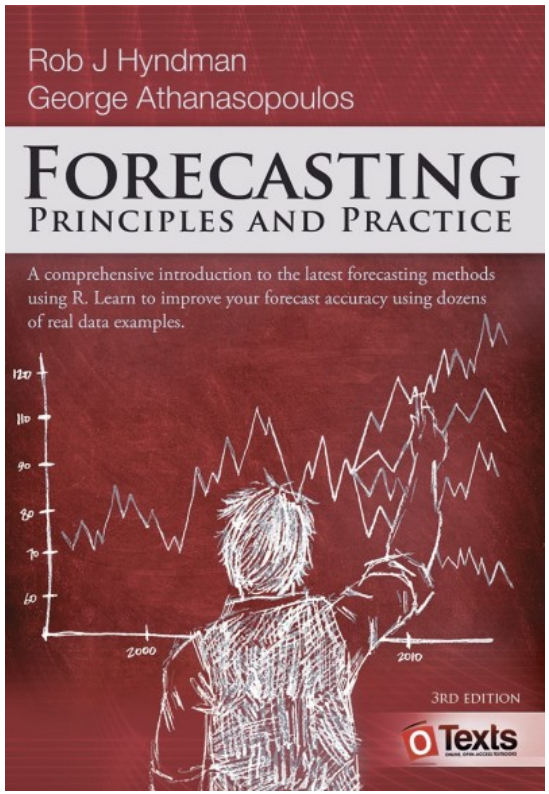
BI Workplace T02

Time series graphics

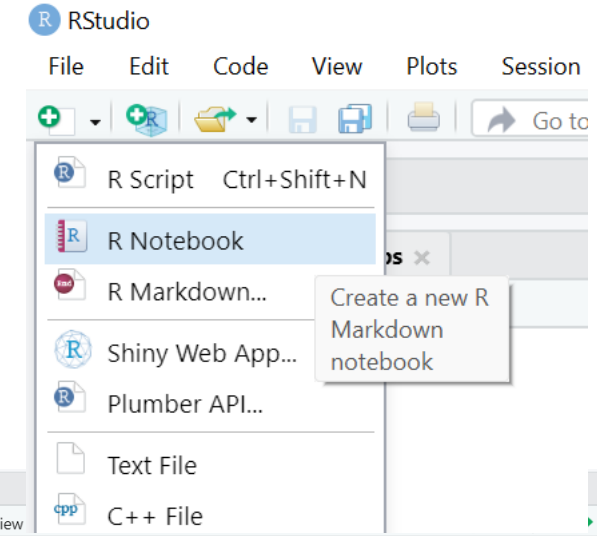
- Eaton, J.W., Bateman, D., Hauberg, S., Wehbring, R. (2021) *GNU Octave Manual: Free Your Numbers*
- Grolemund, G., Wickham. H. (2017) *R for Data Science*
- Gundersen, V.B. (2008) *Mathesaurus*, including the *MATLAB/Octave-Python-R cheatsheet*
- Hiebeler, D. (2014) *MATLAB / R Reference*
- Hyndman,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

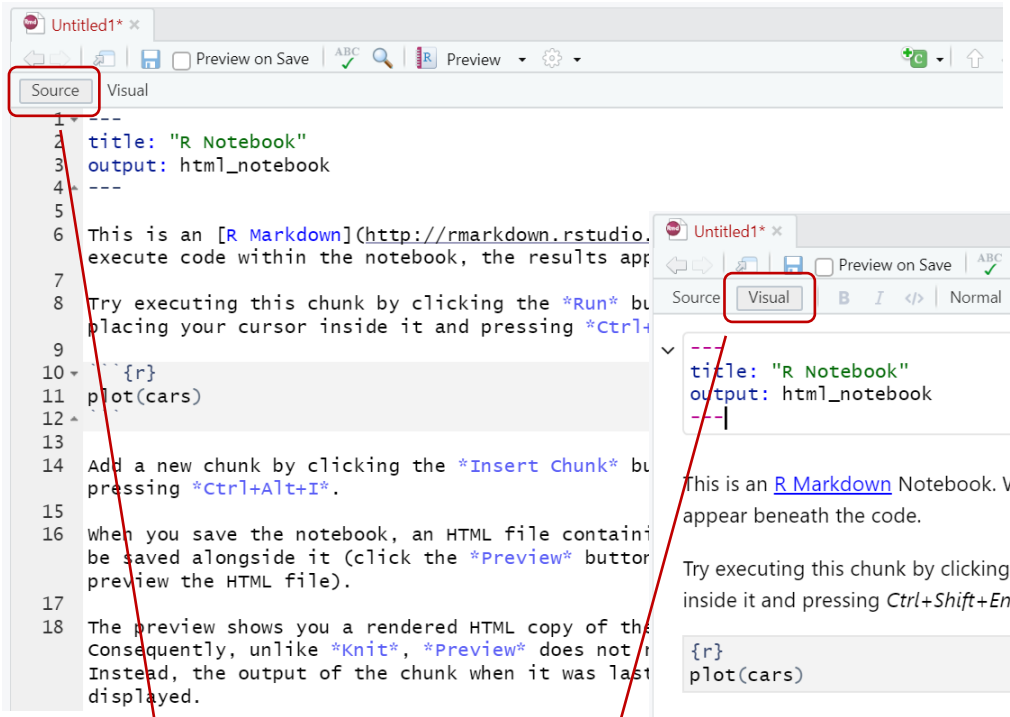
Sec. 2: Time series graphics



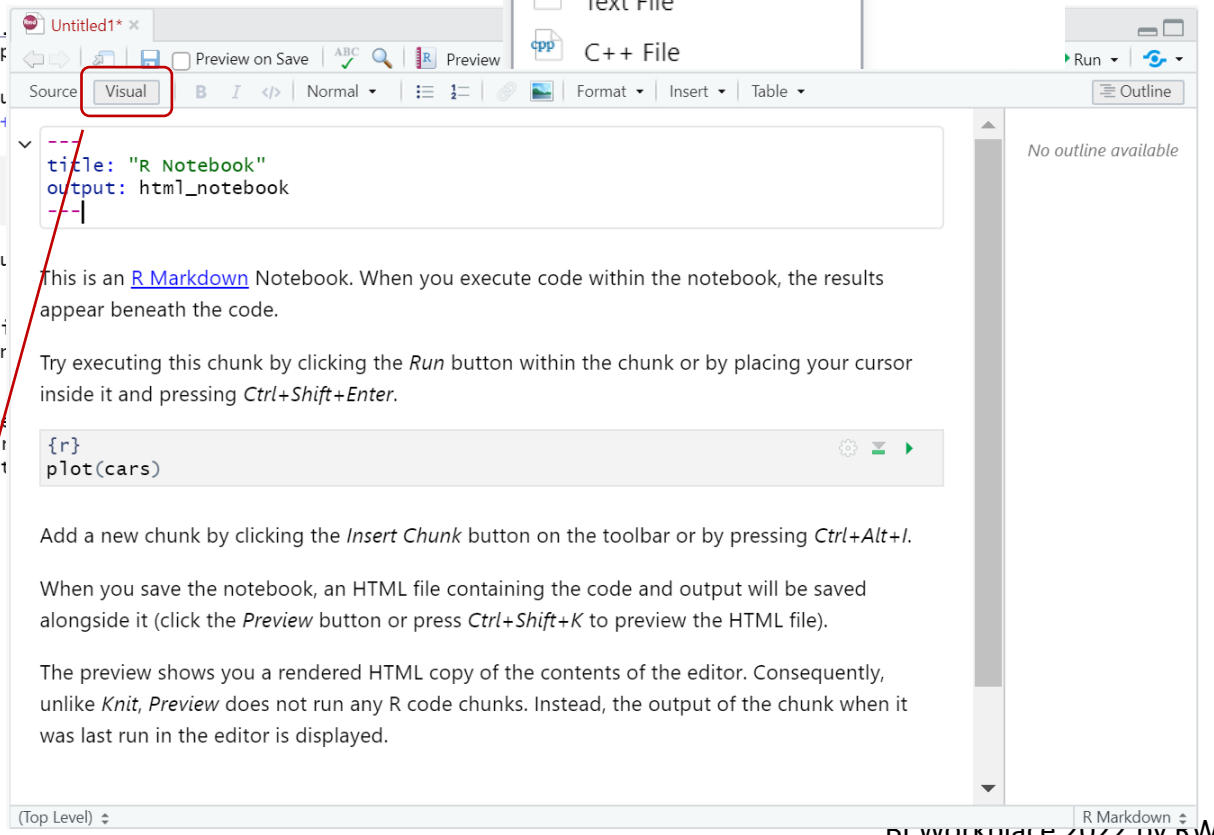
https://otexts.com/fpp3

Create a Notebook



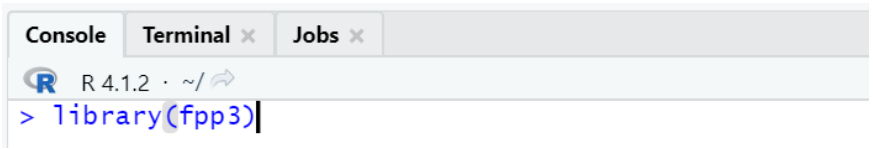


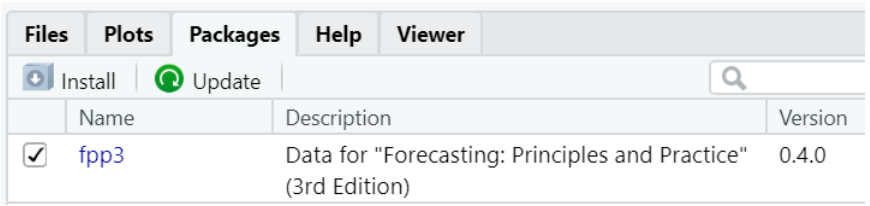
Display mode

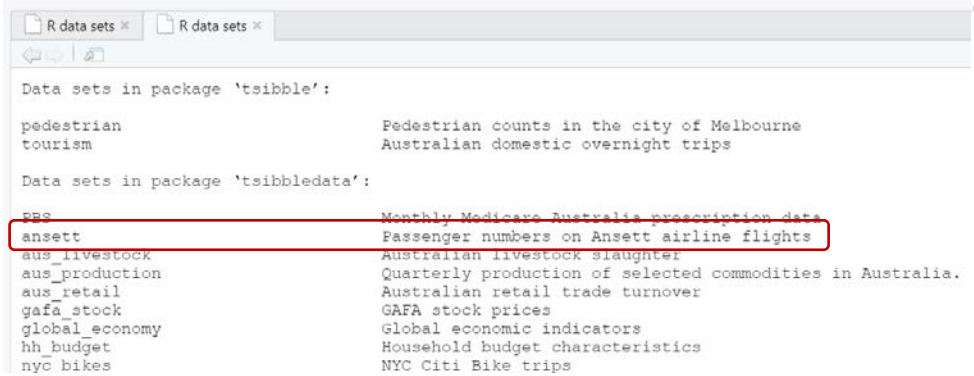


The fpp3 library

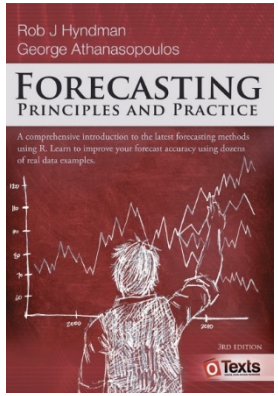
- Load the **fpp3** library
  - From the **Console** or the **Packages** list
- List the library datasets
  > data(package = 'fpp3')
  and all library datasets
  > data()







Now you can copy-paste code



fpp3_2.Rmd

Knit on Save | Knit | Run

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

Outline

- Initialize the FPP3 library
- 2.2 Time plots
- 2.4 Seasonal plots
- 2.5 Seasonal subseries plots

Display mode

fpp3_2.Rmd

Knit on Save | Knit | Run

Source | Visual | Normal | Format | Insert | Table

Outline

- Initialize the FPP3 library
- 2.2 Time plots
- 2.4 Seasonal plots
- 2.5 Seasonal subseries plots

