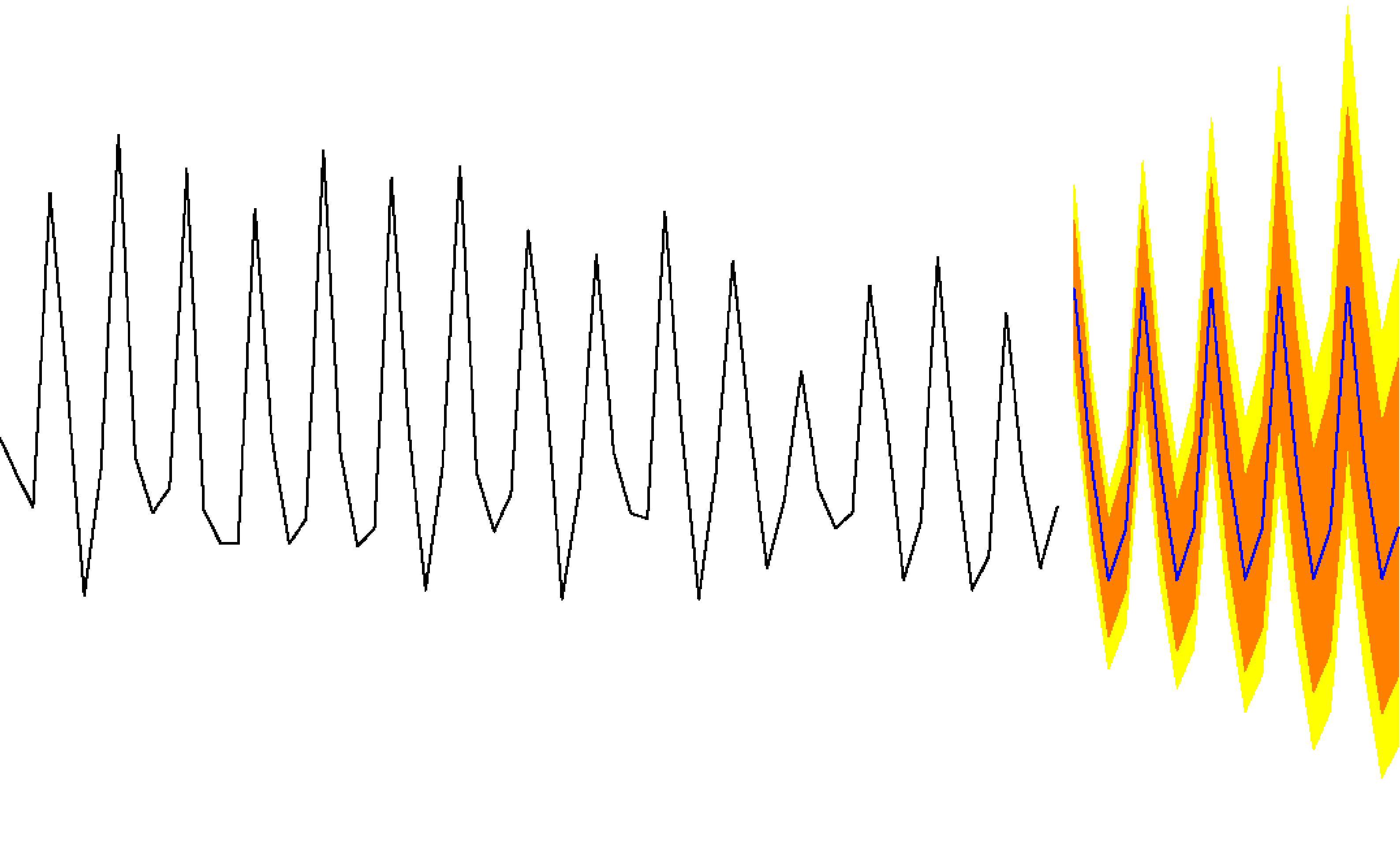
Forecasting: Principles & Practice



Leader: Rob J Hyndman

23-25 September 2014

University of Western Australia

robjhyndman.com/uwa

Contents

1 Introduction to forecasting

1.1 Introduction

Brief bio

- Director of Monash University’s Business & Economic Forecasting Unit
- Editor-in-Chief, *International Journal of Forecasting*

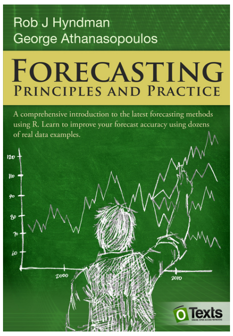
How my forecasting methodology is used:

- Pharmaceutical Benefits Scheme
- Cancer incidence and mortality
- Electricity demand
- Ageing population
- Fertilizer sales

Poll: How experienced are you in forecasting?

1. Guru: I wrote the book, done it for decades, now I do the conference circuit.
2. Expert: It has been my full time job for more than a decade.
3. Skilled: I have been doing it for years.
4. Comfortable: I understand it and have done it.
5. Learner: I am still learning.
6. Beginner: I have heard of it and would like to learn more.
7. Unknown: What is forecasting? Is that what the weather people do?

Key reference


Rob J Hyndman
George Athanasopoulos
FORECASTING
PRINCIPLES AND PRACTICE
A comprehensive introduction to the latest forecasting methods using R. Learn to improve your forecast accuracy using dozens of real data examples.
OTexts

Hyndman, R. J. & Athanasopoulos, G. (2013)
Forecasting: principles and practice.
OTexts.org/fpp/

- Free and online
- Data sets in associated R package
- R code for examples

Poll: How proficient are you in using R?

1. Guru: The R core team come to me for advice.
2. Expert: I have written several packages on CRAN.
3. Skilled: I use it regularly and it is an important part of my job.
4. Comfortable: I use it often and am comfortable with the tool.
5. User: I use it sometimes, but I am often searching around for the right function.
6. Learner: I have used it a few times.
7. Beginner: I've managed to download and install it.
8. Unknown: Why are you speaking like a pirate?

Install required packages

```
install.packages("fpp", dependencies=TRUE)
```

Getting help with R

```
# Search for terms
help.search("forecasting")

# Detailed help
help(forecast)

# Worked examples
example("forecast.ar")

# Similar names
apropos("forecast")

#Help on package
help(package="fpp")
```

Approximate outline

Day	Topic	Chapter
1	The forecaster's toolbox	1,2
1	Seasonality and trends	6
1	Exponential smoothing	7
2	Time series decomposition	6
2	Time series cross-validation	2
2	Transformations	2
2	Stationarity and differencing	8
2	ARIMA models	8
3	State space models	–
3	Dynamic regression	9
3	Hierarchical forecasting	9
3	Advanced methods	9

Assumptions

- This is not an introduction to R. I assume you are broadly comfortable with R code and the R environment.
- This is not a statistics course. I assume you are familiar with concepts such as the mean, standard deviation, quantiles, regression, normal distribution, etc.
- This is not a theory course. I am not going to derive anything. I will teach you forecasting tools, when to use them and how to use them most effectively.

1.2 Some case studies

CASE STUDY 1: Paperware company

Problem: Want forecasts of each of hundreds of items. Series can be stationary, trended or seasonal. They currently have a large forecasting program written in-house but it doesn't seem to produce sensible forecasts. They want me to tell them what is wrong and fix it.

Additional information

- Program written in COBOL making numerical calculations limited. It is not possible to do any optimisation.
- Their programmer has little experience in numerical computing.
- They employ no statisticians and want the program to produce forecasts automatically.

CASE STUDY 1: Paperware company

Methods currently used

A 12 month average
C 6 month average
E straight line regression over last 12 months
G straight line regression over last 6 months
H average slope between last year's and this year's values.
(Equivalent to differencing at lag 12 and taking mean.)
I Same as H except over 6 months.
K I couldn't understand the explanation.

CASE STUDY 2: PBS

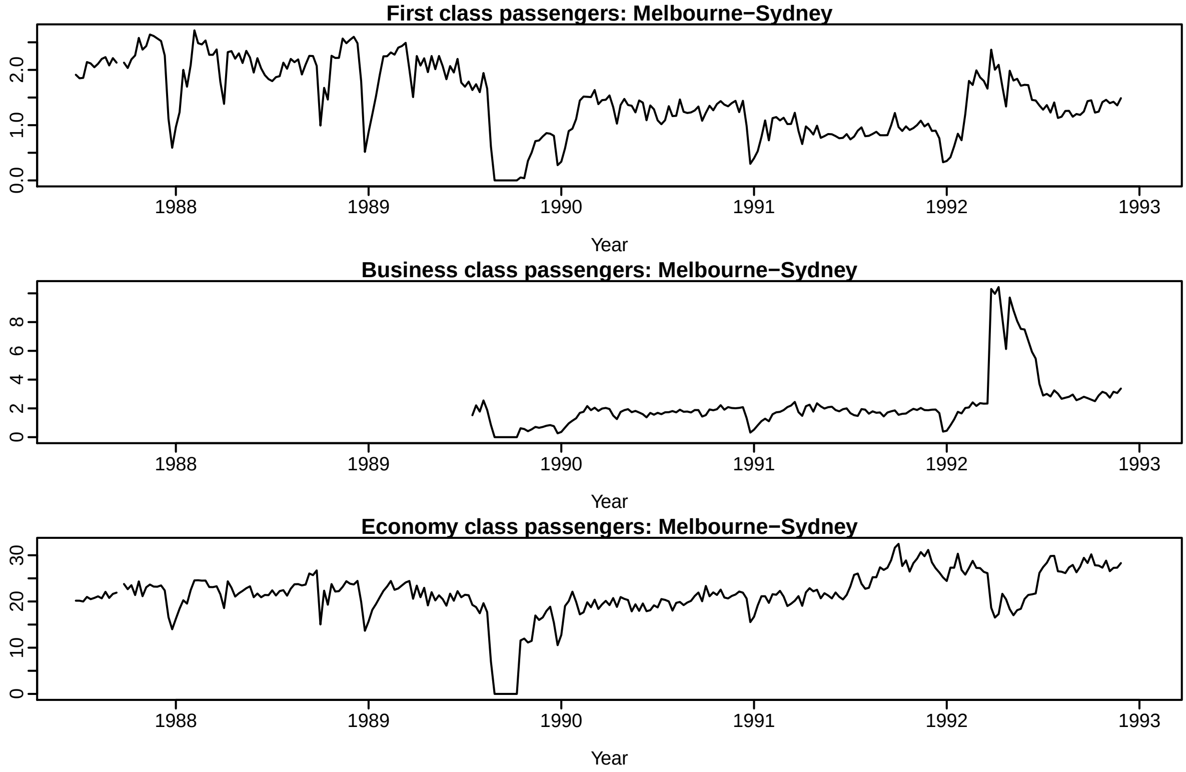
The **Pharmaceutical Benefits Scheme** (PBS) is the Australian government drugs subsidy scheme.

- Many drugs bought from pharmacies are subsidised to allow more equitable access to modern drugs.
- The cost to government is determined by the number and types of drugs purchased. Currently nearly 1% of GDP.
- The total cost is budgeted based on forecasts of drug usage.
- In 2001: $4.5 billion budget, under-forecasted by $800 million.
- Thousands of products. Seasonal demand.
- Subject to covert marketing, volatile products, uncontrollable expenditure.

- Although monthly data available for 10 years, data are aggregated to annual values, and only the first three years are used in estimating the forecasts.
- All forecasts being done with the `FORECAST` function in MS-Excel!

Problem: How to do the forecasting better?

CASE STUDY 3: Airline



Problem: how to forecast passenger traffic on major routes.

Additional information

- They can provide a large amount of data on previous routes.
- Traffic is affected by school holidays, special events such as the Grand Prix, advertising campaigns, competition behaviour, etc.
- They have a highly capable team of people who are able to do most of the computing.

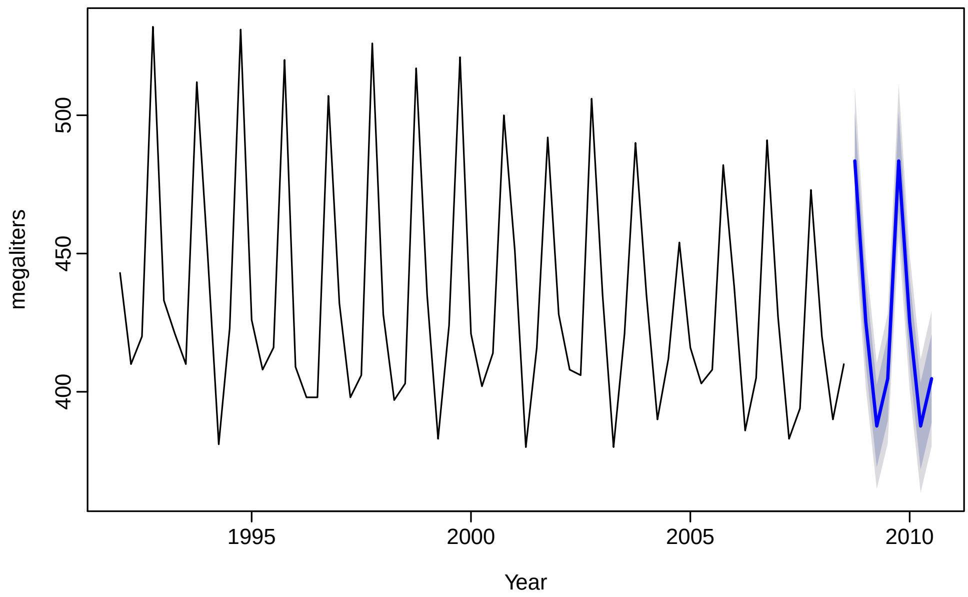
1.3 Time series data

Time series consist of sequences of observations collected over time. We will assume the time periods are equally spaced.

- Daily IBM stock prices
- Monthly rainfall
- Annual Google profits
- Quarterly Australian beer production

Forecasting is estimating how the sequence of observations will continue into the future.

Australian beer production



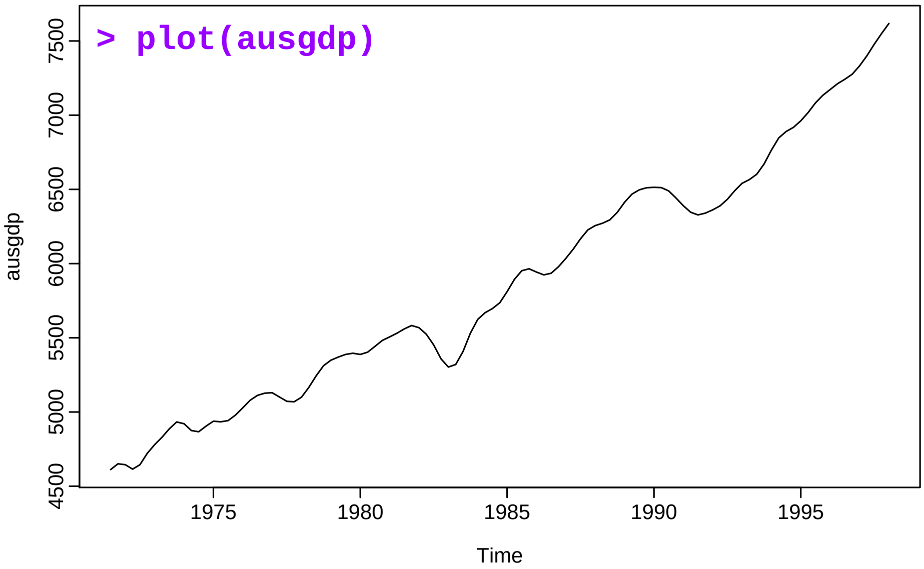
Australian GDP

```
ausgdp <- ts(scan("gdp.dat"),frequency=4, start=1971+2/4)
```

- Class: "ts"
- Print and plotting methods available.

```
> ausgdp
     Qtr1 Qtr2 Qtr3 Qtr4
1971           4612 4651
1972 4645 4615 4645 4722
1973 4780 4830 4887 4933
1974 4921 4875 4867 4905
1975 4938 4934 4942 4979
1976 5028 5079 5112 5127
1977 5130 5101 5072 5069
1978 5100 5166 5244 5312
1979 5349 5370 5388 5396
1980 5388 5403 5442 5482
```

Residential electricity sales



```
> plot(ausgdp)
```

```
> elecsales
Time Series:
Start = 1989
End = 2008
Frequency = 1
 [1] 2354.34 2379.71 2318.52 2468.99 2386.09 2569.47
 [7] 2575.72 2762.72 2844.50 3000.70 3108.10 3357.50
[13] 3075.70 3180.60 3221.60 3176.20 3430.60 3527.48
[19] 3637.89 3655.00
```

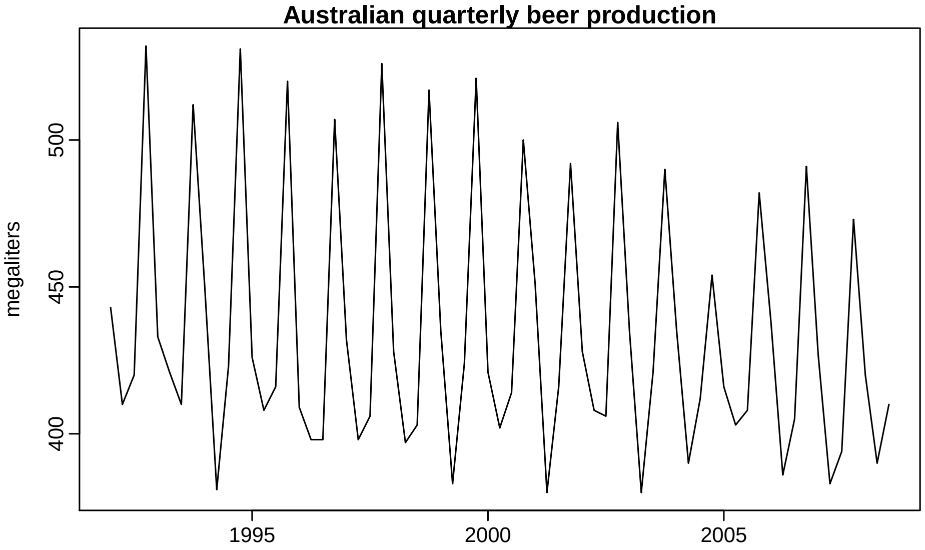
Main package used in this course

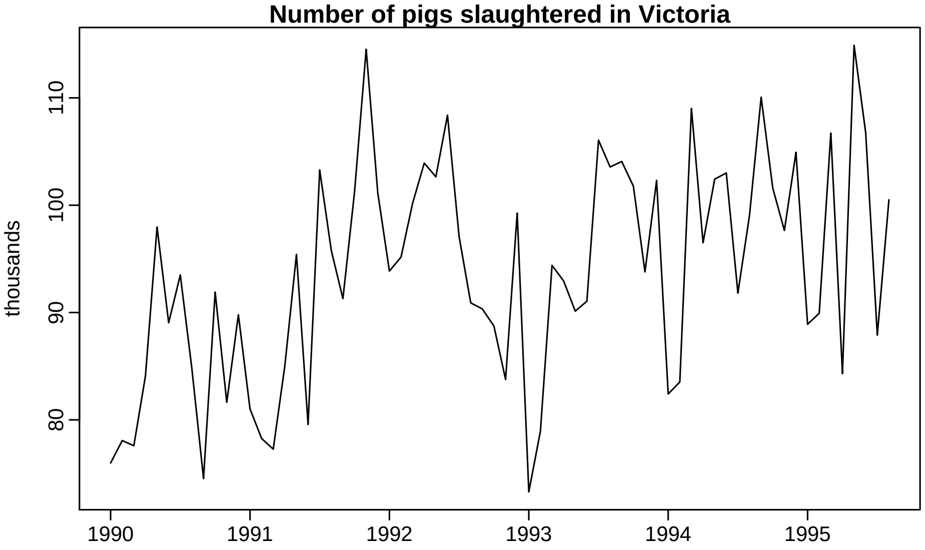
```
> library(fpp)
```

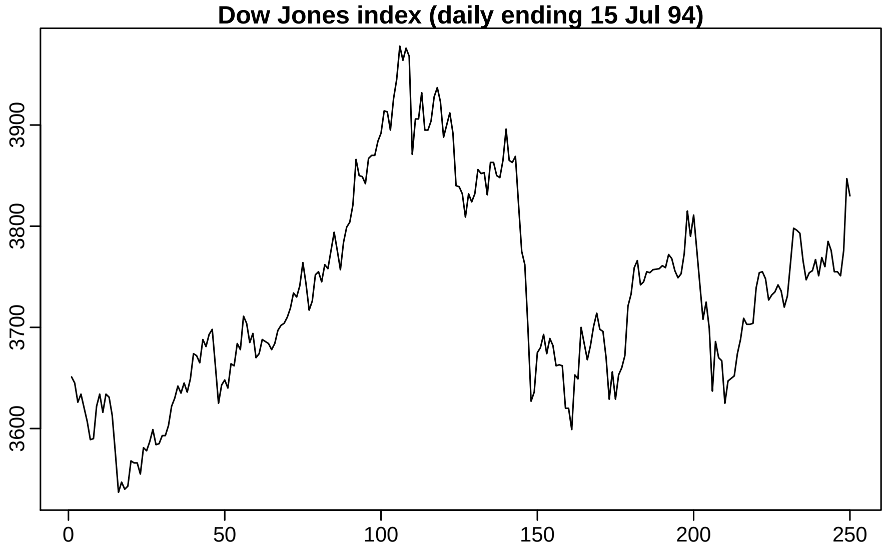
This loads:

- some data for use in examples and exercises
- **forecast** package (for forecasting functions)
- **tseries** package (for a few time series functions)
- **fma** package (for lots of time series data)
- **expsmooth** package (for more time series data)
- **lmtest** package (for some regression functions)

1.4 Some simple forecasting methods







Average method

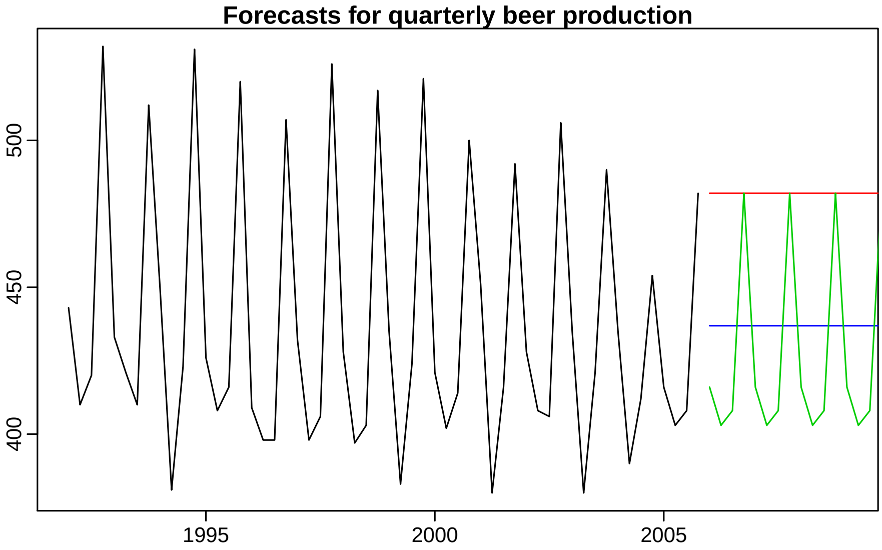
- Forecast of all future values is equal to mean of historical data $\{y_1, \dots, y_T\}$.
- Forecasts: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$

Naïve method (for time series only)

- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{T+h|T} = y_T$.
- Consequence of efficient market hypothesis.

Seasonal naïve method

- Forecasts equal to last value from same season.
- Forecasts: $\hat{y}_{T+h|T} = y_{T+h-km}$ where m = seasonal period and $k = \lfloor (h-1)/m \rfloor + 1$.



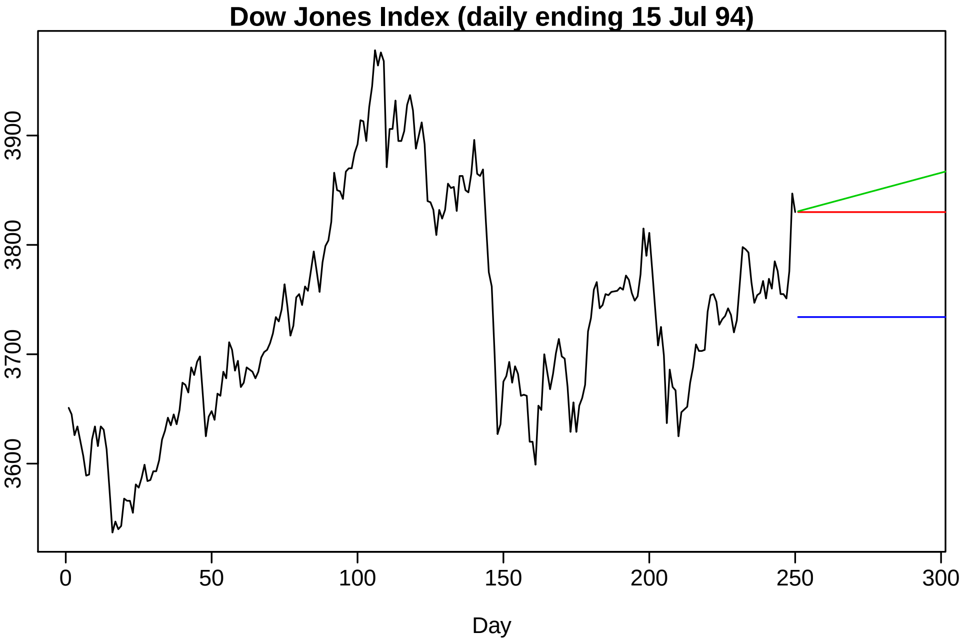
Drift method

- Forecasts equal to last value plus average change.
- Forecasts:

$$\begin{aligned}\hat{y}_{T+h|T} &= y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})\\ &= y_T + \frac{h}{T-1}(y_T - y_1).\end{aligned}$$

- Equivalent to extrapolating a line drawn between first and last observations.

- Mean: `meanf(x, h=20)`
- Naive: `naive(x, h=20)` or `rwf(x, h=20)`
- Seasonal naive: `snaive(x, h=20)`
- Drift: `rwf(x, drift=TRUE, h=20)`



1.5 Lab Session 1

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Use the Dow Jones index (data set `dowjones`) to do the following:
   (a) Produce a time plot of the series.
   (b) Produce forecasts using the drift method and plot them.
   (c) Show that the graphed forecasts are identical to extending the line drawn between the first and last observations.
   (d) Try some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

2. For each of the following series, make a graph of the data with forecasts using the most appropriate of the four benchmark methods: mean, naive, seasonal naive or drift.

   (a) Annual bituminous coal production (1920–1968). Data set `bicoal`.

   (b) Price of chicken (1924–1993). Data set `chicken`.

   (c) Monthly total of people on unemployed benefits in Australia (January 1956–July 1992). Data set `dole`.

   (d) Monthly total of accidental deaths in the United States (January 1973–December 1978). Data set `usdeaths`.

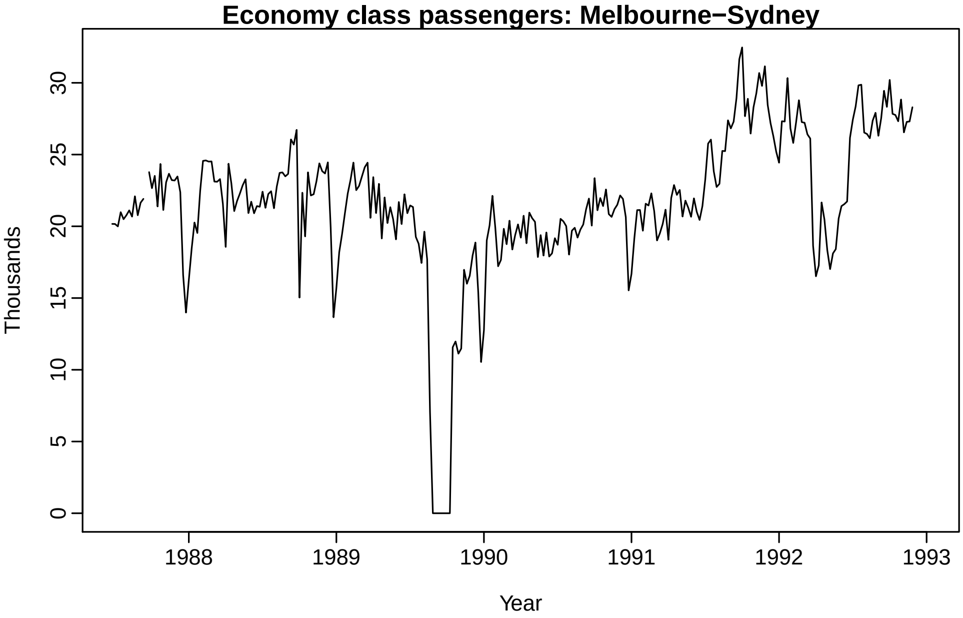
   (e) Quarterly production of bricks (in millions of units) at Portland, Australia (March 1956–September 1994). Data set `bricksq`.

   (f) Annual Canadian lynx trappings (1821–1934). Data set `lynx`.

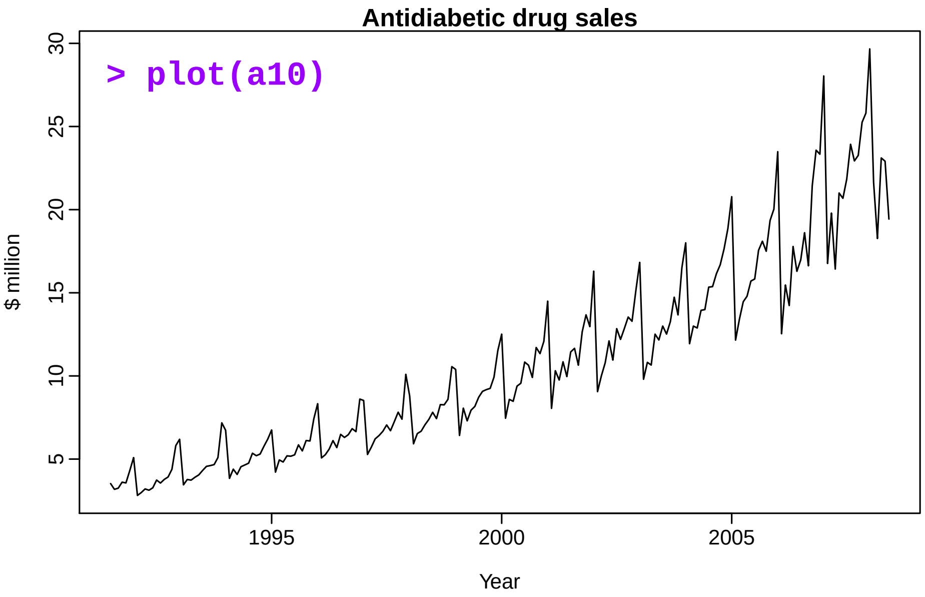
   In each case, do you think the forecasts are reasonable? If not, how could they be improved?

2 The forecaster's toolbox

2.1 Time series graphics

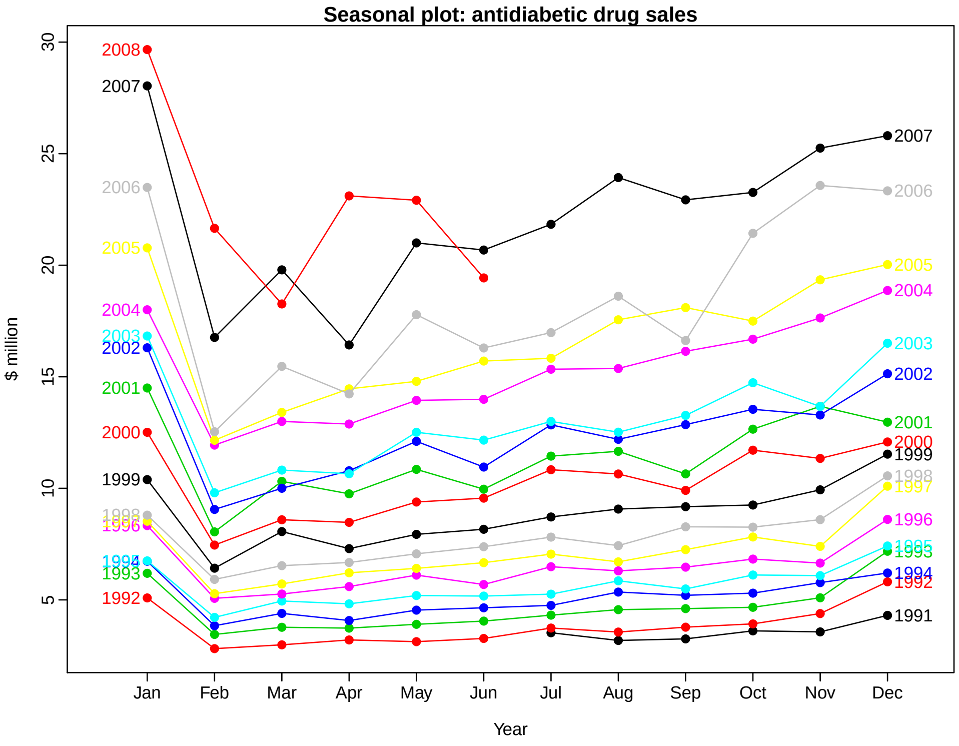


```
plot(melsyd[,"Economy.Class"])
```



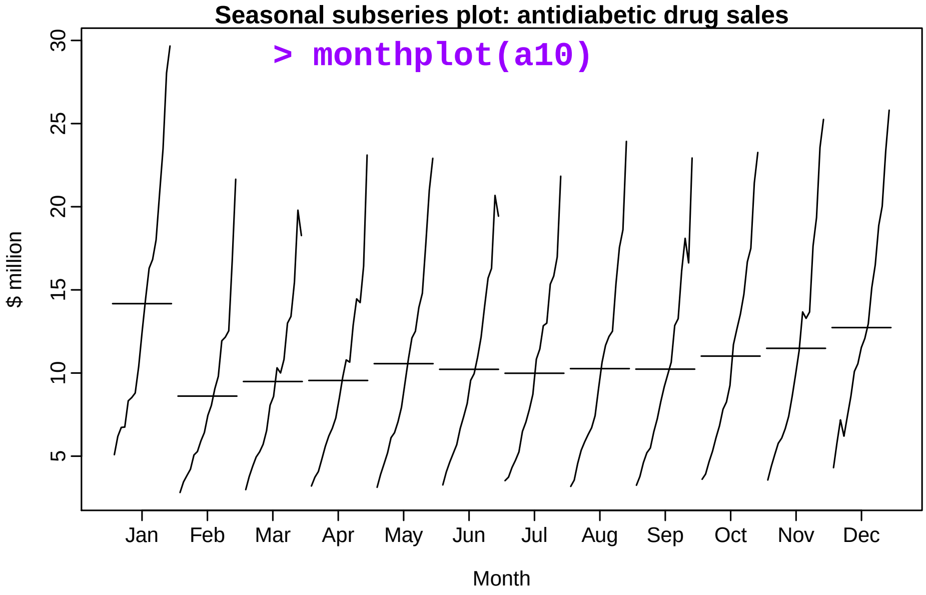
```
> plot(a10)
```

Seasonal plots



- 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: `seasonplot`

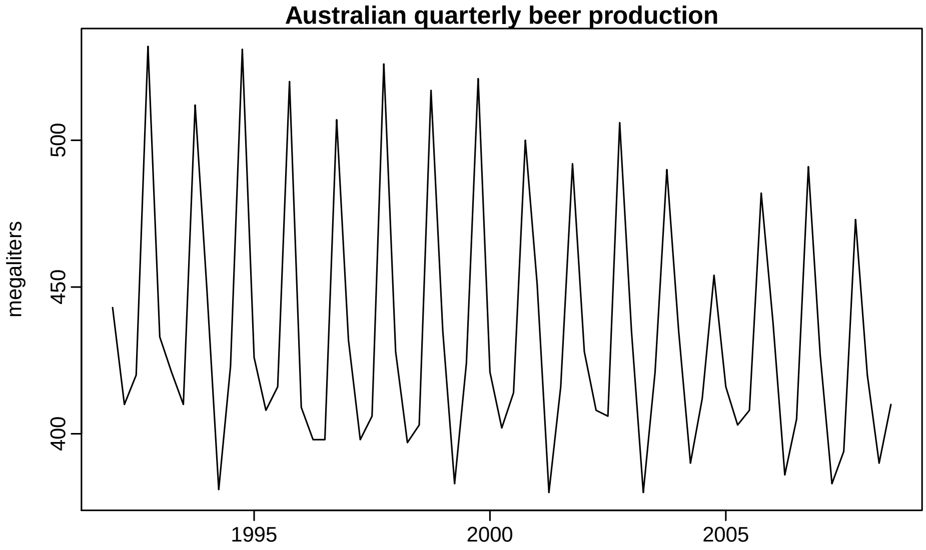
Seasonal subseries plots

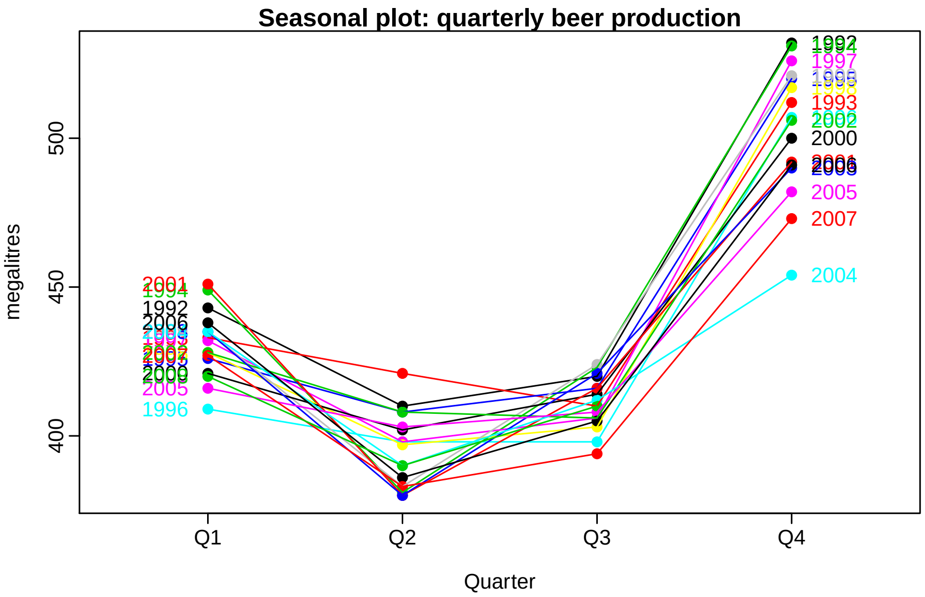


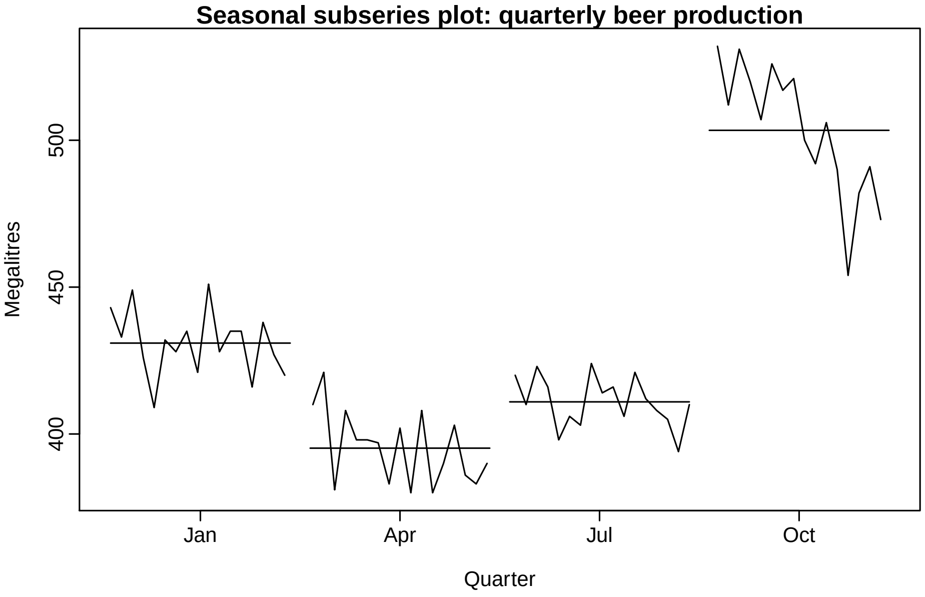
- Data for each season collected together in time plot as separate time series.
- Enables the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.
- In R: `monthplot`

Quarterly Australian Beer Production

```
beer <- window(ausbeer,start=1992)
plot(beer)
seasonplot(beer,year.labels=TRUE)
monthplot(beer)
```





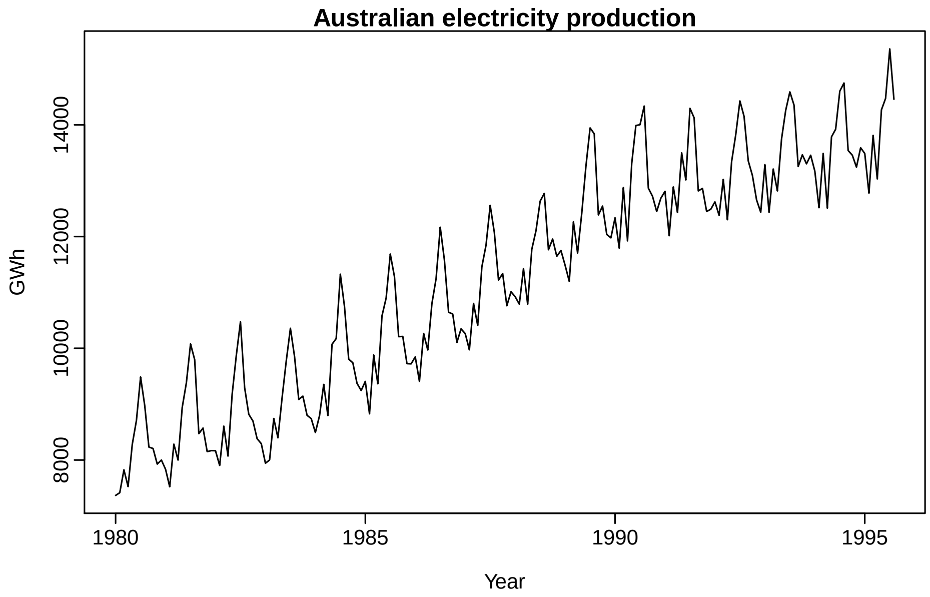


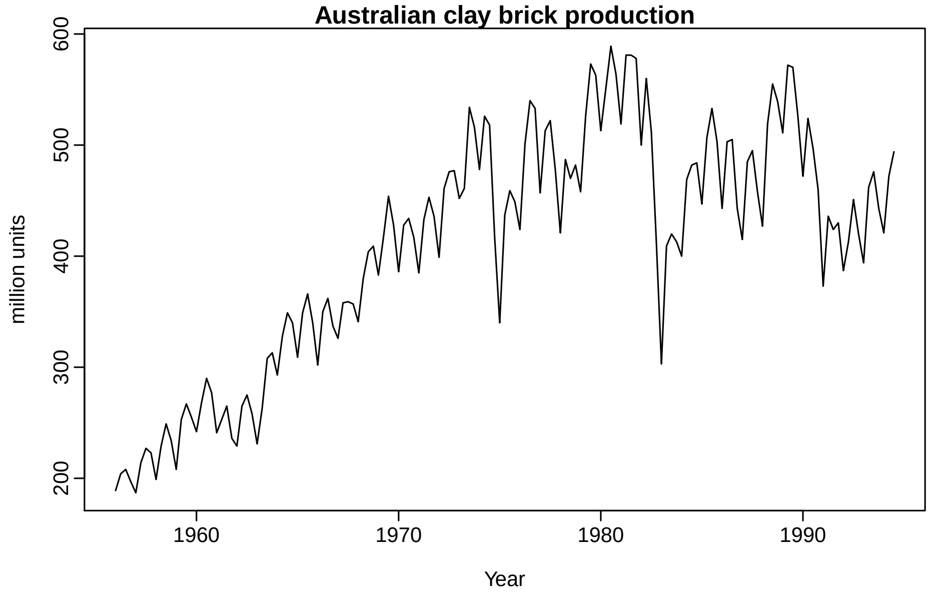
2.2 Seasonal or cyclic?

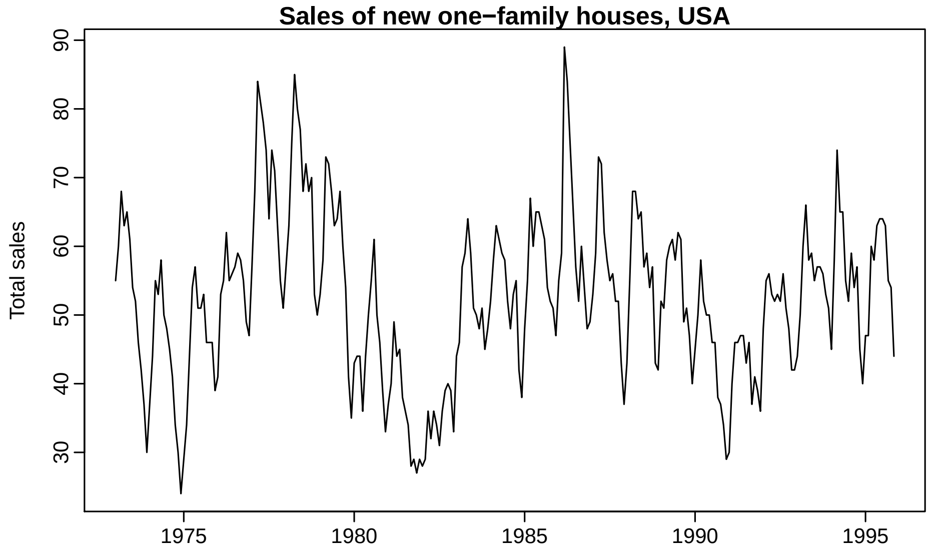
Trend pattern exists when there is a long-term increase or decrease in the data.

