Time series graphics

Lab Session 2

Dr.Priyanga Talagala

08/02/2026

Main packages required

```
# Data manipulation and plotting functions
library(tidyverse)
# Time series manipulation
library(tsibble)
# Tidy time series data
library(tsibbledata)
# Time series graphics and statistics
library(feasts)
# Forecasting functions
library(fable)
```

```
# All of the above
library(fpp3)
```

Install required packages

```
install.packages(c( "tidyverse", "fpp3"))
```

Time Series in R

```
library(tidyverse)
library(fpp3)
```

`tsibble` objects

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

Example 1

```
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
## # ... with 15,140 more rows
```

Example 2

```
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 South Australia Business  135.
##  2 1998 Q2 Adelaide South Australia Business  110.
##  3 1998 Q3 Adelaide South Australia Business  166.
##  4 1998 Q4 Adelaide South Australia Business  127.
##  5 1999 Q1 Adelaide South Australia Business  137.
##  6 1999 Q2 Adelaide South Australia Business  200.
##  7 1999 Q3 Adelaide South Australia Business  169.
##  8 1999 Q4 Adelaide South Australia Business  134.
##  9 2000 Q1 Adelaide South Australia Business  154.
## 10 2000 Q2 Adelaide South Australia Business  169.
## # ... with 24,310 more rows
```

The `tsibble` index

```
set.seed(1)
ts <- tsibble(t = seq(36), y = rnorm(36), index = t)
ts
```

```
## # A tsibble: 36 x 2 [1]
##        t      y
##    <int>  <dbl>
##  1     1 -0.626
##  2     2  0.184
##  3     3 -0.836
##  4     4  1.60
##  5     5  0.330
##  6     6 -0.820
##  7     7  0.487
##  8     8  0.738
##  9     9  0.576
## 10    10 -0.305
## # ... with 26 more rows
```

```
mydata <- tsibble(
  year = 2016:2020,
  y = c(123, 39, 78, 52, 110),
  index = year
)

mydata
```

```
## # A tsibble: 5 x 2 [1Y]
##    year     y
##   <int> <dbl>
## 1  2016   123
## 2  2017    39
## 3  2018    78
## 4  2019    52
## 5  2020   110
```

tibble vs tsibble

```
#tibble
mytibble <- tibble(
  year = 2012:2016,
  y = c(123, 39, 78, 52, 110)
)
mytibble
```

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

```
mytsibble <- mytibble |> as_tsibble(index = year)

mytsibble
```

Pipe |> Operator

- The pipe operator will forward a value, or the result of an expression, into the next function call/expression.
- For instance a function to filter data can be written as:

```
head(as_tsibble(mytibble, index = year))
```

or

```
mytibble |> as_tsibble(index = year) |> head()
```

- Both functions complete the same task
- It improves the readability and clarity
- We read the |> operator as "and then"
- Example: "take mytibble *and then* coerce to a tsibble object *and then* return the first part of the object"

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

```
z <- tibble(
  Month = paste(2019, month.abb[1:5]),
  Observation = c(50, 23, 34, 30, 25))
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
```

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

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

- Common time index variables can be created with these functions:

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

Working with `tsibble` objects

```
# Monthly Medicare Australia prescription data

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-pa~ A     Alimentary~ A01   STOMATOLO~   18228 67877
##  2 1991 Aug Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   15327 57011
##  3 1991 Sep Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   14775 55020
##  4 1991 Oct Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   15380 57222
##  5 1991 Nov Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   14371 52120
##  6 1991 Dec Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   15028 54299
##  7 1992 Jan Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   11040 39753
```

```
##  8 1992 Feb Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   15165 54405
##  9 1992 Mar Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   16898 61108
## 10 1992 Apr Concessional Co-pa~ A     Alimentary~ A01   STOMATOLO~   18141 65356
## # ... with 67,586 more rows
```

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-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
```

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

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

```
## # A tsibble: 816 x 4 [1M]
## # Key:       Concession, Type [4]
##       Month Concession   Type           Cost
##       <mth> <chr>        <chr>         <dbl>
##  1 1991 Jul Concessional Co-payments 2092878
##  2 1991 Aug Concessional Co-payments 1795733
##  3 1991 Sep Concessional Co-payments 1777231
##  4 1991 Oct Concessional Co-payments 1848507
##  5 1991 Nov Concessional Co-payments 1686458
##  6 1991 Dec Concessional Co-payments 1843079
##  7 1992 Jan Concessional Co-payments 1564702
##  8 1992 Feb Concessional Co-payments 1732508
##  9 1992 Mar Concessional Co-payments 2046102
## 10 1992 Apr Concessional Co-payments 2225977
## # i 806 more rows
```

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

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

```
## # A tsibble: 204 x 2 [1M]
```

```
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul    3526591
##  2 1991 Aug    3180891
##  3 1991 Sep    3252221
##  4 1991 Oct    3611003
##  5 1991 Nov    3565869
##  6 1991 Dec    4306371
##  7 1992 Jan    5088335
##  8 1992 Feb    2814520
##  9 1992 Mar    2985811
## 10 1992 Apr    3204780
## # i 194 more rows
```

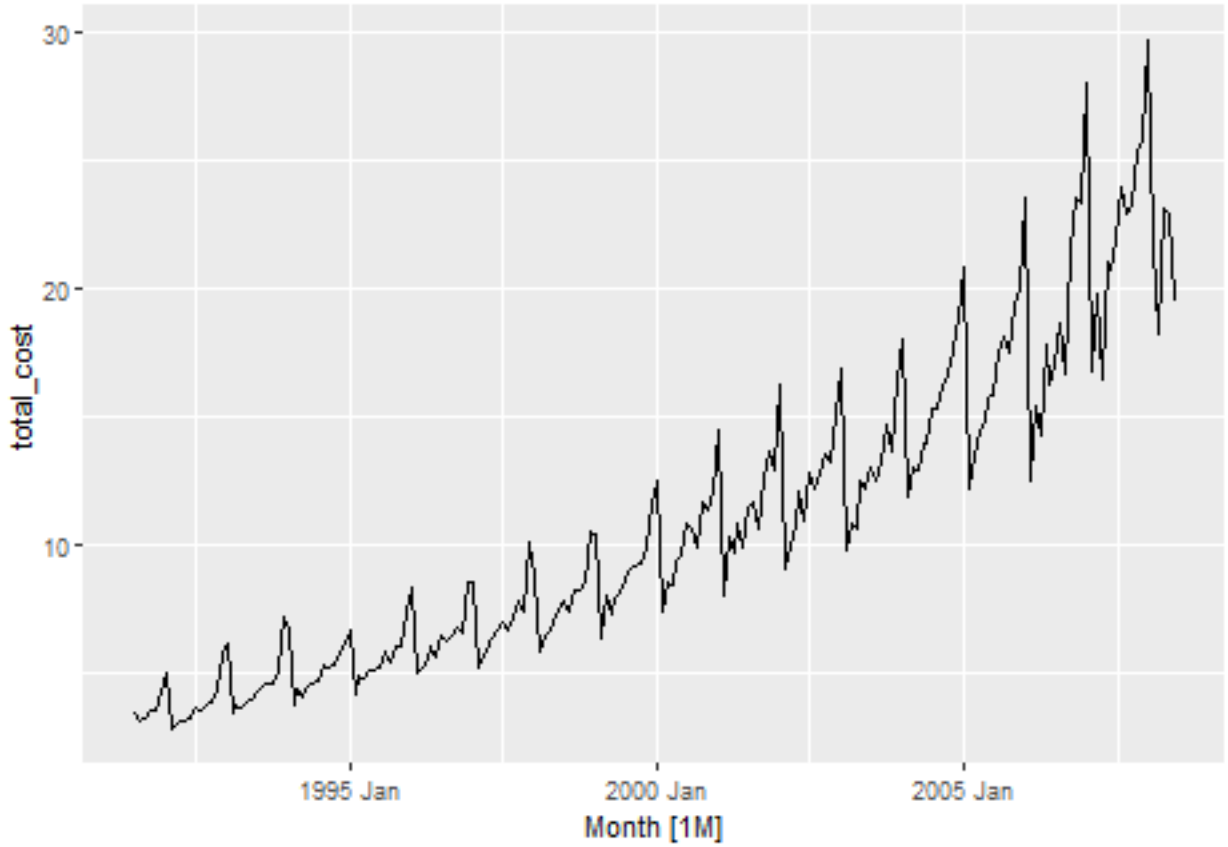
We can use the `mutate()` function to create new variables

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

a10
```

```
## # A tsibble: 204 x 2 [1M]
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul       3.53
##  2 1991 Aug       3.18
##  3 1991 Sep       3.25
##  4 1991 Oct       3.61
##  5 1991 Nov       3.57
##  6 1991 Dec       4.31
##  7 1992 Jan       5.09
##  8 1992 Feb       2.81
##  9 1992 Mar       2.99
## 10 1992 Apr       3.20
## # i 194 more rows
```

Time plots

```
# autoplot() uses ggplot2 to draw a particular plot
# for an object of a particular class in a single command

a10 |> autoplot(total_cost)
```

