

BI Workplace T02

Time series graphics

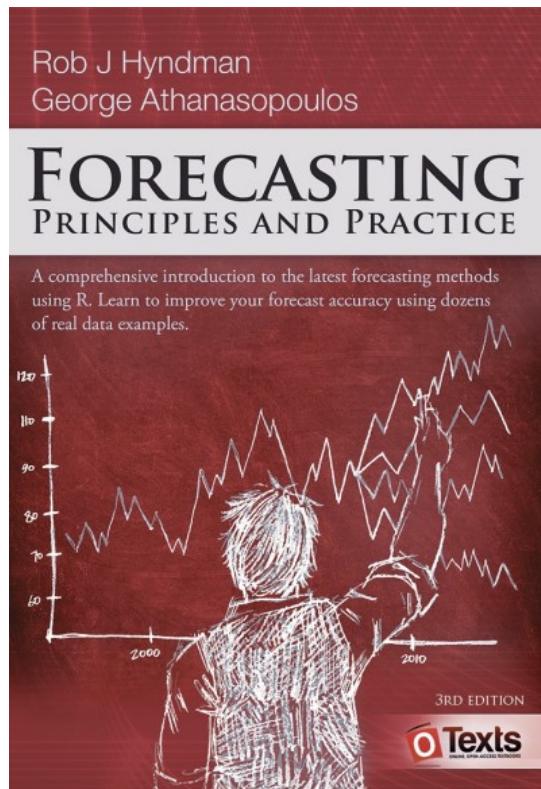




- Eaton, J.W., Bateman, D., Hauberg, S., Wehbring, R. (2021) [GNU Octave Manual: Free Your Numbers](#)
- Grolemund, G., Wickham. H. (2017) [R for Data Science](#)
- Gundersen, V.B. (2008) [Mathesaurus](#), including the [MATLAB/Octave-Python-R cheatsheet](#)
- Hiebeler, D. (2014) [MATLAB / R Reference](#)
- Hyndman, R.J., Athanasopoulos, G. (2021) [Forecasting: principles and practice \(3rd ed.\)](#)
- VanderPlas, J. (2016) [Python Data Science Handbook](#)



Sec. 2: Time series graphics



<https://otexts.com/fpp3>

2 Time series graphics

2.1 tsibble objects

2.2 Time plots

2.3 Time series patterns

2.4 Seasonal plots

2.5 Seasonal subseries plots

2.6 Scatterplots

2.7 Lag plots

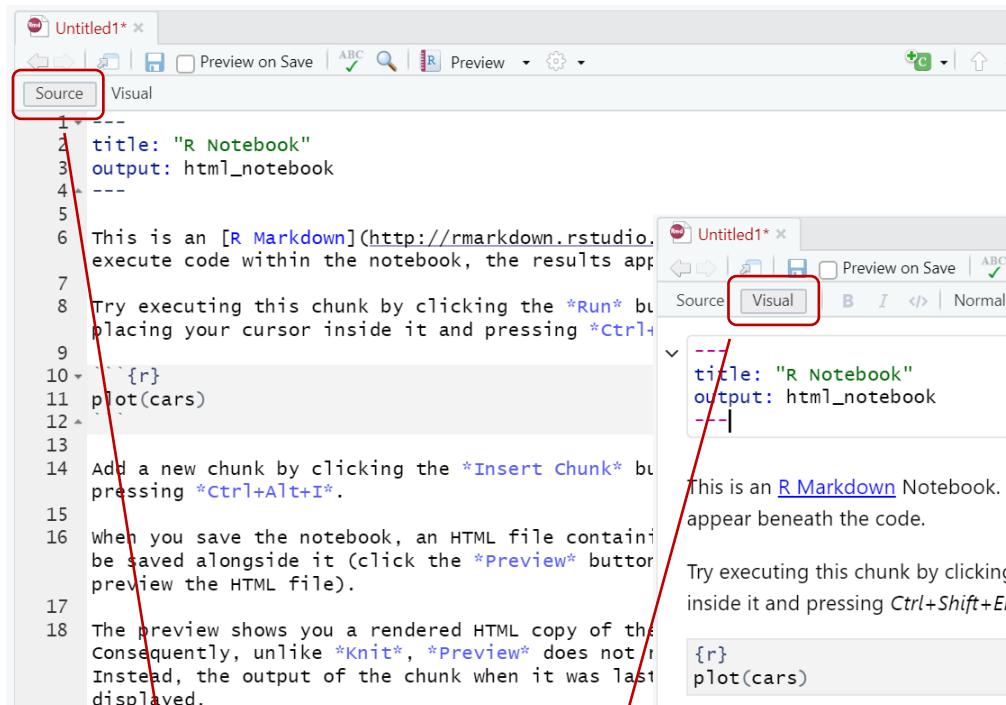
2.8 Autocorrelation

2.9 White noise

2.10 Exercises

2.11 Further reading

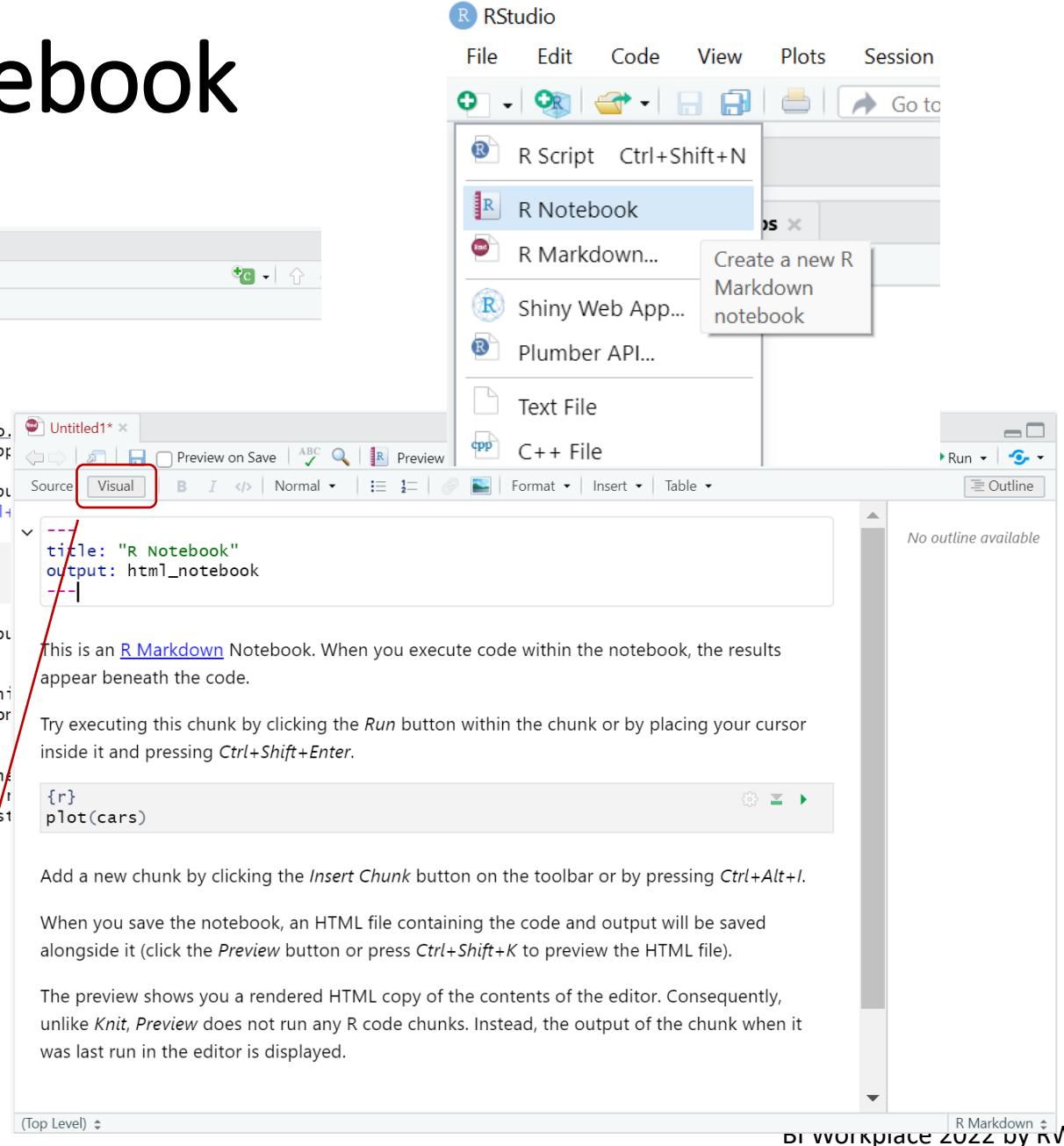
Create a Notebook



The screenshot shows the RStudio interface with the 'Source' tab selected in the top-left corner. The main pane displays R code for creating an R Markdown notebook. A red box highlights the 'Source' tab. A red arrow points from the 'Display mode' callout to this tab.

```
1 ---  
2 title: "R Notebook"  
3 output: html_notebook  
4 ---  
5  
6 This is an [R Markdown](http://rmarkdown.rstudio.com) notebook.  
execute code within the notebook, the results appear  
7  
8 Try executing this chunk by clicking the *Run* button  
or placing your cursor inside it and pressing *Ctrl+Shift+Enter*.  
9  
10 {r}  
11 plot(cars)  
12  
13 Add a new chunk by clicking the *Insert chunk* button  
pressing *Ctrl+Alt+I*.  
14  
15 When you save the notebook, an HTML file containing  
the results will be saved alongside it (click the *Preview* button  
to preview the HTML file).  
16  
17 The preview shows you a rendered HTML copy of the  
notebook. Consequently, unlike *Knit*, *Preview* does not  
run any R code chunks. Instead, the output of the chunk when it was last  
displayed.
```

Display mode



The screenshot shows the RStudio interface with the 'Visual' tab selected in the top-left corner. The main pane displays the rendered HTML content of the R Markdown notebook. A red box highlights the 'Visual' tab. A red arrow points from the 'Display mode' callout to this tab.

R Studio

File Edit Code View Plots Session

R Script Ctrl+Shift+N

R Notebook

R Markdown... Create a new R Markdown notebook

R Shiny Web App...

R Plumber API...

Text File

C++ File

Run Outline

No outline available

This is an [R Markdown](#) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
{r}  
plot(cars)
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

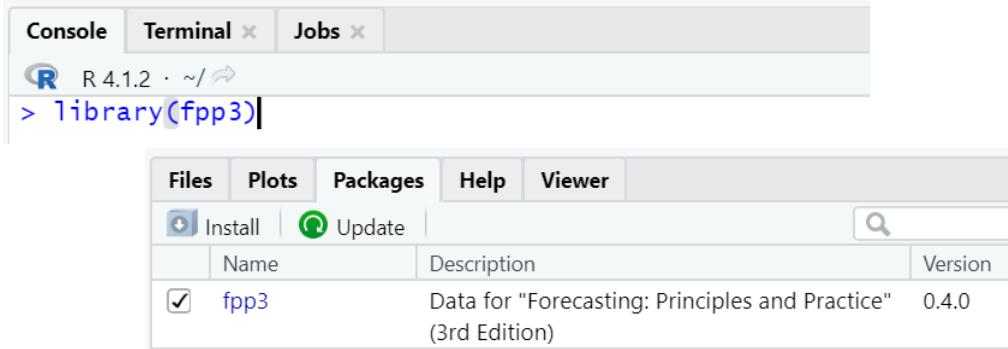
When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

(Top Level) R Markdown

The fpp3 library

- Load the **fpp3** library
 - From the **Console** or the **Packages** list
- List the library datasets
`> data(package = 'fpp3')` and all library datasets
`> data()`



Console Terminal × Jobs ×

R 4.1.2 · ~/

> library(fpp3)

Files Plots Packages Help Viewer

Install Update

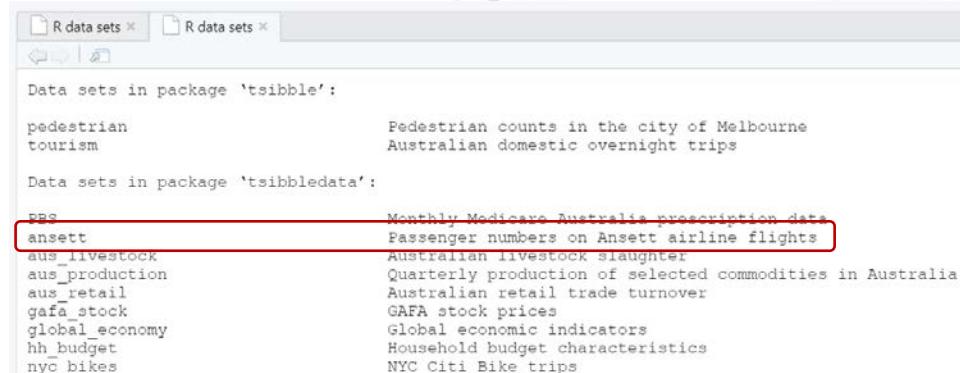
Name	Description	Version
fpp3	Data for "Forecasting: Principles and Practice" (3rd Edition)	0.4.0



R data sets ×

Data sets in package 'fpp3':

aus_accommodation	Australian accommodation data
aus_airpassengers	Air Transport Passengers Australia
aus_arrivals	International Arrivals to Australia
bank_calls	Call volume for a large North American bank
boston_marathon	Boston marathon winning times since 1897
canadian_gas	Monthly Canadian gas production
guinea_rice	Rice production (Guinea)
insurance	Insurance quotations and advertising expenditure
prices	Price series for various commodities
souvenirs	Sales for a souvenir shop
us_change	Percentage changes in economic variables in the USA.
us_employment	US monthly employment data



R data sets × R data sets ×

Data sets in package 'tsibble':

pedestrian	Pedestrian counts in the city of Melbourne
tourism	Australian domestic overnight trips

Data sets in package 'tsibbledata':

pbs	Monthly Medicare Australia prescription data
ansett	Passenger numbers on Ansett airline flights
aus_livestock	Australian Livestock slaughter
aus_production	Quarterly production of selected commodities in Australia.
aus_retail	Australian retail trade turnover
gafa_stock	GAFA stock prices
global_economy	Global economic indicators
hh_budget	Household budget characteristics
nyc_bikes	NYC Citi Bike trips



