Module 1: Fundamental Concepts of Time Series

MATH1307 Forecasting

Prepared by: Dr. Haydar Demirhan

Introduction

To draw statistical inferences about the events observed sequentially predetermined time points, we need to use time series analysis method. The main characteristic of such datasets is that they contain a serial correlation within the set of observations.

We observe time series data in a wide range of analytic applications from seismography to health. Some examples would be minute-by-minute stock prices, hourly temperatures at a weather station, daily numbers of arrivals at a medical clinic, weekly sales of a product, monthly unemployment figures for a region, quarterly imports of a country, and an annual turnover of a company. As a recent example from time series research, Miwa Fukino, Yoshito Hirata, and Kazuyuki Aihara from University of Tokyo

(https://sinews.siam.org/Details-Page/music-visualized-by-nonlinear-time-series-analysis) proposed a new method based on nonlinear time series analysis analyse music's emotional effect.

The time span where the series is observed can be either discrete or continuous. In this course, we will focus on the analysis of series observed at regular intervals over time.

In this module, we will

- start with examples of time series and their main visualisation,
- introduce the model-building strategy that we will follow through the semester,
- basics of forecasting methodology,
- focus on the fundamental concepts in terms of
  - Time Series and Stochastic Processes,
  - Means, Variances, and Covariances, and
  - Stationarity.

Packages to be used

In this course, we will mainly use the package `forecast`. In addition to this package, we will utilize some other packages. You need to install and make sure that there are working properly.

The main packages are

- `forecast`
- `expsmooth`
- `TSA`
- `dynlm`
- `x12`

The package

`dLagM`

has been developed for this course by Dr Haydar Demirhan.

Auxiliary ones are

- `Hmisc`
- `car`
- `AER`

You can install these packages calling

```
install.packages("dLagM", "forecast" , "expsmooth" , "TSA" ,
                 "dylnm" , "Hmisc" , "car", "AER")
```

Time series objects in R

To implement time series analysis and forecasting methods in R, the first thing to consider is the class of objects we send to related functions. We use the function `ts()` to convert other objects to time series objects. The arguments we use with this function are

- `start` used to input the start date of time series
- `end` used to input the end data of the time series. If it is not specified `ts()` function assigns it automatically according to the length of the series.
- `frequency` used to specify the period of seasonal time series. Use of this argument is the most critical part of the `ts()` function as it changes according to the structure of the input data.

Let's take a look at the following examples. In the first example below, *annual* series has entered as a column in the `sampleData1.csv` file.

```
sampleData1 = read_csv("/Users/haydardemirhan/Documents/MATH1307_F
  orecasting/
                       notes/Module 1/sampleData1.csv", col_names
   = FALSE,
                       col_types = cols(X1 = col_number()))
class(sampleData1)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
# The series has loaded as a data.frame object which needs
# to be converted to the ts object.
head(sampleData1)
```

```
## # A tibble: 2 x 1
##      X1
##   <dbl>
## 1  1307
## 2     1
```

```
sampleData1.ts = ts(sampleData1, start = 1990)
# Say starting year is 1990
class(sampleData1.ts)
```

```
## [1] "ts"
```

```
# Now it is a ts object
head(sampleData1)
```

```
## # A tibble: 2 x 1
##      X1
##   <dbl>
## 1  1307
## 2     1
```

```
sampleData1
```

```
## # A tibble: 2 x 1
##      X1
##   <dbl>
## 1  1307
## 2     1
```