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

Initialize the FPP3 library

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

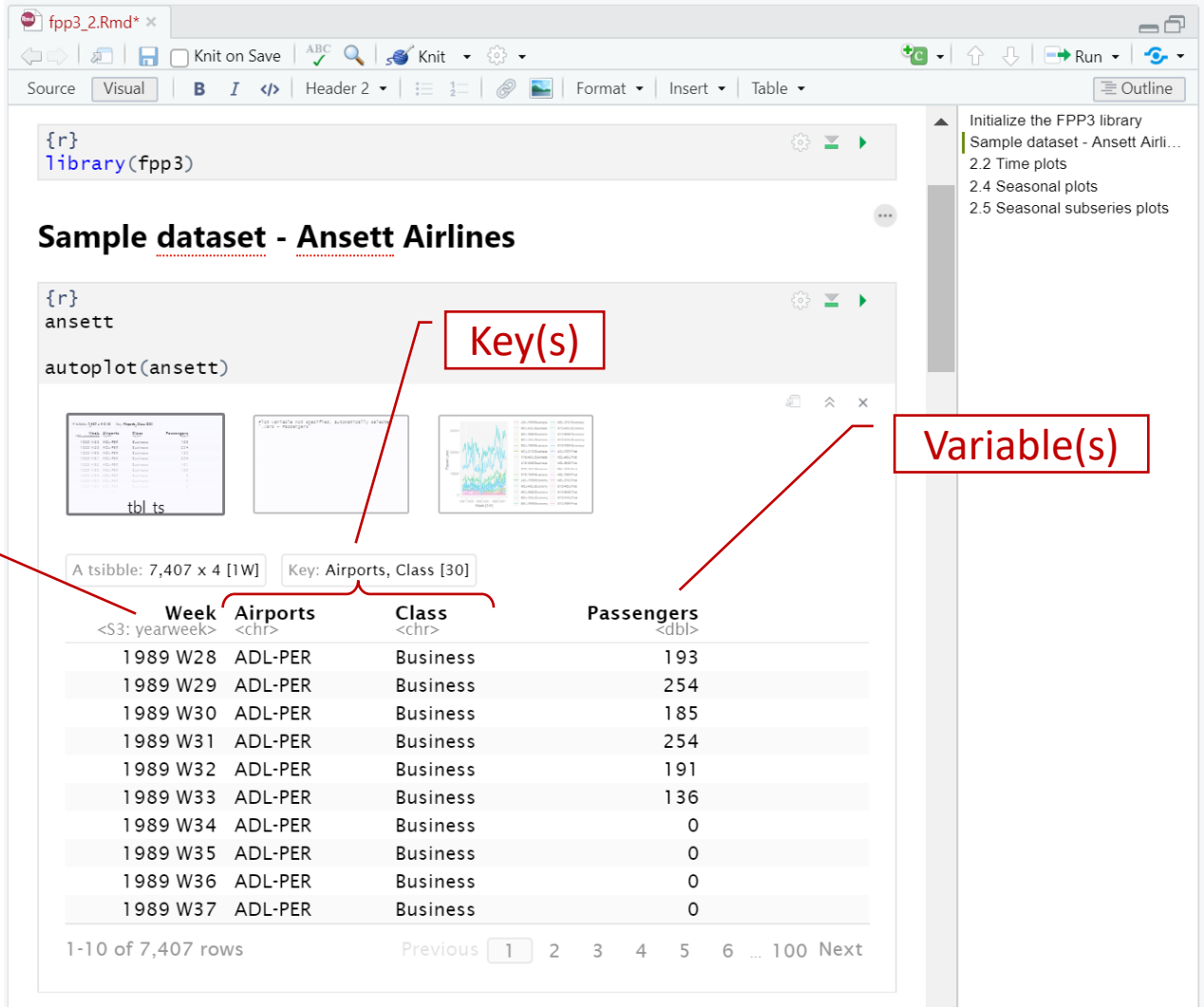
2.2 Time plots

```{r}

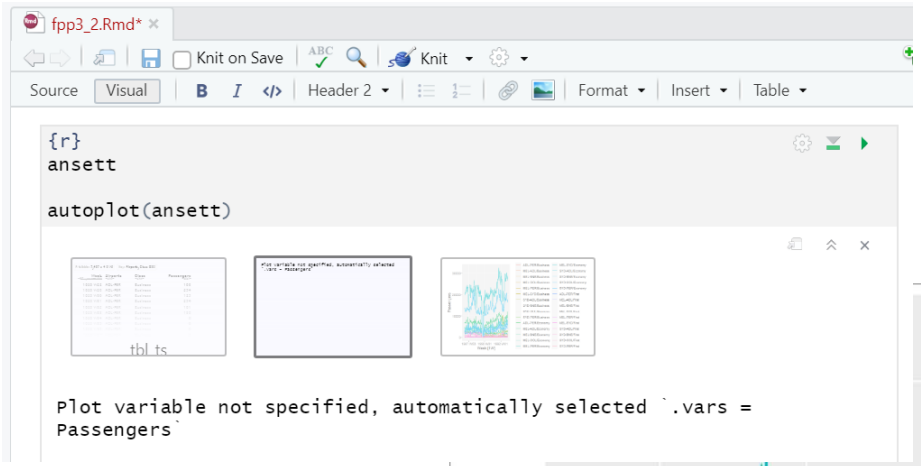
# 2.2 Time plots

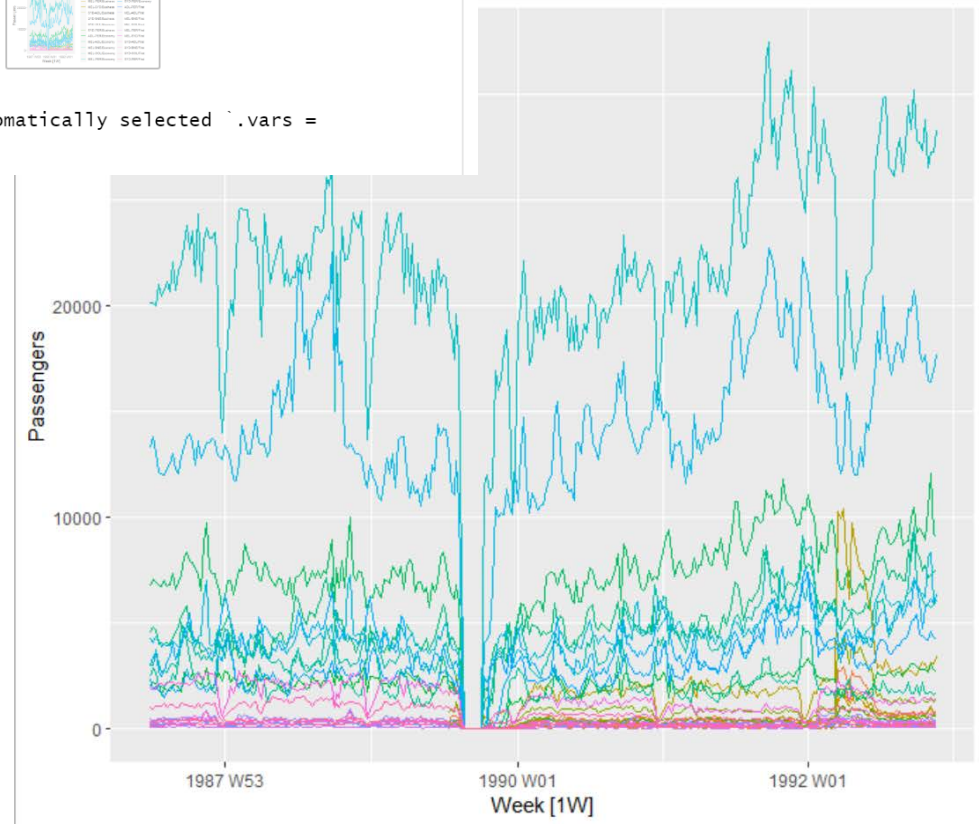
melsyd_economy <- ansett %>%
  filter(Airports == "MEL-SYD", Class == "Economy") %>%
  mutate(Passengers = Passengers/1000)
autoplot(melsyd_economy, Passengers) +
  labs(title = "Ansett airlines economy class",
       subtitle = "Melbourne-Sydney",
       y = "Passengers ('000)")
```
```

… but let's first plot a sample dataset



A warning and the result





Key(s)

Copy-paste code from Sec. 2.2

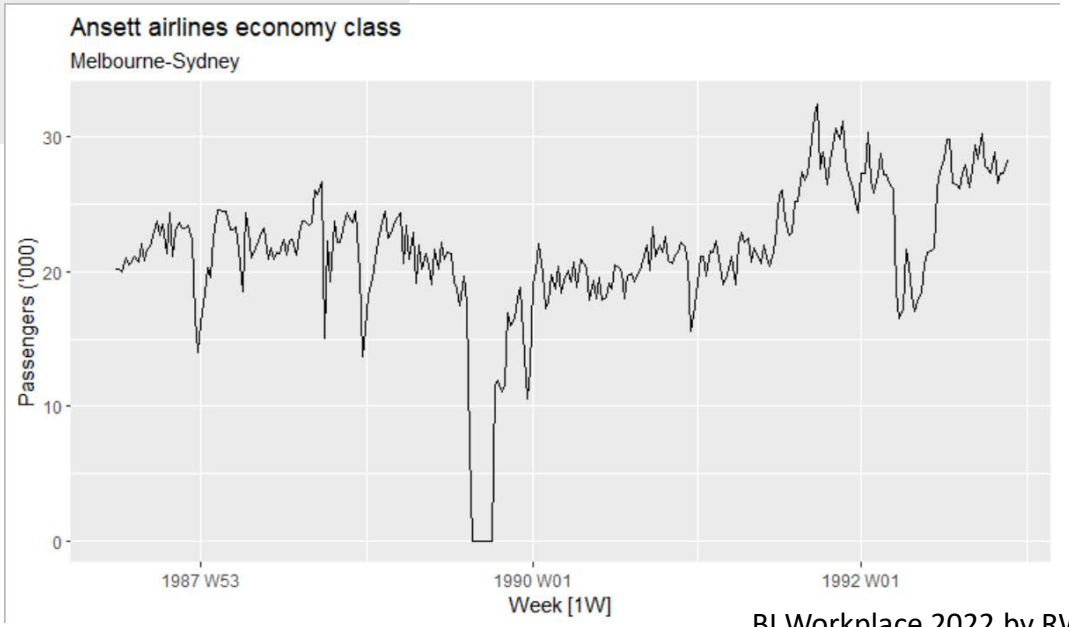
2.2 Time plots

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)")
```

Copy to clipboard

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
  <int>       <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
     <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
# ... 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
{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 <chr>	Gender <chr>	Legal <chr>	Indigenous <chr>	Count <dbl>	Quarter <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)

```
"State","Gender","Legal","Indigenous","Count","Quarter"
"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
```