Now you can copy-paste code

The screenshot shows two RStudio panes side-by-side. The left pane is in 'Source' mode, displaying R code for initializing the FPP3 library and creating time plots. The right pane is in 'Visual' mode, showing the resulting HTML output with sections like 'Initialize the FPP3 library' and '2.2 Time plots'. A red box highlights the 'Source' tab in the left pane, and another red box highlights the 'Visual' tab in the right pane. A red arrow points from the 'Source' tab to the 'Visual' tab, illustrating the process of switching modes. A red box labeled 'Display mode' is overlaid on the bottom left of the image.

fp3_2.Rmd

Source Visual

```
1 ---  
2 title: 'FPP3: Chapter 2 Time series graphics'  
3 output:  
4   html_document:  
5     df_print: paged  
6 ---  
7 ## Initialize the FPP3 library  
8  
9 {r}  
10 library(fpp3)  
11 ...  
12  
13 ## 2.2 Time plots  
14 {r}  
15 ...  
16 {r}  
17  
# 2.2 Time plots  
18  
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)")  
26
```

fp3_2.Rmd

Source Visual

```
---  
title: 'FPP3: Chapter 2 Time series graphics'  
output:  
  html_document:  
    df_print: paged
```

Initialize the FPP3 library

```
{r}  
library(fpp3)
```

2.2 Time plots

```
{r}  
  
# 2.2 Time plots  
  
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)")
```

... but let's first plot a sample dataset

Sample dataset - Ansett Airlines

```
{r}
ansett
autoplot(ansett)
```

tbl ts

A tsibble: 7,407 x 4 [1W]

Key: Airports, Class [30]

Week	Airports	Class	Passengers
1989 W28	ADL-PER	Business	193
1989 W29	ADL-PER	Business	254
1989 W30	ADL-PER	Business	185
1989 W31	ADL-PER	Business	254
1989 W32	ADL-PER	Business	191
1989 W33	ADL-PER	Business	136
1989 W34	ADL-PER	Business	0
1989 W35	ADL-PER	Business	0
1989 W36	ADL-PER	Business	0
1989 W37	ADL-PER	Business	0

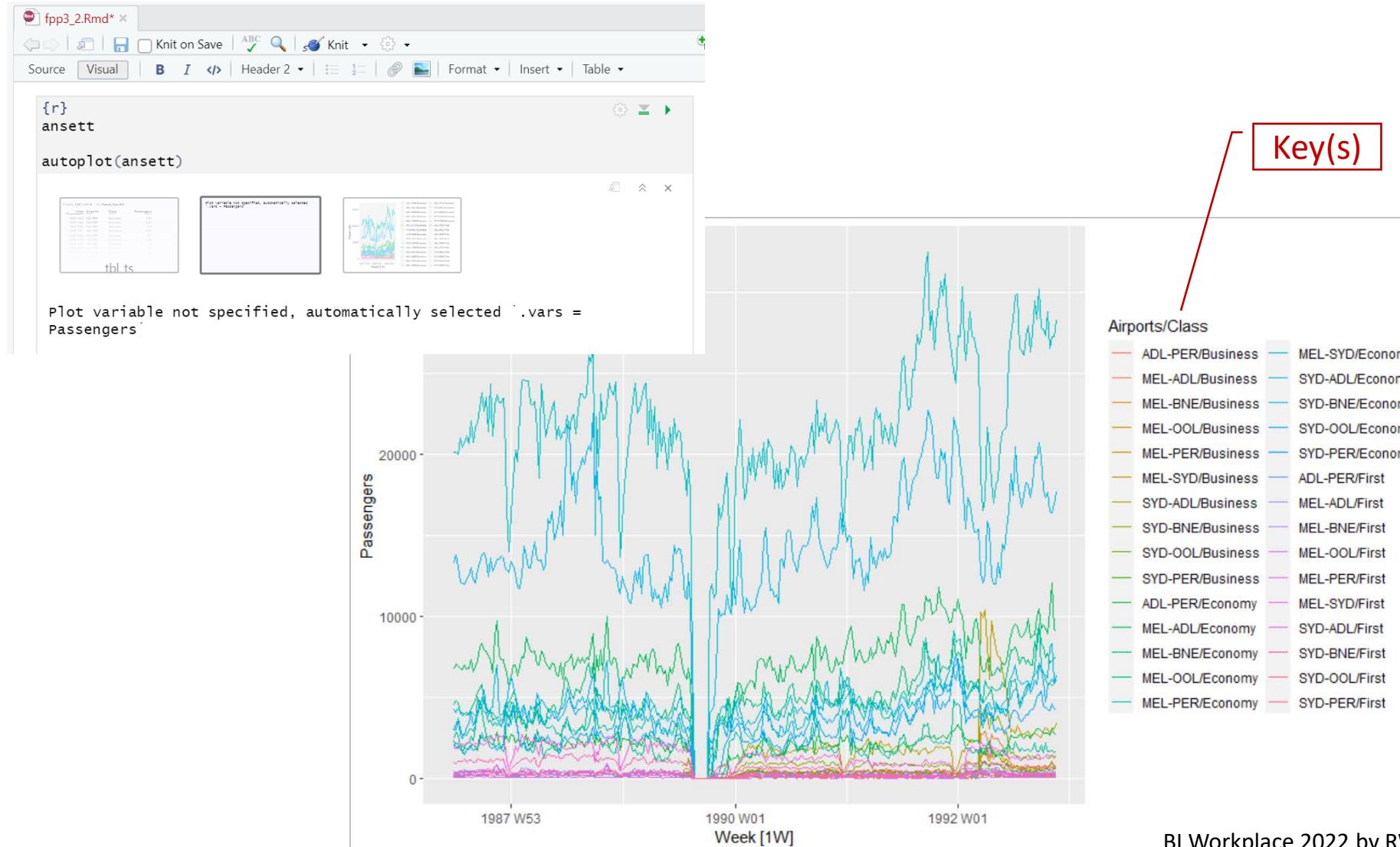
1-10 of 7,407 rows

Previous 1 2 3 4 5 6 ... 100 Next

Initialize the FPP3 library
Sample dataset - Ansett Airli...
2.2 Time plots
2.4 Seasonal plots
2.5 Seasonal subseries plots



A warning and the result



Copy-paste code from Sec. 2.2

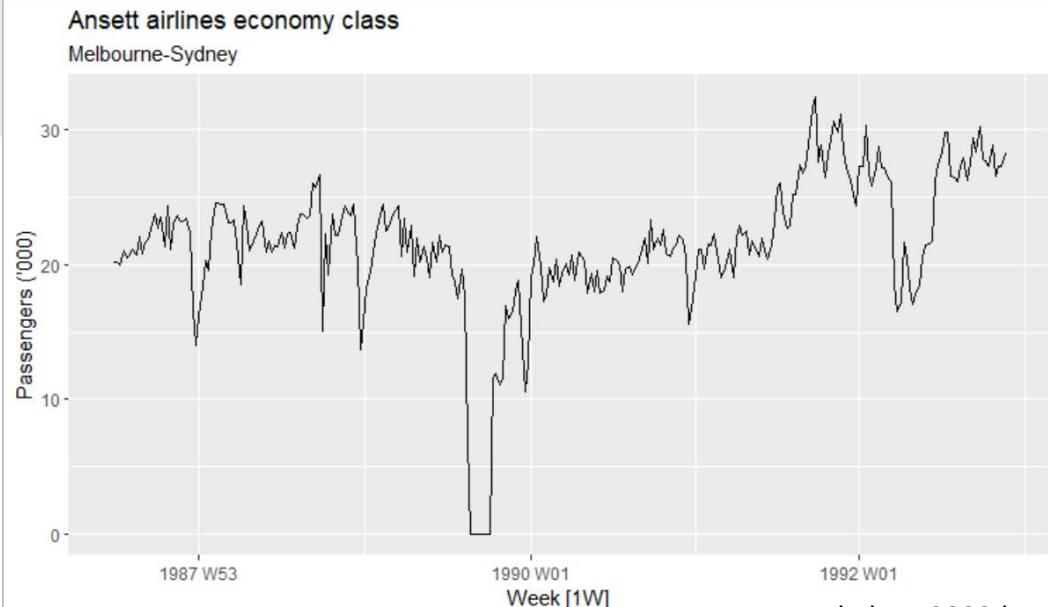
2.2 Time plots

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.

Figure 2.1 shows the weekly economy passenger load on Ansett airlines between Australia's two largest cities.

```
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)")
```

Label properties





The pipe operator: %>%

- The left-hand side of each pipe is passed as the first argument to the function on the right-hand side

```
> melsyd_economy <- ansett %>%  
  filter(Airports == "MEL-SYD", Class == "Economy") %>%  
  mutate(Passengers = Passengers/1000)
```

- is equivalent to

Select keys

```
> var1 <- filter(ansett, Airports == "MEL-SYD", Class == "Economy")  
> melsyd_economy <- mutate(var1, Passengers = Passengers/1000)
```

Transform
variable



tsibble objects

- A time series can be thought of as a list of:
 - Time stamps
→ the **index**
 - Measurement(s)
→ the **variable(s)**
- Optionally with:
 - the **key(s)** to store multiple time series in a single object

```
Console Terminal × Jobs ×
R 4.1.2 · ~/🔗
> y <- tsibble(
+   Year = 2015:2019,
+   observation = c(123, 39, 78, 52, 110),
+   index = Year
+ )
> y
# A tsibble: 5 x 2 [1Y]
#   Year   Observation
#   <dbl>      <dbl>
1 2015        123
2 2016         39
3 2017         78
4 2018         52
5 2019        110
> |
```



```
Console Terminal × Jobs ×
R 4.1.2 · ~/🔗
> ansett
# A tsibble: 7,407 x 4 [1w]
# Key:   Airports, Class [30]
#   Week Airports Class Passengers
#   <dbl> <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
# ... with 7,397 more rows
> |
```



Working with **tsibble** objects

- We can use functions to work with **tsibble** objects:
 - **select()** → select particular columns
 - **filter()** → keep particular rows
 - **mutate()** → create new variables
 - **summarise()** → combine data across keys

```
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
```

```
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
```

CSV to tsibble

- Identify

- which column is the (time) **index**
- which column(s) are the **key(s)**

- The remaining column(s) are the **variable(s)**

```
{r}
prison <- readr::read_csv("https://OTexts.com/fpp3/extrafiles/prison_population.csv")

prison <- prison %>%
  mutate(Quarter = yearquarter(Date)) %>%
  select(-Date) %>%
  as_tsibble(key = c(State, Gender, Legal, Indigenous), index = Quarter)

prison
```

Rows: 3072 Columns: 6
--- Column specification ---

Delimiter: ","
chr (4): State, Gender, Legal, Indigenous
dbl (1): Count
date (1): Date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

A tsibble: 3,072 x 6 [1Q] Key: State, Gender, Legal, Indigenous [64]

State	Gender	Legal	Indigenous	Count	Quarter
<chr>	<chr>	<chr>	<chr>	<dbl>	<S3: yearquarter>
ACT	Female	Remanded	ATSI	0	2005 Q1
ACT	Female	Remanded	ATSI	1	2005 Q2
ACT	Female	Remanded	ATSI	0	2005 Q3
ACT	Female	Remanded	ATSI	0	2005 Q4
ACT	Female	Remanded	ATSI	1	2006 Q1
ACT	Female	Remanded	ATSI	1	2006 Q2
ACT	Female	Remanded	ATSI	1	2006 Q3
ACT	Female	Remanded	ATSI	0	2006 Q4
ACT	Female	Remanded	ATSI	0	2007 Q1
ACT	Female	Remanded	ATSI	1	2007 Q2

1-10 of 3,072 rows Previous 1 2 3 4 5 6 ... 100 Next



tsibble (DataFrame) to CSV

```
> write.csv(prison, "prison.csv", row.names = FALSE)
```

```
1 "State", "Gender", "Legal", "Indigenous", "Count", "Quarter"
2 "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q1
3 "ACT", "Female", "Remanded", "ATSI", 1, 2005 Q2
4 "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q3
5 "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q4
6 "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q1
7 "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q2
8 "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q3
9 "ACT", "Female", "Remanded", "ATSI", 0, 2006 Q4
10 "ACT", "Female", "Remanded", "ATSI", 0, 2007 Q1
```

```
> write.csv(prison, "prison.csv", row.names = TRUE)
```

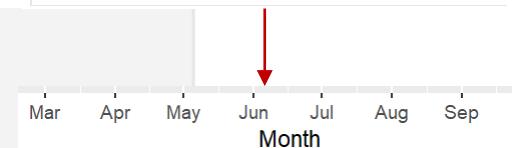
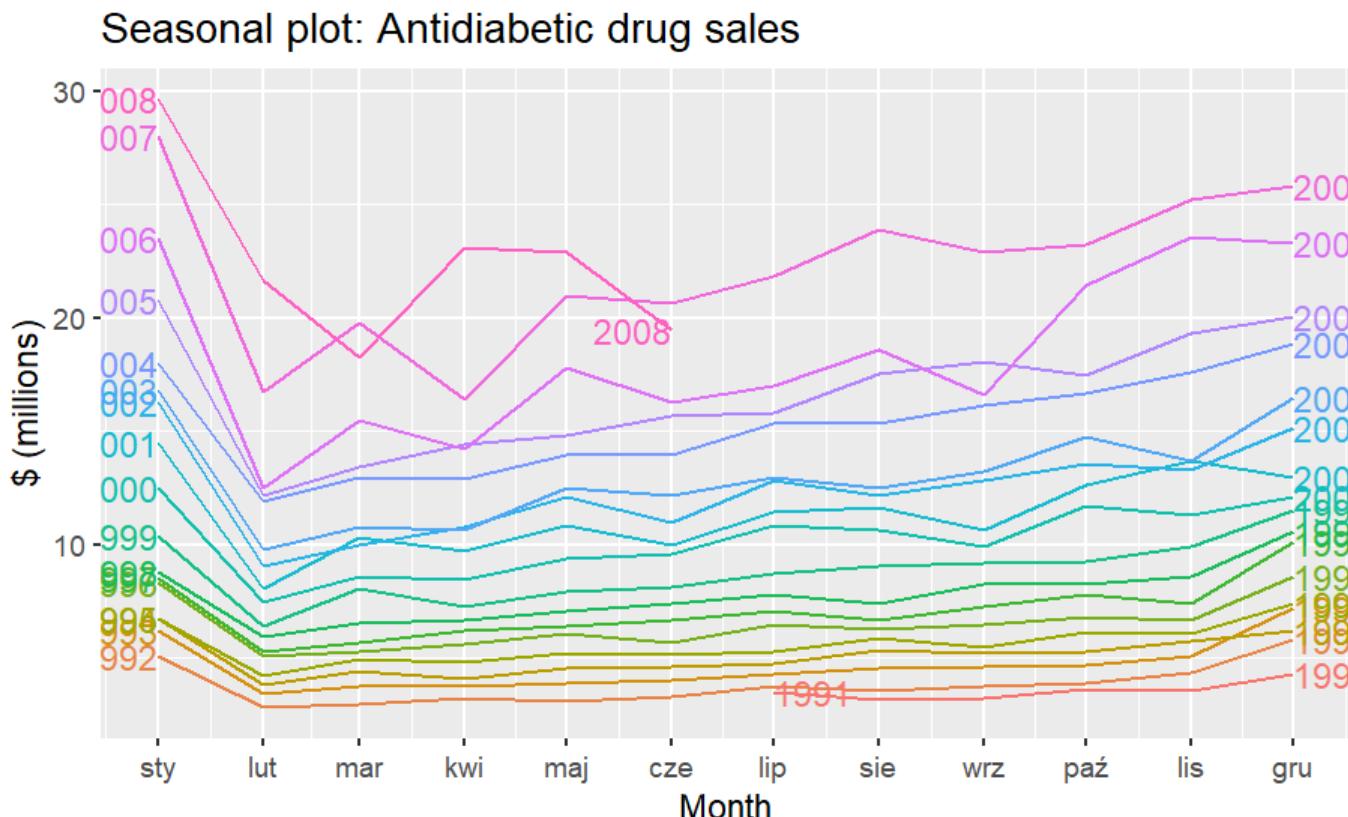
```
1 "", "State", "Gender", "Legal", "Indigenous", "Count", "Quarter"
2 "1", "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q1
3 "2", "ACT", "Female", "Remanded", "ATSI", 1, 2005 Q2
4 "3", "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q3
5 "4", "ACT", "Female", "Remanded", "ATSI", 0, 2005 Q4
6 "5", "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q1
7 "6", "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q2
8 "7", "ACT", "Female", "Remanded", "ATSI", 1, 2006 Q3
9 "8", "ACT", "Female", "Remanded", "ATSI", 0, 2006 Q4
10 "9", "ACT", "Female", "Remanded", "ATSI", 0, 2007 Q1
```