The second example demonstrates converting a monthly series given in the `sampleData2.csv` file. Here the series is given in the second column and dates (in monthly format) are in the first column.

```
sampleData2 = read_csv("/Users/haydardemirhan/Documents/MATH1307_F
  orecasting/notes/Module 1/sampleData2.csv")
```

```
## Parsed with column specification:
## cols(
##   Date = col_character(),
##   Data = col_double()
## )
```

```
class(sampleData2)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
# The series has loaded as a data.frame object which needs
# to be converted to the ts object.
head(sampleData2)
```

```
## # A tibble: 6 x 2
##   Date     Data
##   <chr>   <dbl>
## 1 1953-01  24.0
## 2 1953-02  29.0
## 3 1953-03  28.8
## 4 1953-04  56.8
## 5 1953-05  16.9
## 6 1953-06  18.8
```

```
sampleData2.ts = ts(sampleData2$Data, start = c(1953,1), frequency
   = 12)
# Don't need to specify the end date
class(sampleData2.ts)
```

```
## [1] "ts"
```

```
# Now it is a ts object
head(sampleData2.ts)
```

```
##        Jan   Feb   Mar   Apr   May   Jun
## 1953 24.03 29.01 28.83 56.78 16.88 18.85
```

```
sampleData2.ts
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oc
t   Nov
## 1953 24.03 29.01 28.83 56.78 16.88 18.85  3.15  4.41  5.15 27.6
6 33.94
## 1954 27.90 52.48 41.89 32.76 25.61 12.84 10.56 13.13  4.36 21.6
4 25.75
## 1955 45.25 22.97 25.28 27.77 19.34 34.07 14.97 14.48 12.07 37.9
3 49.21
## 1956 51.20 16.31 24.87 68.07 34.11 17.82 11.26  5.79  9.00 13.5
4 42.65
## 1957 30.94  7.23 28.61 32.47 20.38  5.55  3.72 13.16 39.87 22.9
3 27.39
## 1958 29.05 20.87 22.36 17.85 13.49 14.82 16.56 16.75 27.31 19.8
7 36.81
## 1959 13.34 16.39 12.87 52.01 31.93 14.46  5.30  7.03 13.89 14.4
1 48.98
## 1960 14.12 32.84 15.31 45.04 45.37  8.81  4.03  1.49  2.03  9.4
8 24.22
## 1961 16.01  6.79 28.98 47.65 49.32  9.67  1.85  0.45  0.18  5.7
3 14.86
## 1962 14.07 64.00 65.21 60.72 11.17  9.28 15.18  5.15  9.20 26.7
3 59.98
##        Dec
## 1953 39.19
## 1954 44.89
## 1955 22.15
## 1956 34.72
## 1957 44.83
## 1958 29.77
## 1959 28.98
## 1960 18.21
## 1961 13.46
## 1962 32.23
```

```
# See the column format is changed to
# a matrix format in years x months
```