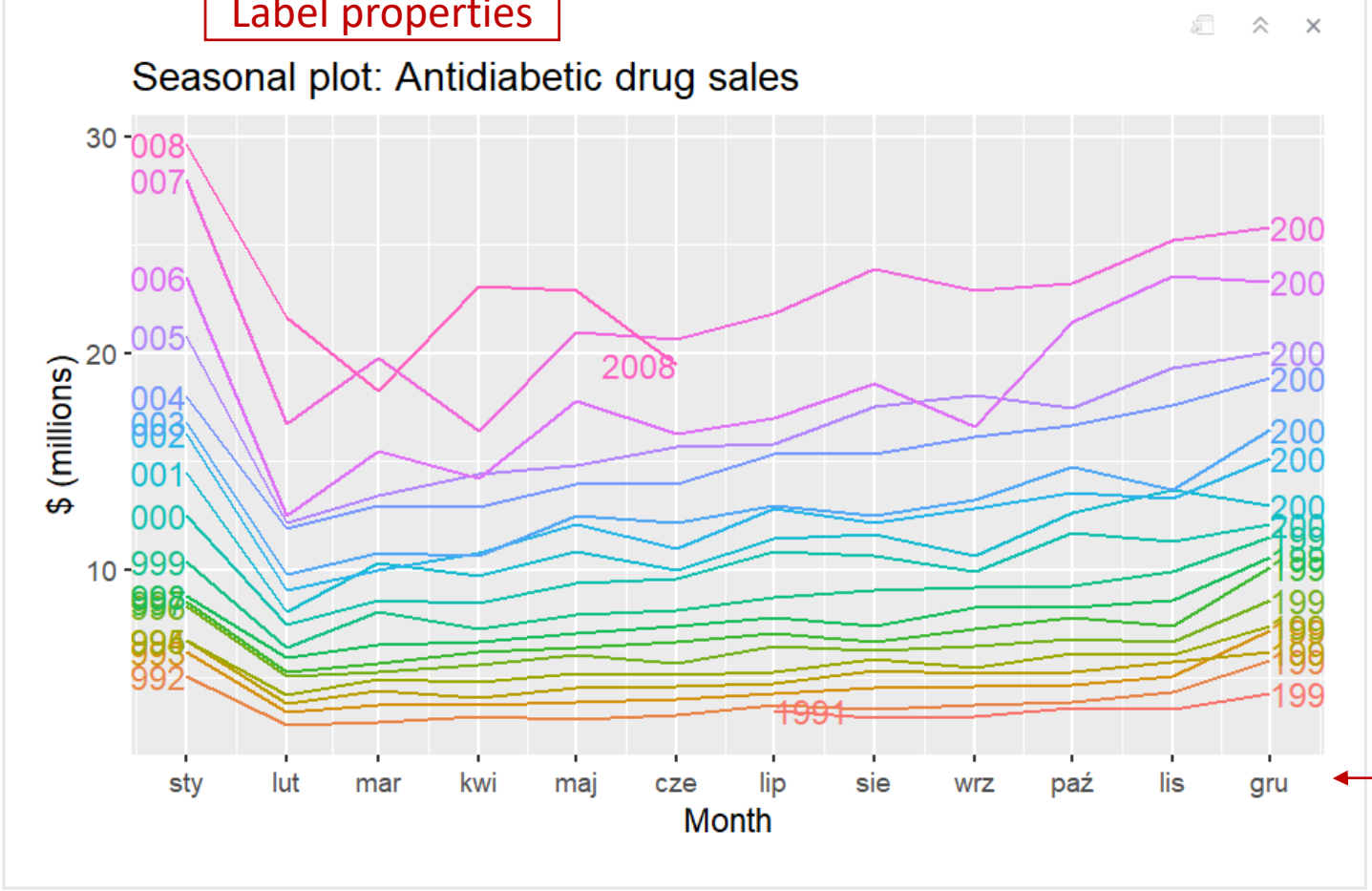
> write.csv(prison,"prison.csv", row.names = TRUE)

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

Seasonal plots: gg_season

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

Label properties

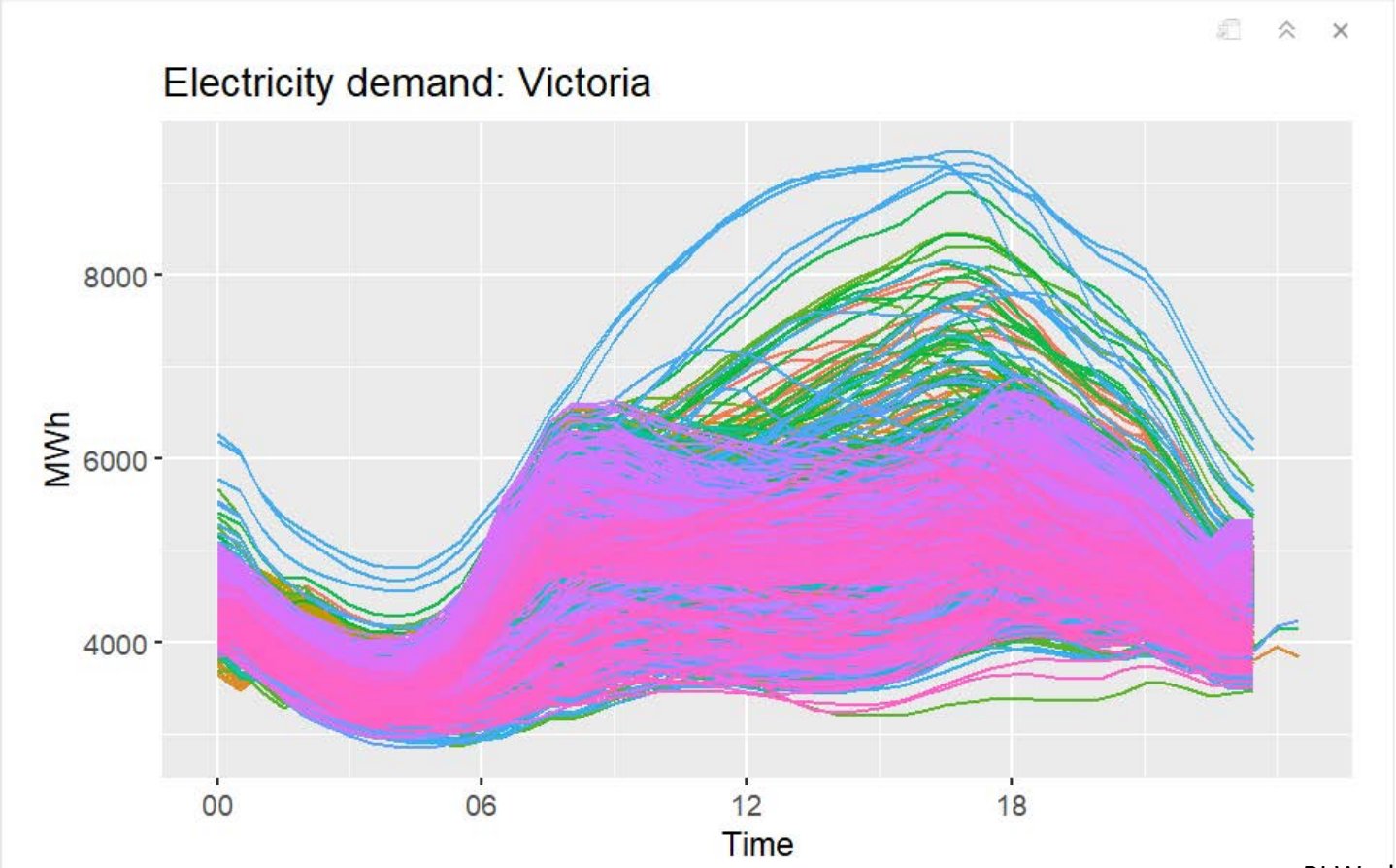



```r
{r}
Sys.setlocale("LC_TIME", "English")

[1] "English_United States.1252"
```

gg_season cont.

```
Console  Terminal  Jobs
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
```

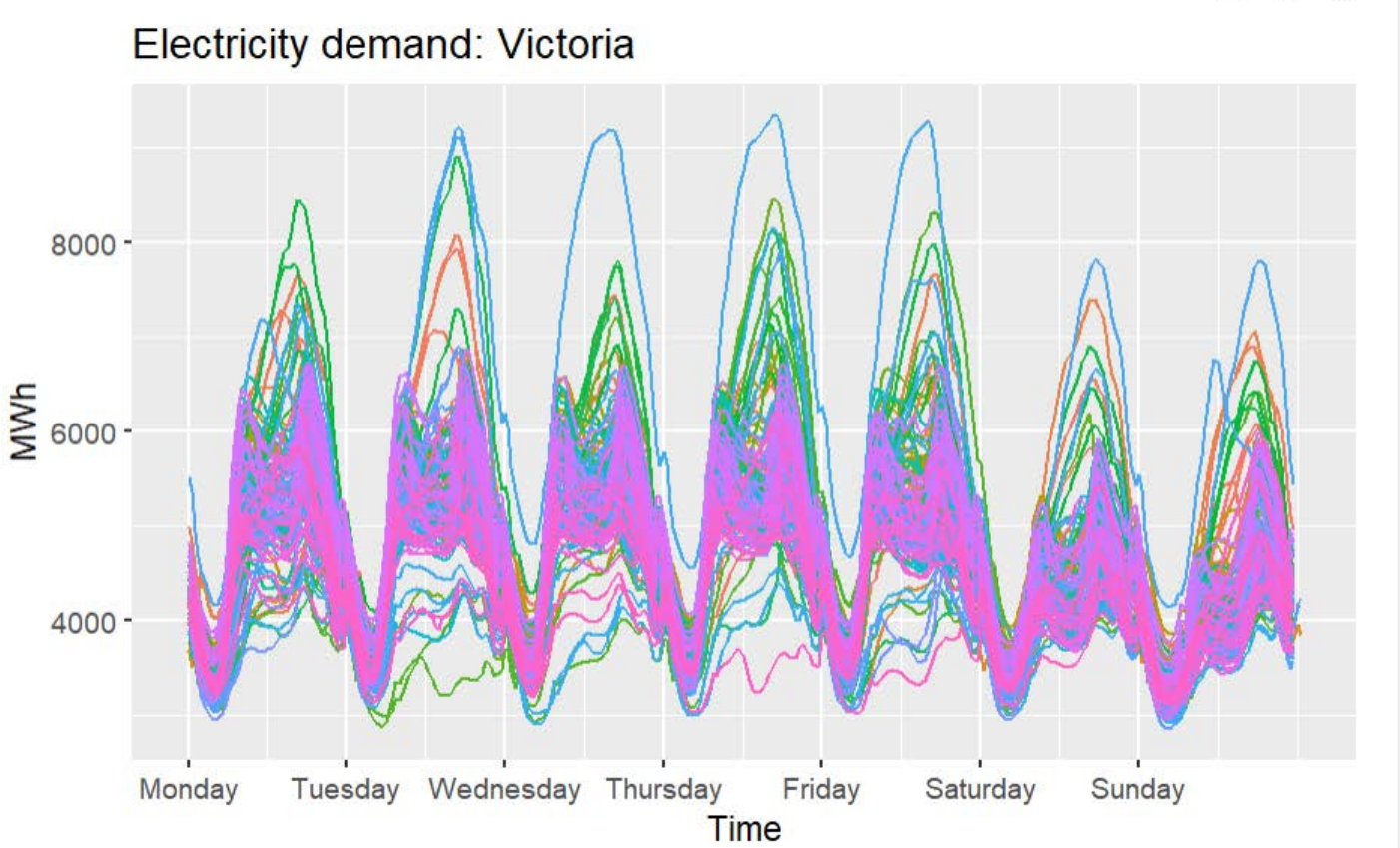
```
vic_elec %>% gg_season(Demand, period = "day") +
  theme(legend.position = "none") +
  labs(y="MWh", title="Electricity demand: Victoria")
```



gg_season cont.

"week", not "day"

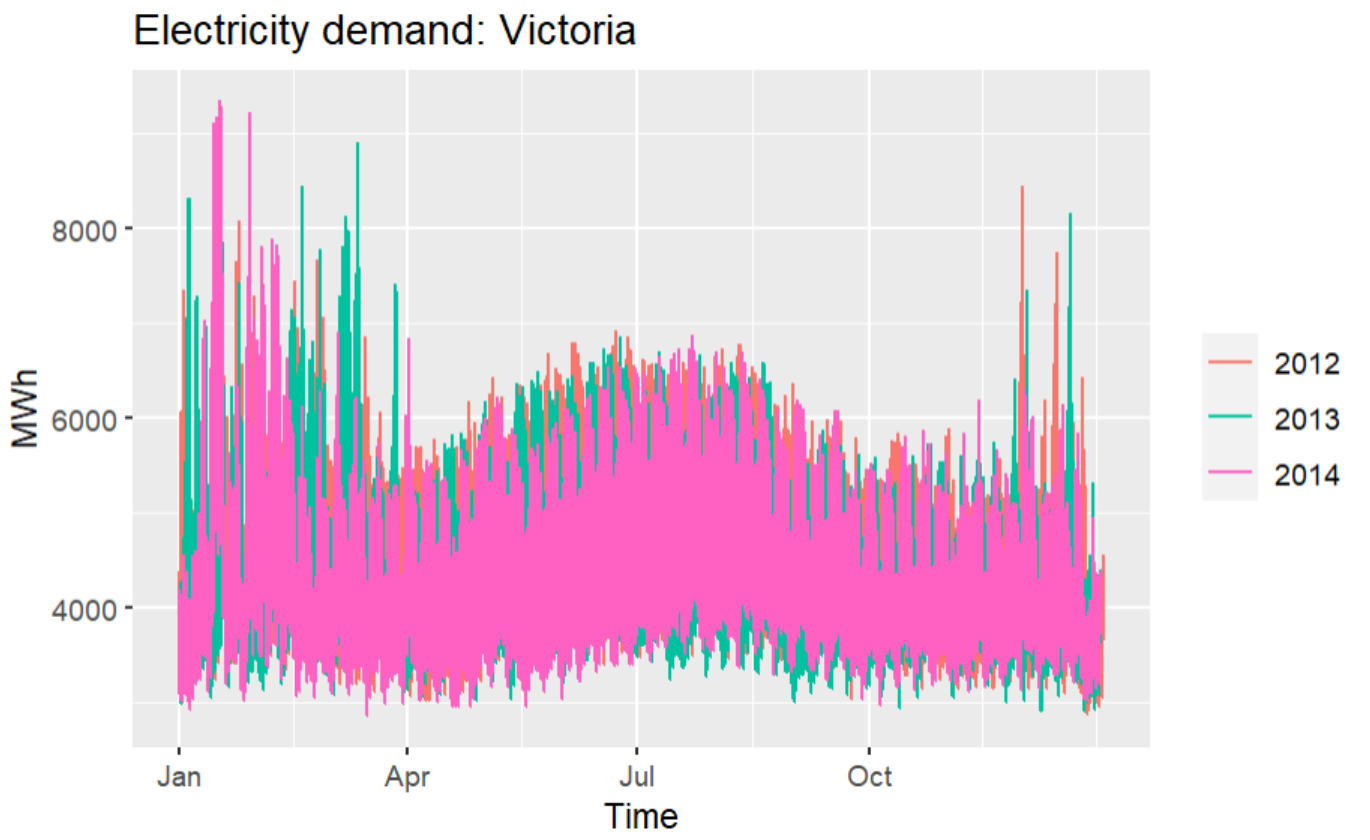
```
vic_elec %>% gg_season(Demand, period = "week") +
  theme(legend.position = "none") +
  labs(y="MWh", title="Electricity demand: Victoria")
```



gg_season cont.