Seasonal plots: gg_season

```
{r}  
sys.setlocale("LC_TIME", "English")  
  
[1] "English_United States.1252"
```

```
a10 %>%  
  gg_season(Cost, labels = "both") +  
  labs(y = "$ (millions)",  
       title = "Seasonal plot: Antidiabetic drug sales")
```

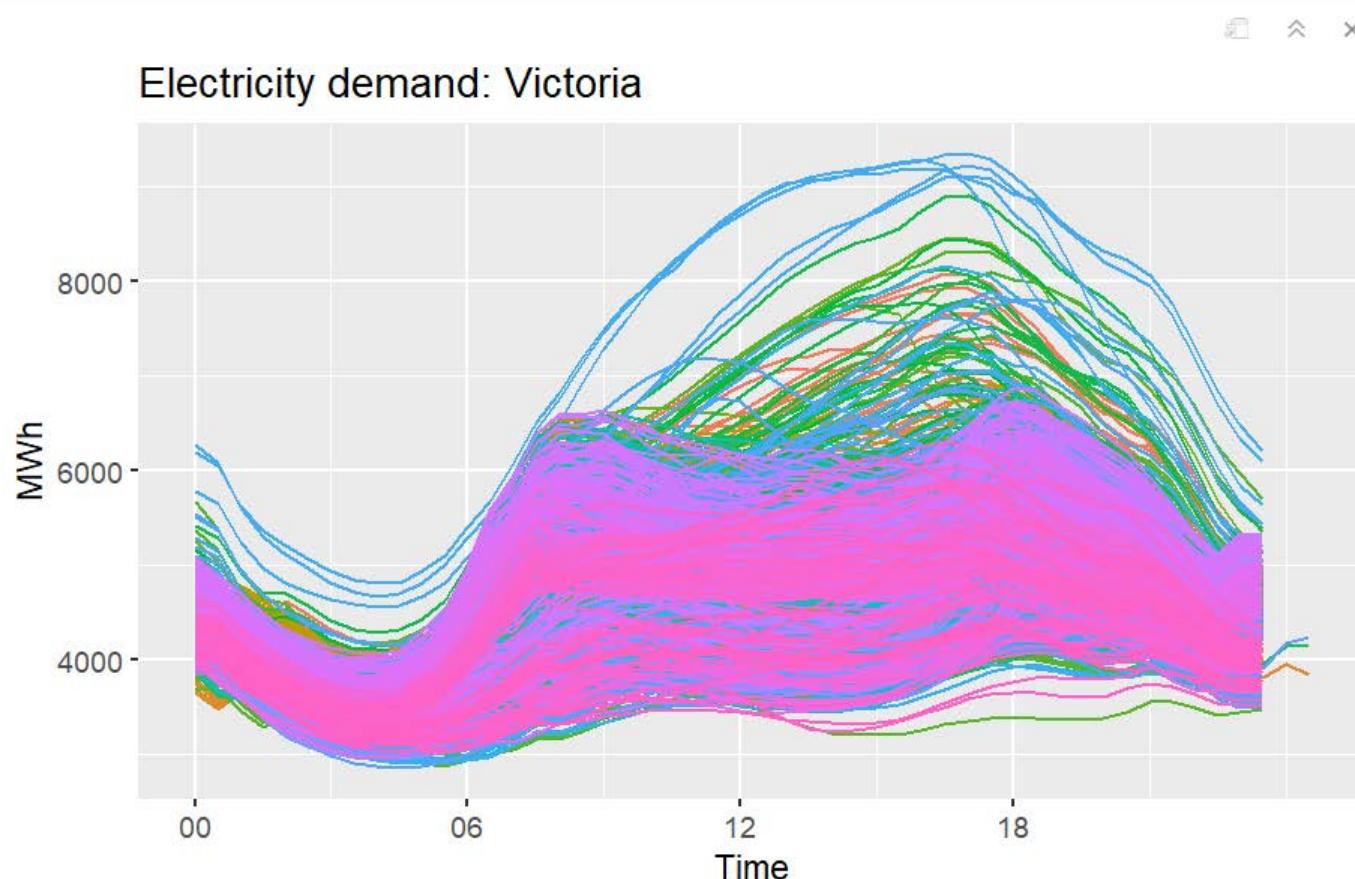
Label properties



gg_season cont.

```
vic_elec %>% gg_season(Demand, period = "day") +  
  theme(legend.position = "none") +  
  labs(y="MWh", title="Electricity demand: Victoria")
```

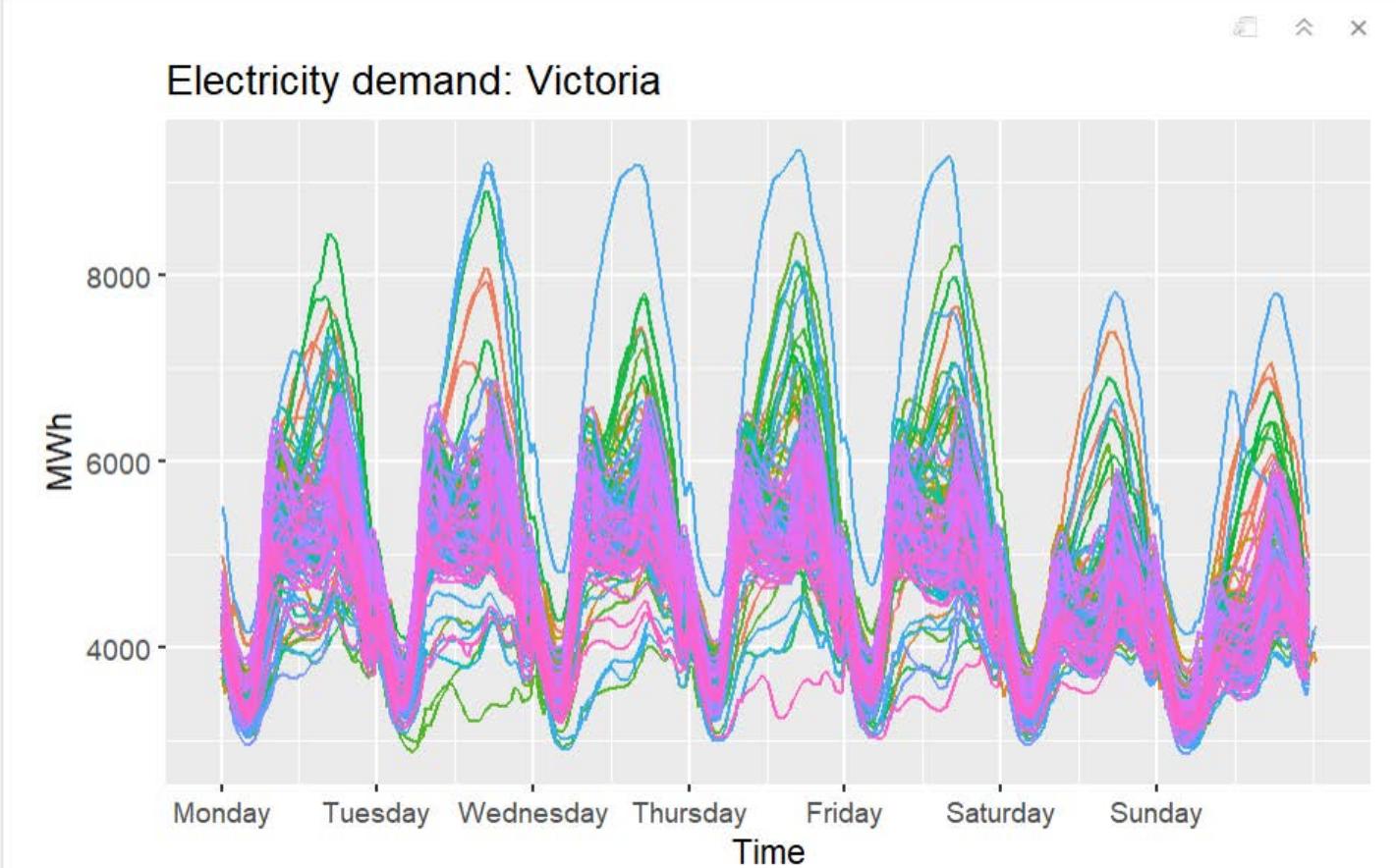
```
Console Terminal x Jobs x  
R 4.1.2 · ~/  
> vic_elec  
# A tsibble: 52,608 x 5 [30m] <Australia/Melbourne>  
  Time           Demand Temperature Date   Holiday  
  <dttm>        <dbl>     <dbl>    <date>  <lgl>  
1 2012-01-01 00:00:00 4383.    21.4 2012-01-01 TRUE  
2 2012-01-01 00:30:00 4263.    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
```



gg_season cont.

“week”, not “day”

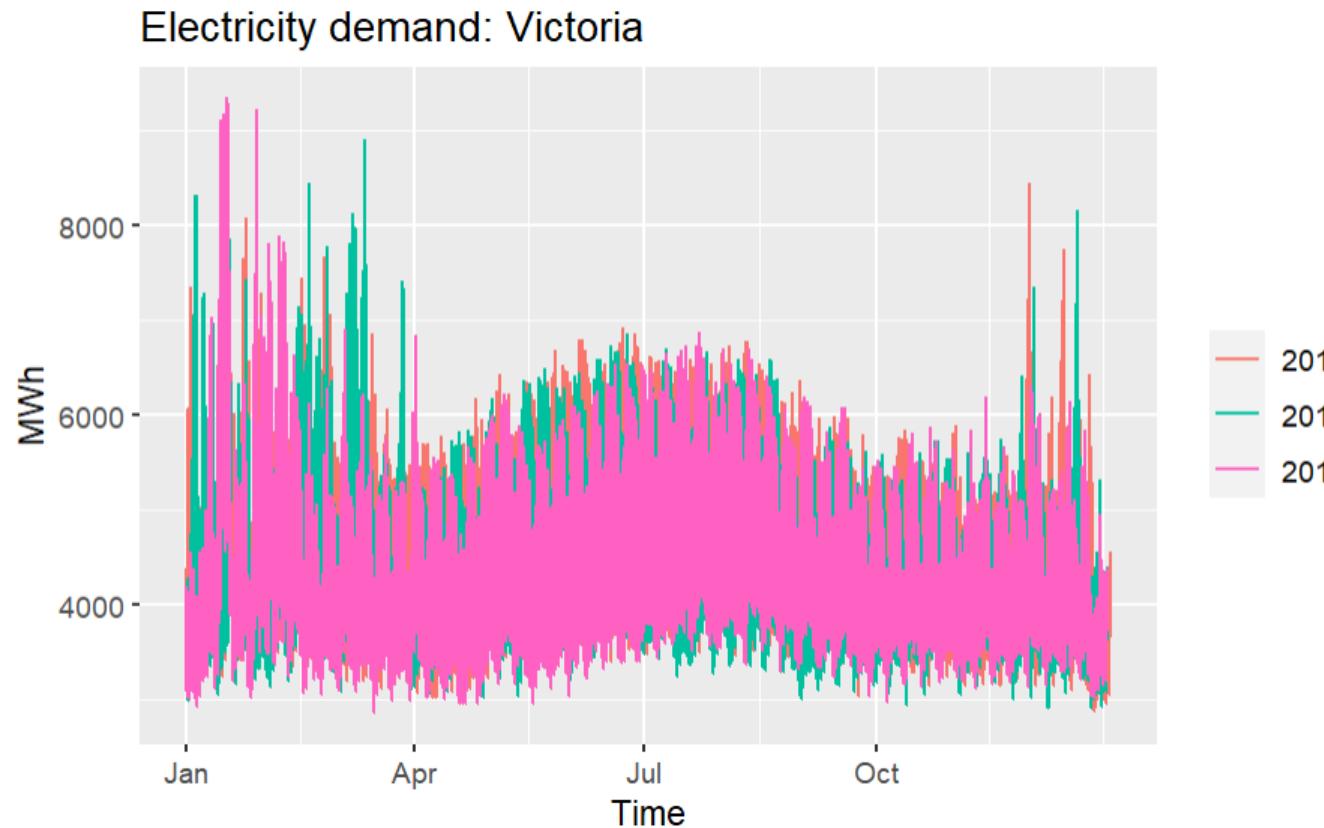
```
vic_elec %>% gg_season(Demand, period = "week") +  
  theme(legend.position = "none") +  
  labs(y="MWh", title="Electricity demand: Victoria")
```



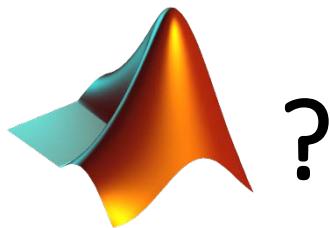
gg_season cont.