In the following example, series has entered in matrix format rather than column format given in the `sampleData3.csv` file. Here, we need to convert it to a vector first and then convert it to the `ts` object.

```
sampleData3 = read_csv("/Users/haydardemirhan/Documents/MATH1307_F
  orecasting/notes/Module 1/sampleData3.csv")
```

```
## Parsed with column specification:
## cols(
##   Year = col_double(),
##   JAN = col_double(),
##   FEB = col_double(),
##   MAR = col_double(),
##   APR = col_double(),
##   MAY = col_double(),
##   JUN = col_double(),
##   JUL = col_double(),
##   AUG = col_double(),
##   SEP = col_double(),
##   OCT = col_double(),
##   NOV = col_double(),
##   DEC = col_double()
## )
```

```
class(sampleData3)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
head(sampleData3)
```

```
## # A tibble: 6 x 13
##    Year    JAN    FEB    MAR    APR    MAY    JUN    JUL    AUG
    SEP
##   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
  <dbl>
## 1  1986 0.0764 0.0669 0.0533 0.0386 0.0347 0.0139 0.0394 0.0334
0.0465
## 2  1987 0.0413 0.0244 0.0406 0.0350 0.0291 0.0280 0.0180 0.0448
0.0758
## 3  1988 0.0338 0.0421 0.0418 0.0360 0.0316 0.0424 0.0273 0.0243
0.0587
## 4  1989 0.0403 0.0311 0.0511 0.0412 0.0292 0.0139 0.0163 0.0247
0.0444
## 5  1990 0.0601 0.0426 0.0476 0.0558 0.0612 0.0430 0.0332 0.0424
0.0708
## 6  1991 0.0516 0.0676 0.0553 0.0548 0.0342 0.0366 0.0335 0.127
0.168
## # … with 3 more variables: OCT <dbl>, NOV <dbl>, DEC <dbl>
```

```
# To put the data into correct order we need to use the transpose
   t() operation.
sampleData3.ts = ts(as.vector(t(as.matrix(sampleData3[,2:13]))),
                    start=c(1986,1),frequency = 12)

class(sampleData3.ts)
```

```
## [1] "ts"
```

```
head(sampleData3.ts)
```

```
##             Jan        Feb        Mar        Apr        May
    Jun
## 1986 0.07644696 0.06693375 0.05330857 0.03857360 0.03472612 0.0
1393000
```

```
sampleData3.ts
```

```
##             Jan        Feb        Mar        Apr        May
    Jun
## 1986 0.07644696 0.06693375 0.05330857 0.03857360 0.03472612 0.0
1393000
## 1987 0.04127712 0.02444500 0.04060167 0.03503417 0.02914923 0.0
2800258
## 1988 0.03378611 0.04210900 0.04182412 0.03604533 0.03158611 0.0
4236588
## 1989 0.04033250 0.03105968 0.05105125 0.04116250 0.02915933 0.0
1390700
## 1990 0.06011767 0.04257700 0.04762340 0.05580000 0.06118347 0.0
4304500
## 1991 0.05156167 0.06759500 0.05526848 0.05479950 0.03423286 0.0
3664429
## 1992 0.13005964 0.12284400 0.11769450 0.12405733 0.13134107 0.1
6305317
## 1993 0.10449963 0.11331238 0.09505317 0.09657937 0.08653889 0.1
0033538
## 1994 0.10057143 0.06497000 0.07386017 0.08899659 0.08221367 0.0
3767484
## 1995 0.10551571 0.08455900 0.08881286 0.07715083 0.07074186 0.0
7083870
## 1996 0.08045941 0.05287056 0.06418800 0.07067033 0.07165800 0.0
7057772
## 1997 0.07247111 0.07051609 0.07607692 0.04395522 0.02611000 0.0
2999692
## 1998 0.06960025 0.06955270 0.06957088 0.06968382 0.06989986 0.0
3673143
## 1999 0.03962883 0.04374367 0.06255306 0.03398250 0.03292483 0.0
4067147
##             Jul        Aug        Sep        Oct        Nov
    Dec
## 1986 0.03939500 0.03335429 0.04654378 0.04007237 0.04300286 0.0
4420794
## 1987 0.01800033 0.04479929 0.07578086 0.04790368 0.03775148 0.0
4359283
## 1988 0.02726917 0.02433727 0.05867650 0.05882667 0.06097667 0.0
2892294
## 1989 0.01628400 0.02474433 0.04441517 0.03625571 0.05392917 0.0
3892264
## 1990 0.03319933 0.04237600 0.07084767 0.05488545 0.06080471 0.0
3968500
## 1991 0.03351000 0.12736938 0.16813121 0.14567300 0.16632667 0.1
6027526
## 1992 0.18081250 0.16036429 0.20109421 0.17397317 0.17869667 0.1
3143150
## 1993 0.11860128 0.10541383 0.09030833 0.09542725 0.10010395 0.0
9919625
## 1994 0.08050500 0.09113238 0.06584067 0.07232133 0.09820425 0.0
```

```
7165370
## 1995 0.03028810 0.09562056 0.06875760 0.05782929 0.07198400 0.0
9321972
## 1996 0.07053076 0.07055954 0.07069755 0.07095454 0.09605500 0.0
5763383
## 1997 0.04748783 0.03070976 0.06158981 0.05785017 0.08445513 0.0
9199250
## 1998 0.03586900 0.03632222 0.06022235 0.06065422 0.04277720 0.0
7560909
## 1999 0.01127950 0.03389750 0.04575783 0.04013700 0.03928364 0.0
5588778
```