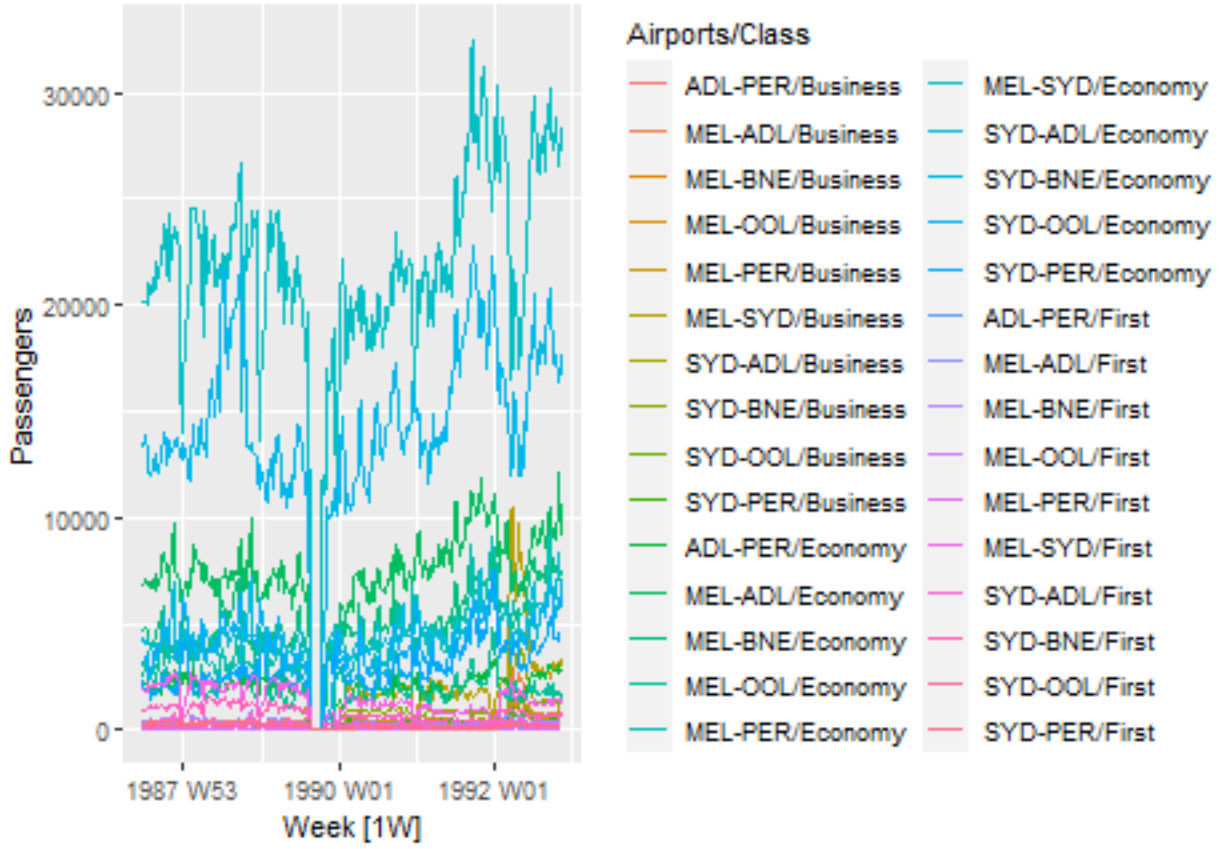


Example 2: Ansett airlines

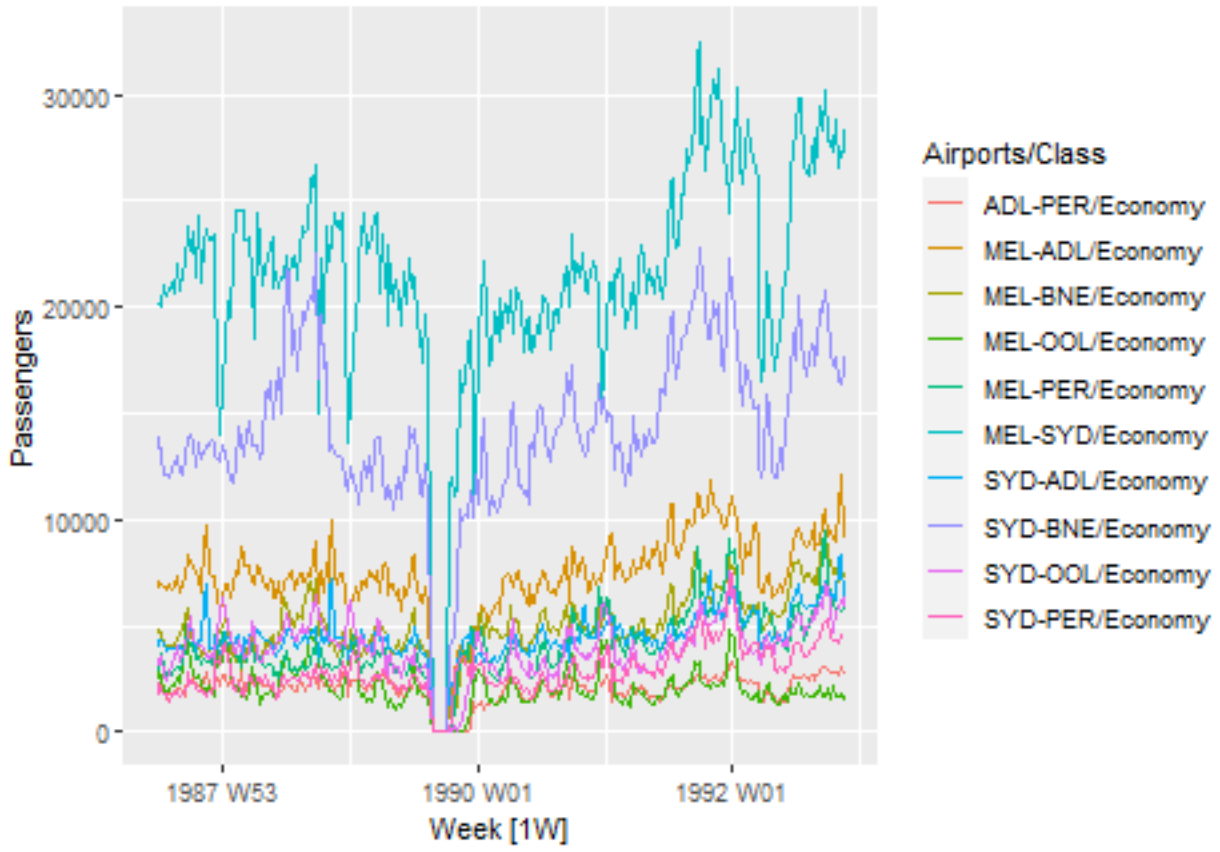
```
ansett
```

```
## # 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
```

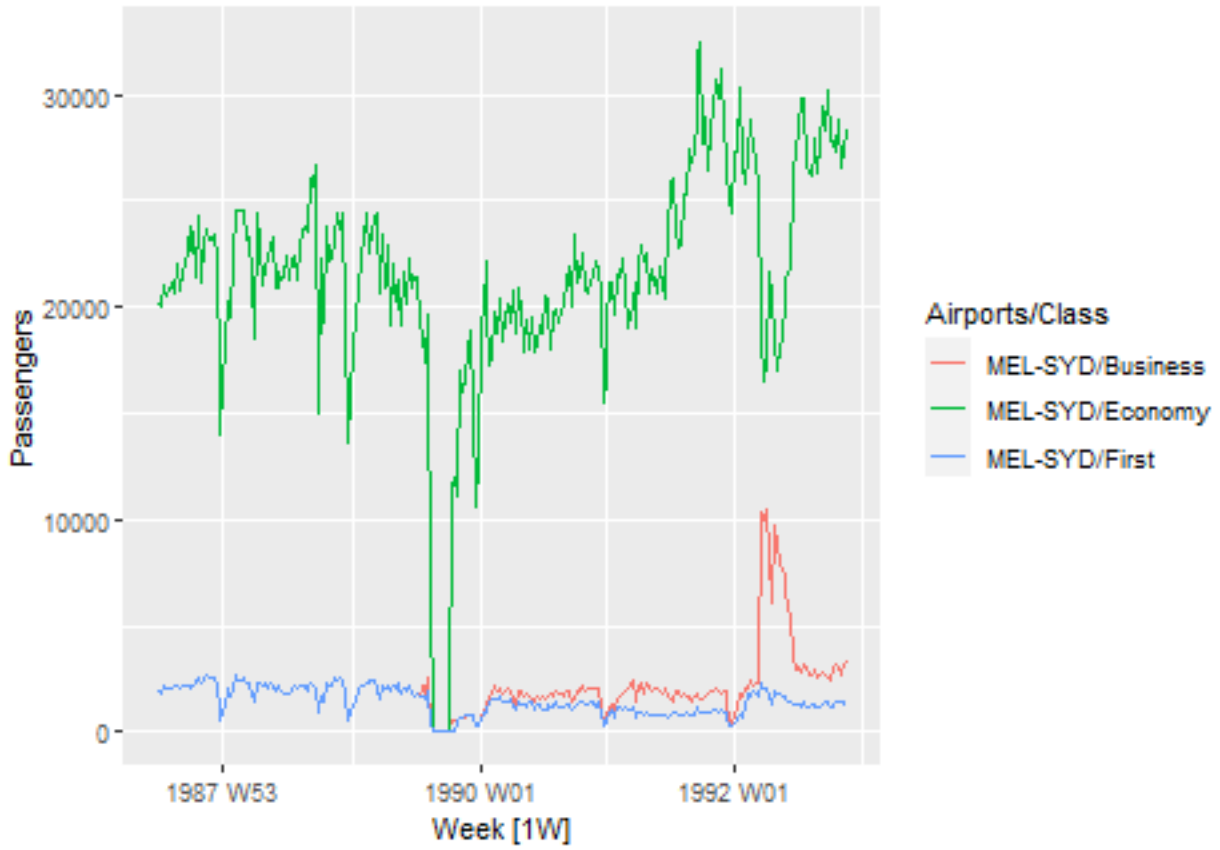
```
ansett |> autoplot(Passengers)
```



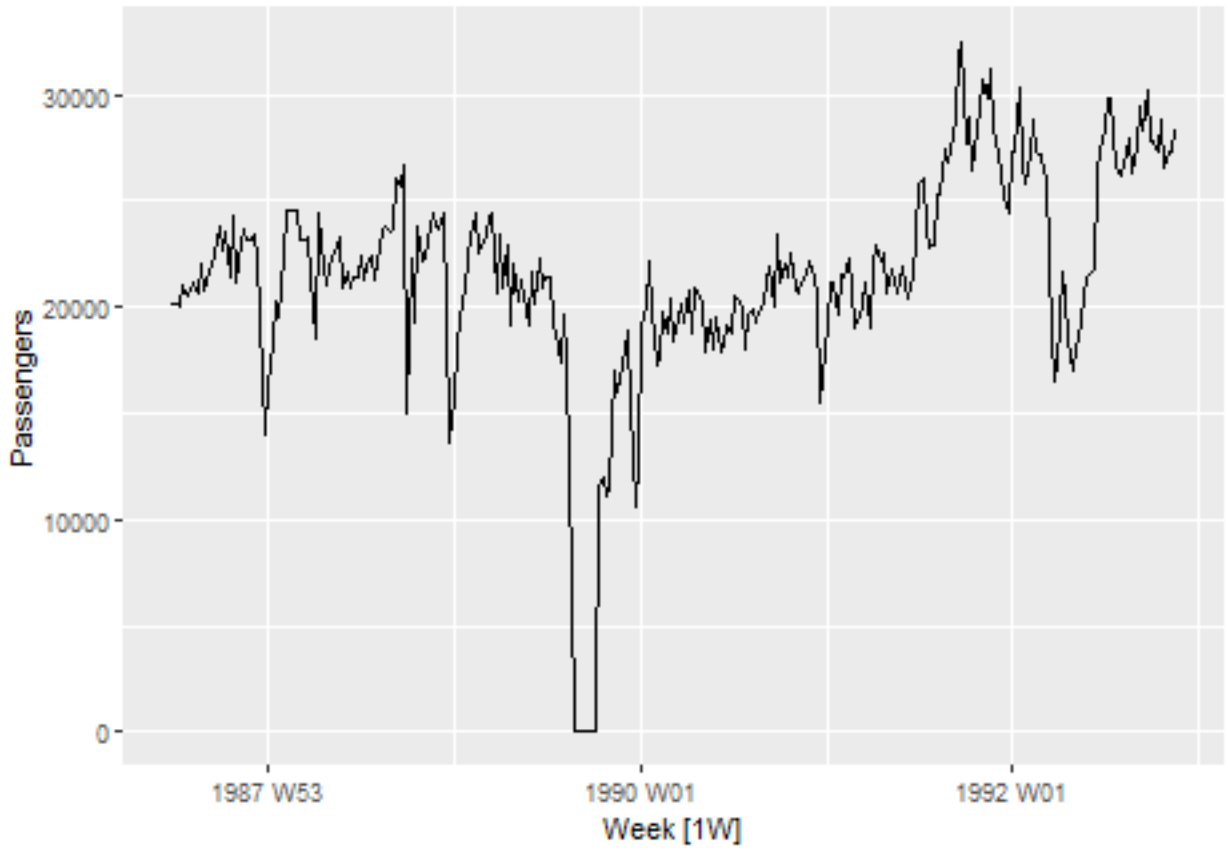

```
ansett |>
filter(Class == "Economy") |> autoplot(Passengers)
```



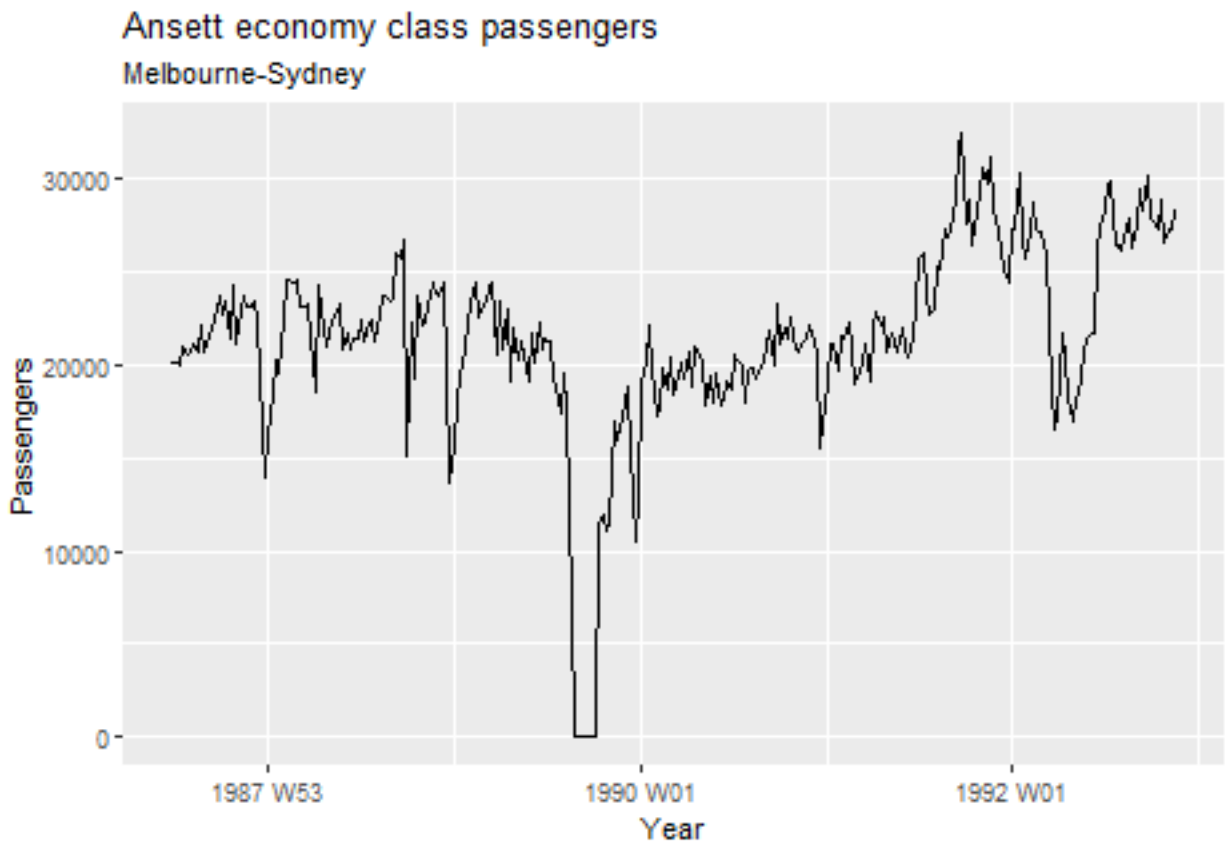

```
ansett |>
filter(Airports == "MEL-SYD") |> autoplot(Passengers)
```




```
ansett |>
filter(Airports == "MEL-SYD", Class == "Economy") |> autoplot(Passengers)
```




```
ansett |>
filter(Airports == "MEL-SYD", Class == "Economy") |> autoplot(Passengers)+
  labs(title = "Ansett economy class passengers", subtitle = "Melbourne-Sydney")+
  xlab("Year")
```