"year", not "week"

```
vic_elec %>% gg_season(Demand, period = "year") +  
  labs(y="MWh", title="Electricity demand: Victoria")
```



What about



?

“Regular” script

Live Script

The screenshot shows the MATLAB interface with the Home tab selected. A red arrow points from the "Live Script" text box to the "LIVE EDITOR" button in the top menu bar. The Live Editor window displays a script named "fpp3_2.mlx" containing MATLAB code for plotting passengers data. The code includes setting the current directory, loading data from a CSV file, and creating a stacked plot. The Home tab menu bar also includes PLOTS, APPS, and INSERT tabs.

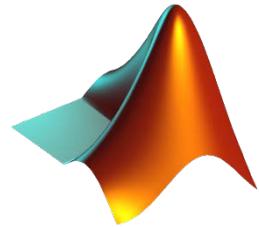
```
% Dock figures in the main window
set(0, 'DefaultFigureWindowState', 'docked')

%% 2.2 Time plots, Ansett airlines economy class

% Load data
melsyd_economy = readtable('melsyd_economy.csv');

figure(1)
stackedplot(melsyd_economy(:, 'Passengers'))
% NOTE: Works for tables, but provides no access to XTicks
```





(Simple) time plots

Live Editor - fpp3_2.mlx

fpp3_2.m fpp3_2.mlx + Help

```
%cd('D:\USERS\rweron\Wykłady\BIW2022\Matlab')
```

2.2 Time plots, Ansett airlines economy class

```
% Load data
melsyd_economy = readtable('melsyd_economy.csv');

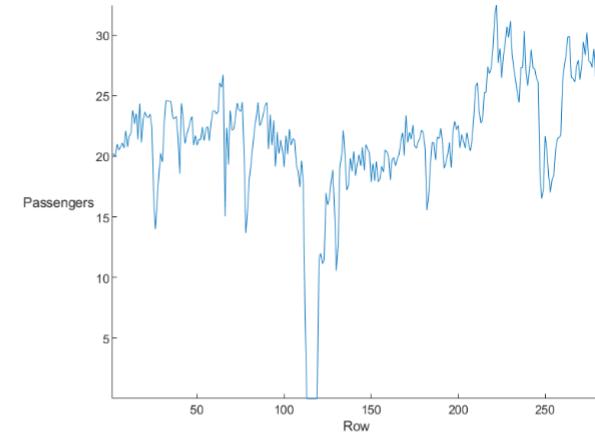
figure
stackedplot(melsyd_economy(:, 'Passengers'))
% NOTE: Works for tables, but provides no access to XTicks
```

```
figure
% Convert table column Passengers to array
melsyd_Passengers = table2array(melsyd_economy(:, 'Passengers'));
plot(melsyd_Passengers)
h = gca;
h.XLim = [1 length(melsyd_Passengers)];
h.XTickLabel = table2cell(melsyd_economy(h.XTick, 'Week'));
% Set axis labels and title
h.XLabel.String = 'Week [1W]';
h.YLabel.String = 'Passengers (~000)';
h.Title.String = 'Ansett airlines economy class: Melbourne-Sydney';
```

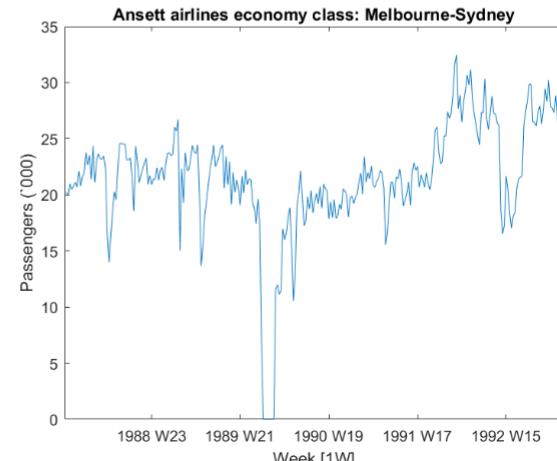
2.2 Time plots, Australian antidiabetic drug sales

```
% Load data
a10 = readtable('a10.csv');

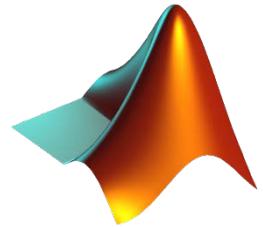
figure
% Convert table column Passengers to array
a10_Cost = table2array(a10(:, 'Cost'));
plot(a10_Cost)
h = gca;
```



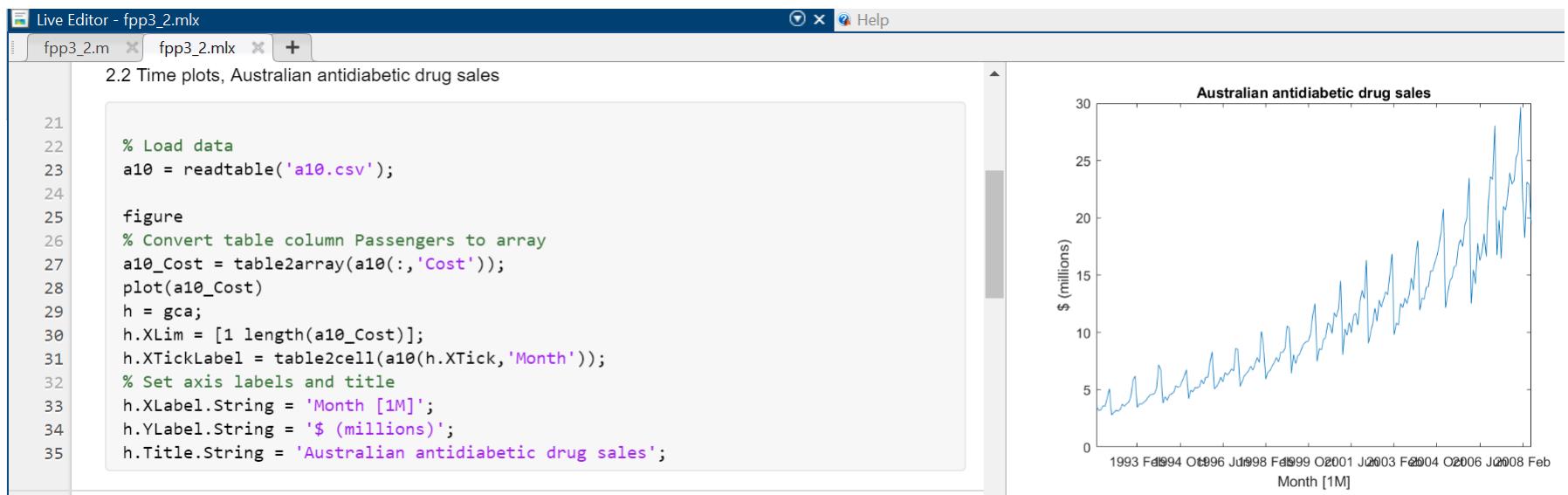
This figure shows a time series plot of passengers over 260 weeks. The y-axis is labeled 'Passengers' and ranges from 0 to 30 in increments of 5. The x-axis is labeled 'Row' and ranges from 0 to 250 in increments of 50. The data shows a general upward trend with significant weekly fluctuations, indicating seasonal patterns.

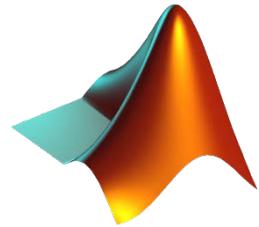


This figure is a zoomed-in view of the time series from week 23 in 1988 to week 15 in 1992. The y-axis is labeled 'Passengers (~000)' and ranges from 0 to 35 in increments of 5. The x-axis is labeled 'Week [1W]' and shows specific weeks: W23, W21, W19, W17, and W15. The plot highlights the weekly seasonal fluctuations and the overall growth in passenger numbers over the four-year period.



(Simple) time plots cont.





Seasonal plots

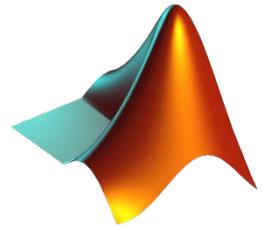
Live Editor - fpp3_2.mlx

fpp3_2.m x fpp3_2.mlx +

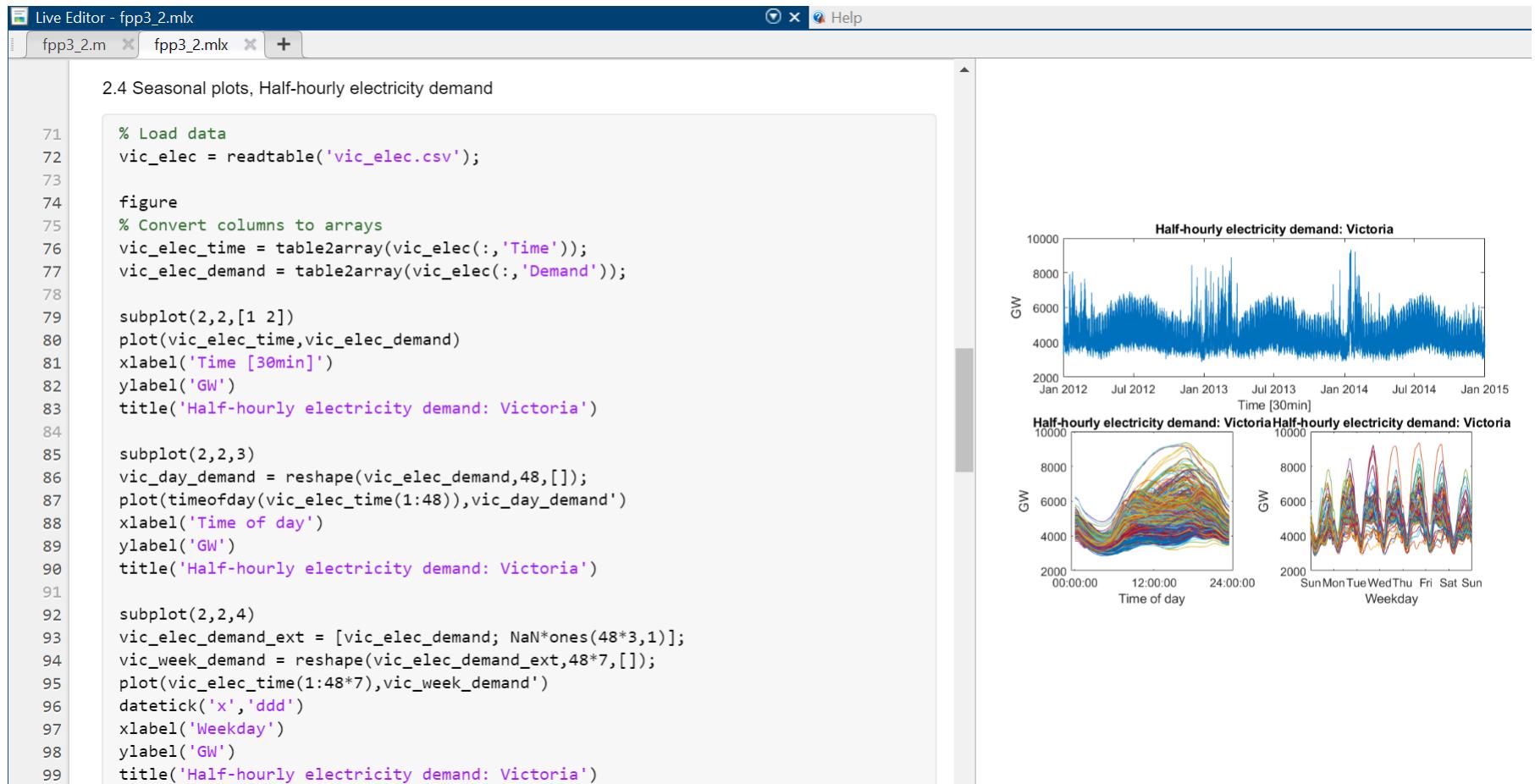
2.4 Seasonal plots, Australian antidiabetic drug sales

```
36
37 % Fill in missing months in 1991 and 2008
38 Month = {'1991 Jan';'1991 Feb';'1991 Mar';...
39     '1991 Apr';'1991 May';'1991 Jun'};
40 TotalC = [NaN; NaN; NaN; NaN; NaN; NaN];
41 Cost = [NaN; NaN; NaN; NaN; NaN; NaN];
42 a10_ext = [table(Month,TotalC,Cost), a10(:,:)];
43
44 Month = {'2008 Jul';'2008 Aug';'2008 Sep';...
45     '2008 Oct';'2008 Nov';'2008 Dec'};
46 TotalC = [NaN; NaN; NaN; NaN; NaN; NaN];
47 Cost = [NaN; NaN; NaN; NaN; NaN; NaN];
48 a10_ext = [a10_ext(:, :); table(Month,TotalC,Cost)];
49
50 Month_Labels = {'Jan','Feb','Mar','Apr','May','Jun',...
51     'Jul','Aug','Sep','Oct','Nov','Dec'};
52
53 a10_ext_Cost = table2array(a10_ext(:, 'Cost'));
54 a10_ext_Cost_r = reshape(a10_ext_Cost,12,length(a10_ext_Cost)/12);
55
56 figure
57 plot(a10_ext_Cost_r)
58 h = gca;
59 h.XLim = [0 13];
60 h.XTick = 1:12;
61 h.XTickLabel = Month_Labels;
62 Years = 1991:2008;
63 h.XLabel.String = 'Month';
64 h.YLabel.String = '$ (millions)';
65 h.Title.String = 'Seasonal plot: Antidiabetic drug sales';
66 for i=1:18
67     text(12.1,a10_ext_Cost_r(12,i),num2str(Years(i)))
68     text(.9,a10_ext_Cost_r(1,i),num2str(Years(i)),...
69         'HorizontalAlignment','right')
70 end
```

A seasonal plot titled "Seasonal plot: Antidiabetic drug sales". The y-axis represents sales in millions of dollars, ranging from 0 to 30. The x-axis represents the months from January to December. Multiple colored lines represent the sales for each year from 1991 to 2008. The plot shows a clear seasonal pattern with peaks in July/August and troughs in January/February. Sales generally show an upward trend over the period.

