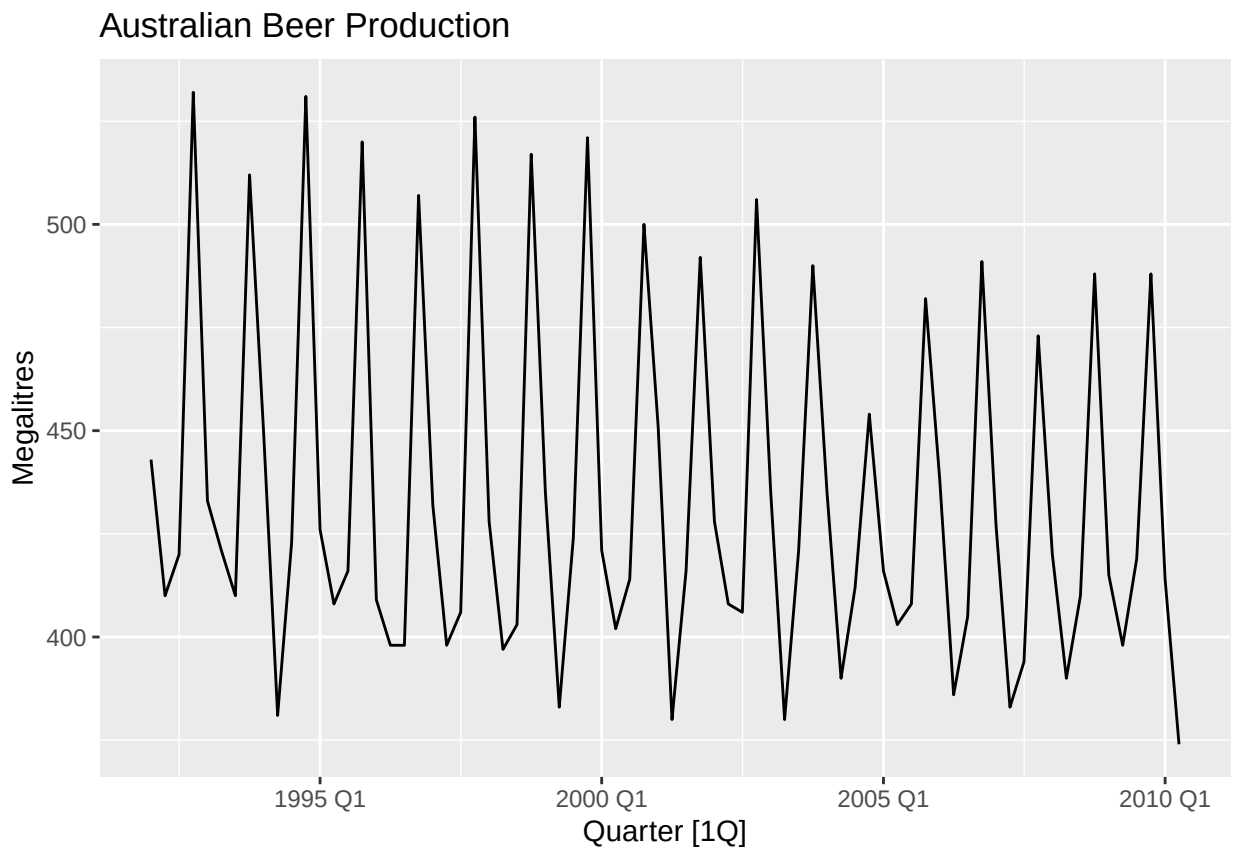
These observations can be identified by time plots as well but what we can't identify the unusual observations like Mar-2008 observation is less than then the feb-2008 observation and not adhering to the pattern that feb to march observations increases for each year. Also June-2008 observation is less than the June-2007 observation and much lower than the may-2008 observation.

So,

- Data plotted against the individual "seasons" in which the data were observed.(In this case, season is a "Month")
- Something like a time plot except that the data from each 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()

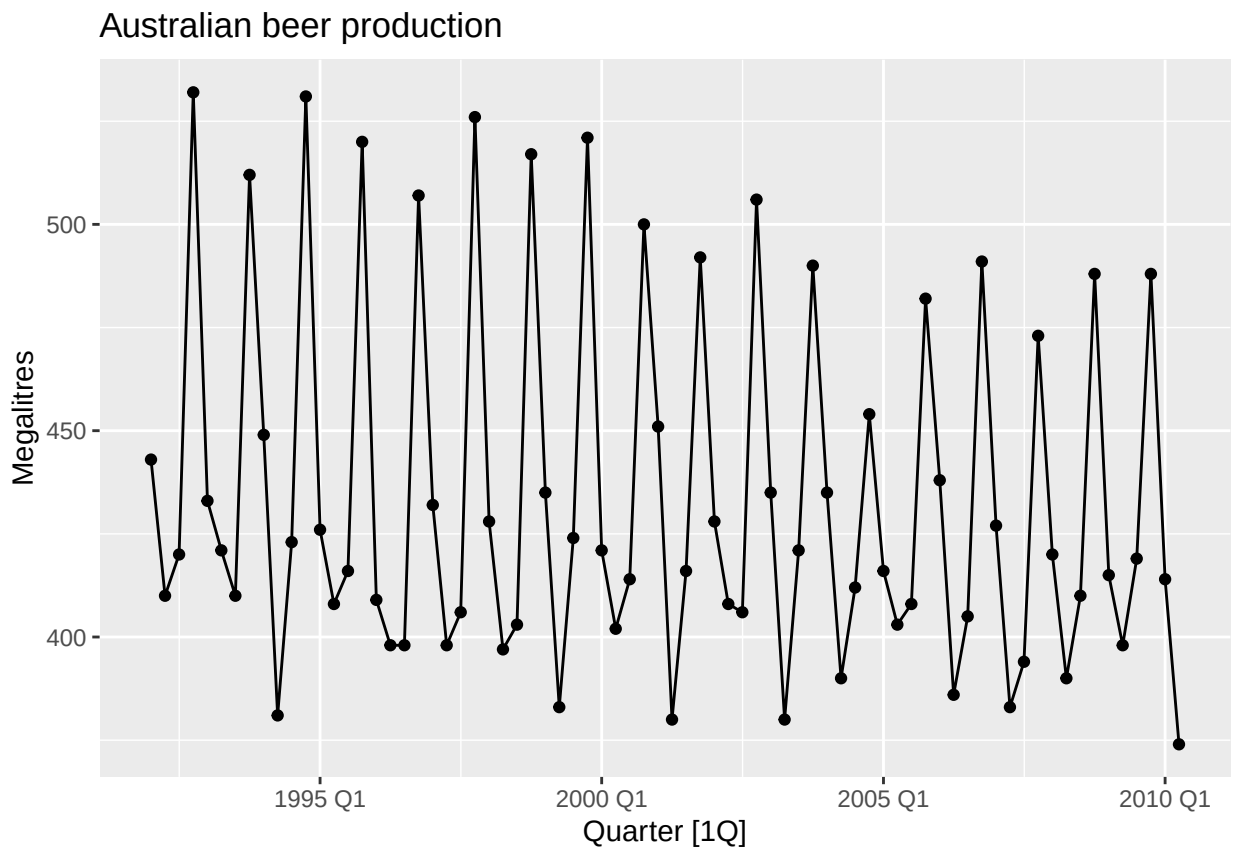
Example

```
beer <- aus_production |> select(Quarter,Beer) |> filter(year(Quarter)>=1992)
beer |> autoplot(Beer)+
  labs(title = "Australian Beer Production",y="Megalitres")
```



Observing this time plot which has highly seasonal data. We can see the spikes by drawing observation points as:

```
beer |> autoplot(Beer) + geom_point() +
  labs(title = "Australian beer production", y= "Megalitres")
```

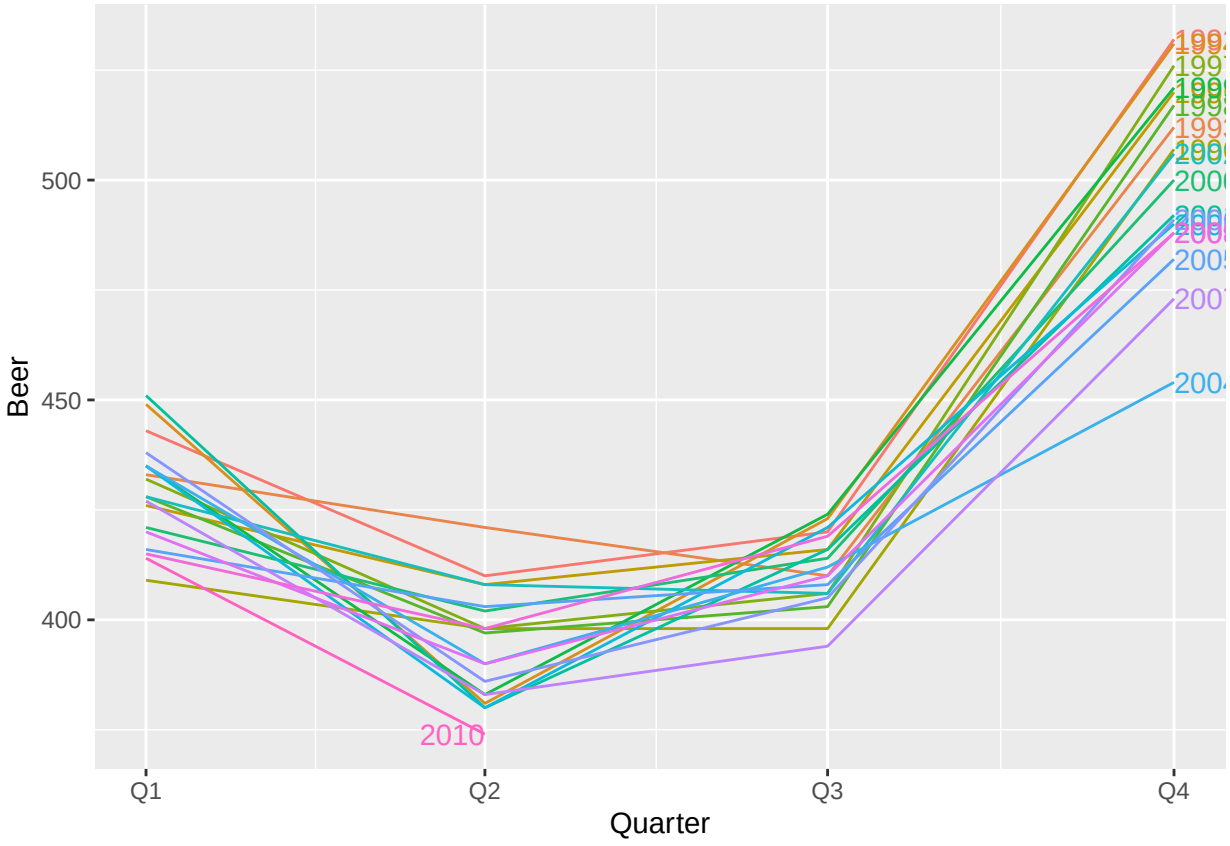


Features

- As we can see that for each year quarter 4 has spikes (4th observation after each year in the plot)
- There is also a general downward trend here

Now, let's move to seasonal plot.

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



Here, we can see clearly that spikes happen at quarter 4. because quarter 4 has summer season and also quarter 2 has lower consumption because it has summer season.

Also, Quarter 4 has spikes lower and out of order as compared to other years. Can be checked by time plot.

Example

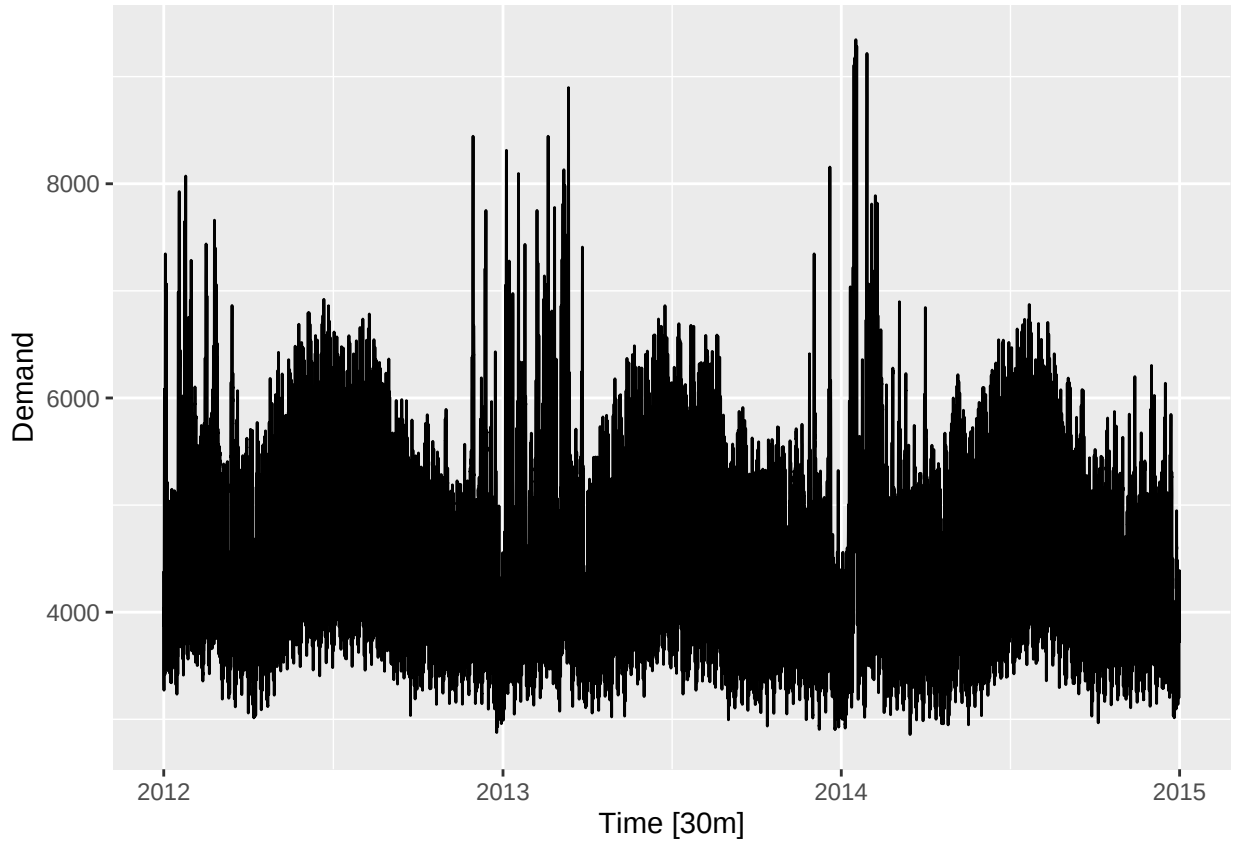
```
vic_elec
```

```
## # 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
## # i 52,598 more rows
```