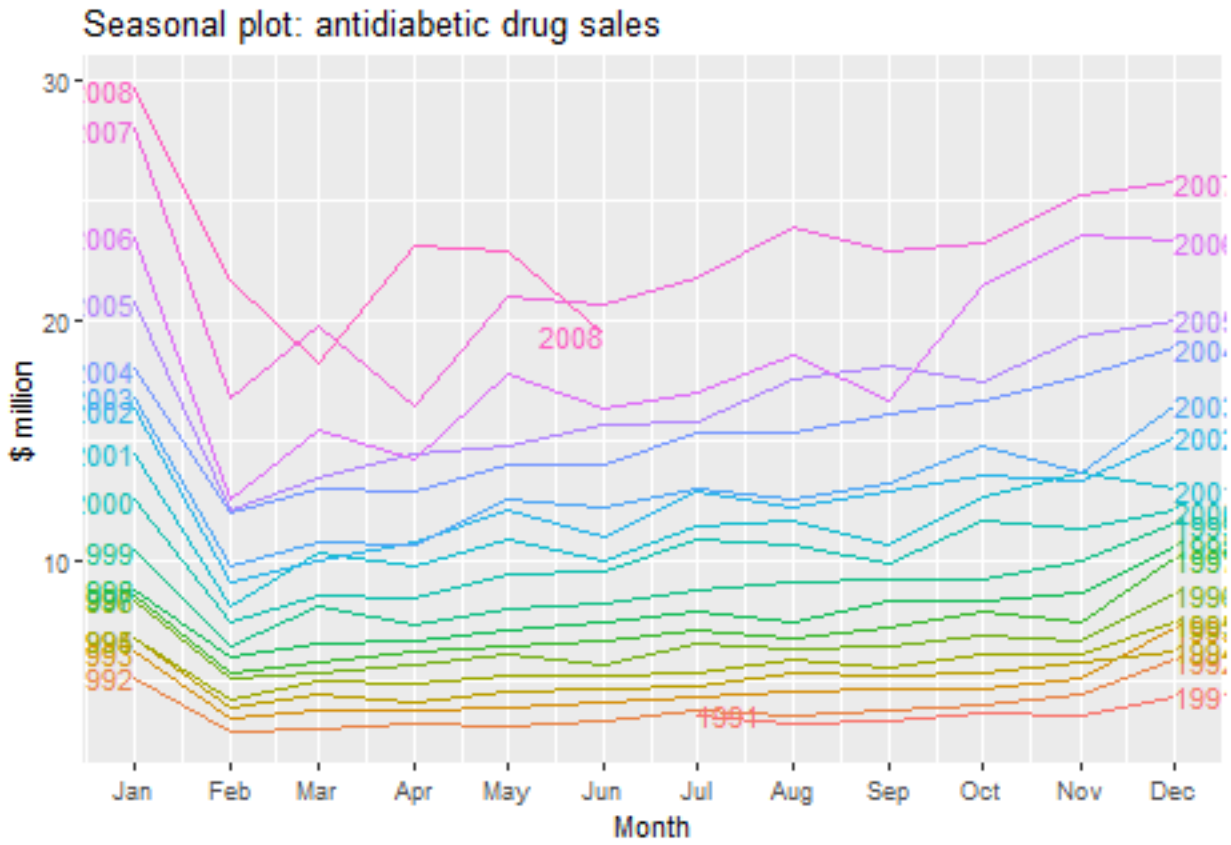


Seasonal plots

```
a10
```

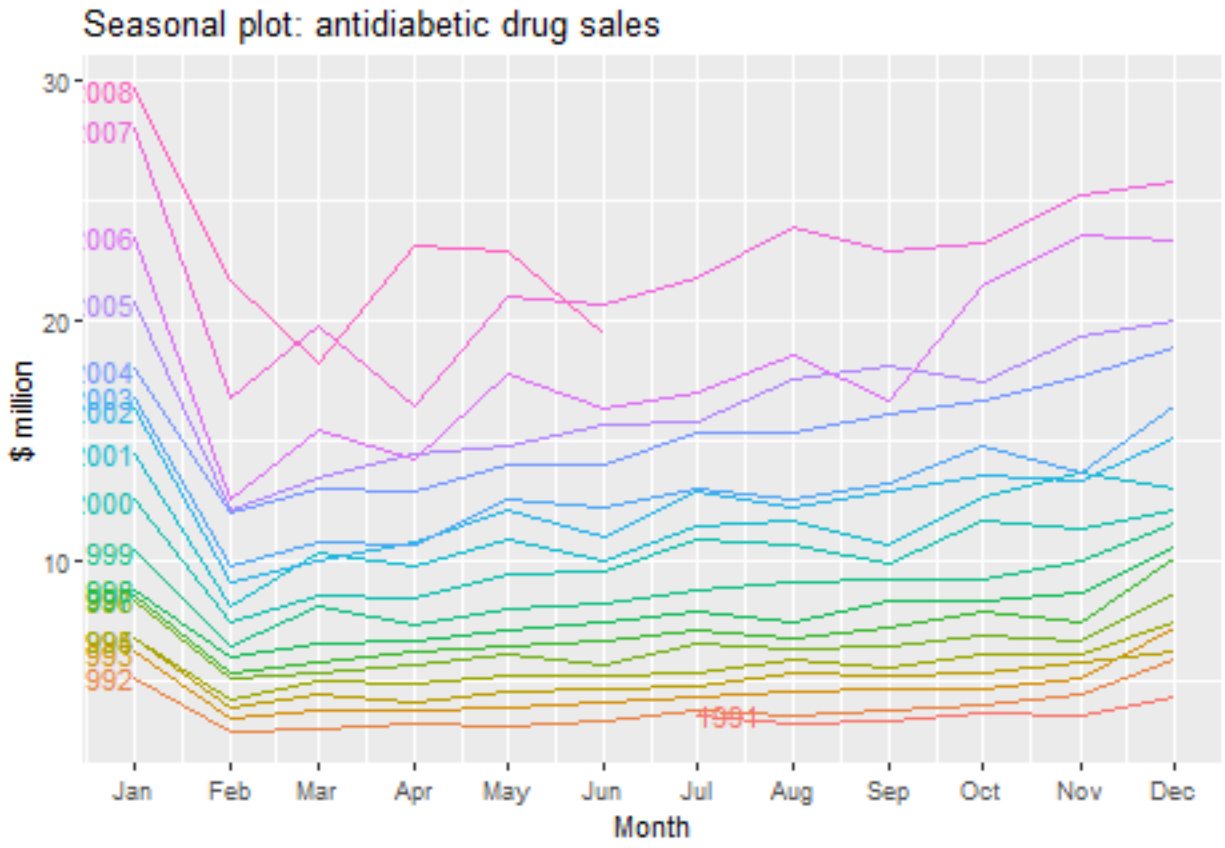
```
## # A tsibble: 204 x 2 [1M]
##       Month total_cost
##       <mth>      <dbl>
##  1 1991 Jul       3.53
##  2 1991 Aug       3.18
##  3 1991 Sep       3.25
##  4 1991 Oct       3.61
##  5 1991 Nov       3.57
##  6 1991 Dec       4.31
##  7 1992 Jan       5.09
##  8 1992 Feb       2.81
##  9 1992 Mar       2.99
## 10 1992 Apr       3.20
## # i 194 more rows
```

```
a10 |> gg_season(total_cost, labels = "both") +
  ylab("$ million") +
  ggtitle("Seasonal plot: antidiabetic drug sales")
```



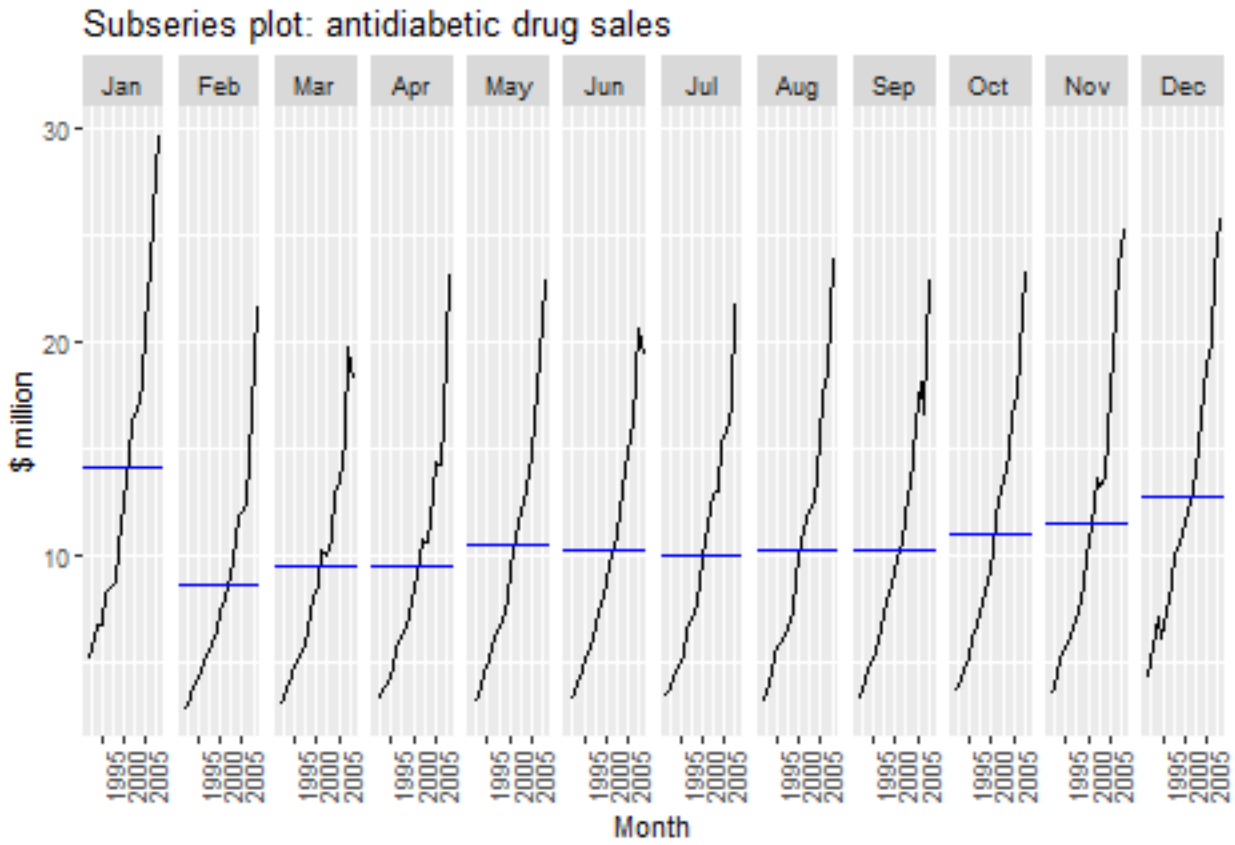
- Data plotted against the individual "seasons" in which the data were observed. (In this case a "season" is a month.)
- Something like a time plot except that the data from each season are overlapped.
- Enables the underlying seasonal pattern to be seen more clearly, and also allows any substantial departures from the seasonal pattern to be easily identified.
- In R: `gg_season()`

```
# labels: Position of the labels for seasonal period identifier.

a10 |> gg_season(total_cost, labels = "left") +
  ylab("$ million") +
  ggtitle("Seasonal plot: antidiabetic drug sales")
```



Seasonal subseries plots

```
a10 |>
gg_subseries(total_cost) + ylab("$ million") + ggtitle("Subseries plot: antidiabetic drug sales")
```