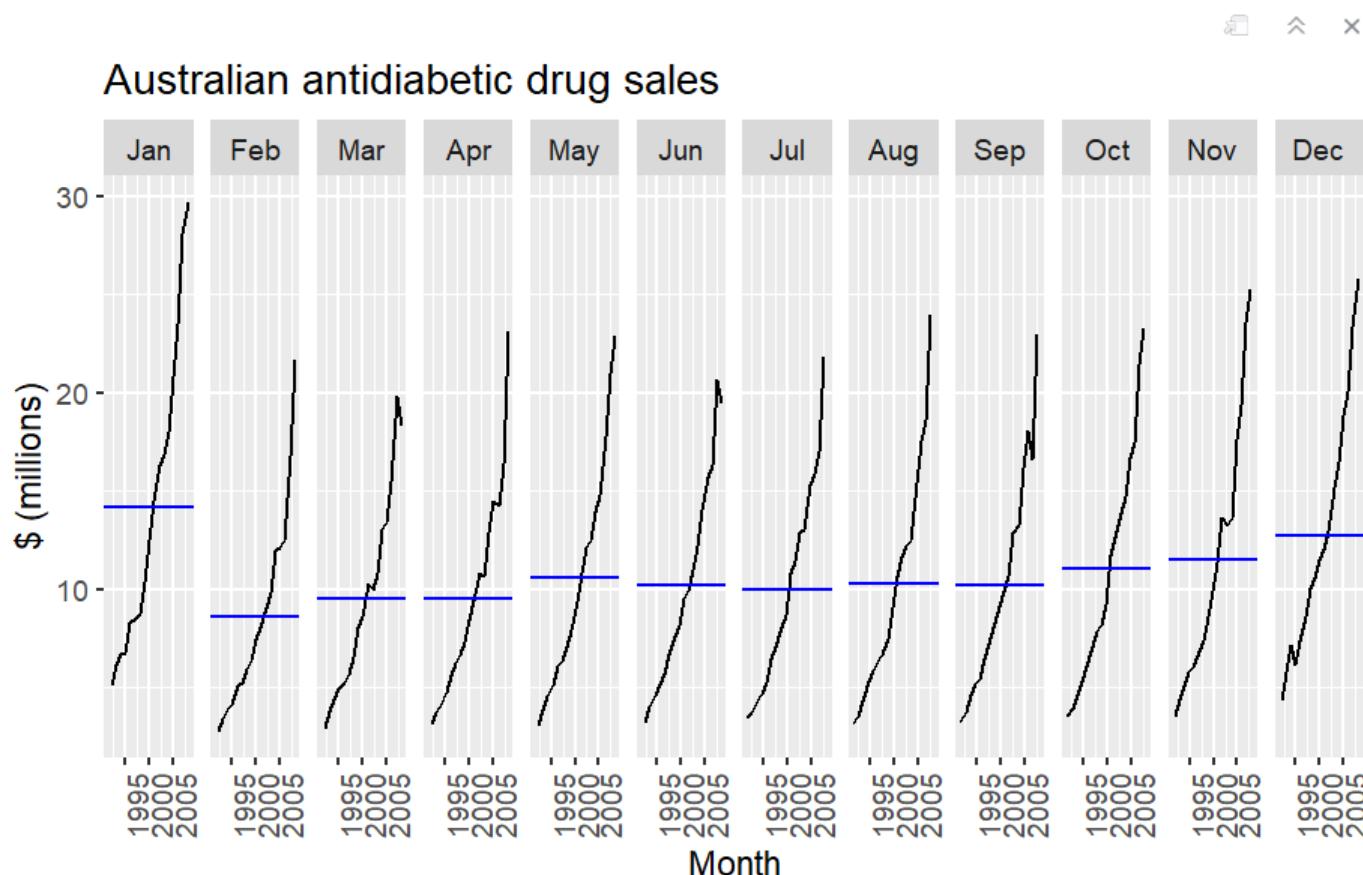


Seasonal plots cont.



Seasonal subplots: gg_subseries

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





Seasonal subplots: autoplot

- Compute the total visitor nights spent on Holiday
 - by State
 - for each quarter
 - ignoring Regions

```
tourism

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

holidays

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

A tsibble: 24,320 x 5 [1Q] Key: Region, State, Purpose [304]

Quarter <S3: yearquarter>	Region <chr>	State <chr>	Purpose <chr>	Trips <dbl>
1998 Q1	Adelaide	South Australia	Business	135.0776903
1998 Q2	Adelaide	South Australia	Business	109.9873160
1998 Q3	Adelaide	South Australia	Business	166.0346866
1998 Q4	Adelaide	South Australia	Business	127.1604643
1999 Q1	Adelaide	South Australia	Business	137.4485333
1999 Q2	Adelaide	South Australia	Business	199.9125861
1999 Q3	Adelaide	South Australia	Business	169.3550898
1999 Q4	Adelaide	South Australia	Business	134.3579372
2000 Q1	Adelaide	South Australia	Business	154.0343980
2000 Q2	Adelaide	South Australia	Business	168.7763637

1-10 of 24,320 rows Previous 1 2 3 4 5 6 ... 100 Next

Australian Capital Territory (ACT)

A tsibble: 640 x 3 [1Q] Key: State [8]

State <chr>	Quarter <S3: yearquarter>	Trips <dbl>
ACT	1998 Q1	196.21855
ACT	1998 Q2	126.77060
ACT	1998 Q3	110.67965
ACT	1998 Q4	170.47221
ACT	1999 Q1	107.77925
ACT	1999 Q2	124.64420
ACT	1999 Q3	177.94687
ACT	1999 Q4	217.65625
ACT	2000 Q1	158.41458
ACT	2000 Q2	154.80648

1-10 of 640 rows Previous 1 2 3 4 5 6 ... 64 Next

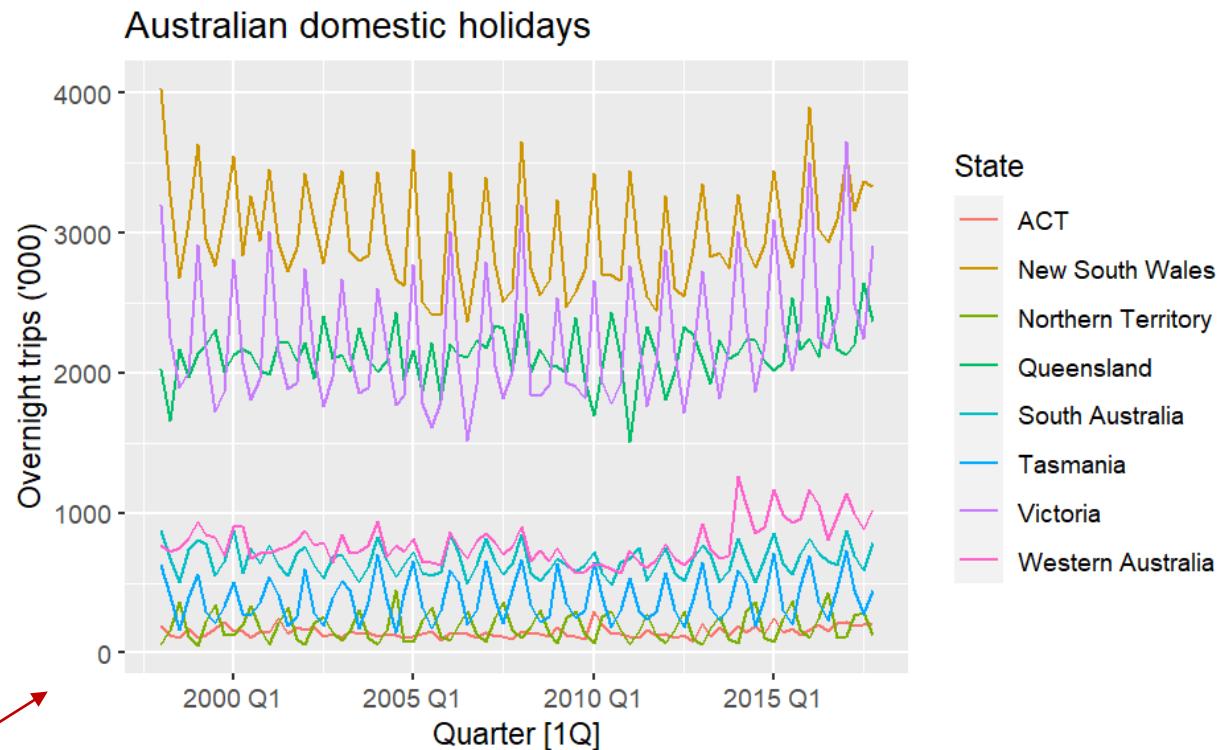
autoplot cont.

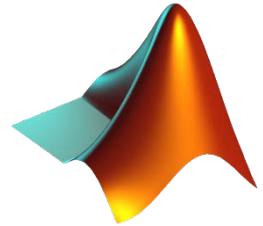
tourism

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

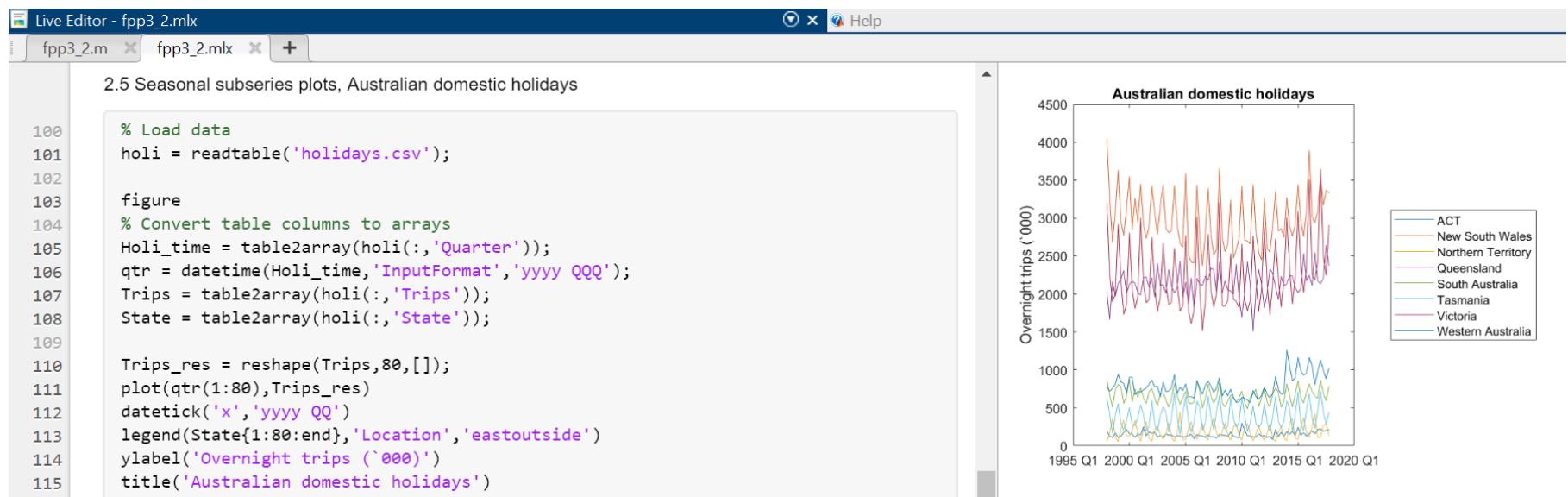
holidays

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





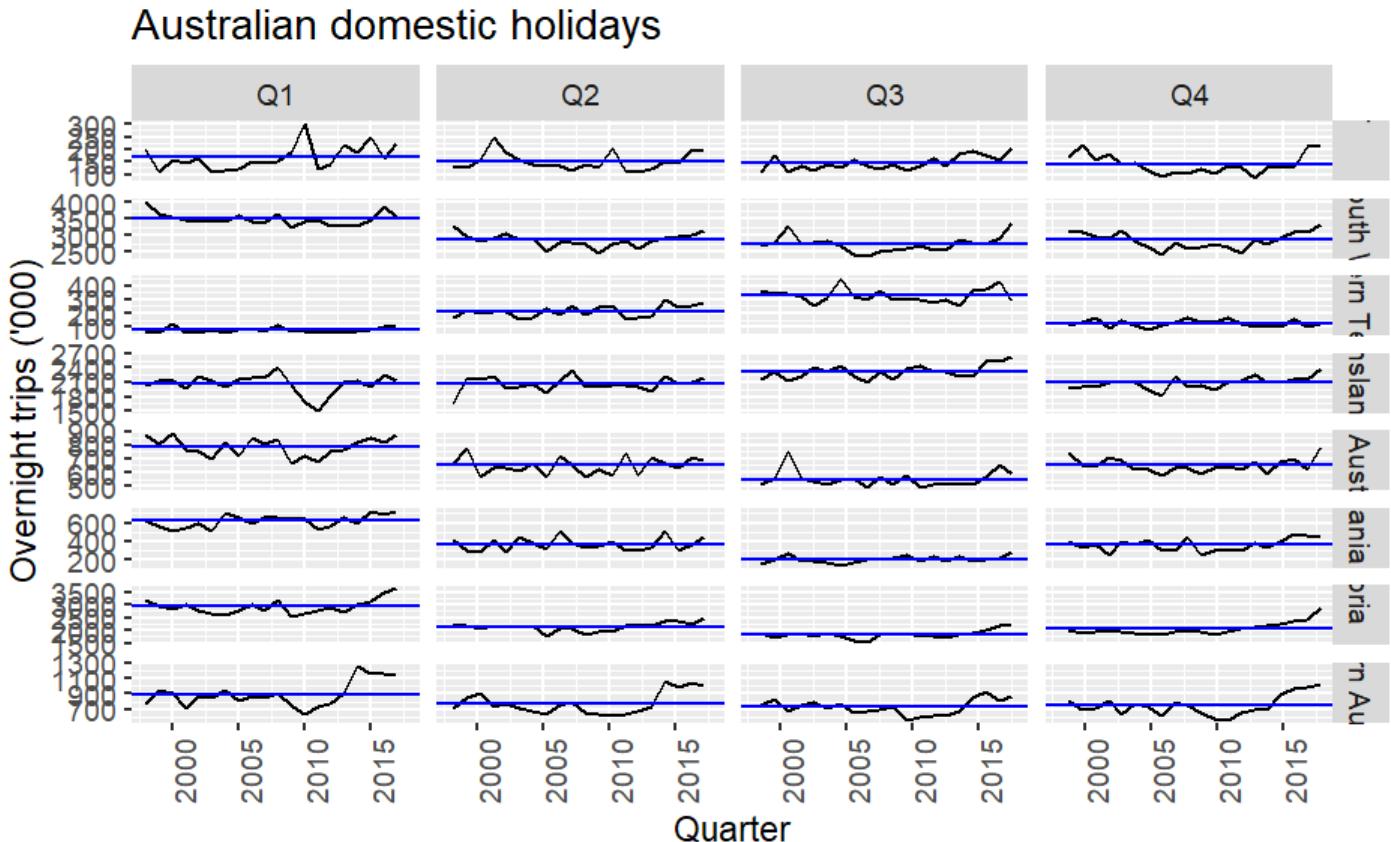
Seasonal subseries plots



gg_subseries cont.

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

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



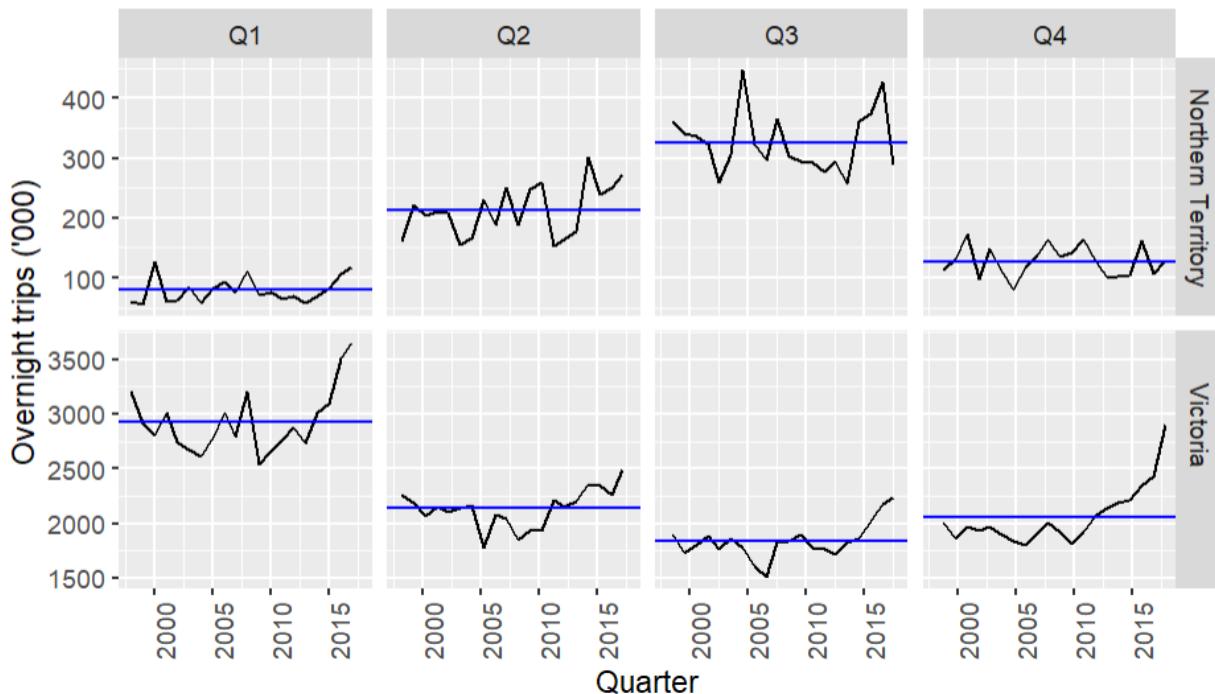


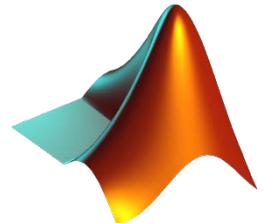
gg_subseries cont.