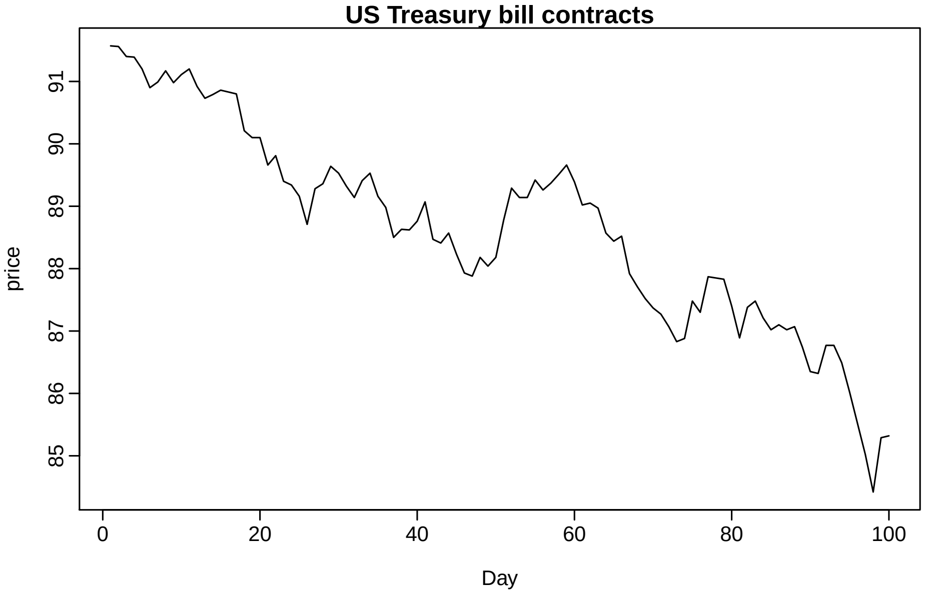
Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

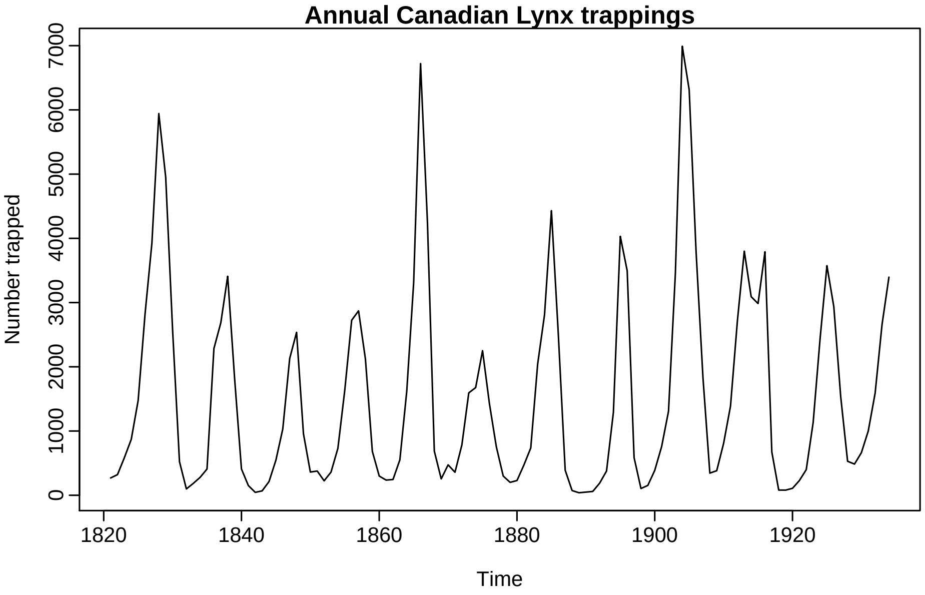
Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).


Australian electricity production


Australian clay brick production


Sales of new one−family houses, USA





Differences between seasonal and cyclic patterns:

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

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

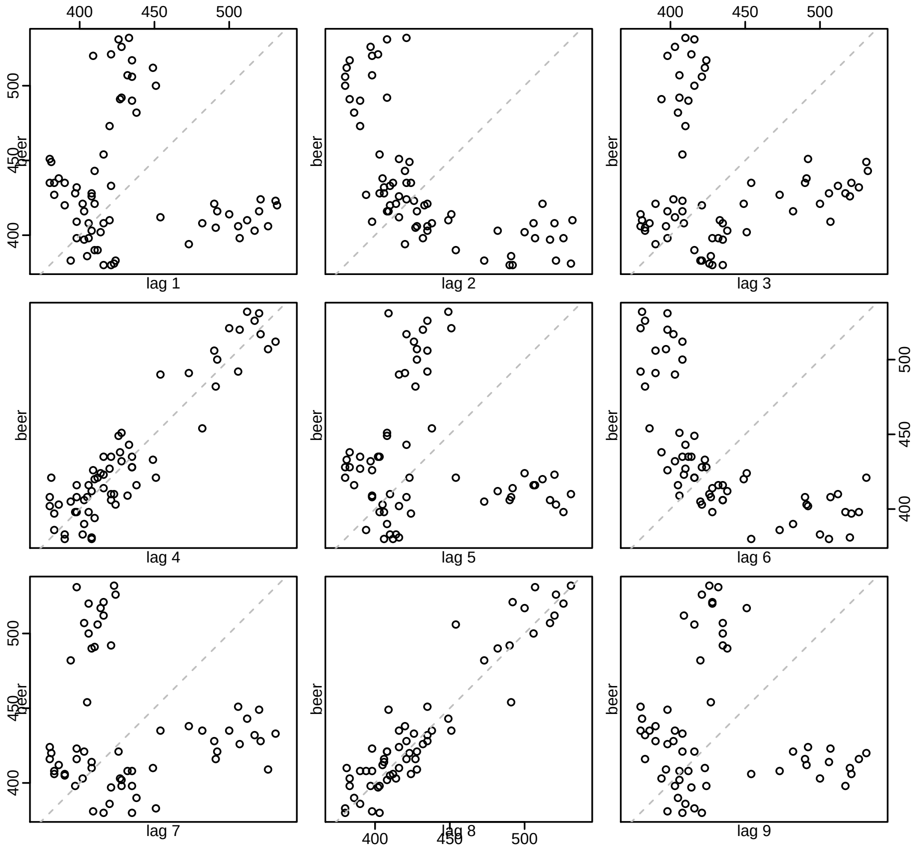
2.3 Autocorrelation

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

Autocovariance and **autocorrelation**: measure linear relationship between **lagged values** of a time series y.

We measure the relationship between: y_t and y_{t-1}
y_t and y_{t-2}
y_t and y_{t-3}
etc.

```
> lag.plot(beer,lags=9,do.lines=FALSE)
```



- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.

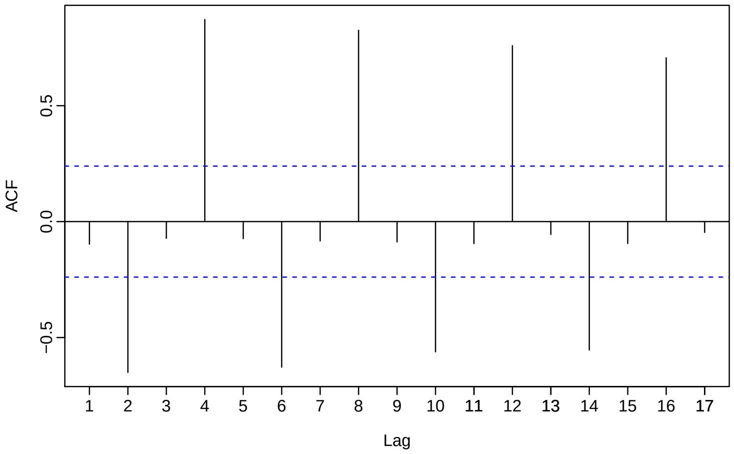
We denote the sample autocovariance at lag k by c_k and the sample autocorrelation at lag k by r_k. Then define

$$c_k = \frac{1}{T}\sum_{t=k+1}^{T}(y_t-\bar{y})(y_{t-k}-\bar{y}) \qquad \text{and} \qquad r_k = c_k/c_0$$

- r_1 indicates how successive values of y relate to each other
- r_2 indicates how y values two periods apart relate to each other
- r_k is *almost* the same as the sample correlation between y_t and y_{t-k}.

Results for first 9 lags for beer data:

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
−0.126	−0.650	−0.094	0.863	−0.099	−0.642	−0.098	0.834	−0.116



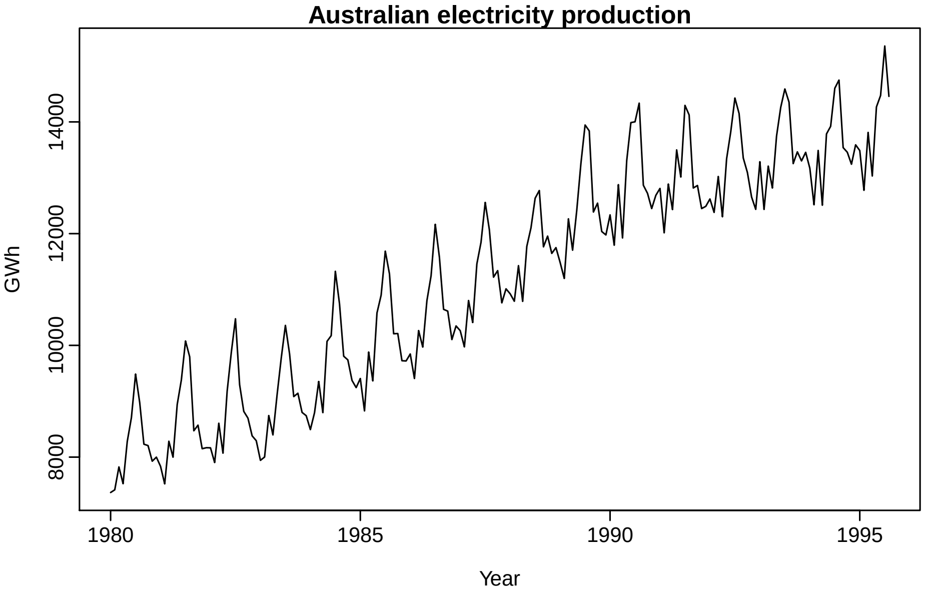
- r_4 higher than for the other lags. This is due to **the seasonal pattern in the data:** the peaks tend to be **4 quarters** apart and the troughs tend to be **2 quarters** apart.
- r_2 is more negative than for the other lags because troughs tend to be 2 quarters behind peaks.
- Together, the autocorrelations at lags 1, 2, . . . , make up the *autocorrelation* or ACF.
- The plot is known as a **correlogram**

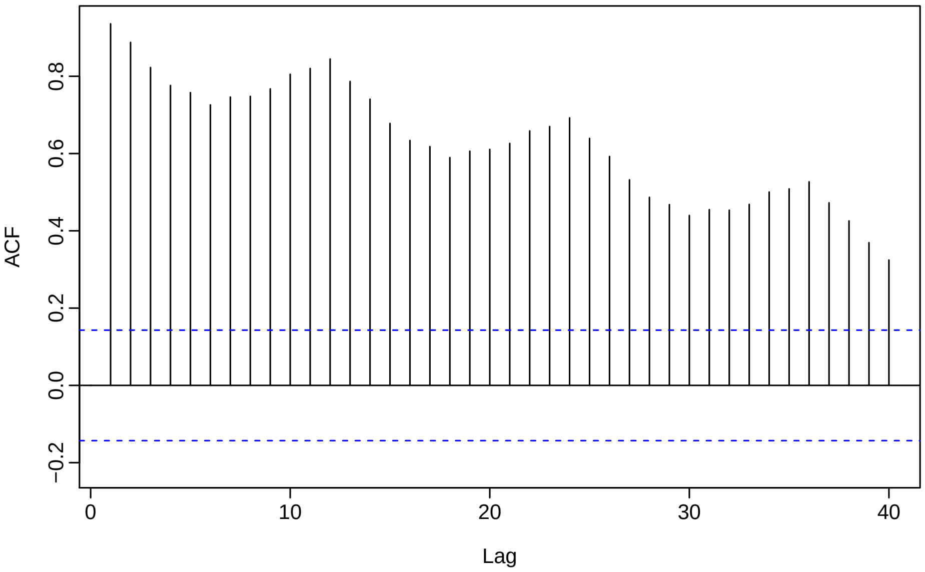
Recognizing seasonality in a time series

If there is seasonality, the ACF at the seasonal lag (e.g., 12 for monthly data) will be **large and positive**.

- For seasonal monthly data, a large ACF value will be seen at lag 12 and possibly also at lags 24, 36, . . .
- For seasonal quarterly data, a large ACF value will be seen at lag 4 and possibly also at lags 8, 12, . . .

Australian monthly electricity production





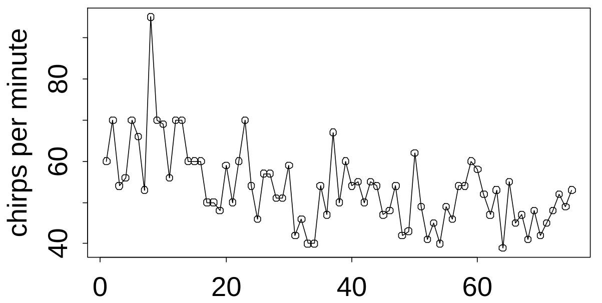
Time plot shows clear trend and seasonality.

The same features are reflected in the ACF.

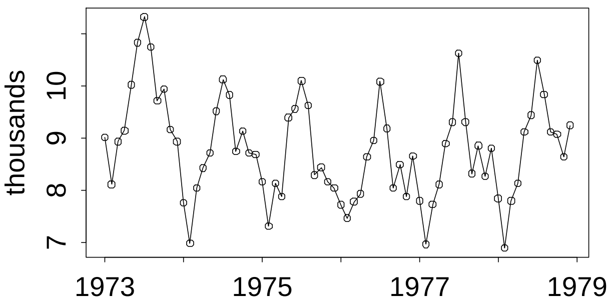
- The slowly decaying ACF indicates trend.
- The ACF peaks at lags 12, 24, 36, . . . , indicate seasonality of length 12.

Which is which?

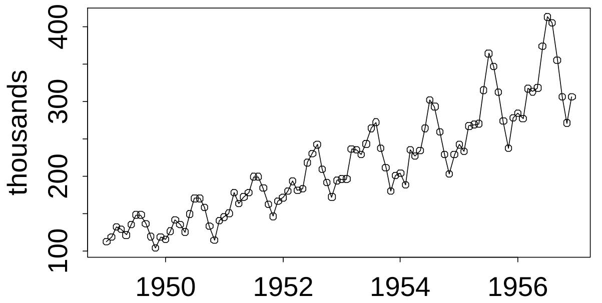
1. Daily morning temperature of a cow



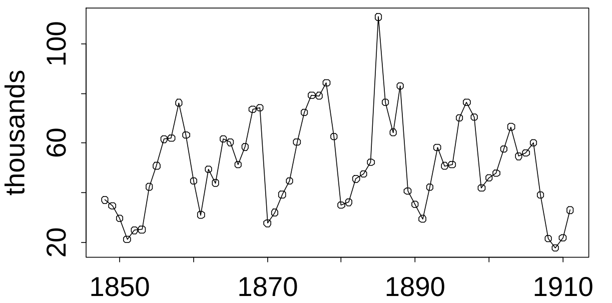
2. Accidental deaths in USA (monthly)



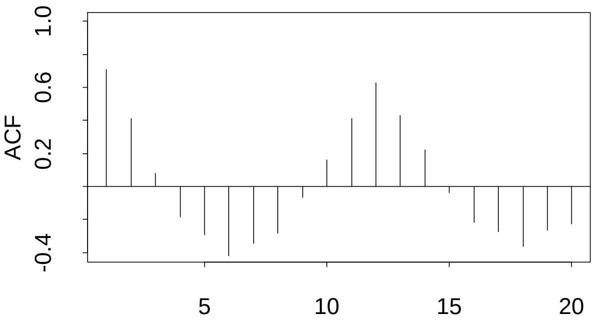
3. International airline passengers



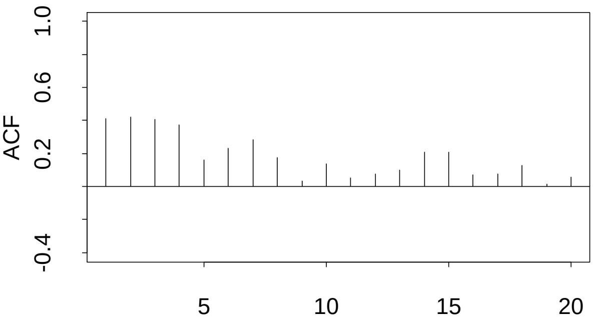
4. Annual mink trappings (Canada)



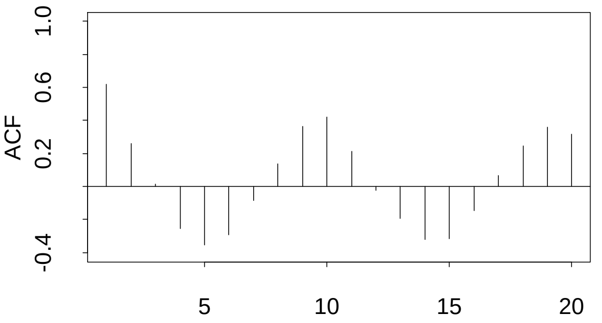
A



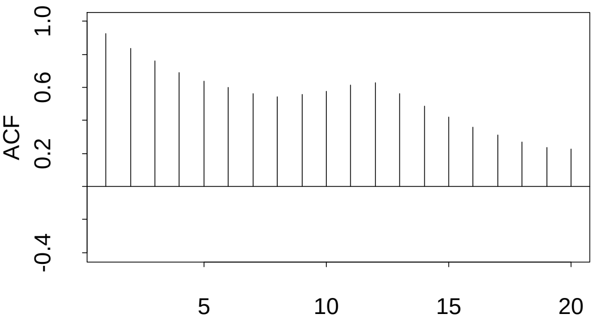
B



C



D



2.4 Forecast residuals

Residuals in forecasting: difference between observed value and its forecast based on all previous observations: $e_t = y_t - \hat{y}_{t|t-1}$.

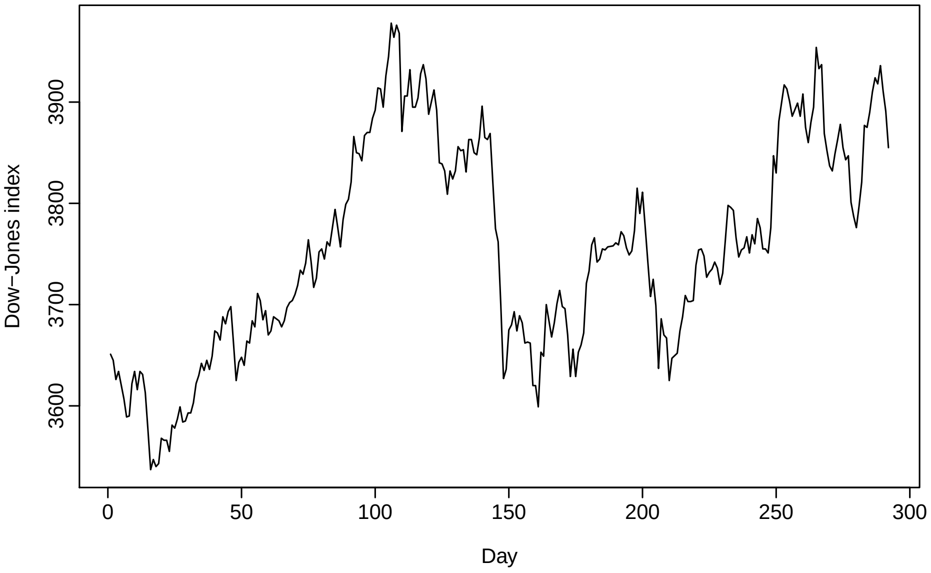
Assumptions

1. $\{e_t\}$ uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero. If they don't, then forecasts are biased.

Useful properties (for Forecast intervals)

3. $\{e_t\}$ have constant variance.
4. $\{e_t\}$ are normally distributed.

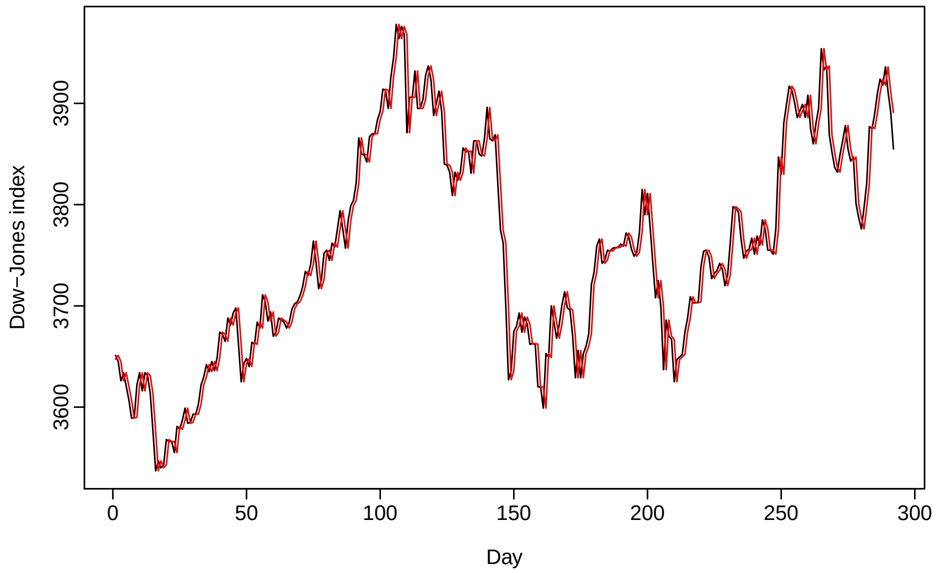
Forecasting Dow-Jones index

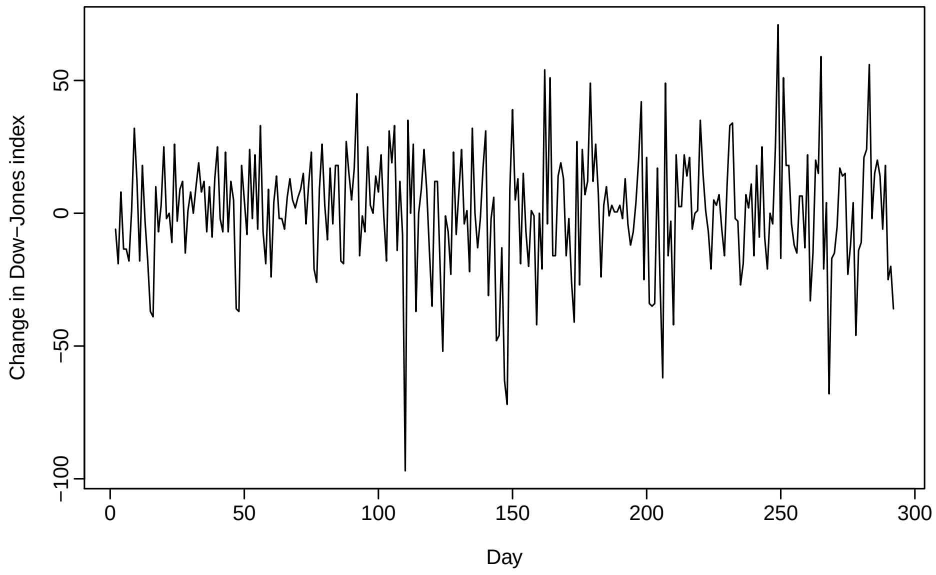


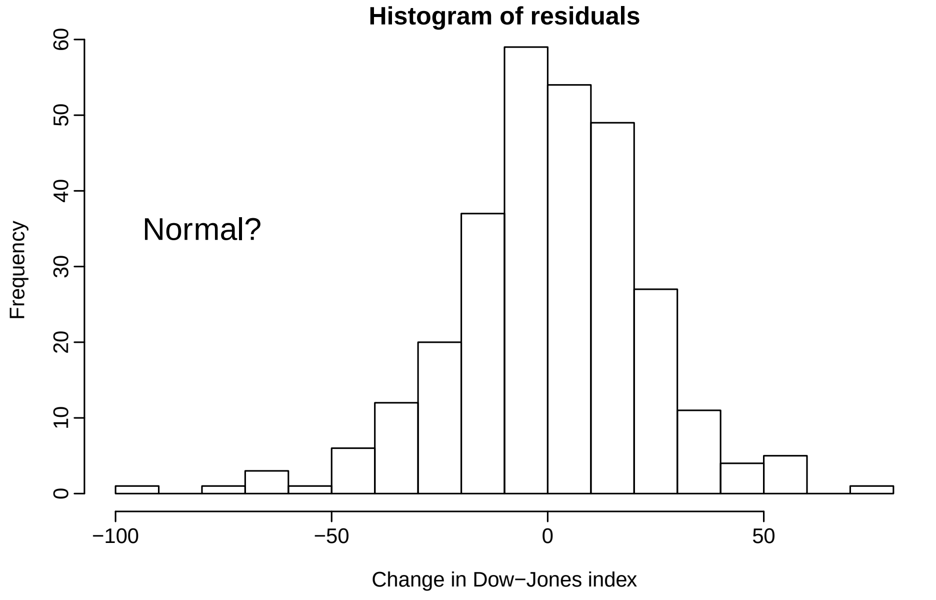
Naïve forecast:

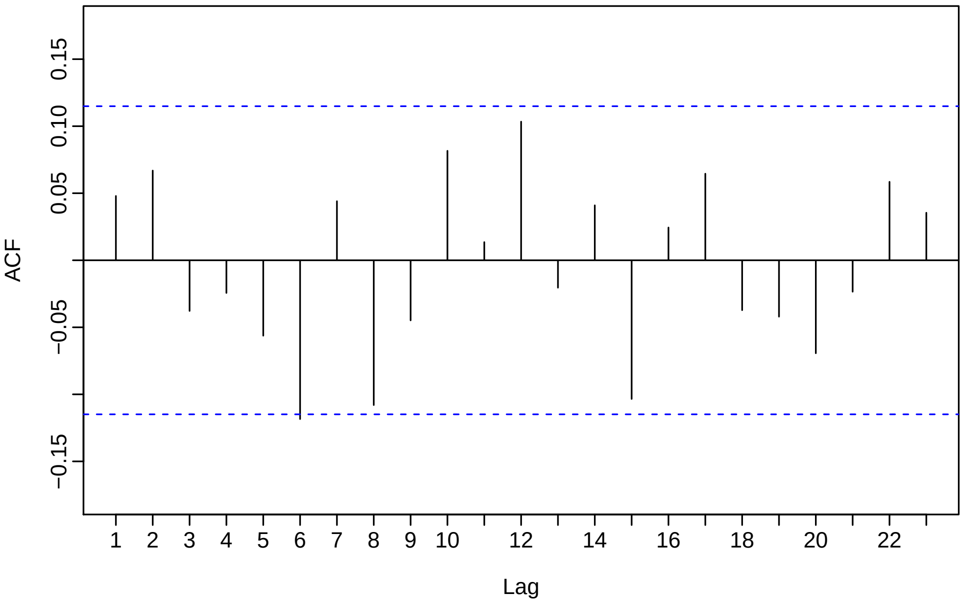
$$\hat{y}_{t|t-1} = y_{t-1}$$
$$e_t = y_t - y_{t-1}$$

Note: e_t are one-step-forecast residuals


Dow−Jones index
3600
3700
3800
3900
0
50
100
150
200
250
300
Day

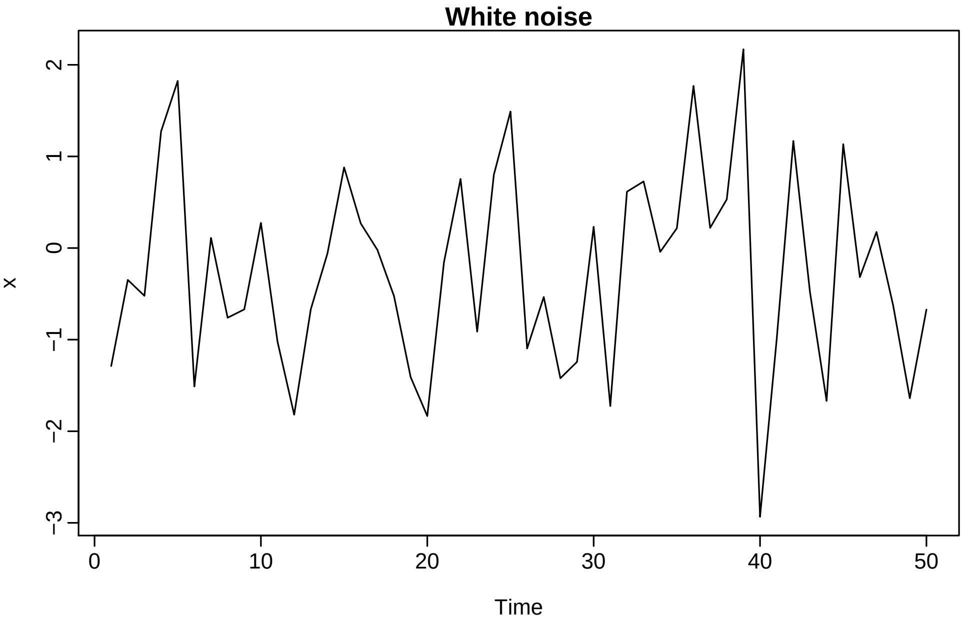

Change in Dow−Jones index
−100
−50
0
50
0
50
100
150
200
250
300
Day


Histogram of residuals
Frequency
0
10
20
30
40
50
60
Normal?
−100
−50
0
50
Change in Dow−Jones index



```
fc <- rwf(dj)
res <- residuals(fc)
plot(res)
hist(res,breaks="FD")
Acf(res,main="")
```

2.5 White noise

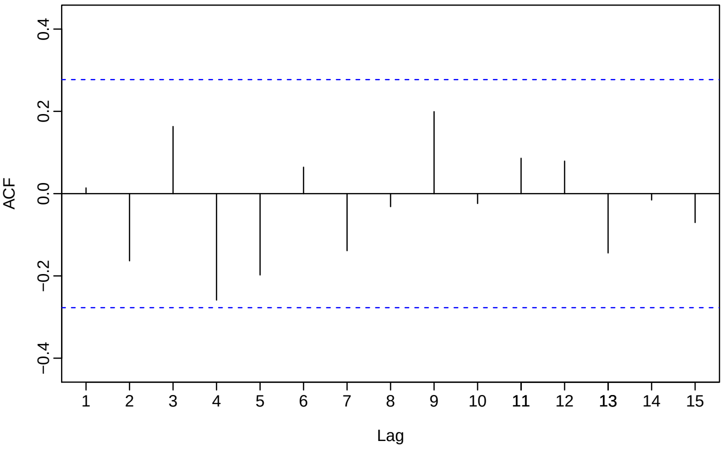


White noise data is uncorrelated across time with zero mean and constant variance.

(Technically, we require independence as well.)

Think of white noise as completely uninteresting with no predictable patterns.

$$\begin{aligned}
r_1 &= 0.013\\
r_2 &= -0.163\\
r_3 &= 0.163\\
r_4 &= -0.259\\
r_5 &= -0.198\\
r_6 &= 0.064\\
r_7 &= -0.139\\
r_8 &= -0.032\\
r_9 &= 0.199\\
r_{10} &= -0.240
\end{aligned}$$



For uncorrelated data, we would expect each autocorrelation to be close to zero.

Sampling distribution of autocorrelations

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

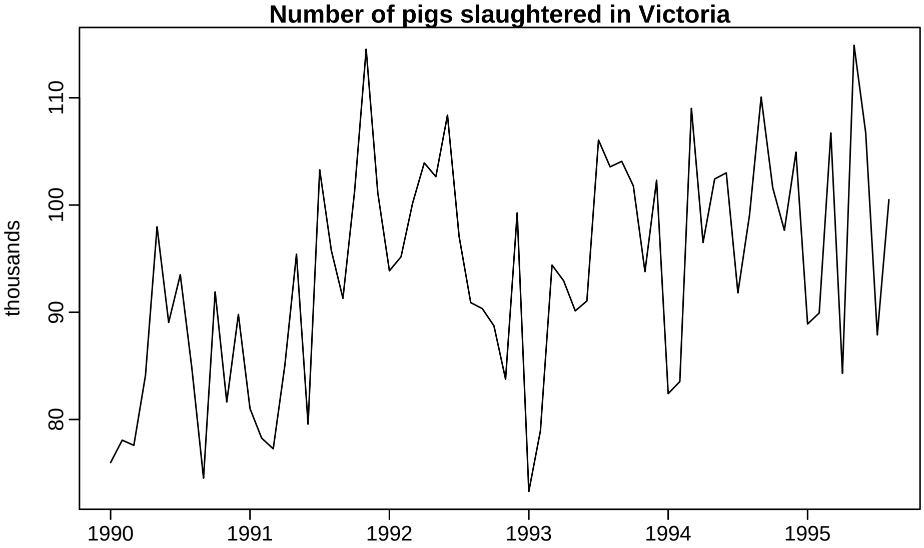
- 95% of all r_k for white noise must lie within $\pm 1.96/\sqrt{T}$.
- If this is not the case, the series is probably not WN.
- Common to plot lines at $\pm 1.96/\sqrt{T}$ when plotting ACF. These are the ***critical values***.

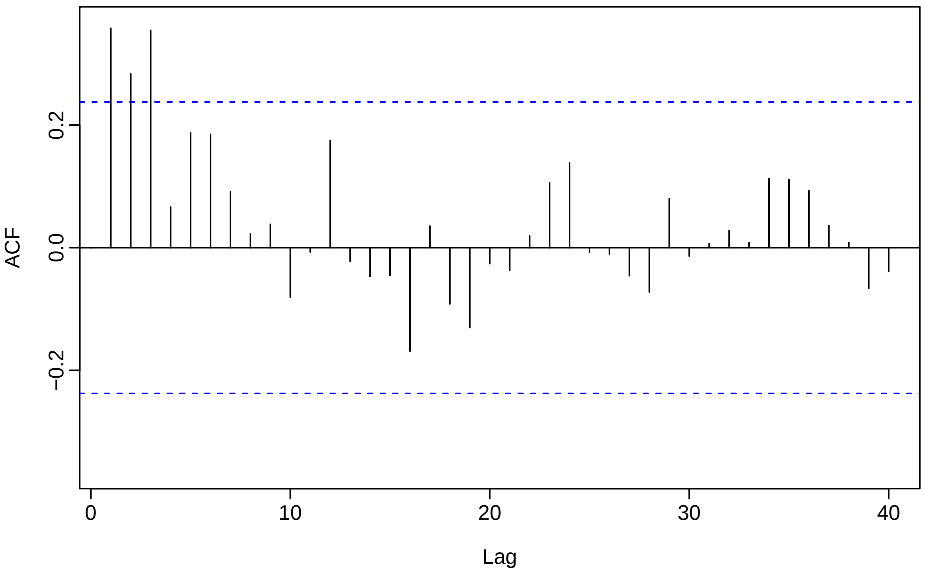
Example: $T = 50$ and so critical values at $\pm 1.96/\sqrt{50} = \pm 0.28$.

All autocorrelation coefficients lie within these limits, confirming that the data are white noise. (More precisely, the data cannot be distinguished from white noise.)

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 1990 through August 1995. (Source: Australian Bureau of Statistics.)





- Difficult to detect pattern in time plot.
- ACF shows some significant autocorrelation at lags 1, 2, and 3.
- r_{12} relatively large although not significant. This may indicate some slight seasonality.

These show the series is **not a white noise series**.

ACF of residuals

- We assume that the residuals are white noise (uncorrelated, mean zero, constant variance). If they aren't, then there is information left in the residuals that should be used in computing forecasts.
- So a standard residual diagnostic is to check the ACF of the residuals of a forecasting method.
- We *expect* these to look like white noise.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Box-Pierce test

$$Q = T\sum_{k=1}^{h} r_k^2$$

where h is max lag being considered and T is number of observations.

- My preferences: $h = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- If each r_k close to zero, Q will be **small**.
- If some r_k values large (positive or negative), Q will be **large**.

Ljung-Box test

$$Q^* = T(T+2)\sum_{k=1}^{h}(T-k)^{-1}r_k^2$$

where h is max lag being considered and T is number of observations.

- My preferences: $h = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- Better performance, especially in small samples.
- If data are WN, Q^* has χ^2 distribution with $(h-K)$ degrees of freedom where K = no. parameters in model.
- When applied to raw data, set $K = 0$.
- For the Dow-Jones example,

```
res <- residuals(naive(dj))

# lag=h and fitdf=K
> Box.test(res, lag=10, fitdf=0)
  Box-Pierce test
X-squared = 14.0451, df = 10, p-value = 0.1709
> Box.test(res, lag=10, fitdf=0, type="Lj")
  Box-Ljung test
X-squared = 14.4615, df = 10, p-value = 0.153
```

Exercise

1. Calculate the residuals from a seasonal naive forecast applied to the quarterly Australian beer production data from 1992.
2. Test if the residuals are white noise.

```
beer <- window(ausbeer,start=1992)
fc <- snaive(beer)
res <- residuals(fc)
Acf(res)
Box.test(res, lag=8, fitdf=0, type="Lj")
```

2.6 Evaluating forecast accuracy

Let y_t denote the tth observation and $\hat{y}_{t|t-1}$ denote its forecast based on all previous data, where $t = 1,\dots,T$. Then the following measures are useful.

$$\text{MAE} = T^{-1}\sum_{t=1}^{T}|y_t - \hat{y}_{t|t-1}|$$

$$\text{MSE} = T^{-1}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2 \qquad \text{RMSE} = \sqrt{T^{-1}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2}$$

$$\text{MAPE} = 100T^{-1}\sum_{t=1}^{T}|y_t - \hat{y}_{t|t-1}|/|y_t|$$

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all t, and y has a natural zero.

Mean Absolute Scaled Error

$$\text{MASE} = T^{-1} \sum_{t=1}^{T} |y_t - \hat{y}_{t|t-1}|/Q$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006)

For non-seasonal time series,

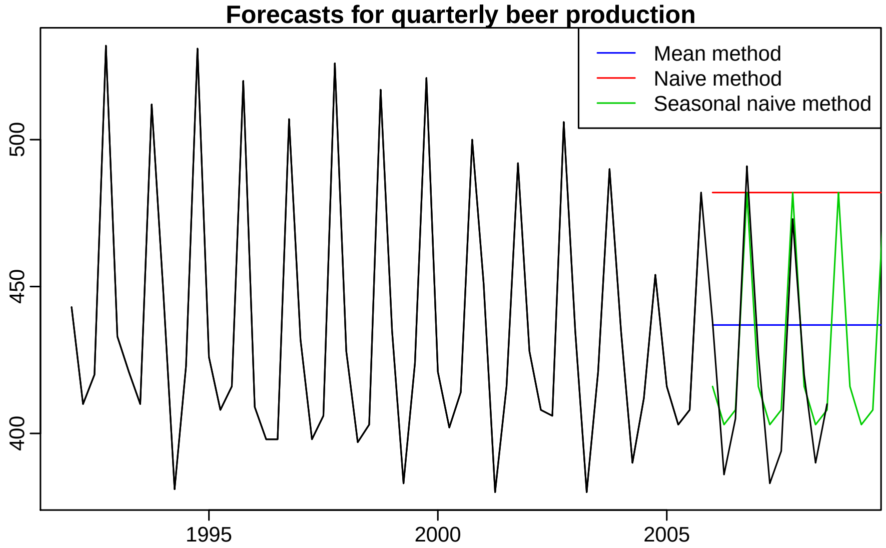
$$Q = (T-1)^{-1} \sum_{t=2}^{T} |y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naive method.

For seasonal time series,

$$Q = (T-m)^{-1} \sum_{t=m+1}^{T} |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naive method.



Mean method

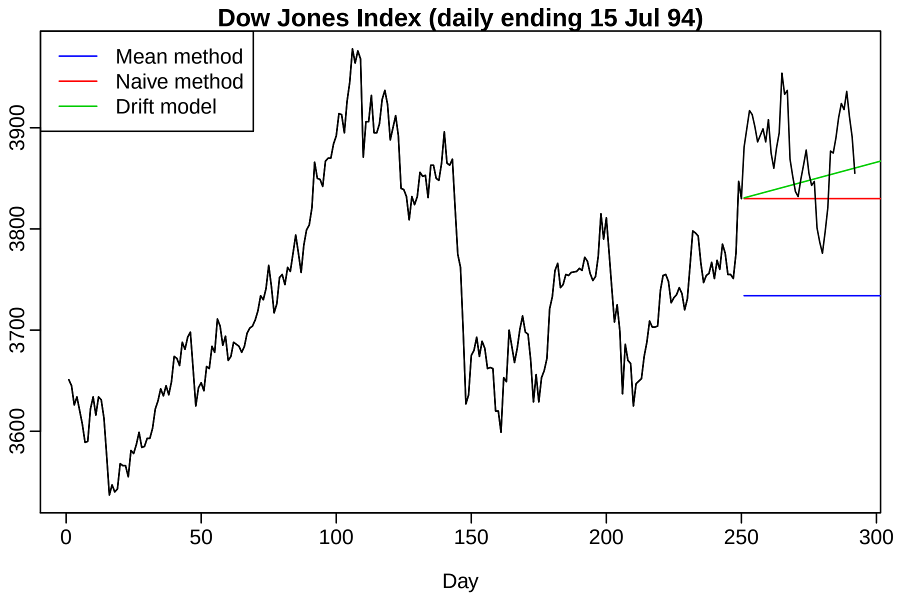
```
   RMSE       MAE      MAPE      MASE
38.0145   33.7776    8.1700    2.2990
```

Naïve method

```
   RMSE       MAE      MAPE      MASE
70.9065   63.9091   15.8765    4.3498
```

Seasonal naïve method

```
   RMSE       MAE      MAPE      MASE
12.9685   11.2727    2.7298    0.7673
```



Measures of forecast accuracy

Mean method

```
    RMSE        MAE      MAPE     MASE
148.2357   142.4185    3.6630   8.6981
```

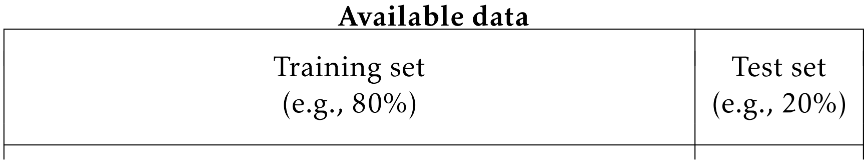
Naïve method

```
    RMSE        MAE      MAPE     MASE
 62.0285    54.4405    1.3979   3.3249
```

Drift model

```
    RMSE        MAE      MAPE     MASE
 53.6977    45.7274    1.1758   2.7928
```

Training and test sets


Available data

Training set
(e.g., 80%)

Test set
(e.g., 20%)

- The test set must not be used for *any* aspect of model development or calculation of forecasts.
- Forecast accuracy is based only on the test set.

```
beer3 <- window(ausbeer,start=1992,end=2005.99)
beer4 <- window(ausbeer,start=2006)

fit1 <- meanf(beer3,h=20)
fit2 <- rwf(beer3,h=20)

accuracy(fit1,beer4)
accuracy(fit2,beer4)
```

In-sample accuracy (one-step forecasts)

```
accuracy(fit1)
accuracy(fit2)
```

Beware of over-fitting

- A model which fits the data well does not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters. (Compare R^2)
- Over-fitting a model to data is as bad as failing to identify the systematic pattern in the data.
- Problems can be overcome by measuring true *out-of-sample* forecast accuracy. That is, total data divided into “training” set and “test” set. Training set used to estimate parameters. Forecasts are made for test set.
- Accuracy measures computed for errors in test set only.

Poll: true or false?

1. Good forecast methods should have normally distributed residuals.
2. A model with small residuals will give good forecasts.
3. The best measure of forecast accuracy is MAPE.
4. If your model doesn’t forecast well, you should make it more complicated.
5. Always choose the model with the best forecast accuracy as measured on the test set.

2.7 Lab Session 2

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. The function `tsdisplay(data, plot.type="scatter")` is useful for showing a time plot, ACF plot and lagged scatterplot on the same graph. Use it to produce plots of the following time series:

   `bricksq`, `hsales`, `ibmclose`

   Can you spot the effects of seasonality, cyclicity and trend?

2. For each of the same series (`bricksq`, `ibmclose`, `hsales`):

   (a) Use either the naive or seasonal naive forecasting method and apply it to the full data set.

   (b) Compute the residuals and plot their ACF. Do the residuals appear to be white noise? What did your forecasting method miss?

   (c) Do a Ljung-Box test on the residuals. What do the results mean?

3. For the data set `bricksq`:
   (a) Split the data into two parts using
   ```
   bricks1 <- window(bricksq, end=1987.99)
   bricks2 <- window(bricksq, start=1988)
   ```
   (b) Check that your data have been split appropriately by producing the following plot.
   ```
   plot(bricksq)
   lines(bricks1,col="red")
   lines(bricks2,col="blue")
   ```
   (c) Calculate forecasts using each of the four benchmark methods applied to `bricks1`.
   (d) Compare the accuracy of your forecasts against the actual values stored in `bricks2`. For example:
   ```
   f1 <- meanf(bricks1)
   accuracy(f1,bricks2)
   ```
   (e) Which method does best? Why?
   (f) For the best method, compute the residuals and plot them. For example
   ```
   res <- residuals(f1)
   plot(res)
   hist(res, breaks="FD")
   Acf(res)
   ```
   Do the residuals appear to be uncorrelated and normally distributed?

4. Consider the daily closing IBM stock prices (data set `ibmclose`).
   (a) Produce some plots of the data in order to become familiar with it.
   (b) Split the data into a training set of 300 observations and a test set of 69 observations.
   (c) Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?
   (d) For the best method, compute the residuals and plot them. What do the plots tell you?
   (e) Can you invent a better forecasting method than any of the benchmark methods for these data?

5. Consider the sales of new one-family houses in the USA (Jan 1987 – Nov 1995). Data set: `hsales`.
   (a) Produce some plots of the data in order to become familiar with it.
   (b) Split the data into a training set and a test set, where the test set is the last two years of data.
   (c) Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?
   (d) For the best method, compute the residuals and plot them. What do the plots tell you?
   (e) Can you invent a better forecasting method than any of the benchmark methods for these data?

3 Exponential smoothing

3.1 The state space perspective

- Observed data: $y_1, \dots, y_T$.
- Unobserved state: $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$.
- Forecast $\hat{y}_{T+h|T} = \mathrm{E}(y_{T+h}|\boldsymbol{x}_T)$.
- The "forecast variance" is $\mathrm{Var}(y_{T+h}|\boldsymbol{x}_T)$.
- A prediction interval or "interval forecast" is a range of values of y_{T+h} with high probability.