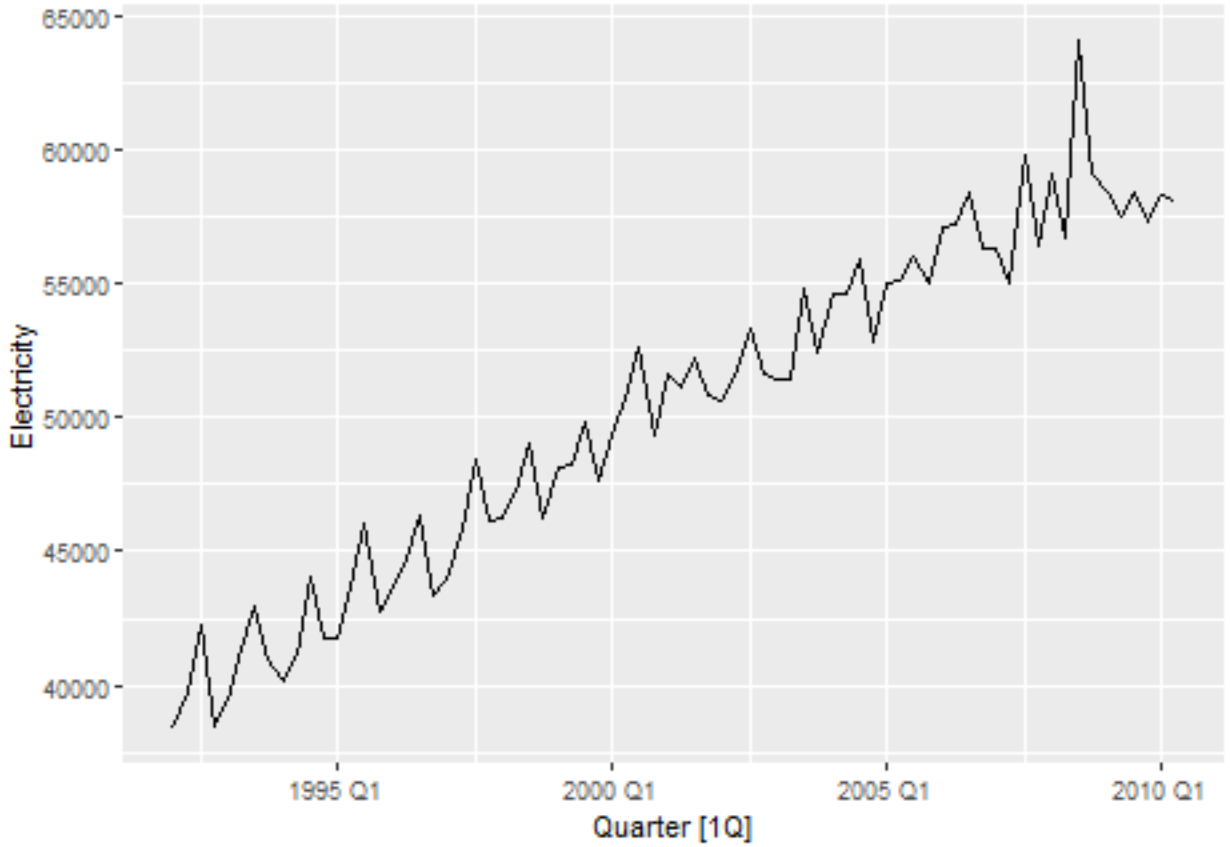
- Data for each season collected together in time plot as separate time series.
- Enables the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.
- In R: `gg_subseries()`

Example 2: Quarterly Australian Electricity Production

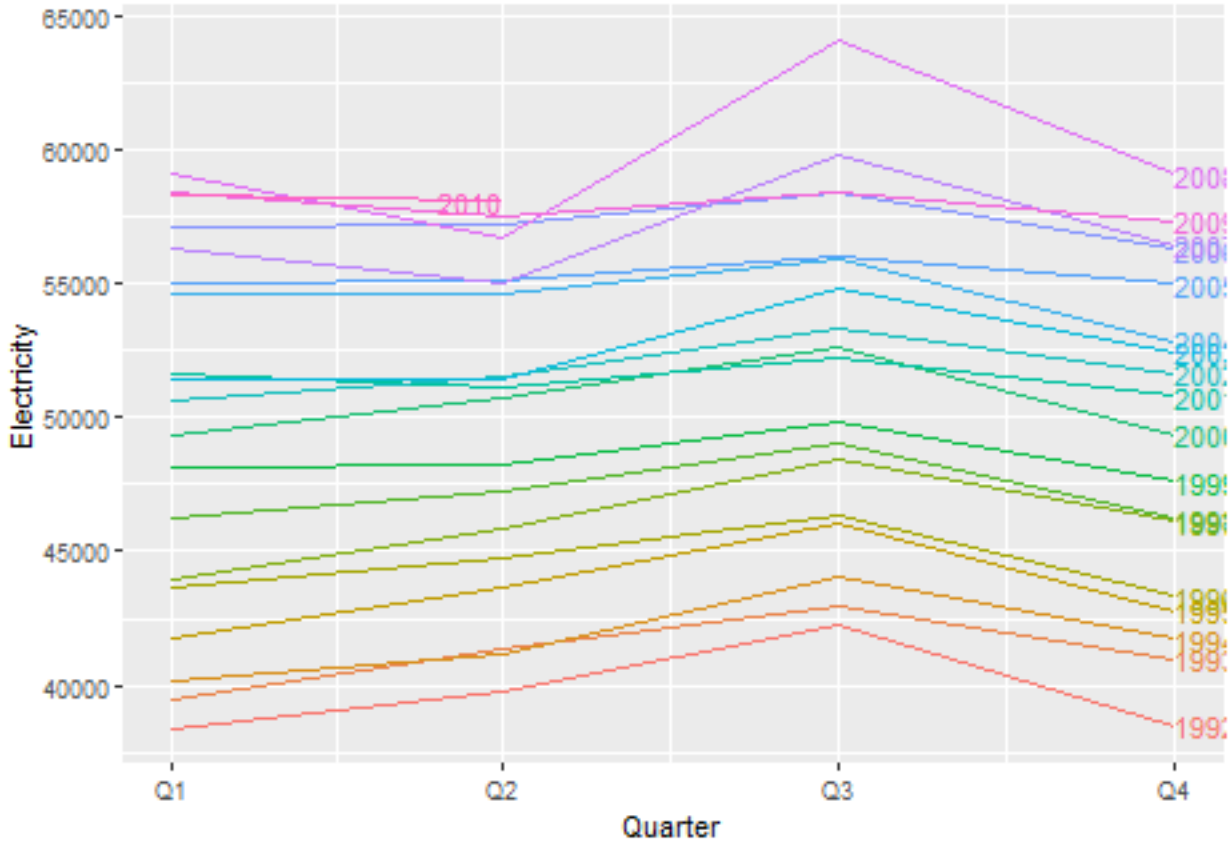
```
elec <- aus_production |> select(Quarter, Electricity) |> filter(year(Quarter) >= 1992)
elec
```

```
## # A tsibble: 74 x 2 [1Q]
##    Quarter Electricity
##      <qtr>       <dbl>
##  1 1992 Q1       38332
##  2 1992 Q2       39774
##  3 1992 Q3       42246
##  4 1992 Q4       38498
##  5 1993 Q1       39460
##  6 1993 Q2       41356
##  7 1993 Q3       42949
##  8 1993 Q4       40974
##  9 1994 Q1       40162
## 10 1994 Q2       41199
## # i 64 more rows
```

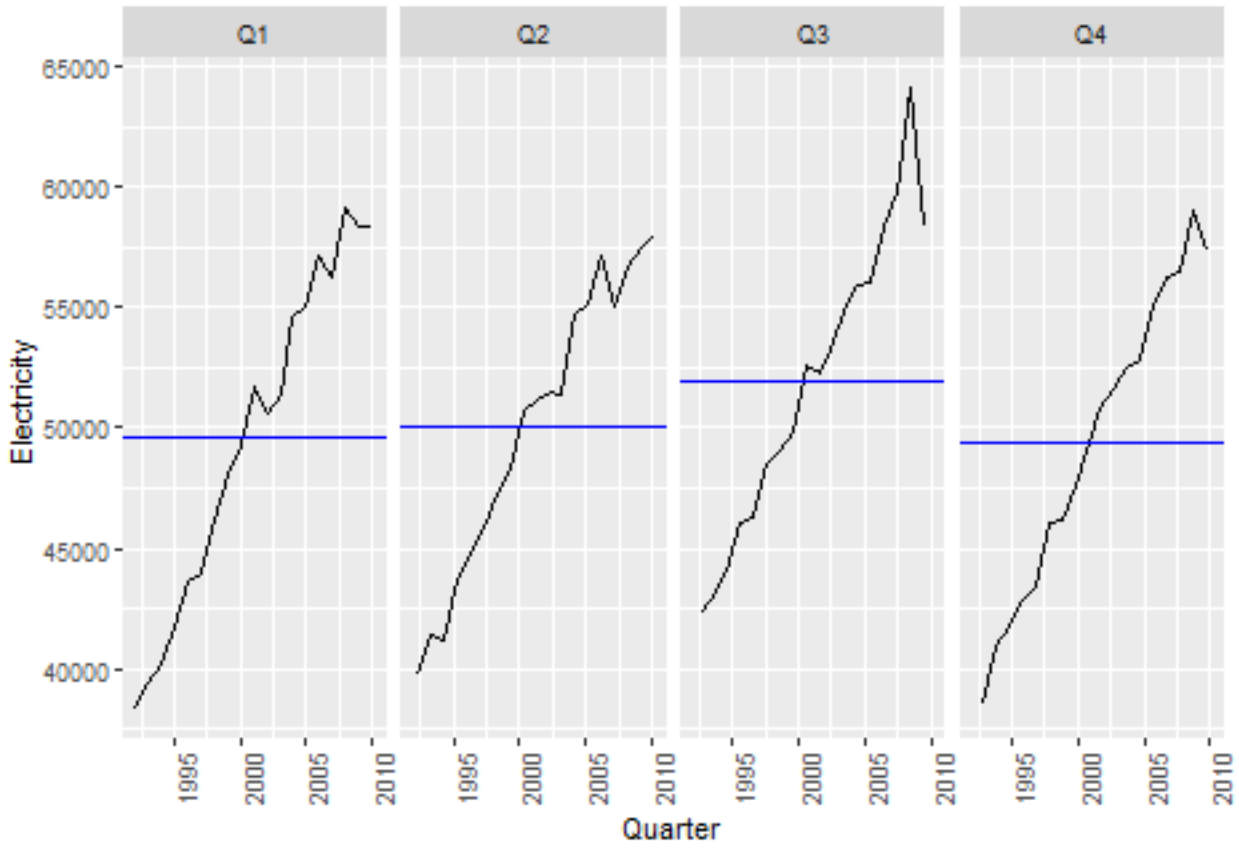
```
elec |> autoplot(Electricity)
```




```
elec |> gg_season(Electricity, labels="right")
```




```
elec |> gg_subseries(Electricity)
```