```
holidays %>%
  filter(State == "Northern Territory" | State == "Victoria") %>%
  gg_subseries(Trips) +
  labs(y = "Overnight trips ('000)",
       title = "Australian domestic holidays")
```

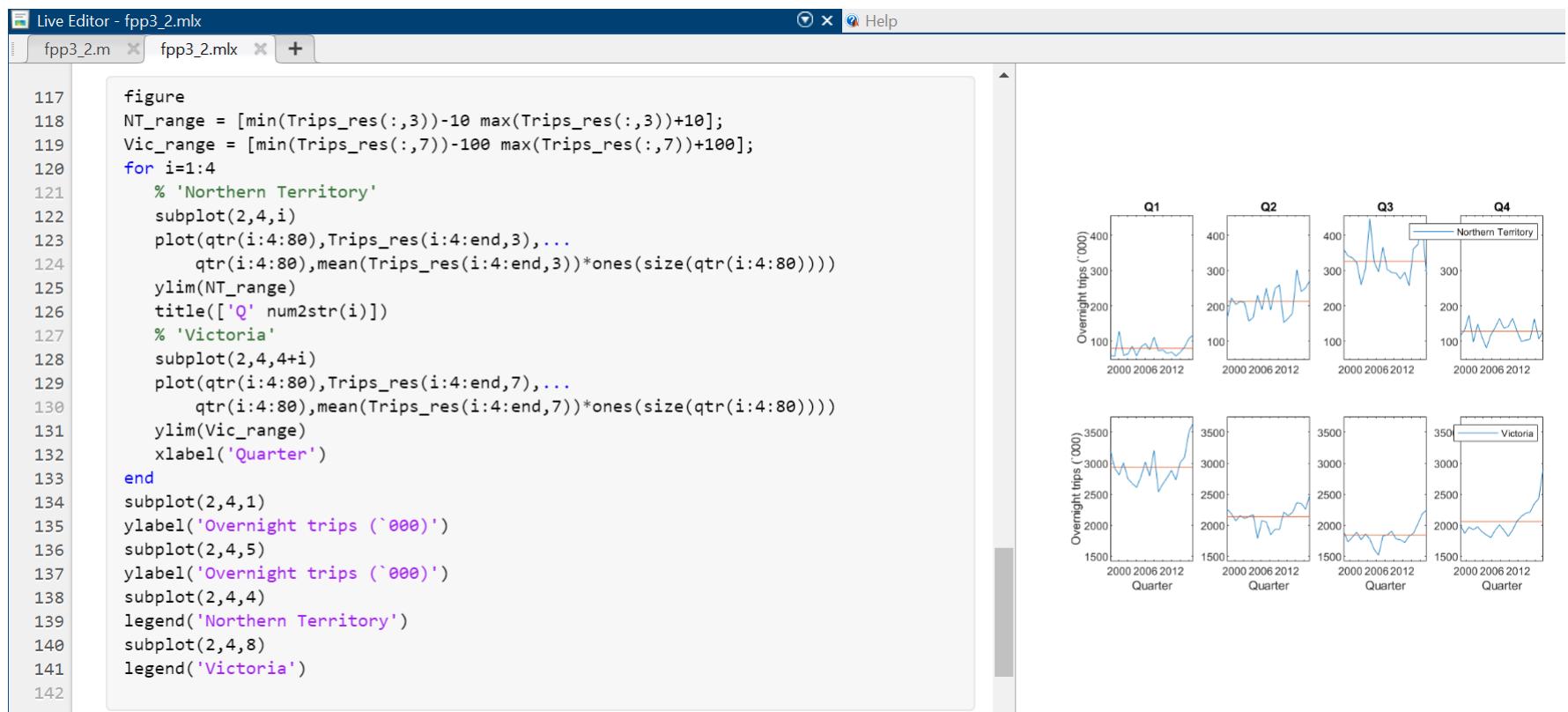


Australian domestic holidays





Seasonal subseries plots cont.

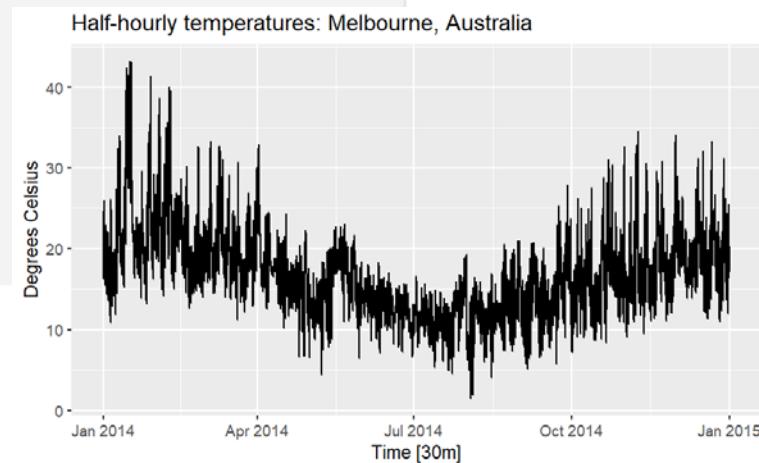
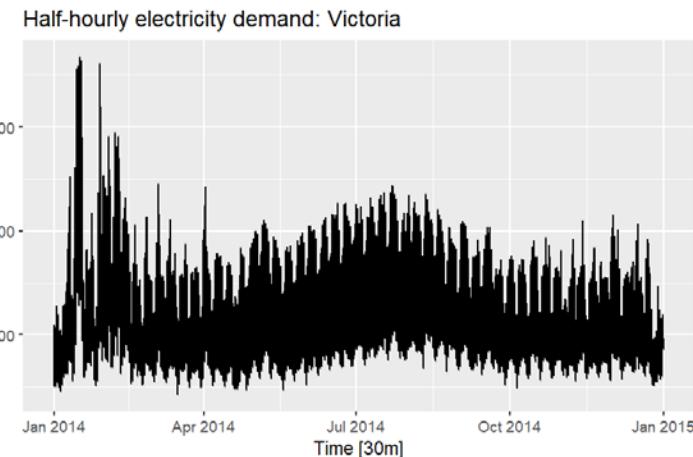
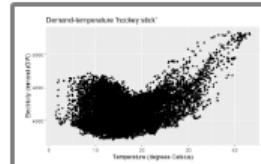
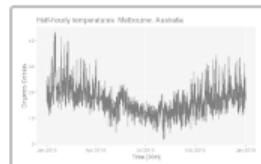
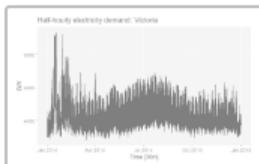


Scatterplots: ggplot

```
vic_elec %>%
  filter(year(Time) == 2014) %>%
  autoplot(Demand) +
  labs(y = "GW",
       title = "Half-hourly electricity demand: Victoria")
```

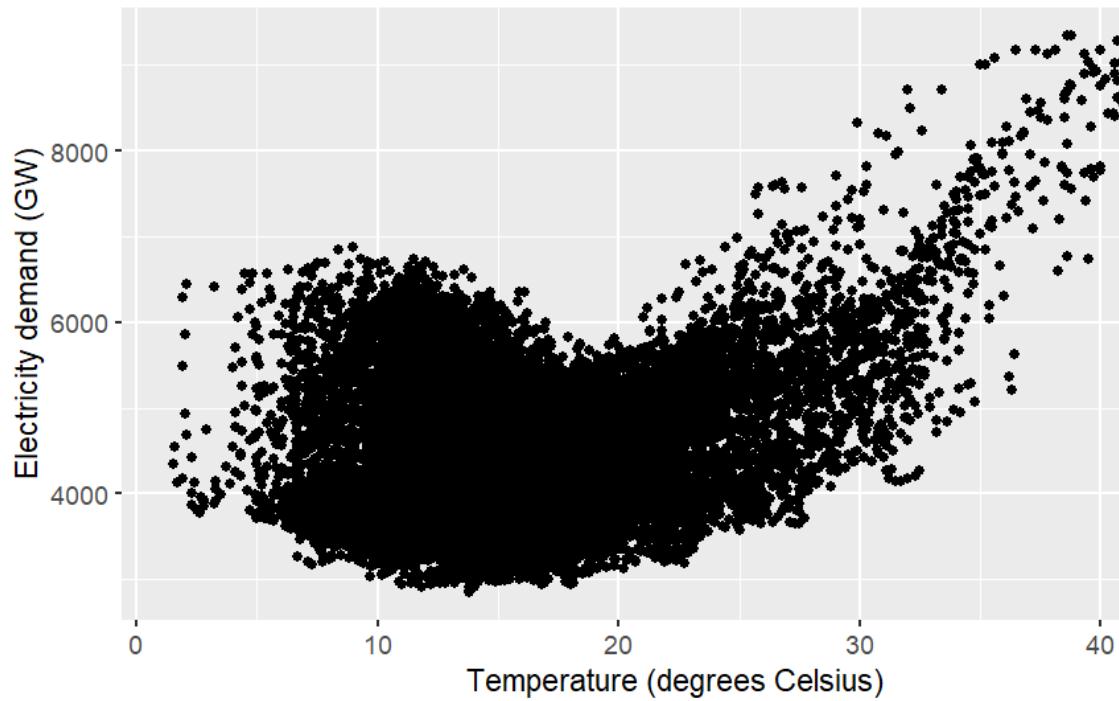
```
vic_elec %>%
  filter(year(Time) == 2014) %>%
  autoplot(Temperature) +
  labs(y = "Degrees Celsius",
       title = "Half-hourly temperatures: Melbourne, Australia")
```