"year", not "week"

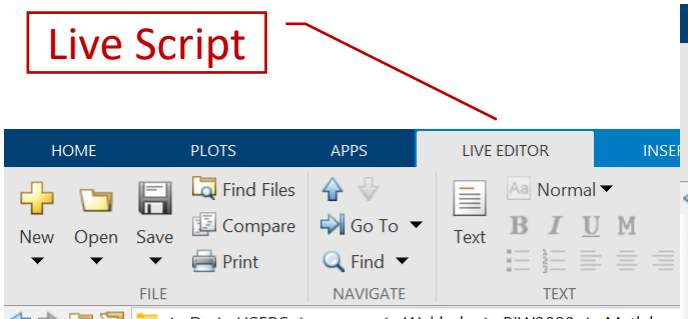
```
vic_elec %>% gg_season(Demand, period = "year") +
  labs(y="MWh", title="Electricity demand: Victoria")
```

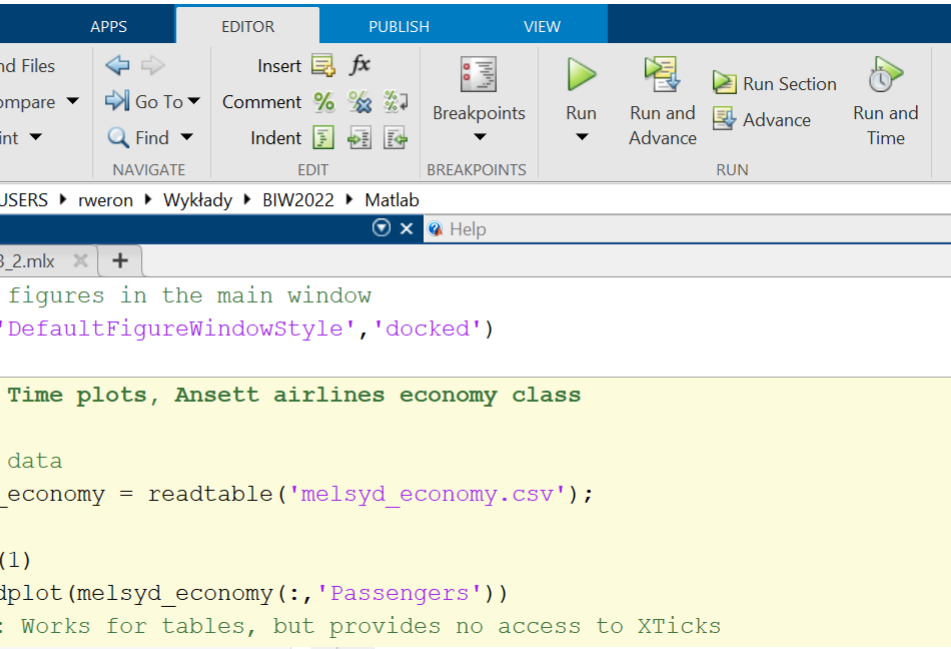


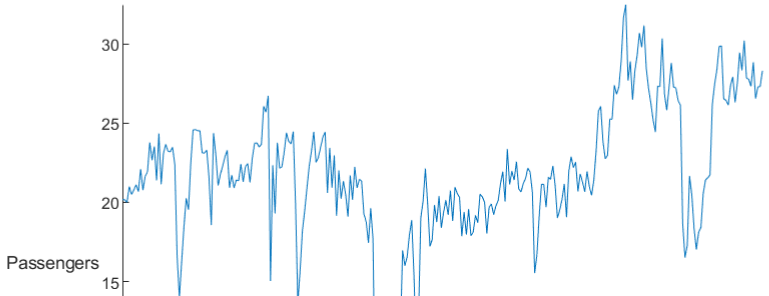
What about ?

"Regular" script

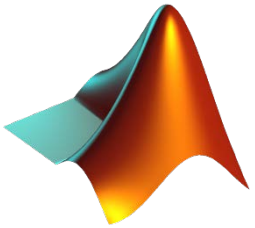
Live Script







(Simple) time plots



Live Editor - fpp3_2.mlx

fpp3_2.m | fpp3_2.mlx

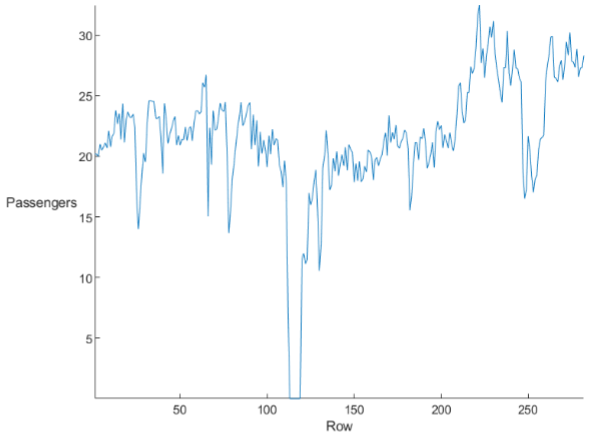
```
%cd('D:\USERS\rweron\Wykłady\BIW2022\Matlab')
```

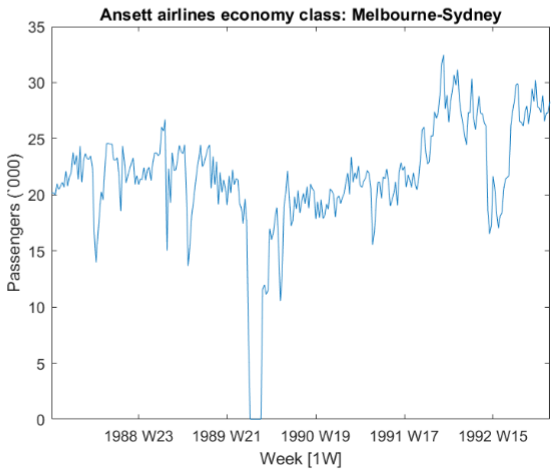
2.2 Time plots, Ansett airlines economy class

```
% 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';
```

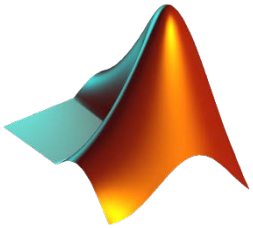
2.2 Time plots, Australian antidiabetic drug sales

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

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





(Simple) time plots cont.

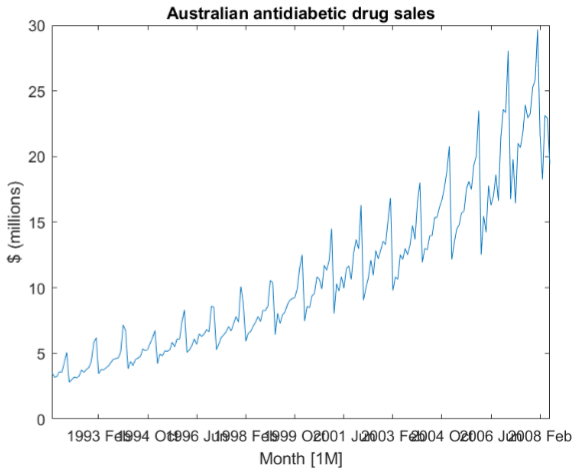


Live Editor - fpp3_2.mlx

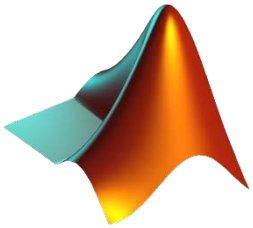
fpp3_2.m | fpp3_2.mlx

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;
h.XLim = [1 length(a10_Cost)];
h.XTickLabel = table2cell(a10(h.XTick,'Month'));
% Set axis labels and title
h.XLabel.String = 'Month [1M]';
h.YLabel.String = '$ (millions)';
h.Title.String = 'Australian antidiabetic drug sales';
```



Seasonal plots

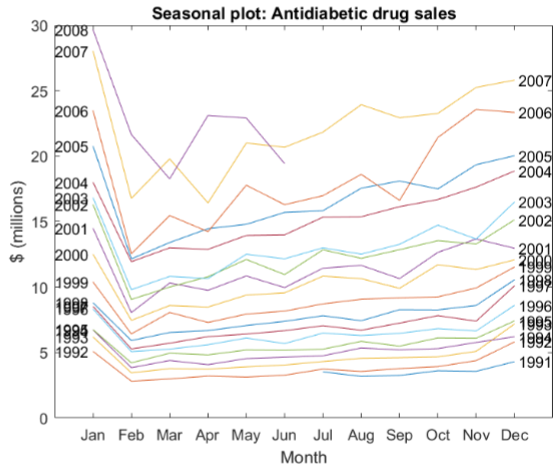


Live Editor - fpp3_2.mlx

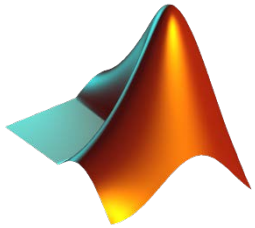
fpp3_2.m | fpp3_2.mlx

2.4 Seasonal plots, Australian antidiabetic drug sales

```
% Fill in missing months in 1991 and 2008
Month = {'1991 Jan';'1991 Feb';'1991 Mar';...
    '1991 Apr';'1991 May';'1991 Jun'};
TotalC = [NaN; NaN; NaN; NaN; NaN; NaN];
Cost = [NaN; NaN; NaN; NaN; NaN; NaN];
a10_ext = [table(Month,TotalC,Cost); a10(:,:)];

Month = {'2008 Jul';'2008 Aug';'2008 Sep';...
    '2008 Oct';'2008 Nov';'2008 Dec'};
TotalC = [NaN; NaN; NaN; NaN; NaN; NaN];
Cost = [NaN; NaN; NaN; NaN; NaN; NaN];
a10_ext = [a10_ext(:,:); table(Month,TotalC,Cost)];