3.2 Simple exponential smoothing

Component form

$$\begin{aligned} \text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t \\ \text{Smoothing equation} \quad & \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \end{aligned}$$

$$\begin{aligned} \ell_1 &= \alpha y_1 + (1-\alpha)\ell_0 \\ \ell_2 &= \alpha y_2 + (1-\alpha)\ell_1 = \alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0 \\ \ell_3 &= \alpha y_3 + (1-\alpha)\ell_2 = \sum_{j=0}^{2} \alpha(1-\alpha)^j y_{3-j} + (1-\alpha)^3\ell_0 \\ &\vdots \\ \ell_t &= \sum_{j=0}^{t-1} \alpha(1-\alpha)^j y_{t-j} + (1-\alpha)^t\ell_0 \end{aligned}$$

Forecast equation

$$\hat{y}_{t+h|t} = \sum_{j=1}^{t} \alpha(1-\alpha)^{t-j} y_j + (1-\alpha)^t\ell_0, \qquad (0 \le \alpha \le 1)$$

	Weights assigned to observations for:			
Observation	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$
y_t	0.2	0.4	0.6	0.8
y_{t-1}	0.16	0.24	0.24	0.16
y_{t-2}	0.128	0.144	0.096	0.032
y_{t-3}	0.1024	0.0864	0.0384	0.0064
y_{t-4}	$(0.2)(0.8)^4$	$(0.4)(0.6)^4$	$(0.6)(0.4)^4$	$(0.8)(0.2)^4$
y_{t-5}	$(0.2)(0.8)^5$	$(0.4)(0.6)^5$	$(0.6)(0.4)^5$	$(0.8)(0.2)^5$

- Limiting cases: $\alpha \to 1, \quad \alpha \to 0.$

State space form

$$\begin{array}{rl} \text{Observation equation} & y_t = \ell_{t-1} + e_t \\ \text{State equation} & \ell_t = \ell_{t-1} + \alpha e_t \end{array}$$

- $e_t = y_t - \ell_{t-1} = y_t - \hat{y}_{t|t-1}$ for $t = 1, \dots, T$, the one-step within-sample forecast error at time t.
- ℓ_t is an unobserved "state".
- Need to estimate α and ℓ_0.

Optimisation

- Need to choose value for α and ℓ_0
- Similarly to regression — we choose α and ℓ_0 by minimising MSE:

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2 = \frac{1}{T}\sum_{t=1}^{T} e_t^2.$$

- Unlike regression there is no closed form solution — use numerical optimization.

Multi-step forecasts

$$\hat{y}_{T+h|T} = \hat{y}_{T+1|T}, \qquad h = 2, 3, \dots$$

- A "flat" forecast function.
- Remember, a forecast is an estimated mean of a future value.
- So with no trend, no seasonality, and no other patterns, the forecasts are constant.

SES in R

```
library(fpp)
fit <- ses(oil, h=3)
plot(fit)
summary(fit)
```

3.3 Trend methods

Holt's local trend method

- Holt (1957) extended SES to allow forecasting of data with trends.
- Two smoothing parameters: α and β^* (with values between 0 and 1).

$$\begin{aligned}
\text{Forecast} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\
\text{Level} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1},
\end{aligned}$$

- ℓ_t denotes an estimate of the level of the series at time t
- b_t denotes an estimate of the slope of the series at time t.

$$\begin{aligned}
\text{Observation equation} \quad & y_t = \ell_{t-1} + b_{t-1} + e_t \\
\text{State equations} \quad & \ell_t = \ell_{t-1} + b_{t-1} + \alpha e_t \\
& b_t = b_{t-1} + \beta e_t
\end{aligned}$$

- $\beta = \alpha\beta^*$
- $e_t = y_t - (\ell_{t-1} + b_{t-1}) = y_t - \hat{y}_{t|t-1}$
- Need to estimate $\alpha, \beta, \ell_0, b_0$.

Holt's method in R

```
fit2 <- holt(ausair, h=5)
plot(fit2)
summary(fit2)
fit1 <- holt(strikes)
plot(fit1$model)
plot(fit1, plot.conf=FALSE)
lines(fitted(fit1), col="red")
fit1$model

fit2 <- ses(strikes)
plot(fit2$model)
plot(fit2, plot.conf=FALSE)
lines(fit1$mean, col="red")

accuracy(fit1)
accuracy(fit2)
```

Comparing Holt and SES

- Holt's method will almost always have better in-sample RMSE because it is optimized over one additional parameter.
- It may not be better on other measures.
- You need to compare out-of-sample RMSE (using a test set) for the comparison to be useful.
- But we don't have enough data.
- A better method for comparison will be in the next session!

Exponential trend method

$$\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t b_t^h \\
\text{Observation equation} \quad & y_t = (\ell_{t-1} b_{t-1}) + e_t \\
\text{State equations} \quad & \ell_t = \ell_{t-1} b_{t-1} + \alpha e_t \\
& b_t = b_{t-1} + \beta e_t / \ell_{t-1}
\end{aligned}$$

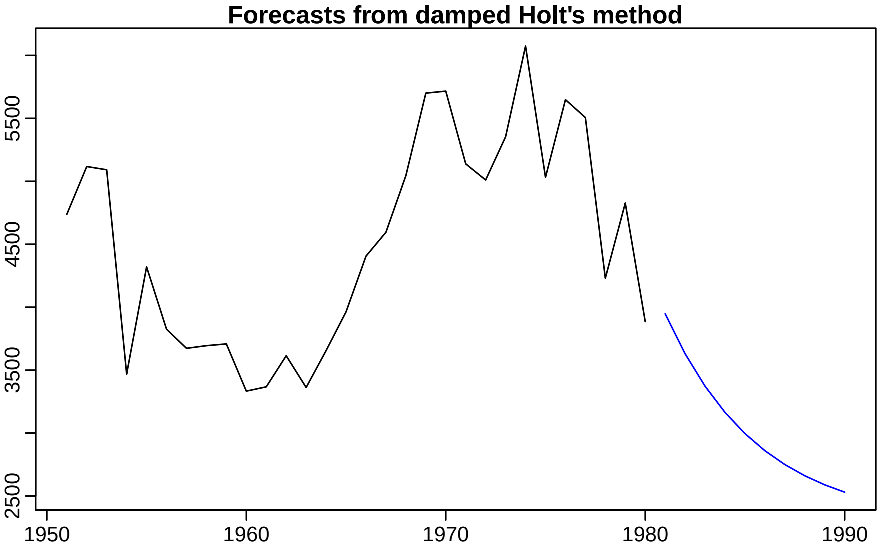
- ℓ_t denotes an estimate of the level of the series at time t
- b_t denotes an estimate of the relative growth of the series at time t.
- In R: `holt(x, exponential=TRUE)`

Additive damped trend

- Gardner and McKenzie (1985) suggested that the trends should be "damped" to be more conservative for longer forecast horizons.
- Damping parameter $0 < \phi < 1$.

$$\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + \cdots + \phi^h) b_t \\
\text{Observation equation} \quad & y_t = \ell_{t-1} + \phi b_{t-1} + e_t \\
\text{State equations} \quad & \ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha e_t \\
& b_t = \phi b_{t-1} + \beta e_t
\end{aligned}$$

- If $\phi = 1$, identical to Holt's linear trend.
- As $h \to \infty$, $\hat{y}_{T+h|T} \to \ell_T + \phi b_T / (1 - \phi)$.
- Short-run forecasts trended, long-run forecasts constant.



Trend methods in R

```
fit4 <- holt(air, h=5, damped=TRUE)
plot(fit4)
summary(fit4)
```

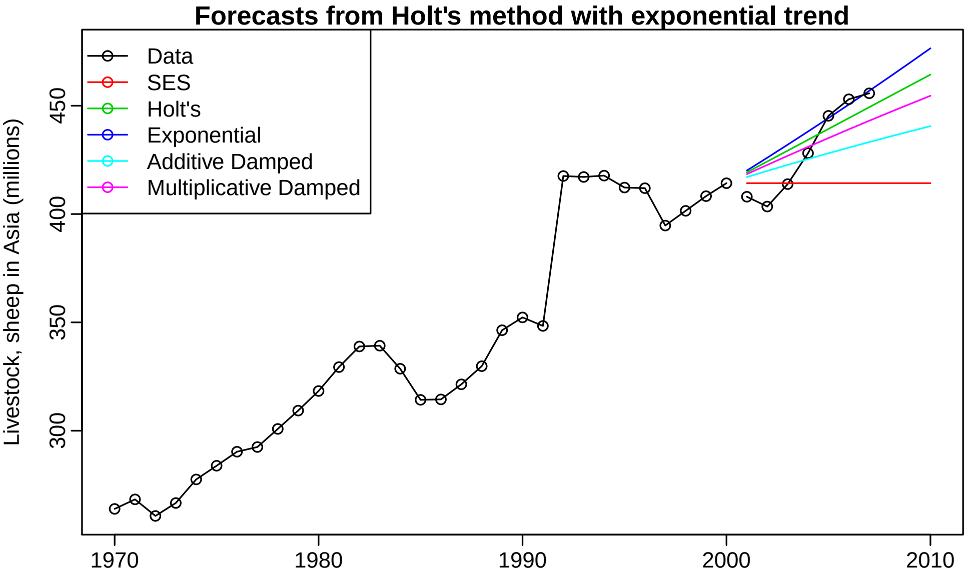
Multiplicative damped trend method

Taylor (2003) introduced multiplicative damping.

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t b_t^{(\phi+\phi^2+\dots+\phi^h)}\\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} b_{t-1}^{\phi})\\ b_t &= \beta^*(\ell_t/\ell_{t-1}) + (1-\beta^*) b_{t-1}^{\phi}\end{aligned}$$

- $\phi = 1$ gives exponential trend method
- Forecasts converge to $\ell_T + b_T^{\phi/(1-\phi)}$ as $h \to \infty$.

Example: Sheep in Asia



3.4 Seasonal methods

Holt-Winters additive method

- Holt and Winters extended Holt's method to capture seasonality.
- Three smoothing equations—one for the level, one for trend, and one for seasonality.
- Parameters: $0 \le \alpha \le 1$, $0 \le \beta^* \le 1$, $0 \le \gamma \le 1-\alpha$ and $m =$ period of seasonality.

State space form

$$\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t-m+h_m^+} \qquad\qquad h_m^+ = \lfloor (h-1) \mod m \rfloor + 1\\
y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + e_t\\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha e_t\\
b_t &= b_{t-1} + \beta e_t\\
s_t &= s_{t-m} + \gamma e_t.
\end{aligned}$$

$$\begin{aligned}
\hat{y}_{t+h|t} &= (\ell_t + hb_t)s_{t-m+h_m^+}\\
y_t &= (\ell_{t-1} + b_{t-1})s_{t-m} + e_t\\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha e_t/s_{t-m}\\
b_t &= b_{t-1} + \beta e_t/s_{t-m}\\
s_t &= s_{t-m} + \gamma e_t/(\ell_{t-1} + b_{t-1}).
\end{aligned}$$

- Most textbooks use $s_t = \gamma(y_t/\ell_t) + (1-\gamma)s_{t-m}$
- We optimize for α, β^*, γ, ℓ_0, b_0, s_0, $s_{-1},\dots,s_{1-m}$.

Seasonal methods in R

```
aus1 <- hw(austourists)
aus2 <- hw(austourists, seasonal="mult")

plot(aus1)
plot(aus2)

summary(aus1)
summary(aus2)
```

Holt-Winters damped method

Often the single most accurate forecasting method for seasonal data:

$$\begin{aligned}
y_t &= (\ell_{t-1} + \phi b_{t-1})s_{t-m} + e_t\\
\ell_t &= \ell_{t-1} + \phi b_{t-1} + \alpha e_t/s_{t-m}\\
b_t &= \phi b_{t-1} + \beta e_t/s_{t-m}\\
s_t &= s_{t-m} + \gamma e_t/(\ell_{t-1} + \phi b_{t-1}).
\end{aligned}$$

```
aus3 <- hw(austourists, seasonal="mult", damped=TRUE)
summary(aus3)
plot(aus3)
```

A confusing array of methods?

- All these methods can be confusing!
- How to choose between them?
- The ETS framework provides an automatic way of selecting the best method.
- It was developed to solve the problem of automatically forecasting pharmaceutical sales across thousands of products.

3.5 Lab Session 3

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. For this exercise, use the price of a dozen eggs in the United States from 1900–1993 (data set `eggs`). Experiment with the various options in the `holt()` function to see how much the forecasts change with damped or exponential trend. Also try changing the parameter values for α and β to see how they affect the forecasts. Try to develop an intuition of what each parameter and argument is doing to the forecasts.

   [Hint: use `h=100` when calling `holt()` so you can clearly see the differences between the various options when plotting the forecasts.]

   Which model gives the best RMSE?

   Do the residuals from the best model look like white noise?

2. For this exercise, use the monthly Australian short-term overseas visitors data, May 1985–April 2005. (Data set: `visitors`.)

   (a) Make a time plot of your data and describe the main features of the series.

   (b) Forecast the next two years using Holt-Winters' multiplicative method.

   (c) Why is multiplicative seasonality necessary here?

   (d) Experiment with making the trend exponential and/or damped.

   (e) Compare the RMSE of the one-step forecasts from the various methods. Which do you prefer?

   (f) Check that the residuals from the best model look like white noise.

3. Forecast one of the series considered in the previous session using an exponential smoothing method. Try to find the best trend and seasonal specification for the series. Check if the residuals resemble white noise.

3.6 Taxonomy of exponential smoothing methods

Trend Component		**Seasonal Component** N (None)	 A (Additive)	 M (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A_d	(Additive damped)	A_d,N	A_d,A	A_d,M
M	(Multiplicative)	M,N	M,A	M,M
M_d	(Multiplicative damped)	M_d,N	M_d,A	M_d,M

There are 15 separate exponential smoothing methods.

3.7 Innovations state space models

- Generate same point forecasts but can also generate forecast intervals.
- A stochastic (or random) data generating process that can generate an entire forecast distribution.
- Allow for "proper" model selection.

ETS models

- Each model has an *observation* equation and *transition* equations, one for each state (level, trend, seasonal), i.e., state space models.
- Two models for each method: one with additive and one with multiplicative errors, i.e., in total 30 models.
- ETS(Error,Trend,Seasonal):
  - Error= $\{A, M\}$
  - Trend = $\{N, A, A_d, M, M_d\}$
  - Seasonal = $\{N, A, M\}$.
- All ETS models can be written in innovations state space form.
- Additive and multiplicative versions give the same point forecasts but different Forecast intervals.

ETS(A,N,N)

$$\begin{aligned} \text{Observation equation} \qquad & y_t = \ell_{t-1} + \varepsilon_t, \\ \text{State equation} \qquad & \ell_t = \ell_{t-1} + \alpha\varepsilon_t \end{aligned}$$

- $e_t = y_t - \hat{y}_{t|t-1} = \varepsilon_t$
- Assume $\varepsilon_t \sim \text{NID}(0,\sigma^2)$
- "innovations" or "single source of error" because same error process, ε_t.

ETS(M,N,N)

- Specify relative errors $\varepsilon_t = \frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}} \sim \text{NID}(0,\sigma^2)$
- Substituting $\hat{y}_{t|t-1} = \ell_{t-1}$ gives:
  - $y_t = \ell_{t-1} + \ell_{t-1}\varepsilon_t$
  - $e_t = y_t - \hat{y}_{t|t-1} = \ell_{t-1}\varepsilon_t$

$$\begin{aligned} \text{Observation equation} \qquad & y_t = \ell_{t-1}(1+\varepsilon_t) \\ \text{State equation} \qquad & \ell_t = \ell_{t-1}(1+\alpha\varepsilon_t) \end{aligned}$$

- Models with additive and multiplicative errors with the same parameters generate the same point forecasts but different Forecast intervals.

ETS(A,A,N)

$$\begin{aligned} y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t \end{aligned}$$

ETS(M,A,N)

$$\begin{aligned} y_t &= (\ell_{t-1} + b_{t-1})(1+\varepsilon_t) \\ \ell_t &= (\ell_{t-1} + b_{t-1})(1+\alpha\varepsilon_t) \\ b_t &= b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t \end{aligned}$$

ETS(A,A,A)

$$\begin{aligned} \text{Forecast equation} \qquad & \hat{y}_{t+h|t} = \ell_t + hb_t + s_{t-m+h_m^+} \\ \text{Observation equation} \qquad & y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\ \text{State equations} \qquad & \ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ & b_t = b_{t-1} + \beta\varepsilon_t \\ & s_t = s_{t-m} + \gamma\varepsilon_t \end{aligned}$$

- Forecast errors: $\varepsilon_t = y_t - \hat{y}_{t|t-1}$
- $h_m^+ = \lfloor (h-1) \mod m \rfloor + 1.$

Additive error models

Trend	Seasonal		
	N	A	M
N	$y_t = \ell_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$	$y_t = \ell_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/\ell_{t-1}$
A	$y_t = \ell_{t-1} + b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + b_{t-1})$
A$_\mathrm{d}$	$y_t = \ell_{t-1} + \phi b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + \phi b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = \phi b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + \phi b_{t-1})$
M	$y_t = \ell_{t-1}b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t/\ell_{t-1}$	$y_t = \ell_{t-1}b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t/(s_{t-m}\ell_{t-1})$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1}b_{t-1})$
M$_\mathrm{d}$	$y_t = \ell_{t-1}b_{t-1}^{\phi} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/\ell_{t-1}$	$y_t = \ell_{t-1}b_{t-1}^{\phi} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}^{\phi}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/(s_{t-m}\ell_{t-1})$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1}b_{t-1}^{\phi})$

Multiplicative error models

Trend	Seasonal		
	N	A	M
N	$y_t = \ell_{t-1}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$	$y_t = (\ell_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \alpha(\ell_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A	$y_t = (\ell_{t-1} + b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A$_\mathrm{d}$	$y_t = (\ell_{t-1} + \phi b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
M	$y_t = \ell_{t-1}b_{t-1}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}(1 + \beta\varepsilon_t)$	$y_t = (\ell_{t-1}b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}(1 + \beta\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
M$_\mathrm{d}$	$y_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}^{\phi}(1 + \beta\varepsilon_t)$	$y_t = (\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}^{\phi}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}^{\phi}(1 + \beta\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$

Innovations state space models

Let $\boldsymbol{x}_t = (\ell_t, b_t, s_t, s_{t-1}, \ldots, s_{t-m+1})$ and $\varepsilon_t \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$.

$$
\begin{aligned}
y_t &= \underbrace{h(\boldsymbol{x}_{t-1})}_{\mu_t} + \underbrace{k(\boldsymbol{x}_{t-1})\varepsilon_t}_{e_t} \\
\boldsymbol{x}_t &= f(\boldsymbol{x}_{t-1}) + g(\boldsymbol{x}_{t-1})\varepsilon_t
\end{aligned}
$$

Additive errors:

$k(x) = 1. \qquad y_t = \mu_t + \varepsilon_t.$

Multiplicative errors:

$k(\boldsymbol{x}_{t-1}) = \mu_t. \qquad y_t = \mu_t(1 + \varepsilon_t).$

$\varepsilon_t = (y_t - \mu_t)/\mu_t$ is relative error.

- All the methods can be written in this state space form.
- The only difference between the additive error and multiplicative error models is in the observation equation.
- Additive and multiplicative versions give the same point forecasts.

Some unstable models

- Some of the combinations of (Error, Trend, Seasonal) can lead to numerical difficulties; see equations with division by a state.
- These are: ETS(M,M,A), ETS(M,M$_\text{d}$,A), ETS(A,N,M), ETS(A,A,M), ETS(A,A$_\text{d}$,M), ETS(A,M,N), ETS(A,M,A), ETS(A,M,M), ETS(A,M$_\text{d}$,N), ETS(A,M$_\text{d}$,A), and ETS(A,M$_\text{d}$,M).
- Models with multiplicative errors are useful for strictly positive data – but are not numerically stable with data containing zeros or negative values. In that case only the six fully additive models will be applied.

		Seasonal Component		
Trend		N	A	M
Component		(None)	(Additive)	(Multiplicative)
N	(None)	A,N,N	A,N,A	A~~,N,M~~
A	(Additive)	A,A,N	A,A,A	A~~,A,M~~
A$_\text{d}$	(Additive damped)	A,A$_\text{d}$,N	A,A$_\text{d}$,A	A,~~A$_\text{d}$,M~~
M	(Multiplicative)	A~~,M,N~~	A~~,M,A~~	A~~,M,M~~
M$_\text{d}$	(Multiplicative damped)	A,~~M$_\text{d}$,N~~	A,~~M$_\text{d}$,A~~	A,~~M$_\text{d}$,M~~

		Seasonal Component		
Trend		N	A	M
Component		(None)	(Additive)	(Multiplicative)
N	(None)	M,N,N	M,N,A	M,N,M
A	(Additive)	M,A,N	M,A,A	M,A,M
A$_\text{d}$	(Additive damped)	M,A$_\text{d}$,N	M,A$_\text{d}$,A	M,A$_\text{d}$,M
M	(Multiplicative)	M,M,N	M~~,M,A~~	M,M,M
M$_\text{d}$	(Multiplicative damped)	M,M$_\text{d}$,N	M,~~M$_\text{d}$,A~~	M,M$_\text{d}$,M

Estimation

$$\begin{aligned} L^*(\boldsymbol{\theta},\boldsymbol{x}_0) &= n\log\left(\sum_{t=1}^{n}\varepsilon_t^2/k^2(\boldsymbol{x}_{t-1})\right) + 2\sum_{t=1}^{n}\log|k(\boldsymbol{x}_{t-1})| \\ &= -2\log(\text{Likelihood}) + \text{constant} \end{aligned}$$

- Estimate parameters $\boldsymbol{\theta} = (\alpha,\beta,\gamma,\phi)$ and initial states $\boldsymbol{x}_0 = (\ell_0,b_0,s_0,s_{-1},\dots,s_{-m+1})$ by minimizing L^*.

Parameter restrictions

Usual region

- Traditional restrictions in the methods $0 < \alpha,\beta^*,\gamma^*,\phi < 1$ — equations interpreted as weighted averages.
- In models we set $\beta = \alpha\beta^*$ and $\gamma = (1-\alpha)\gamma^*$ therefore $0 < \alpha < 1,\ 0 < \beta < \alpha$ and $0 < \gamma < 1-\alpha$.
- $0.8 < \phi < 0.98$ — to prevent numerical difficulties.

Admissible region

- To prevent observations in the distant past having a continuing effect on current forecasts.
- Usually (but not always) less restrictive than the *usual* region.
- For example for ETS(A,N,N):
  usual $0 < \alpha < 1$ — *admissible* is $0 < \alpha < 2$.

Model selection

Akaike's Information Criterion

$$\text{AIC} = -2\log(\text{Likelihood}) + 2p$$

where p is the number of estimated parameters in the model. *Minimizing* the AIC gives the best model for prediction.

AIC corrected (for small sample bias)

$$\text{AIC}_\text{C} = \text{AIC} + \frac{2(p+1)(p+2)}{n-p}$$

Schwartz' Bayesian IC

$$\text{BIC} = \text{AIC} + p(\log(n) - 2)$$

Akaike's Information Criterion

- Value of AIC/AICc/BIC given in the R output.
- AIC does not have much meaning by itself. Only useful in comparison to AIC value for another model fitted to *same data set.*
- Consider several models with AIC values close to the minimum.
- A difference in AIC values of 2 or less is not regarded as substantial and you may choose the simpler but non-optimal model.
- AIC can be negative.

Automatic forecasting

From Hyndman et al. (IJF, 2002):

- Apply each model that is appropriate to the data. Optimize parameters and initial values using MLE (or some other criterion).
- Select best method using AICc:
- Produce forecasts using best method.
- Obtain Forecast intervals using underlying state space model.

Method performed very well in M3 competition.

3.8 ETS in R

```
fit <- ets(ausbeer)
fit2 <- ets(ausbeer,model="AAA",damped=FALSE)
fcast1 <- forecast(fit, h=20)
fcast2 <- forecast(fit2, h=20)
```

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL,
    beta=NULL, gamma=NULL, phi=NULL,
    additive.only=FALSE,
    lower=c(rep(0.0001,3),0.80),
    upper=c(rep(0.9999,3),0.98),
    opt.crit=c("lik","amse","mse","sigma"), nmse=3,
    bounds=c("both","usual","admissible"),
    ic=c("aic","aicc","bic"), restrict=TRUE)
```

```
> fit
ETS(M,Md,M)

  Smoothing parameters:
    alpha = 0.1776
    beta  = 0.0454
    gamma = 0.1947
    phi   = 0.9549

  Initial states:
    l = 263.8531
    b = 0.9997
    s = 1.1856 0.9109 0.8612 1.0423

  sigma:  0.0356

     AIC     AICc      BIC
2272.549 2273.444 2302.715
```

```
> fit2
ETS(A,A,A)

  Smoothing parameters:
    alpha = 0.2079
    beta  = 0.0304
    gamma = 0.2483

  Initial states:
    l = 255.6559
    b = 0.5687
    s = 52.3841 -27.1061 -37.6758 12.3978

  sigma:  15.9053

     AIC     AICc      BIC
2312.768 2313.481 2339.583
```

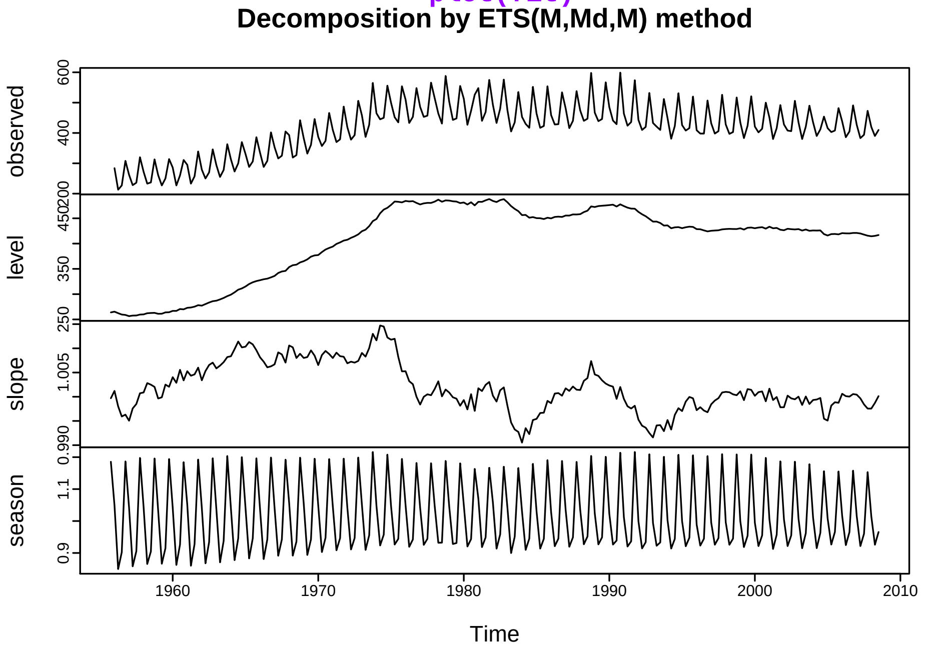
`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces Forecast intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class "ets".

`ets` objects

- **Methods:** "coef()", "plot()", "summary()", "residuals()", "fitted()", "simulate()" and "forecast()"
- "plot()" function shows time plots of the original time series along with the extracted components (level, growth and seasonal).

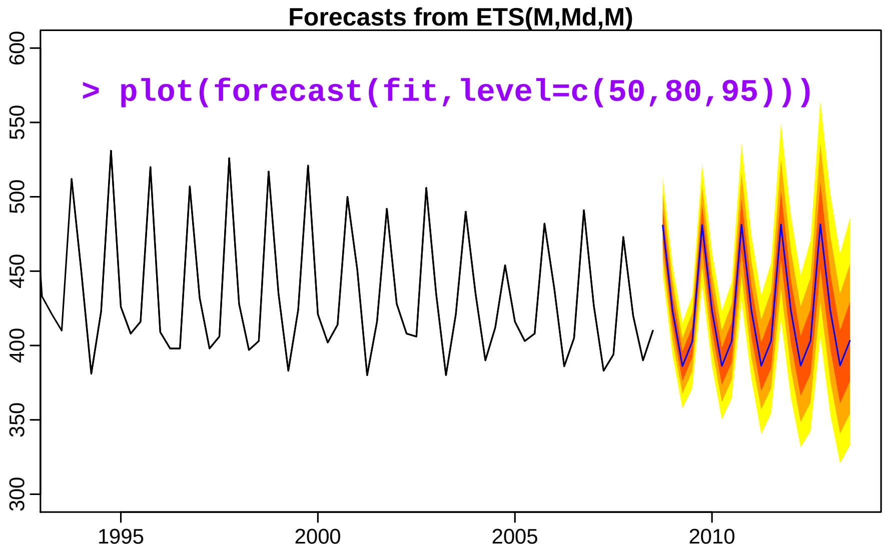
```
plot(fit)
```



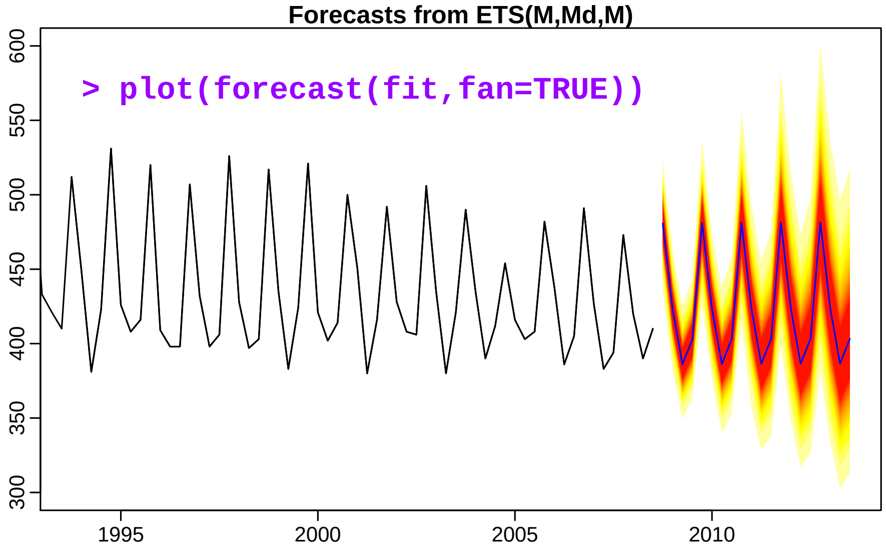
Goodness-of-fit

```
> accuracy(fit)
      ME      RMSE       MAE       MPE      MAPE      MASE
 0.17847 15.48781 11.77800  0.07204  2.81921  0.20705

> accuracy(fit2)
      ME      RMSE       MAE       MPE      MAPE      MASE
-0.11711 15.90526 12.18930 -0.03765  2.91255  0.21428
```

Forecast intervals



```
> plot(forecast(fit,level=c(50,80,95)))
```



```
> plot(forecast(fit,fan=TRUE))
```

Re-fitting ETS models

"ets()" function also allows refitting model to new data set.

```
> usfit <- ets(usnetelec[1:45])
> test <- ets(usnetelec[46:55], model = usfit)

> accuracy(test)
      ME      RMSE       MAE       MPE      MAPE      MASE
-3.35419 58.02763 43.85545 -0.07624  1.18483  0.52452

> accuracy(forecast(usfit,10), usnetelec[46:55])
        ME       RMSE        MAE        MPE       MAPE       MASE
  40.7034   61.2075   46.3246     1.0980    1.2620    0.6776
```

The ets() function in R

```
ets(y, model="ZZZ", damped=NULL,
    alpha=NULL, beta=NULL,
    gamma=NULL, phi=NULL,
    additive.only=FALSE,
    lambda=NULL
    lower=c(rep(0.0001,3),0.80),
    upper=c(rep(0.9999,3),0.98),
    opt.crit=c("lik","amse","mse","sigma"),
    nmse=3,
    bounds=c("both","usual","admissible"),
    ic=c("aic","aicc","bic"), restrict=TRUE)
```

- `y`
  The time series to be forecast.
- `model`
  use the ETS classification and notation: “N” for none, “A” for additive, “M” for multiplicative, or “Z” for automatic selection. Default ZZZ all components are selected using the information criterion.
- `damped`
  - If `damped=TRUE`, then a damped trend will be used (either A_d or M_d).
  - `damped=FALSE`, then a non-damped trend will used.
  - If `damped=NULL` (the default), then either a damped or a non-damped trend will be selected according to the information criterion chosen.
- `alpha, beta, gamma, phi`
  The values of the smoothing parameters can be specified using these arguments. If they are set to `NULL` (the default value for each of them), the parameters are estimated.
- `additive.only`
  Only models with additive components will be considered if `additive.only=TRUE`. Otherwise all models will be considered.
- `lambda`
  Box-Cox transformation parameter. It will be ignored if `lambda=NULL` (the default value). Otherwise, the time series will be transformed before the model is estimated. When `lambda` is not `NULL`, `additive.only` is set to `TRUE`.
- `lower,upper` bounds for the parameter estimates of α, β, γ and ϕ.

- `opt.crit=lik` (default) optimisation criterion used for estimation.
- `bounds` Constraints on the parameters.
  - *usual* region – "`bounds=usual`";
  - *admissible* region – "`bounds=admissible`";
  - "`bounds=both`" (the default) requires the parameters to satisfy both sets of constraints.
- `ic=aic` (the default) information criterion to be used in selecting models.
- `restrict=TRUE` (the default) models that cause numerical difficulties are not considered in model selection.

3.9 Forecasting with ETS models

- Point forecasts obtained by iterating equations for $t = T + 1,\dots,T + h$, setting $\varepsilon_t = 0$ for $t > T$.
- Not the same as $\mathrm{E}(y_{t+h}|\boldsymbol{x}_t)$ unless trend and seasonality are both additive.
- Point forecasts for ETS(A,x,y) are identical to ETS(M,x,y) if the parameters are the same.
- Forecast intervals will differ between models with additive and multiplicative methods.
- Exact PI available for many models.
- Otherwise, simulate future sample paths, conditional on last estimate of states, and obtain PI from percentiles of simulated paths.

Point forecasts: iterate the equations for $t = T + 1, T + 2,\dots,T + h$ and set all $\varepsilon_t = 0$ for $t > T$.

For example, for ETS(M,A,N):

- $y_{T+1} = (\ell_T + b_T)(1 + \varepsilon_{T+1})$
- Therefore $\hat{y}_{T+1|T} = \ell_T + b_T$
- $y_{T+2} = (\ell_{T+1} + b_{T+1})(1 + \varepsilon_{T+1}) = [(\ell_T + b_T)(1 + \alpha\varepsilon_{T+1}) + b_T + \beta(\ell_T + b_T)\varepsilon_{T+1}](1 + \varepsilon_{T+1})$
- Therefore $\hat{y}_{T+2|T} = \ell_T + 2b_T$ and so on.

Identical forecast with Holt's linear method and ETS(A,A,N). So the point forecasts obtained from the method and from the two models that underly the method are identical (assuming the same parameter values are used).

Forecast intervals: cannot be generated using the methods.

- The Forecast intervals will differ between models with additive and multiplicative methods.
- Exact formulae for some models.
- More general to simulate future sample paths, conditional on the last estimate of the states, and to obtain Forecast intervals from the percentiles of these simulated future paths.
- Options are available in R using the `forecast` function in the forecast package.

3.10 Lab Session 4

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Use `ets()` to find the best ETS model for the price of eggs (data set `eggs`). How does this model compare to the one you found in the previous lab session?

2. Use `ets()` on the various other series we have considered today. Does it always give good forecasts? Find an example where it does not work well. Can you figure out why?

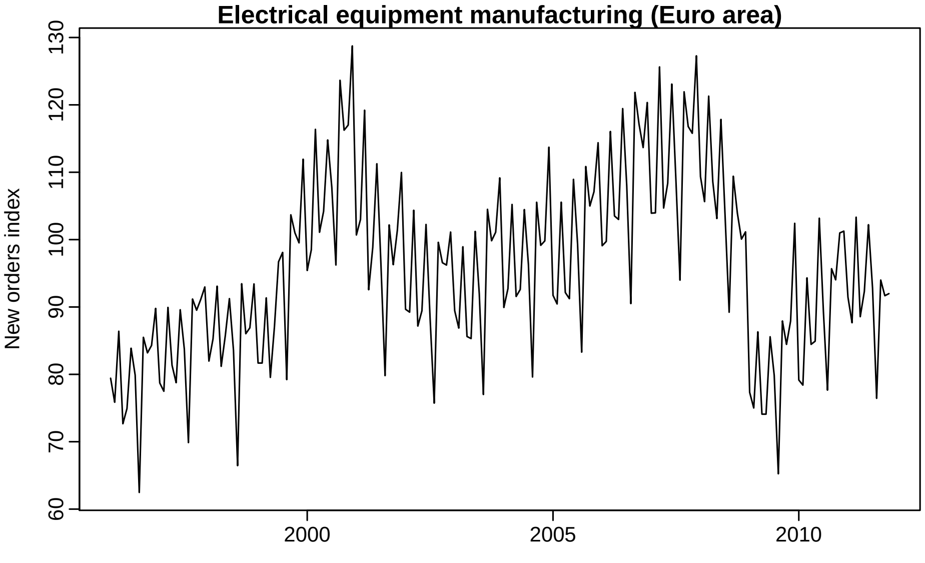
4 Time series decomposition

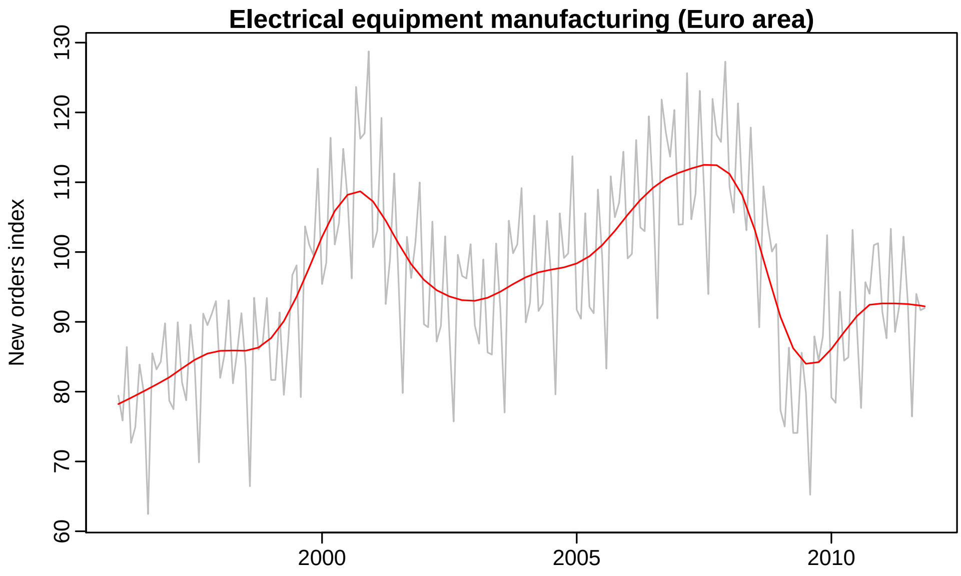
$$Y_t = f(S_t, T_t, E_t)$$

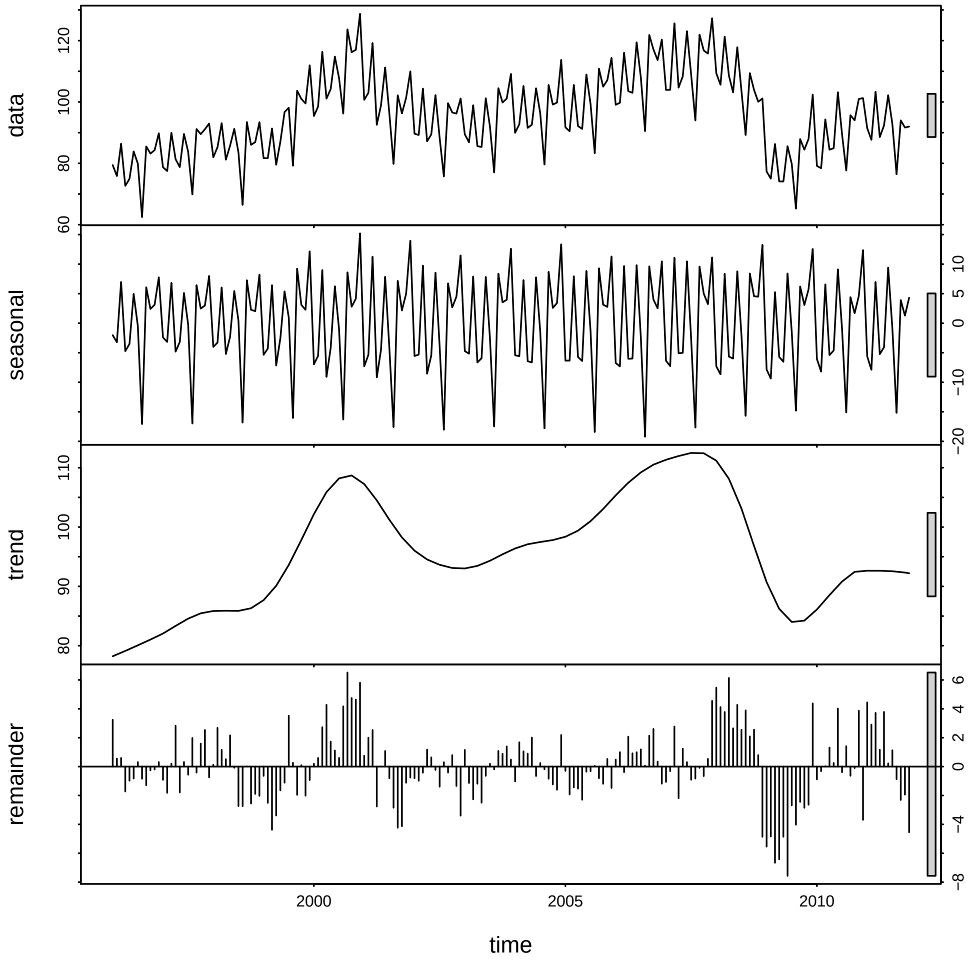
where $Y_t =$ data at period t
$S_t =$ seasonal component at period t
$T_t =$ trend component at period t
$E_t =$ remainder (or irregular or error) component at period t

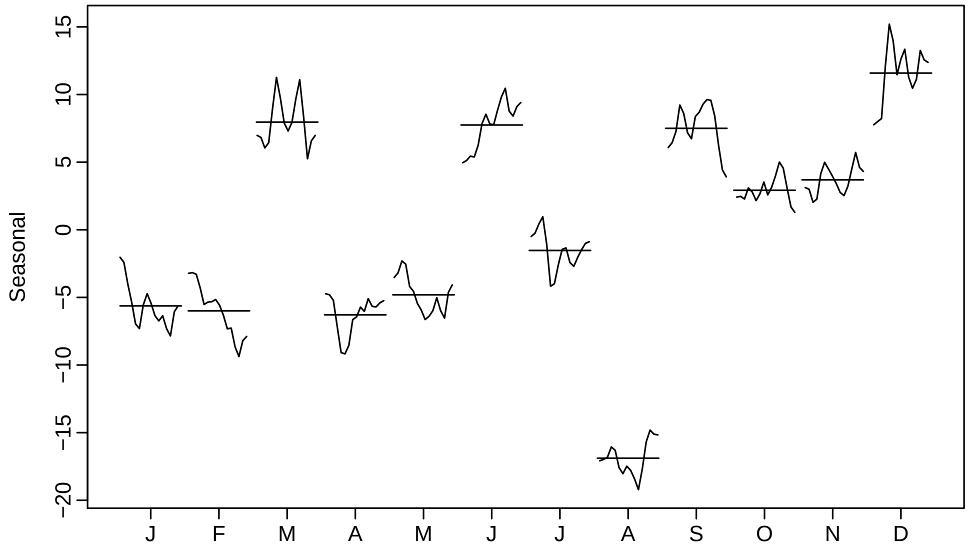
Additive decomposition: $Y_t = S_t + T_t + E_t$.

4.1 Example: Euro electrical equipment




Electrical equipment manufacturing (Euro area)
New orders index
60
70
80
90
100
110
120
130
2000
2005
2010

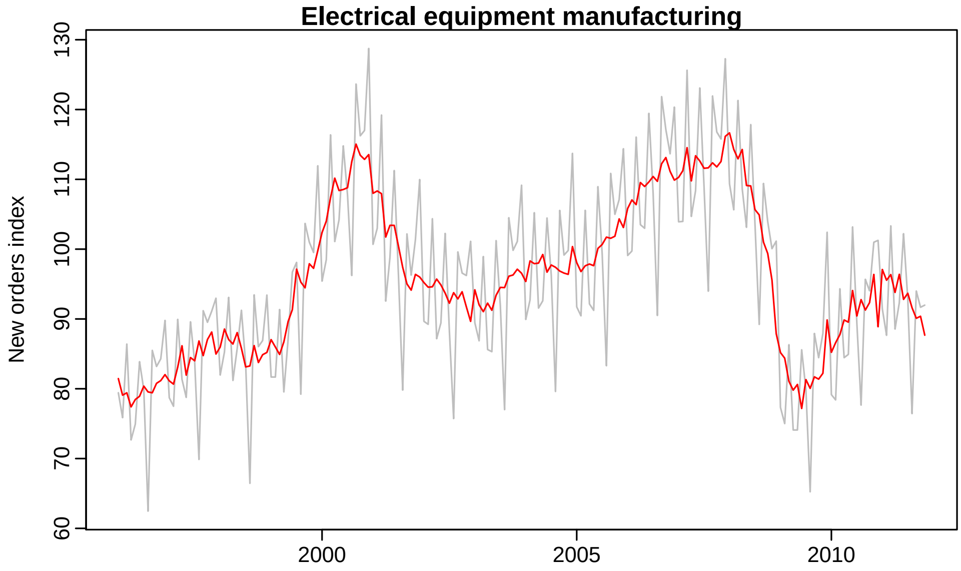

data
60
80
100
120
seasonal
−20
−10
0
5
10
trend
80
90
100
110
remainder
−8
−4
0
2
4
6
2000
2005
2010
time



4.2 Seasonal adjustment

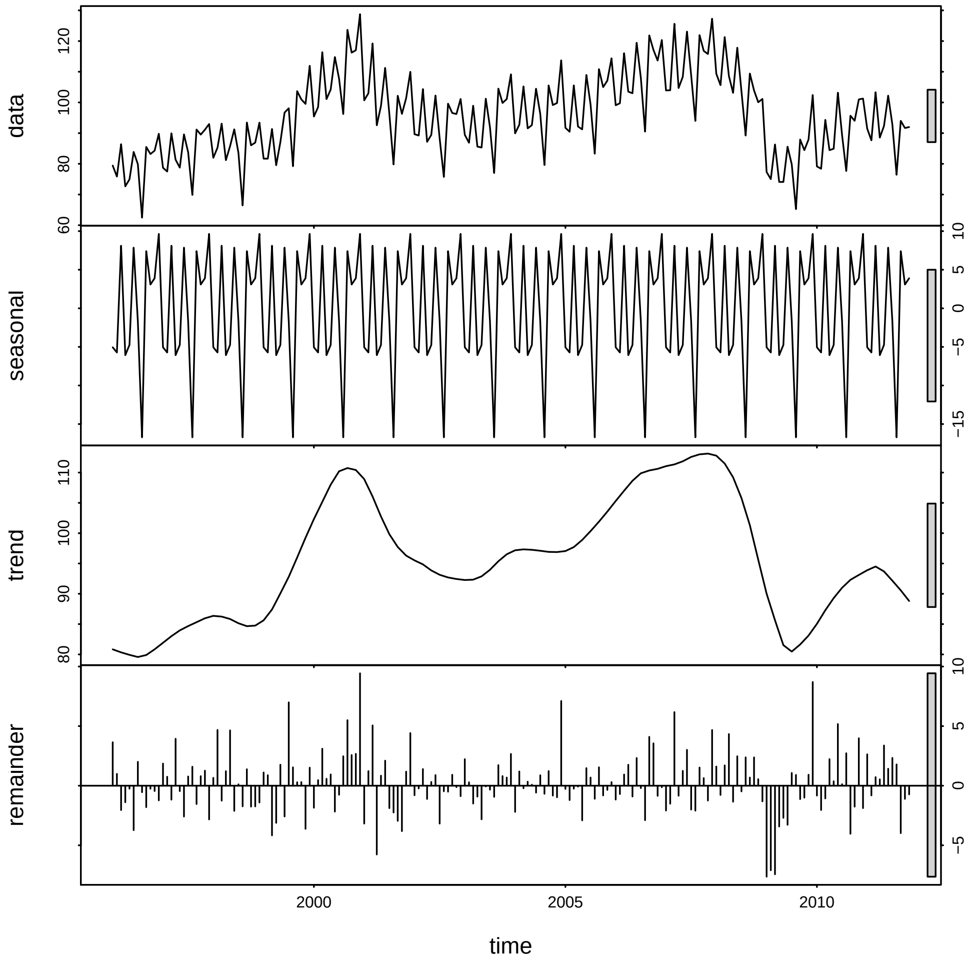
- Useful by-product of decomposition: an easy way to calculate seasonally adjusted data.
- Additive decomposition: seasonally adjusted data given by

$$Y_t - S_t = T_t + E_t$$



4.3 STL decomposition

- STL: "Seasonal and Trend decomposition using Loess",
- Very versatile and robust.
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Robust to outliers
- Only additive.
- Use Box-Cox transformations to get other decompositions.

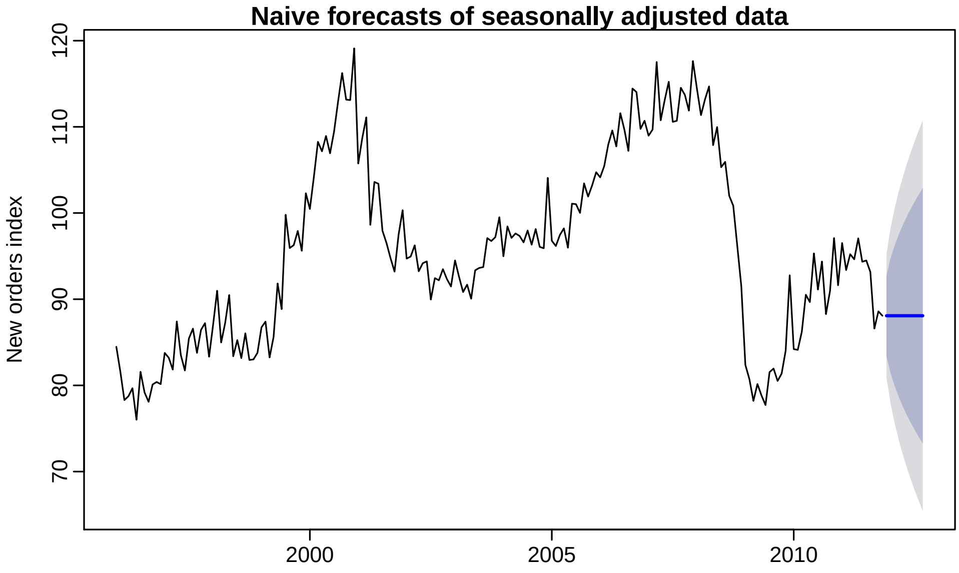


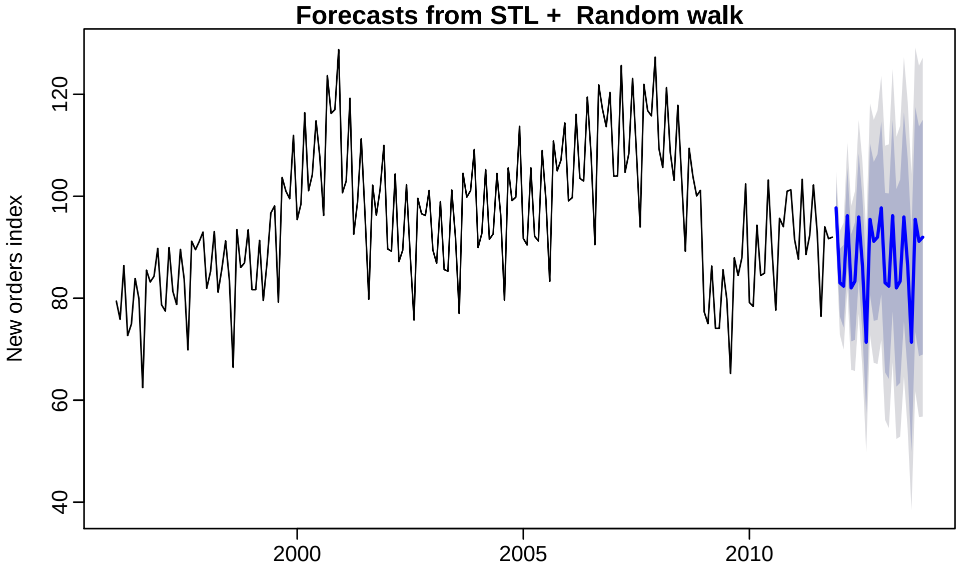
```
fit <- stl(elecequip, t.window=15, s.window="periodic", robust=TRUE)
plot(fit)
```

- `t.window` controls wiggliness of trend component.
- `s.window` controls variation on seasonal component.

4.4 Forecasting and decomposition

- Forecast seasonal component by repeating the last year
- Forecast seasonally adjusted data using non-seasonal time series method. E.g., ETS model.
- Combine forecasts of seasonal component with forecasts of seasonally adjusted data to get forecasts of original data.
- Sometimes a decomposition is useful just for understanding the data before building a separate forecasting model.





How to do this in R