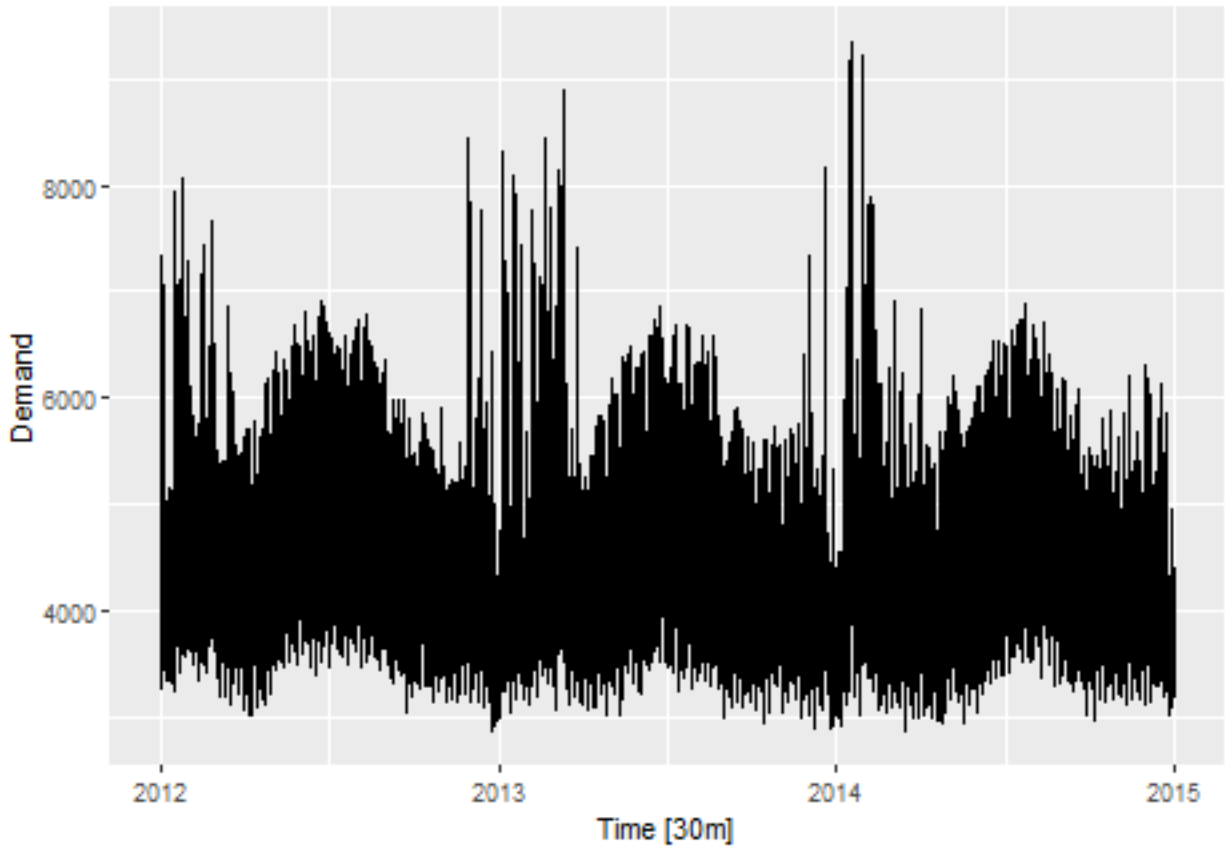


Multiple seasonal periods

```
vic_elec
```

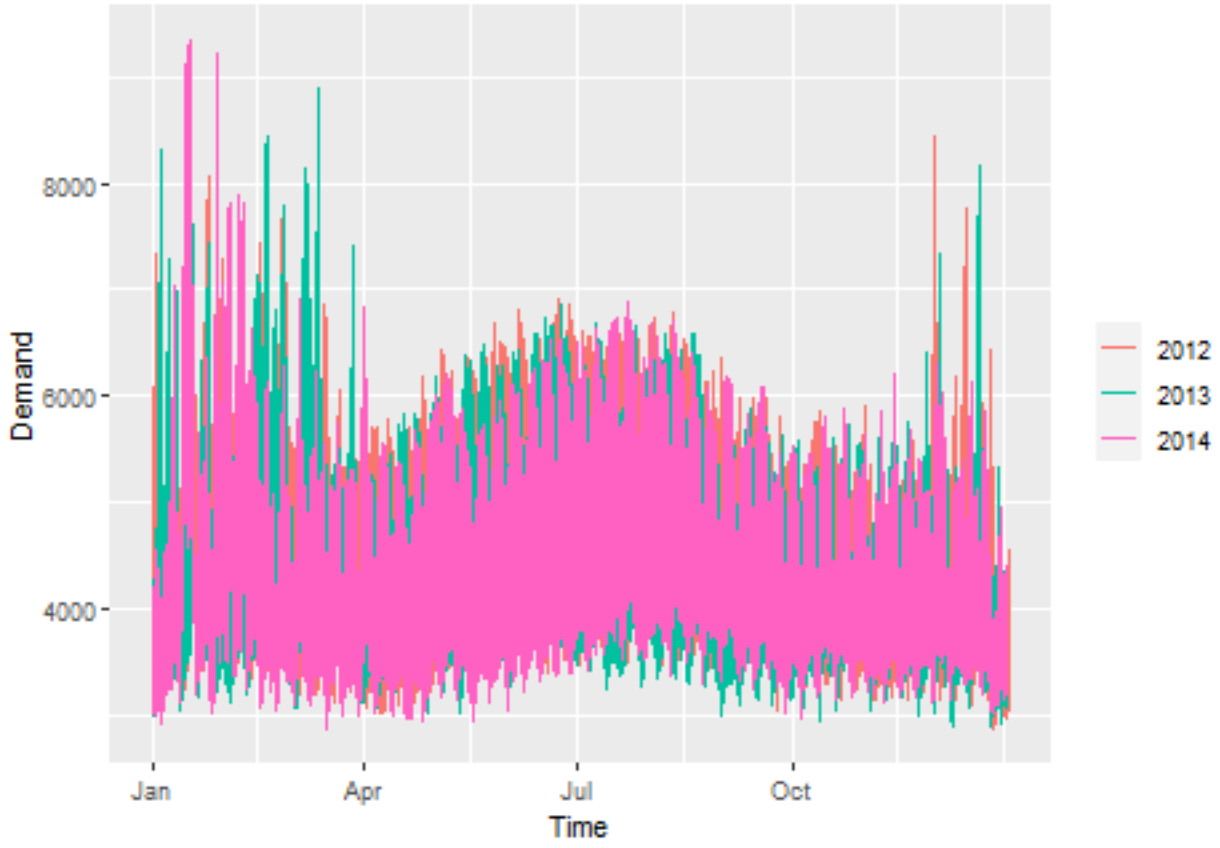
```
## # A tsibble: 52,608 x 5 [30m] <Australia/Melbourne>
##    Time                Demand Temperature Date       Holiday
##    <dttm>               <dbl>       <dbl> <date>     <lgl>
##  1 2012-01-01 00:00:00  4383.        21.4 2012-01-01 TRUE
##  2 2012-01-01 00:30:00  4263.        21.0 2012-01-01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

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

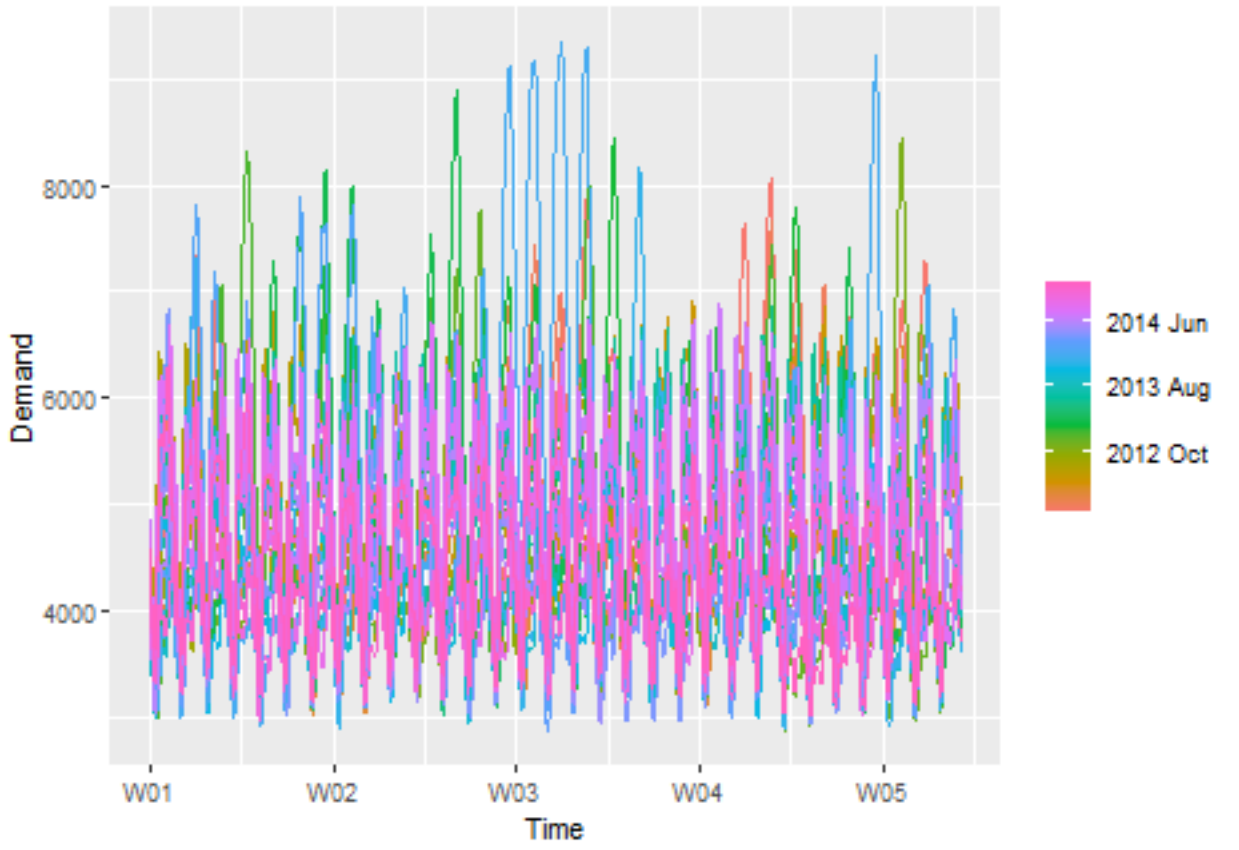