Month_Labels = {'Jan','Feb','Mar','Apr','May','Jun',...
    'Jul','Aug','Sep','Oct','Nov','Dec'};

a10_ext_Cost = table2array(a10_ext(:,'Cost'));
a10_ext_Cost_r = reshape(a10_ext_Cost,12,length(a10_ext_Cost)/12);

figure
plot(a10_ext_Cost_r)
h = gca;
h.XLim = [0 13];
h.XTick = 1:12;
h.XTickLabel = Month_Labels;
Years = 1991:2008;
h.XLabel.String = 'Month';
h.YLabel.String = '$ (millions)';
h.Title.String = 'Seasonal plot: Antidiabetic drug sales';
for i=1:18
    text(12.1,a10_ext_Cost_r(12,i),num2str(Years(i)))
    text(.9,a10_ext_Cost_r(1,i),num2str(Years(i)),...
        'HorizontalAlignment','right')
end
```



Seasonal plots cont.



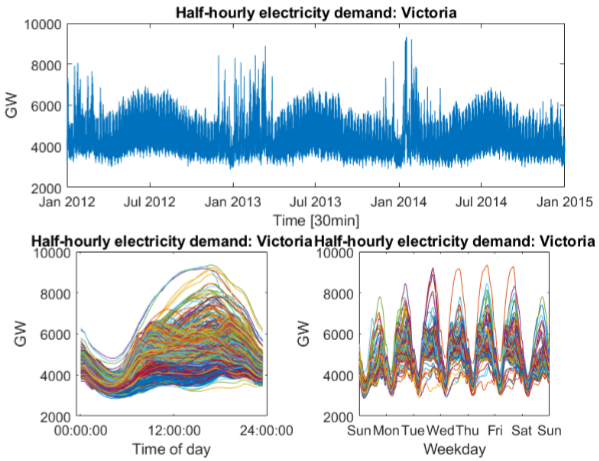
Live Editor - fpp3_2.mlx

Help

fpp3_2.m | fpp3_2.mlx

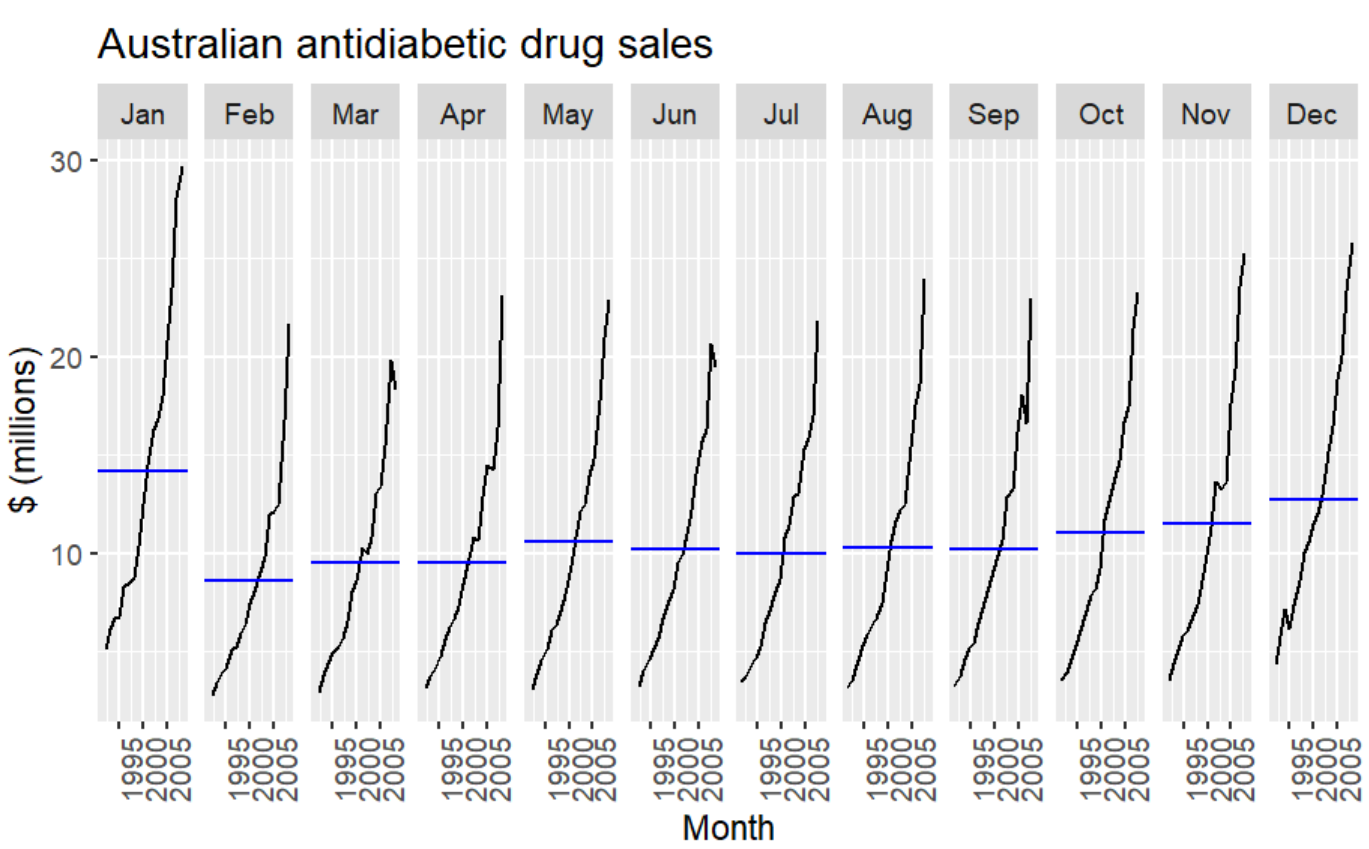
2.4 Seasonal plots, Half-hourly electricity demand

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

figure
% Convert columns to arrays
vic_elec_time = table2array(vic_elec(:,'Time'));
vic_elec_demand = table2array(vic_elec(:,'Demand'));

subplot(2,2,[1 2])
plot(vic_elec_time,vic_elec_demand)
xlabel('Time [30min]')
ylabel('GW')
title('Half-hourly electricity demand: Victoria')

subplot(2,2,3)
vic_day_demand = reshape(vic_elec_demand,48,[]);
plot(timeofday(vic_elec_time(1:48)),vic_day_demand')
xlabel('Time of day')
ylabel('GW')
title('Half-hourly electricity demand: Victoria')

subplot(2,2,4)
vic_elec_demand_ext = [vic_elec_demand; NaN*ones(48*3,1)];
vic_week_demand = reshape(vic_elec_demand_ext,48*7,[]);
plot(vic_elec_time(1:48*7),vic_week_demand')
datetick('x','ddd')
xlabel('Weekday')
ylabel('GW')
title('Half-hourly electricity demand: Victoria')
```



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

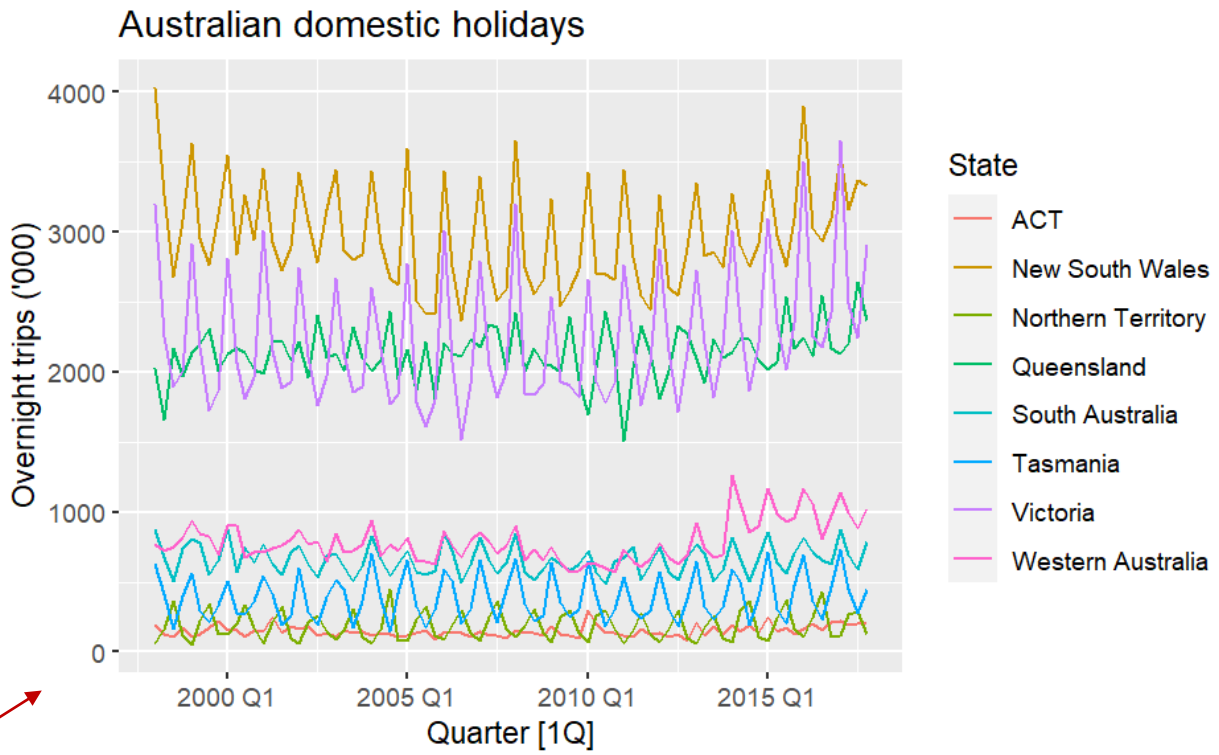
Australian Capital Territory (ACT)

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

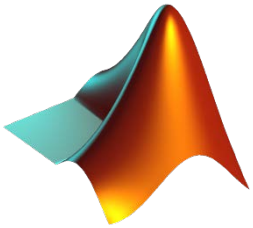
1-10 of 640 rows Previous 1 2 3 4 5 6 … 64 Next

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

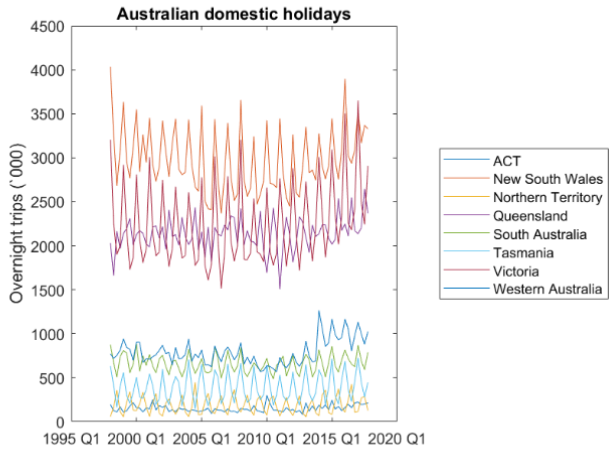


Live Editor - fpp3_2.mlx

fpp3_2.m | fpp3_2.mlx

2.5 Seasonal subseries plots, Australian domestic holidays

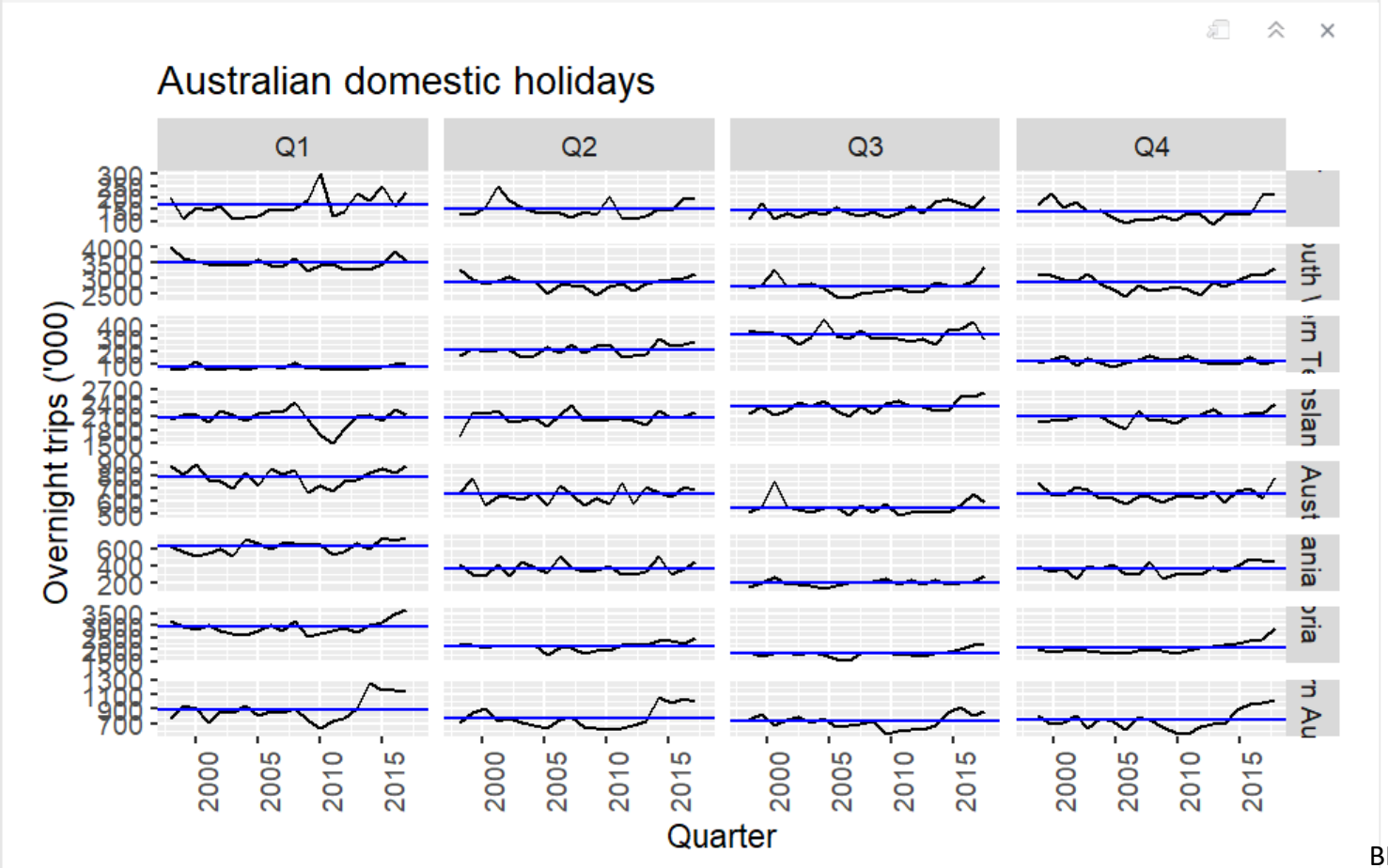
```
% Load data
holi = readtable('holidays.csv');

figure
% Convert table columns to arrays
Holi_time = table2array(holi(:,'Quarter'));
qtr = datetime(Holi_time,'InputFormat','yyyy QQQ');
Trips = table2array(holi(:,'Trips'));
State = table2array(holi(:,'State'));

Trips_res = reshape(Trips,80,[]);
plot(qtr(1:80),Trips_res)
datetick('x','yyyy QQ')
legend(State{1:80:end},'Location','eastoutside')
ylabel('Overnight trips (`000)')
title('Australian domestic holidays')
```