```
fit <- stl(elecequip, t.window=15, s.window="periodic", robust=TRUE)

eeadj <- seasadj(fit)
plot(naive(eeadj), xlab="New orders index")

fcast <- forecast(fit, method="naive")
plot(fcast, ylab="New orders index")
```

Decomposition and forecast intervals

- It is common to take the Forecast intervals from the seasonally adjusted forecasts and modify them with the seasonal component.
- This ignores the uncertainty in the seasonal component estimate.
- It also ignores the uncertainty in the future seasonal pattern.

4.5 Lab Session 5a

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Consider the monthly sales of product A for a plastics manufacturer for years 1 through 5 (data set `plastics`).

   (a) Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend?

   (b) Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality.)

   (c) Do the results support the graphical interpretation from part (a)?

   (d) Compute and plot the seasonally adjusted data.

   (e) Use a random walk to produce forecasts of the seasonally adjusted data.

   (f) Reseasonalize the results to give forecasts on the original scale. [Hint: you can use the `stlf` function with `method="naive"`.]

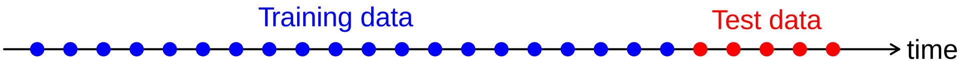
   (g) Why do the forecasts look too low?

5

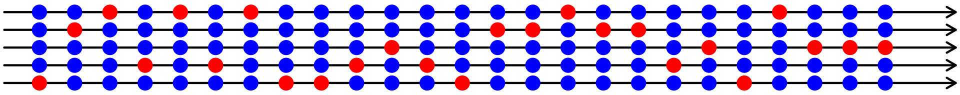
Time series cross-validation

5.1 Cross-validation

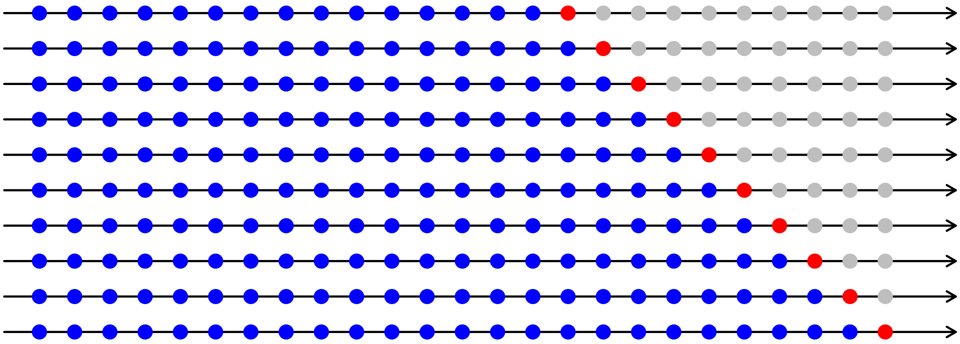
Traditional evaluation



Standard cross-validation



Time series cross-validation



Assume k is the minimum number of observations for a training set.

- Select observation $k+i$ for test set, and use observations at times $1,2,\ldots,k+i-1$ to estimate model. Compute error on forecast for time $k+i$.
- Repeat for $i=0,1,\ldots,T-k$ where T is total number of observations.
- Compute accuracy measure over all errors.

Also called **rolling forecasting origin** because the origin $(k+i-1)$ at which forecast is based rolls forward in time.

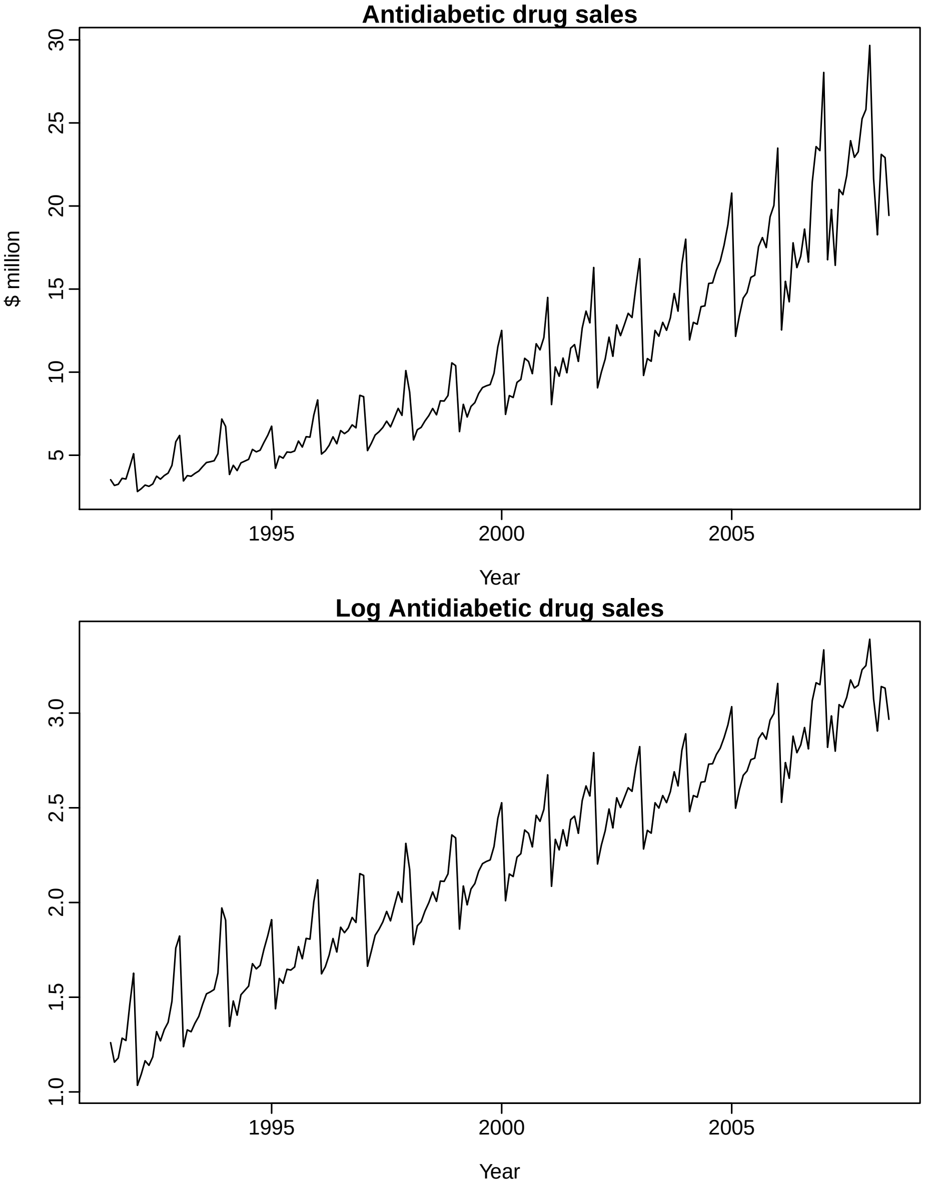
Cross-sectional data

- Minimizing the AIC is asymptotically equivalent to minimizing MSE via leave-one-out cross-validation. (Stone, 1977).

Time series cross-validation

- Minimizing the AIC is asymptotically equivalent to minimizing MSE via one-step cross-validation. (Akaike, 1969,1973).

5.2 Example: Pharmaceutical sales

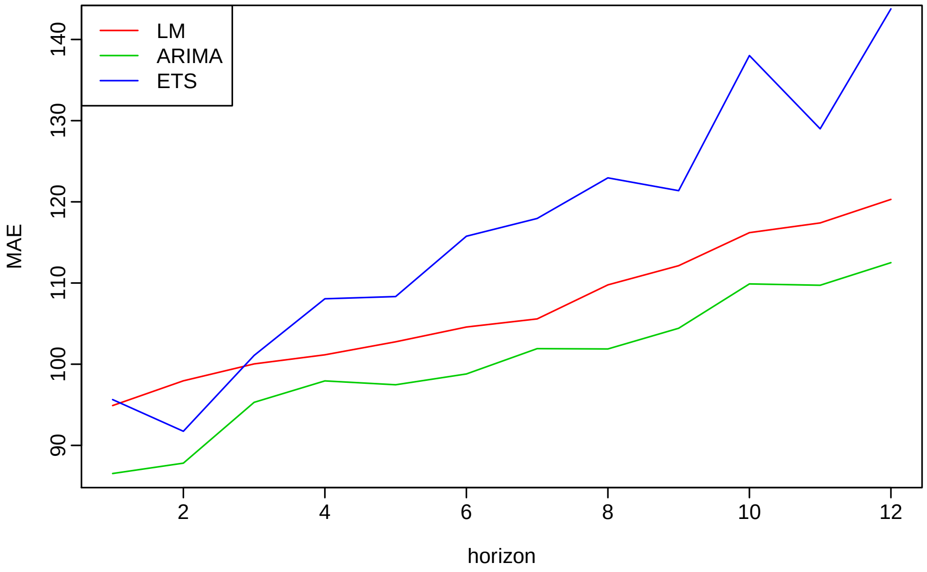


Which of these models is best?

1. Linear model with trend and seasonal dummies applied to log data.
2. ARIMA model applied to log data
3. ETS model applied to original data

- Set $k = 48$ as minimum training set.
- Forecast 12 steps ahead based on data to time $k + i - 1$ for $i = 1,2,\ldots,156$.
- Compare MAE values for each forecast horizon.

```
k <- 48
n <- length(a10)
mae1 <- mae2 <- mae3 <- matrix(NA,n-k-12,12)
for(i in 1:(n-k-12))
{
  xshort <- window(a10,end=1995+(5+i)/12)
  xnext <- window(a10,start=1995+(6+i)/12,end=1996+(5+i)/12)
  fit1 <- tslm(xshort ~ trend + season, lambda=0)
  fcast1 <- forecast(fit1,h=12)
  fit2 <- auto.arima(xshort,D=1, lambda=0)
  fcast2 <- forecast(fit2,h=12)
  fit3 <- ets(xshort)
  fcast3 <- forecast(fit3,h=12)
  mae1[i,] <- abs(fcast1[['mean']]-xnext)
  mae2[i,] <- abs(fcast2[['mean']]-xnext)
  mae3[i,] <- abs(fcast3[['mean']]-xnext)
}
plot(1:12,colMeans(mae1),type="l",col=2,xlab="horizon",ylab="MAE",
     ylim=c(0.58,1.0))
lines(1:12,colMeans(mae2),type="l",col=3)
lines(1:12,colMeans(mae3),type="l",col=4)
legend("topleft",legend=c("LM","ARIMA","ETS"),col=2:4,lty=1)
```



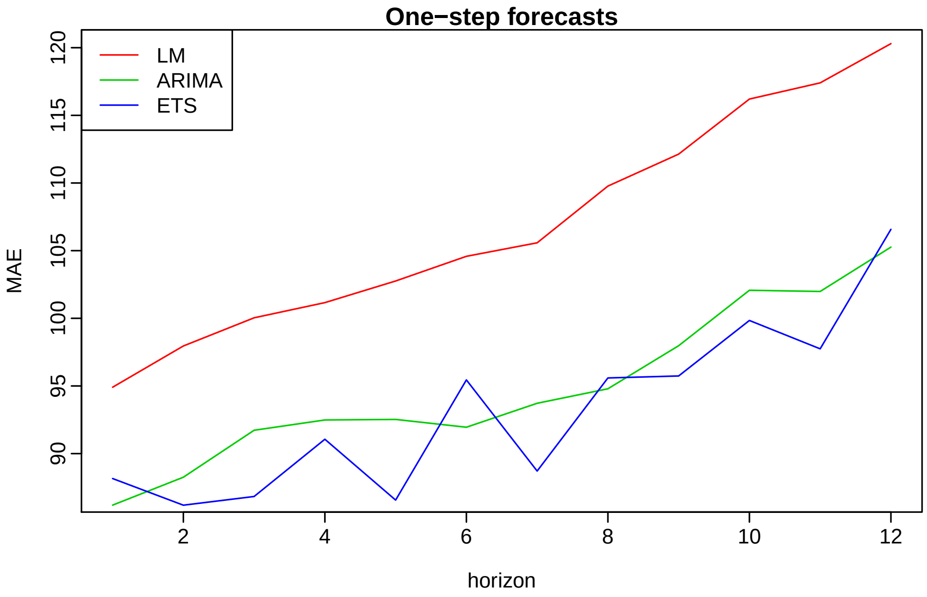
Variations on time series cross validation

Keep training window of fixed length.

```
xshort <- window(a10,start=i+1/12,end=1995+(5+i)/12)
```

Compute one-step forecasts in out-of-sample period.

```
for(i in 1:(n-k))
{
  xshort <- window(a10,end=1995+(5+i)/12)
  xlong <- window(a10,start=1995+(6+i)/12)
  fit2 <- auto.arima(xshort,D=1, lambda=0)
  fit2a <- Arima(xlong,model=fit2)
  fit3 <- ets(xshort)
  fit3a <- ets(xlong,model=fit3)
  mae2a[i,] <- abs(residuals(fit3a))
  mae3a[i,] <- abs(residuals(fit2a))
}
```



5.3 Lab Session 5b

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. For this exercise, use the monthly Australian short-term overseas visitors data, May 1985–April 2005. (Data set: `visitors` in `expsmooth` package.)

   (a) Use `ets` to find the best model for these data and record the training set RMSE. You should find that the best model is ETS(M,A,M).

   (b) We will now check how much larger the one-step RMSE is on out-of-sample data using time series cross-validation. The following code will compute the result, beginning with four years of data in the training set.

   ```
   k <- 48 # minimum size for training set
   n <- length(visitors) # Total number of observations
   e <- visitors*NA # Vector to record one-step forecast errors
   for(i in 48:(n-1))
   {
     train <- ts(visitors[1:i],freq=12)
     fit <- ets(train, "MAM", damped=FALSE)
     fc <- forecast(fit,h=1)$mean
     e[i] <- visitors[i+1]-fc
   }
   sqrt(mean(e^2,na.rm=TRUE))
   ```

   Check that you understand what the code is doing. Ask if you don't.

   (c) What would happen in the above loop if I had set `train <- visitors[1:i]`?

   (d) Plot e. What do you notice about the error variances? Why does this occur?

   (e) How does this problem bias the comparison of the RMSE values from (1a) and (1b)? (Hint: think about the effect of the missing values in e.)

   (f) In practice, we will not know that the best model on the whole data set is ETS(M,A,M) until we observe all the data. So a more realistic analysis would be to allow `ets` to select a different model each time through the loop. Calculate the RMSE using this approach. (Warning: it will take a while as there are a lot of models to fit.)

   (g) How does the RMSE computed in (1f) compare to that computed in (1b)? Does the re-selection of a model at each step make much difference?

2. Try a similar cross-validation approach on one of the other time series considered yesterday.

   (a) Does the `ets()` model selection via AICc give the same model as obtained using cross-validation?

   (b) Which model would you use in practice?

6 Making time series stationary

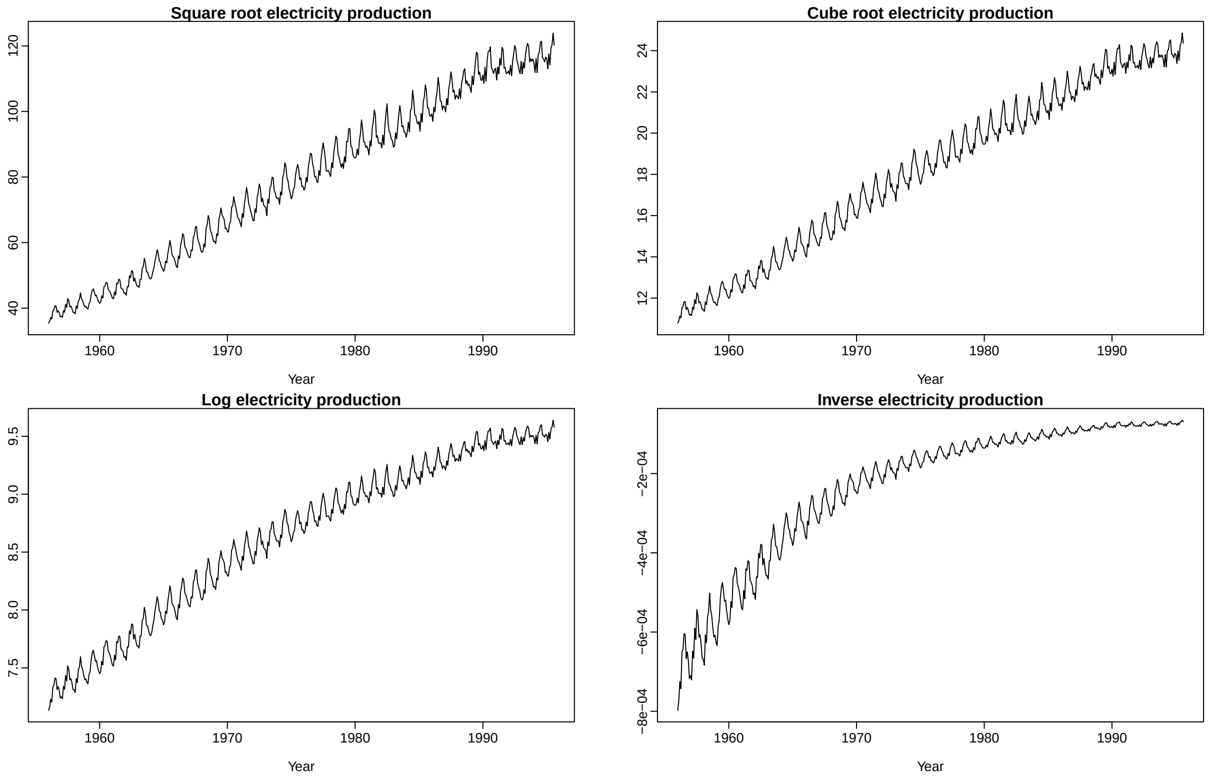
6.1 Transformations

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \ldots, y_n$ and transformed observations as $w_1, \ldots, w_n$.

Square root	$w_t = \sqrt{y_t}$	↓
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale.**



Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

$$y_t = \begin{cases} \exp(w_t), & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda}, & \lambda \neq 0. \end{cases}$$

```
plot(BoxCox(elec,lambda=1/3))
fit <- snaive(elec, lambda=1/3)
plot(fit)
plot(fit, include=120)
```

Automated Box-Cox transformations

```
BoxCox.lambda(elec)
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large Forecast intervals.

ETS and transformations

- A Box-Cox transformation followed by an additive ETS model is often better than an ETS model without transformation.
- It makes no sense to use a Box-Cox transformation and a *non-additive* ETS model.

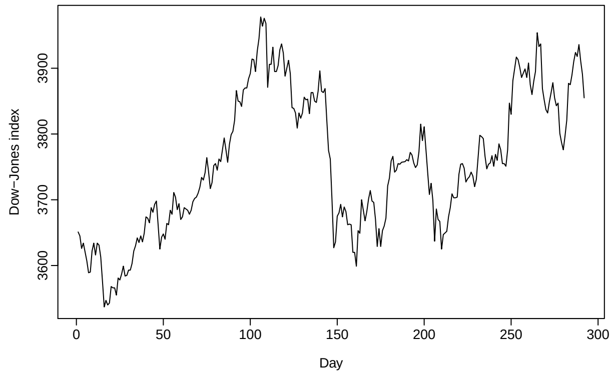
6.2 Stationarity

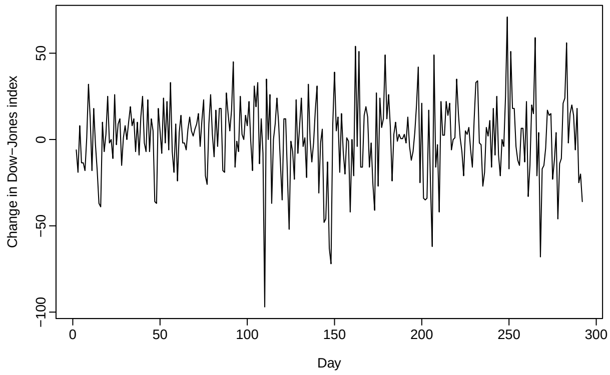
Definition If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \dots, y_{t+s})$ does not depend on t.

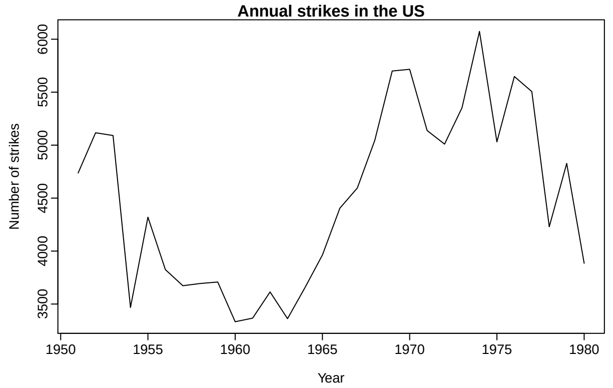
A **stationary series** is:

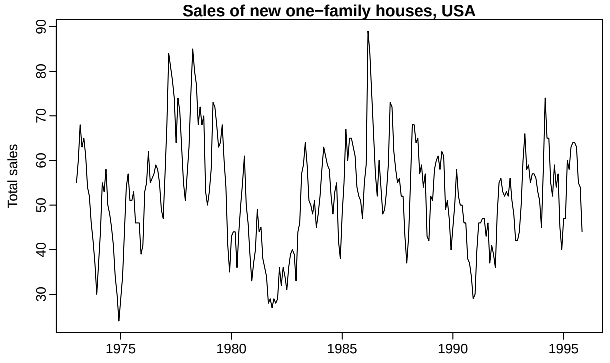
- roughly horizontal
- constant variance
- no patterns predictable in the long-term

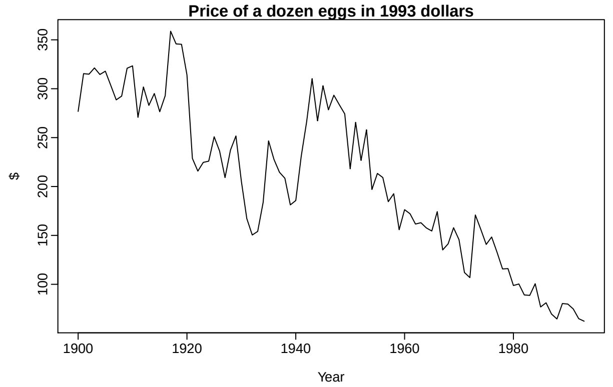
Stationary?

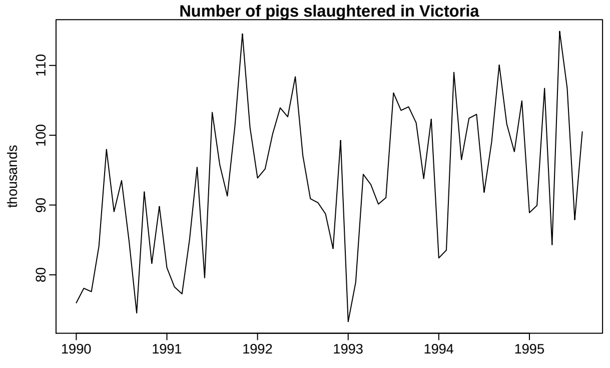


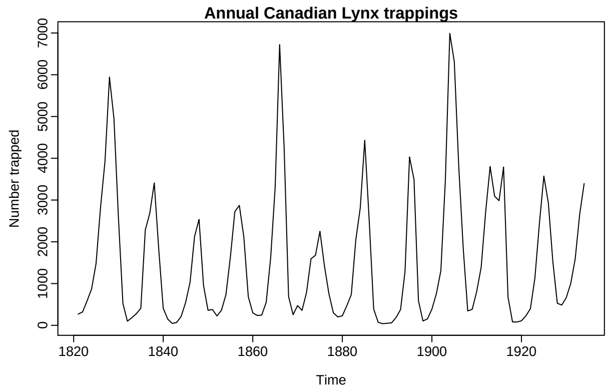


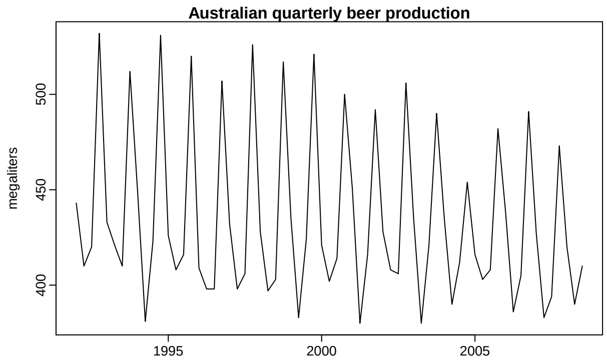












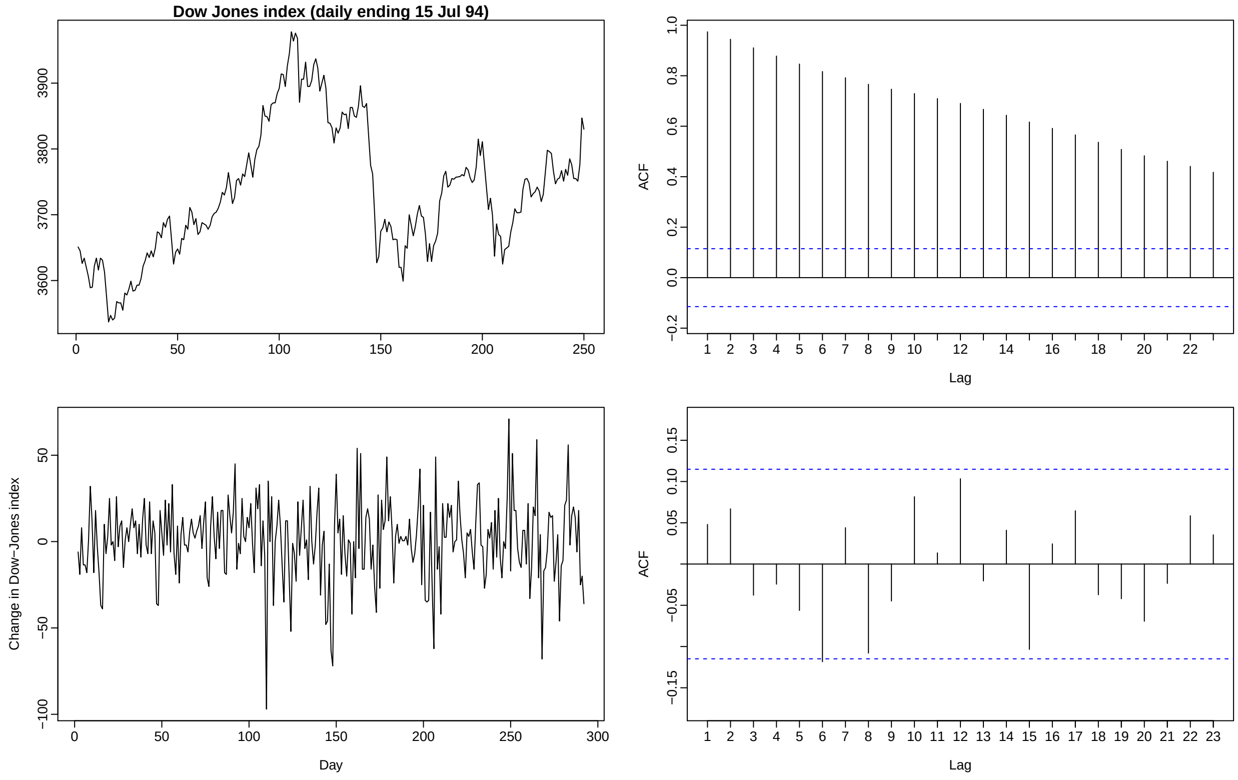
Transformations help to **stabilize the variance**.

For ARIMA modelling, we also need to **stabilize the mean**.

Identifying non-stationary series

- time plot.
- The ACF of stationary data drops to zero relatively quickly
- The ACF of non-stationary data decreases slowly.
- For non-stationary data, the value of r_1 is often large and positive.

Example: Dow-Jones index



6.3 Ordinary differencing

Differencing

- Differencing helps to **stabilize the mean**.
- The differenced series is the *change* between each observation in the original series: $y'_t = y_t - y_{t-1}$.
- The differenced series will have only $T - 1$ values since it is not possible to calculate a difference y'_1 for the first observation.

Example: Dow-Jones index

- The differences of the Dow-Jones index are the **day-today changes**.
- Now the series looks just like **a white noise series**:
  - no autocorrelations outside the 95% limits.
  - Ljung-Box Q^* statistic has a p-value 0.153 for $h = 10$.
- **Conclusion:** The *daily change* in the Dow-Jones index is essentially a random amount uncorrelated with previous days.

Random walk model

Graph of differenced data suggests model for Dow-Jones index:

$$y_t - y_{t-1} = e_t \quad \text{or} \quad y_t = y_{t-1} + e_t\,.$$

- "Random walk" model very widely used for non-stationary data.
- This is the model behind the naïve method.
- Random walks typically have:
  - long periods of apparent trends up or down
  - sudden and unpredictable changes in direction.

Random walk with drift model

$$y_t - y_{t-1} = c + e_t \quad \text{or} \quad y_t = c + y_{t-1} + e_t\,.$$

- c is the average change between consecutive observations.
- This is the model behind the drift method.

Second-order differencing

Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time:

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

- y''_t will have $T - 2$ values.
- In practice, it is almost never necessary to go beyond second-order differences.

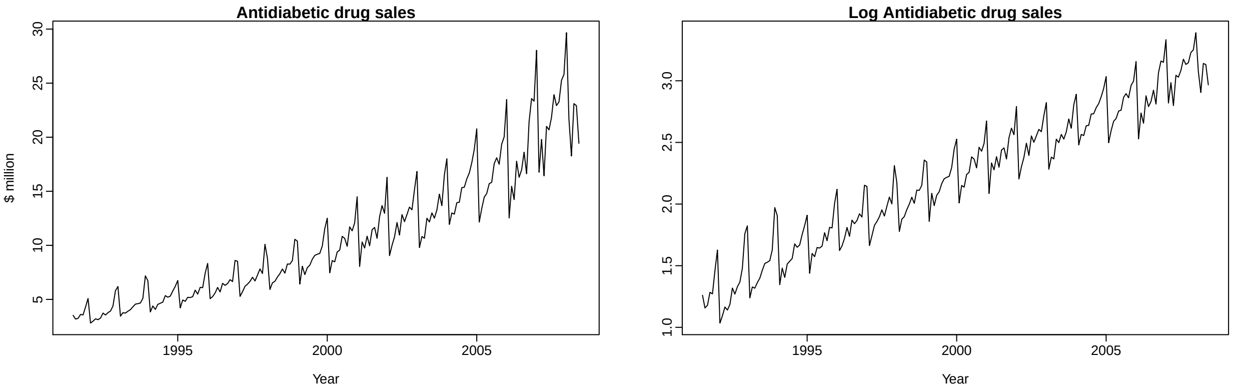
6.4 Seasonal differencing

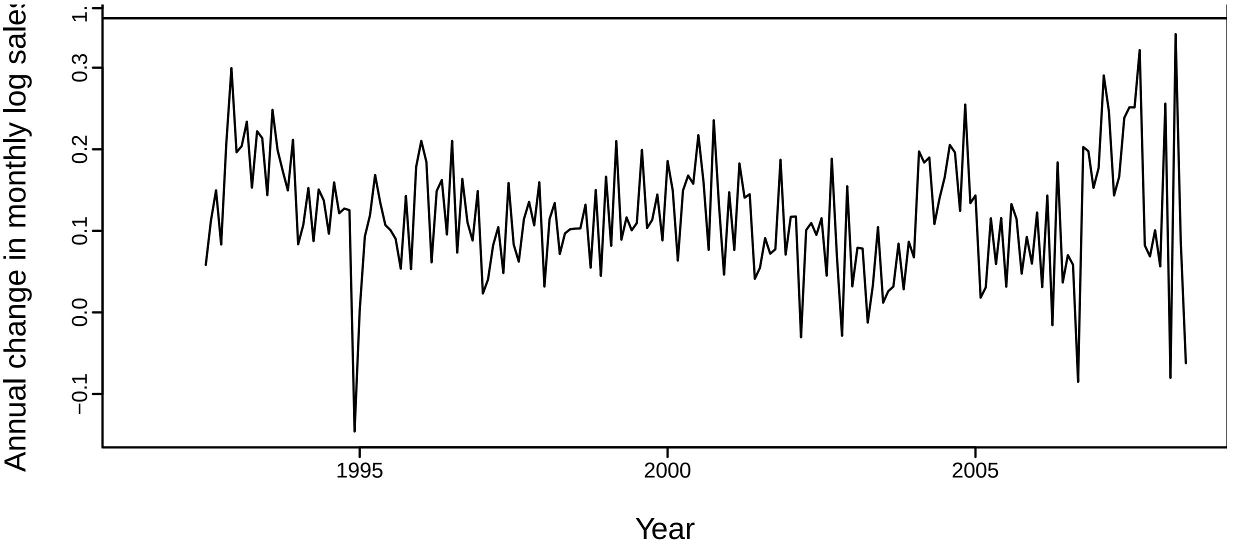
A seasonal difference is the difference between an observation and the corresponding observation from the previous year.

$$y'_t = y_t - y_{t-m}$$

where m = number of seasons. e.g., for monthly data $m = 12$.

Antidiabetic drug sales





- Seasonally differenced series is closer to being stationary.
- Remaining non-stationarity can be removed with further first difference.

If $y'_t = y_t - y_{t-12}$ denotes seasonally differenced series, then twice-differenced series is

$$\begin{aligned} y^*_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-12}) - (y_{t-1} - y_{t-13}) \\ &= y_t - y_{t-1} - y_{t-12} + y_{t-13}\,. \end{aligned}$$

When both seasonal and first differences are applied...

- it makes no difference which is done first—the result will be the same.
- If seasonality is strong, we recommend that seasonal differencing be done first because sometimes the resulting series will be stationary and there will be no need for further first difference.

It is important that if differencing is used, the differences are interpretable.

Interpretation of differencing

- first differences are the change between **one observation and the next**;
- seasonal differences are the change between **one year to the next**.

But taking lag 3 differences for yearly data, for example, results in a model which cannot be sensibly interpreted.

6.5 Unit root tests

Statistical tests to determine the required order of differencing.

1. Augmented Dickey Fuller test: null hypothesis is that the data are non-stationary and non-seasonal.
2. Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test: null hypothesis is that the data are stationary and non-seasonal.
3. Other tests available for seasonal data.

Dickey-Fuller test

- Estimate regression model

  $$y'_t = \phi y_{t-1} + b_1 y'_{t-1} + b_2 y'_{t-2} + \cdots + b_k y'_{t-k}$$

  where y'_t denotes differenced series $y_t - y_{t-1}$.
- Number of lagged terms, k, is usually set to be about 3.
- If original series, y_t, needs differencing, $\hat{\phi} \approx 0$.
- If y_t is already stationary, $\hat{\phi} < 0$.
- In R: Use `adf.test()`.
- Default $k = \lfloor T-1 \rfloor^{1/3}$

```
> adf.test(dj)

Augmented Dickey-Fuller Test

data:  dj
Dickey-Fuller = -1.9872, Lag order = 6, p-value = 0.5816
alternative hypothesis: stationary
```

How many differences?

```
ndiffs(x)
nsdiffs(x)
```

Automated differencing

```
ns <- nsdiffs(x)
if(ns > 0)
  xstar <- diff(x,lag=frequency(x), differences=ns)
else
  xstar <- x
nd <- ndiffs(xstar)
if(nd > 0)
  xstar <- diff(xstar,differences=nd)
```

6.6 Backshift notation

A very useful notational device is the backward shift operator, B, which is used as follows:

$$By_t = y_{t-1}\,.$$

In other words, B, operating on y_t, has the effect of **shifting the data back one period**. Two applications of B to y_t **shifts the data back two periods**:

$$B(By_t) = B^2y_t = y_{t-2}\,.$$

For monthly data, if we wish to shift attention to "the same month last year," then B^{12} is used, and the notation is $B^{12}y_t = y_{t-12}$.

The backward shift operator is convenient for describing the process of *differencing*. A first difference can be written as

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1-B)y_t\,.$$

Note that a first difference is represented by $(1-B)$.

Similarly, if second-order differences (i.e., first differences of first differences) have to be computed, then:

$$y''_t = y_t - 2y_{t-1} + y_{t-2} = (1-B)^2y_t\,.$$

- Second-order difference is denoted $(1-B)^2$.
- *Second-order difference* is not the same as a *second difference*, which would be denoted $1-B^2$;
- In general, a dth-order difference can be written as

$$(1-B)^dy_t.$$

- A seasonal difference followed by a first difference can be written as

$$(1-B)(1-B^m)y_t\,.$$

The "backshift" notation is convenient because the terms can be multiplied together to see the combined effect.

$$\begin{aligned}(1-B)(1-B^m)y_t &= (1-B-B^m+B^{m+1})y_t\\ &= y_t - y_{t-1} - y_{t-m} + y_{t-m-1}.\end{aligned}$$

For monthly data, $m = 12$ and we obtain the same result as earlier.

6.7 Lab Session 6

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. For the following series, find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.
   (a) `usnetelec`
   (b) `usgdp`
   (c) `mcopper`
   (d) `enplanements`
   (e) `visitors`

2. Why is a Box-Cox transformation unhelpful for the `cangas` data?

3. Download the data at `http://robjhyndman.com/data/retail.xls`. Choose *one* of the series and find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.

4. For the same retail data, compare:
   (a) an ETS model;
   (b) an additive ETS model applied to a Box-Cox transformed series;
   (c) an STL model applied to a Box-Cox transformed series, followed by ETS on the seasonally adjusted data;
   (d) a seasonal naive method applied to the Box-Cox transformed series;

   For each model, look at the residual diagnostics and compare the forecasts on a test set of the last two years.

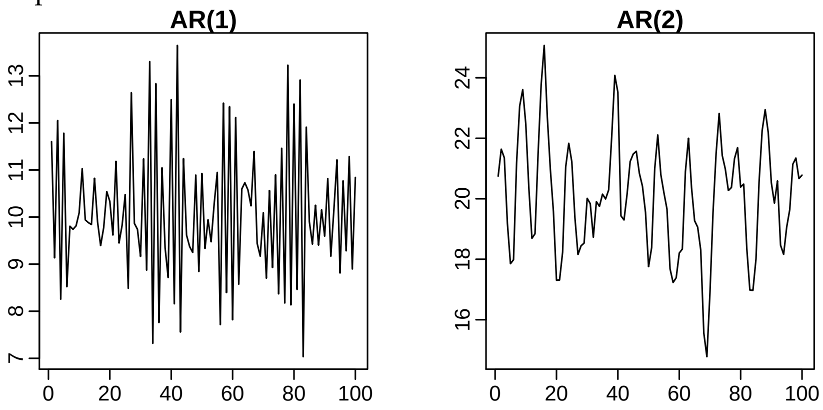
5. Repeat the previous question but use time series cross-validation to compare the four models.

7 Non-seasonal ARIMA models

7.1 Autoregressive models

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t,$$

where e_t is white noise. This is a multiple regression with **lagged values** of y_t as predictors.