```
vic_elec %>%
  filter(year(Time) == 2014) %>%
  ggplot(aes(x = Temperature, y = Demand)) +
  geom_point() +
  labs(x = "Temperature (degrees Celsius)",
       y = "Electricity demand (GW)",
       title = "Demand-temperature 'hockey stick'")
```

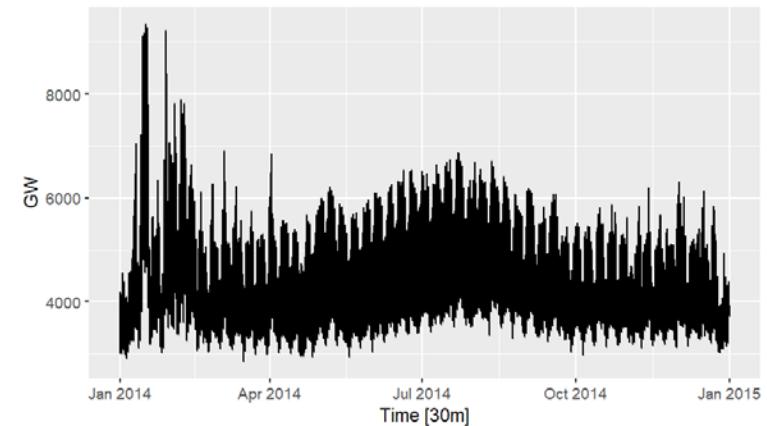


Scatterplots: ggplot

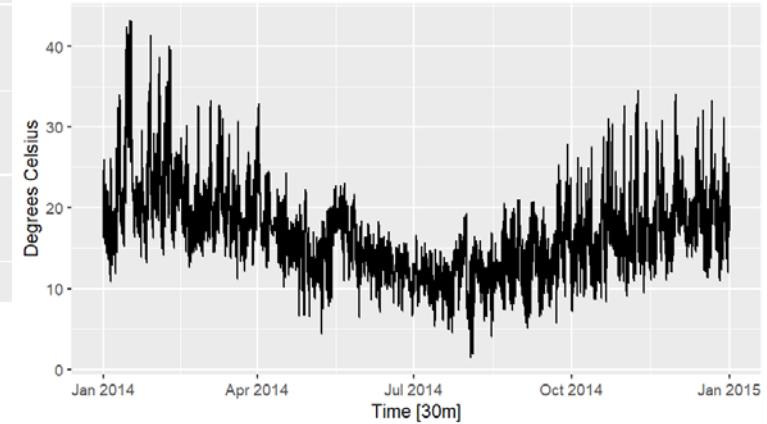
Demand-temperature 'hockey stick'

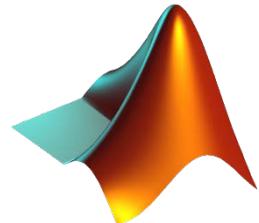


Half-hourly electricity demand: Victoria

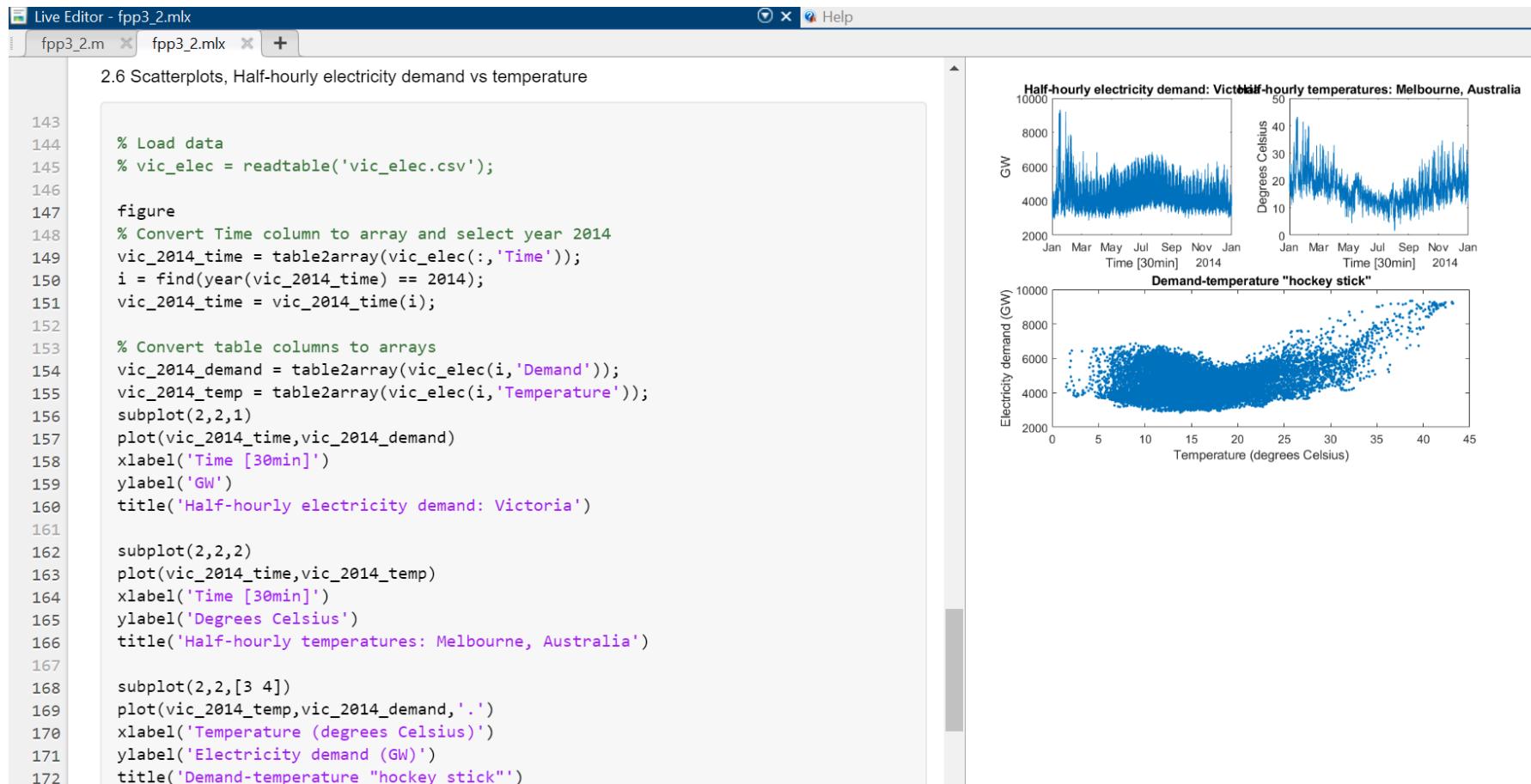


Half-hourly temperatures: Melbourne, Australia

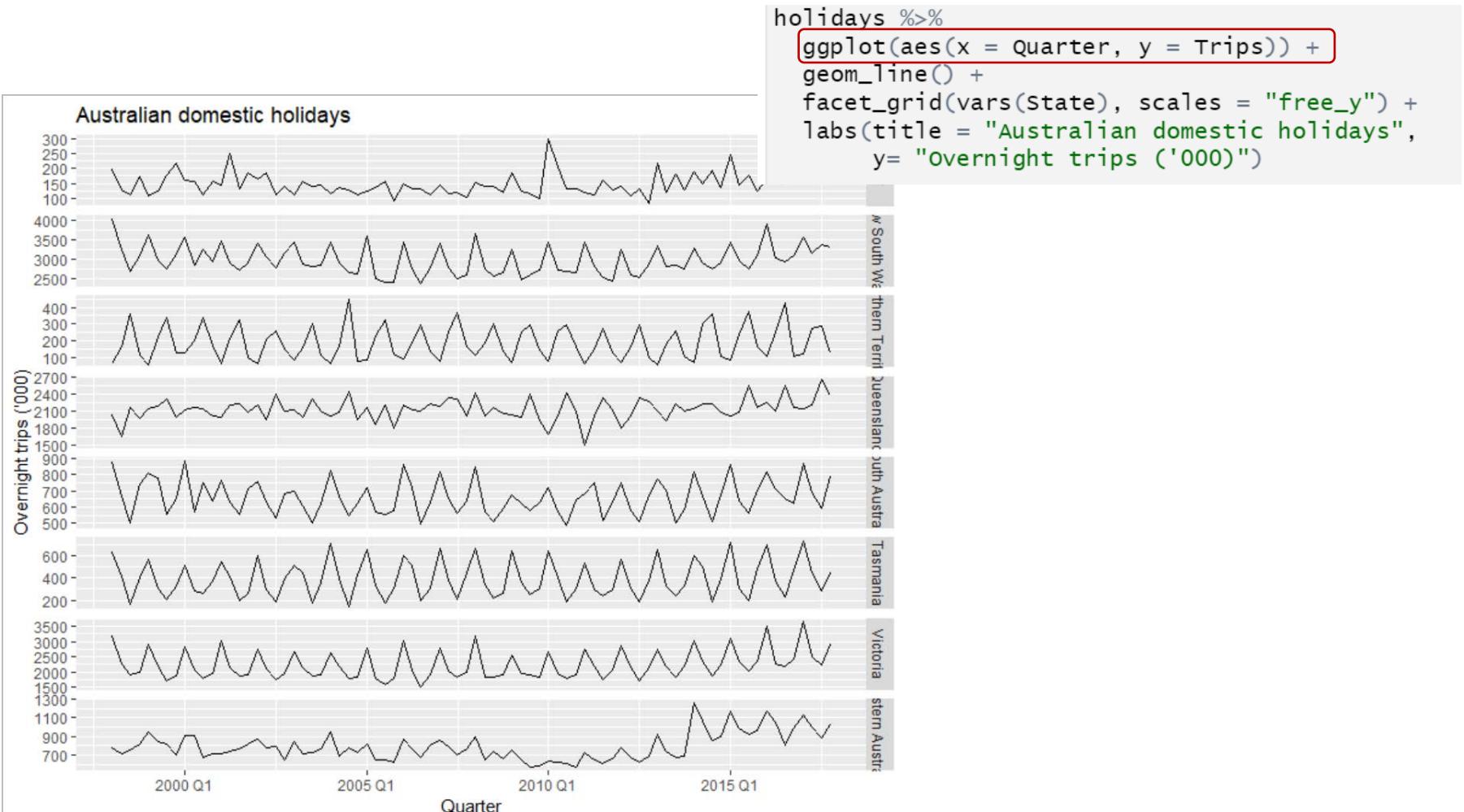




Scatterplots



Scatterplot matrices: ggplot

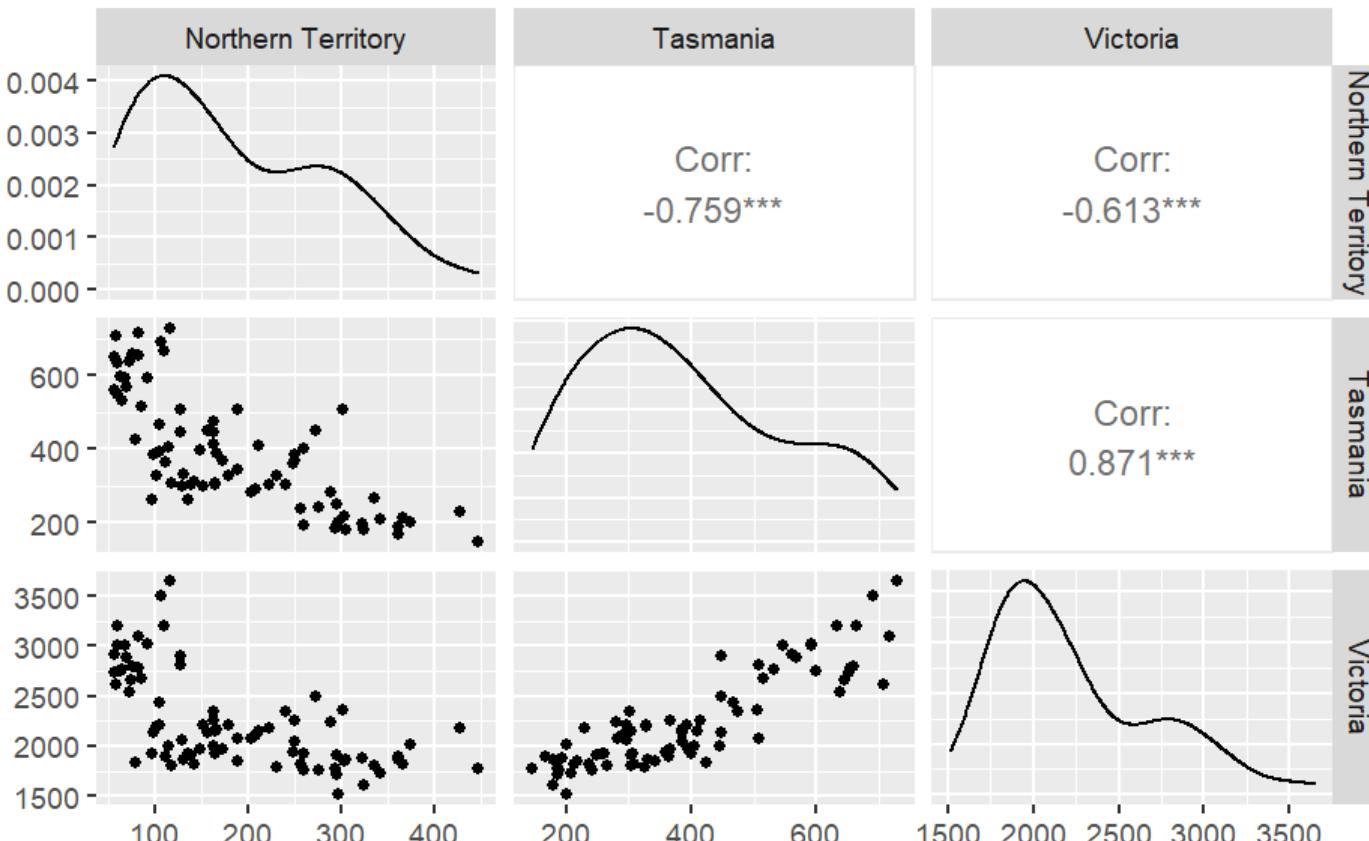


Load this library

Scatterplot matrices: GGally::ggpairs

```
library(GGally)
library(tidyverse)
library(ggfortify)
```

```
holidays %>%
  filter(state == "Northern Territory" | state == "Victoria"
    | state == "Tasmania") %>%
  pivot_wider(values_from=Trips, names_from=State) %>%
GGally::ggpairs(columns = 2:4)
```





Lagged scatterplots: gg_lag

```
recent_production <- aus_production %>%
  filter(year(Quarter) >= 2000)

recent_production

recent_production %>%
  gg_lag(Beer) +
  labs(x = "lag(Beer, k)")

recent_production %>%
  gg_lag(Beer, geom = "point") +
  labs(x = "lag(Beer, k)")