gg_subseries cont.

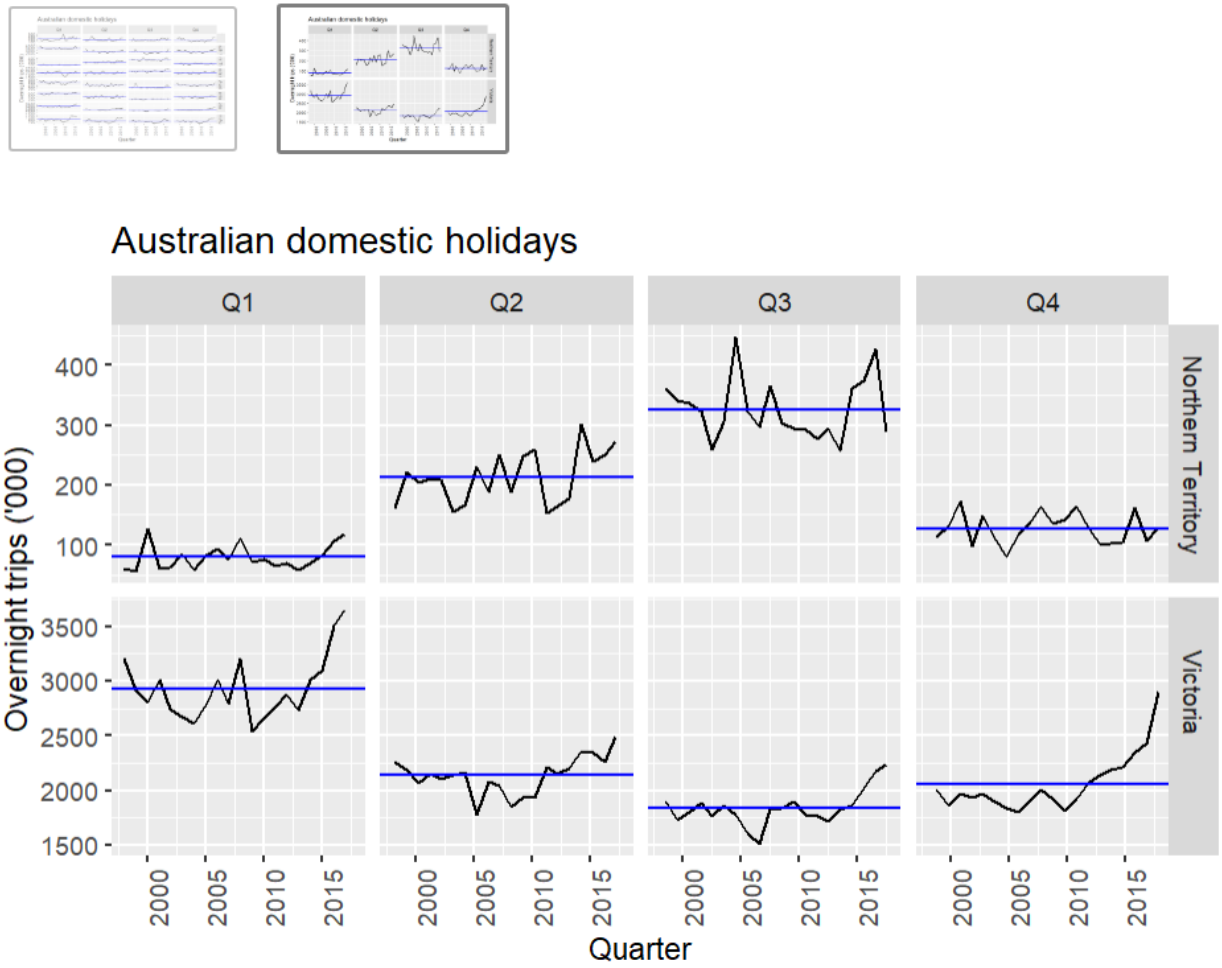
```r
holidays %>%
  gg_subseries(Trips) +
  labs(y = "Overnight trips ('000)",
       title = "Australian domestic holidays")
```

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

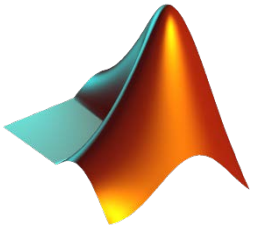


gg_subseries cont.

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



Seasonal subseries plots cont.

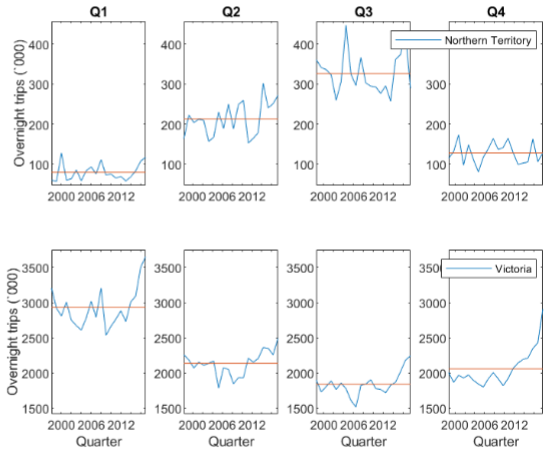


Live Editor - fpp3_2.mlx

Help

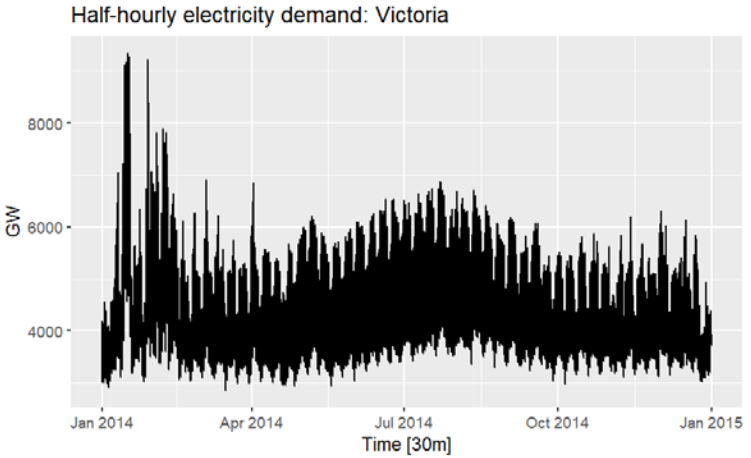
fpp3_2.m | fpp3_2.mlx

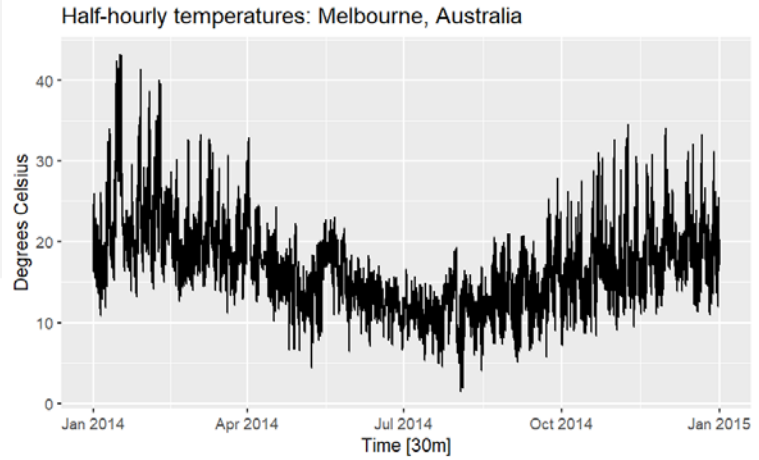
```
117  figure
118  NT_range = [min(Trips_res(:,3))-10 max(Trips_res(:,3))+10];
119  Vic_range = [min(Trips_res(:,7))-100 max(Trips_res(:,7))+100];
120  for i=1:4
121     % 'Northern Territory'
122     subplot(2,4,i)
123     plot(qtr(i:4:80),Trips_res(i:4:end,3),...
124         qtr(i:4:80),mean(Trips_res(i:4:end,3))*ones(size(qtr(i:4:80))))
125     ylim(NT_range)
126     title(['Q' num2str(i)])
127     % 'Victoria'
128     subplot(2,4,4+i)
129     plot(qtr(i:4:80),Trips_res(i:4:end,7),...
130         qtr(i:4:80),mean(Trips_res(i:4:end,7))*ones(size(qtr(i:4:80))))
131     ylim(Vic_range)
132     xlabel('Quarter')
133  end
134  subplot(2,4,1)
135  ylabel('Overnight trips (`000)')
136  subplot(2,4,5)
137  ylabel('Overnight trips (`000)')
138  subplot(2,4,4)
139  legend('Northern Territory')
140  subplot(2,4,8)
141  legend('Victoria')
142
```

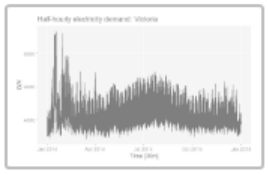


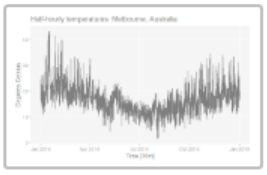
Scatterplots: ggplot

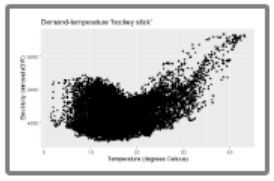
```
vic_elec %>%
  filter(year(Time) == 2014) %>%
  autoplot(Demand) +
  labs(y = "GW",
       title = "Half-hourly electricity demand: Victoria")

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'")
```