AR(1) model

$$y_t = c + \phi_1 y_{t-1} + e_t$$

- When $\phi_1 = 0$, y_t is **equivalent to WN**
- When $\phi_1 = 1$ and $c = 0$, y_t is **equivalent to a RW**
- When $\phi_1 = 1$ and $c \neq 0$, y_t is **equivalent to a RW with drift**
- When $\phi_1 < 0$, y_t tends to **oscillate between positive and negative values**.

Stationarity conditions

We normally restrict autoregressive models to stationary data, and then some constraints on the values of the parameters are required.

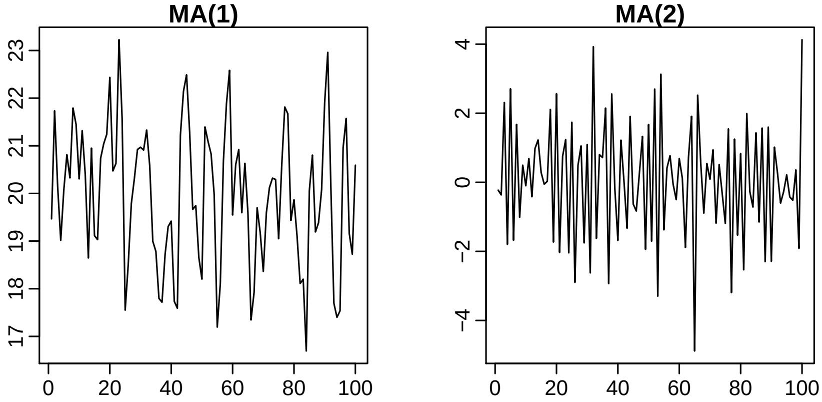
General condition for stationarity: Complex roots of $1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p$ lie outside the unit circle on the complex plane.

- For $p = 1$: $-1 < \phi_1 < 1$.
- For $p = 2$: $-1 < \phi_2 < 1 \qquad \phi_2 + \phi_1 < 1 \qquad \phi_2 - \phi_1 < 1$.
- More complicated conditions hold for $p \geq 3$.
- Estimation software takes care of this.

7.2 Moving average models

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q},$$

where e_t is white noise. This is a multiple regression with **past *errors*** as predictors. *Don't confuse this with moving average smoothing!*



Invertibility

- Any MA(q) process can be written as an AR(∞) process if we impose some constraints on the MA parameters.
- Then the MA model is called "invertible".
- Invertible models have some mathematical properties that make them easier to use in practice.
- Invertibility of an ARIMA model is equivalent to forecastability of an ETS model.

General condition for invertibility: Complex roots of $1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_q z^q$ lie outside the unit circle on the complex plane.

- For $q = 1$: $-1 < \theta_1 < 1$.
- For $q = 2$: $-1 < \theta_2 < 1 \qquad \theta_2 + \theta_1 > -1 \qquad \theta_1 - \theta_2 < 1$.
- More complicated conditions hold for $q \geq 3$.
- Estimation software takes care of this.

7.3 ARIMA models

Autoregressive Moving Average models:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t.$$

- Predictors include both **lagged values of y_t and lagged errors.**
- Conditions on coefficients ensure stationarity.
- Conditions on coefficients ensure invertibility.

Autoregressive Integrated Moving Average models

- Combine ARMA model with **differencing**.
- $(1 - B)^d y_t$ follows an ARMA model.

ARIMA(p,d,q) model

$$\begin{array}{rl} \text{AR:} & p = \text{order of the autoregressive part} \\ \text{I:} & d = \text{degree of first differencing involved} \\ \text{MA:} & q = \text{order of the moving average part.} \end{array}$$

- White noise model: ARIMA(0,0,0)
- Random walk: ARIMA(0,1,0) with no constant
- Random walk with drift: ARIMA(0,1,0) with const.
- AR(p): ARIMA(p,0,0)
- MA(q): ARIMA(0,0,q)

Backshift notation for ARIMA

- ARMA model:

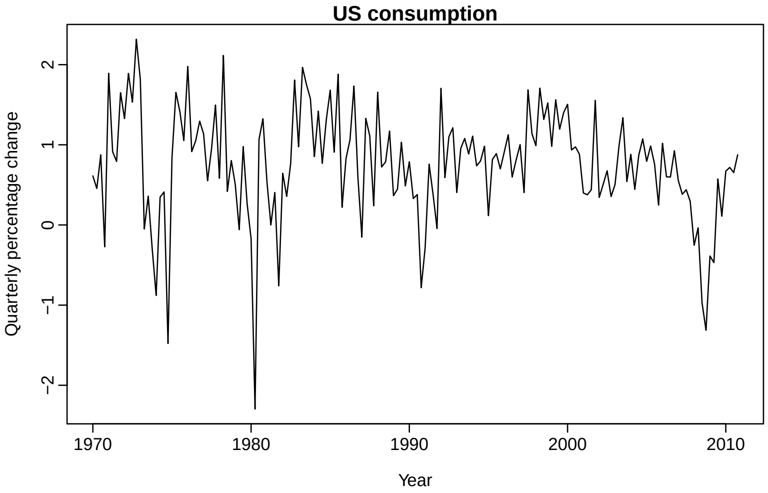
$$\begin{aligned} & y_t = c + \phi_1 B y_t + \cdots + \phi_p B^p y_t + e_t + \theta_1 B e_t + \cdots + \theta_q B^q e_t \\ \text{or} \quad & (1 - \phi_1 B - \cdots - \phi_p B^p) y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q) e_t \end{aligned}$$

- ARIMA(1,1,1) model:

$$\begin{array}{cccc} (1-\phi_1 B) & (1-B)y_t & = & c + (1+\theta_1 B)e_t \\ \uparrow & \uparrow & & \uparrow \\ \text{AR(1)} & \text{First} & & \text{MA(1)} \\ & \text{difference} & & \end{array}$$

Written out: $y_t = c + y_{t-1} + \phi_1 y_{t-1} - \phi_1 y_{t-2} + \theta_1 e_{t-1} + e_t$

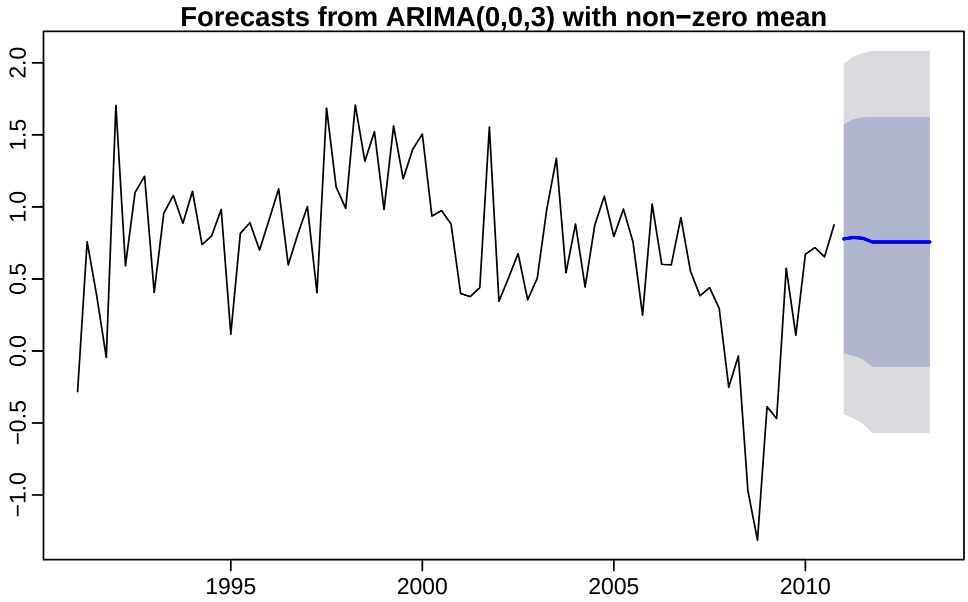
US personal consumption



```
> fit <- auto.arima(usconsumption[,1], max.P=0,max.Q=0,D=0)

ARIMA(0,0,3) with non-zero mean
         ma1     ma2     ma3  intercept
      0.2542  0.2260  0.2695     0.7562
s.e.  0.0767  0.0779  0.0692     0.0844

sigma^2 estimated as 0.3856:  log likelihood=-154.73
AIC=319.46   AICc=319.84   BIC=334.96
```

$$y_t = 0.756 + e_t + 0.254 e_{t-1} + 0.226 e_{t-2} + 0.269 e_{t-3},$$

where e_t is white noise with standard deviation $0.62 = \sqrt{0.3856}$.



```
plot(forecast(fit,h=10),include=80)
```

Understanding ARIMA models

- If $c = 0$ and $d = 0$, the long-term forecasts will go to zero.
- If $c = 0$ and $d = 1$, the long-term forecasts will go to a non-zero constant.
- If $c = 0$ and $d = 2$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 0$, the long-term forecasts will go to the mean of the data.
- If $c \neq 0$ and $d = 1$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 2$, the long-term forecasts will follow a quadratic trend.

Forecast variance and *d*

- The higher the value of d, the more rapidly the Forecast intervals increase in size.
- For $d = 0$, the long-term forecast standard deviation will go to the standard deviation of the historical data.

Cyclic behaviour

- For cyclic forecasts, $p > 2$ and some restrictions on coefficients are required.
- If $p = 2$, we need $\phi_1^2 + 4\phi_2 < 0$. Then average cycle of length

$$(2\pi)/[\text{arc}\cos(-\phi_1(1-\phi_2)/(4\phi_2))].$$

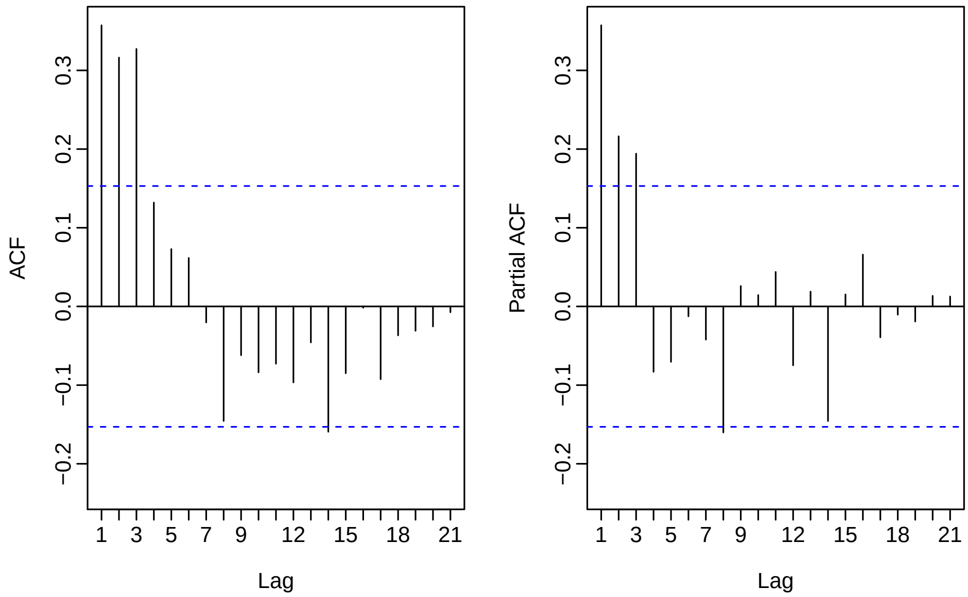
Partial autocorrelations

Partial autocorrelations measure relationship between y_t and y_{t-k}, when the effects of other time lags $(1, 2, 3, \dots, k-1)$ are removed.

$$\begin{aligned}\alpha_k &= k\text{th partial autocorrelation coefficient}\\ &= \text{equal to the estimate of } b_k \text{ in regression:}\\ &\qquad y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_k y_{t-k}.\end{aligned}$$

- Varying number of terms on RHS gives α_k for different values of k.
- There are more efficient ways of calculating α_k.
- $\alpha_1 = \rho_1$
- same critical values of $\pm 1.96/\sqrt{T}$ as for ACF.

Example: US consumption



ACF and PACF interpretation

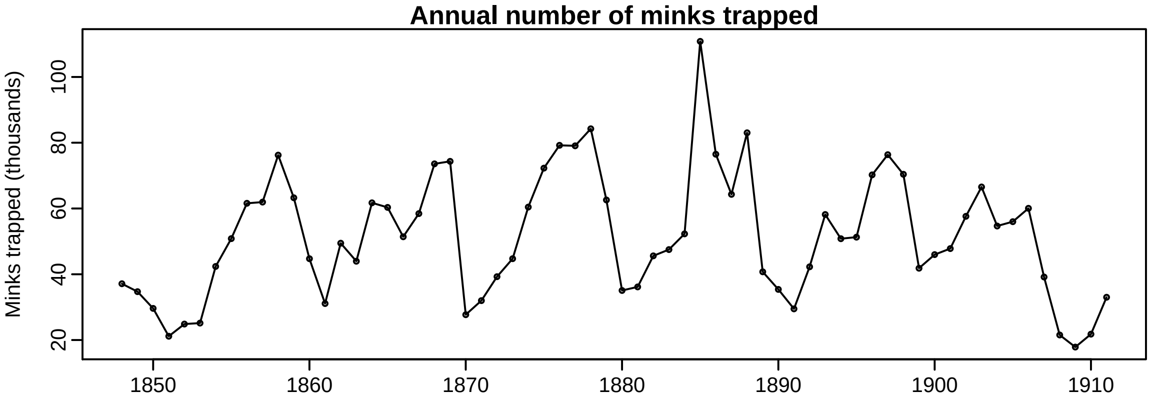
ARIMA(p,d,0) model if ACF and PACF plots of differenced data show:

- the ACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag p in PACF, but none beyond lag p.

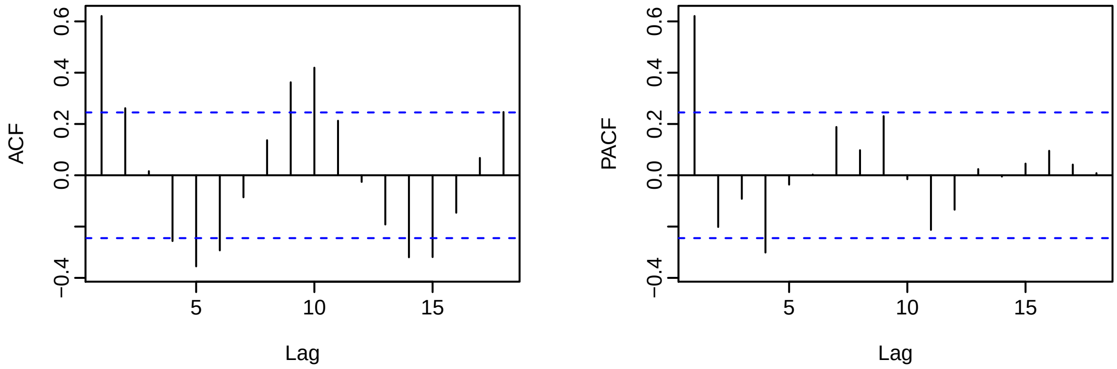
ARIMA(0,d,q) model if ACF and PACF plots of differenced data show:

- the PACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag q in ACF, but none beyond lag q.

Example: Mink trapping



Annual number of minks trapped



7.4 Estimation and order selection

Maximum likelihood estimation

Having identified the model order, we need to estimate the parameters c, $\phi_1,\dots,\phi_p$, $\theta_1,\dots,\theta_q$.

- MLE is very similar to least squares estimation obtained by minimizing $\sum_{t-1}^{T} e_t^2$.
- The `Arima()` command allows CLS or MLE estimation.
- Non-linear optimization must be used in either case.
- Different software will give different estimates.

Information criteria

Akaike's Information Criterion (AIC):

$$\text{AIC} = -2\log(L) + 2(p+q+k+1),$$

where L is the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$.

Corrected AIC:

$$\text{AIC}_\text{c} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2}.$$

Bayesian Information Criterion:

$$\text{BIC} = \text{AIC} + \log(T)(p+q+k-1).$$

Good models are obtained by minimizing either the AIC, AIC_c or BIC. Our preference is to use the AIC_c.

7.5 ARIMA modelling in R

`auto.arima()`: **Hyndman and Khandakar (JSS, 2008) algorithm**

- Select no. differences d and D via unit root tests.
- Select p, q by minimising AICc.
- Use stepwise search to traverse model space.

Step 1: Select current model (with smallest AIC) from:
ARIMA$(2, d, 2)$
ARIMA$(0, d, 0)$
ARIMA$(1, d, 0)$
ARIMA$(0, d, 1)$

Step 2: Consider variations of current model:
• vary one of p, q, from current model by ±1
• p, q both vary from current model by ±1
• Include/exclude c from current model
Model with lowest AIC becomes current model.

Repeat Step 2 until no lower AIC can be found.

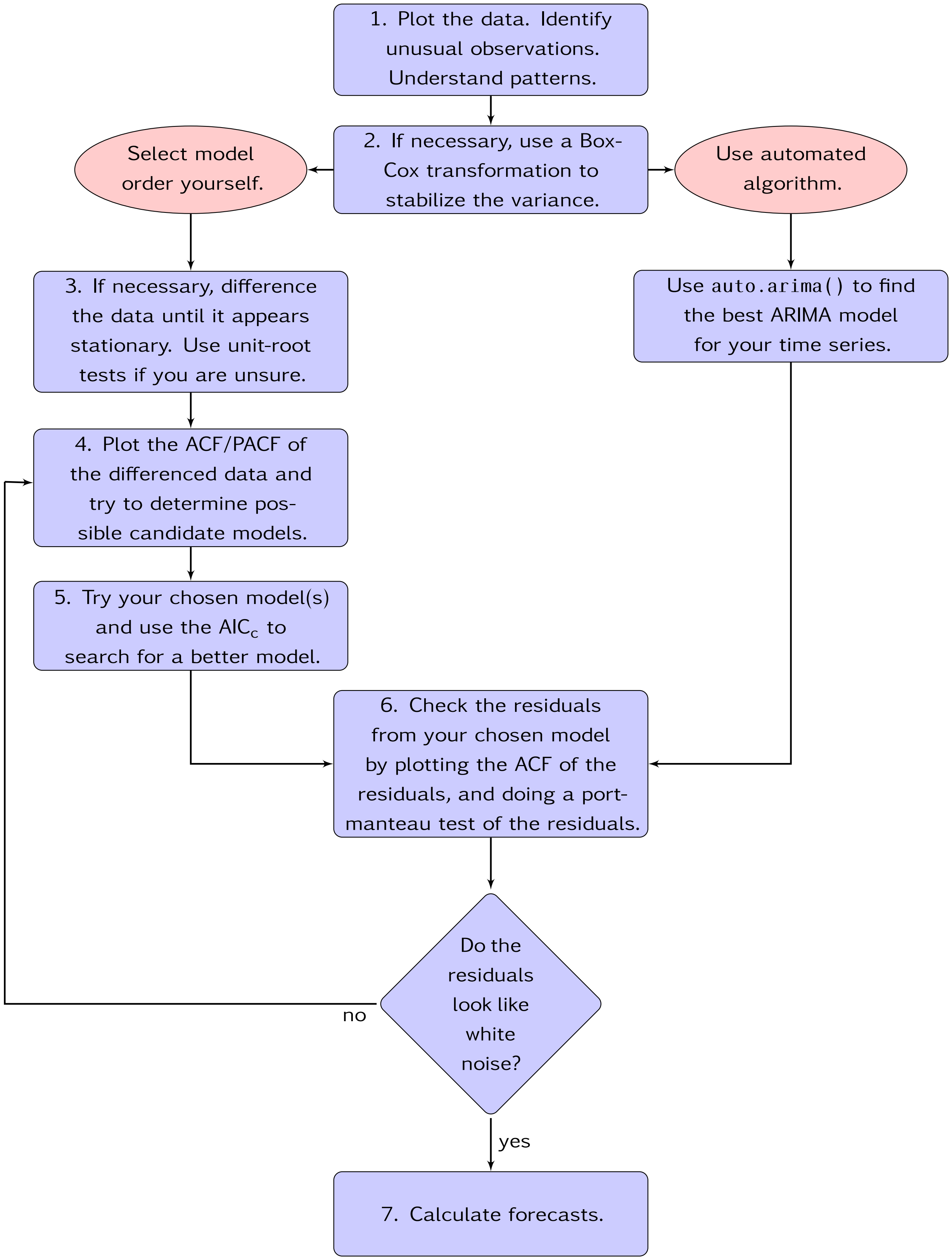
Choosing your own model

```
tsdisplay(internet)
adf.test(internet)
kpss.test(internet)
kpss.test(diff(internet))
tsdisplay(diff(internet))
fit <- Arima(internet,order=c(3,1,0))
fit2 <- auto.arima(internet)
Acf(residuals(fit))
Box.test(residuals(fit), fitdf=3, lag=10, type="Ljung")
tsdiag(fit)
forecast(fit)
plot(forecast(fit))
```

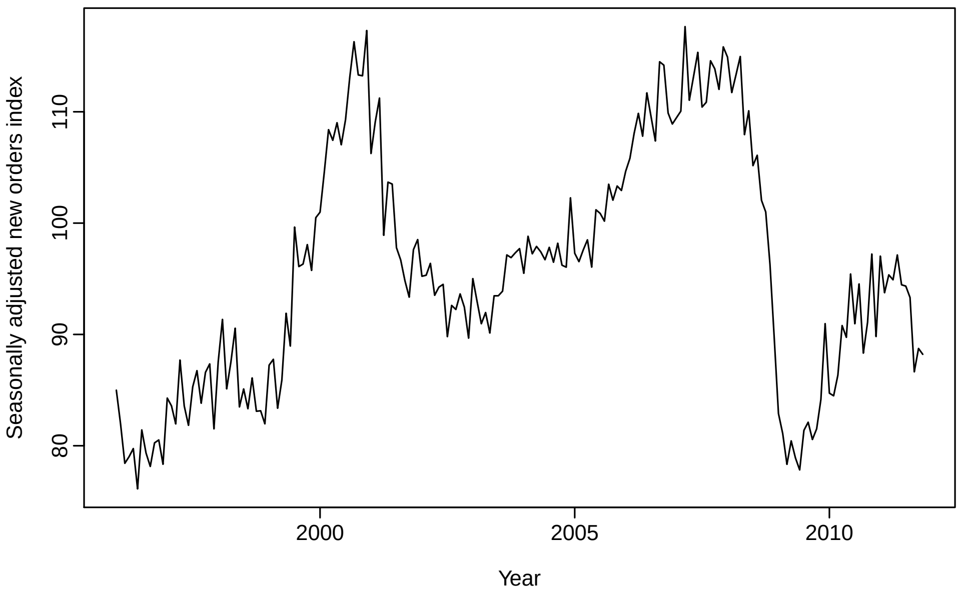
Modelling procedure

1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an AR(p) or MA(q) model appropriate?
5. Try your chosen model(s), and use the AIC$_c$ to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

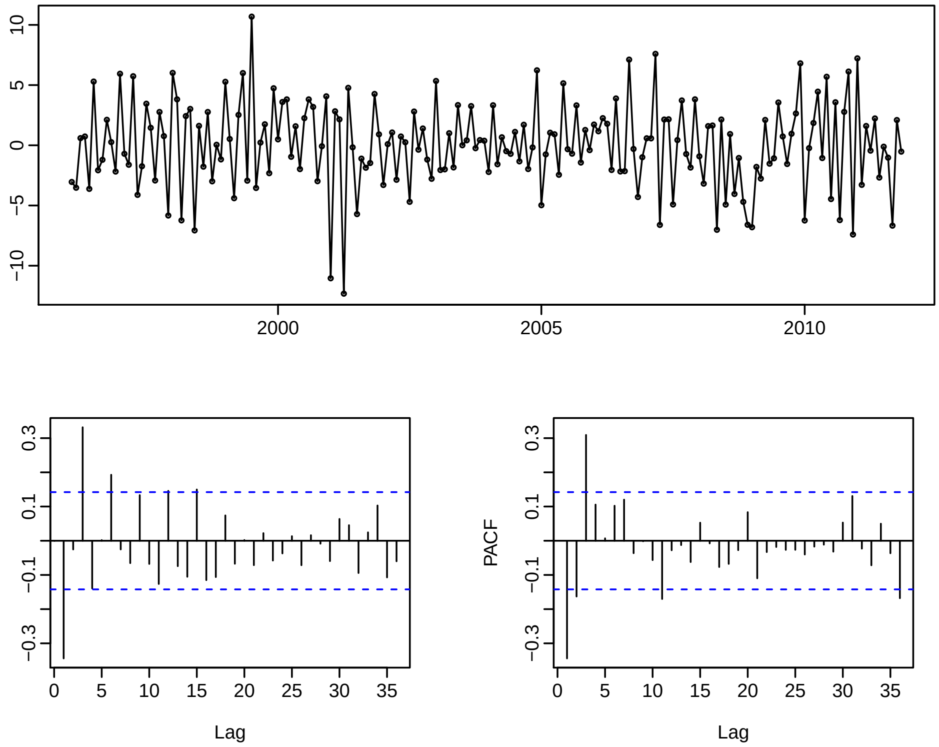
The automated algorithm only takes care of steps 3–5. So even if you use it, you will still need to take care of the other steps yourself.


1. Plot the data. Identify unusual observations. Understand patterns.
2. If necessary, use a Box-Cox transformation to stabilize the variance.
Select model order yourself.
Use automated algorithm.
3. If necessary, difference the data until it appears stationary. Use unit-root tests if you are unsure.
Use auto.arima() to find the best ARIMA model for your time series.
4. Plot the ACF/PACF of the differenced data and try to determine possible candidate models.
5. Try your chosen model(s) and use the AIC$_c$ to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals.
Do the residuals look like white noise?
no
yes
7. Calculate forecasts.

Seasonally adjusted electrical equipment



1. Time plot shows sudden changes, particularly big drop in 2008/2009 due to global economic environment. Otherwise nothing unusual and no need for data adjustments.
2. No evidence of changing variance, so no Box-Cox transformation.
3. Data are clearly non-stationary, so we take first differences.



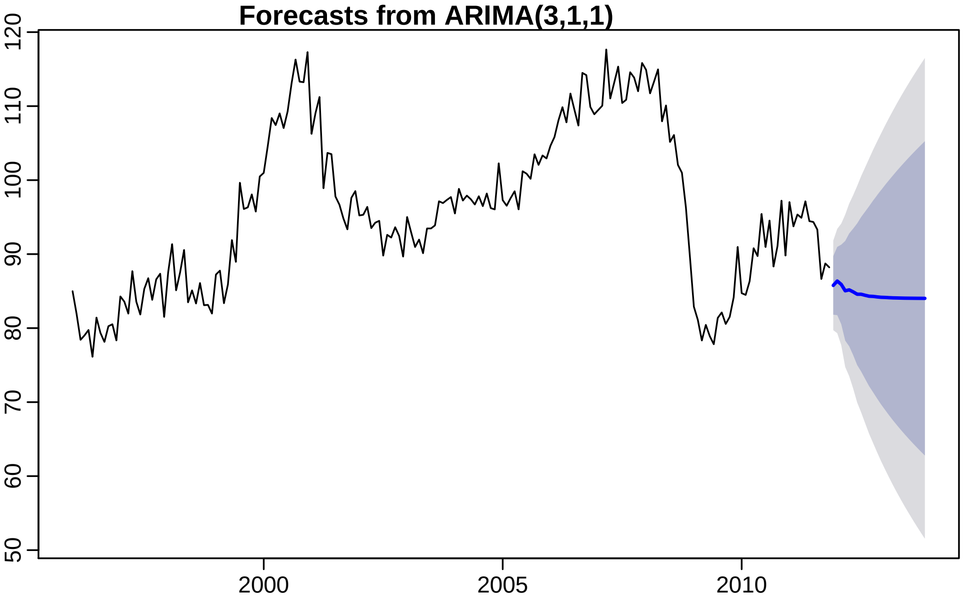
```
> tsdisplay(diff(eeadj))
```

4. PACF is suggestive of AR(3). So initial candidate model is ARIMA(3,1,0). No other obvious candidates.
5. Fit ARIMA(3,1,0) model along with variations: ARIMA(4,1,0), ARIMA(2,1,0), ARIMA(3,1,1), etc. ARIMA(3,1,1) has smallest AIC$_c$ value.

```
> fit <- Arima(eeadj, order=c(3,1,1))
> summary(fit)
Series: eeadj
ARIMA(3,1,1)

Coefficients:
         ar1     ar2     ar3      ma1
      0.0519  0.1191  0.3730  -0.4542
s.e.  0.1840  0.0888  0.0679   0.1993

sigma^2 estimated as 9.532:  log likelihood=-484.08
AIC=978.17   AICc=978.49   BIC=994.4
```

6. ACF plot of residuals from ARIMA(3,1,1) model look like white noise.

```
Acf(residuals(fit))
Box.test(residuals(fit), lag=24, fitdf=4, type="Ljung")
```



```
> plot(forecast(fit))
```

7.6 Forecasting

Point forecasts

1. Rearrange ARIMA equation so y_t is on LHS.
2. Rewrite equation by replacing t by $T+h$.
3. On RHS, replace future observations by their forecasts, future errors by zero, and past errors by corresponding residuals.

Start with $h = 1$. Repeat for $h = 2, 3, \ldots$.

Use US consumption model ARIMA(3,1,1) to demonstrate:

$$(1-\hat\phi_1 B-\hat\phi_2 B^2-\hat\phi_3 B^3)(1-B)y_t = (1+\hat\theta_1 B)e_t,$$

where $\hat\phi_1 = 0.0519$, $\hat\phi_2 = 0.1191$, $\hat\phi_3 = 0.3730$ and $\hat\theta_1 = -0.4542$.

Expand LHS:

$$\left[1-(1+\hat\phi_1)B+(\hat\phi_1-\hat\phi_2)B^2+(\hat\phi_2-\hat\phi_3)B^3+\hat\phi_3 B^4\right]y_t = (1+\hat\theta_1 B)e_t,$$

Apply backshift operator:

$$y_t - (1+\hat\phi_1)y_{t-1} + (\hat\phi_1 - \hat\phi_2)y_{t-2} + (\hat\phi_2 - \hat\phi_3)y_{t-3} + \hat\phi_3 y_{t-4} = e_t + \hat\theta_1 e_{t-1}.$$

Move all terms other than y_t to RHS:

$$y_t = (1+\hat\phi_1)y_{t-1} - (\hat\phi_1 - \hat\phi_2)y_{t-2} - (\hat\phi_2 - \hat\phi_3)y_{t-3} - \hat\phi_3 y_{t-4} + e_t + \hat\theta_1 e_{t-1}. \quad (7.1)$$

$h = 1$

Replace t by $T+1$ in (7.1):

$$y_{T+1} = (1+\hat\phi_1)y_T - (\hat\phi_1 - \hat\phi_2)y_{T-1} - (\hat\phi_2 - \hat\phi_3)y_{T-2} - \hat\phi_3 y_{T-3} + e_{T+1} + \hat\theta_1 e_T.$$

Replace e_{T+1} by 0 and e_T by $\hat e_T$:

$$\hat y_{T+1|T} = (1+\hat\phi_1)y_T - (\hat\phi_1 - \hat\phi_2)y_{T-1} - (\hat\phi_2 - \hat\phi_3)y_{T-2} - \hat\phi_3 y_{T-3} + \hat\theta_1 \hat e_T.$$

$h = 2$

Replace t by $T+2$ in (7.1), replace y_{T+1} by $\hat y_{T+1}$, replace e_{T+h} by 0 for $h > 0$:

$$\hat y_{T+2|T} = (1+\hat\phi_1)\hat y_{T+1|T} - (\hat\phi_1 - \hat\phi_2)y_T - (\hat\phi_2 - \hat\phi_3)y_{T-1} - \hat\phi_3 y_{T-2}.$$

Forecast intervals

95% forecast interval: $\hat y_{T+h|T} \pm 1.96\sqrt{v_{T+h|T}}$ where $v_{T+h|T}$ is estimated forecast variance.

- $v_{T+1|T} = \hat\sigma^2$ for all ARIMA models regardless of parameters and orders.
- Multi-step forecast intervals for ARIMA(0,0,q):
$$y_t = e_t + \sum_{i=1}^{q} \theta_i e_{t-i}.$$
$$v_{T|T+h} = \hat\sigma^2\left[1 + \sum_{i=1}^{h-1}\theta_i^2\right], \qquad \text{for } h = 2,3,\ldots.$$
- Forecast intervals **increase in size with forecast horizon.**
- Forecast intervals can be difficult to calculate by hand
- Calculations assume residuals are **uncorrelated** and **normally distributed**.
- Forecast intervals tend to be too narrow.
  - the uncertainty in the parameter estimates has not been accounted for.
  - the ARIMA model assumes historical patterns will not change during the forecast period.
  - the ARIMA model assumes uncorrelated future errors

7.7 Lab Session 7

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. For the `wmurders` data:
   (a) if necessary, find a suitable Box-Cox transformation for the data;
   (b) fit a suitable ARIMA model to the transformed data using `auto.arima()`;
   (c) try some other plausible models by experimenting with the orders chosen;
   (d) choose what you think is the best model and check the residual diagnostics;
   (e) produce forecasts of your fitted model. Do the forecasts look reasonable?
   (f) compare the results with what you would obtain using `ets()` (with no transformation).

2. For the `usgdp` data:
   (a) if necessary, find a suitable Box-Cox transformation for the data;
   (b) fit a suitable ARIMA model to the transformed data using `auto.arima()`;
   (c) try some other plausible models by experimenting with the orders chosen;
   (d) choose what you think is the best model and check the residual diagnostics;
   (e) produce forecasts of your fitted model. Do the forecasts look reasonable?
   (f) compare the results with what you would obtain using `ets()` (with no transformation).

3. For the `mcopper` data:
   (a) if necessary, find a suitable Box-Cox transformation for the data;
   (b) fit a suitable ARIMA model to the transformed data using `auto.arima()`;
   (c) try some other plausible models by experimenting with the orders chosen;
   (d) choose what you think is the best model and check the residual diagnostics;
   (e) produce forecasts of your fitted model. Do the forecasts look reasonable?
   (f) compare the results with what you would obtain using `ets()` (with no transformation).

8 Seasonal ARIMA models

$$\text{ARIMA}\ \ \underbrace{(p,d,q)}_{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Non-}\\ \text{seasonal}\\ \text{part of}\\ \text{the model}\end{pmatrix}\end{array}}\ \underbrace{(P,D,Q)_m}_{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Seasonal}\\ \text{part of}\\ \text{the model}\end{pmatrix}\end{array}} \qquad \text{where } m = \text{number of periods per season.}$$

E.g., ARIMA$(1,1,1)(1,1,1)_4$ model (without constant)

$$\underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Non-seasonal}\\ \text{AR(1)}\end{pmatrix}\end{array}}{(1-\phi_1 B)}\underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Seasonal}\\ \text{AR(1)}\end{pmatrix}\end{array}}{(1-\Phi_1 B^4)}\underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Non-seasonal}\\ \text{difference}\end{pmatrix}\end{array}}{(1-B)}\underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Seasonal}\\ \text{difference}\end{pmatrix}\end{array}}{(1-B^4)}y_t \;=\; \underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Non-seasonal}\\ \text{MA(1)}\end{pmatrix}\end{array}}{(1+\theta_1 B)}\underset{\begin{array}{c}\uparrow\\ \begin{pmatrix}\text{Seasonal}\\ \text{MA(1)}\end{pmatrix}\end{array}}{(1+\Theta_1 B^4)}e_t.$$

E.g., ARIMA$(1,1,1)(1,1,1)_4$ model (without constant)

$$(1-\phi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t \;=\; (1+\theta_1 B)(1+\Theta_1 B^4)e_t.$$

All the factors can be multiplied out and the general model written as follows:

$$\begin{aligned} y_t = {} & (1+\phi_1)y_{t-1} - \phi_1 y_{t-2} + (1+\Phi_1)y_{t-4} \\ & - (1+\phi_1+\Phi_1+\phi_1\Phi_1)y_{t-5} + (\phi_1+\phi_1\Phi_1)y_{t-6} \\ & - \Phi_1 y_{t-8} + (\Phi_1+\phi_1\Phi_1)y_{t-9} - \phi_1\Phi_1 y_{t-10} \\ & + e_t + \theta_1 e_{t-1} + \Theta_1 e_{t-4} + \theta_1\Theta_1 e_{t-5}. \end{aligned}$$

8.1 Common ARIMA models

In the US Census Bureau uses the following models most often:

ARIMA(0,1,1)(0,1,1)$_m$	with log transformation
ARIMA(0,1,2)(0,1,1)$_m$	with log transformation
ARIMA(2,1,0)(0,1,1)$_m$	with log transformation
ARIMA(0,2,2)(0,1,1)$_m$	with log transformation
ARIMA(2,1,2)(0,1,1)$_m$	with no transformation

8.2 ACF and PACF of seasonal ARIMA models

The seasonal part of an AR or MA model will be seen in the seasonal lags of the PACF and ACF.

ARIMA(0,0,0)(0,0,1)$_{12}$ will show:

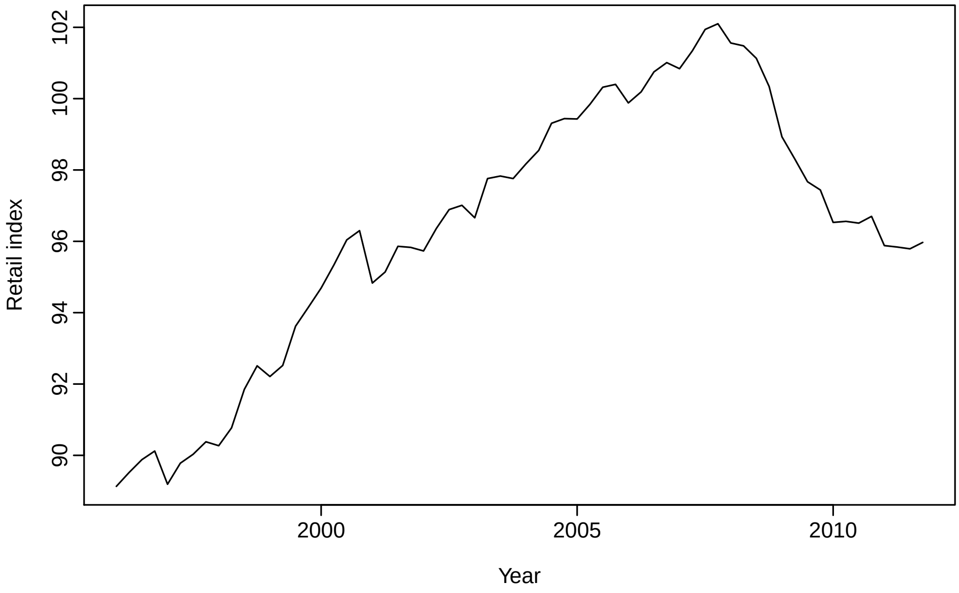
- a spike at lag 12 in the ACF but no other significant spikes.
- The PACF will show exponential decay in the seasonal lags; that is, at lags 12, 24, 36, ...

ARIMA(0,0,0)(1,0,0)$_{12}$ will show:

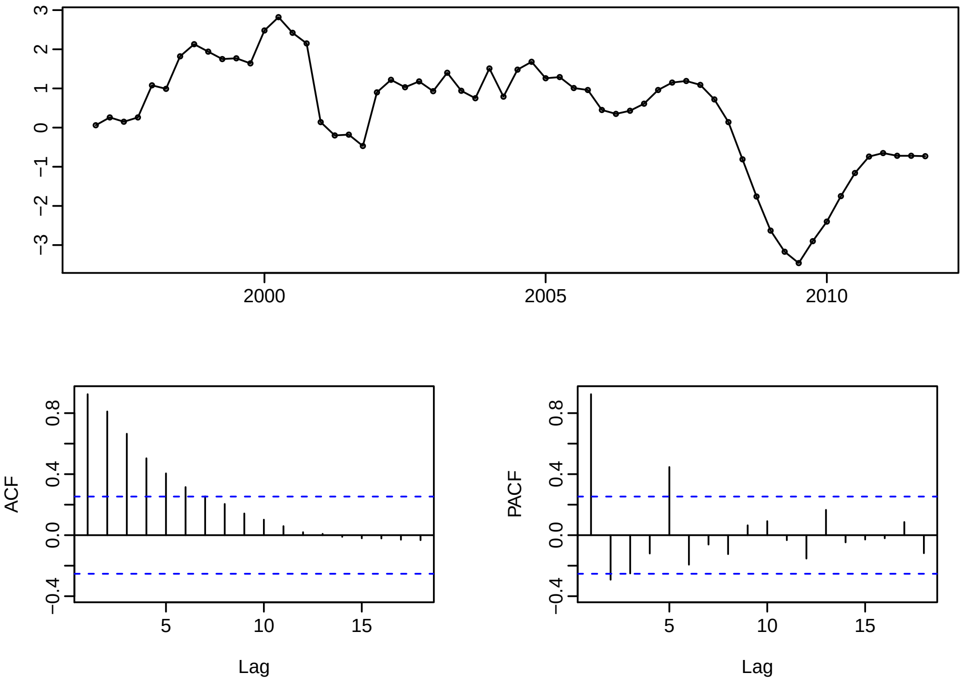
- exponential decay in the seasonal lags of the ACF
- a single significant spike at lag 12 in the PACF.

8.3 Example: European quarterly retail trade

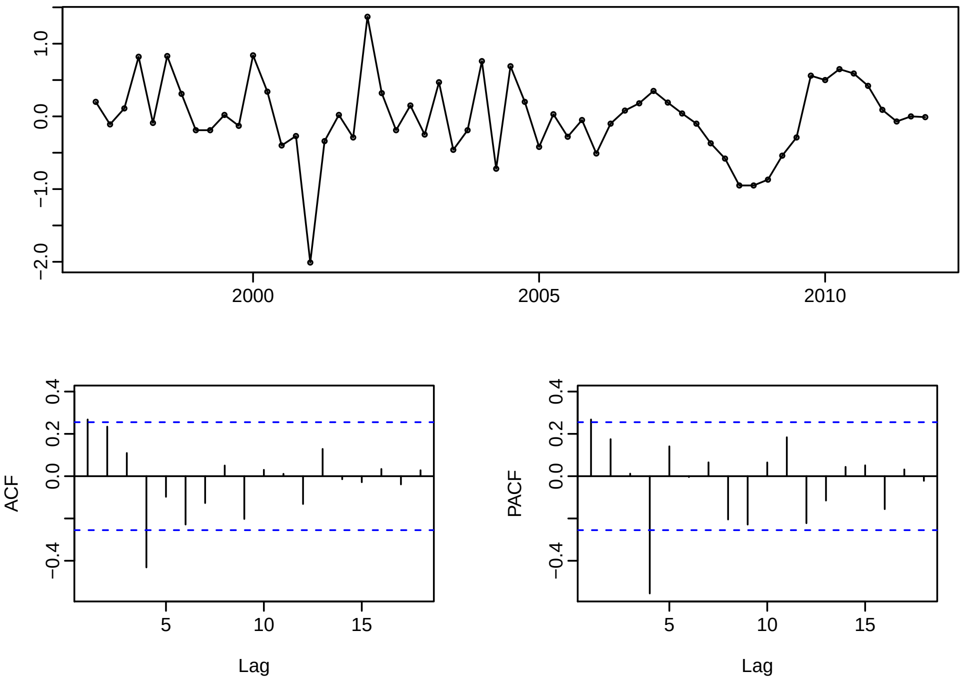
```
> plot(euretail)
```



```
> tsdisplay(diff(euretail,4))
```

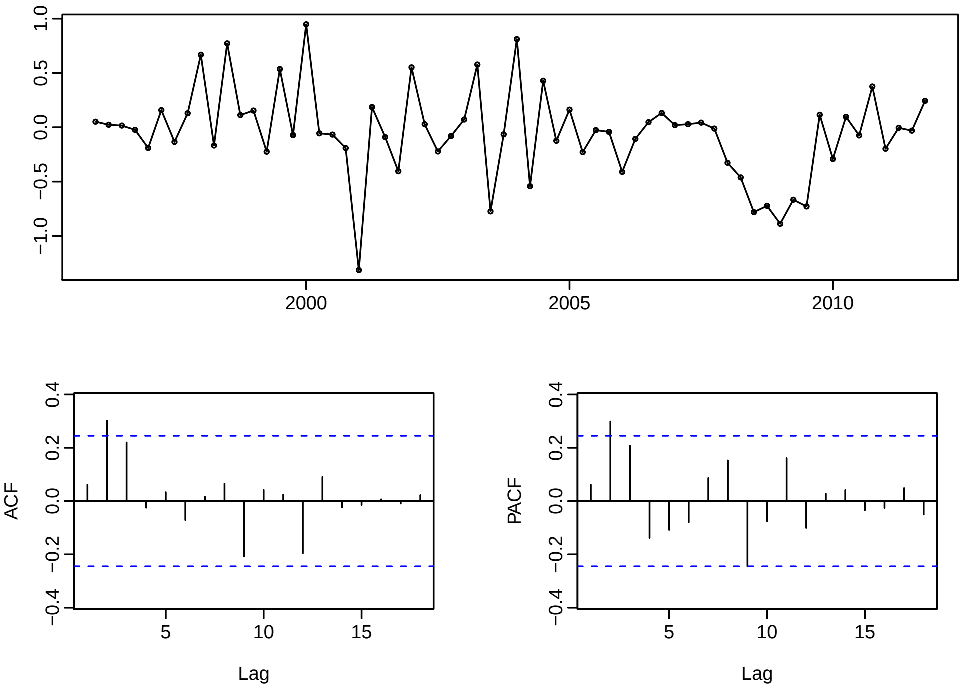


```
> tsdisplay(diff(diff(euretail,4)))
```



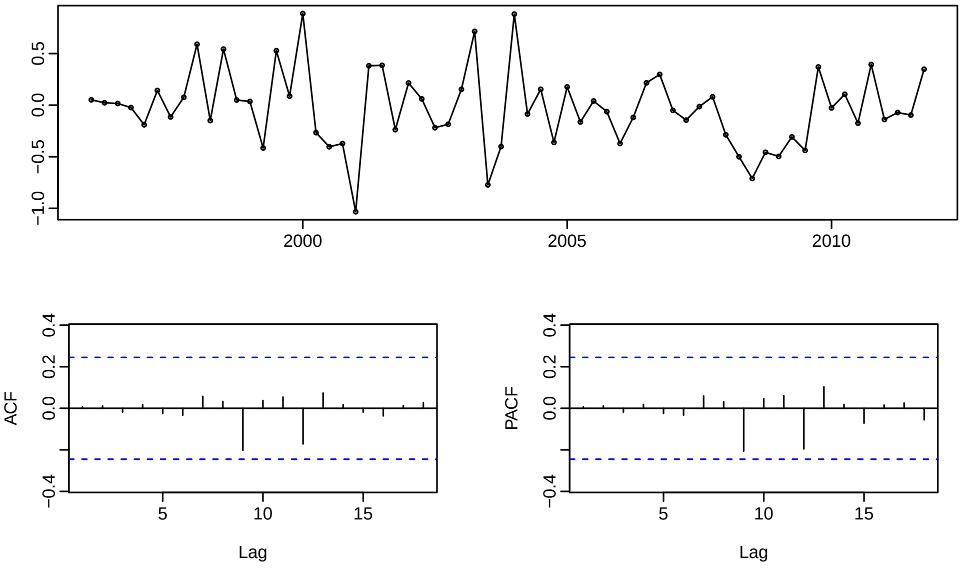
- $d = 1$ and $D = 1$ seems necessary.
- Significant spike at lag 1 in ACF suggests non-seasonal MA(1) component.
- Significant spike at lag 4 in ACF suggests seasonal MA(1) component.
- Initial candidate model: ARIMA(0,1,1)(0,1,1)$_4$.
- We could also have started with ARIMA(1,1,0)(1,1,0)$_4$.