```
vic_elec |> gg_season(Demand)
```

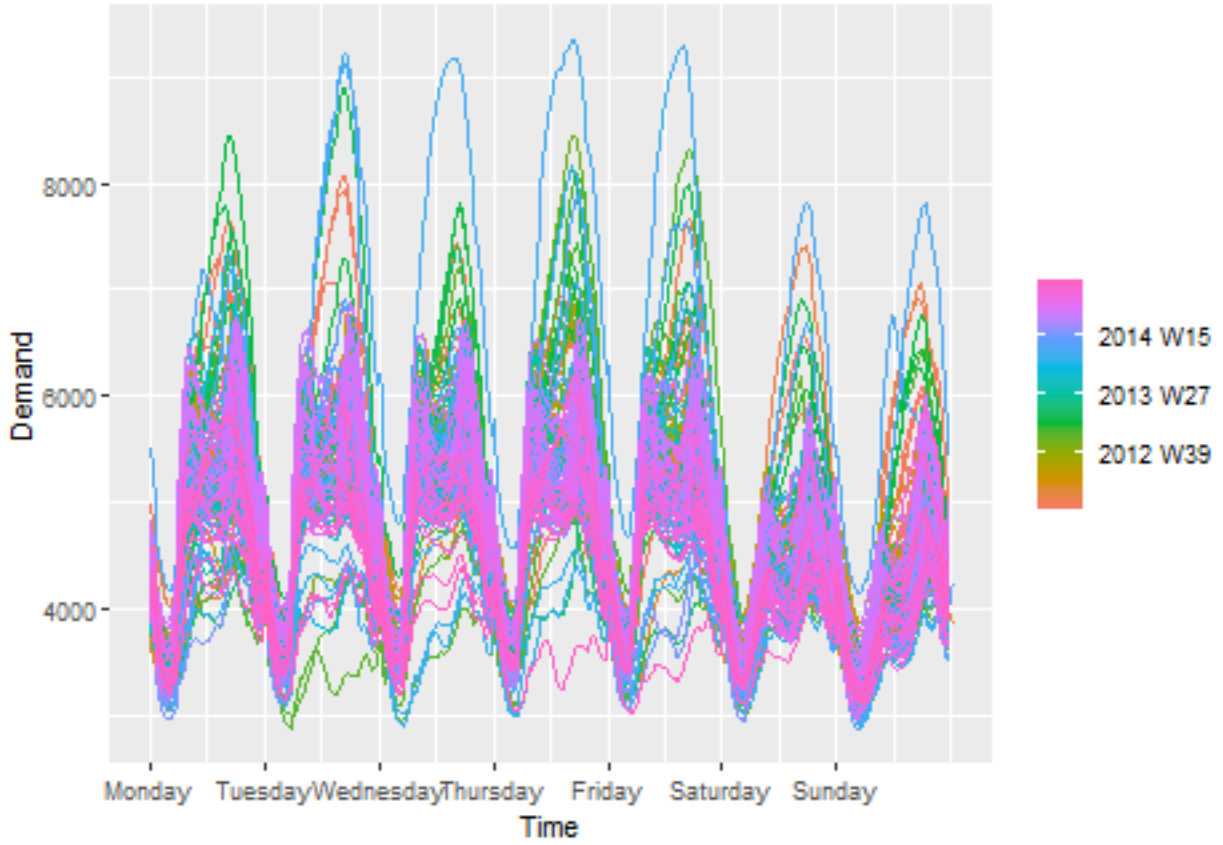

```
vic_elec |> gg_season(Demand, period = "year")
```



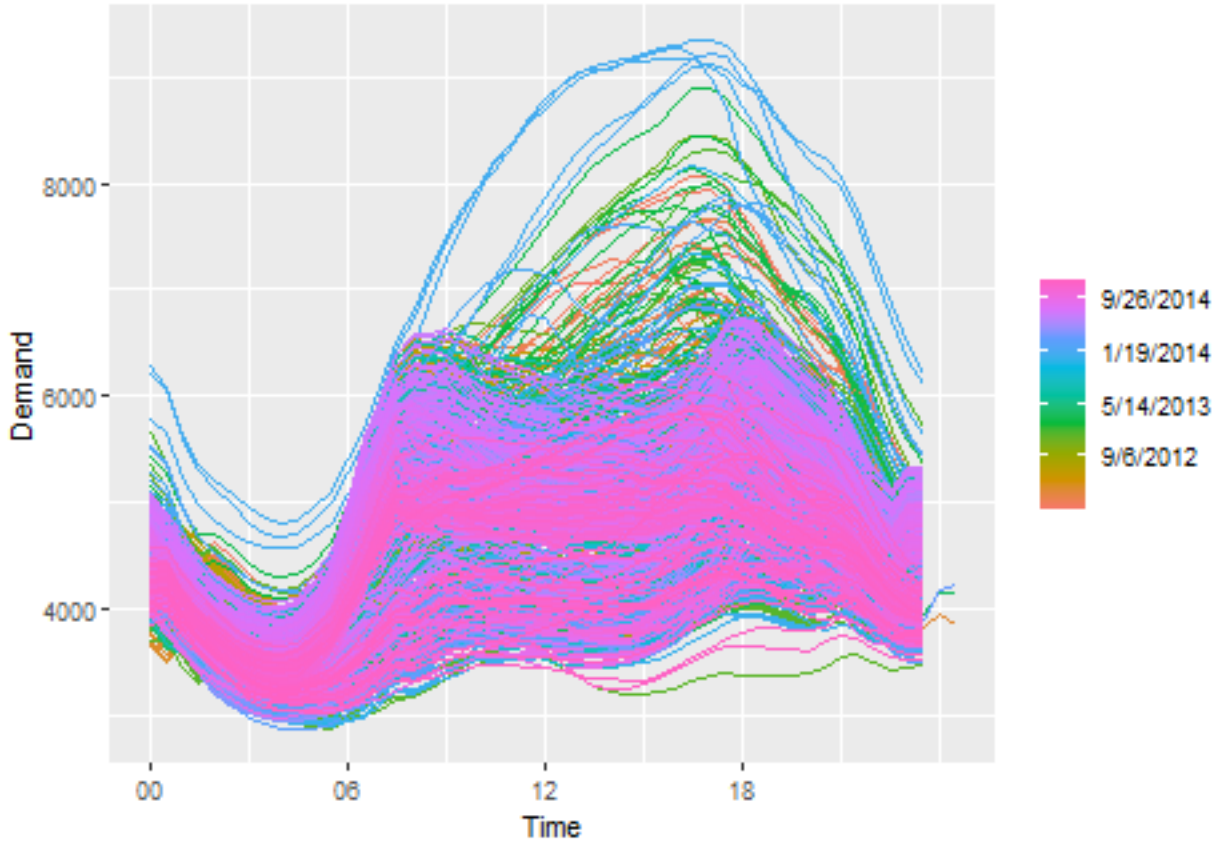

```
vic_elec |> gg_season(Demand, period = "month")
```




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




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



Example 2: Australian holidays

```
tourism
```

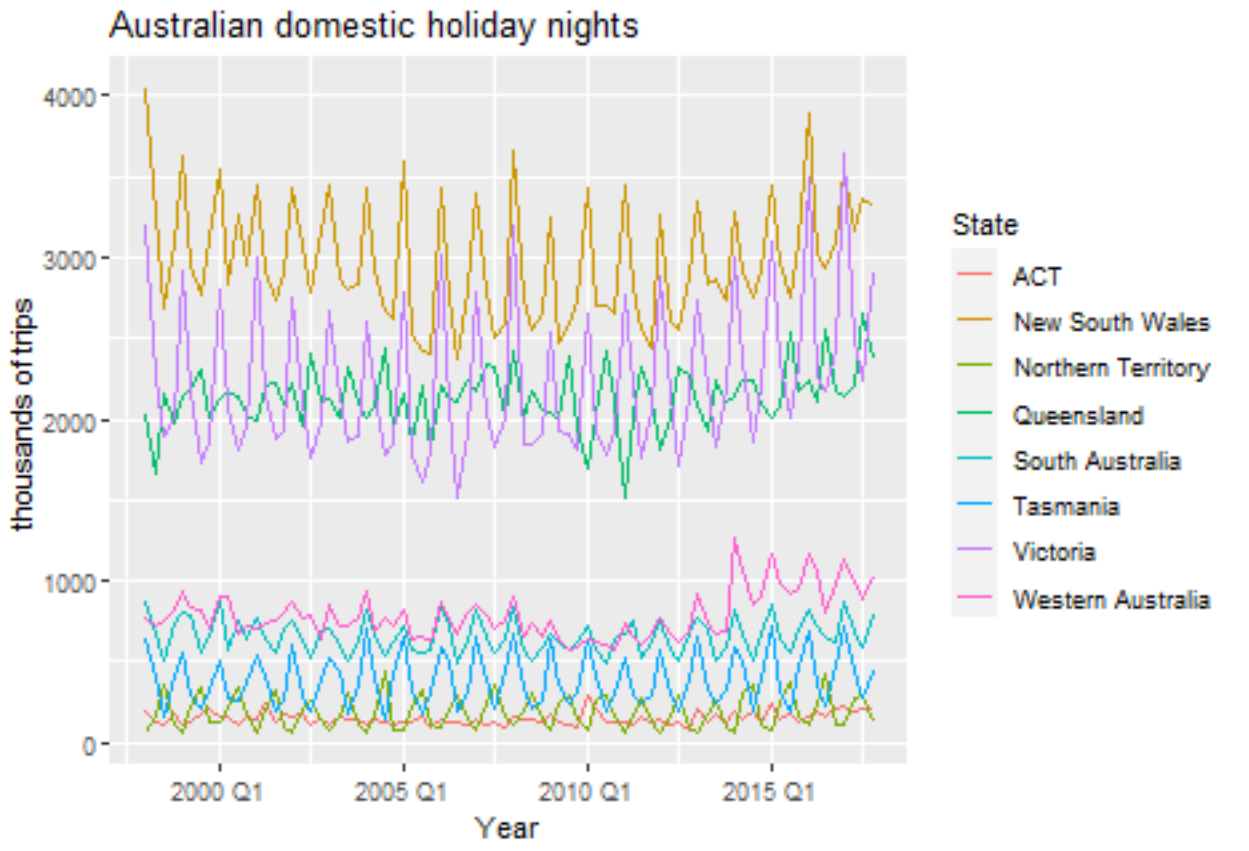
```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:       Region, State, Purpose [304]
##    Quarter Region   State           Purpose  Trips
##      <qtr> <chr>    <chr>           <chr>    <dbl>
##  1 1998 Q1 Adelaide South Australia Business  135.
##  2 1998 Q2 Adelaide South Australia Business  110.
##  3 1998 Q3 Adelaide South Australia Business  166.
##  4 1998 Q4 Adelaide South Australia Business  127.
##  5 1999 Q1 Adelaide South Australia Business  137.
##  6 1999 Q2 Adelaide South Australia Business  200.
##  7 1999 Q3 Adelaide South Australia Business  169.
##  8 1999 Q4 Adelaide South Australia Business  134.
##  9 2000 Q1 Adelaide South Australia Business  154.
## 10 2000 Q2 Adelaide South Australia Business  169.
## # i 24,310 more rows
```

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

holidays
```

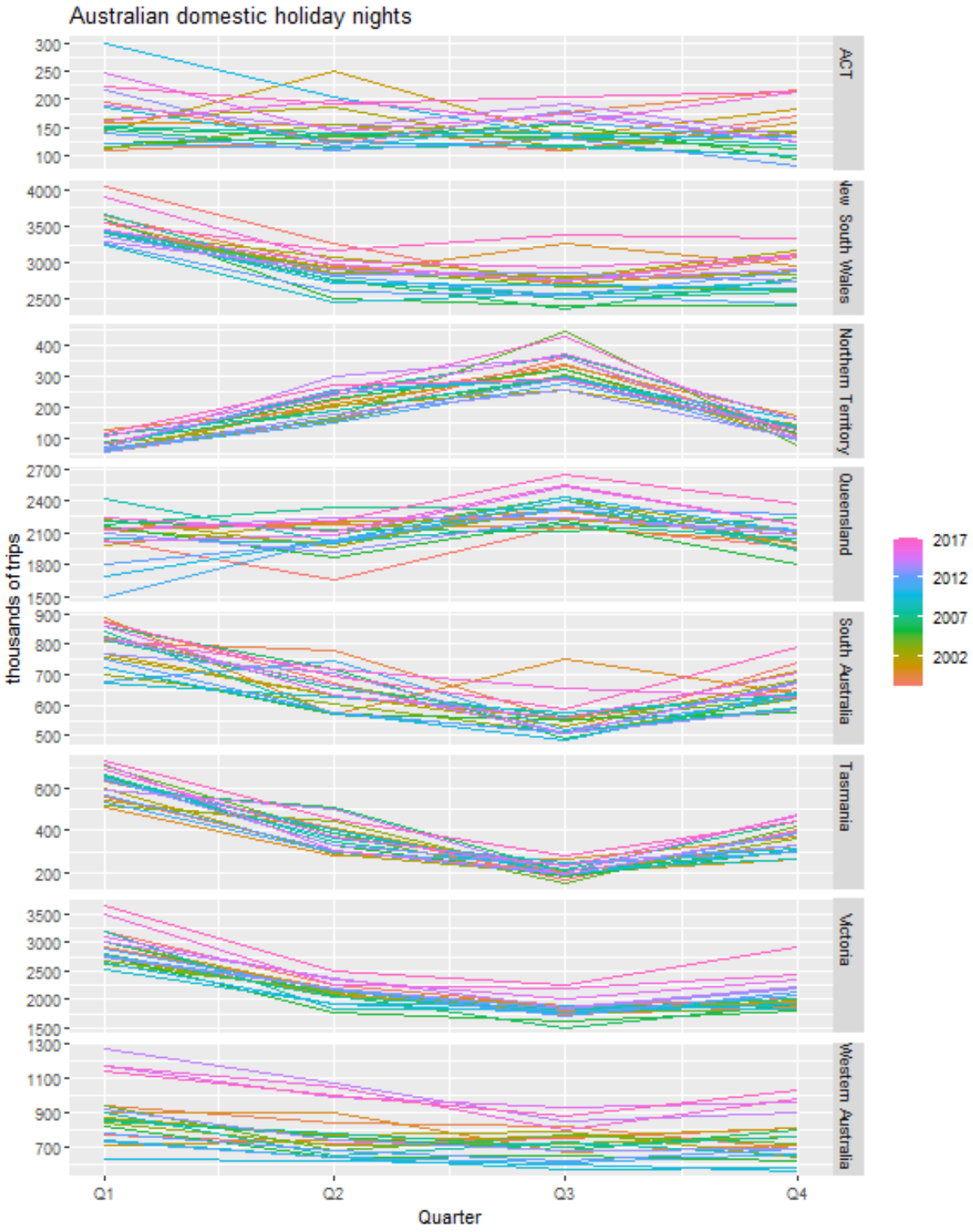
```
## # A tsibble: 640 x 3 [1Q]
```

```
## # 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
```

```
holidays |> autoplot(Trips) +
ylab("thousands of trips") +
  xlab("Year") +
  ggtitle("Australian domestic holiday nights")
```