Visualisation of time series

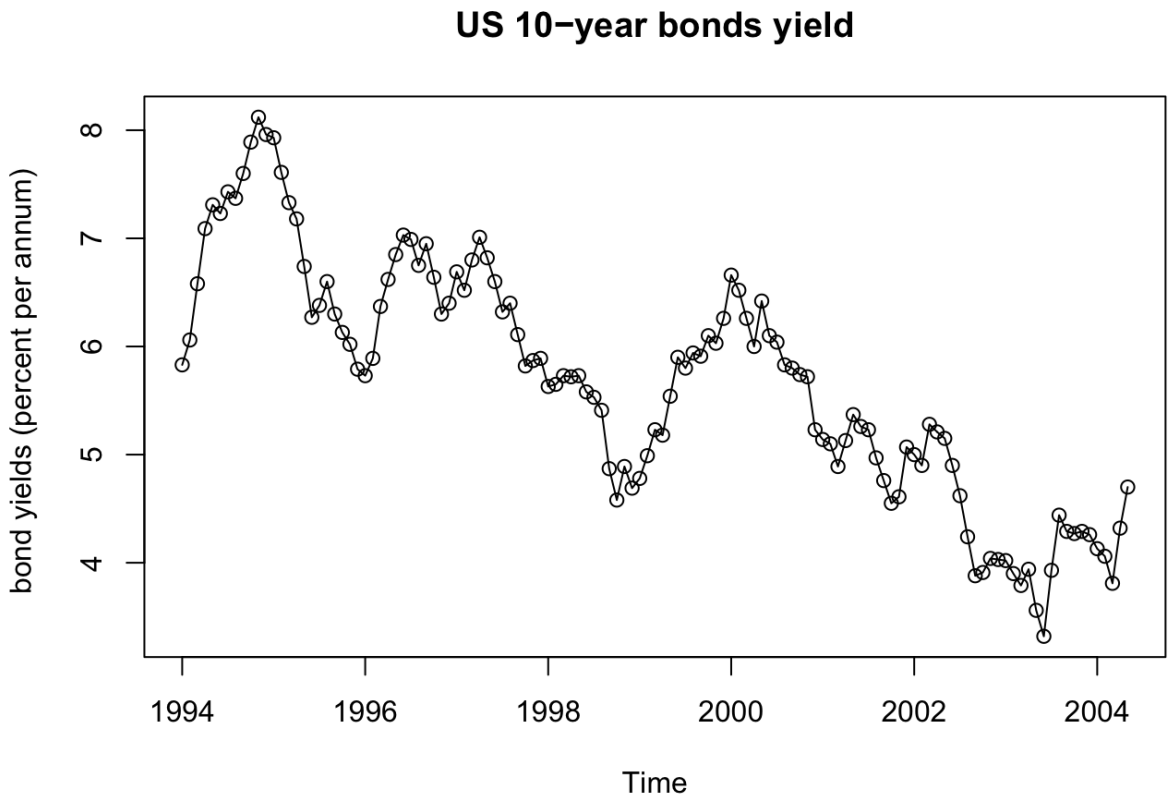
The main visualisation we will use in time series analysis is the **time series plot**. Time series plots display important characteristics of time series data. We start with interpreting the time series plot of data in all of the analyses.

There are four main patterns we would like to comment on from a time series plot:

1. Existence of a trend,
2. Evidence of seasonality,
3. Evidence of changing variance through time,
4. Patterns of moving average and autoregression behaviour,
5. Any sign of an intervention.

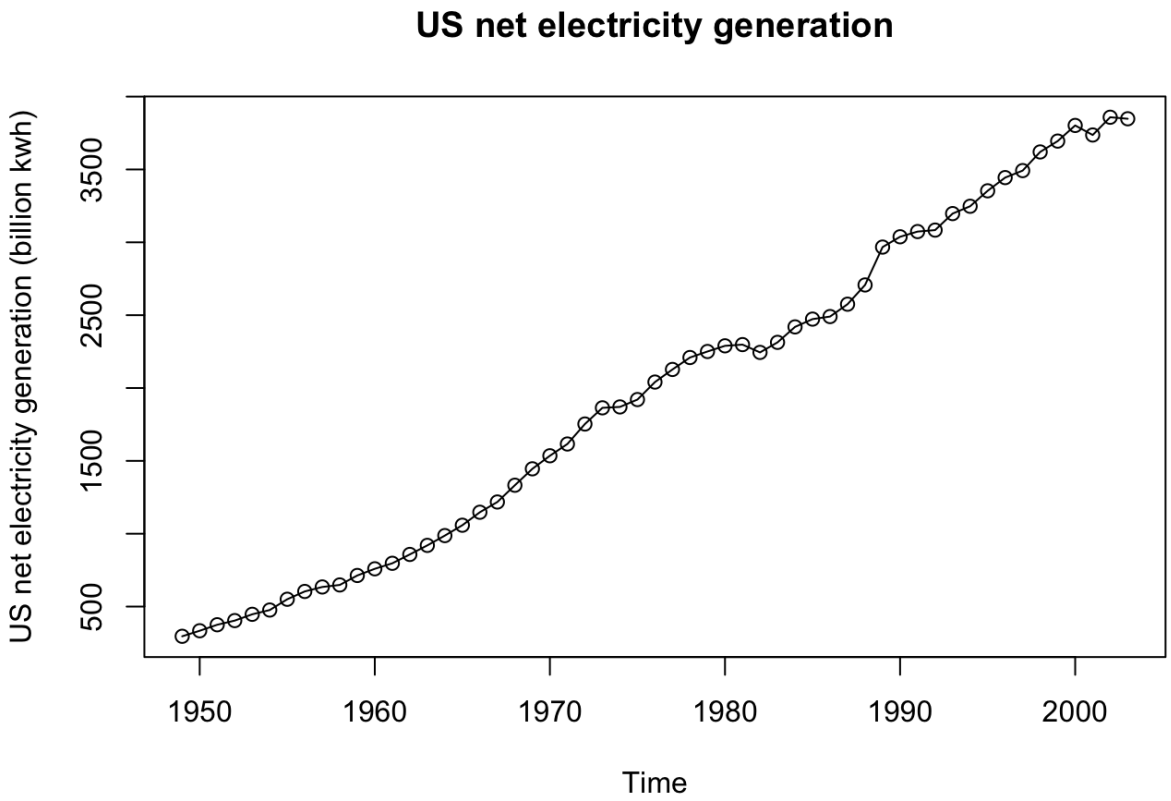
The following figure shows 125 monthly US government bond yields (percent per annum) from January 1994 to May 2004. This time series appears to have a trend caused by the changing mean level with a downward drift that one would be reluctant to forecast as continuing into the future, and it seems to have no discernible seasonal pattern. Changing variance is not obvious. Succeeding points imply the existence of a moving average pattern and there is no sign of an intervention.

```
library(expsmooth)
data(bonds)
plot(bonds,ylab='bond yields (percent per annum) ',xlab='Time',
     type='o', main="US 10-year bonds yield")
```