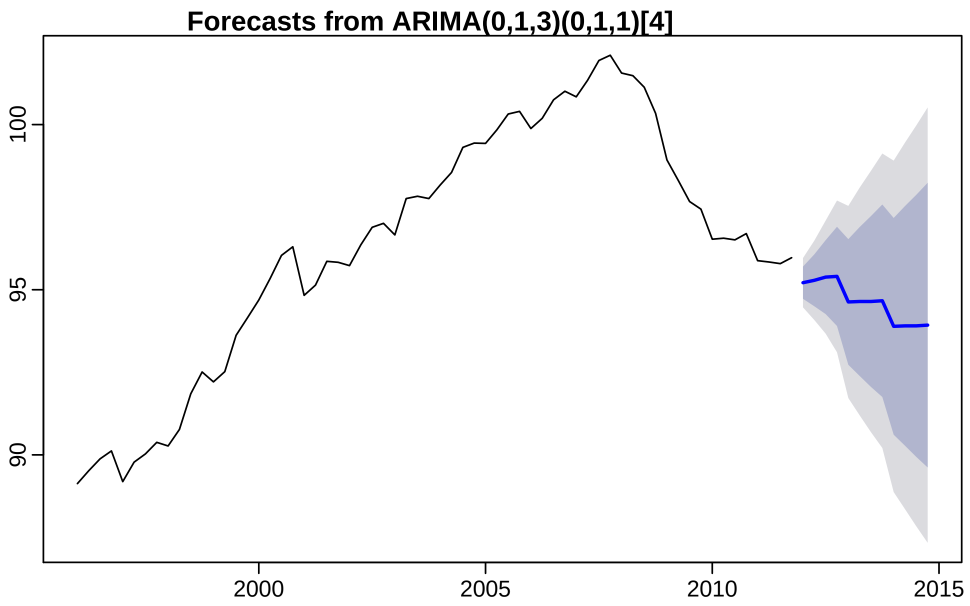
```
fit <- Arima(euretail, order=c(0,1,1), seasonal=c(0,1,1))
tsdisplay(residuals(fit))
```



- ACF and PACF of residuals show significant spikes at lag 2, and maybe lag 3.
- AIC$_c$ of ARIMA(0,1,2)(0,1,1)$_4$ model is 74.36.
- AIC$_c$ of ARIMA(0,1,3)(0,1,1)$_4$ model is 68.53.

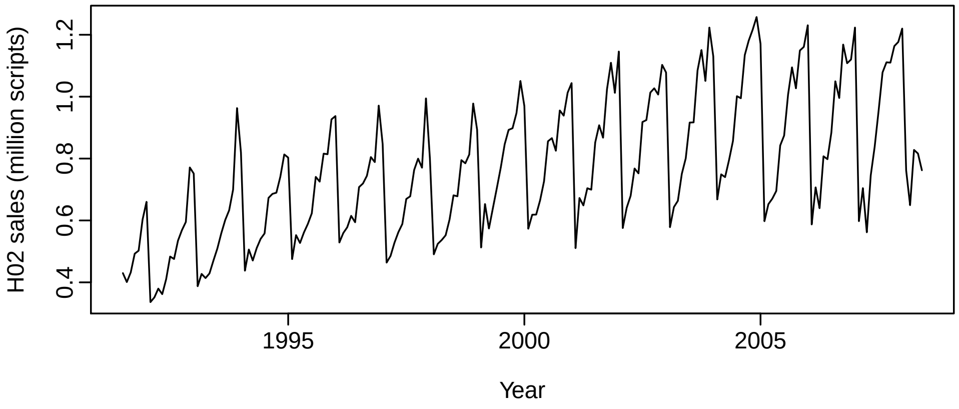
```
fit <- Arima(euretail, order=c(0,1,3), seasonal=c(0,1,1))
tsdisplay(residuals(fit))
Box.test(res, lag=16, fitdf=4, type="Ljung")
plot(forecast(fit3, h=12))
```

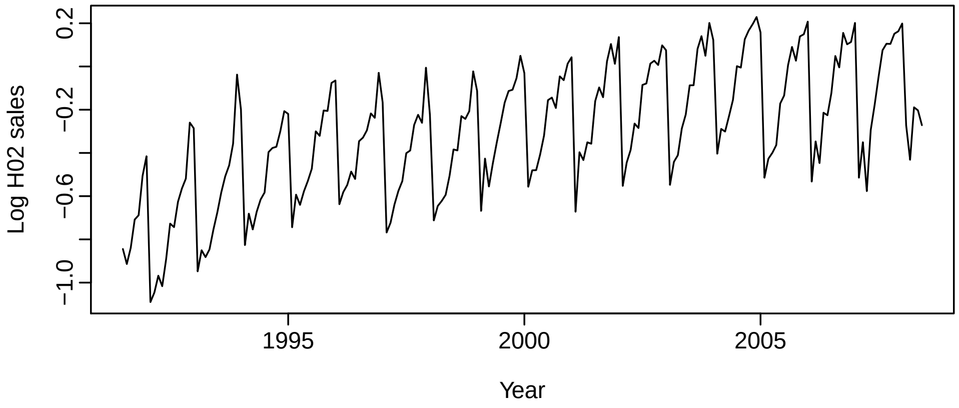


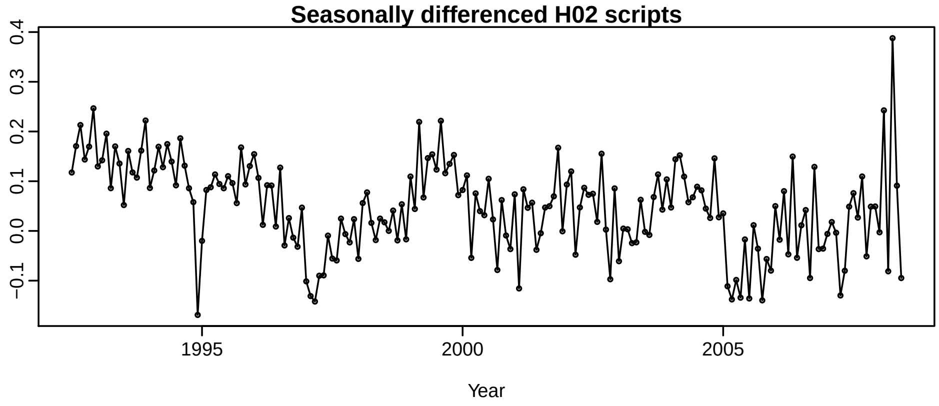


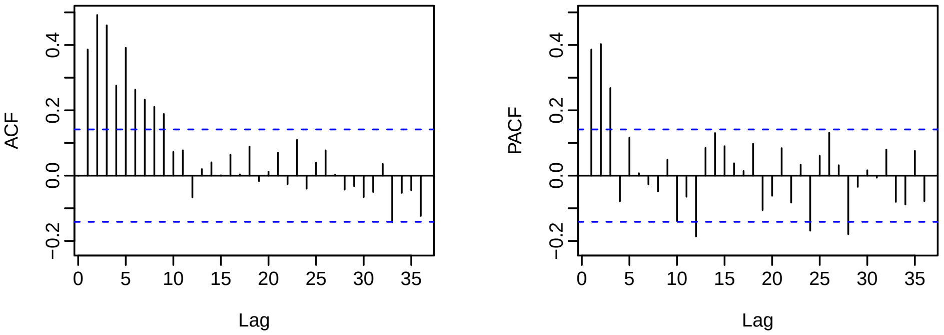
```
> auto.arima(euretail)
ARIMA(1,1,1)(0,1,1)[4]

Coefficients:
         ar1      ma1     sma1
      0.8828  -0.5208  -0.9704
s.e.  0.1424   0.1755   0.6792

sigma^2 estimated as 0.1411:  log likelihood=-30.19
AIC=68.37   AICc=69.11   BIC=76.68


> auto.arima(euretail, stepwise=FALSE, approximation=FALSE)
ARIMA(0,1,3)(0,1,1)[4]

Coefficients:
         ma1     ma2     ma3     sma1
      0.2625  0.3697  0.4194  -0.6615
s.e.  0.1239  0.1260  0.1296   0.1555

sigma^2 estimated as 0.1451:  log likelihood=-28.7
AIC=67.4   AICc=68.53   BIC=77.78
```

8.4 Example: Cortecosteroid drug sales





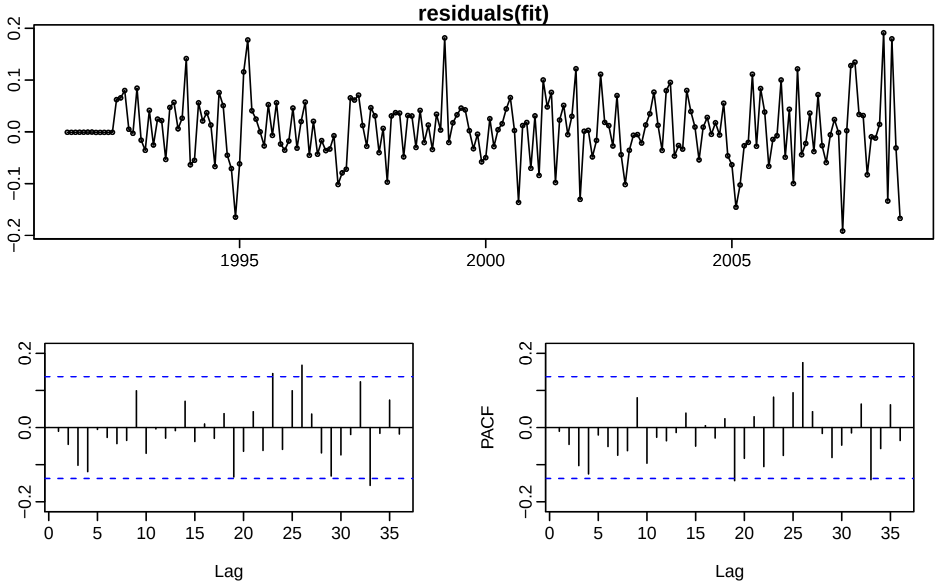




- Choose $D = 1$ and $d = 0$.
- Spikes in PACF at lags 12 and 24 suggest seasonal AR(2) term.
- Spikes in PACF sugges possible non-seasonal AR(3) term.
- Initial candidate model: ARIMA(3,0,0)$(2,1,0)_{12}$.

Model	AIC$_c$
ARIMA(3,0,0)(2,1,0)$_{12}$	−475.12
ARIMA(3,0,1)(2,1,0)$_{12}$	−476.31
ARIMA(3,0,2)(2,1,0)$_{12}$	−474.88
ARIMA(3,0,1)(1,1,0)$_{12}$	−463.40
ARIMA(3,0,1)(0,1,1)$_{12}$	−483.67
ARIMA(3,0,1)(0,1,2)$_{12}$	−485.48
ARIMA(3,0,1)(1,1,1)$_{12}$	−484.25

```
> fit <- Arima(h02, order=c(3,0,1), seasonal=c(0,1,2), lambda=0)

ARIMA(3,0,1)(0,1,2)[12]
Box Cox transformation: lambda= 0

Coefficients:
          ar1     ar2     ar3     ma1     sma1     sma2
      -0.1603  0.5481  0.5678  0.3827  -0.5222  -0.1768
s.e.   0.1636  0.0878  0.0942  0.1895   0.0861   0.0872

sigma^2 estimated as 0.004145:  log likelihood=250.04
AIC=-486.08   AICc=-485.48   BIC=-463.28
```



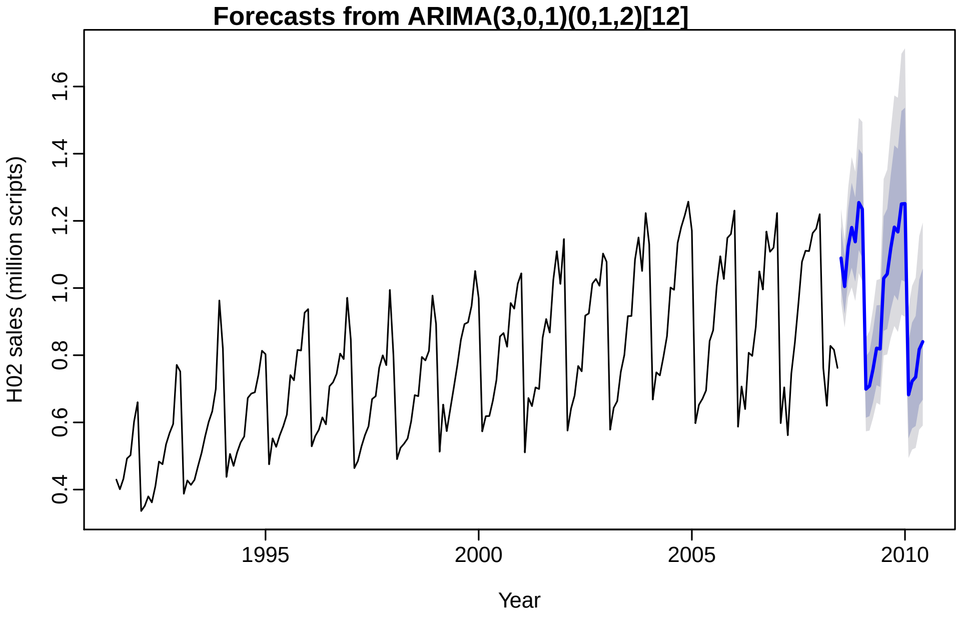
```
tsdisplay(residuals(fit))
Box.test(residuals(fit), lag=36, fitdf=6, type="Ljung")
auto.arima(h02,lambda=0)
```

Training: July 91 – June 06
Test: July 06 – June 08

Model	RMSE
ARIMA(3,0,0)(2,1,0)$_{12}$	0.0661
ARIMA(3,0,1)(2,1,0)$_{12}$	0.0646
ARIMA(3,0,2)(2,1,0)$_{12}$	0.0645
ARIMA(3,0,1)(1,1,0)$_{12}$	0.0679
ARIMA(3,0,1)(0,1,1)$_{12}$	0.0644
ARIMA(3,0,1)(0,1,2)$_{12}$	0.0622
ARIMA(3,0,1)(1,1,1)$_{12}$	0.0630
ARIMA(4,0,3)(0,1,1)$_{12}$	0.0648
ARIMA(3,0,3)(0,1,1)$_{12}$	0.0640
ARIMA(4,0,2)(0,1,1)$_{12}$	0.0648
ARIMA(3,0,2)(0,1,1)$_{12}$	0.0644
ARIMA(2,1,3)(0,1,1)$_{12}$	0.0634
ARIMA(2,1,4)(0,1,1)$_{12}$	0.0632
ARIMA(2,1,5)(0,1,1)$_{12}$	0.0640

```
getrmse <- function(x,h,...)
{
  train.end <- time(x)[length(x)-h]
  test.start <- time(x)[length(x)-h+1]
  train <- window(x,end=train.end)
  test <- window(x,start=test.start)
  fit <- Arima(train,...)
  fc <- forecast(fit,h=h)
  return(accuracy(fc,test)[2,"RMSE"])
}
getrmse(h02,h=24,order=c(3,0,0),seasonal=c(2,1,0),lambda=0)
getrmse(h02,h=24,order=c(3,0,1),seasonal=c(2,1,0),lambda=0)
getrmse(h02,h=24,order=c(3,0,2),seasonal=c(2,1,0),lambda=0)
getrmse(h02,h=24,order=c(3,0,1),seasonal=c(1,1,0),lambda=0)
getrmse(h02,h=24,order=c(3,0,1),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(3,0,1),seasonal=c(0,1,2),lambda=0)
getrmse(h02,h=24,order=c(3,0,1),seasonal=c(1,1,1),lambda=0)
getrmse(h02,h=24,order=c(4,0,3),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(3,0,3),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(4,0,2),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(3,0,2),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(2,1,3),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(2,1,4),seasonal=c(0,1,1),lambda=0)
getrmse(h02,h=24,order=c(2,1,5),seasonal=c(0,1,1),lambda=0)
```

- Models with lowest AIC$_c$ values tend to give slightly better results than the other models.
- AIC$_c$ comparisons must have the same orders of differencing. But RMSE test set comparisons can involve any models.
- No model passes all the residual tests.
- Use the best model available, even if it does not pass all tests.
- In this case, the ARIMA(3,0,1)(0,1,2)$_{12}$ has the lowest RMSE value and the best AIC$_c$ value for models with fewer than 6 parameters.



8.5 ARIMA vs ETS

- Myth that ARIMA models are more general than exponential smoothing.
- Linear exponential smoothing models all special cases of ARIMA models.
- Non-linear exponential smoothing models have no equivalent ARIMA counterparts.
- Many ARIMA models have no exponential smoothing counterparts.
- ETS models all non-stationary. Models with seasonality or non-damped trend (or both) have two unit roots; all other models have one unit root.

Equivalences

Simple exponential smoothing

- Forecasts equivalent to $\mathbf{ARIMA(0,1,1)}$.
- Parameters: $\theta_1 = \alpha - 1$.

Holt's method

- Forecasts equivalent to $\mathbf{ARIMA(0,2,2)}$.
- Parameters: $\theta_1 = \alpha + \beta - 2$ and $\theta_2 = 1 - \alpha$.

Damped Holt's method

- Forecasts equivalent to $\mathbf{ARIMA(1,1,2)}$.
- Parameters: $\phi_1 = \phi$, $\theta_1 = \alpha + \phi\beta - 2$, $\theta_2 = (1 - \alpha)\phi$.

Holt-Winters' additive method

- Forecasts equivalent to $\mathbf{ARIMA(0,1,m{+}1)(0,1,0)_m}$.
- Parameter restrictions because ARIMA has $m + 1$ parameters whereas HW uses only three parameters.

Holt-Winters' multiplicative method

- No ARIMA equivalence

8.6 Lab Session 8

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Choose one of the following seasonal time series: `condmilk`, `hsales`, `uselec`
   (a) Do the data need transforming? If so, find a suitable transformation.
   (b) Are the data stationary? If not, find an appropriate differencing which yields stationary data.
   (c) Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?
   (d) Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.
   (e) Forecast the next 24 months of data using your preferred model.
   (f) Compare the forecasts obtained using `ets()`.

2. For the time series you selected from the retail data set in Lab Session 6, develop an appropriate seasonal ARIMA model, and compare the forecasts with those you obtained earlier.

   Obtain up-to-date data from January 2008 onwards from the ABS website (`www.abs.gov.au`) (Cat. 8501.0, Table 11), and compare your forecasts with the actual numbers. How good were the forecasts from the various models?

9 State space models

9.1 Simple structural models

Analogous to additive ETS models except:

- y_t depends on $\boldsymbol{x}_t$.
- A different error process affects $\boldsymbol{x}_t|\boldsymbol{x}_{t-1}$ and $y_t|\boldsymbol{x}_t$.

Local level model

$$\begin{aligned} y_t &= \ell_t + \varepsilon_t \\ \ell_t &= \ell_{t-1} + \xi_t \end{aligned}$$

- ε_t and ξ_t are independent Gaussian white noise processes.
- Compare ETS(A,N,N) where $\xi_t = \alpha\varepsilon_{t-1}$.
- Parameters to estimate: σ_ε^2 and σ_ξ^2.
- If $\sigma_\xi^2 = 0$, $y_t \sim \text{NID}(\ell_0, \sigma_\varepsilon^2)$.

Local linear trend model

$$\begin{aligned} y_t &= \ell_t + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \xi_t \\ b_t &= b_{t-1} + \zeta_t \end{aligned}$$

- ε_t, ξ_t and ζ_t are independent Gaussian white noise processes.
- Compare ETS(A,A,N) where $\xi_t = (\alpha+\beta)\varepsilon_{t-1}$ and $\zeta_t = \beta\varepsilon_{t-1}$
- Parameters to estimate: σ_ε^2, σ_ξ^2, and σ_ζ^2.
- If $\sigma_\zeta^2 = \sigma_\xi^2 = 0$, $y_t = \ell_0 + tb_0 + \varepsilon_t$.
- Model is a time-varying linear regression.

Basic structural model

$$\begin{aligned} y_t &= \ell_t + s_{1,t} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \xi_t \\ b_t &= b_{t-1} + \zeta_t \\ s_{1,t} &= -\sum_{j=1}^{m-1} s_{j,t-1} + \eta_t \qquad s_{j,t} = s_{j-1,t-1}, \qquad j = 2,\dots,m-1 \end{aligned}$$

- ε_t, ξ_t, ζ_t and η_t are independent Gaussian white noise processes.
- Compare ETS(A,A,A).
- Parameters to estimate: σ^2_ε, σ^2_ξ, σ^2_ζ and σ^2_η
- Deterministic seasonality if $\sigma^2_\eta = 0$.

Trigonometric models

$$
\begin{aligned}
y_t &= \ell_t + \sum_{j=1}^{J} s_{j,t} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + \xi_t \\
b_t &= b_{t-1} + \zeta_t \\
s_{j,t} &= \cos\lambda_j s_{j,t-1} + \sin\lambda_j s^*_{j,t-1} + \omega_{j,t} \\
s^*_{j,t} &= -\sin\lambda_j s_{j,t-1} + \cos\lambda_j s^*_{j,t-1} + \omega^*_{j,t}
\end{aligned}
$$

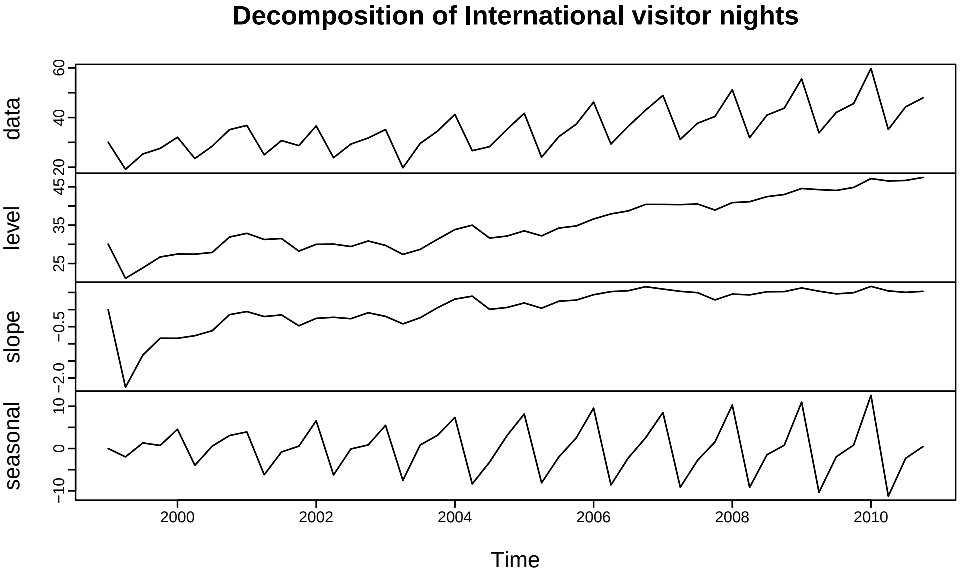
- $\lambda_j = 2\pi j/m$
- ε_t, ξ_t, ζ_t, $\omega_{j,t}$, $\omega^*_{j,t}$ are independent Gaussian white noise processes
- $\omega_{j,t}$ and $\omega^*_{j,t}$ have same variance $\sigma^2_{\omega,j}$
- Equivalent to BSM when $\sigma^2_{\omega,j} = \sigma^2_\omega$ and $J = m/2$
- Choose $J < m/2$ for fewer degrees of freedom

ETS vs Structural models

- ETS models are much more general as they allow non-linear (multiplicative components).
- ETS allows automatic forecasting due to its larger model space.
- Additive ETS models are almost equivalent to the corresponding structural models.
- ETS models have a larger parameter space. Structural models parameters are always non-negative (variances).
- Structural models are much easier to generalize (e.g., add covariates).
- It is easier to handle missing values with structural models.

Structural models in R

```
StructTS(oil, type="level")
StructTS(ausair, type="trend")
StructTS(austourists, type="BSM")

fit <- StructTS(austourists, type = "BSM")
decomp <- cbind(austourists, fitted(fit))
colnames(decomp) <- c("data","level","slope", "seasonal")
plot(decomp, main="Decomposition of International visitor nights")
```



9.2 Linear Gaussian state space models

$$\begin{aligned} \text{Observation equation} \qquad & y_t = \boldsymbol{f}'\boldsymbol{x}_t + \varepsilon_t \\ \text{State equation} \qquad & \boldsymbol{x}_t = \boldsymbol{G}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t \end{aligned}$$

- State vector $\boldsymbol{x}_t$ of length p
- $\boldsymbol{G}$ a $p \times p$ matrix, $\boldsymbol{f}$ a vector of length p
- $\varepsilon_t \sim \text{NID}(0, \sigma^2)$, $\quad \boldsymbol{w}_t \sim \text{NID}(\boldsymbol{0}, \boldsymbol{W})$.

Local level model:

$\boldsymbol{f} = \boldsymbol{G} = 1, \qquad \boldsymbol{x}_t = \ell_t.$

Local linear trend model:

$\boldsymbol{f}' = [1 \ 0]$,

$$\boldsymbol{x}_t = \begin{bmatrix} \ell_t \\ b_t \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{W} = \begin{bmatrix} \sigma_\xi^2 & 0 \\ 0 & \sigma_\zeta^2 \end{bmatrix}$$

Basic structural model

$\boldsymbol{f}' = [1 \ 0 \ 1 \ 0 \ \cdots \ 0], \quad \boldsymbol{W} = \text{diagonal}(\sigma_\xi^2, \sigma_\zeta^2, \sigma_\eta^2, 0, \dots, 0)$

$$\boldsymbol{x}_t = \begin{bmatrix} \ell_t \\ b_t \\ s_{1,t} \\ s_{2,t} \\ s_{3,t} \\ \vdots \\ s_{m-1,t} \end{bmatrix} \boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -1 & -1 & \dots & -1 & -1 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}$$

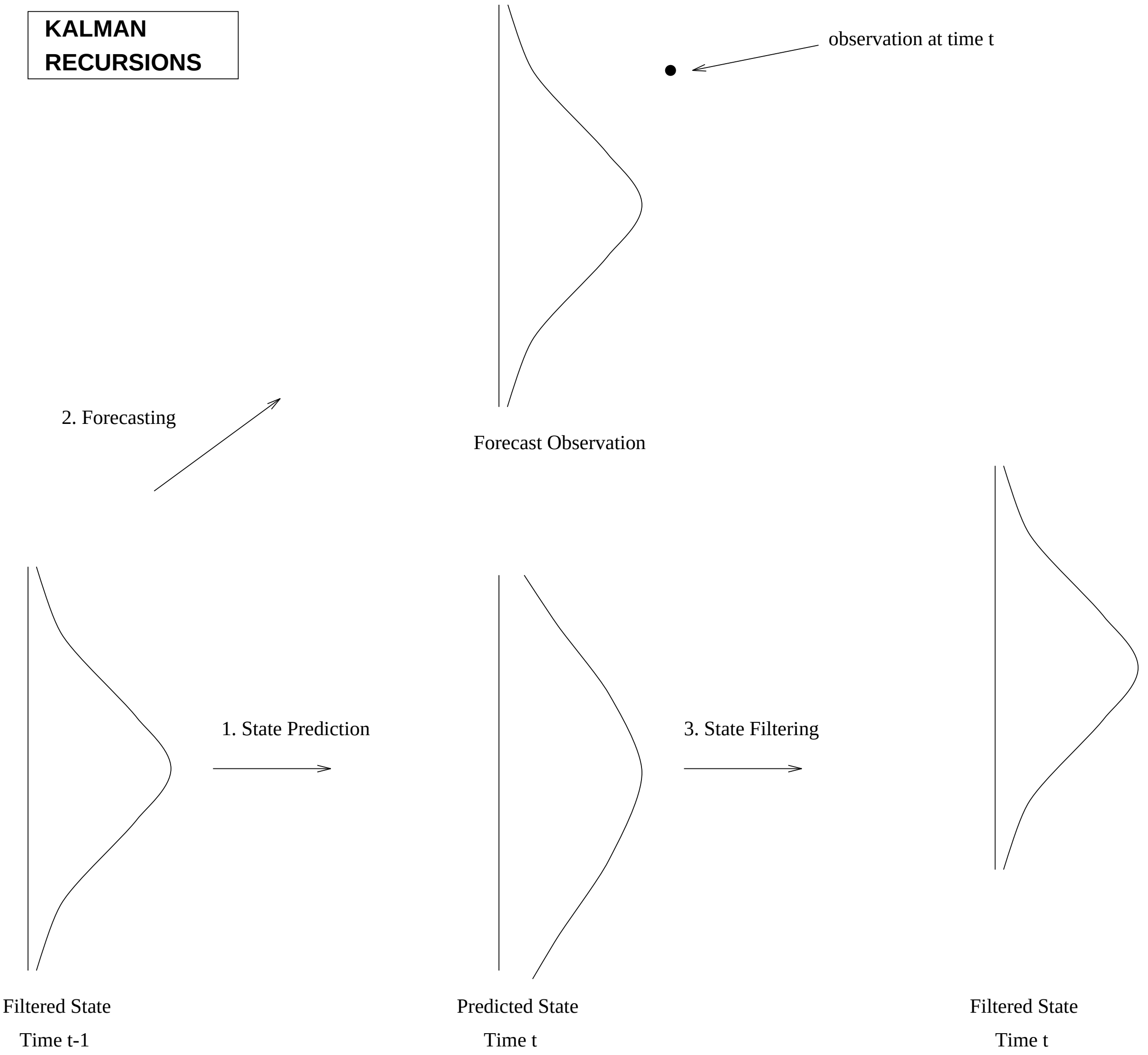
9.3 Kalman filter

Notation:

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{t|t} &= \mathrm{E}[\boldsymbol{x}_t|y_1,\dots,y_t] & \hat{\boldsymbol{P}}_{t|t} &= \mathrm{Var}[\boldsymbol{x}_t|y_1,\dots,y_t]\\
\hat{\boldsymbol{x}}_{t|t-1} &= \mathrm{E}[\boldsymbol{x}_t|y_1,\dots,y_{t-1}] & \hat{\boldsymbol{P}}_{t|t-1} &= \mathrm{Var}[\boldsymbol{x}_t|y_1,\dots,y_{t-1}]\\
\hat{y}_{t|t-1} &= \mathrm{E}[y_t|y_1,\dots,y_{t-1}] & \hat{v}_{t|t-1} &= \mathrm{Var}[y_t|y_1,\dots,y_{t-1}]
\end{aligned}
$$

$$
\begin{aligned}
\textbf{Forecasting:}\quad & \hat{y}_{t|t-1} = \boldsymbol{f}'\hat{\boldsymbol{x}}_{t|t-1}\\
& \hat{v}_{t|t-1} = \boldsymbol{f}'\hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f} + \sigma^2\\
\textbf{Updating or State Filtering:}\quad & \hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}\,\hat{v}_{t|t-1}^{-1}(y_t - \hat{y}_{t|t-1})\\
& \hat{\boldsymbol{P}}_{t|t} = \hat{\boldsymbol{P}}_{t|t-1} - \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}\,\hat{v}_{t|t-1}^{-1}\boldsymbol{f}'\hat{\boldsymbol{P}}_{t|t-1}\\
\textbf{State Prediction}\quad & \hat{\boldsymbol{x}}_{t+1|t} = \boldsymbol{G}\hat{\boldsymbol{x}}_{t|t}\\
& \hat{\boldsymbol{P}}_{t+1|t} = \boldsymbol{G}\hat{\boldsymbol{P}}_{t|t}\boldsymbol{G}' + \boldsymbol{W}
\end{aligned}
$$

Iterate for $t = 1,\dots,T$
Assume we know $\boldsymbol{x}_{1|0}$ and $\boldsymbol{P}_{1|0}$.
Just conditional expectations. So this gives minimum MSE estimates.



Initializing Kalman filter

- Need $\boldsymbol{x}_{1|0}$ and $\boldsymbol{P}_{1|0}$ to get started.
- Common approach for structural models:
  set $\boldsymbol{x}_{1|0} = 0$ and $\boldsymbol{P}_{1|0} = k\boldsymbol{I}$ for a very large k.
- Lots of research papers on optimal initialization choices for Kalman recursions.
- ETS approach was to estimate $\boldsymbol{x}_{1|0}$ and avoid $\boldsymbol{P}_{1|0}$ by assuming error processes identical.
- A random $\boldsymbol{x}_{1|0}$ could be used with ETS models, and then a form of Kalman filter would be required for estimation and forecasting.
- This gives more realistic Forecast intervals.

Local level model

$$\begin{aligned} y_t &= \ell_t + \varepsilon_t & \varepsilon_t &\sim \text{NID}(0,\sigma^2) \\ \ell_t &= \ell_{t-1} + u_t & u_t &\sim \text{NID}(0,q^2) \end{aligned}$$

$$\begin{aligned} \hat{y}_{t|t-1} &= \hat{\ell}_{t-1|t-1} \\ \hat{v}_{t|t-1} &= \hat{p}_{t|t-1} + \sigma^2 \\ \hat{\ell}_{t|t} &= \hat{\ell}_{t-1|t-1} + \hat{p}_{t|t-1}\hat{v}_{t|t-1}^{-1}(y_t - \hat{y}_{t|t-1}) \\ \hat{p}_{t+1|t} &= \hat{p}_{t|t-1}(1 - \hat{v}_{t|t-1}^{-1}\hat{p}_{t|t-1}) + q^2 \end{aligned}$$

Handling missing values

$$\begin{aligned} \textbf{Forecasting:} \quad \hat{y}_{t|t-1} &= \boldsymbol{f}'\hat{\boldsymbol{x}}_{t|t-1} \\ \hat{v}_{t|t-1} &= \boldsymbol{f}'\hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f} + \sigma^2 \\ \textbf{Updating or State Filtering:} \quad \hat{\boldsymbol{x}}_{t|t} &= \hat{\boldsymbol{x}}_{t|t-1} + \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}\hat{v}_{t|t-1}^{-1}(y_t - \hat{y}_{t|t-1}) \\ \hat{\boldsymbol{P}}_{t|t} &= \hat{\boldsymbol{P}}_{t|t-1} - \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}\hat{v}_{t|t-1}^{-1}\boldsymbol{f}'\hat{\boldsymbol{P}}_{t|t-1} \\ \textbf{State Prediction} \quad \hat{\boldsymbol{x}}_{t+1|t} &= \boldsymbol{G}\hat{\boldsymbol{x}}_{t|t} \\ \hat{\boldsymbol{P}}_{t+1|t} &= \boldsymbol{G}\hat{\boldsymbol{P}}_{t|t}\boldsymbol{G}' + \boldsymbol{W} \end{aligned}$$

Multi-step forecasting

Iterate for $t = T+1,\dots,T+h$ starting with $\boldsymbol{x}_{T|T}$ and $\boldsymbol{P}_{T|T}$.

Treat future values as missing.

What's so special about the Kalman filter

- Very general equations for any model in state space format.
- Any model in state space format can easily be generalized.
- Optimal MSE forecasts
- Easy to handle missing values.
- Easy to compute likelihood.

Likelihood calculation

$\boldsymbol{\theta}$ = all unknown parameters
$f_{\boldsymbol{\theta}}(y_t|y_1, y_2, \dots, y_{t-1})$ = one-step forecast density.

$$L(y_1, \dots, y_T; \boldsymbol{\theta}) = \prod_{t=1}^{T} f_{\boldsymbol{\theta}}(y_t|y_1, \dots, y_{t-1})$$

$$\log L = -\frac{T}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\log \hat{v}_{t|t-1} - \frac{1}{2}\sum_{t=1}^{T} e_t^2/\hat{v}_{t|t-1}$$

where $e_t = y_t - \hat{y}_{t|t-1}$.

All terms obtained from Kalman filter equations.

9.4 ARIMA models in state space form

AR(2) model

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + e_t, \qquad e_t \sim \text{NID}(0, \sigma^2)$$

Let $\boldsymbol{x}_t = \begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix}$ and $\boldsymbol{w}_t = \begin{bmatrix} e_t \\ 0 \end{bmatrix}$.

$$\begin{aligned} \text{Then} \qquad y_t &= [1 \;\; 0]\boldsymbol{x}_t \\ \boldsymbol{x}_t &= \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix} \boldsymbol{x}_{t-1} + \boldsymbol{w}_t \end{aligned}$$

- Now in state space form
- We can use Kalman filter to compute likelihood and forecasts.

Alternative formulation

Let $\boldsymbol{x}_t = \begin{bmatrix} y_t \\ \phi_2 y_{t-1} \end{bmatrix}$ and $\boldsymbol{w}_t = \begin{bmatrix} e_t \\ 0 \end{bmatrix}$.

$$\begin{aligned} y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix}\boldsymbol{x}_t \\ \boldsymbol{x}_t &= \begin{bmatrix} \phi_1 & 1 \\ \phi_2 & 0 \end{bmatrix} \boldsymbol{x}_{t-1} + \boldsymbol{w}_t \end{aligned}$$

- Alternative state space form
- We can use Kalman filter to compute likelihood and forecasts.

AR(p) model

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + e_t, \qquad e_t \sim \text{NID}(0, \sigma^2)$$

Let $\boldsymbol{x}_t = \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-p+1} \end{bmatrix}$ and $\boldsymbol{w}_t = \begin{bmatrix} e_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

$$
\begin{aligned}
y_t &= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix} \boldsymbol{x}_t \\
\boldsymbol{x}_t &= \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_{p-1} & \phi_p \\ 1 & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}_{t-1} + \boldsymbol{w}_t
\end{aligned}
$$

ARMA$(1,1)$ model

$$y_t = \phi y_{t-1} + \theta e_{t-1} + e_t, \qquad e_t \sim \text{NID}(0,\sigma^2)$$

Let $\boldsymbol{x}_t = \begin{bmatrix} y_t \\ \theta e_t \end{bmatrix}$ and $\boldsymbol{w}_t = \begin{bmatrix} e_t \\ \theta e_t \end{bmatrix}$.

$$
\begin{aligned}
y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}_t \\
\boldsymbol{x}_t &= \begin{bmatrix} \phi & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}_{t-1} + \boldsymbol{w}_t
\end{aligned}
$$

ARMA(p,q) model

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$$

Let $r = \max(p, q+1)$, $\quad \theta_i = 0,\ q < i \le r, \quad \phi_j = 0,\ p < j \le r.$

$$
\begin{aligned}
y_t &= \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \boldsymbol{x}_t \\
\boldsymbol{x}_t &= \begin{bmatrix} \phi_1 & 1 & 0 & \dots & 0 \\ \phi_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \phi_{r-1} & 0 & \dots & 0 & 1 \\ \phi_r & 0 & 0 & \dots & 0 \end{bmatrix} \boldsymbol{x}_{t-1} + \begin{bmatrix} 1 \\ \theta_1 \\ \vdots \\ \theta_{r-1} \end{bmatrix} e_t
\end{aligned}
$$

The `arima` function in R is implemented using this formulation.

9.5 Kalman smoothing

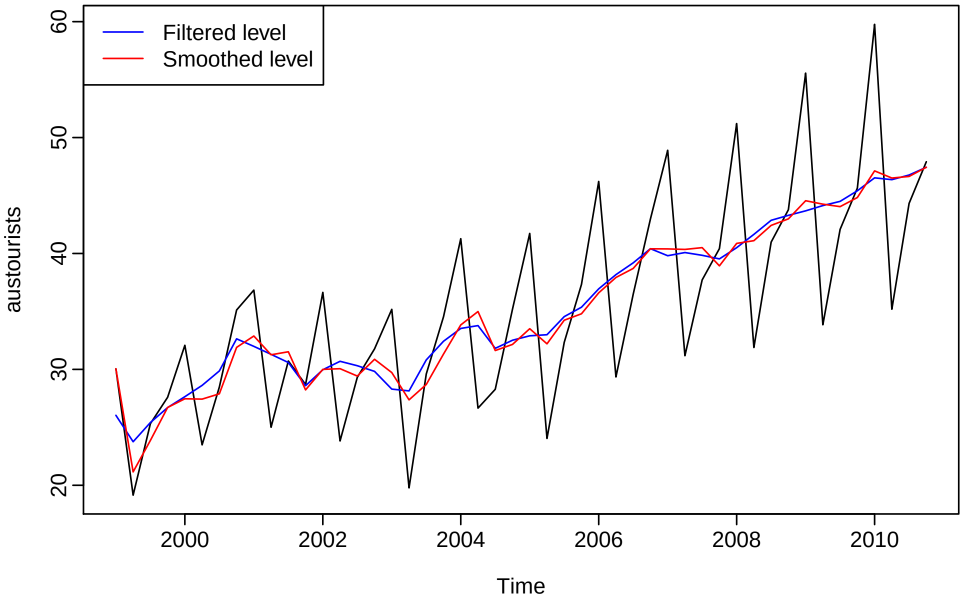
Want estimate of $\boldsymbol{x}_t | y_1, \dots, y_T$ where $t < T$. That is, $\hat{\boldsymbol{x}}_{t|T}$.