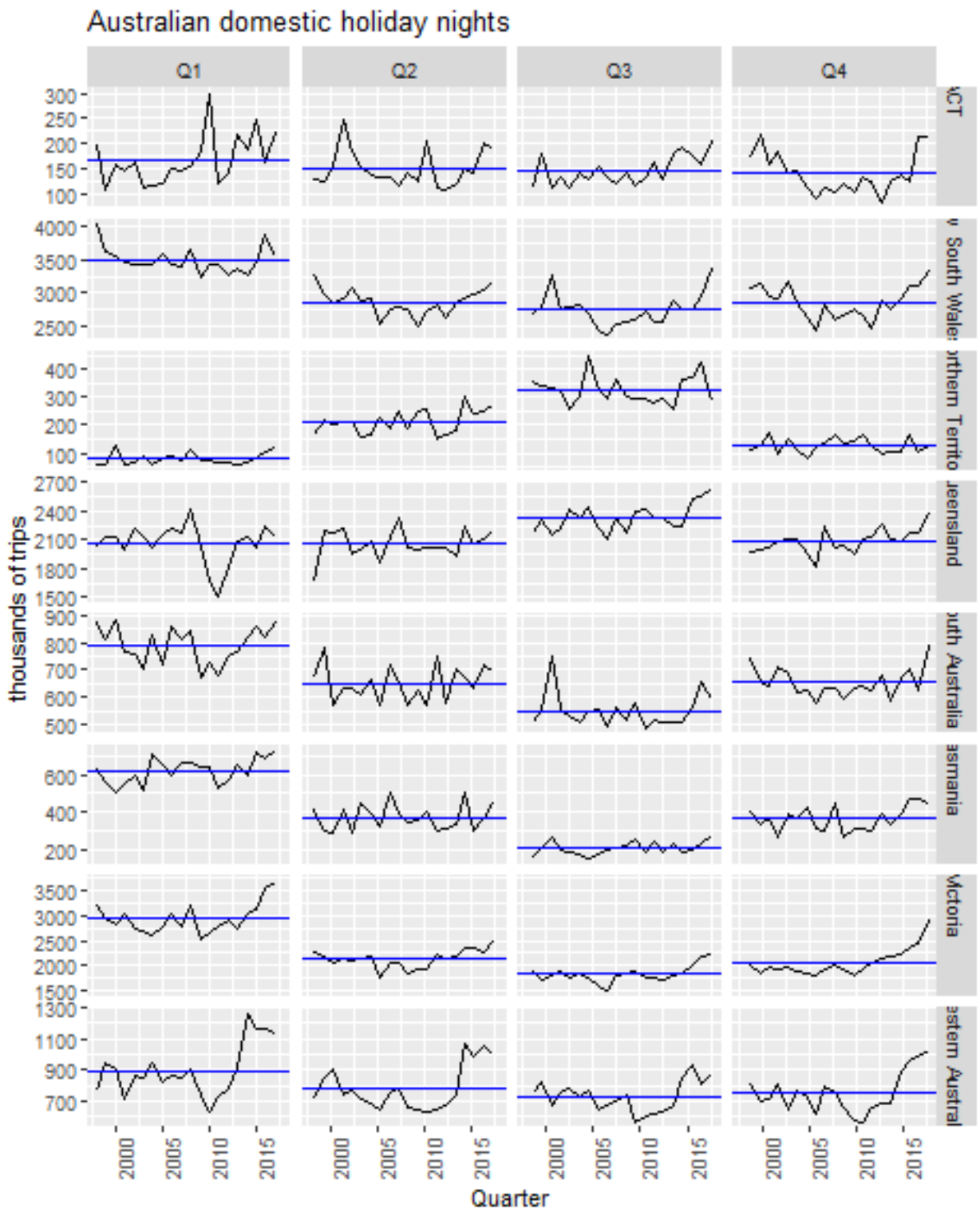



```
holidays |> gg_season(Trips) +
ylab("thousands of trips") + ggtitle("Australian domestic holiday nights")
```



Seasonal subseries plots

```
holidays |>
gg_subseries(Trips) + ylab("thousands of trips") + ggtitle("Australian domestic holiday nights")
```

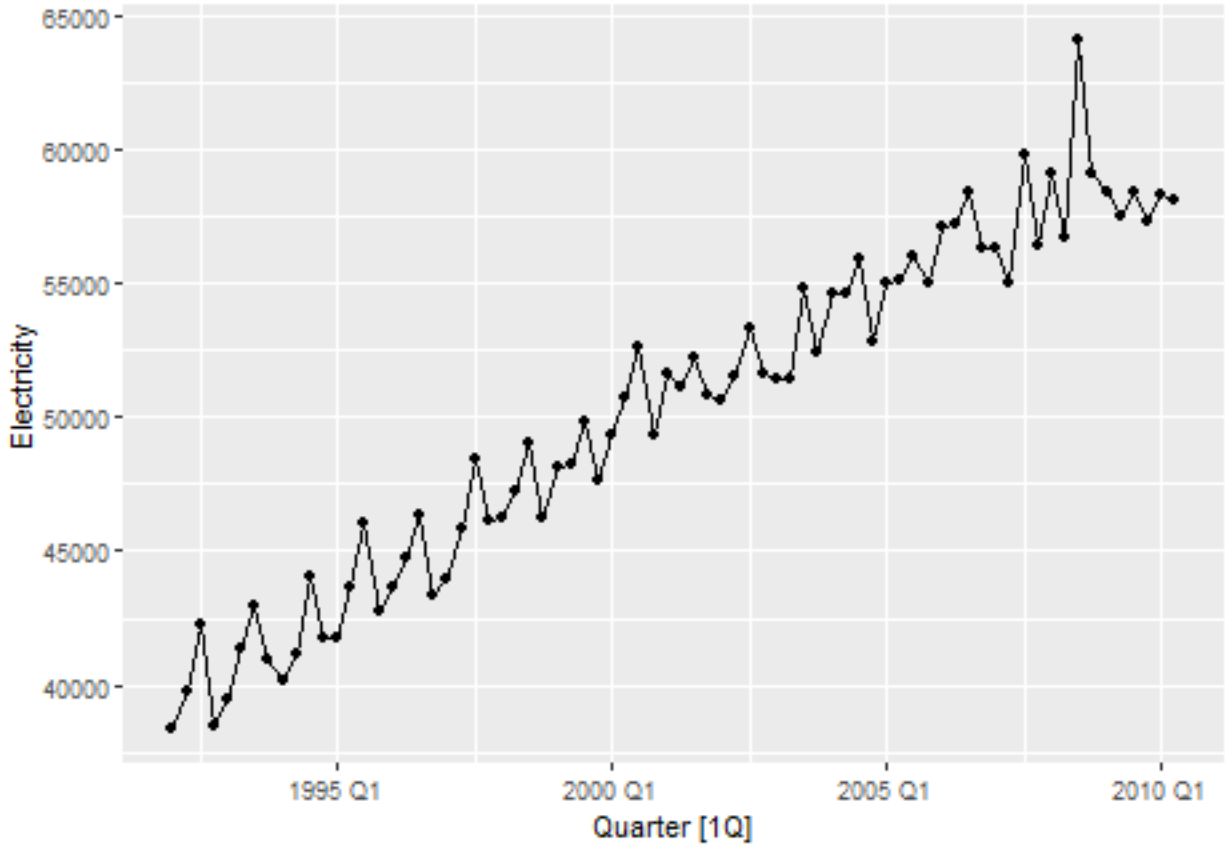


Lag plots and autocorrelation

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

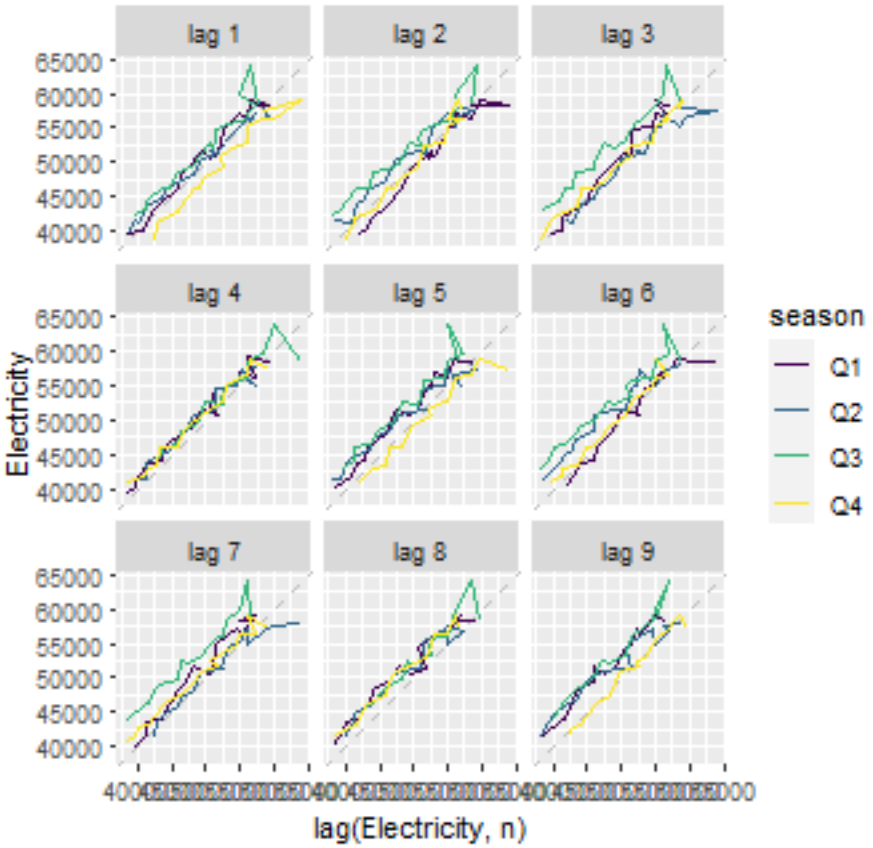
```
## # A tsibble: 74 x 7 [1Q]
##    Quarter  Beer Tobacco Bricks Cement Electricity   Gas
##      <qtr> <dbl>   <dbl>  <dbl>  <dbl>       <dbl> <dbl>
##  1 1992 Q1   443    5777    383   1289       38332   117
##  2 1992 Q2   410    5853    404   1501       39774   151
##  3 1992 Q3   420    6416    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 |> autoplot(Electricity) +
  geom_point()
```

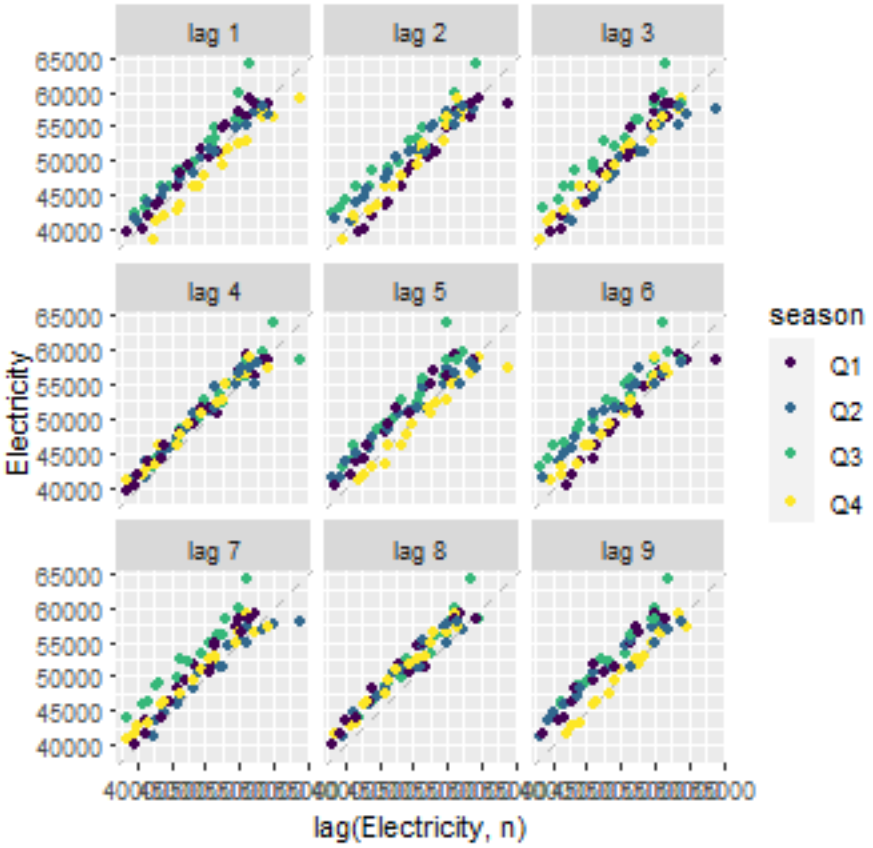


Lagged scatterplots

```
new_production |> gg_lag(Electricity)
```




```
new_production |> gg_lag(Electricity, geom='point')
```