Now, we see the time plot as:

```
vic_elec |> autoplot()
```

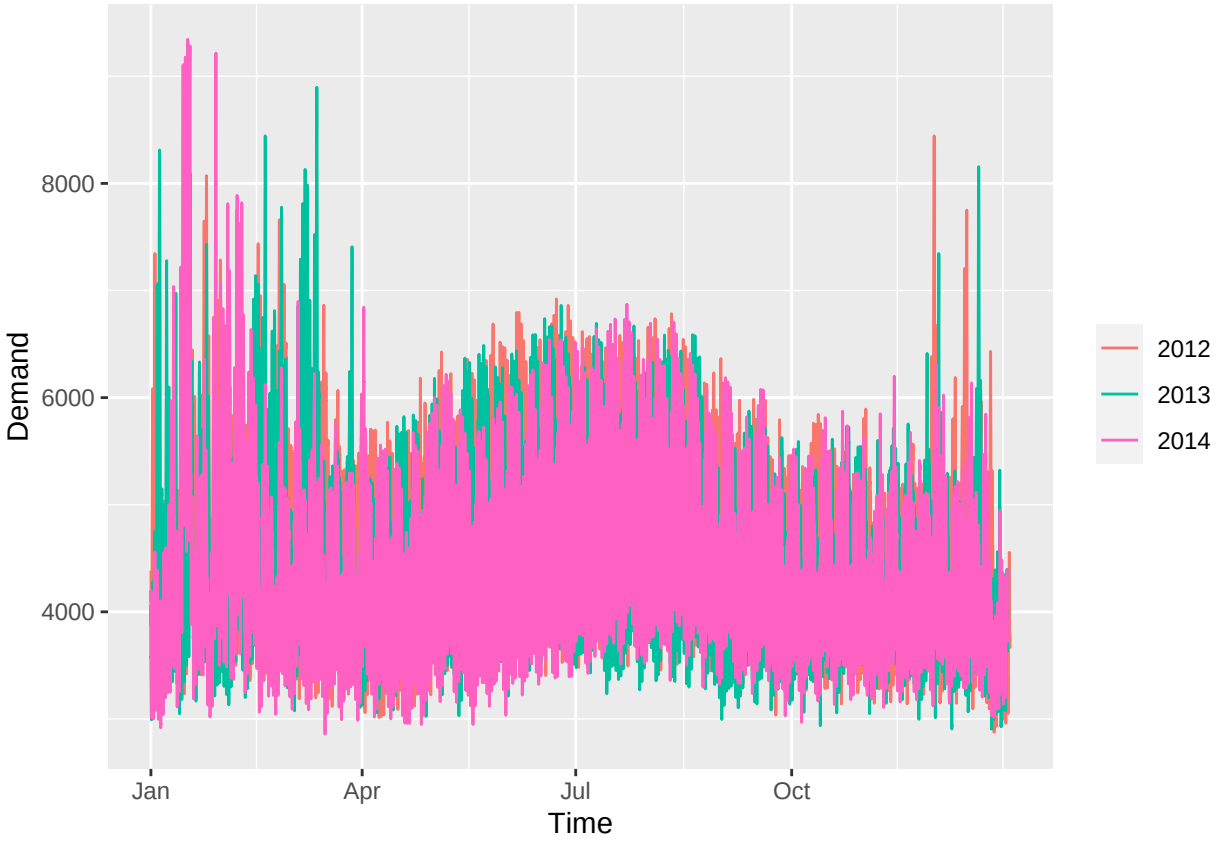
```
## Plot variable not specified, automatically selected `.vars = Demand`
```



Here we have a lot of observations (52658 for 3 years).

Now, to see the features, we see the seasonal plot:

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



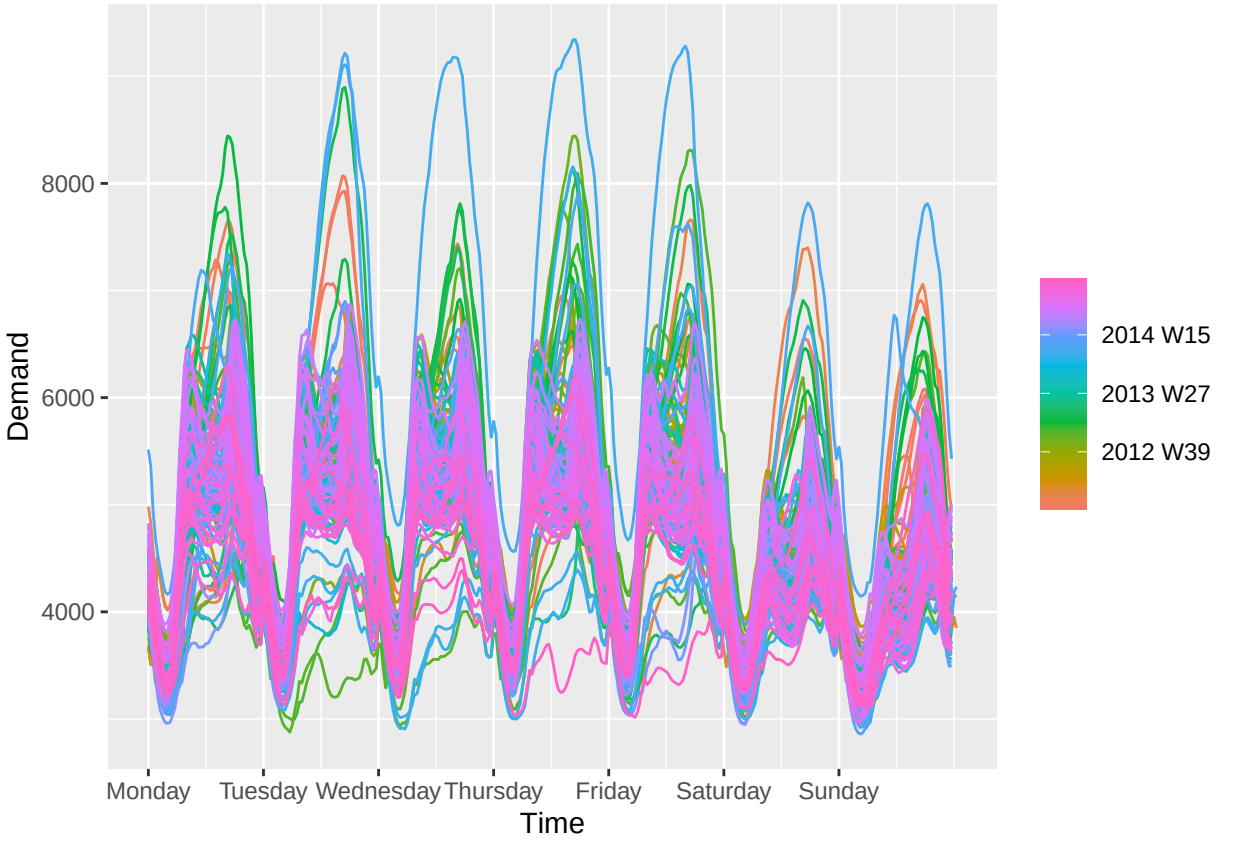
Features

- Jan, Feb and December are summer months in Australia and hence we see a lot of variations and spikes for electricity demand.
- Between apr to dec, autumn,winter (comparably more out of three) and spring so lower demands.

(We can now see in time plot as well)

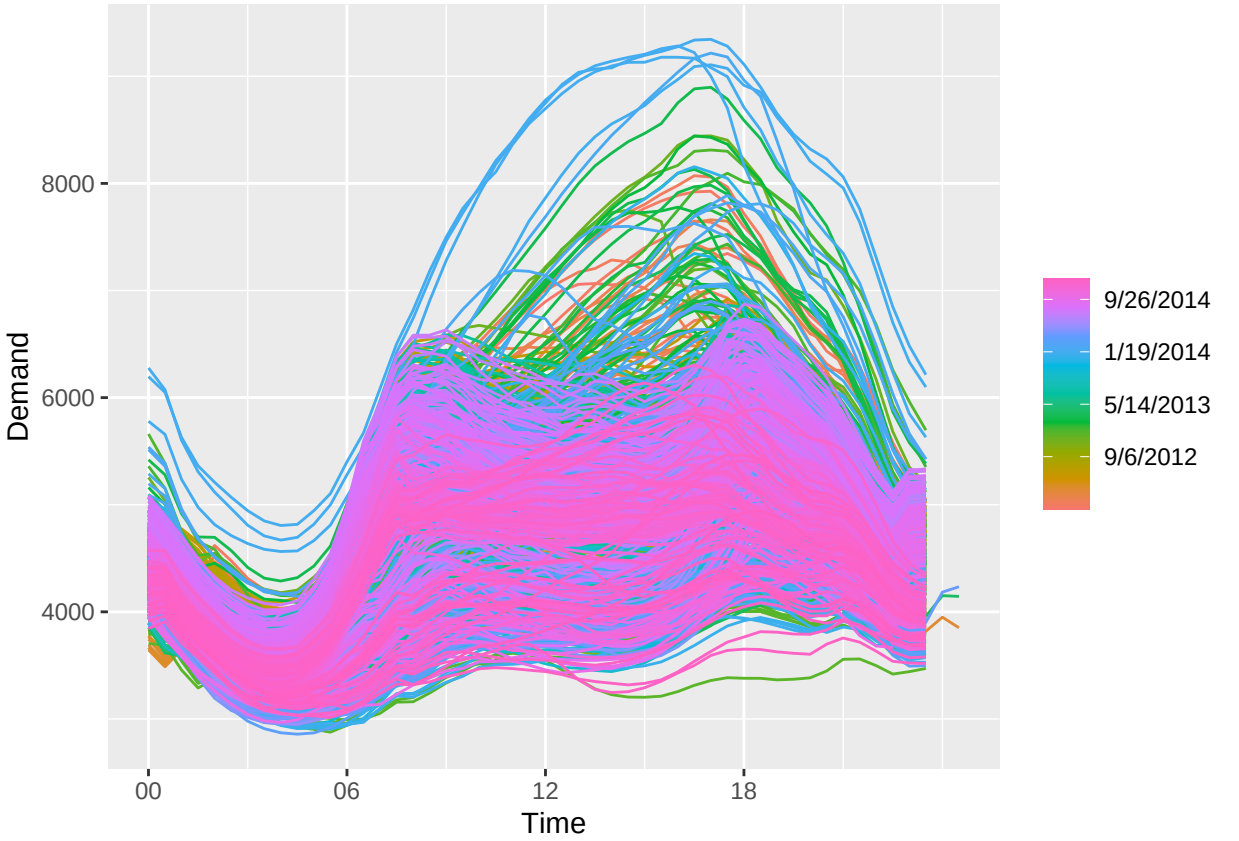
Multiple Seasonal periods

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



Observe that weekend demand is lower than weekday demand. The seasonal pattern on weekend is different than seasonal pattern in weekday.Eveneing spike is higher than morning on weekends but on weekdays, it is almost equal.

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



Here, till 6 am, people are sleeping so lower demand after people wake up and demand increases. Now, after 10 am, people go to work so demand decreases and when come home at 5 o' clock then demand increases and then decrease and in night they may need hot water and so demand increases slightly.

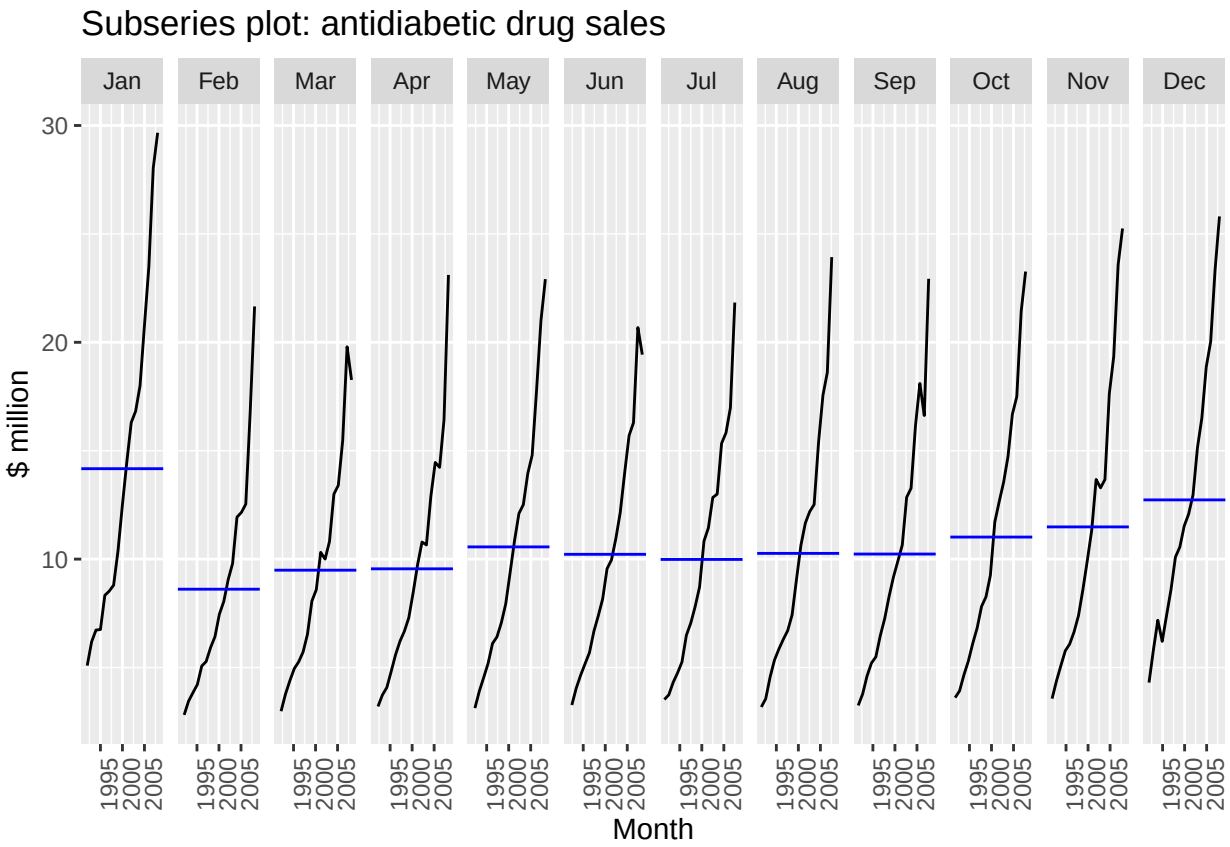
A seasonal plot allows the underlying seasonal pattern to be seen more clearly, and is especially useful in identifying years in which the pattern changes.

2.5 Seasonal subseries plots

A subseries plot is a plot that plots together as their own time series that each month/quarter/etc i.e. each season basically.

Example

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



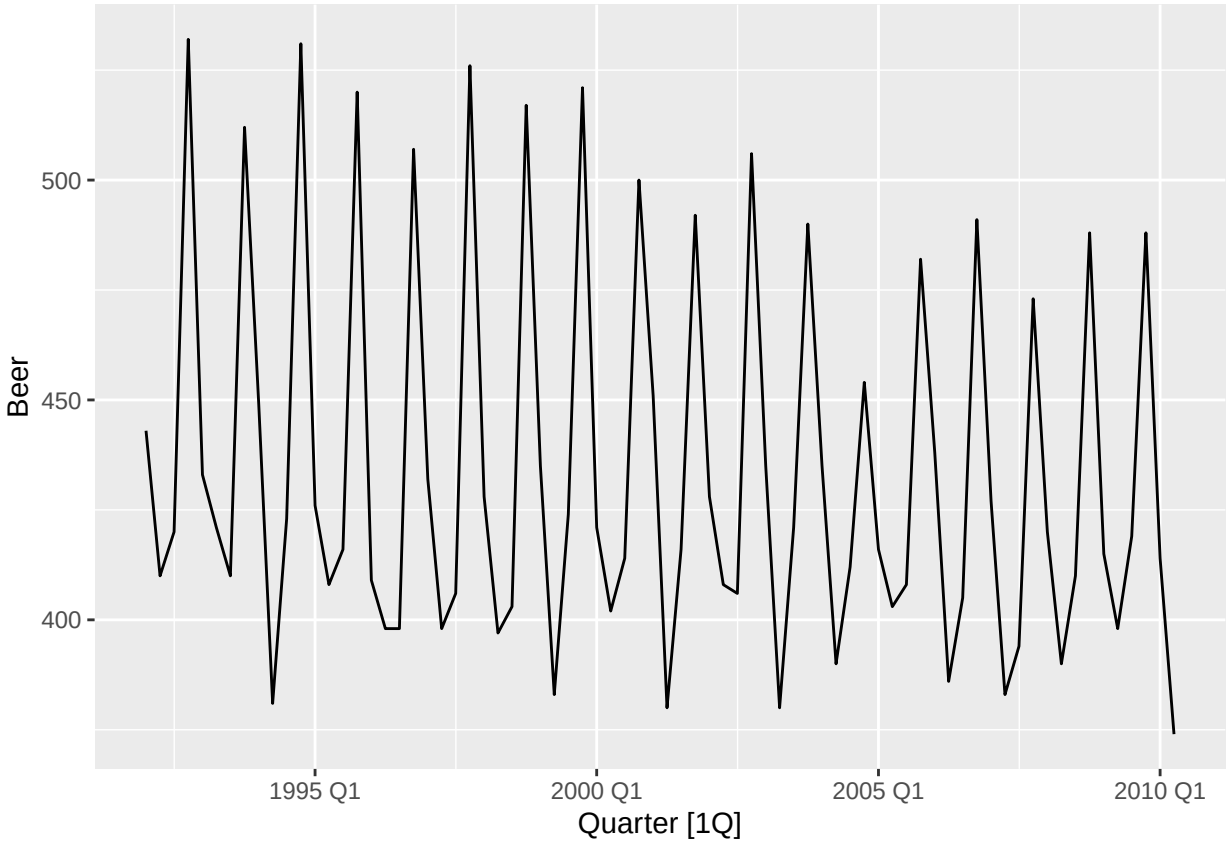
Features

- There is an increasing trend in each of the months
- Blue line represents the average value i.e. average seasonality for each month. Jan is highest average value.