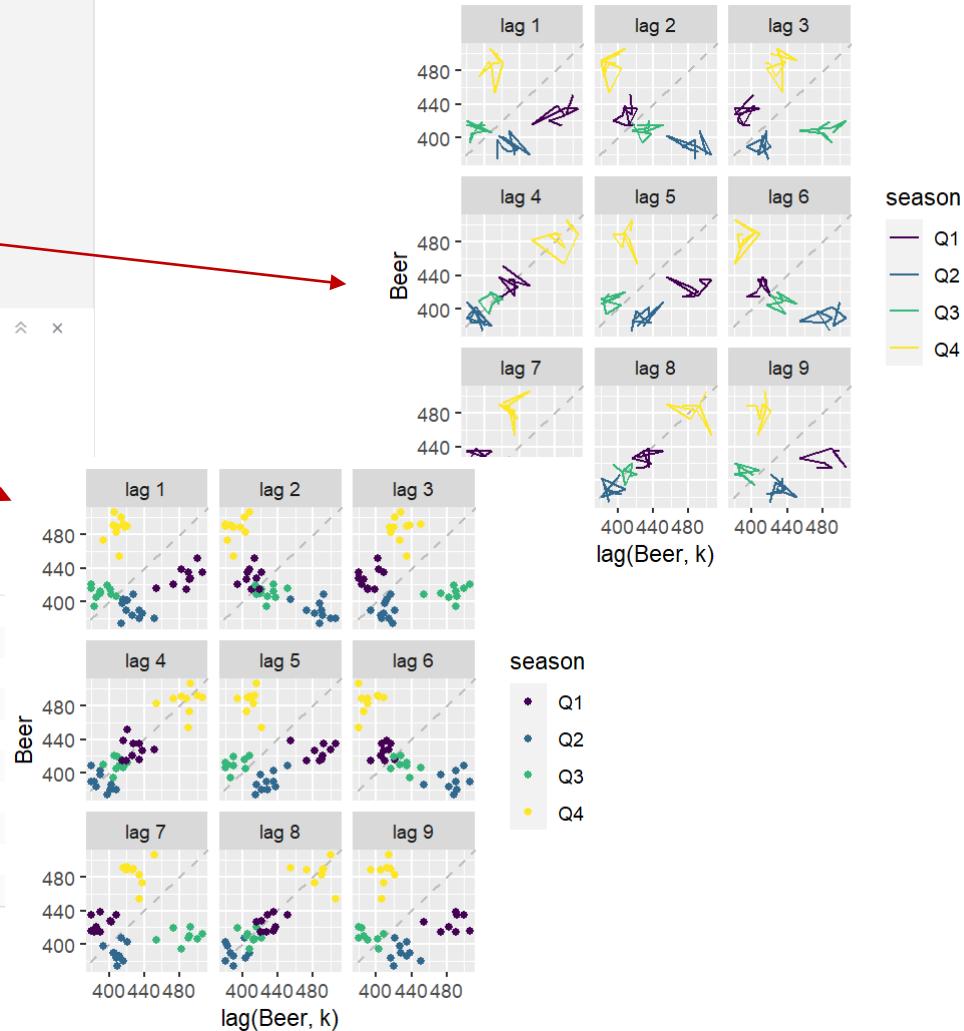
tbl_ts
```

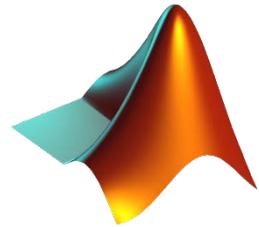
A tsibble: 42 x 7 [1Q]

Quarter	Beer <dbl>	Tobacco <dbl>	Bricks <dbl>	Cement <dbl>	Electricity <dbl>
2000 Q1	421	5169	416	1835	49320
2000 Q2	402	4860	447	2070	50670
2000 Q3	414	5185	421	1898	52623
2000 Q4	500	4763	379	1652	49350
2001 Q1	451	4217	304	1554	51658
2001 Q2	380	4959	337	1717	51103
2001 Q3	416	5196	385	1679	52226
2001 Q4	492	4522	381	1836	50778
2002 Q1	428	3843	345	1729	50639
2002 Q2	408	4806	405	1992	51486

1-10 of 42 rows

Previous 1 2 3 4 5





Lagged scatterplots

Live Editor - fpp3_2.mlx

fpp3_2.m fpp3_2mlx +

2.7 Lagged scatterplots, Beer production (2000-...)

```
% Load data
recent_production = readtable('recent_production.csv');

figure
% Convert Quarter column to datetime format and select years 2000-...
prod_time = table2array(recent_production(:, 'Quarter'));
qtr = datetime(prod_time, 'InputFormat', 'yyyy QQQ');
i = find(year(qtr) >= 2000);
qtr = qtr(i);
beer = table2array(recent_production(i, 'Beer'));

beer_range = [min(beer)-10 max(beer)+10];
for i=1:9
    subplot(3,3,i)
    plot(beer(1:4:end-i),beer(i+1:4:end),'.',...
        beer(2:4:end-i),beer(i+2:4:end),'.',...
        beer(3:4:end-i),beer(i+3:4:end),'.',...
        beer(4:4:end-i),beer(i+4:4:end),'.')
    hold on
    plot(beer_range,beer_range,'--','color',.5*[1 1 1])
    hold off
    xlim(beer_range)
    ylim(beer_range)
    axis('square')
    title(['lag ' num2str(i)])
end

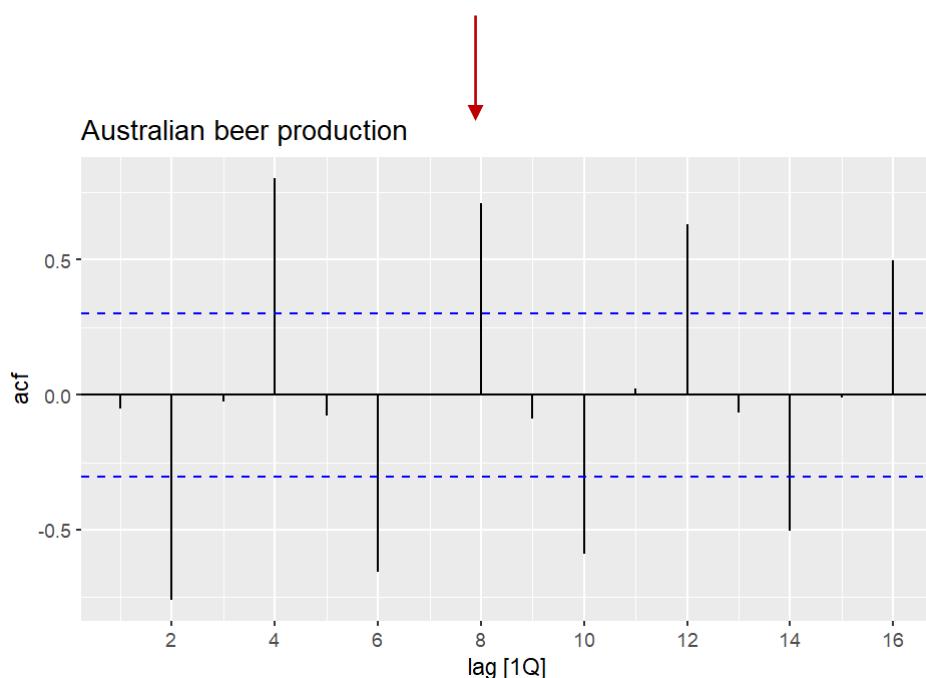
subplot(3,3,6)
legend('Q1','Q2','Q3','Q4')
subplot(3,3,4)
ylabel('Beer')
subplot(3,3,8)
xlabel('lag(Beer,k)')
```

A 3x3 grid of 9 lagged scatterplots showing beer production over time. Each plot has 'Beer' on the y-axis and 'lag(Beer,k)' on the x-axis, ranging from 400 to 500. A dashed diagonal line represents the identity line. Data points are colored by quarter: Q1 (blue), Q2 (red), Q3 (orange), and Q4 (purple). The plots are labeled 'lag 1' through 'lag 9'.



Autocorrelation Function (ACF)

```
recent_production %>% ACF(Beer)  
recent_production %>% ACF(Beer) %>% autoplot() +  
  labs(title="Australian beer production")
```



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

A tsibble: 16 x 2 [1Q]

lag	acf
1Q	-0.052981076
2Q	-0.758175440
3Q	-0.026233757
4Q	0.802204530
5Q	-0.077471204
6Q	-0.657451271
7Q	0.0011194922
8Q	0.707254078
9Q	-0.088756255
10Q	-0.587629908

ACF cont.

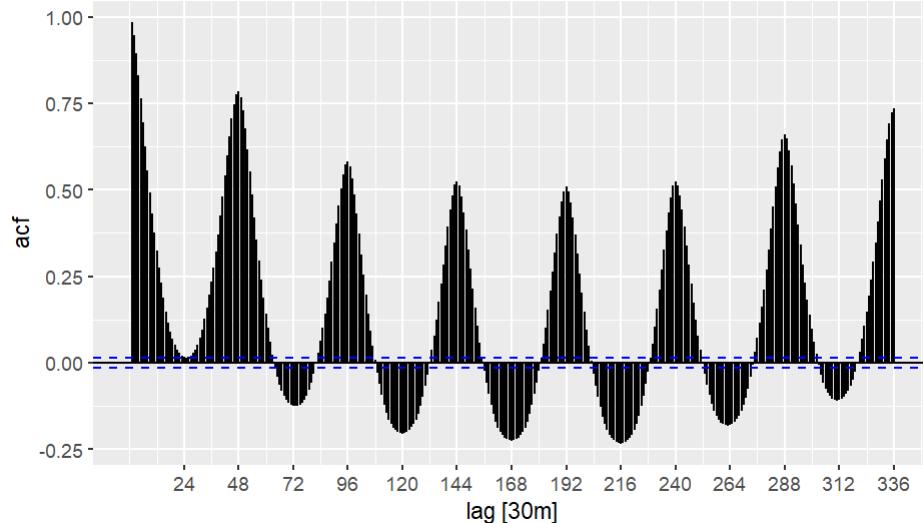
```

vic_elec %>%
  filter(year(Time) == 2014) %>% ACF(Demand, lag_max = 336) %>%
  autoplot() +
  labs(title = "Half-hourly electricity demand: Victoria")

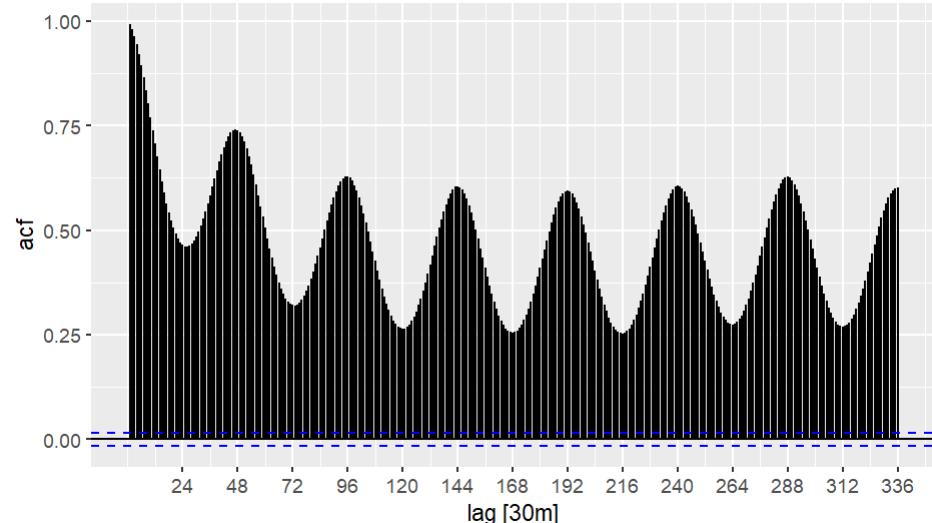
vic_elec %>%
  filter(year(Time) == 2014) %>% ACF(Temperature, lag_max = 336) %>%
  autoplot() +
  labs(title = "Half-hourly temperatures: Melbourne, Australia")

```

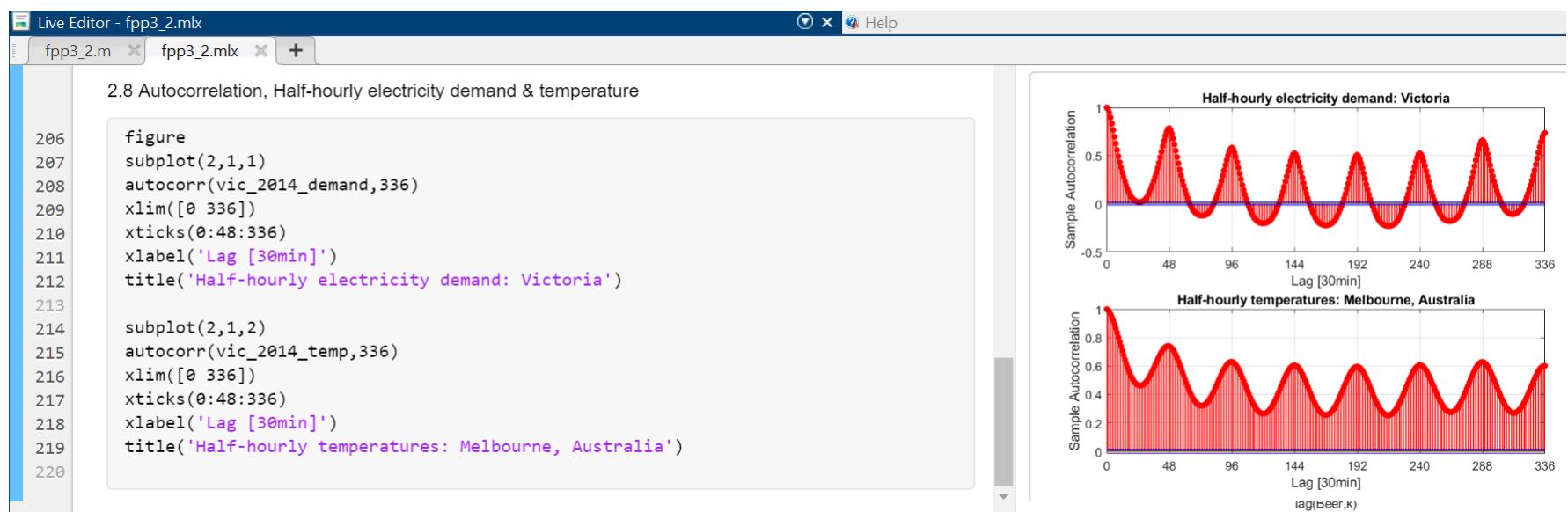
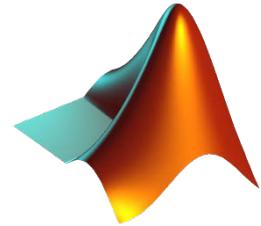
Half-hourly electricity demand: Victoria



Half-hourly temperatures: Melbourne, Australia



ACF



Which is which?

- When data are
 - **Trended** – the ACF for small lags tends to be large and positive
 - **Seasonal** – the ACF will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency)
 - **Trended & seasonal** – we see a combination of these effects