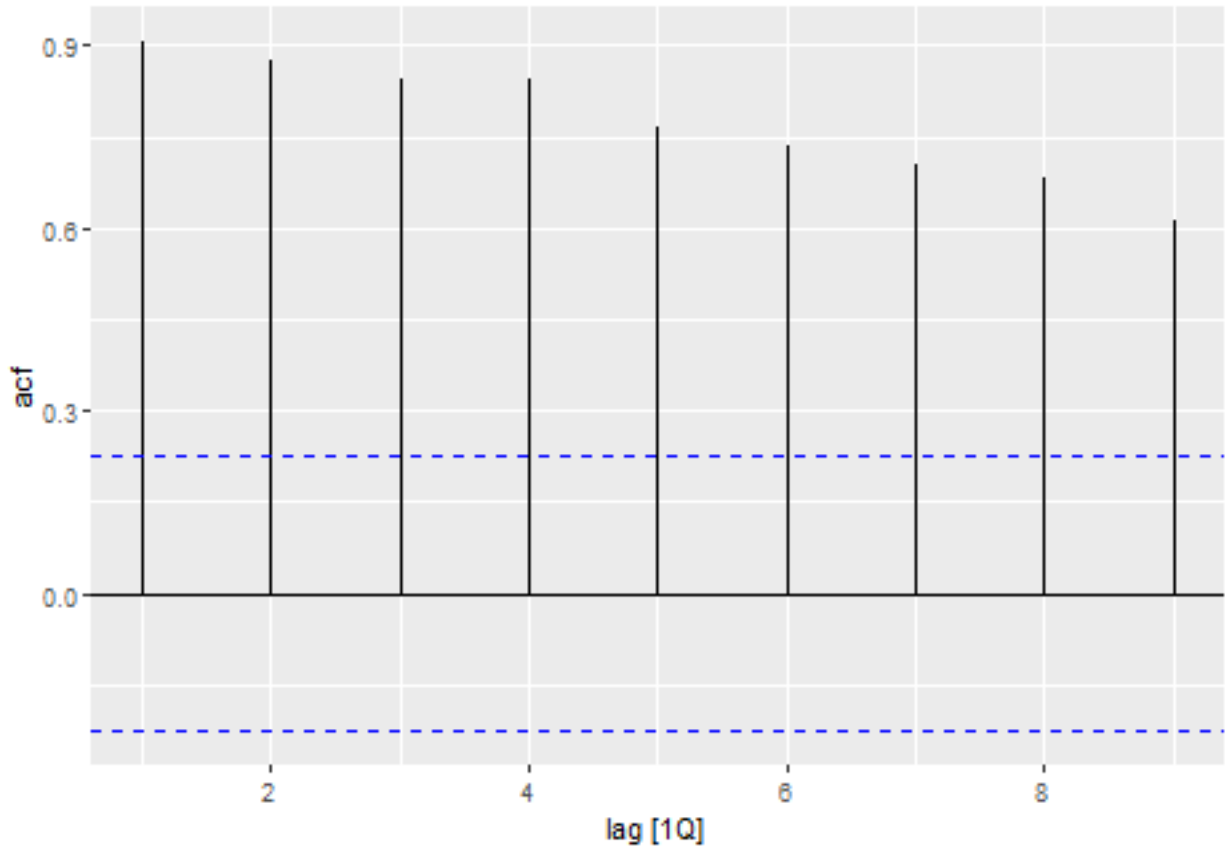
- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.
- ACF (autocorrelation function):
  - $r_1 = \text{Correlation}(y_t, y_{t-1})$
  - $r_2 = \text{Correlation}(y_t, y_{t-2})$
  - $r_3 = \text{Correlation}(y_t,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

Autocorrelation

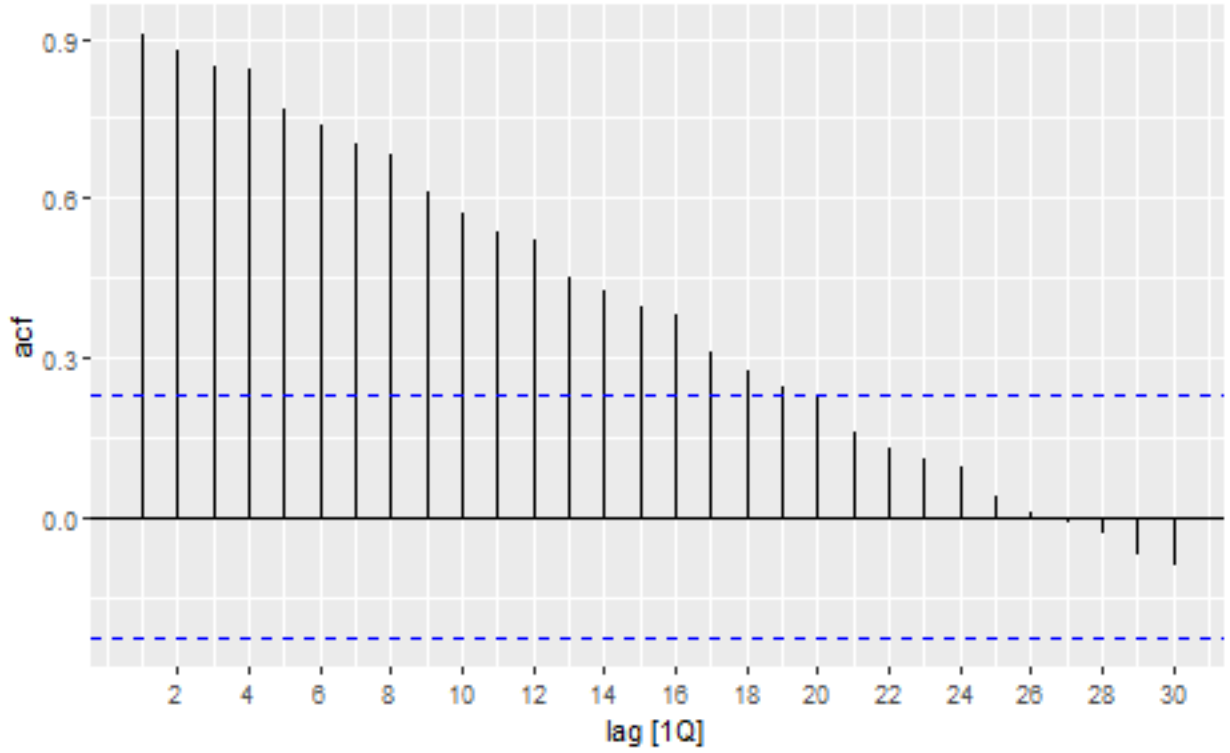
```
new_production |>
  ACF(Electricity, lag_max = 9)
```

```
## # A tsibble: 9 x 2 [1Q]
##        lag   acf
##   <cf_lag> <dbl>
## 1       1Q 0.907
## 2       2Q 0.878
## 3       3Q 0.846
## 4       4Q 0.844
## 5       5Q 0.766
## 6       6Q 0.737
## 7       7Q 0.703
## 8       8Q 0.683
## 9       9Q 0.613
```

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




```
new_production |>
 ACF(Electricity, lag_max = 30) |>
 autoplot()
```

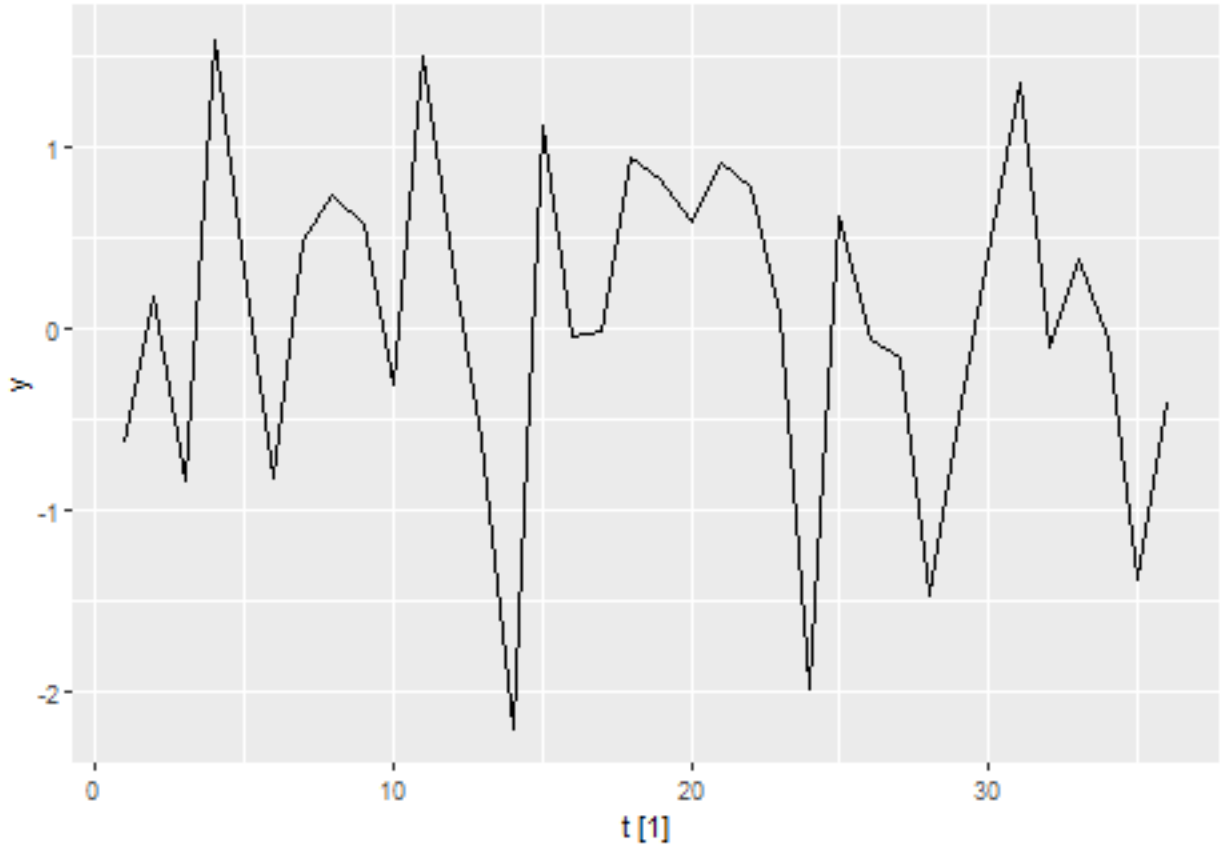


White noise

```
set.seed(1)
wn <- tsibble(t = seq(36), y = rnorm(36), index = t)

wn
```

```
## # A tsibble: 36 x 2 [1]
##        t      y
##    <int>  <dbl>
##  1     1 -0.626
##  2     2  0.184
##  3     3 -0.836
##  4     4  1.60
##  5     5  0.330
##  6     6 -0.820
##  7     7  0.487
##  8     8  0.738
##  9     9  0.576
## 10    10 -0.305
## # i 26 more rows
```

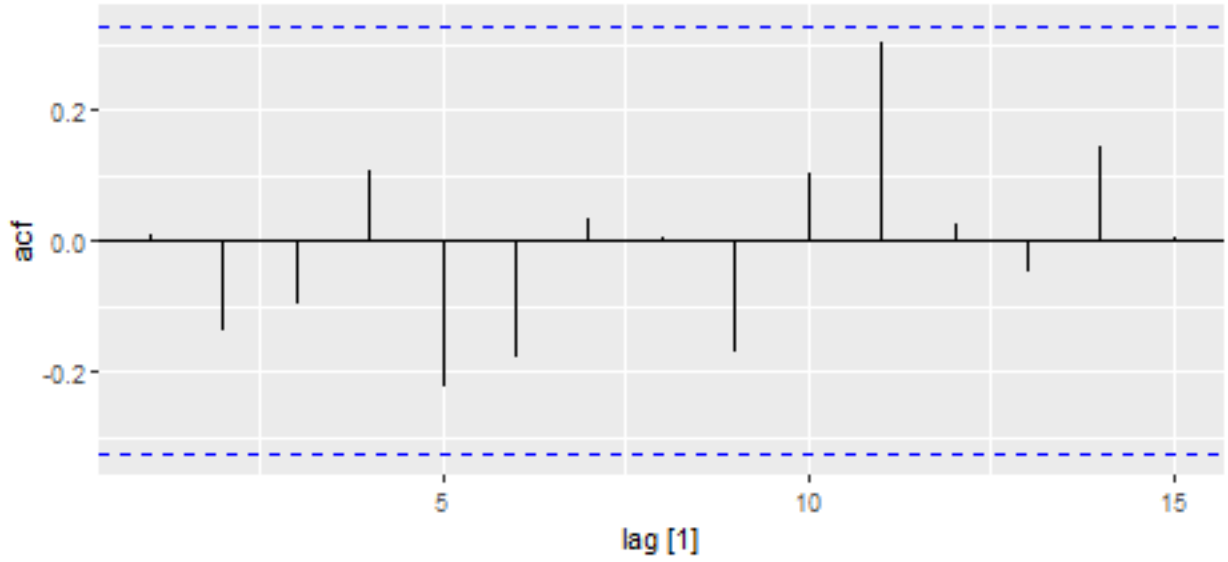
```
wn |> autoplot(y)
```




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

```
## # A tsibble: 15 x 2 [1]
##         lag      acf
##    <cf_lag>    <dbl>
##  1        1  0.00964
##  2        2 -0.137
##  3        3 -0.0975
##  4        4  0.107
##  5        5 -0.222
##  6        6 -0.177
##  7        7  0.0342
##  8        8  0.00646
##  9        9 -0.171
## 10       10  0.103
## 11       11  0.301
## 12       12  0.0246
## 13       13 -0.0469
## 14       14  0.144
## 15       15  0.00649
```

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



Portmanteau tests for autocorrelation

$H_0 : \rho_1 = \rho_2 = \cdots = \rho_9 = 0$

H_1 : at least one $\rho_k \neq 0$, for $1 \leq k \leq 9$

```
# Method 1 - using stat pakcage
Box.test(wn$y, lag=10, fitdf=0, type="Lj")
```

```
##
##  Box-Ljung test
##
## data:  wn$y
## X-squared = 7.3501, df = 10, p-value = 0.692
```

```
# Method 2 - using fabletools package
wn |> features(y, ljung_box, lag=10, dof = 0)
```

```
## # A tibble: 1 x 2
##   lb_stat lb_pvalue
##     <dbl>     <dbl>
## 1    7.35     0.692
```

References

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.