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{t|T} &= \hat{\boldsymbol{x}}_{t|t} + \boldsymbol{A}_t \left(\hat{\boldsymbol{x}}_{t+1|T} - \hat{\boldsymbol{x}}_{t+1|t} \right) \\
\hat{P}_{t|T} &= \hat{P}_{t|t} + \boldsymbol{A}_t \left(\hat{P}_{t+1|T} - \hat{P}_{t+1|t} \right) \boldsymbol{A}_t' \\
\text{where} \quad A_t &= \hat{P}_{t|t} \boldsymbol{G}' \left(\hat{P}_{t+1|t} \right)^{-1}.
\end{aligned}
$$

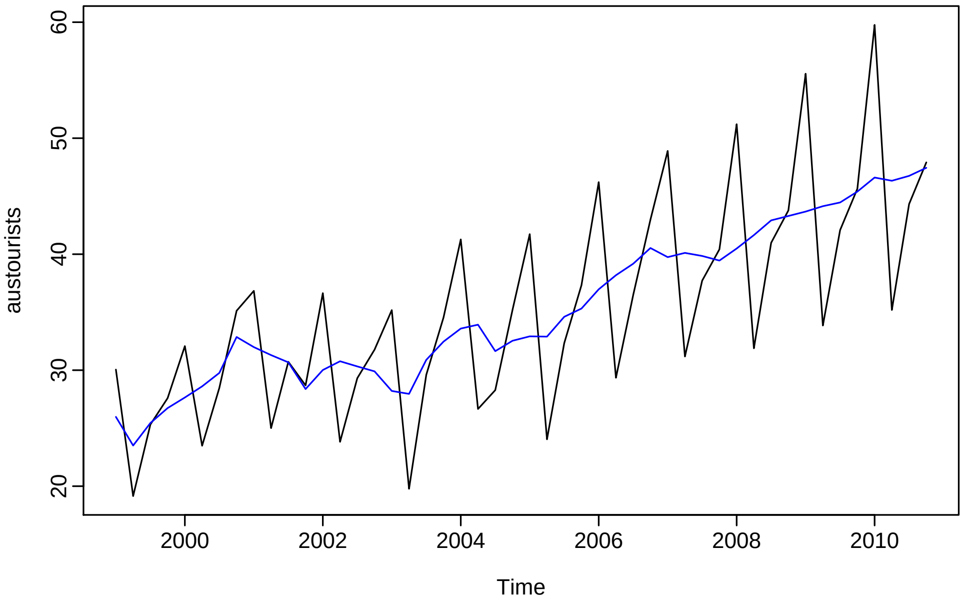
- Uses all data, not just previous data.
- Useful for estimating missing values: $\hat{y}_{t|T} = \boldsymbol{f}' \hat{\boldsymbol{x}}_{t|T}$.
- Useful for seasonal adjustment when one of the states is a seasonal component.

```
fit <- StructTS(austourists, type = "BSM")
sm <- tsSmooth(fit)

plot(austourists)
lines(sm[,1],col=’blue’)
lines(fitted(fit)[,1],col=’red’)
legend("topleft",col=c(’blue’,’red’),lty=1,
  legend=c("Filtered level","Smoothed level"))
```

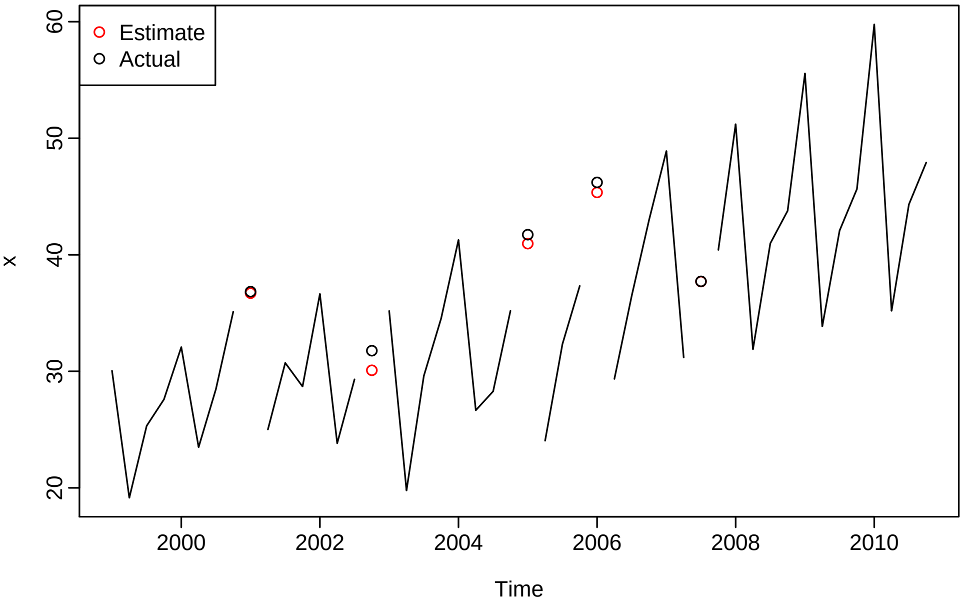
```
fit <- StructTS(austourists, type = "BSM")
sm <- tsSmooth(fit)

plot(austourists)
lines(sm[,1],col=’blue’)
lines(fitted(fit)[,1],col=’red’)
legend("topleft",col=c(’blue’,’red’),lty=1,
 legend=c("Filtered level","Smoothed level"))

plot(austourists)
# Seasonally adjusted data
aus.sa <- austourists - sm[,3]
lines(aus.sa, col=’blue’)
```



```
fit <- StructTS(austourists, type = "BSM")
sm <- tsSmooth(fit)

plot(austourists)

# Seasonally adjusted data
aus.sa <- austourists - sm[,3]
lines(aus.sa,col=’blue’)
```



```
x <- austourists
miss <- sample(1:length(x), 5)
x[miss] <- NA
fit <- StructTS(x, type = "BSM")
sm <- tsSmooth(fit)
estim <- sm[,1]+sm[,3]

plot(x, ylim=range(austourists))
points(time(x)[miss], estim[miss], col='red', pch=1)
points(time(x)[miss], austourists[miss], col='black', pch=1)
legend("topleft", pch=1, col=c(2,1), legend=c("Estimate","Actual"))
```



9.6 Time varying parameter models

Linear Gaussian state space model

$$\begin{aligned} y_t &= \boldsymbol{f}_t{}'\boldsymbol{x}_t + \varepsilon_t, & \varepsilon_t &\sim \mathrm{N}(0, \sigma_t^2) \\ \boldsymbol{x}_t &= \boldsymbol{G}_t\boldsymbol{x}_{t-1} + \boldsymbol{w}_t & \boldsymbol{w}_t &\sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{W}_t) \end{aligned}$$

Kalman recursions:

$$\begin{aligned} \hat{y}_{t|t-1} &= \boldsymbol{f}_t{}'\hat{\boldsymbol{x}}_{t|t-1} \\ \hat{v}_{t|t-1} &= \boldsymbol{f}_t{}'\hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}_t + \sigma_t^2 \\ \hat{\boldsymbol{x}}_{t|t} &= \hat{\boldsymbol{x}}_{t|t-1} + \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}_t\hat{v}_{t|t-1}^{-1}(y_t - \hat{y}_{t|t-1}) \\ \hat{\boldsymbol{P}}_{t|t} &= \hat{\boldsymbol{P}}_{t|t-1} - \hat{\boldsymbol{P}}_{t|t-1}\boldsymbol{f}_t\hat{v}_{t|t-1}^{-1}\boldsymbol{f}_t{}'\hat{\boldsymbol{P}}_{t|t-1} \\ \hat{\boldsymbol{x}}_{t|t-1} &= \boldsymbol{G}_t\hat{\boldsymbol{x}}_{t-1|t-1} \\ \hat{\boldsymbol{P}}_{t|t-1} &= \boldsymbol{G}_t\hat{\boldsymbol{P}}_{t-1|t-1}\boldsymbol{G}_t' + \boldsymbol{W}_t \end{aligned}$$

Local level model with covariate

$$\begin{aligned} y_t &= \ell_t + \beta z_t + \varepsilon_t \\ \ell_t &= \ell_{t-1} + \xi_t \end{aligned}$$

$$\boldsymbol{f}_t{}' = [1\ z_t] \quad \boldsymbol{x}_t = \begin{bmatrix} \ell_t \\ \beta \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{W}_t = \begin{bmatrix} \sigma_\xi^2 & 0 \\ 0 & 0 \end{bmatrix}$$

- Assumes z_t is fixed and known (as in regression)
- Estimate of β is given by $\hat{x}_{T|T}$.
- Equivalent to simple linear regression with time varying intercept.
- Easy to extend to multiple regression with additional terms.

Simple linear regression with time varying parameters

$$\begin{aligned} y_t &= \ell_t + \beta_t z_t + \varepsilon_t \\ \ell_t &= \ell_{t-1} + \xi_t \\ \beta_t &= \beta_{t-1} + \zeta_t \end{aligned}$$

$$\boldsymbol{f}_t{}' = [1\ z_t] \quad \boldsymbol{x}_t = \begin{bmatrix} \ell_t \\ \beta_t \end{bmatrix} \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{W}_t = \begin{bmatrix} \sigma_\xi^2 & 0 \\ 0 & \sigma_\zeta^2 \end{bmatrix}$$

- Allows for a linear regression with parameters that change slowly over time.
- Parameters follow independent random walks.
- Estimates of parameters given by $\hat{x}_{t|t}$ or $\hat{x}_{t|T}$.

Simple linear regression with updating parameters

Same idea can be used to estimate a regression iteratively as new data arrives. Just set

$$\boldsymbol{W}_t = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- Updated parameter estimates given by $\hat{x}_{t|t}$.
- Recursive residuals given by $y_t - \hat{y}_{t|t-1}$.

9.7 Lab Session 9

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Use `StructTS` to forecast each of the time series you used in Lab session 4. How do your results compare to those obtained earlier in terms of their point forecasts and prediction intervals?

   Check the residuals of each fitted model to ensure they look like white noise.

2. In this exercise, you will write your own code for updating regression coefficients using a Kalman filter. We will model quarterly growth rates in US personal consumption expenditure (y) against quarterly growth rates in US real personal disposable income (z). So the model is $y_t = a + bz_t + \varepsilon_t$. The corresponding state space model is

   $$\begin{aligned} y_t &= a_t + b_t z_t + \varepsilon_t \\ a_t &= a_{t-1} \\ b_t &= b_{t-1} \end{aligned}$$

   which can be written in matrix form as follows:

   $$\begin{aligned} y_t &= \boldsymbol{f}_t' \boldsymbol{x}_t + \varepsilon_t, & \varepsilon_t &\sim \mathrm{N}(0, \sigma_t^2) \\ \boldsymbol{x}_t &= \boldsymbol{G}_t \boldsymbol{x}_{t-1} + \boldsymbol{u}_t & \boldsymbol{w}_t &\sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{W}_t) \end{aligned}$$

   where

   $$\boldsymbol{f}_t' = [1\ z_t], \quad \boldsymbol{x}_t = \begin{bmatrix} a_t \\ b_t \end{bmatrix}, \quad \boldsymbol{G}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad \boldsymbol{W}_t = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

   (a) Plot the data using `plot(usconsumption[,1],usconsumption[,2])` and fit a linear regression model to the data using the `lm` function.

   (b) Write some R code to implement the Kalman filter using the above state space model. You can

   (c) Estimate the parameters a and b by applying a Kalman filter and calculating $\hat{x}_{T|T}$. You will need to write your own code to implement the Kalman filter. [The only parameter that has not been specified is σ^2. It makes no difference what value you use in your code. Why?]

   (d) Check that your estimates are identical to the usual OLS estimates obtained with the `lm` function?

   (e) Use your code to obtain the sequence of parameter estimates given by $\hat{\boldsymbol{x}}_{1|1}, \hat{\boldsymbol{x}}_{2|2}, \ldots, \hat{\boldsymbol{x}}_{T|T}$.

   (f) Plot the parameters over time. Does it appear that a model with time-varying parameters would be better?

   (g) How would you estimate σ^2 using your code?

10 Dynamic regression models

10.1 Regression with ARIMA errors

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + e_t,$$

- y_t modeled as function of k explanatory variables $x_{1,t}, \dots, x_{k,t}$.
- Previously, we assumed that e_t was WN.
- Now we want to allow e_t to be autocorrelated.

Example: ARIMA(1,1,1) errors

$$\begin{aligned} y_t &= \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + n_t, \\ (1 - \phi_1 B)(1 - B)n_t &= (1 + \theta_1 B)e_t, \end{aligned}$$

where e_t is white noise .

- Be careful in distinguishing n_t from e_t.
- Only the errors n_t are assumed to be white noise.
- In ordinary regression, n_t is assumed to be white noise and so $n_t = e_t$.

Estimation

If we minimize $\sum n_t^2$ (by using ordinary regression):

- Estimated coefficients $\hat{\beta}_0, \dots, \hat{\beta}_k$ are no longer optimal as some information ignored.
- Statistical tests associated with the model (e.g., t-tests on the coefficients) are incorrect.
- p-values for coefficients usually too small ("spurious regression").
- AIC of fitted models misleading.

Minimizing $\sum e_t^2$ avoids these problems:

- Maximizing likelihood is similar to minimizing $\sum e_t^2$.
- All variables in the model must be stationary.
- If we estimate the model while any of these are non-stationary, the estimated coefficients can be incorrect.
- Difference variables until all stationary.
- If necessary, apply same differencing to all variables.

Model with ARIMA(1,1,1) errors

$$\begin{aligned} y_t &= \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + n_t, \\ (1-\phi_1 B)(1-B)n_t &= (1+\theta_1 B)e_t, \end{aligned}$$

$$\begin{aligned} y'_t &= \beta_1 x'_{1,t} + \cdots + \beta_k x'_{k,t} + n'_t, \\ (1-\phi_1 B)n'_t &= (1+\theta_1 B)e_t, \end{aligned}$$

where $y'_t = y_t - y_{t-1}$, $x'_{t,i} = x_{t,i} - x_{t-1,i}$ and $n'_t = n_t - n_{t-1}$.

Any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model.

$$\begin{aligned} & y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + n_t \\ \text{where} \quad & \phi(B)(1-B)^d N_t = \theta(B)e_t \\ Afterdifferencing: \quad & y'_t = \beta_1 x'_{1,t} + \cdots + \beta_k x'_{k,t} + n'_t. \\ \text{where} \quad & \phi(B)N_t = \theta(B)e_t \\ \text{and} \quad & y'_t = (1-B)^d y_t \end{aligned}$$

Model selection

- To determine ARIMA error structure, first need to calculate n_t.
- We can't get n_t without knowing $\beta_0, \dots, \beta_k$.
- To estimate these, we need to specify ARIMA error structure.

Solution: Begin with a proxy model for the ARIMA errors.

- Assume AR(2) model for for non-seasonal data;
- Assume ARIMA(2,0,0)(1,0,0)$_m$ model for seasonal data.

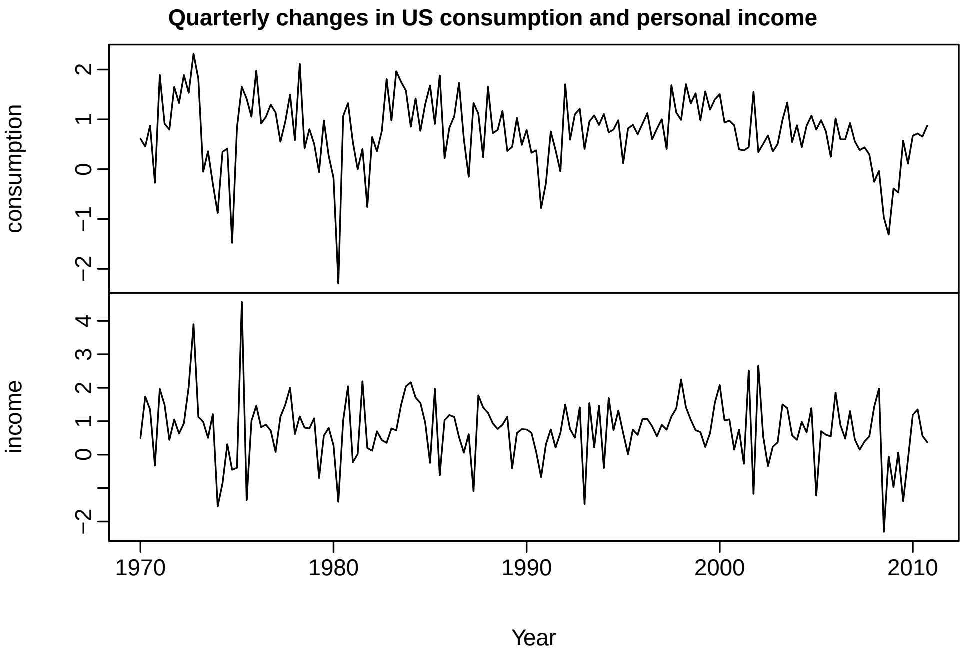
Estimate model, determine better error structure, and re-estimate.

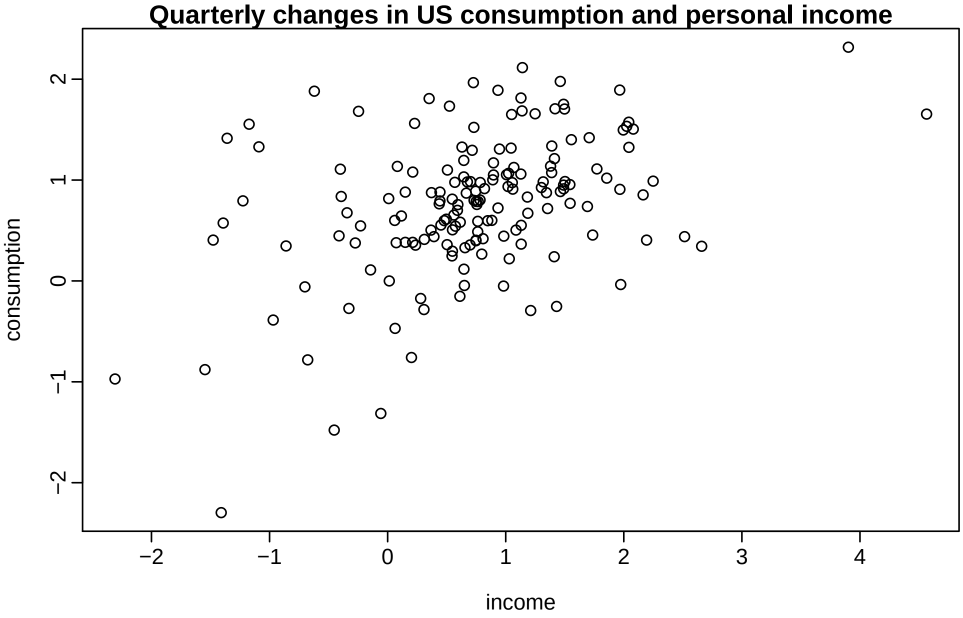
1. Check that all variables are stationary. If not, apply differencing. Where appropriate, use the same differencing for all variables to preserve interpretability.
2. Fit regression model with AR(2) errors for non-seasonal data or ARIMA(2,0,0)(1,0,0)$_m$ errors for seasonal data.
3. Calculate errors (n_t) from fitted regression model and identify ARMA model for them.
4. Re-fit entire model using new ARMA model for errors.
5. Check that e_t series looks like white noise.

Selecting predictors

- AIC can be calculated for final model.
- Repeat procedure for all subsets of predictors to be considered, and select model with lowest AIC value.

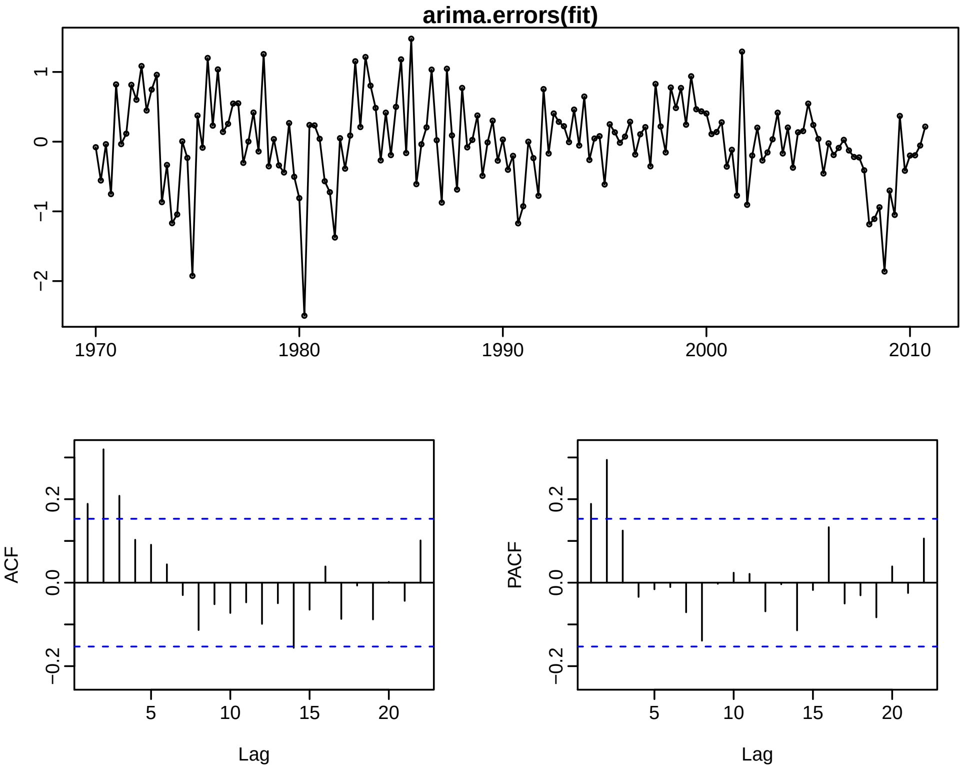
10.2 Example: US personal consumption & income





- No need for transformations or further differencing.
- Increase in income does not necessarily translate into instant increase in consumption (e.g., after the loss of a job, it may take a few months for expenses to be reduced to allow for the new circumstances). We will ignore this for now.
- Try a simple regression with AR(2) proxy model for errors.

```
fit <- Arima(usconsumption[,1], xreg=usconsumption[,2], order=c(2,0,0))
tsdisplay(arima.errors(fit), main="ARIMA errors")
```



- Candidate ARIMA models include MA(3) and AR(2).
- ARIMA(1,0,2) has lowest AIC$_c$ value.
- Refit model with ARIMA(1,0,2) errors.

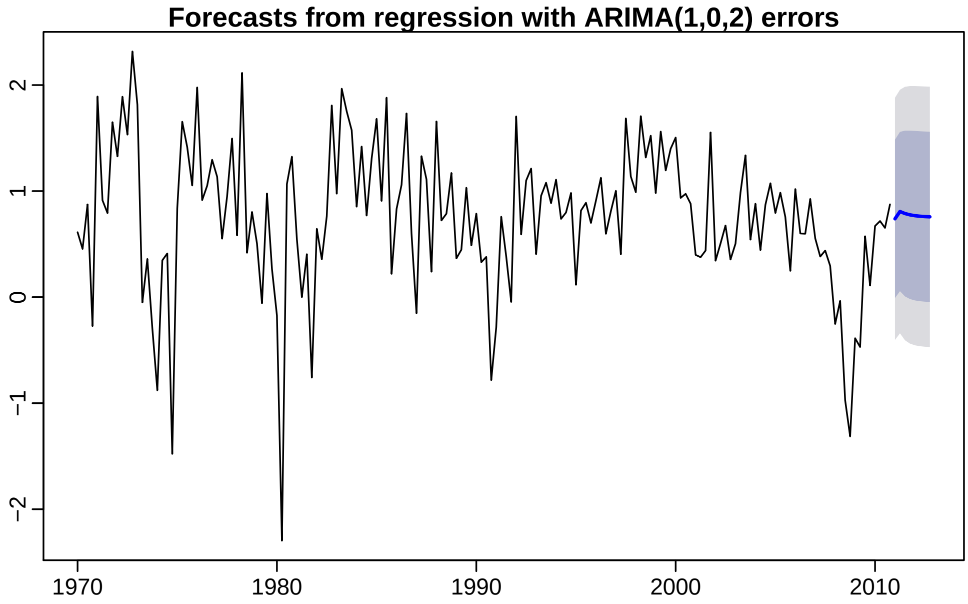
```
> (fit2 <- Arima(usconsumption[,1], xreg=usconsumption[,2], order=c(1,0,2)))

Coefficients:
        ar1     ma1    ma2 intercept usconsumption[,2]
     0.6516 -0.5440 0.2187    0.5750    0.2420
s.e. 0.1468  0.1576 0.0790    0.0951    0.0513

sigma^2 estimated as 0.3396:  log likelihood=-144.27
AIC=300.54   AICc=301.08   BIC=319.14
```

The whole process can be automated:

```
> auto.arima(usconsumption[,1], xreg=usconsumption[,2])
Series: usconsumption[, 1]
ARIMA(1,0,2) with non-zero mean

        ar1     ma1    ma2 intercept usconsumption[,2]
     0.6516 -0.5440 0.2187    0.5750         0.2420
s.e. 0.1468  0.1576 0.0790    0.0951         0.0513

sigma^2 estimated as 0.3396:  log likelihood=-144.27
AIC=300.54   AICc=301.08   BIC=319.14
```

```
> Box.test(residuals(fit2), fitdf=5, lag=10, type="Ljung")
  Box-Ljung test
data:  residuals(fit2)
X-squared = 4.5948, df = 5, p-value = 0.4673
```

```
fcast <- forecast(fit2, xreg=rep(mean(usconsumption[,2]),8), h=8)
plot(fcast, main="Forecasts from regression with ARIMA(1,0,2) errors")
```



10.3 Forecasting

- To forecast a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model and combine the results.
- Forecasts of macroeconomic variables may be obtained from the ABS, for example.
- Separate forecasting models may be needed for other explanatory variables.
- Some explanatory variable are known into the future (e.g., time, dummies).

10.4 Stochastic and deterministic trends

Deterministic trend

$$y_t = \beta_0 + \beta_1 t + n_t$$

where n_t is ARMA process.

Stochastic trend

$$y_t = \beta_0 + \beta_1 t + n_t$$

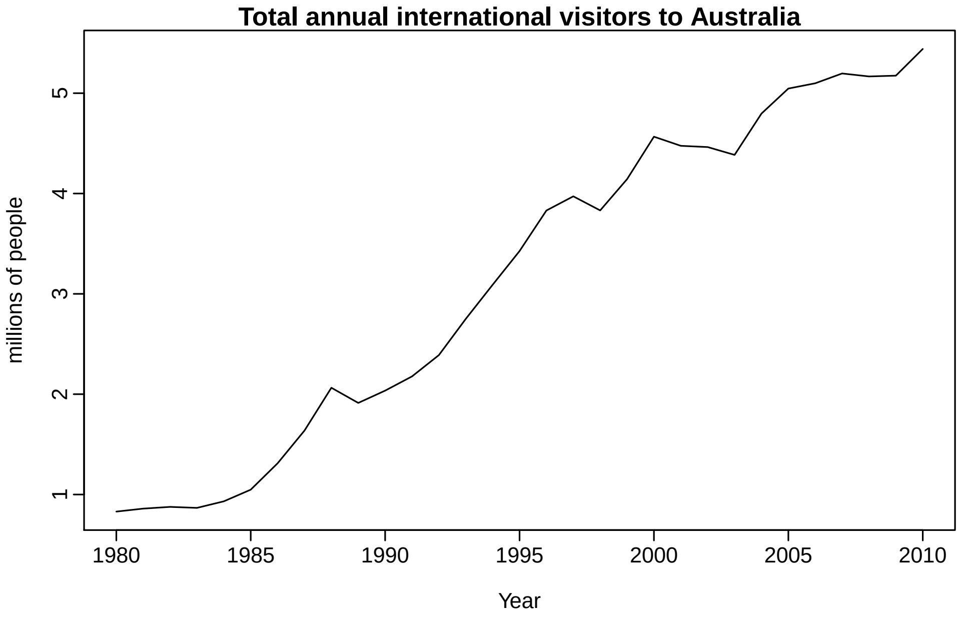
where n_t is ARIMA process with $d \geq 1$.

Difference both sides until n_t is stationary:

$$y'_t = \beta_1 + n'_t$$

where n'_t is ARMA process.

International visitors



Deterministic trend

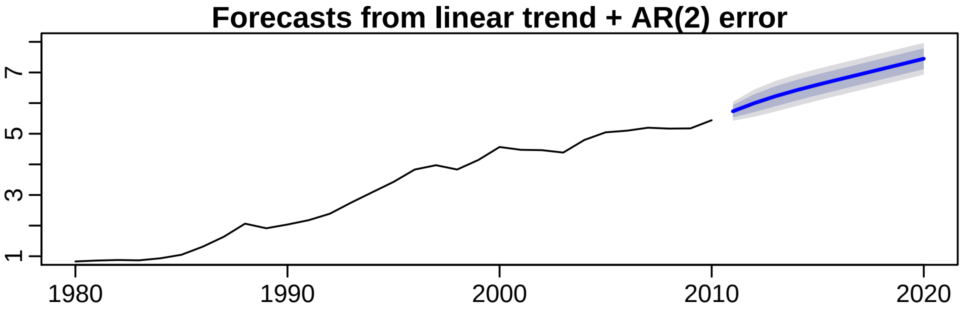
```
> auto.arima(austa,d=0,xreg=1:length(austa))
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1      ar2  intercept  1:length(austa)
      1.0371  -0.3379     0.4173           0.1715
s.e.  0.1675   0.1797     0.1866           0.0102

sigma^2 estimated as 0.02486:  log likelihood=12.7
AIC=-15.4   AICc=-13   BIC=-8.23
```

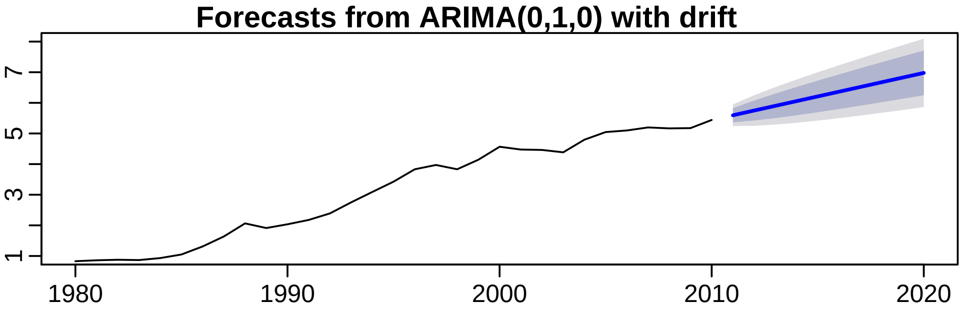
$$\begin{aligned}
y_t &= 0.4173 + 0.1715t + n_t \\
n_t &= 1.0371 n_{t-1} - 0.3379 n_{t-2} + e_t \\
e_t &\sim \text{NID}(0, 0.02486).
\end{aligned}$$

Stochastic trend

```
> auto.arima(austa,d=1)
ARIMA(0,1,0) with drift

Coefficients:
       drift
      0.1538
s.e.  0.0323

sigma^2 estimated as 0.03132:  log likelihood=9.38
AIC=-14.76   AICc=-14.32   BIC=-11.96
```

$$\begin{aligned} y_t - y_{t-1} &= 0.1538 + e_t \\ y_t &= y_0 + 0.1538t + n_t \\ n_t &= n_{t-1} + e_t \\ e_t &\sim \text{NID}(0, 0.03132). \end{aligned}$$





Forecasting with trend

- Point forecasts are almost identical, but forecast intervals differ.
- Stochastic trends have much wider forecast intervals because the errors are non-stationary.
- Be careful of forecasting with deterministic trends too far ahead.

10.5 Periodic seasonality

Fourier terms for seasonality

Periodic seasonality can be handled using pairs of Fourier terms:

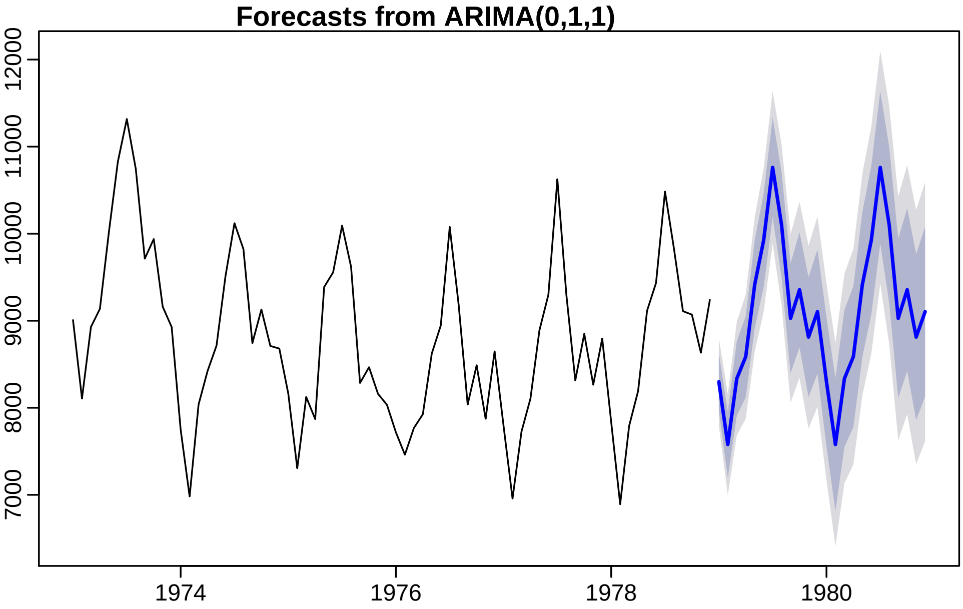
$$s_k(t) = \sin\left(\frac{2\pi kt}{m}\right) \qquad c_k(t) = \cos\left(\frac{2\pi kt}{m}\right)$$

$$y_t = \sum_{k=1}^{K} [\alpha_k s_k(t) + \beta_k c_k(t)] + n_t$$

- n_t is non-seasonal ARIMA process.
- Every periodic function can be approximated by sums of sin and cos terms for large enough K.
- Choose K by minimizing AICc.

US Accidental Deaths

```
fit <- auto.arima(USAccDeaths, xreg=fourier(USAccDeaths, 5),
          seasonal=FALSE)

fc <- forecast(fit, xreg=fourierf(USAccDeaths, 5, 24))

plot(fc)
```



10.6 Dynamic regression models

Sometimes a change in x_t does not affect y_t instantaneously

1. y_t = sales, x_t = advertising.
2. y_t = stream flow, x_t = rainfall.
3. y_t = size of herd, x_t = breeding stock.

- These are dynamic systems with input (x_t) and output (y_t).
- x_t is often a leading indicator.
- There can be multiple predictors.

Lagged explanatory variables

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \dots$.

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \cdots + \nu_k x_{t-k} + n_t$$

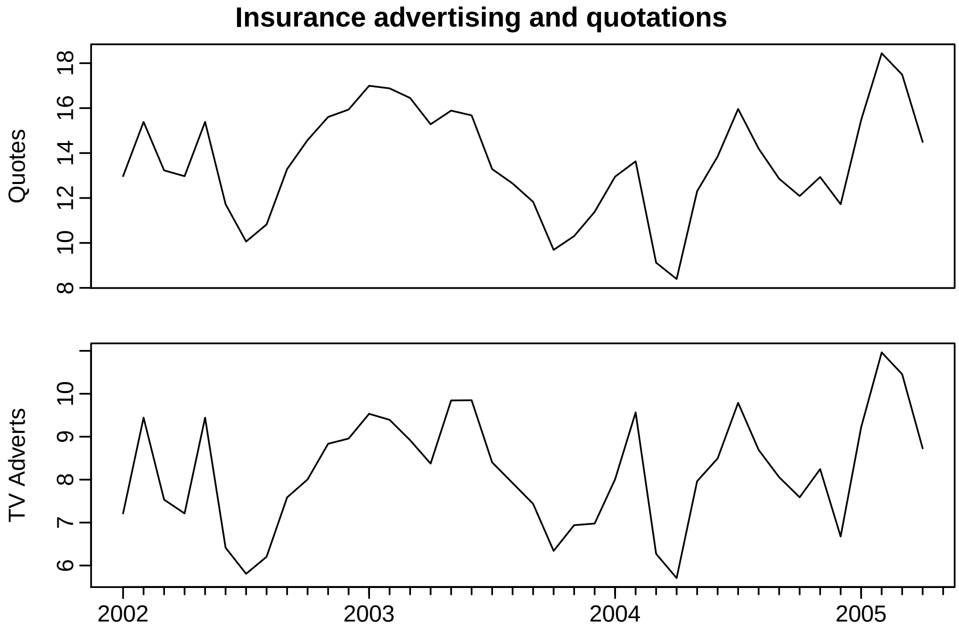
where n_t is an ARIMA process.

Rewrite model as

$$\begin{aligned} y_t &= a + (\nu_0 + \nu_1 B + \nu_2 B^2 + \cdots + \nu_k B^k) x_t + n_t \\ &= a + \nu(B) x_t + n_t. \end{aligned}$$

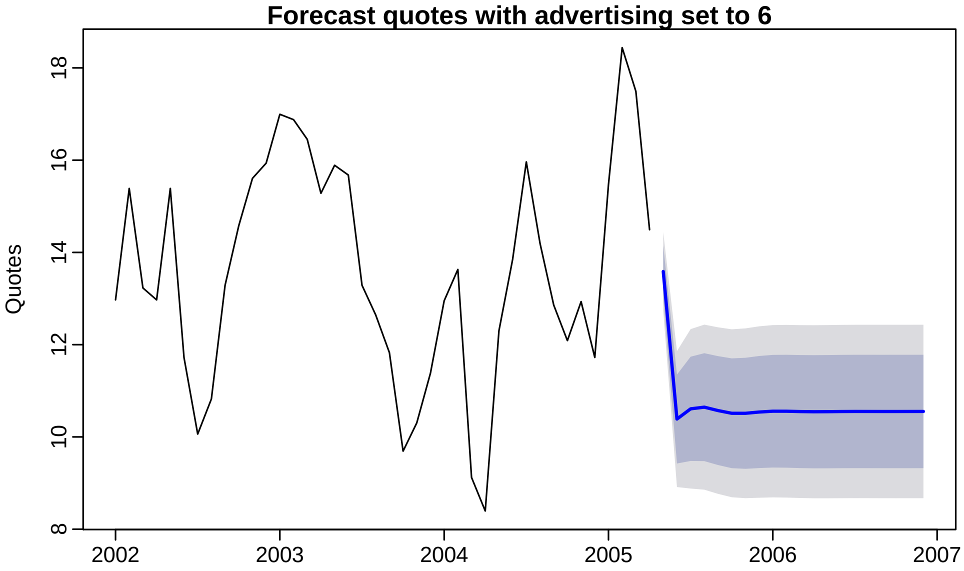
- $\nu(B)$ is called a *transfer function* since it describes how change in x_t is transferred to y_t.
- x can influence y, but y is not allowed to influence x.

Example: Insurance quotes and TV adverts



```
> Advert <- cbind(tv[,2], c(NA,tv[1:39,2]))
> colnames(Advert) <- c("AdLag0","AdLag1")
> fit <- auto.arima(tv[,1], xreg=Advert, d=0)
ARIMA(3,0,0) with non-zero mean

Coefficients:
         ar1      ar2     ar3  intercept  AdLag0  AdLag1
      1.4117  -0.9317  0.3591     2.0393  1.2564  0.1625
s.e.  0.1698   0.2545  0.1592     0.9931  0.0667  0.0591

sigma^2 estimated as 0.1887:  log likelihood=-23.89
AIC=61.78   AICc=65.28   BIC=73.6
```

$$y_t = 2.04 + 1.26x_t + 0.16x_{t-1} + n_t$$
$$n_t = 1.41n_{t-1} - 0.93n_{t-2} + 0.36n_{t-3}$$



```
fc <- forecast(fit, h=20,
  xreg=cbind(c(Advert[40,1],rep(6,19)), rep(6,20)))
plot(fc)
```

10.7 Rational transfer function models

$$y_t = a + \nu(B)x_t + n_t$$

where n_t is an ARMA process. So

$$\phi(B)n_t = \theta(B)e_t \qquad \text{or} \qquad n_t = \frac{\theta(B)}{\phi(B)}e_t = \psi(B)e_t.$$

$$y_t = a + \nu(B)x_t + \psi(B)e_t$$

- ARMA models are rational approximations to general transfer functions of e_t.
- We can also replace $\nu(B)$ by a rational approximation.
- There is no R package for forecasting using a general transfer function approach.

10.8 Lab Session 10

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. For the time series you selected from the retail data set in previous lab sessions:

   (a) Develop an appropriate dynamic regression model with Fourier terms for the seasonality. Use the AIC to select the number of Fourier terms to include in the model. (You may need to use the same Box-Cox transformation you identified previously.)

   (b) Check the residuals of the fitted model. Does the residual series look like white noise?

   (c) Compare the forecasts with those you obtained earlier using alternative models.

2. This exercise concerns the total monthly takings from accommodation and the total room nights occupied at hotels, motels, and guest houses in Victoria, Australia, between January 1980 and June 1995 (Data set `motel`). Total monthly takings are in thousands of Australian dollars; total room nights occupied are in thousands.

   (a) Use the data to calculate the average cost of a night's accommodation in Victoria each month.

   (b) Plot this cost time series against CPI.

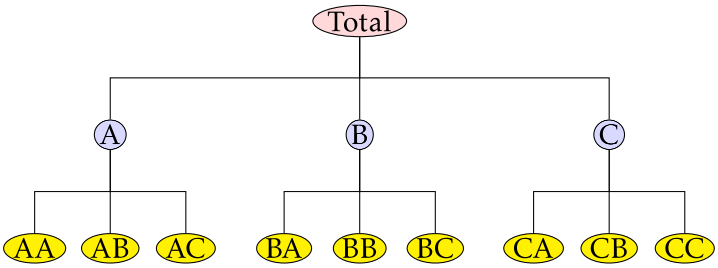
   (c) Produce time series plots of both variables and explain why logarithms of both variables need to be taken before fitting any models.

   (d) Fit an appropriate regression model with ARIMA errors.

   (e) Forecast the average price per room for the next twelve months using your fitted model. (Hint: You will need to produce forecasts of the CPI figures first.)

11 Hierarchical forecasting

11.1 Hierarchical and grouped time series



Examples

- Manufacturing product hierarchies
- Net labour turnover
- Pharmaceutical sales
- Tourism demand by region and purpose

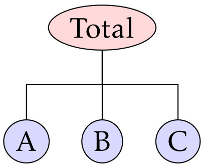
A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.

Example: Pharmaceutical products are organized in a hierarchy under the Anatomical Therapeutic Chemical (ATC) Classification System.

A **grouped time series** is a collection of time series that are aggregated in a number of non-hierarchical ways.

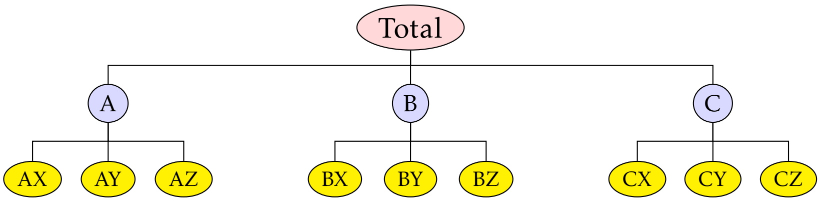
Example: Australian tourism demand is grouped by region and purpose of travel.

Hierarchical data



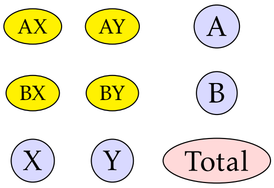
Y_t : observed aggregate of all series at time t.
$Y_{X,t}$: observation on series X at time t.
$\boldsymbol{B}_t$: vector of all series at bottom level in time t.

$$\boldsymbol{Y}_t = [Y_t, Y_{A,t}, Y_{B,t}, Y_{C,t}]' = \underbrace{\begin{pmatrix}1 & 1 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix}Y_{A,t}\\ Y_{B,t}\\ Y_{C,t}\end{pmatrix}}_{\boldsymbol{B}_t} = \boldsymbol{S}\boldsymbol{B}_t$$



$$\boldsymbol{Y}_t = \begin{pmatrix}Y_t\\ Y_{A,t}\\ Y_{B,t}\\ Y_{C,t}\\ Y_{AX,t}\\ Y_{AY,t}\\ Y_{AZ,t}\\ Y_{BX,t}\\ Y_{BY,t}\\ Y_{BZ,t}\\ Y_{CX,t}\\ Y_{CY,t}\\ Y_{CZ,t}\end{pmatrix} = \underbrace{\begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1\\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix}Y_{AX,t}\\ Y_{AY,t}\\ Y_{AZ,t}\\ Y_{BX,t}\\ Y_{BY,t}\\ Y_{BZ,t}\\ Y_{CX,t}\\ Y_{CY,t}\\ Y_{CZ,t}\end{pmatrix}}_{\boldsymbol{B}_t} = \boldsymbol{S}\boldsymbol{B}_t$$

Grouped data



$$\boldsymbol{Y}_t = \begin{pmatrix}Y_t\\ Y_{A,t}\\ Y_{B,t}\\ Y_{X,t}\\ Y_{Y,t}\\ Y_{AX,t}\\ Y_{AY,t}\\ Y_{BX,t}\\ Y_{BY,t}\end{pmatrix} = \underbrace{\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix}Y_{AX,t}\\ Y_{AY,t}\\ Y_{BX,t}\\ Y_{BY,t}\end{pmatrix}}_{\boldsymbol{B}_t} = \boldsymbol{S}\boldsymbol{B}_t$$

11.2 Forecasting framework

Forecasting notation

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$. (They may not add up.)

Hierarchical forecasting methods are of the form:

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{Y}}_n(h)$$

for some matrix $\boldsymbol{P}$.

- $\boldsymbol{P}$ extracts and combines base forecasts $\hat{\boldsymbol{Y}}_n(h)$ to get bottom-level forecasts.
- S adds them up
- Revised reconciled forecasts: $\tilde{\boldsymbol{Y}}_n(h)$.

Bottom-up forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{Y}}_n(h)$$

Bottom-up forecasts are obtained using $\boldsymbol{P} = [\mathbf{0} \mid \mathbf{I}]$, where $\mathbf{0}$ is null matrix and $\mathbf{I}$ is identity matrix.

- $\boldsymbol{P}$ matrix extracts only bottom-level forecasts from $\hat{\boldsymbol{Y}}_n(h)$
- S adds them up to give the bottom-up forecasts.

Top-down forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{Y}}_n(h)$$

Top-down forecasts are obtained using $\boldsymbol{P} = [\boldsymbol{p} \mid \mathbf{0}]$, where $\boldsymbol{p} = [p_1, p_2, \dots, p_{m_K}]'$ is a vector of proportions that sum to one.

- $\boldsymbol{P}$ distributes forecasts of the aggregate to the lowest level series.
- Different methods of top-down forecasting lead to different proportionality vectors $\boldsymbol{p}$.

General properties: bias

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{Y}}_n(h)$$

Assume: base forecasts $\hat{\boldsymbol{Y}}_n(h)$ are unbiased:

$$\mathrm{E}[\hat{\boldsymbol{Y}}_n(h) | \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n] = \mathrm{E}[\boldsymbol{Y}_{n+h} | \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$$

- Let $\hat{\boldsymbol{B}}_n(h)$ be bottom level base forecasts with $\beta_n(h) = \mathrm{E}[\hat{\boldsymbol{B}}_n(h) | \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- Then $\mathrm{E}[\hat{\boldsymbol{Y}}_n(h)] = \boldsymbol{S}\beta_n(h)$.
- We want the revised forecasts to be unbiased: $\mathrm{E}[\tilde{\boldsymbol{Y}}_n(h)] = \boldsymbol{S}\boldsymbol{P}\boldsymbol{S}\beta_n(h) = \boldsymbol{S}\beta_n(h)$.
- Result will hold provided $\boldsymbol{S}\boldsymbol{P}\boldsymbol{S} = \boldsymbol{S}$.
- True for bottom-up, but not for *any* top-down method or middle-out method.

General properties: variance

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{Y}}_n(h)$$

Let variance of base forecasts $\hat{\boldsymbol{Y}}_n(h)$ be given by

$$\Sigma_h = \mathrm{Var}[\hat{\boldsymbol{Y}}_n(h) | \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$$

Then the variance of the revised forecasts is given by

$$\mathrm{Var}[\tilde{\boldsymbol{Y}}_n(h) | \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n] = \boldsymbol{S}\boldsymbol{P}\Sigma_h\boldsymbol{P}'\boldsymbol{S}'.$$

This is a general result for all existing methods.

11.3 Optimal forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\boldsymbol{\beta}_n(h) + \varepsilon_h$.

- $\boldsymbol{\beta}_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean and covariance Σ_h.
- Estimate $\boldsymbol{\beta}_n(h)$ using GLS?

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\boldsymbol{\beta}}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Sigma_h^{\dagger}\boldsymbol{S})^{-1}\boldsymbol{S}'\Sigma_h^{\dagger}\hat{\boldsymbol{Y}}_n(h)$$

- $\Sigma_h^{\dagger}$ is generalized inverse of Σ_h.
- $\mathrm{Var}[\tilde{\boldsymbol{Y}}_n(h)|\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n] = \boldsymbol{S}(\boldsymbol{S}'\Sigma_h^{\dagger}\boldsymbol{S})^{-1}\boldsymbol{S}'$
- **Problem:** Σ_h hard to estimate.

11.4 OLS reconciled forecasts

- Approximate $\Sigma_1^{\dagger}$ by $c\mathbf{I}$.
- Or assume $\varepsilon_h \approx \boldsymbol{S}\varepsilon_{B,h}$ where $\varepsilon_{B,h}$ is the forecast error at bottom level.
- Then $\Sigma_h \approx \boldsymbol{S}\boldsymbol{\Omega}_h\boldsymbol{S}'$ where $\boldsymbol{\Omega}_h = \mathrm{Var}(\varepsilon_{B,h})$.
- If Moore-Penrose generalized inverse used, then

$$(\boldsymbol{S}'\Sigma_h^{\dagger}\boldsymbol{S})^{-1}\boldsymbol{S}'\Sigma_h^{\dagger} = (\boldsymbol{S}'\boldsymbol{S})^{-1}\boldsymbol{S}'.$$

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\boldsymbol{S})^{-1}\boldsymbol{S}'\hat{\boldsymbol{Y}}_n(h)$$

Features

- Covariates can be included in initial forecasts.
- Adjustments can be made to initial forecasts at any level.
- Very simple and flexible method. Can work with *any* hierarchical or grouped time series.
- $\boldsymbol{SPS} = \boldsymbol{S}$ so reconciled forcasts are unbiased.
- Conceptually easy to implement: OLS on base forecasts.
- Weights are independent of the data and of the covariance structure of the hierarchy.