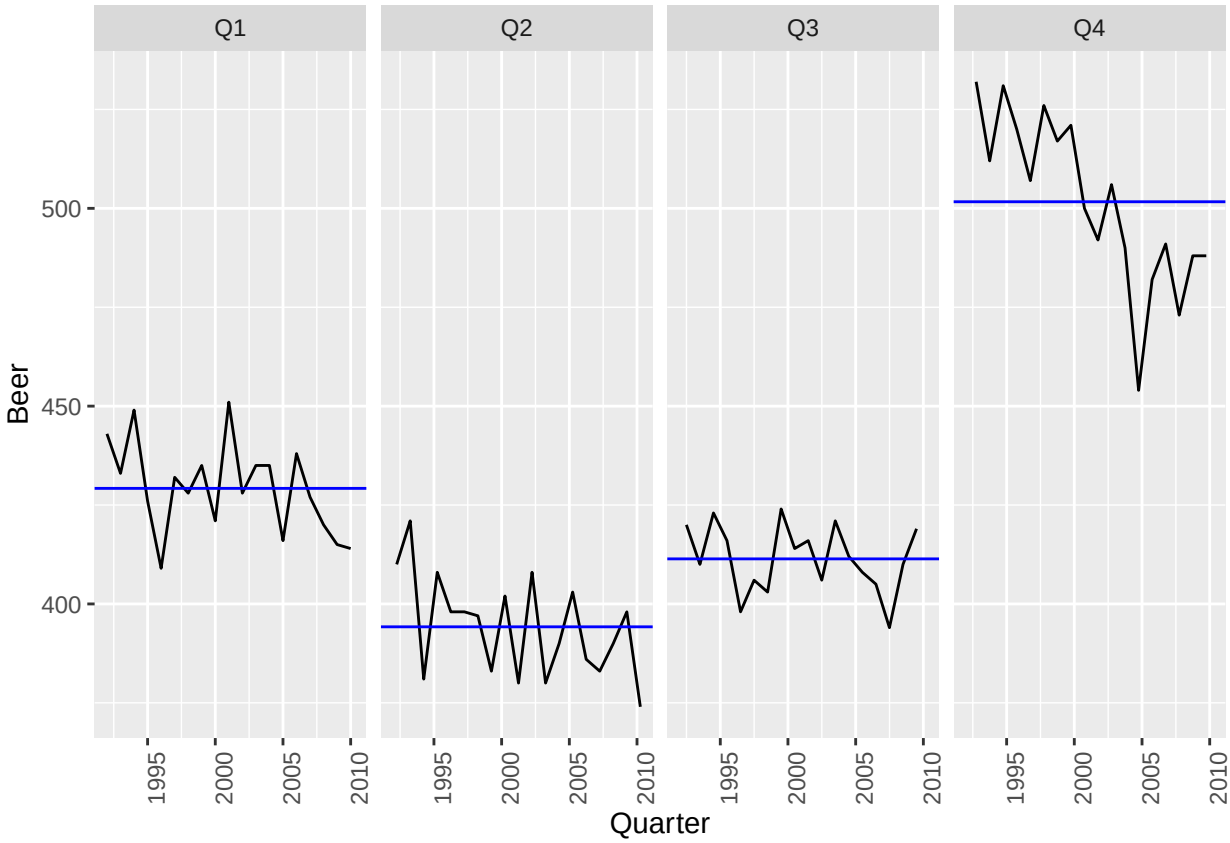
Example

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

```
## Plot variable not specified, automatically selected `.vars = Beer`
```




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



Features

- Q4 is highest consumption and Q2 is lowest.
- there is decreasing trend in Q4

Example

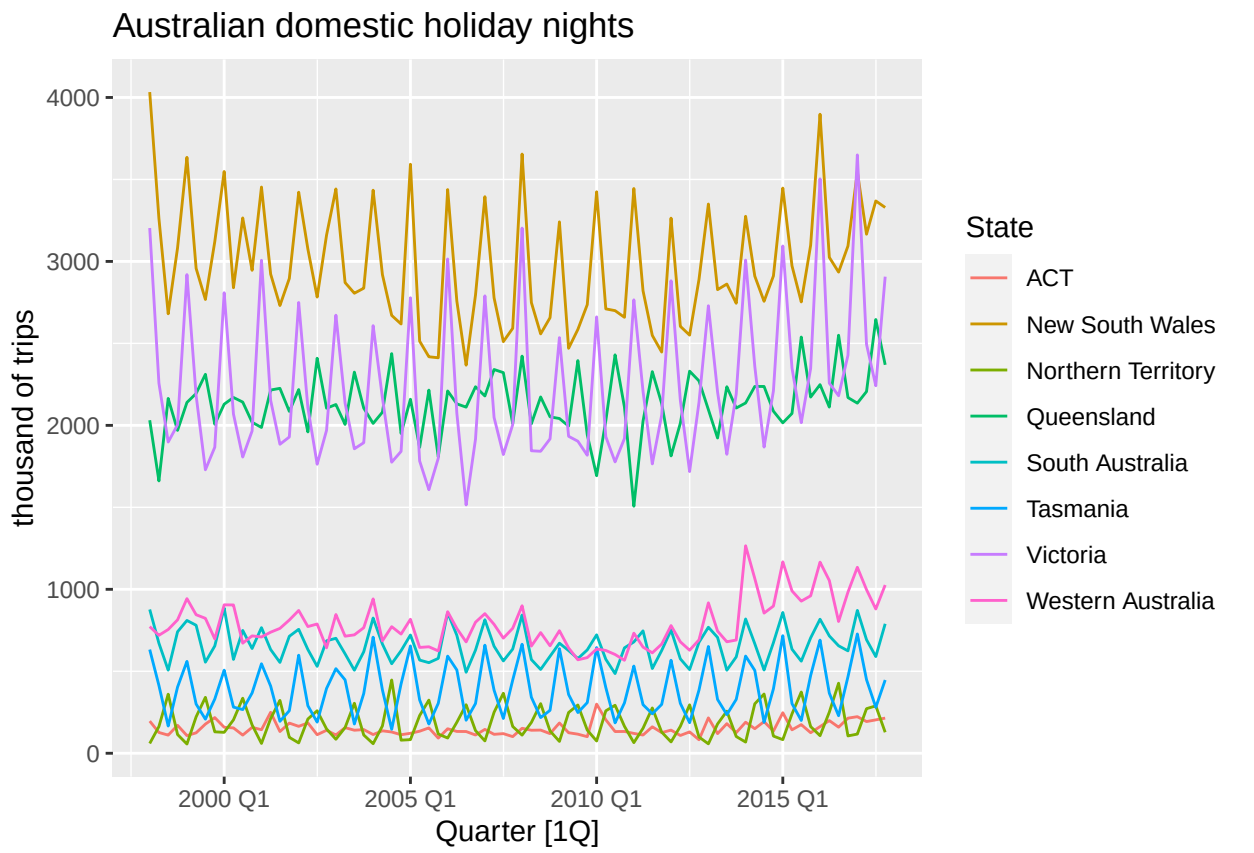
```
holidays <- tourism |>
  filter(Purpose=="Holiday") |>
  group_by(State) |>
  summarise(Trips=sum(Trips))
holidays
```

```
## # 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.
## 10 ACT   2000 Q2  155.
## # i 630 more rows
```

Now time plot will be

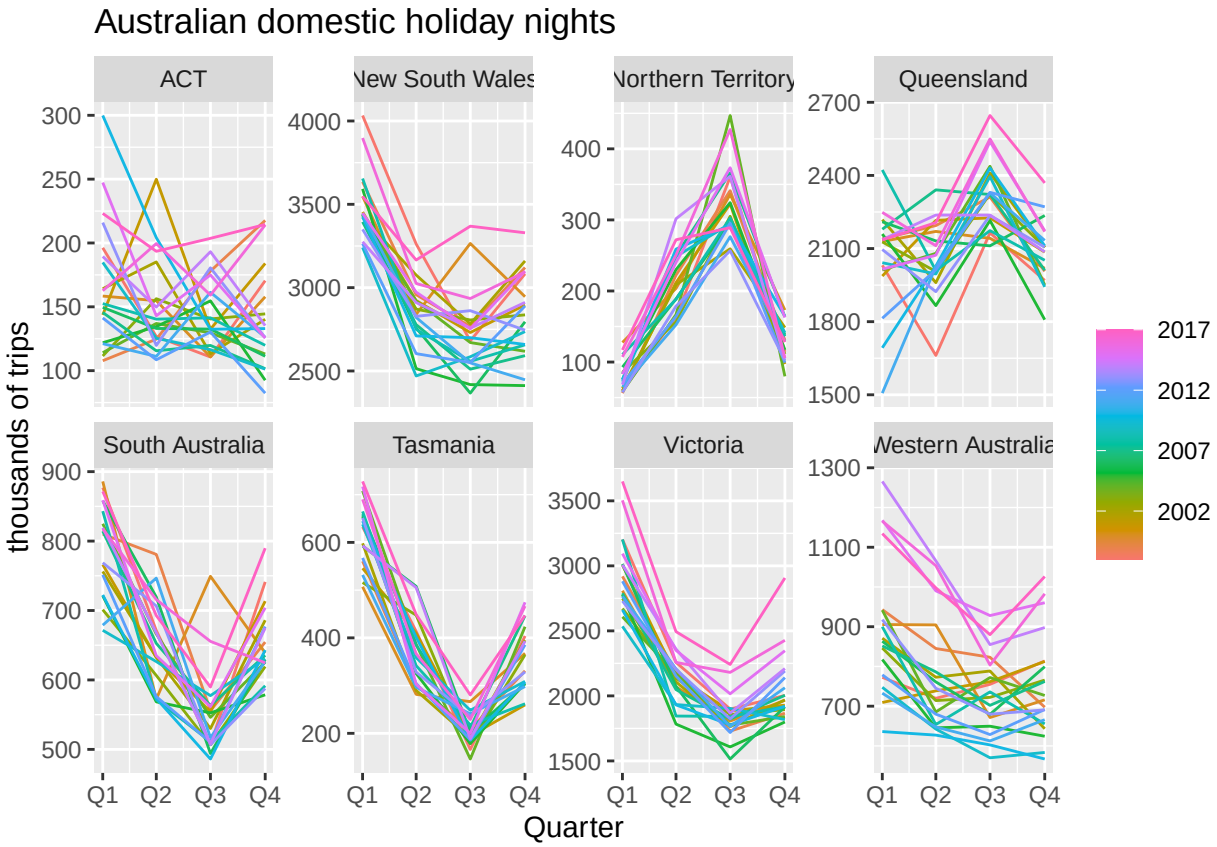
```
holidays |> autoplot(Trips) +
  labs(y="thousand of trips", title = "Australian domestic holiday nights")
```



Here, we cann a strong time seasonal component across all the time series and 2010 to 2015, slightly increasing trend is there.

Now, comes to seasonal plot.

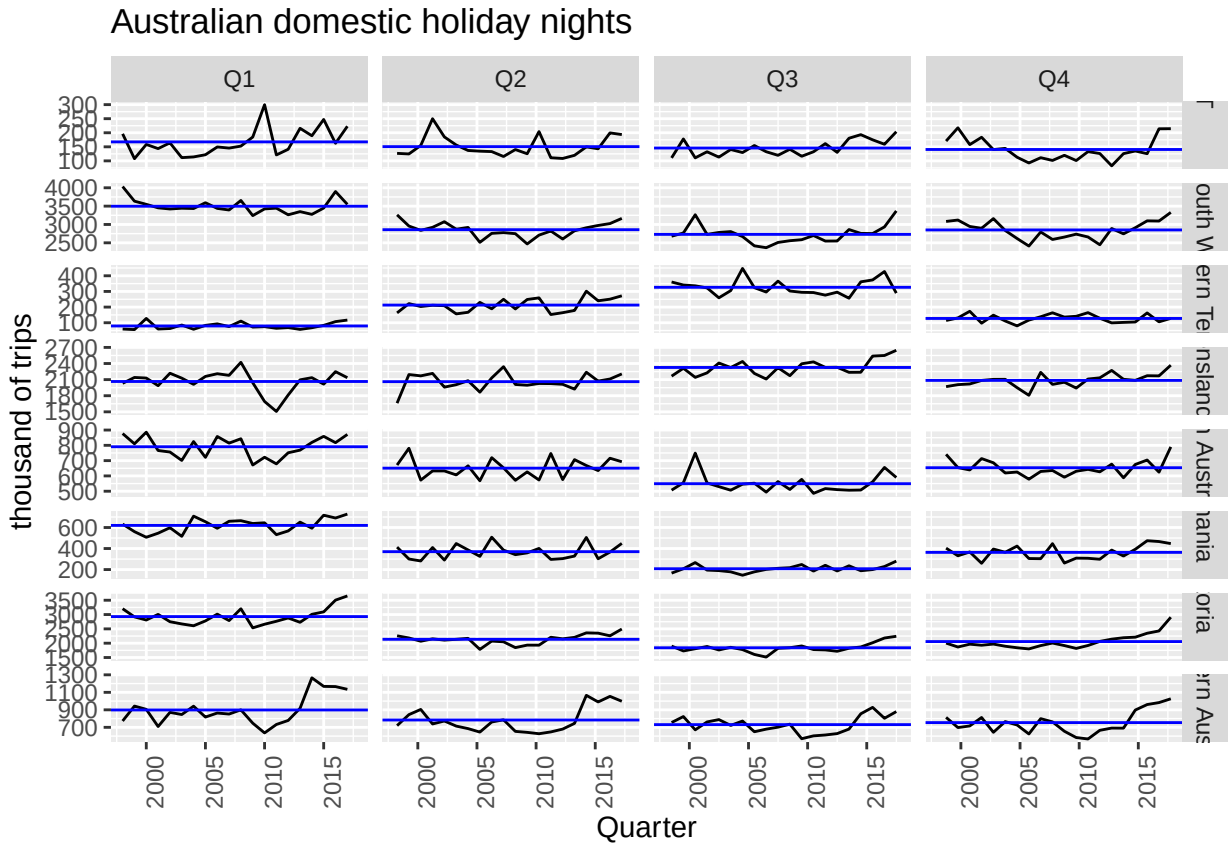
```
holidays |> gg_season(Trips) +
  facet_wrap(vars(State),nrow=2,scales = "free_y")+
  labs(y="thousands of trips",title = "Australian domestic holiday nights")
```



Here, we can see that seasonality in some states is different than some other states and same pattern exist when states occur in same geographical region on australia map. All eastern and southern states mostly visited in summer and northern states are mostly visited in Q3.