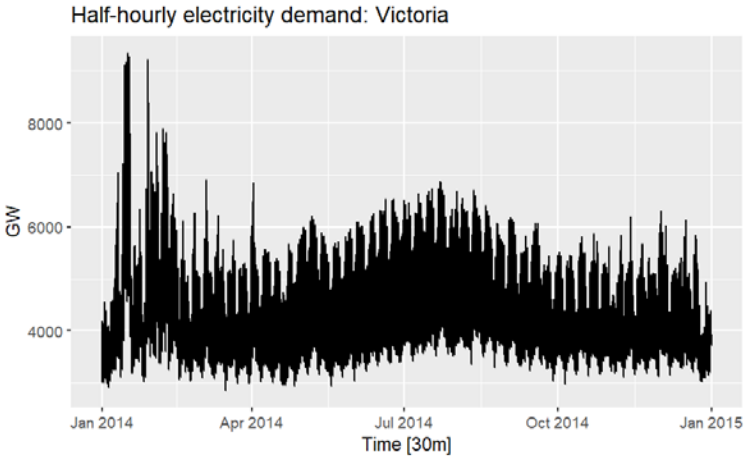


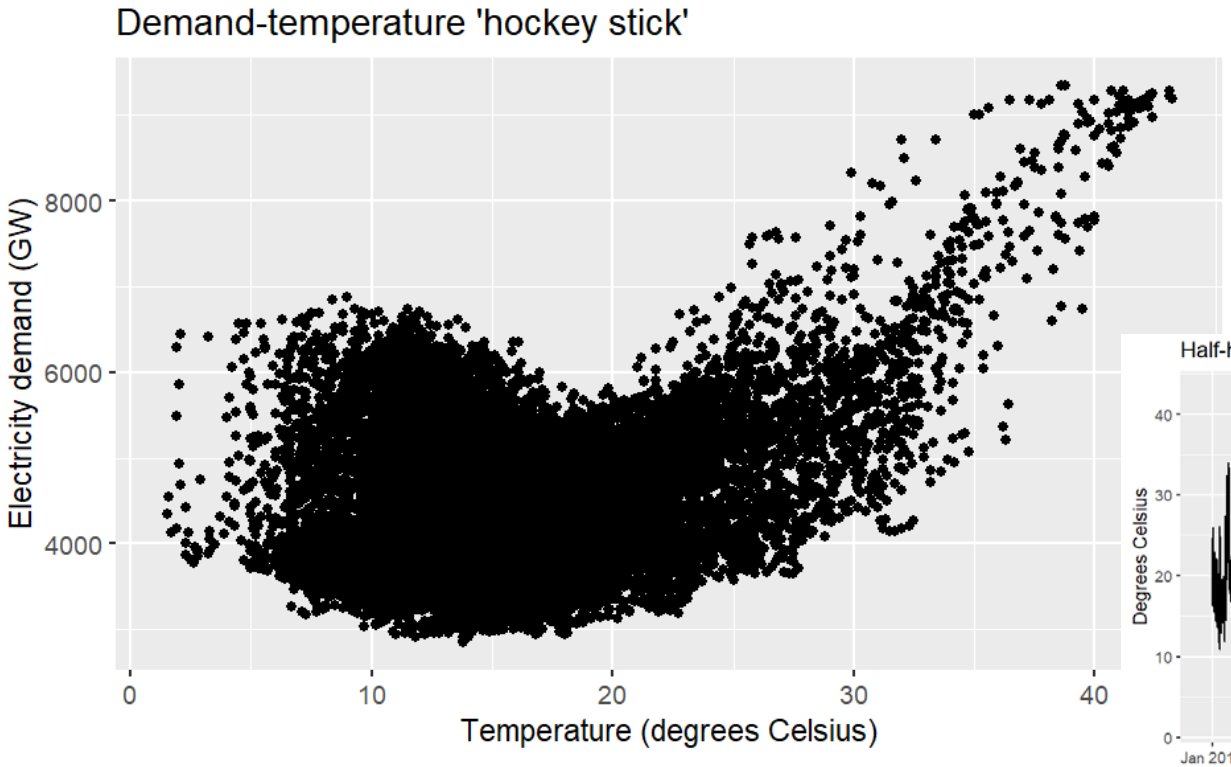


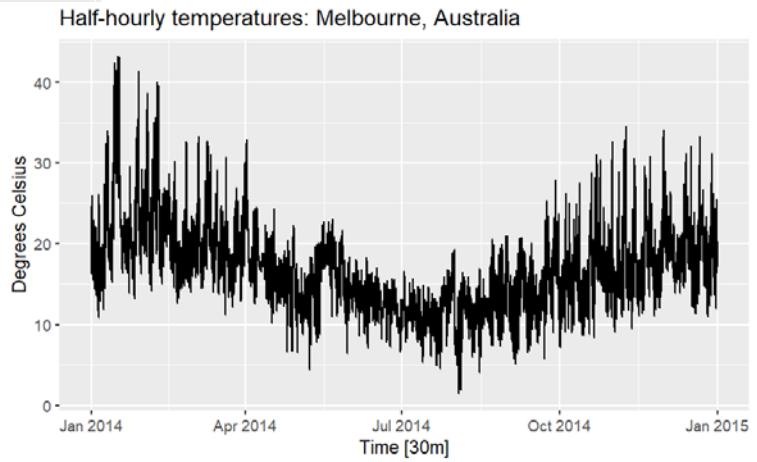




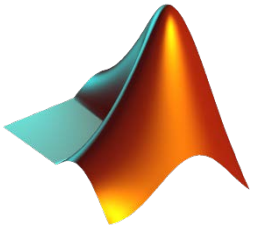
Scatterplots: ggplot







Scatterplots

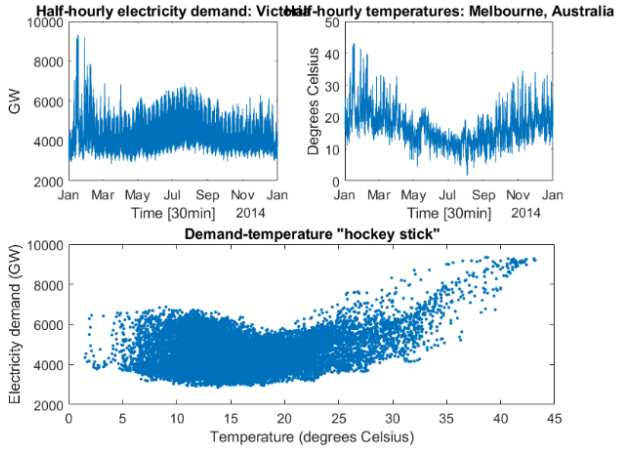


Live Editor - fpp3_2.mlx

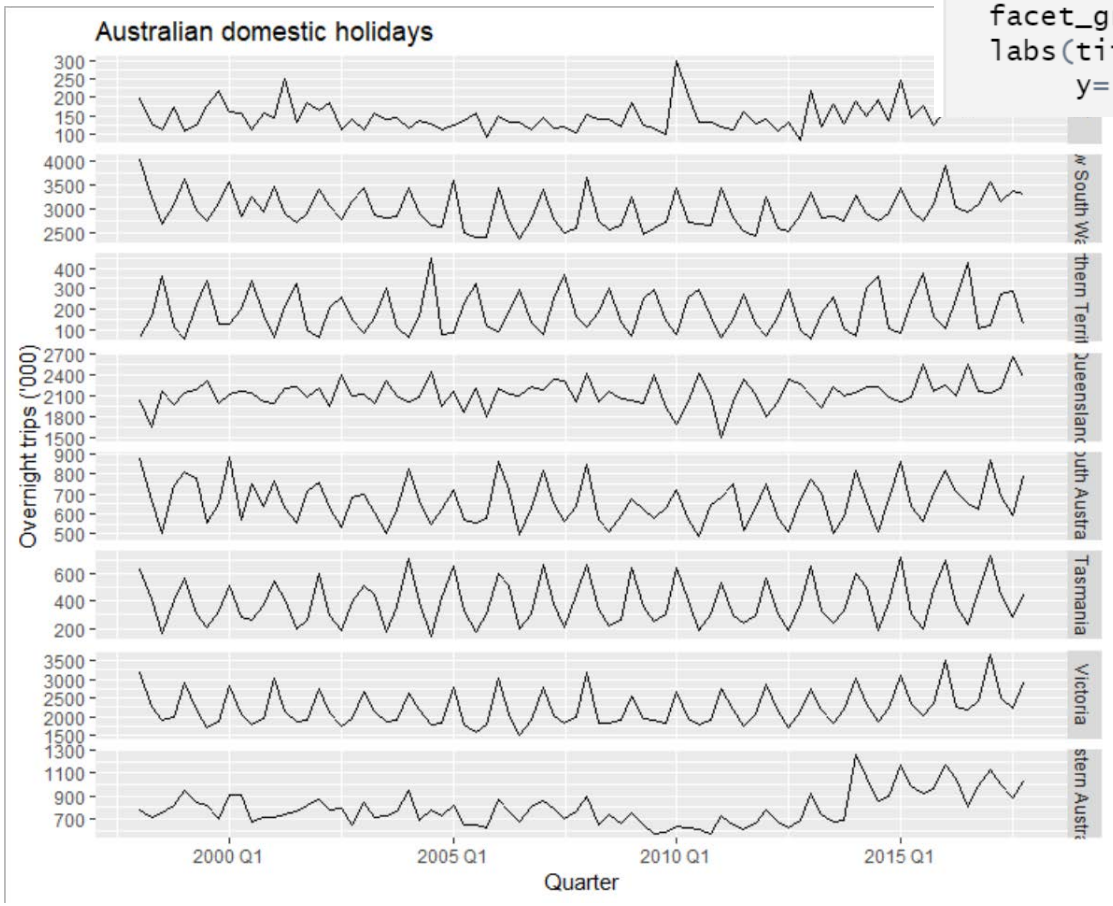
fpp3_2.m | fpp3_2.mlx

2.6 Scatterplots, Half-hourly electricity demand vs temperature

```
143
144 % Load data
145 % vic_elec = readtable('vic_elec.csv');
146
147 figure
148 % Convert Time column to array and select year 2014
149 vic_2014_time = table2array(vic_elec(:,'Time'));
150 i = find(year(vic_2014_time) == 2014);
151 vic_2014_time = vic_2014_time(i);
152
153 % Convert table columns to arrays
154 vic_2014_demand = table2array(vic_elec(i,'Demand'));
155 vic_2014_temp = table2array(vic_elec(i,'Temperature'));
156 subplot(2,2,1)
157 plot(vic_2014_time,vic_2014_demand)
158 xlabel('Time [30min]')
159 ylabel('GW')
160 title('Half-hourly electricity demand: Victoria')
161
162 subplot(2,2,2)
163 plot(vic_2014_time,vic_2014_temp)
164 xlabel('Time [30min]')
165 ylabel('Degrees Celsius')
166 title('Half-hourly temperatures: Melbourne, Australia')
167
168 subplot(2,2,[3 4])
169 plot(vic_2014_temp,vic_2014_demand,'.')
170 xlabel('Temperature (degrees Celsius)')
171 ylabel('Electricity demand (GW)')
172 title('Demand-temperature "hockey stick"')
```



Scatterplot matrices: ggplot

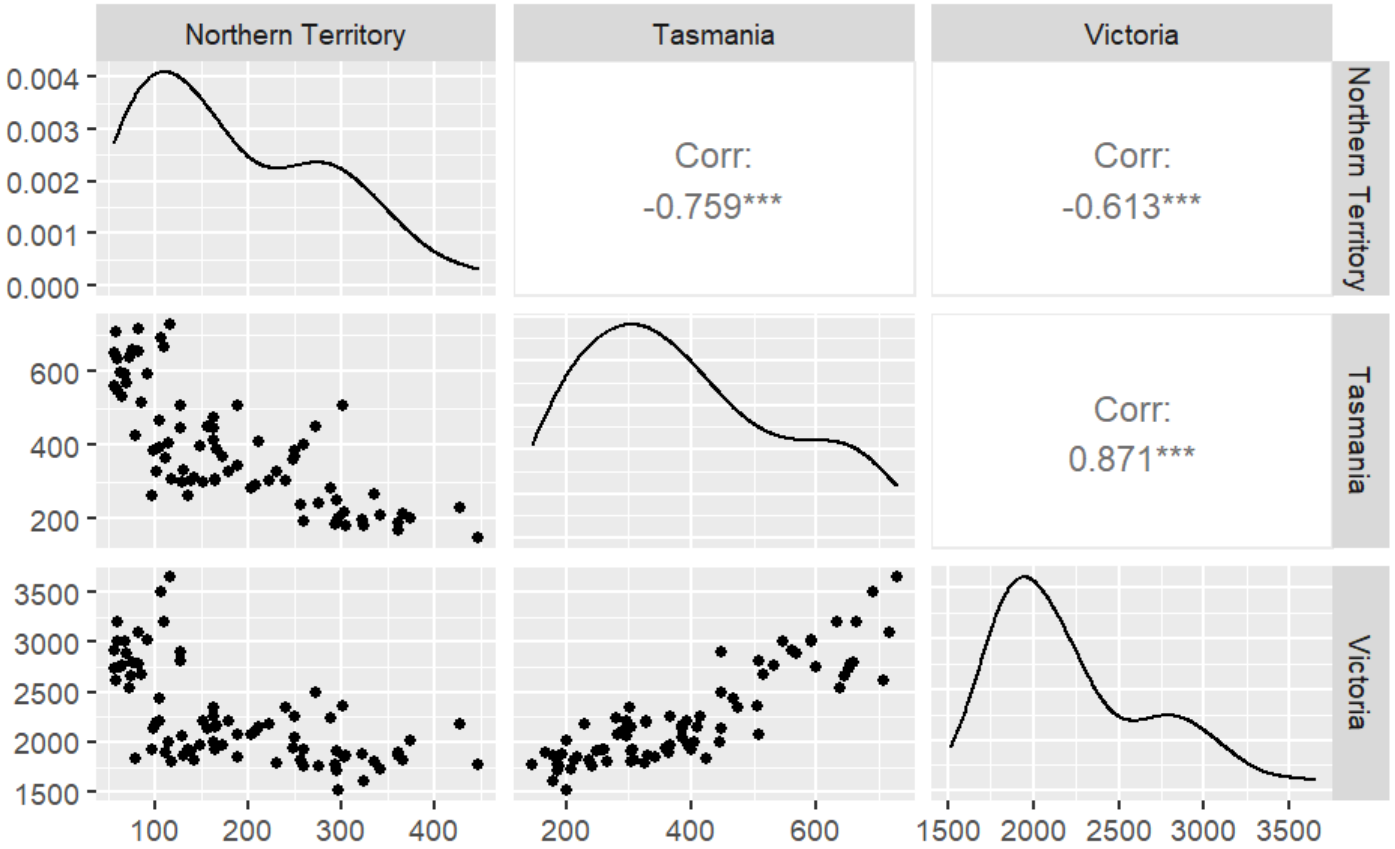



```
holidays %>%
  ggplot(aes(x = Quarter, y = Trips)) +
  geom_line() +
  facet_grid(vars(State), scales = "free_y") +
  labs(title = "Australian domestic holidays",
       y= "Overnight trips ('000)")
```