Challenges

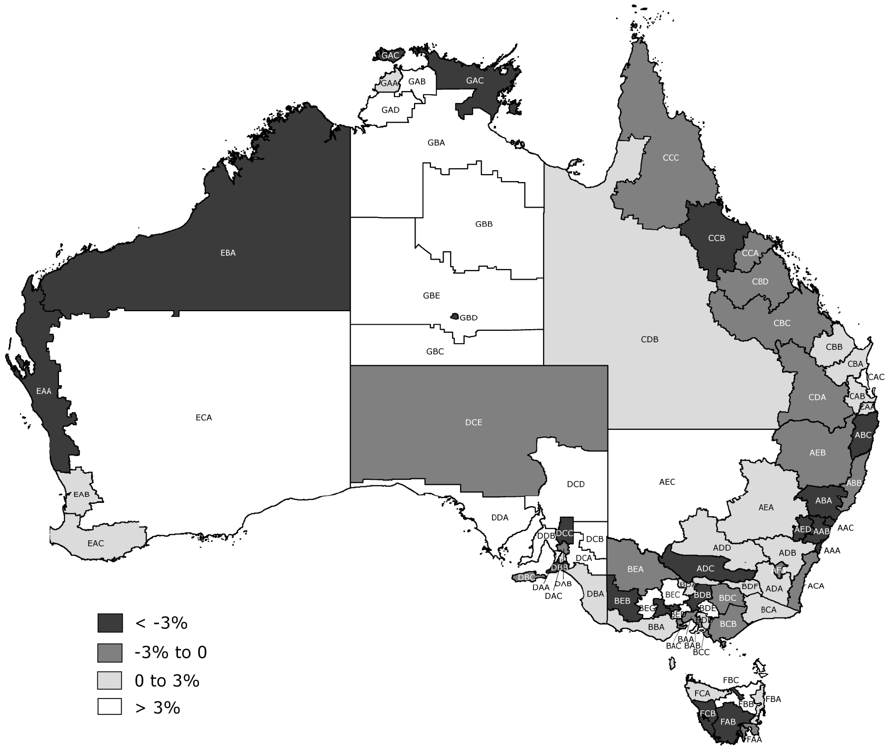
- Computational difficulties in big hierarchies due to size of the $\boldsymbol{S}$ matrix and singular behavior of $(\boldsymbol{S}'\boldsymbol{S})$.
- Need to estimate covariance matrix to produce Forecast intervals.
- Ignores covariance matrix in computing point forecasts.

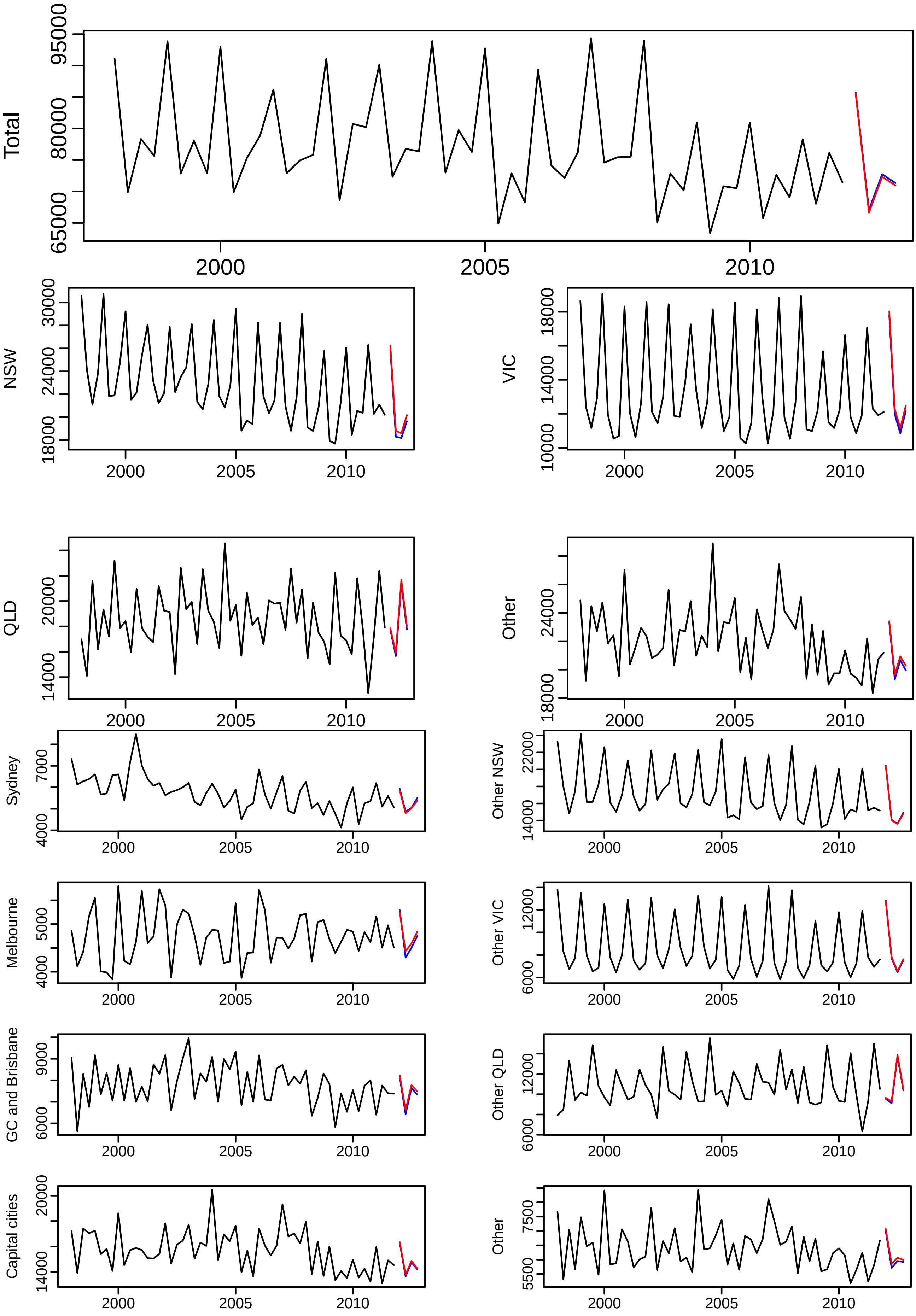
11.5 WLS reconciled forecasts

- Suppose we approximate Σ_1 by its diagonal.
- Let $\boldsymbol{\Lambda} = \left[\text{diagonal}\left(\Sigma_1\right)\right]^{-1}$ contain inverse one-step forecast variances.
$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{Y}}_n(h)$$
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- For large numbers of time series, we need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\boldsymbol{\Lambda}$.

11.6 Application: Australian tourism

Quarterly data: 1998 – 2006.
From: *National Visitor Survey*, based on annual interviews of 120,000 Australians aged 15+, collected by Tourism Research Australia.




Total
65000
80000
95000
2000
2005
2010
NSW
18000
24000
30000
2000
2005
2010
VIC
10000
14000
18000
2000
2005
2010
QLD
14000
20000
2000
2005
2010
Other
18000
24000
2000
2005
2010
Sydney
4000
7000
2000
2005
2010
Other NSW
14000
22000
2000
2005
2010
Melbourne
4000
5000
2000
2005
2010
Other VIC
6000
12000
2000
2005
2010
GC and Brisbane
6000
9000
2000
2005
2010
Other QLD
6000
12000
2000
2005
2010
Capital cities
14000
20000
2000
2005
2010
Other
5500
7500
2000
2005
2010

Forecast evaluation

- Select models using all observations;
- Re-estimate models using first 12 observations and generate 1- to 8-step-ahead forecasts;
- Increase sample size one observation at a time, re-estimate models, generate forecasts until the end of the sample;
- In total 24 1-step-ahead, 23 2-steps-ahead, up to 17 8-steps-ahead for forecast evaluation.

MAPE	$h = 1$	$h = 2$	$h = 4$	$h = 6$	$h = 8$	Average
Top Level: Australia						
Bottom-up	3.79	3.58	4.01	4.55	4.24	4.06
OLS	3.83	3.66	**3.88**	**4.19**	4.25	**3.94**
Scaling (st. dev.)	**3.68**	**3.56**	3.97	4.57	4.25	4.04
Level: States						
Bottom-up	10.70	10.52	10.85	11.46	11.27	11.03
OLS	11.07	10.58	11.13	11.62	12.21	11.35
Scaling (st. dev.)	**10.44**	**10.17**	**10.47**	**10.97**	**10.98**	**10.67**
Level: Zones						
Bottom-up	14.99	14.97	14.98	15.69	15.65	15.32
OLS	15.16	15.06	15.27	15.74	16.15	15.48
Scaling (st. dev.)	**14.63**	**14.62**	**14.68**	**15.17**	**15.25**	**14.94**
Bottom Level: Regions						
Bottom-up	33.12	32.54	32.26	33.74	33.96	33.18
OLS	35.89	33.86	34.26	36.06	37.49	35.43
Scaling (st. dev.)	**31.68**	**31.22**	**31.08**	**32.41**	**32.77**	**31.89**

11.7 Application: Australian labour market

ANZSCO

Australia and New Zealand Standard Classification of Occupations

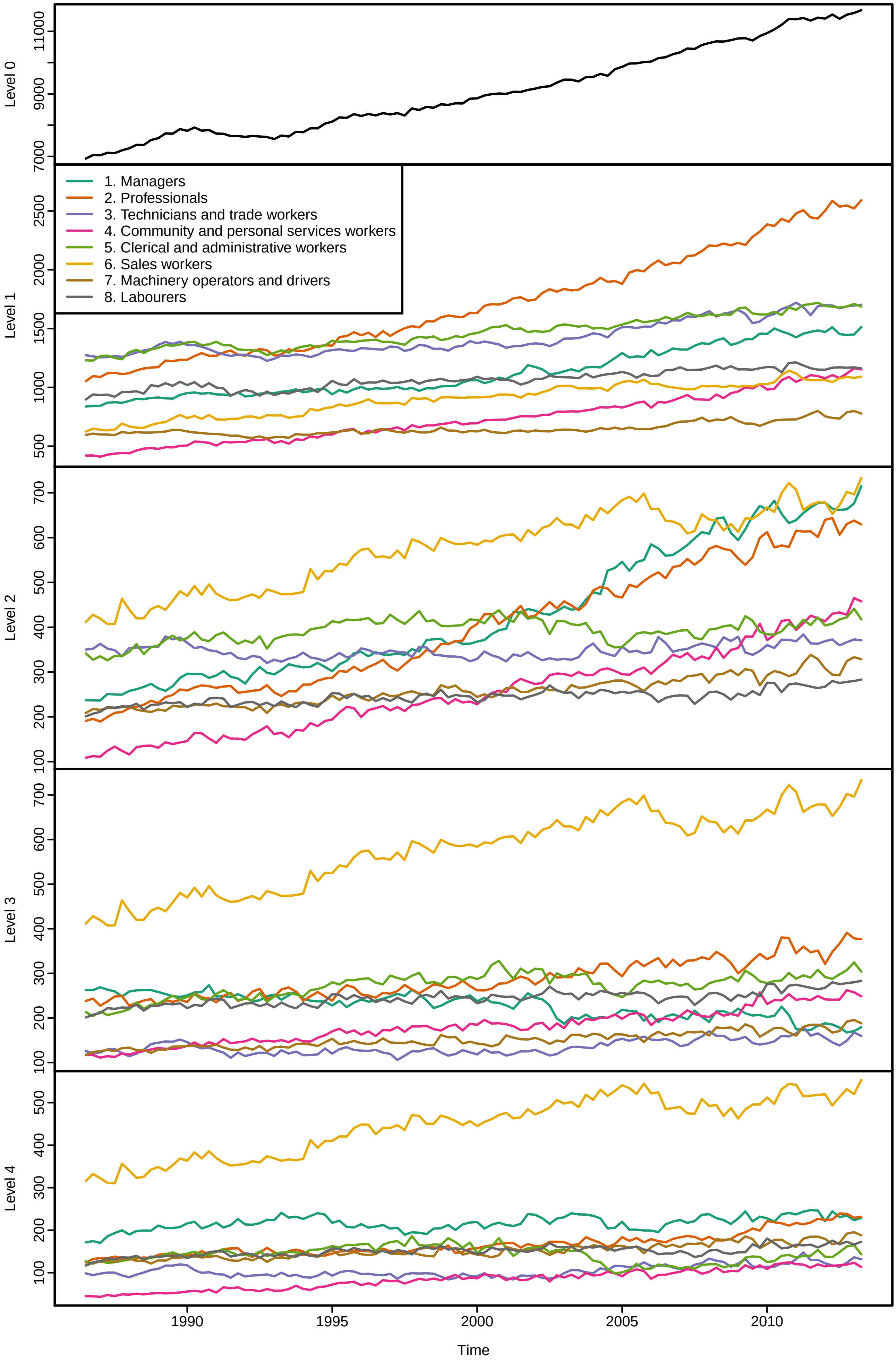
- 8 major groups
  - 43 sub-major groups
    - 97 minor groups
      – 359 unit groups
      * 1023 occupations

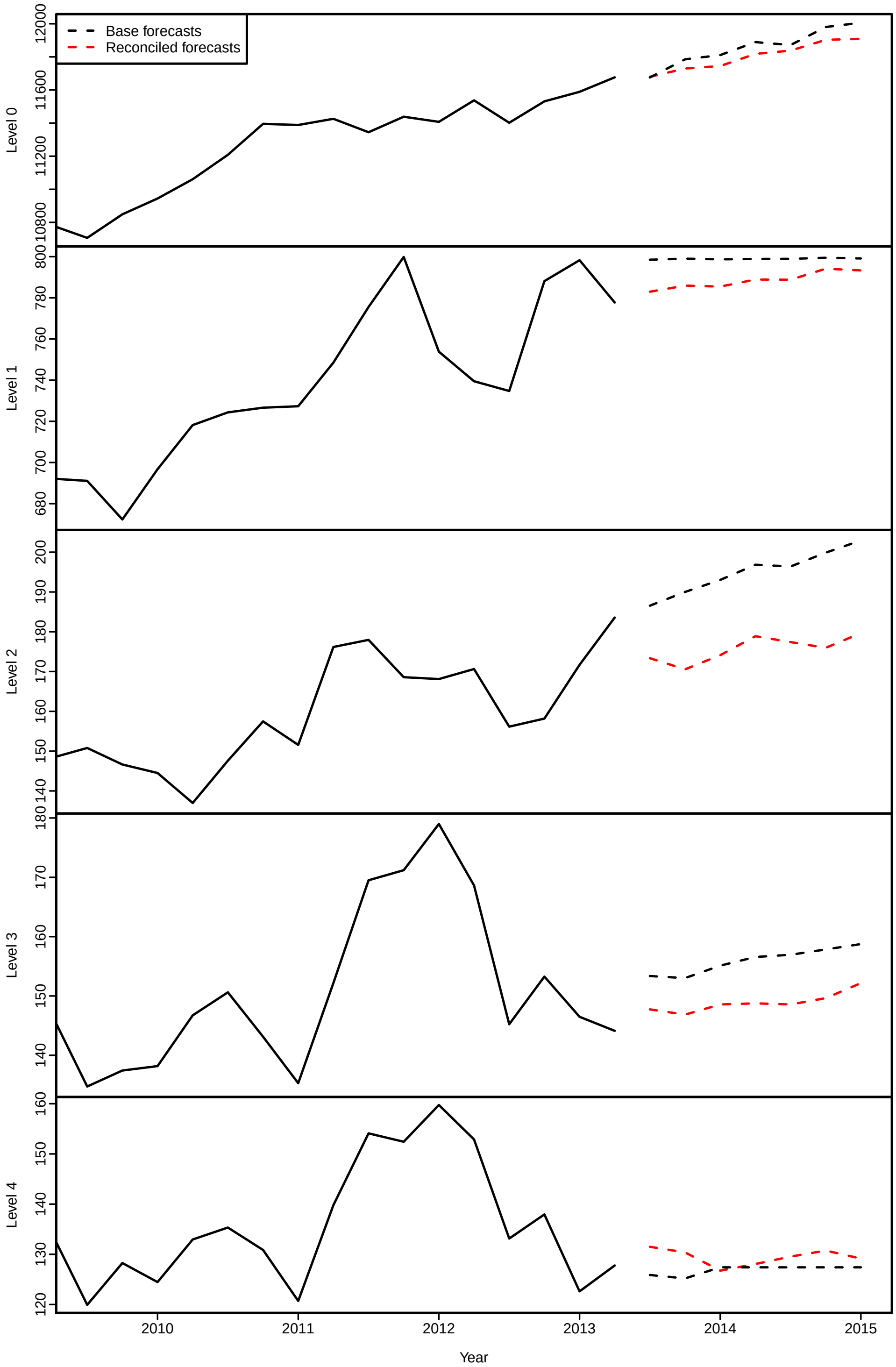
Example: statistician

2 Professionals

22 Business, Human Resource and Marketing Professionals

224 Information and Organisation Professionals
2241 Actuaries, Mathematicians and Statisticians
224113 Statistician


Level 0
7000
9000
11000
Level 1
500
1000
1500
2000
2500
1. Managers
2. Professionals
3. Technicians and trade workers
4. Community and personal services workers
5. Clerical and administrative workers
6. Sales workers
7. Machinery operators and drivers
8. Labourers
Level 2
100
200
300
400
500
600
700
Level 3
100
200
300
400
500
600
700
Level 4
100
200
300
400
500
1990
1995
2000
2005
2010
Time



- Base forecasts from `auto.arima()`
- Largest changes shown for each level
- Lower three panels show largest sub-groups at each level.

Forecast evaluation (rolling origin)

RMSE	$h = 1$	$h = 2$	$h = 3$	$h = 4$	$h = 5$	$h = 6$	$h = 7$	$h = 8$	Average
Top level									
Bottom-up	74.71	102.02	121.70	131.17	147.08	157.12	169.60	178.93	135.29
OLS	**52.20**	**77.77**	**101.50**	**119.03**	138.27	150.75	160.04	166.38	**120.74**
WLS	61.77	86.32	107.26	119.33	**137.01**	**146.88**	**156.71**	**162.38**	122.21
Level 1									
Bottom-up	21.59	27.33	30.81	32.94	35.45	37.10	39.00	40.51	33.09
OLS	21.89	28.55	32.74	35.58	38.82	41.24	43.34	45.49	35.96
WLS	**20.58**	**26.19**	**29.71**	**31.84**	**34.36**	**35.89**	**37.53**	**38.86**	**31.87**
Level 2									
Bottom-up	8.78	10.72	11.79	12.42	13.13	13.61	14.14	14.65	12.40
OLS	9.02	11.19	12.34	13.04	13.92	14.56	15.17	15.77	13.13
WLS	**8.58**	**10.48**	**11.54**	**12.15**	**12.88**	**13.36**	**13.87**	**14.36**	**12.15**
Level 3									
Bottom-up	5.44	6.57	7.17	7.53	7.94	8.27	8.60	8.89	7.55
OLS	5.55	6.78	7.42	7.81	8.29	8.68	9.04	9.37	7.87
WLS	**5.35**	**6.46**	**7.06**	**7.42**	**7.84**	**8.17**	**8.48**	**8.76**	**7.44**
Bottom Level									
Bottom-up	2.35	2.79	3.02	3.15	3.29	3.42	3.54	3.65	3.15
OLS	2.40	2.86	3.10	3.24	3.41	3.55	3.68	3.80	3.25
WLS	**2.34**	**2.77**	**2.99**	**3.12**	**3.27**	**3.40**	**3.52**	**3.63**	**3.13**

11.8 hts package for R

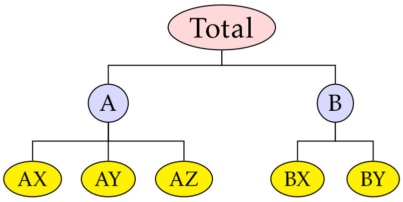
hts: Hierarchical and grouped time series

Methods for analysing and forecasting hierarchical and grouped time series

Version:	4.3
Depends:	forecast (≥ 5.0)
Imports:	SparseM, parallel, utils
Published:	2014-06-10
Author:	Rob J Hyndman, Earo Wang and Alan Lee
Maintainer:	Rob J Hyndman <Rob.Hyndman at monash.edu>
BugReports:	`https://github.com/robjhyndman/hts/issues`
License:	GPL (≥ 2)

Example using R

```
library(hts)

# bts is a matrix containing the bottom level time series
# nodes describes the hierarchical structure
y <- hts(bts, nodes=list(2, c(3,2)))

# Forecast 10-step-ahead using WLS combination method
# ETS used for each series by default
fc <- forecast(y, h=10)
```



forecast.gts function

Usage

```
forecast(object, h,
  method = c("comb", "bu", "mo", "tdgsf", "tdgsa", "tdfp"),
  fmethod = c("ets", "rw", "arima"),
  weights = c("sd", "none", "nseries"),
  positive = FALSE,
  parallel = FALSE, num.cores = 2, ...)
```

Arguments

`object`	Hierarchical time series object of class `gts`.
`h`	Forecast horizon
`method`	Method for distributing forecasts within the hierarchy.
`fmethod`	Forecasting method to use
`positive`	If TRUE, forecasts are forced to be strictly positive
`weights`	Weights used for "optimal combination" method. When weights = "sd", it takes account of the standard deviation of forecasts.
`parallel`	If TRUE, allow parallel processing
`num.cores`	If parallel = TRUE, specify how many cores are going to be used

11.9 Lab Session 11

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. We will reconcile the forecasts for the infant deaths data. The following code can be used. Check that you understand what each step is doing. You will probably need to read the help files for some functions.

```
library(hts)
plot(infantgts)
smatrix(infantgts)

# Forecast 10-step-ahead and reconcile the forecasts
infantforecast <- forecast(infantgts, h=10)

# plot the forecasts including the last ten historical years
plot(infantforecast, include=10)

# Create a matrix of all aggregated time series
allts_infant <- aggts(infantgts)

# Forecast all series using ARIMA models
allf <- matrix(, nrow=10, ncol=ncol(allts_infant))
for(i in 1:ncol(allts_infant))
  allf[,i] <- forecast(auto.arima(allts_infant[,i]), h=10)$mean
allf <- ts(allf, start=2004)

# combine the forecasts with the group matrix to get a gts object
y.f <- combinef(allf, groups = infantgts$groups)

# set up training and testing samples
data <- window(infantgts, start=1933, end=1993)
test <- window(infantgts, start=1994, end=2003)

# Compute forecasts on training data
forecast <- forecast(data, h=10)

# calculate ME, RMSE, MAE, MAPE, MPE and MASE
accuracy.gts(forecast, test)
```

2. How would you measure overall forecast accuracy across the whole collection of time series?

3. Repeat the training/test set analysis using “bottom-up” and “top-down”forecasting. (e.g., set `method="bu"` in the `forecast` function.)

4. Does the “optimal” reconciliation method work best here?

12 Vector autoregressions

Dynamic regression assumes a unidirectional relationship: forecast variable influenced by predictor variables, but not vice versa.

Vector AR allow for feedback relationships. All variables treated symmetrically.

i.e., all variables are now treated as "endogenous".

- Personal consumption may be affected by disposable income, and vice-versa.
- e.g., Govt stimulus package in Dec 2008 increased Christmas spending which increased incomes.

VAR(1)

$$\begin{aligned} y_{1,t} &= c_1 + \phi_{11,1}y_{1,t-1} + \phi_{12,1}y_{2,t-1} + e_{1,t} \\ y_{2,t} &= c_2 + \phi_{21,1}y_{1,t-1} + \phi_{22,1}y_{2,t-1} + e_{2,t} \end{aligned}$$

Forecasts:

$$\begin{aligned} \hat{y}_{1,T+1|T} &= \hat{c}_1 + \hat{\phi}_{11,1}y_{1,T} + \hat{\phi}_{12,1}y_{2,T} \\ \hat{y}_{2,T+1|T} &= \hat{c}_2 + \hat{\phi}_{21,1}y_{1,T} + \hat{\phi}_{22,1}y_{2,T}. \\ \hat{y}_{1,T+2|T} &= \hat{c}_1 + \hat{\phi}_{11,1}\hat{y}_{1,T+1} + \hat{\phi}_{12,1}\hat{y}_{2,T+1} \\ \hat{y}_{2,T+2|T} &= \hat{c}_2 + \hat{\phi}_{21,1}\hat{y}_{1,T+1} + \hat{\phi}_{22,1}\hat{y}_{2,T+1}. \end{aligned}$$

VARs are useful when

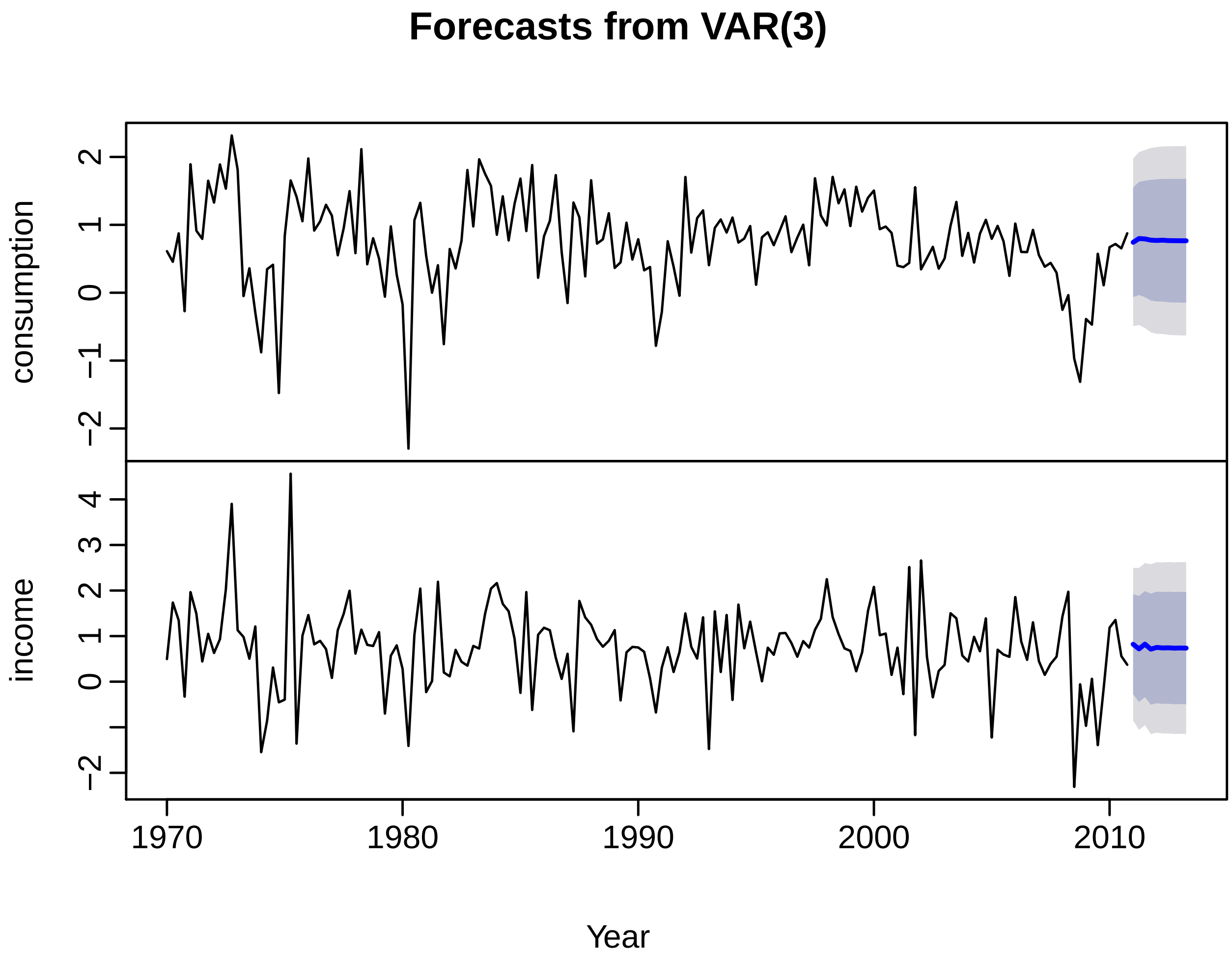
- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

VAR example

```
> ar(usconsumption,order=3)
$ar
, , 1       consumption  income
consumption       0.222  0.0424
income            0.475 -0.2390

, , 2       consumption  income
consumption      0.2001 -0.0977
income           0.0288 -0.1097

, , 3       consumption  income
consumption       0.235 -0.0238
income            0.406 -0.0923

$var.pred
            consumption income
consumption       0.393  0.193
income            0.193  0.735
```

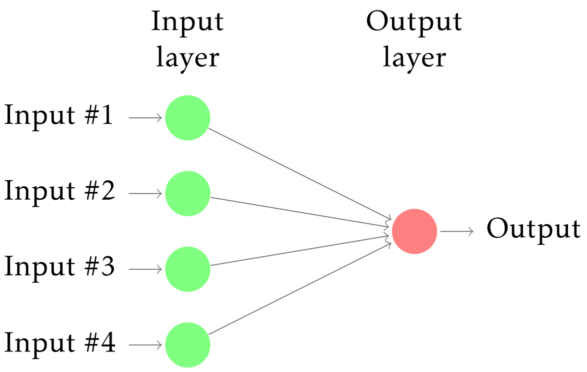
```
> library(vars)

> VARselect(usconsumption, lag.max=8, type="const")$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     5      1      1      5
> var <- VAR(usconsumption, p=3, type="const")
> serial.test(var, lags.pt=10, type="PT.asymptotic")
Portmanteau Test (asymptotic)
data:  Residuals of VAR object var
Chi-squared = 33.3837, df = 28, p-value = 0.2219

> summary(var)
VAR Estimation Results:
=========================
Endogenous variables: consumption, income
Deterministic variables: const
Sample size: 161

Estimation results for equation consumption:
============================================
               Estimate Std. Error t value Pr(>|t|)
consumption.l1  0.22280    0.08580   2.597 0.010326 *
income.l1       0.04037    0.06230   0.648 0.518003
consumption.l2  0.20142    0.09000   2.238 0.026650 *
income.l2      -0.09830    0.06411  -1.533 0.127267
consumption.l3  0.23512    0.08824   2.665 0.008530 **
income.l3      -0.02416    0.06139  -0.394 0.694427
const           0.31972    0.09119   3.506 0.000596 ***
```

```
Estimation results for equation income:
=======================================
               Estimate Std. Error t value Pr(>|t|)
consumption.l1  0.48705    0.11637   4.186 4.77e-05 ***
income.l1      -0.24881    0.08450  -2.945 0.003736 **
consumption.l2  0.03222    0.12206   0.264 0.792135
income.l2      -0.11112    0.08695  -1.278 0.203170
consumption.l3  0.40297    0.11967   3.367 0.000959 ***
income.l3      -0.09150    0.08326  -1.099 0.273484
const           0.36280    0.12368   2.933 0.003865 **
--
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Correlation matrix of residuals:
            consumption income
consumption      1.0000 0.3639
income           0.3639 1.0000
fcst <- forecast(var)
plot(fcst, xlab="Year")
```



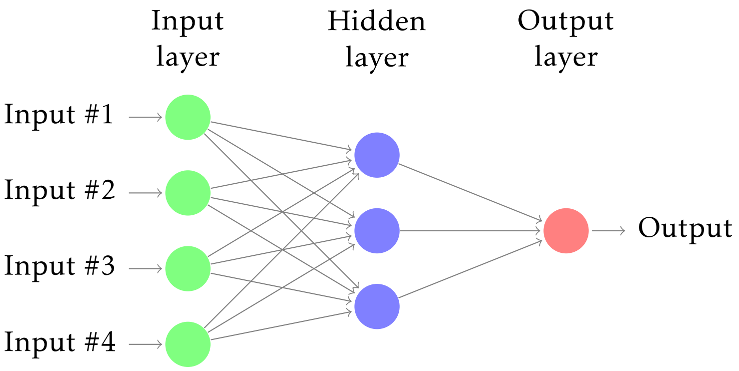
13 Neural network models

Simplest version: linear regression



- Coefficients attached to predictors are called “weights”.
- Forecasts are obtained by a linear combination of inputs.
- Weights selected using a “learning algorithm” that minimises a “cost function”.

Nonlinear model with one hidden layer



- A **multilayer feed-forward network** where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using linear combination.
- Result modified by nonlinear function before being output.

Inputs to hidden neuron j linearly combined:

$$z_j = b_j + \sum_{i=1}^{4} w_{i,j} x_i.$$

Modified using nonlinear function such as a sigmoid:

$$s(z) = \frac{1}{1+e^{-z}},$$

This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

- Weights take random values to begin with, which are then updated using the observed data.
- There is an element of randomness in the predictions. So the network is usually trained several times using different random starting points, and the results are averaged.
- Number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p,k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p,0$) model is equivalent to an ARIMA($p,0,0$) model but without stationarity restrictions.
- Seasonal NNAR(p,P,k): inputs $(y_{t-1}, y_{t-2}, \dots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p,P,0)_m$ model is equivalent to an ARIMA$(p,0,0)(P{,}0{,}0)_m$ model but without stationarity restrictions.

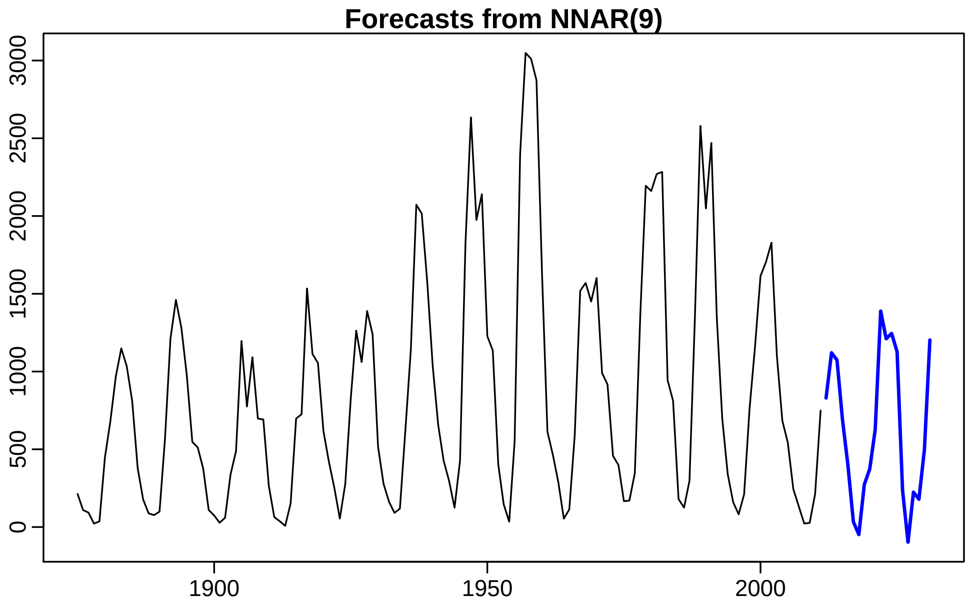
NNAR models in R

- The `nnetar()` function fits an NNAR$(p,P,k)_m$ model.
- If p and P are not specified, they are automatically selected.
- For non-seasonal time series, default p = optimal number of lags (according to the AIC) for a linear AR(p) model.
- For seasonal time series, defaults are $P = 1$ and p is chosen from the optimal linear model fitted to the seasonally adjusted data.
- Default $k = (p+P+1)/2$ (rounded to the nearest integer).

Sunspots

- Surface of the sun contains magnetic regions that appear as dark spots.
- These affect the propagation of radio waves and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties.
- Sunspots follow a cycle of length between 9 and 14 years.

NNAR model for sunspots

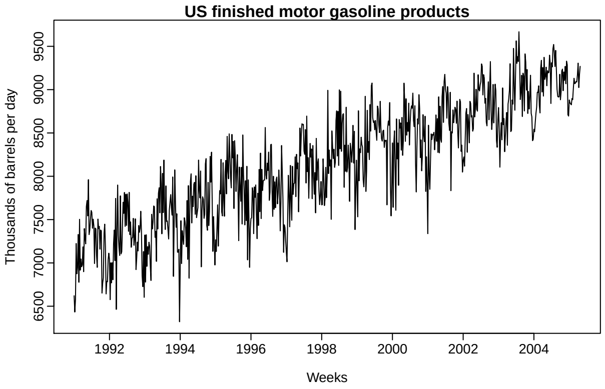


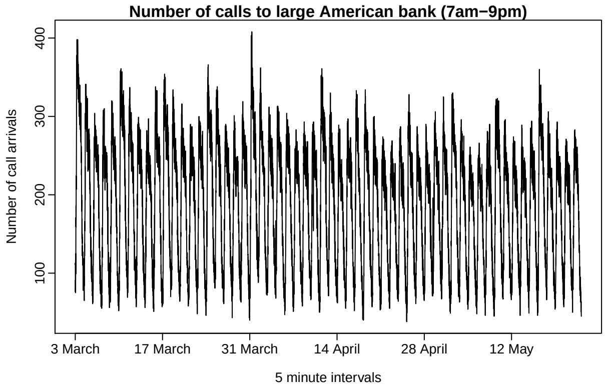
```
fit <- nnetar(sunspotarea)
plot(forecast(fit,h=20))
```

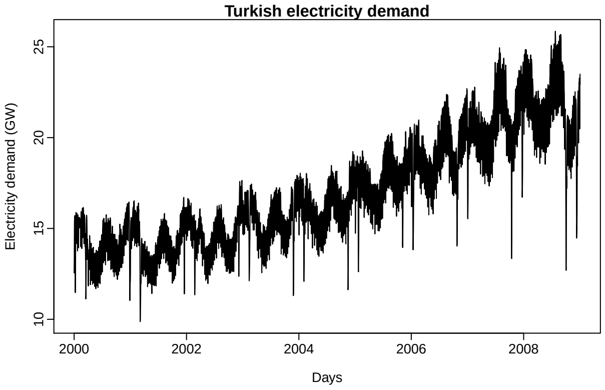
To restrict to positive values:

```
fit <- nnetar(sunspotarea,lambda=0)
plot(forecast(fit,h=20))
```

14 Forecasting complex seasonality


US finished motor gasoline products


Number of calls to large American bank (7am−9pm)

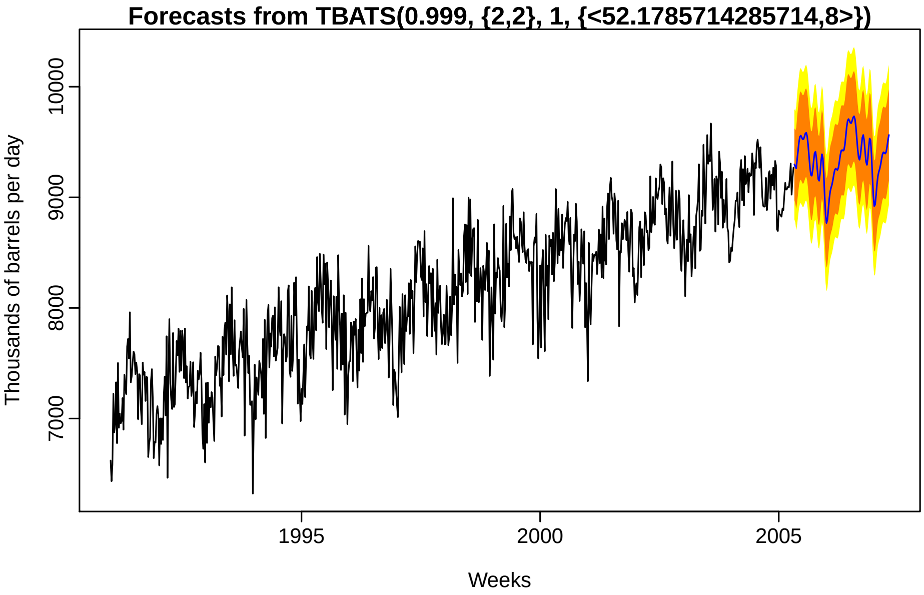

Turkish electricity demand

14.1 TBATS model

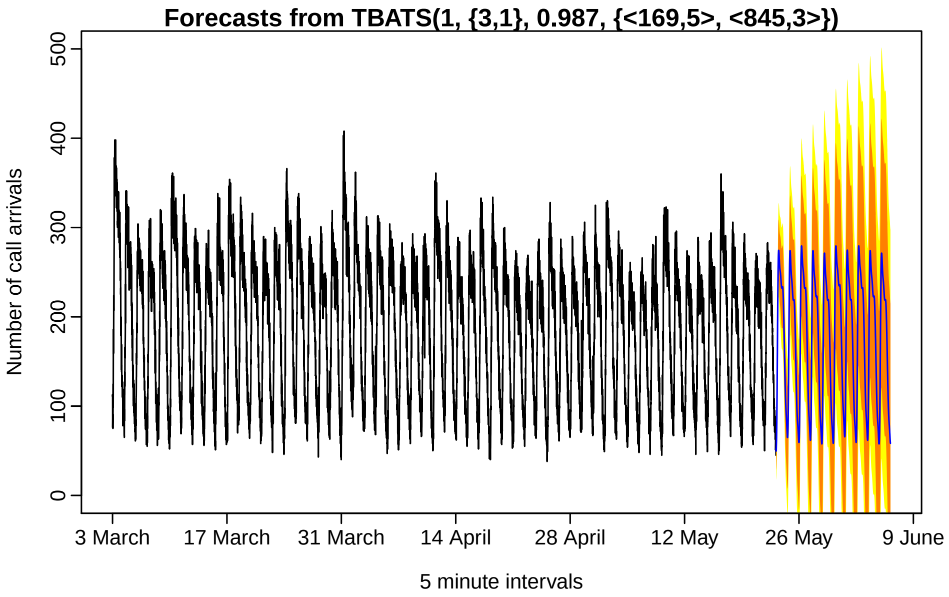
Trigonometric terms for seasonality
Box-Cox transformations for heterogeneity
ARMA errors for short-term dynamics
Trend (possibly damped)
Seasonal (including multiple and non-integer periods)

$$
\begin{aligned}
y_t &= \text{observation at time } t\\
y_t^{(\omega)} &= \begin{cases}(y_t^{\omega}-1)/\omega & \text{if } \omega \neq 0;\\ \log y_t & \text{if } \omega = 0.\end{cases}\\
y_t^{(\omega)} &= \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t\\
\ell_t &= \ell_{t-1} + \phi b_{t-1} + \alpha d_t\\
b_t &= (1-\phi)b + \phi b_{t-1} + \beta d_t\\
d_t &= \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t\\
s_t^{(i)} &= \sum_{j=1}^{k_i} s_{j,t}^{(i)}\\
s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos\lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin\lambda_j^{(i)} + \gamma_1^{(i)} d_t\\
s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin\lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos\lambda_j^{(i)} + \gamma_2^{(i)} d_t
\end{aligned}
$$

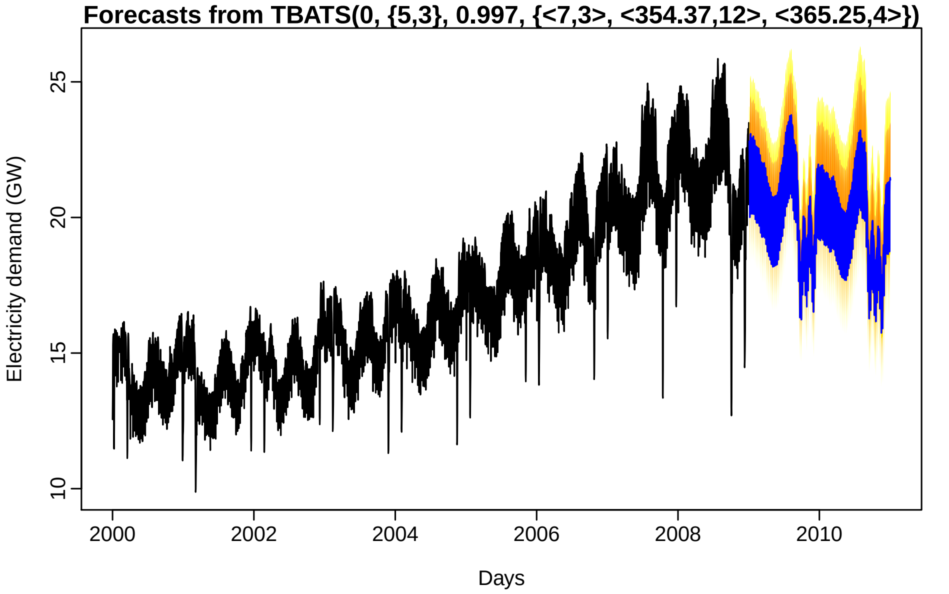
Examples



```
fit <- tbats(gasoline)
fcast <- forecast(fit)
plot(fcast)
```



```
fit <- tbats(callcentre)
fcast <- forecast(fit)
plot(fcast)
```



```
fit <- tbats(turk)
fcast <- forecast(fit)
plot(fcast)
```

14.2 Lab Session 12

Before doing any exercises in R, load the **fpp** package using `library(fpp)`.

1. Use the `tbats` function to model the `visitors` time series.
   (a) Check the residuals and produce forecasts.
   (b) Does this completely automated approach work for these data?
   (c) Have you saved any degrees of freedom by using Fourier terms rather than seasonal differencing?

2. The following code will read weekly data on US finished motor gasoline products supplied (thousands of barrels per day):

```
gas <- read.csv("http://robjhyndman.com/data/gasoline.csv")[,1]
gas <- ts(gas, start=1991+31/365.25, frequency = 365.25/7)
```

   (a) Fit a `tbats` model to these data.
   (b) Check the residuals and produce forecasts.
   (c) Could you model these data using any of the other methods we have considered in this course?

3. Experiment with using `nnetar()` on some of the data considered in previous lab sessions.

4. Over this course, you have developed several models for the retail data. The last exercise is to use cross-validation to objectively compare the models you have developed. Compute cross-validated MAE values for each of the time series models you have considered. It will take some time to run, so perhaps leave it running overnight and check the results the next morning.

Congratulations on finishing the forecasting course! I hope you have learned things that will be useful in forecasting whatever it is you want to forecast.

For more information about forecasting resources:

`robjhyndman.com/hyndsight/forecasting/`

`OTexts.org/fpp`