Now comes to subseries plot.

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



Example: Australian holiday tourism

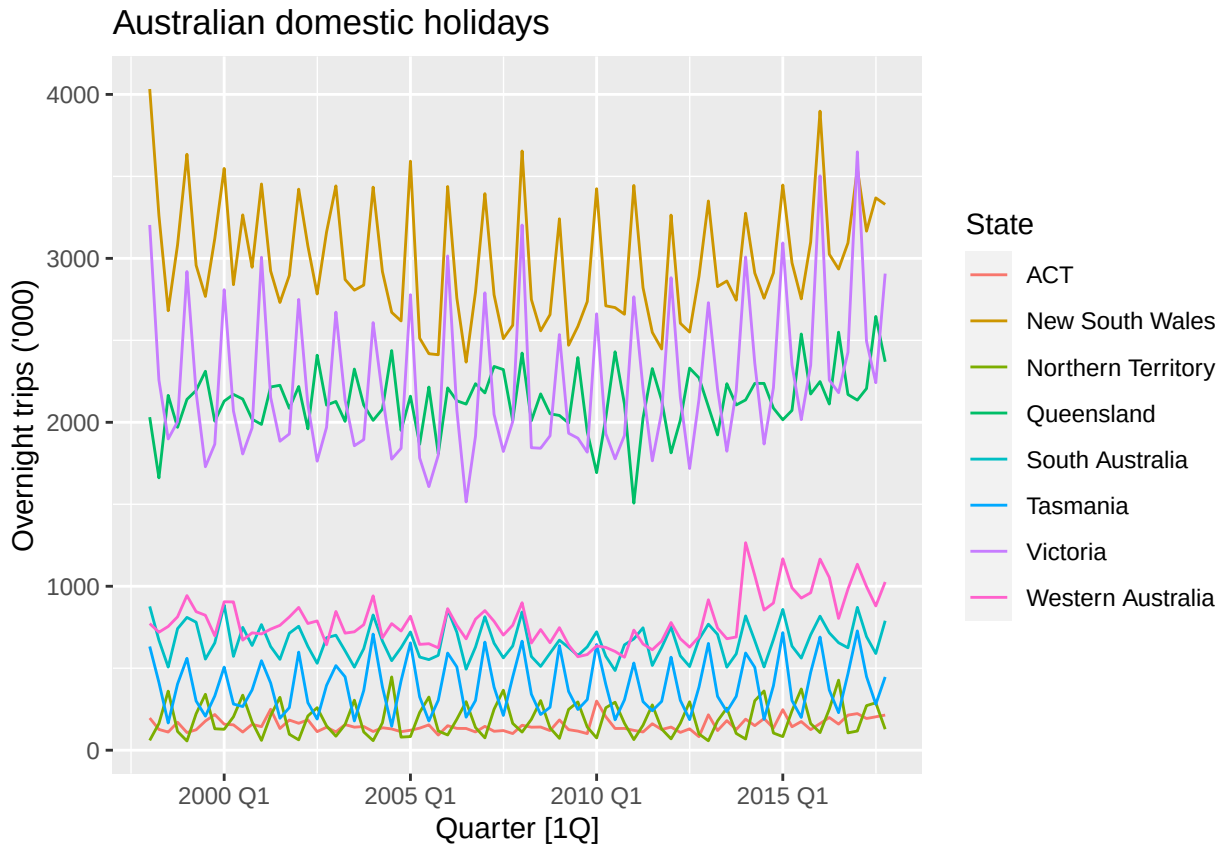
```
holidays <- tourism |>
  filter(Purpose == "Holiday") |>
  group_by(State) |>
  summarise(Trips = sum(Trips))

holidays
```

```
## # 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.
## 10 ACT   2000 Q2  155.
## # i 630 more rows
```

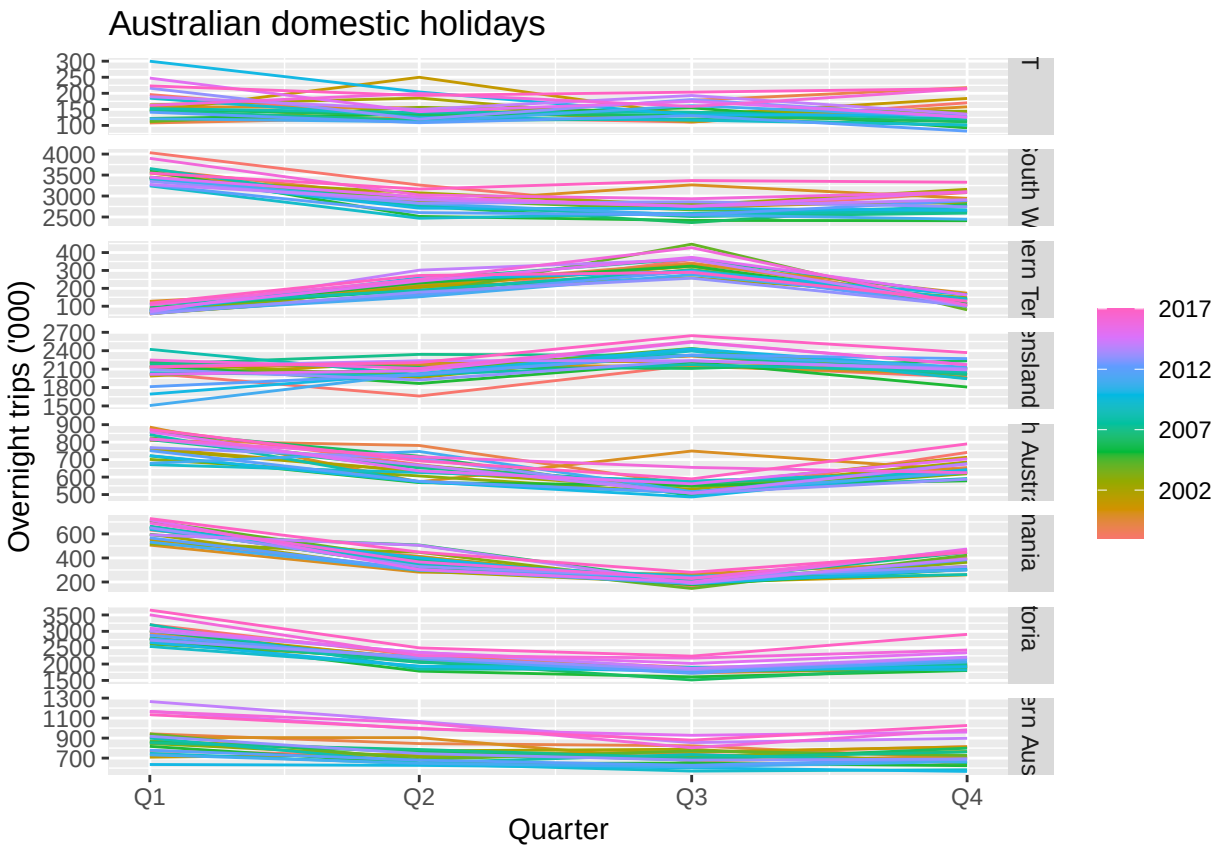
Time plots of each series show that there is strong seasonality for most states, but that the seasonal peaks do not coincide.

```
autoplot(holidays, Trips) +
  labs(y = "Overnight trips ('000)",
       title = "Australian domestic holidays")
```



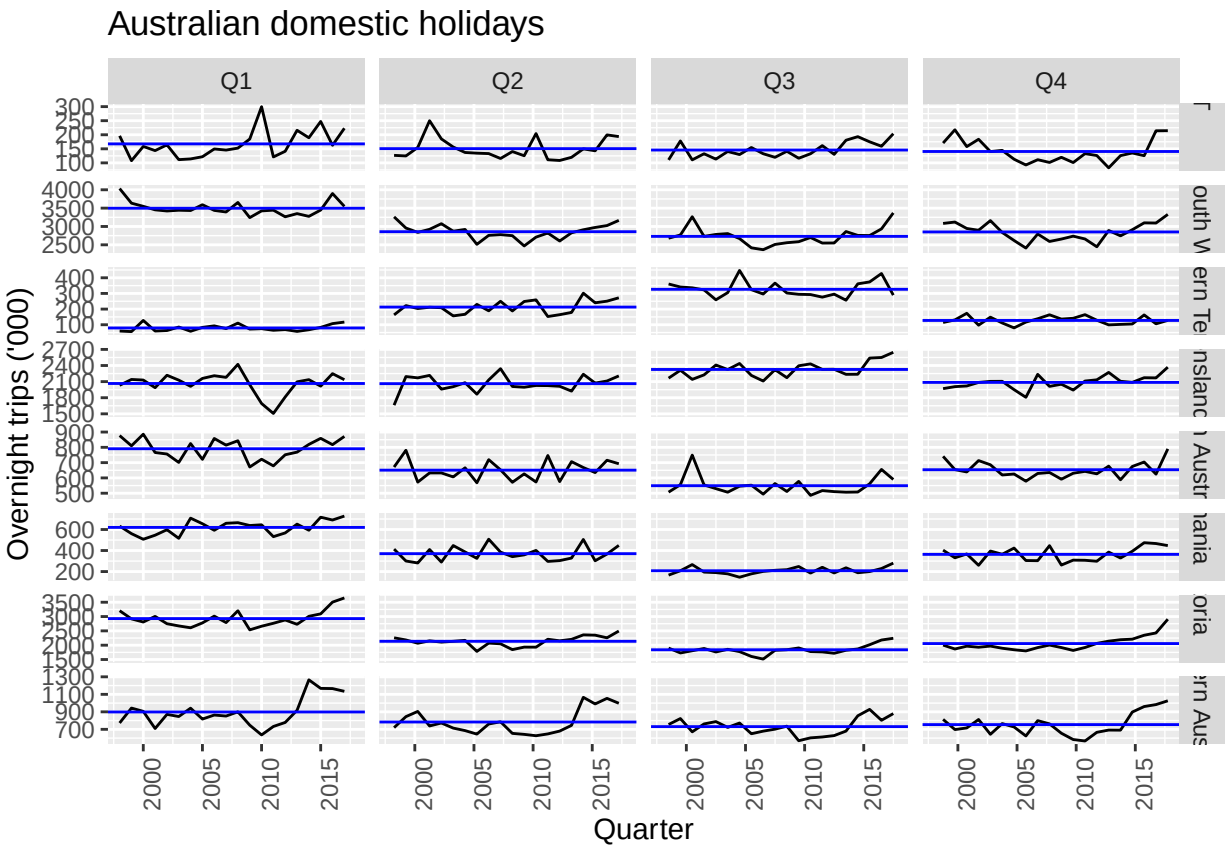
To see the timing of the seasonal peaks in each state, we can use a season plot. Figure 2.10 makes it clear that the southern states of Australia (Tasmania, Victoria and South Australia) have strongest tourism in Q1 (their summer), while the northern states (Queensland and the Northern Territory) have the strongest tourism in Q3 (their dry season).

```
gg_season(holidays, Trips) +
  labs(y = "Overnight trips ('000)",
       title = "Australian domestic holidays")
```



The corresponding subseries plots are shown in Figure 2.11.

```
holidays |>
  gg_subseries(Trips) +
  labs(y = "Overnight trips ('000)",
       title = "Australian domestic holidays")
```



This figure makes it evident that Western Australian tourism has jumped markedly in recent years, while Victorian tourism has increased in Q1 and Q4 but not in the middle of the year.

2.6 Scatterplots

Sometimes we have to identify relationship between multiple time series. Now we see how to visulaise it using scatter plots.

Example

```
vic_elec_day_type <- vic_elec |>
  filter(year(Time) == 2014) |>
  mutate(Day_Type = case_when(
    Holiday ~ "Holiday",
    wday(Date) %in% 2:6 ~ "Weekday",
    TRUE ~ "Weekend"
  ))
vic_elec_day_type
```

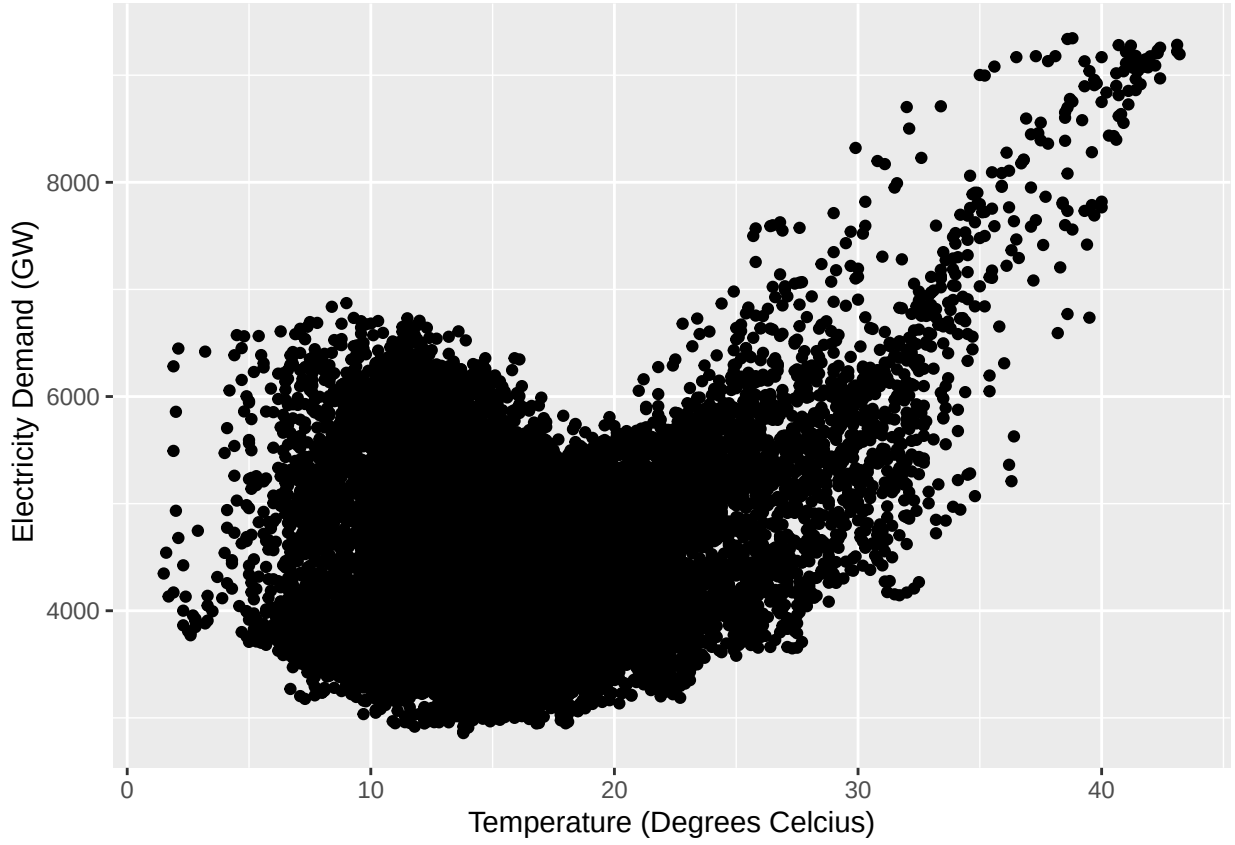
```
## # A tsibble: 17,520 x 6 [30m] <Australia/Melbourne>
##    Time                Demand Temperature Date       Holiday Day_Type
```

```
##    <dttm>              <dbl>       <dbl> <date>     <lgl>   <chr>
##  1 2014-01-01 00:00:00  4092.        18.7 2014-01-01 TRUE    Holiday
##  2 2014-01-01 00:30:00  4198.        18.1 2014-01-01 TRUE    Holiday
##  3 2014-01-01 01:00:00  3915.        18.2 2014-01-01 TRUE    Holiday
##  4 2014-01-01 01:30:00  3673.        17.9 2014-01-01 TRUE    Holiday
##  5 2014-01-01 02:00:00  3498.        17.6 2014-01-01 TRUE    Holiday
##  6 2014-01-01 02:30:00  3339.        16.8 2014-01-01 TRUE    Holiday
##  7 2014-01-01 03:00:00  3204.        16.3 2014-01-01 TRUE    Holiday
##  8 2014-01-01 03:30:00  3100.        16.6 2014-01-01 TRUE    Holiday
##  9 2014-01-01 04:00:00  3039.        16.6 2014-01-01 TRUE    Holiday
## 10 2014-01-01 04:30:00  3012.        16.7 2014-01-01 TRUE    Holiday
## # i 17,510 more rows
```

when Holiday column value is true then day_type value is holiday otherwise values will be weekday and weekend according to the day of the date.