Load this library

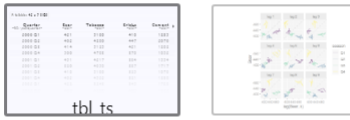
Scatterplot matrices: GGally::ggpairs

```
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

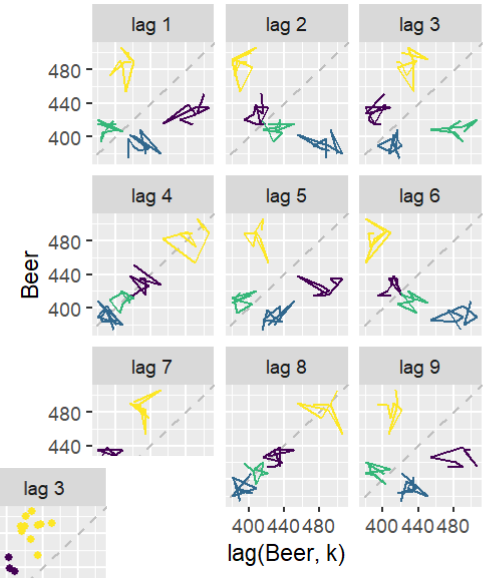
```
recent_production <- aus_production %>%
  filter(year(Quarter) >= 2000)

recent_production

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

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

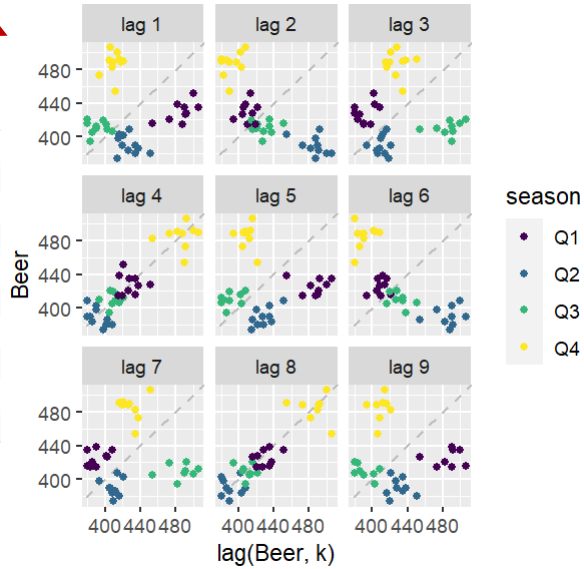


A tsibble: 42 x 7 [1Q]

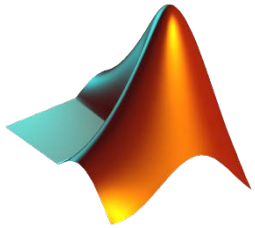
Quarter <S3: yearquart>	Beer <dbl>	Tobacco <dbl>	Bricks <dbl>	Cement <dbl>	Electricity <dbl>
2000 Q1	421	5169	416	1835	49320
2000 Q2	402	4860	447	2070	50670
2000 Q3	414	5185	421	1898	52623
2000 Q4	500	4763	379	1652	49350
2001 Q1	451	4217	304	1554	51658
2001 Q2	380	4959	337	1717	51103
2001 Q3	416	5196	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

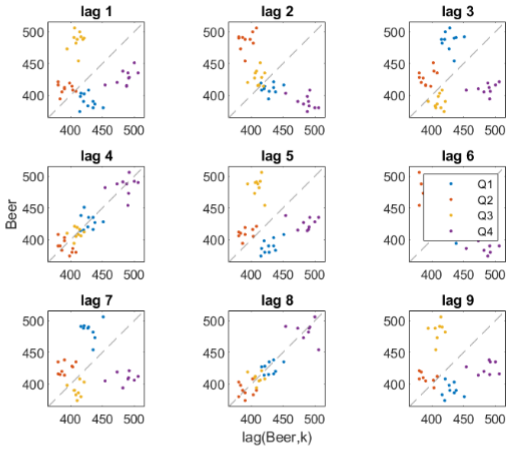


Live Editor - fpp3_2.mlx

fpp3_2.m | fpp3_2.mlx

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

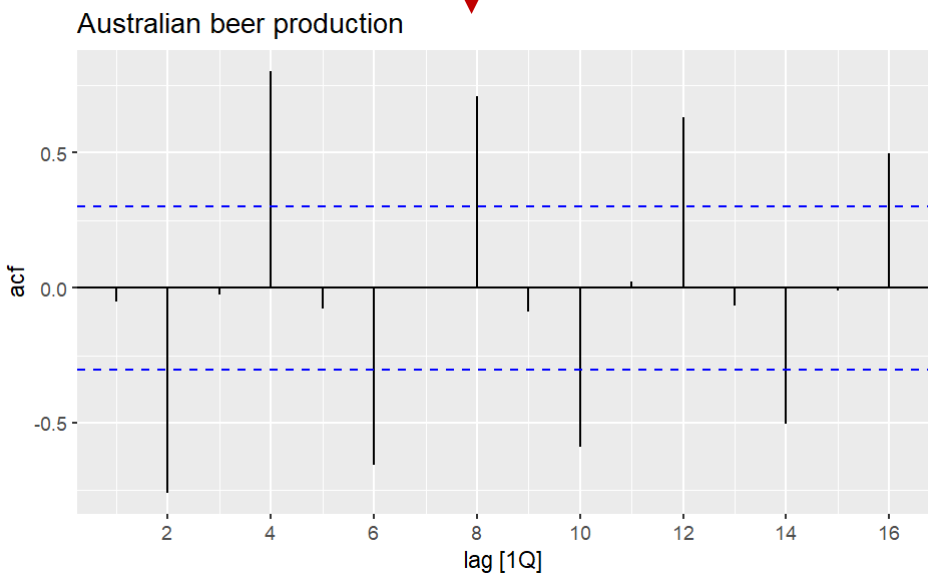
```
% Load data
recent_production = readtable('recent_production.csv');

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)')
```



Autocorrelation Function (ACF)

```
recent_production %>% ACF(Beer)

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

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



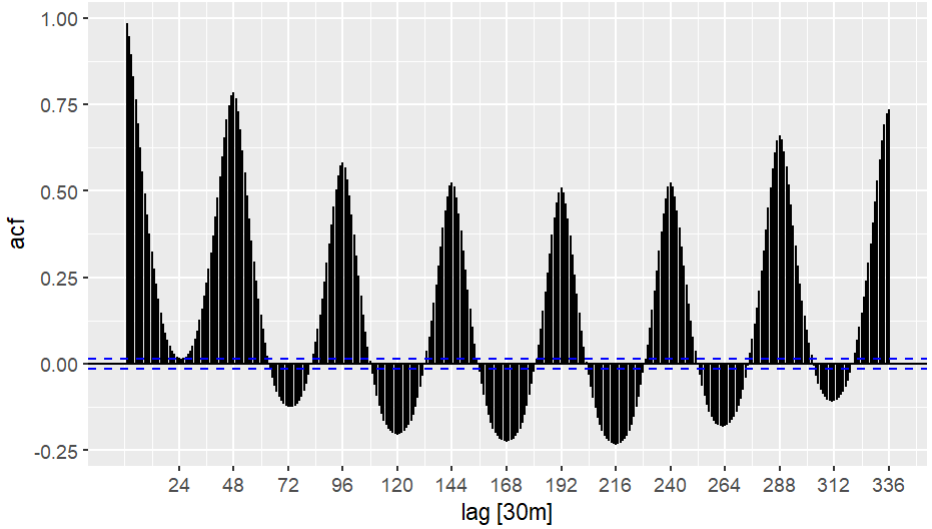
A tsibble: 16 x 2 [1Q]

lag <S3: cf_lag>	acf <dbl>
1Q	-0.052981076
2Q	-0.758175440
3Q	-0.026233757
4Q	0.802204530
5Q	-0.077471204
6Q	-0.657451271
7Q	0.001194922
8Q	0.707254078
9Q	-0.088756255
10Q	-0.587629908

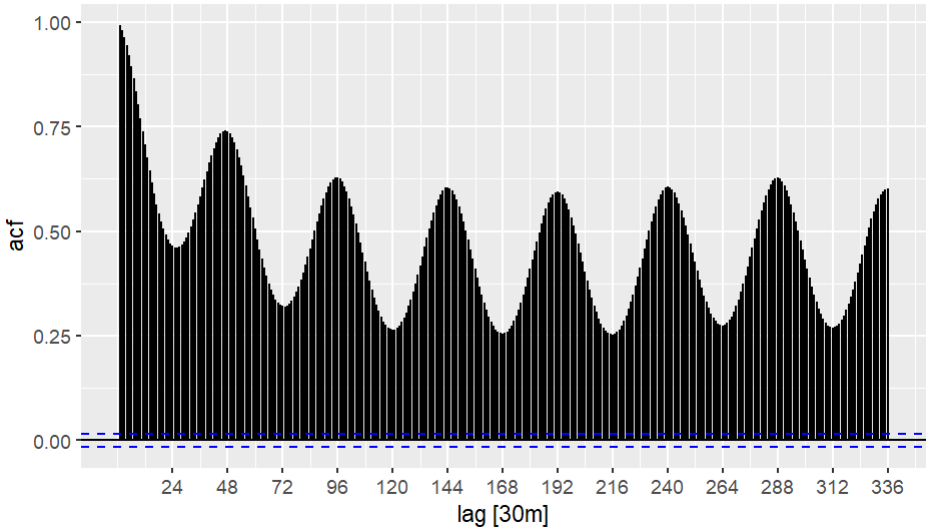
ACF cont.

```r
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


Live Editor - fpp3_2.mlx
Help
fpp3_2.m
fpp3_2.mlx
2.8 Autocorrelation, Half-hourly electricity demand & temperature
206 figure
207 subplot(2,1,1)
208 autocorr(vic_2014_demand,336)
209 xlim([0 336])
210 xticks(0:48:336)
211 xlabel('Lag [30min]')
212 title('Half-hourly electricity demand: Victoria')
213
214 subplot(2,1,2)
215 autocorr(vic_2014_temp,336)
216 xlim([0 336])
217 xticks(0:48:336)
218 xlabel('Lag [30min]')
219 title('Half-hourly temperatures: Melbourne, Australia')
220
Half-hourly electricity demand: Victoria
Sample Autocorrelation
Lag [30min]
Half-hourly temperatures: Melbourne, Australia
Sample Autocorrelation
Lag [30min]

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