Now scatter plot will be:

```
# half-hourly electricity demand (in Gigawatts) and temperature (in degrees Celsius), for 2014 in Victo
vic_elec_day_type |>
  ggplot(aes(x=Temperature,y=Demand))+
  geom_point()+
  labs(x="Temperature (Degrees Celcius)", y=" Electricity Demand (GW)")
```

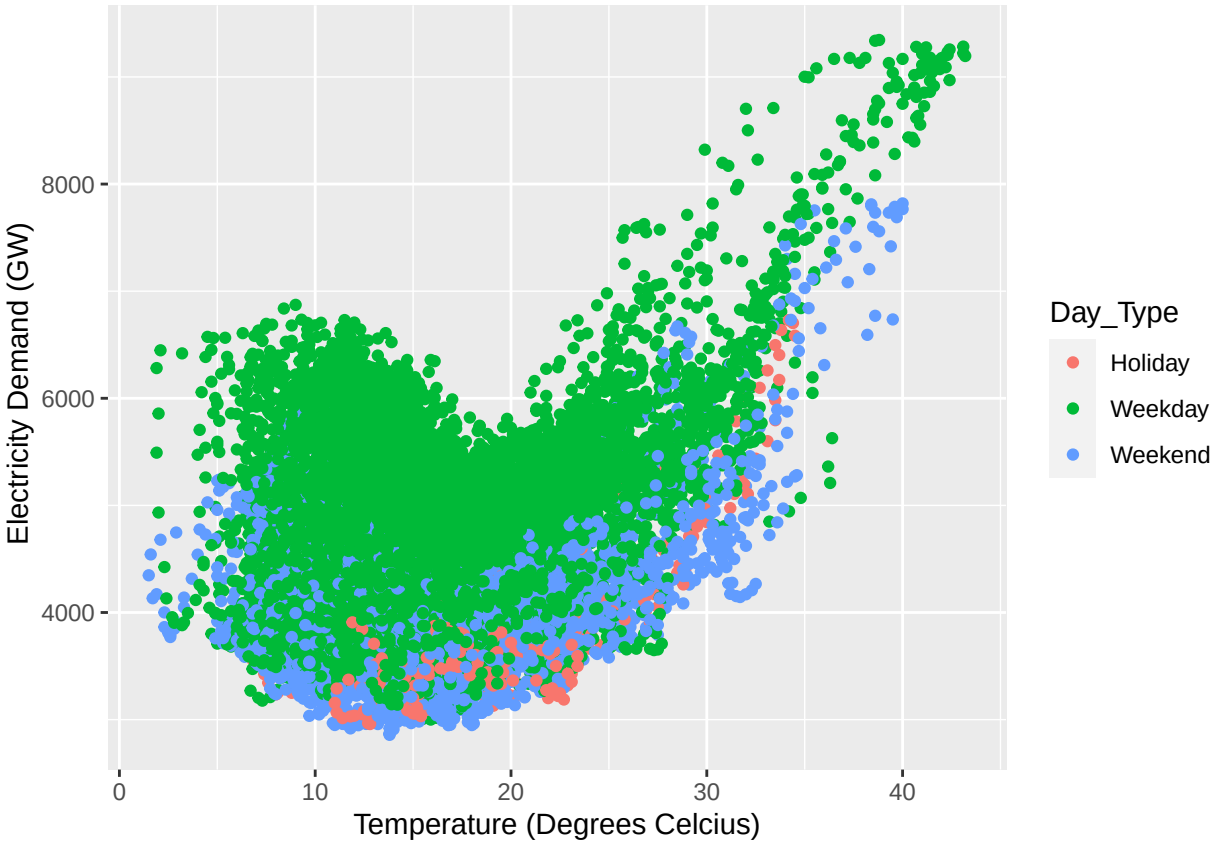


This plot allows you to relationship between two variables.This is clearly a non-linear relationship.

Here, we can see that after temperature 20 degree celcius when temperature increases, demand increases. So there is upward trend or positive relationship.

Also, when temperature is low (cold) then demand also increases.

```
vic_elec_day_type |>
  ggplot(aes(x=Temperature,y=Demand,colour=Day_Type))+
  geom_point()+
  labs(x="Temperature (Degrees Celcius)", y=" Electricity Demand (GW)")
```



Here we can see the low demand on the weekend and high demand on weekdays but shape is same.

Correlation Coefficient

It measures the extent of a linear relationship between two variables (y and x)

$$r = \frac{\Sigma_{t=1}^{T}(y_t - \bar{y})(x_t - \bar{x})}{\sqrt{\Sigma_{t=1}^{T}(y_t - \bar{y})^2}\sqrt{\Sigma_{t=1}^{T}(x_t - \bar{x})^2}}$$

Value of r lies between -1 and 1 (included both).

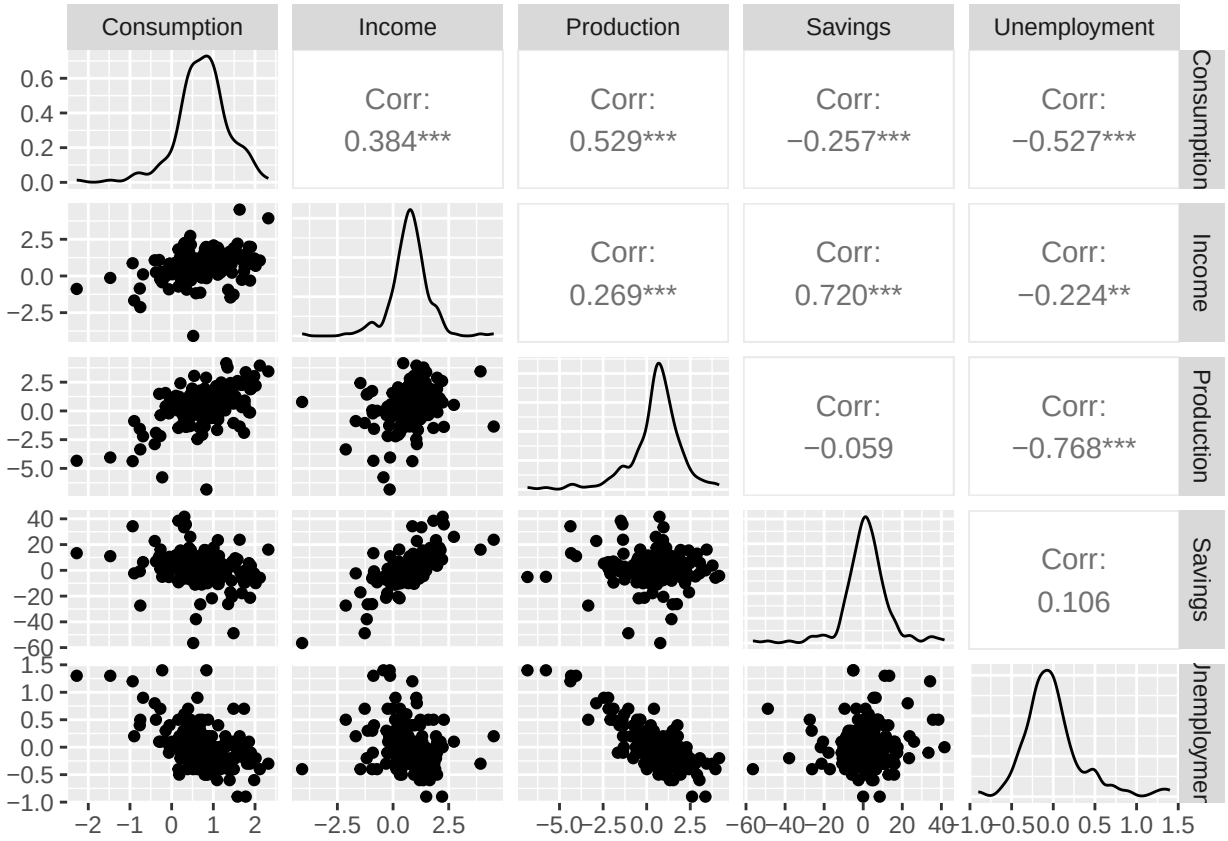
It does not capture the non-linear relationship.

Scatter Plot for more than 2 variables.

```
library(GGally) # gg all y
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
us_change |> GGally :: ggpairs(columns=2:6)
```



ggpairs() plots the matrix of the above form. Here, in diagonals we have densities of the variables (observe feature as positive skewed graph on unemployment) and upper half shows the correlation coefficient value between variables and in lower half it shows the plot between variables.

The correlation coefficient only measures the strength of the linear relationship between two variables, and can sometimes be misleading. For example, the correlation for the electricity demand and temperature data shown in Figure 2.14 is 0.28, but the non-linear relationship is stronger than that.

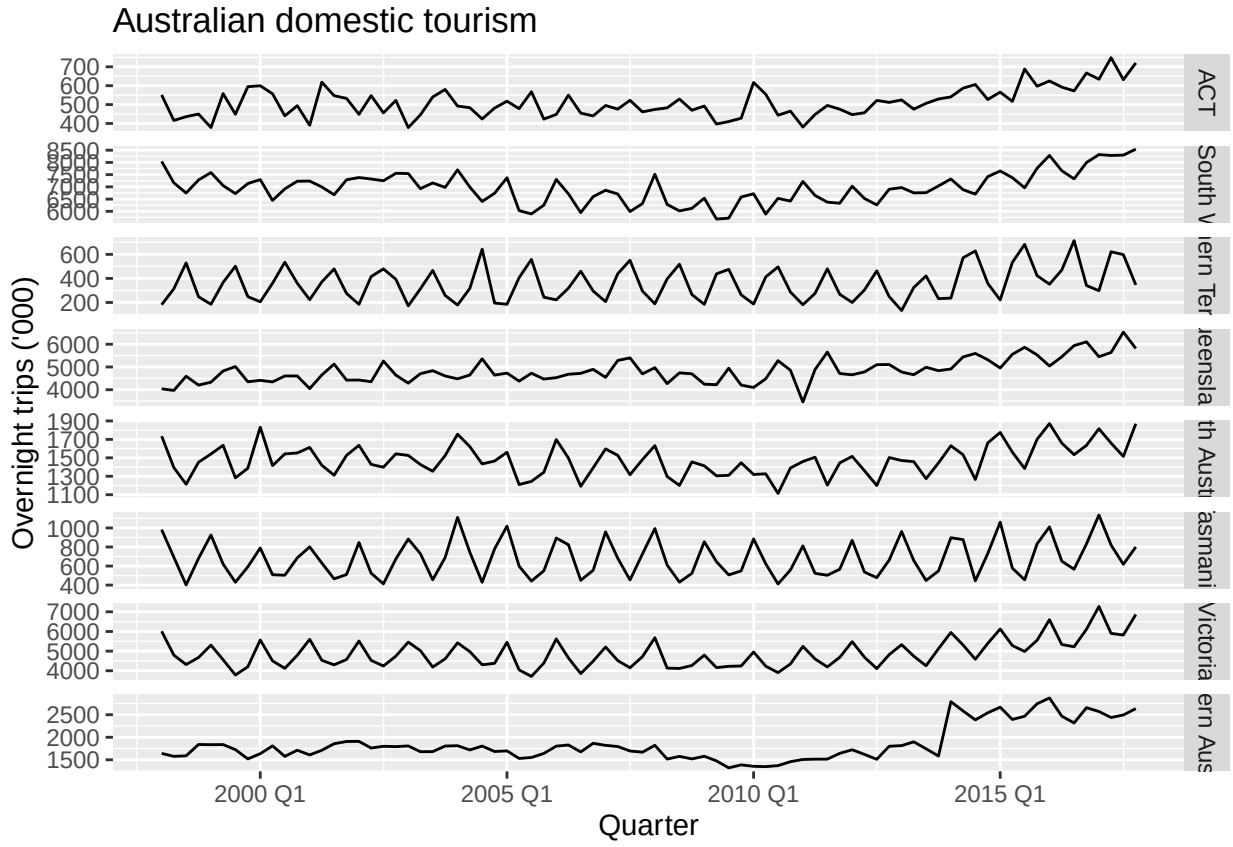
The plots in Figure 2.16 all have correlation coefficients of 0.82, but they have very different relationships. This shows how important it is to look at the plots of the data and not simply rely on correlation values.

Scatterplot matrices

For 8 time series data (8 dependent columns in the dataset)

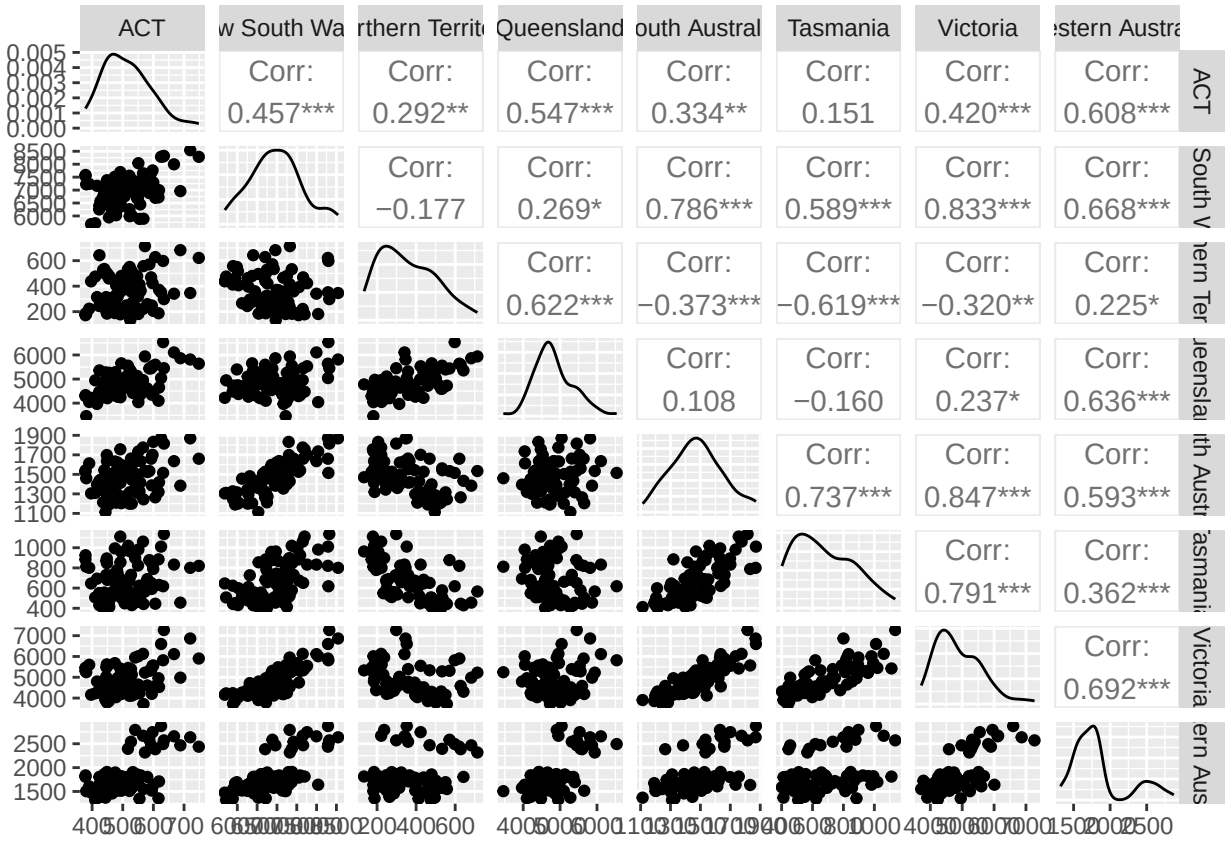
```
visitors <- tourism |>
  group_by(State) |>
  summarise(Trips = sum(Trips))
visitors |>
```

```
  ggplot(aes(x = Quarter, y = Trips)) +
  geom_line() +
  facet_grid(vars(State), scales = "free_y") +
  labs(title = "Australian domestic tourism",
       y= "Overnight trips ('000)")
```



To see the relationships between these eight time series, we can plot each time series against the others. These plots can be arranged in a scatterplot matrix, as shown in Figure 2.18. (This plot requires the GGally package to be installed.)

```
visitors |>
  pivot_wider(values_from=Trips, names_from=State) |>
  GGally::ggpairs(columns = 2:9)
```



For each panel, the variable on the vertical axis is given by the variable name in that row, and the variable on the horizontal axis is given by the variable name in that column. There are many options available to produce different plots within each panel. In the default version, the correlations are shown in the upper right half of the plot, while the scatterplots are shown in the lower half. On the diagonal are shown density plots.

The value of the scatterplot matrix is that it enables a quick view of the relationships between all pairs of variables. In this example, mostly positive relationships are revealed, with the strongest relationships being between the neighbouring states located in the south and south east coast of Australia, namely, New South Wales, Victoria and South Australia. Some negative relationships are also revealed between the Northern Territory and other regions. The Northern Territory is located in the north of Australia famous for its outback desert landscapes visited mostly in winter. Hence, the peak visitation in the Northern Territory is in the July (winter) quarter in contrast to January (summer) quarter for the rest of the regions.

2.7 Lag plots

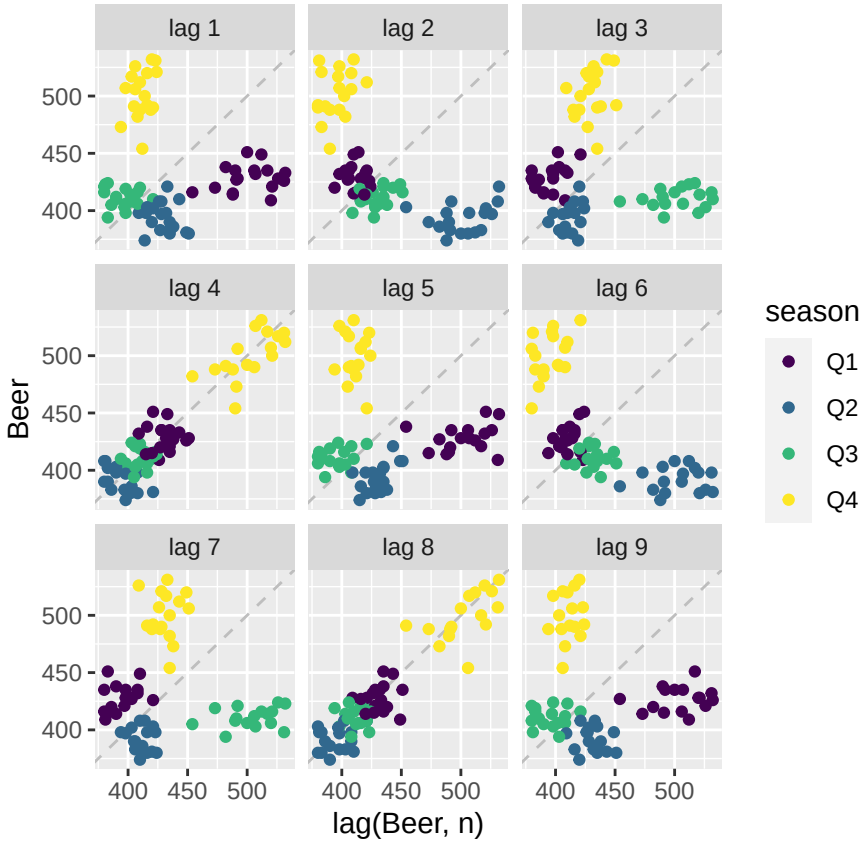
This is a particular scatter plot.

Example

```
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
## # i 64 more rows
```

```
new_production |> gg_lag(Beer,geom = "point")
```



It produces a set of scatter plots which are shown above and labelled as lag 1 to lag 9. These plots show the beer data to lags of the beer data.

Each graph shows the plot between y_t and y_{t-k} for different values of k.

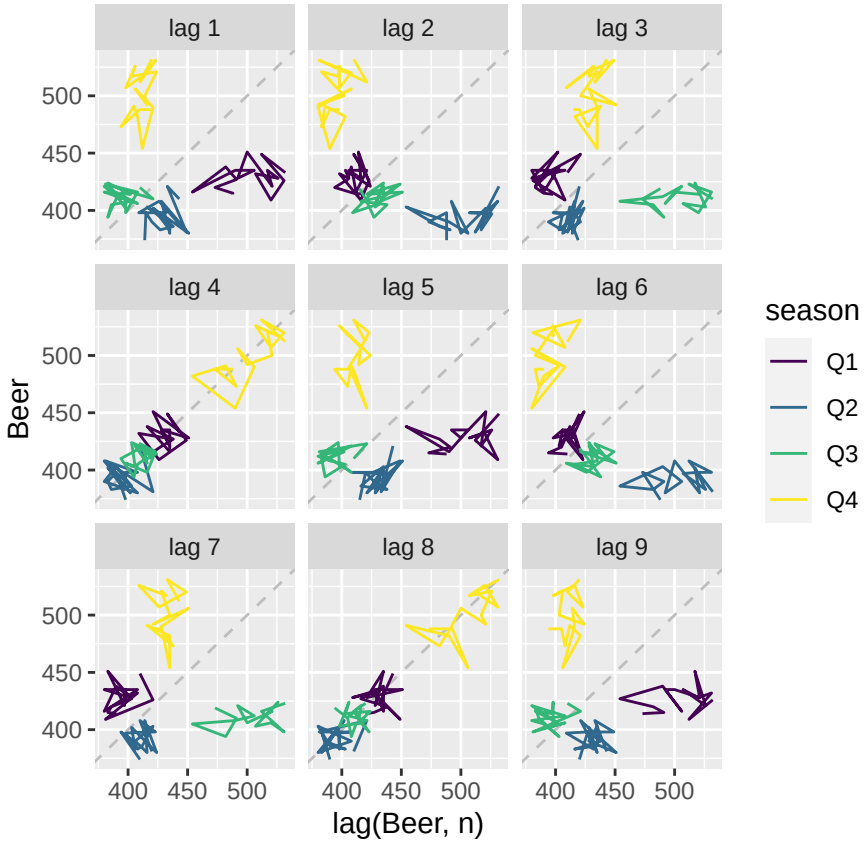
So, for lag 1 plot, yellow points represent the q4 data on y-axis and corresponding q3 data on x-axis. Similarly, green colored q3 data on y-axis and corresponding q2 data on x-axis and so on.

Foe lag 2 plot, we use the same logic but here we use the lag 2 i.e. for Q-1 data on y-axis corresponds to Q-3 of previous year on x-axis.

Now, see the plot 4 (lag 4), there is a strong linear relationship due to high seasonality because at that Q-4 of current year on y-axis corresponds to Q-4 of last year in x-axis.

If we don't use the geom_point() then it would look like:

```
new_production |> gg_lag(Beer)
```



It is the same graph but instead of points we have lines and they are connected in time order.

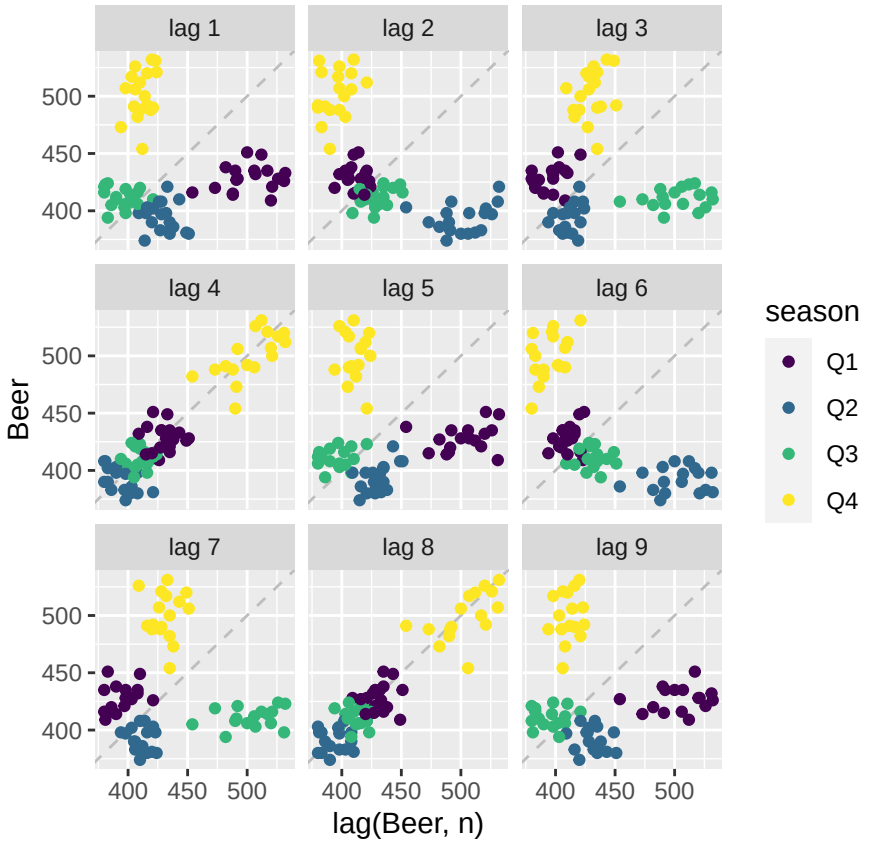
2.8 Autocorrelation

It is one of the key features of the time series.

Example

Let's start by a lag 1 plot

```
new_production |> gg_lag(Beer,geom="point")
```



Here, y is the quarterly beer production in Australia.

1. Each graph shows y_t plotted against y_{t-k} for different values of k.
2. The autocorrelations are the correlations associated with these scatterplots.
3. $r_1 = Coorelation(y_t, y_{t-1})$

$r_2 = Coorelation(y_t, y_{t-2})$
$r_3 = Coorelation(y_t, y_{t-3})$
...
The correlations associated with these scatterplots are called autocorrelations.
It gives a series of correlations related to how values of a time series are correlated with each other.

Autocorrelation

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

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

and

$$r_k = \frac{c_k}{c_0}$$

where c_0 is variance

Here, r_1 indicates how successive values of y relate to each other.

r_2 indicates how y values 2 periods apart relate to each other.

r_k is almost the same as the sample correlation between y_t and y_{t-k}.

Results for first 9 lags for beer data:

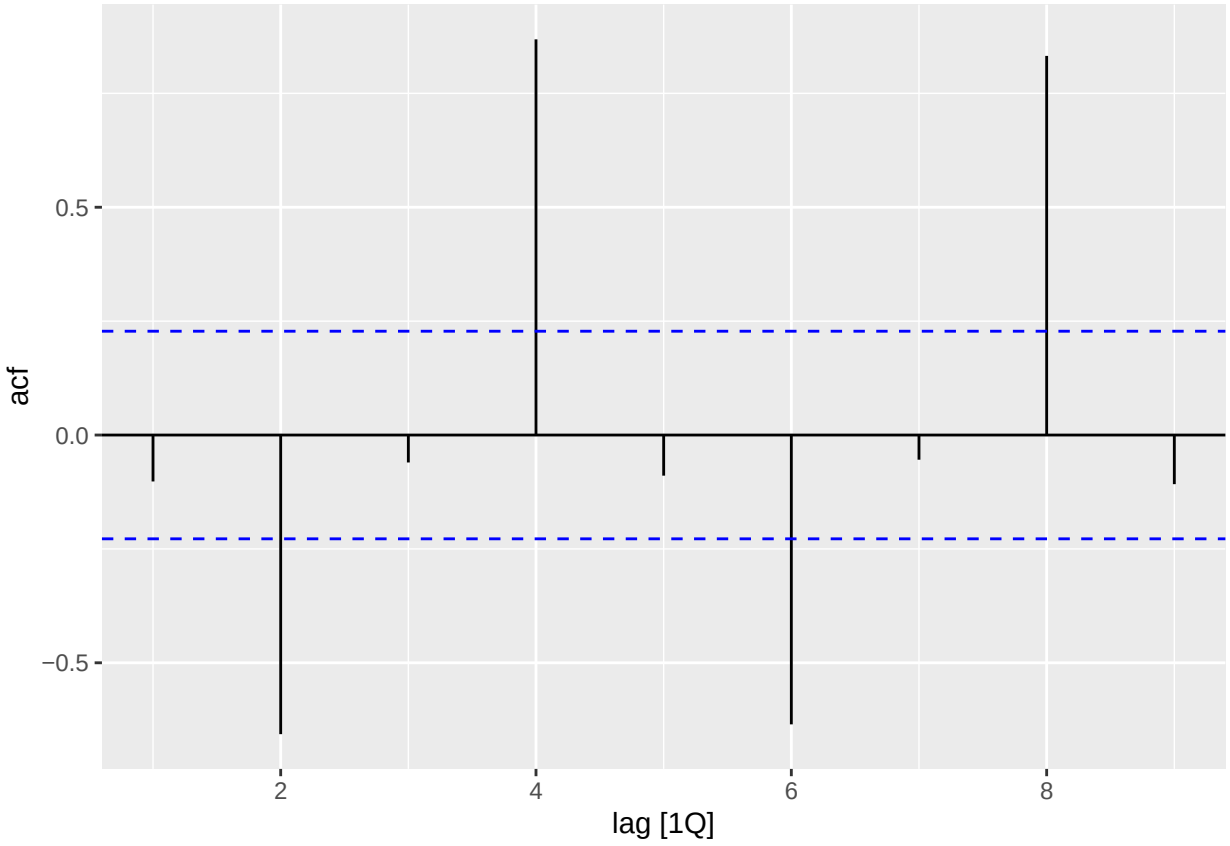
```
new_production |> ACF(Beer,lag_max = 9)
```

```
## # A tsibble: 9 x 2 [1Q]
##        lag     acf
##   <cf_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
```

Here we are showing the 9 lags. "lag" is a time index and second column is the correlation of those scatterplots.

Result for first 9 lags for beer data

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



Here black vertical lines represent the autocorrelation values from the above table.

Blue line tells that how large the correlations are compared to what we expect if we did really have a white noise(will see later)

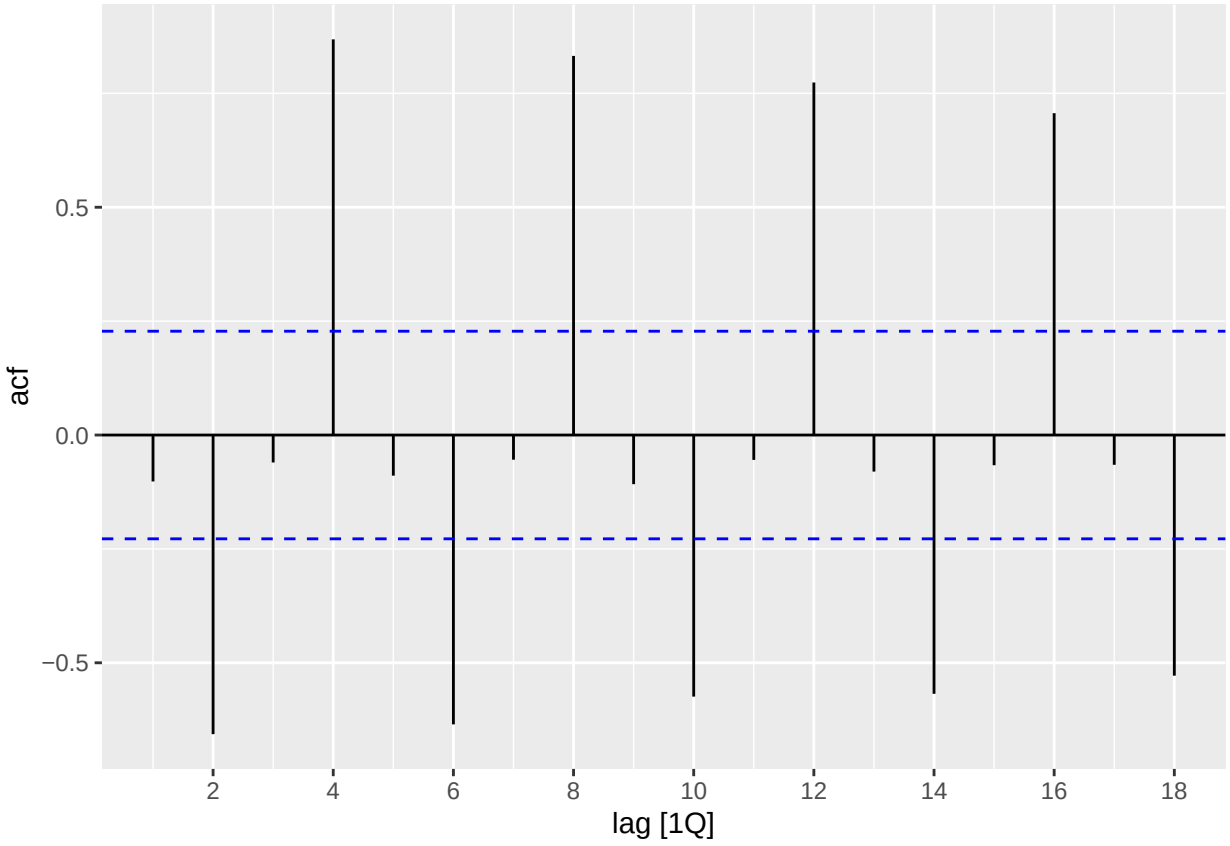
Quarters Q-4 and Q-8 have a very strong positive correlation.

Quarters Q-2 and Q-6 have a very strong negative correlation.

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

If we leave the lag_max=9 then we get:

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



- r4 is higher than for the other lags due to seasonal pattern in the data: peaks tend to be 4 quarters apart and troughs tend to be 4 quarters apart.
- r2 is more negative than for the other lags because troughs tend to be 2 quarters behind peaks.

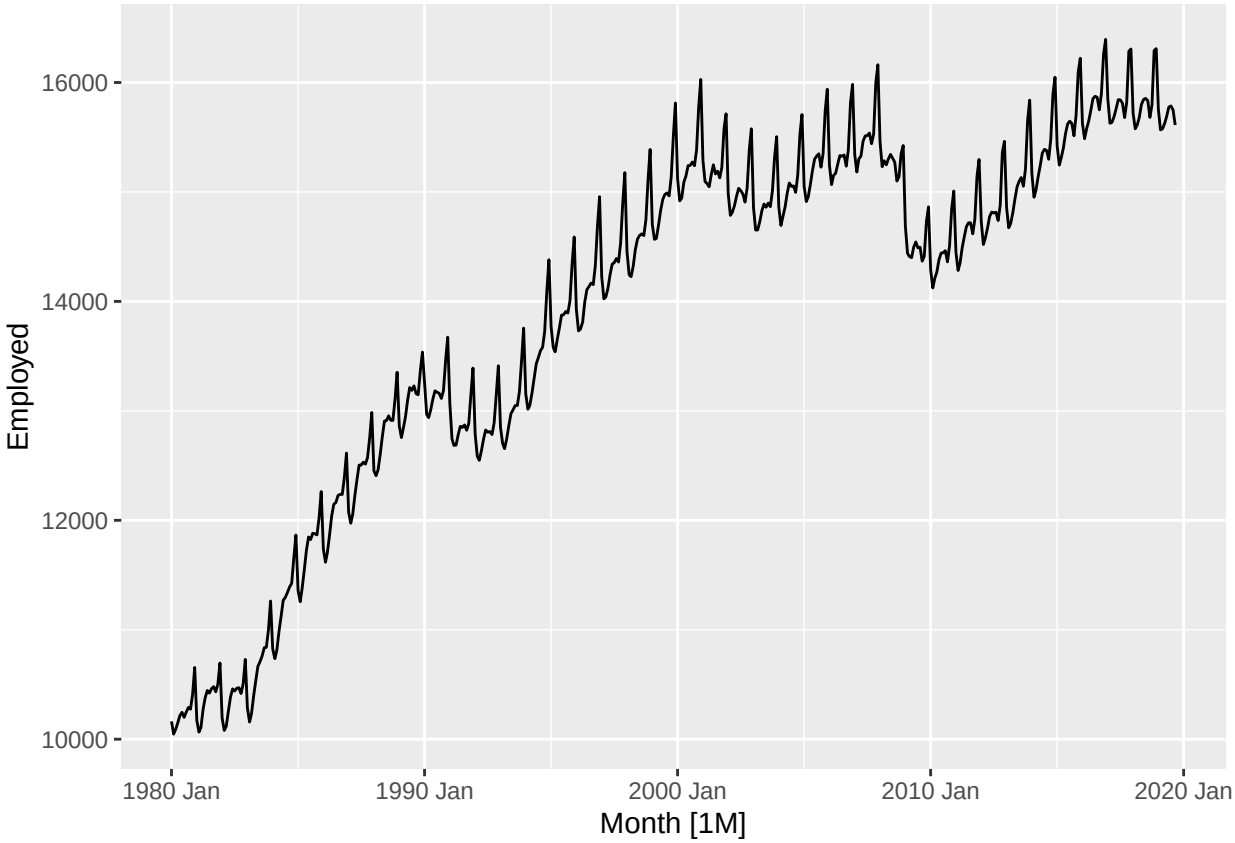
The autocorrelation function (ACF) tells us the correlation between observations and those that came before them, separated by different lags (like monster generations)

Trend and seasonality in ACF plots

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

Example

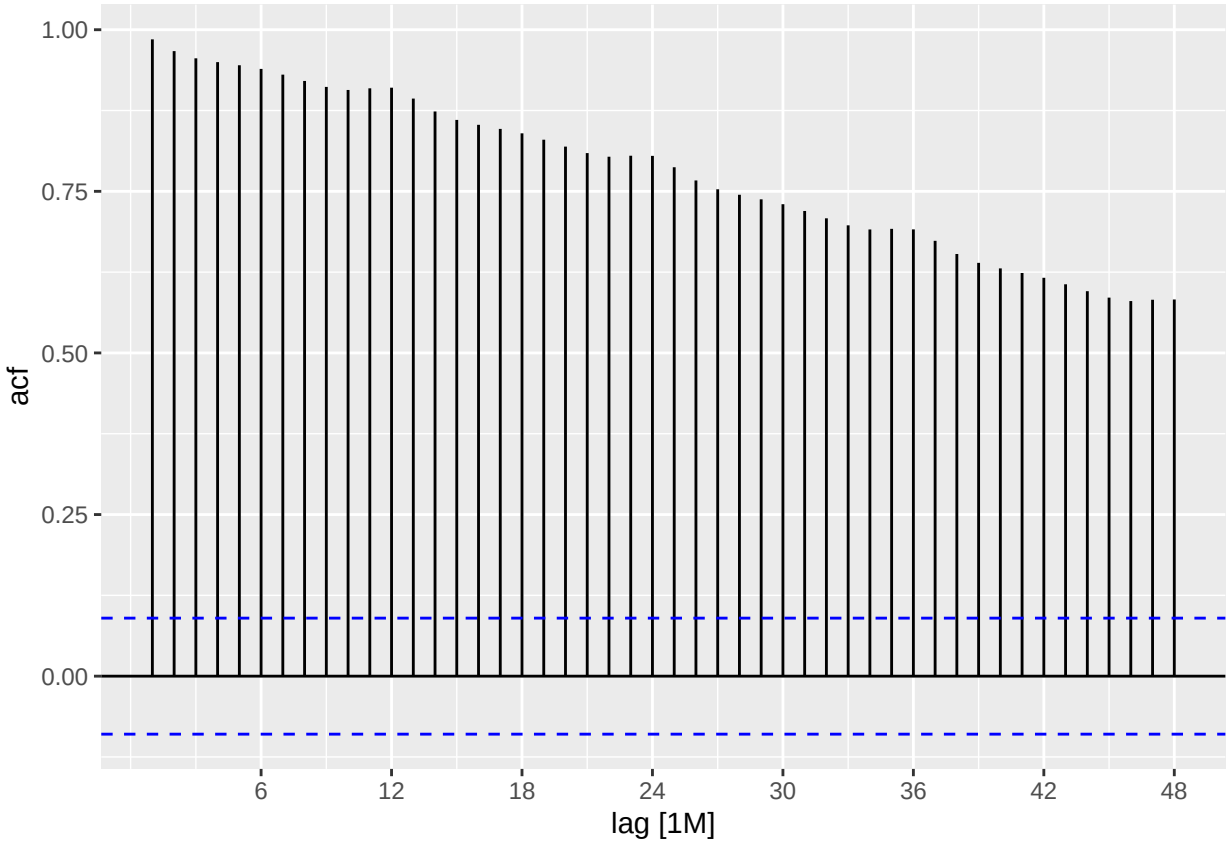
```
retail <- us_employment |>
  filter(Title=="Retail Trade",year(Month) >= 1980)
retail |> autoplot(Employed)
```



It is both trended and seasonal. There is also a strong correlation with a spike every year when the employment in retail trade peaks for a year.

If we do autocorrelation function here, we get,

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



Here, all the lags are showing positive correlation because of trend. Also there is a little bit peak at seasonal multiplier like 12,24,36,... months.

Example

Here for google stock price data, we don't find any seasonality at all.

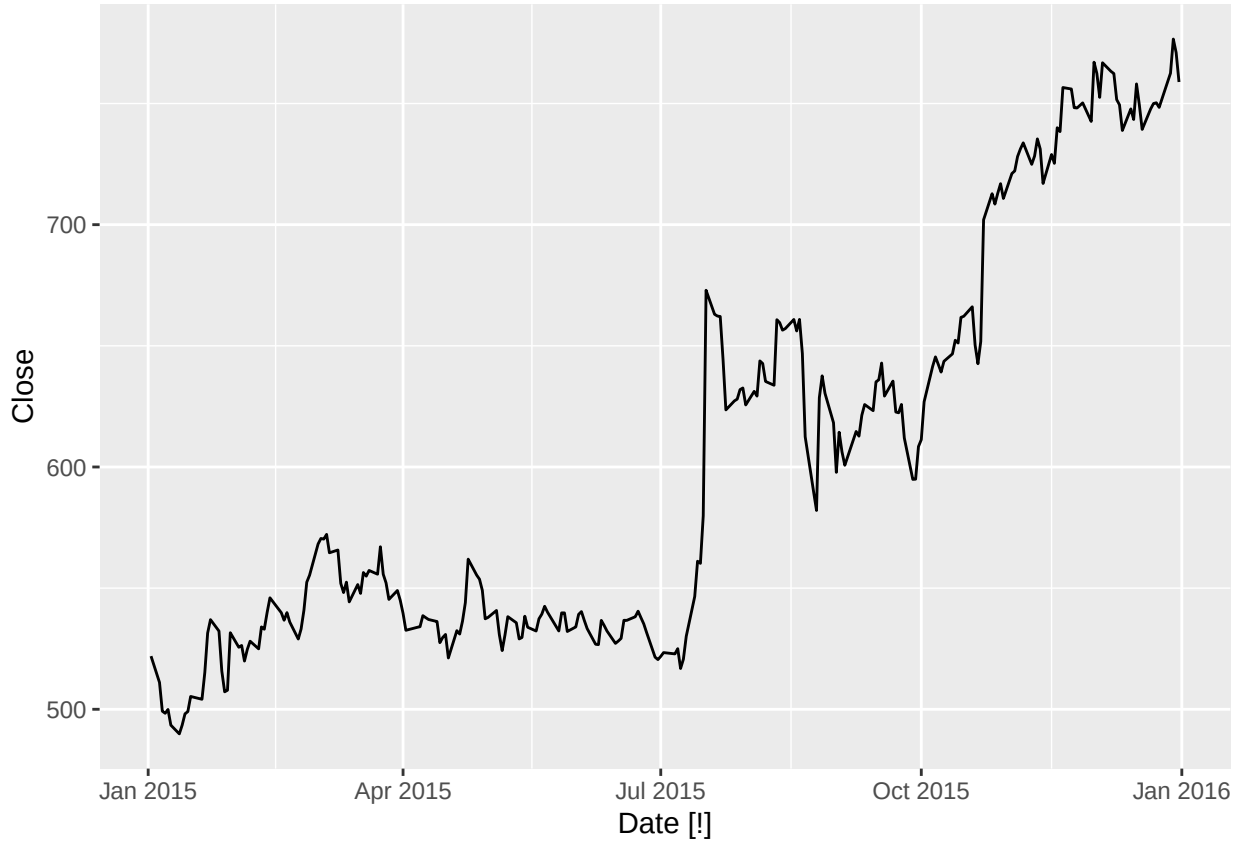
```
google_2015 <- gafa_stock |>
  filter(Symbol=="GOOG", year(Date) == 2015) |>
  select(Date,Close)
google_2015
```

```
## # 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.
## # i 242 more rows
```

This is not daily data because stocks are not traded daily due to holidays. "!" shows that this is irregular data.

```
google_2015 |> autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = Close`
```



It was trending upwards.

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

```
## Warning: Provided data has an irregular interval, results should be treated
## with caution. Computing ACF by observation.
```

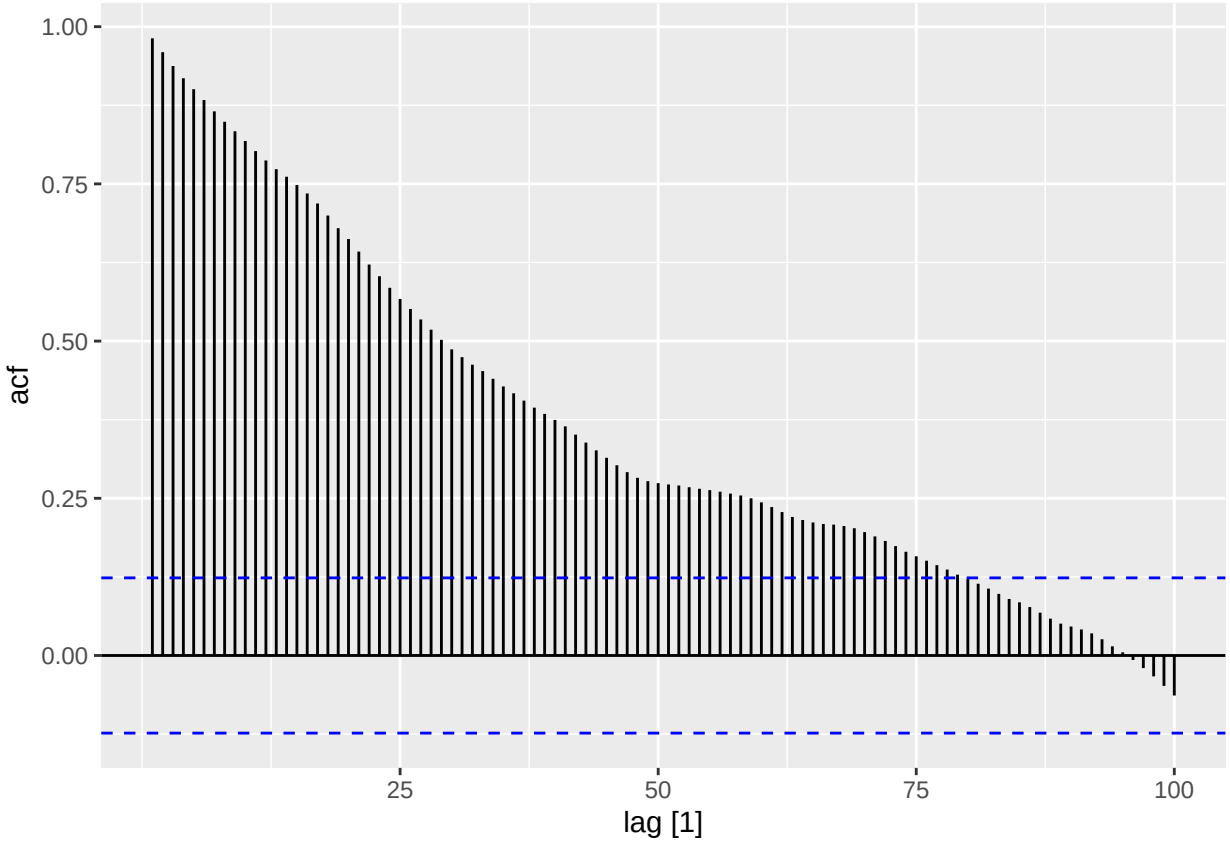
```
## # A tsibble: 100 x 2 [1]
##         lag   acf
##    <cf_lag> <dbl>
##  1        1 0.982
##  2        2 0.959
##  3        3 0.937
##  4        4 0.918
##  5        5 0.901
##  6        6 0.883
##  7        7 0.865
```

```
##  8        8 0.849
##  9        9 0.834
## 10       10 0.818
## # i 90 more rows
```

Here, acf values are positive and close to 1.

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

```
## Warning: Provided data has an irregular interval, results should be treated
## with caution. Computing ACF by observation.
```



It's for a typical stock price data and a typical trended data. It is still positive till lag 75.

Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time series.

There are several autocorrelation coefficients, corresponding to each panel in the lag plot. For example, r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2}, and so on.

The value of r_k can be written as:

$$r_k = \frac{\Sigma_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\Sigma_{t=1}^{T}(y_t - \bar{y})^2}$$

where T is the length of the time series. The autocorrelation coefficients make up the autocorrelation function or ACF.

We usually plot the ACF to see how the correlations change with the lag k. The plot is sometimes known as a correlogram.

When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in value. So the ACF of a trended time series tends to have positive values that slowly decrease as the lags increase.

When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal period) than for other lags.

When data are both trended and seasonal, you see a combination of these effects.

2.9 White noise

It is most boring time series. It is called white noise because it contains all the frequencies in the spectrum just like white light which contains all frequencies.

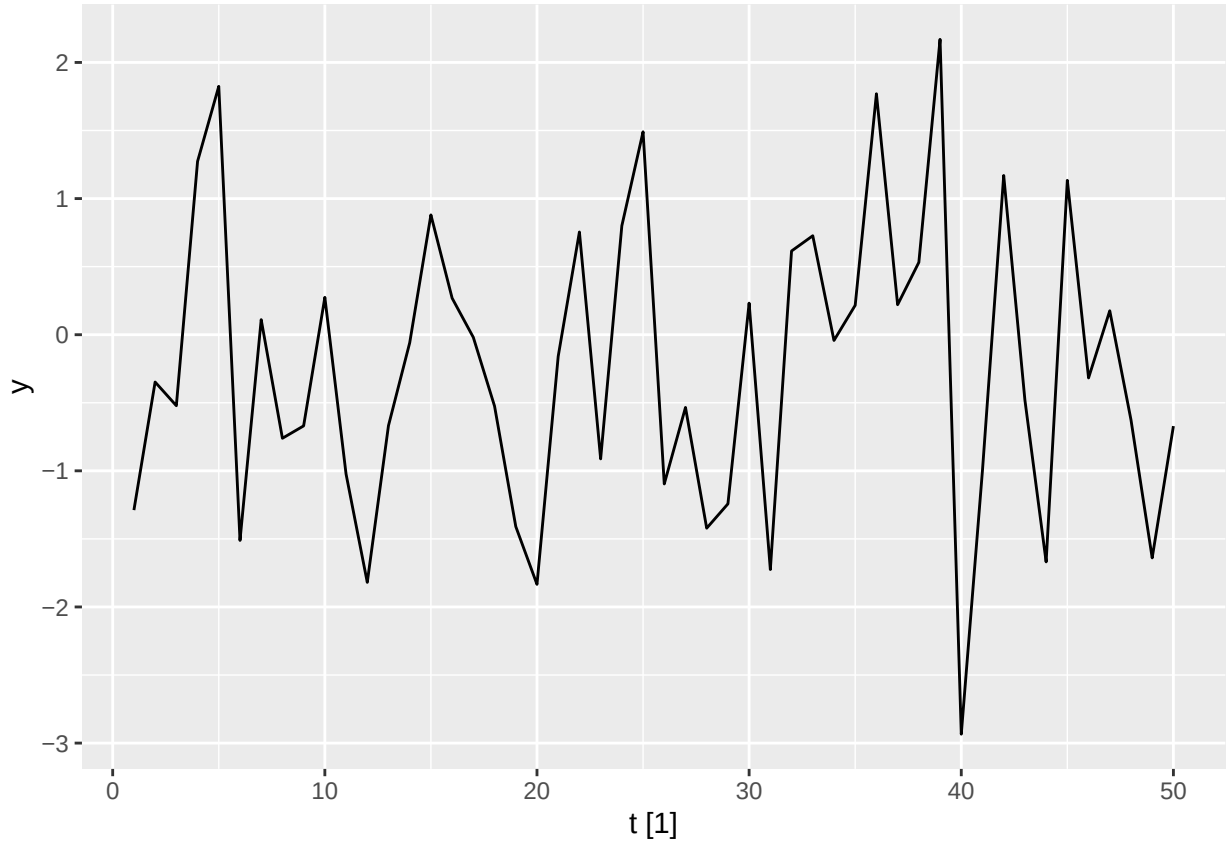
If you consider the time series as audio signal then the one that contains all the frequencies is called a white noise. it is same as the untuned radio sound "sssshhh".

In statistics, it is independent and identically distributed random values.

Example

Here, we generate randomly a white noise.

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



It is an example of white noise. There is no particular pattern. It's simply the random values.

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

```
wn |> ACF(y)
```

```
## # A tsibble: 16 x 2 [1]
##         lag     acf
##    <cf_lag>   <dbl>
##  1        1  0.0138
##  2        2 -0.163
##  3        3  0.163
##  4        4 -0.259
##  5        5 -0.198
##  6        6  0.0642
##  7        7 -0.139
##  8        8 -0.0316
##  9        9  0.199
## 10       10 -0.0240
## 11       11  0.0860
## 12       12  0.0790
## 13       13 -0.144
## 14       14 -0.0154
## 15       15 -0.0702
## 16       16 -0.266
```

There is no real correlation exist between consecutive values just because of randomness. Here for all the 16 lags , it is within the Blue line.

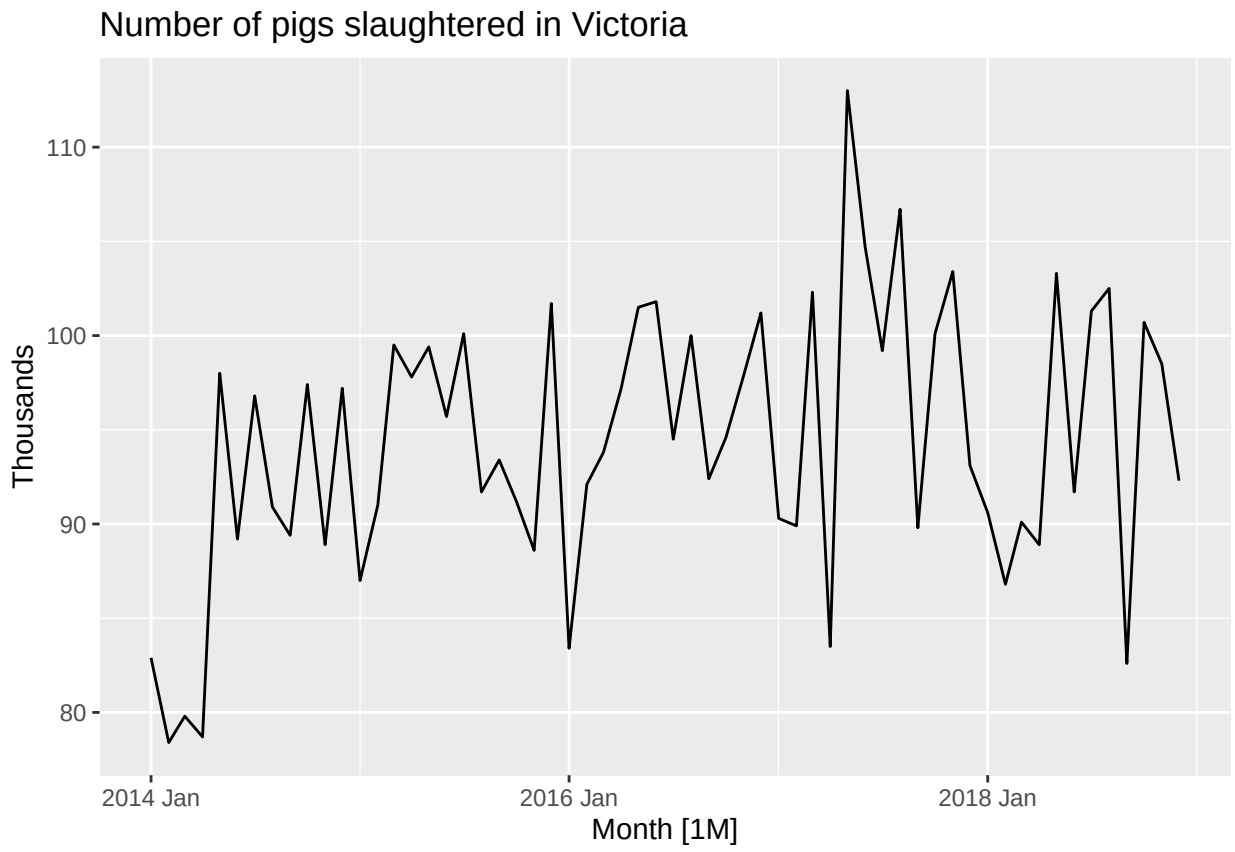
- Sample autocorrelations for white noise series.
- Expect each autocorrelations to be close to zero.
- Blue line shows 95% critical values. So 5% of the time, even with white noise data, we can get spikes outside those limits but 95% of the time it stays within the limits.

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 then the series is probably not WN.
- Common to plot lines at $\pm 1.96/\sqrt{T}$ when plotting ACF. These are the *critical values.*

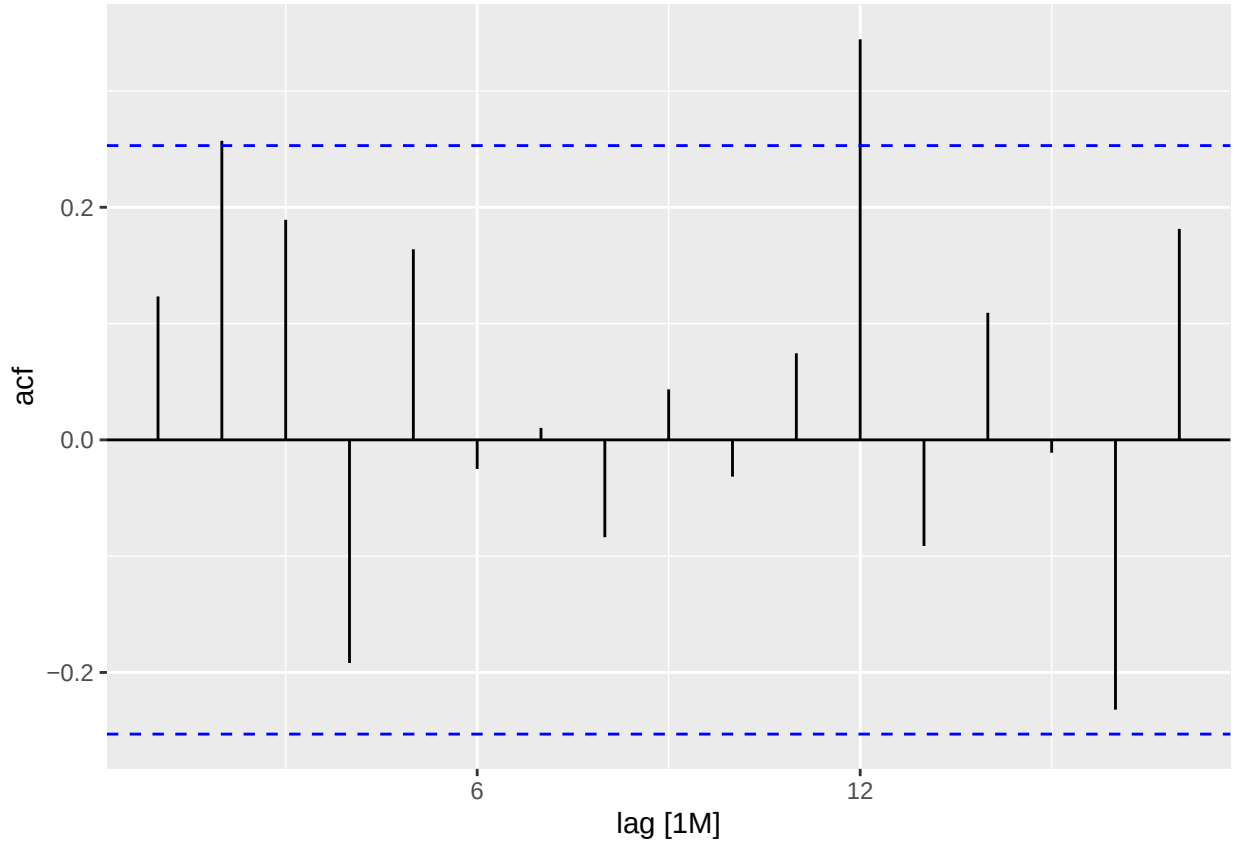
Example

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



Here from the time plot, it does not look like any trend or seasonality. It looks like random and we might think that it is a white noise but we really don't know unless we do a proper test using ACF plot.

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



There is actually two spikes in first 16 lags. There is 5% chance for each lag that spike could be outside of the range. Here, we are getting one out of 16 that's not surprising so ignore the spike at lag 2 because to get two of them it's getting a little less likely to occur. Also, at lag 12, it is seasonal lag so it suggests that there might be small amount of seasonality left in the series which we don't see in the time plot.

These shows that the series is *not a white noise series.*

Time series that show no autocorrelation are called white noise.

For white noise series, we expect each autocorrelation to be close to zero. Of course, they will not be exactly equal to zero as there is some random variation.

For a white noise series, we expect 95% of the spikes in the ACF to lie within $\pm\frac{2}{\sqrt{T}}$ where T is the length of the time series. It is common to plot these bounds on a graph of the ACF (the blue dashed lines above). If one or more large spikes are outside these bounds, or if substantially more than 5% of spikes are outside these bounds, then the series is probably not white noise.