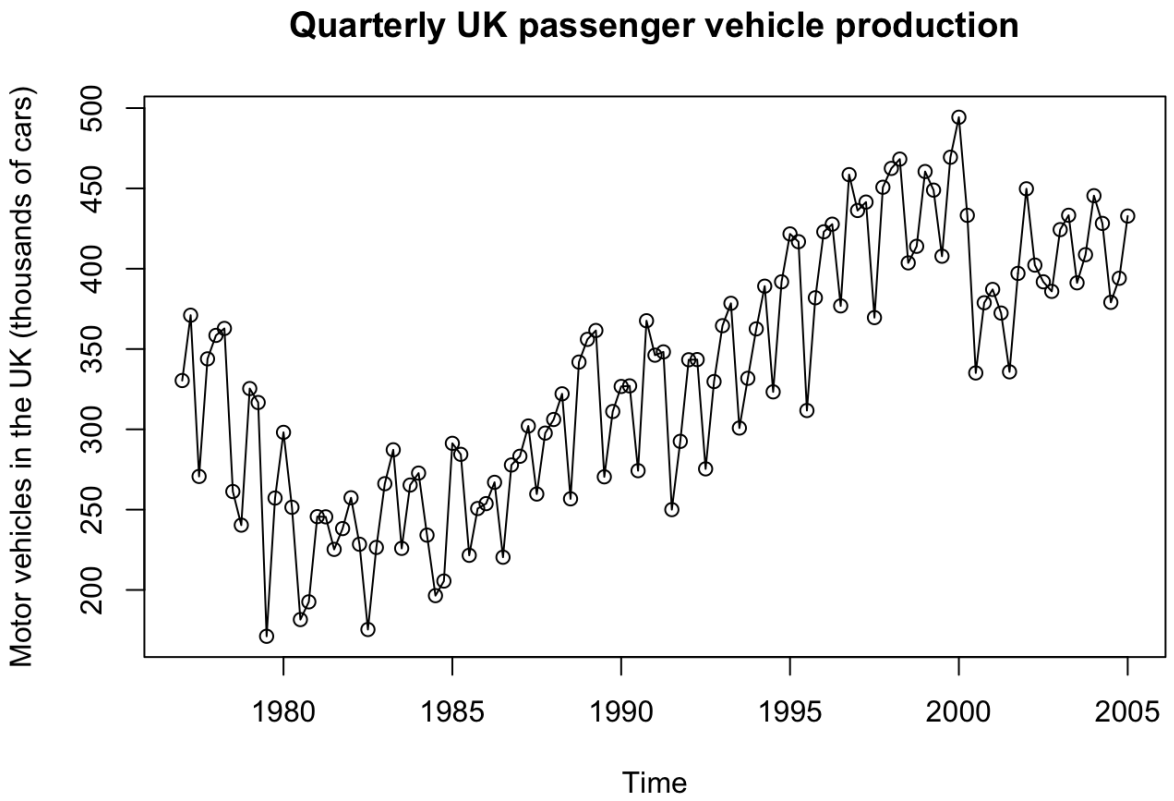
The next visualisation shows the time series lot of 55 observations of annual US net electricity generation (billion kWh) for 1949 through 2003. This time series contains a definite upward trend that changes somewhat over time. There is no sign of a seasonality, changing variance, or an intervention.

```
data(usnetelec)
plot(usnetelec,ylab='US net electricity generation (billion kwh)',
     xlab='Time',type='o', main = "US net electricity generation")
```



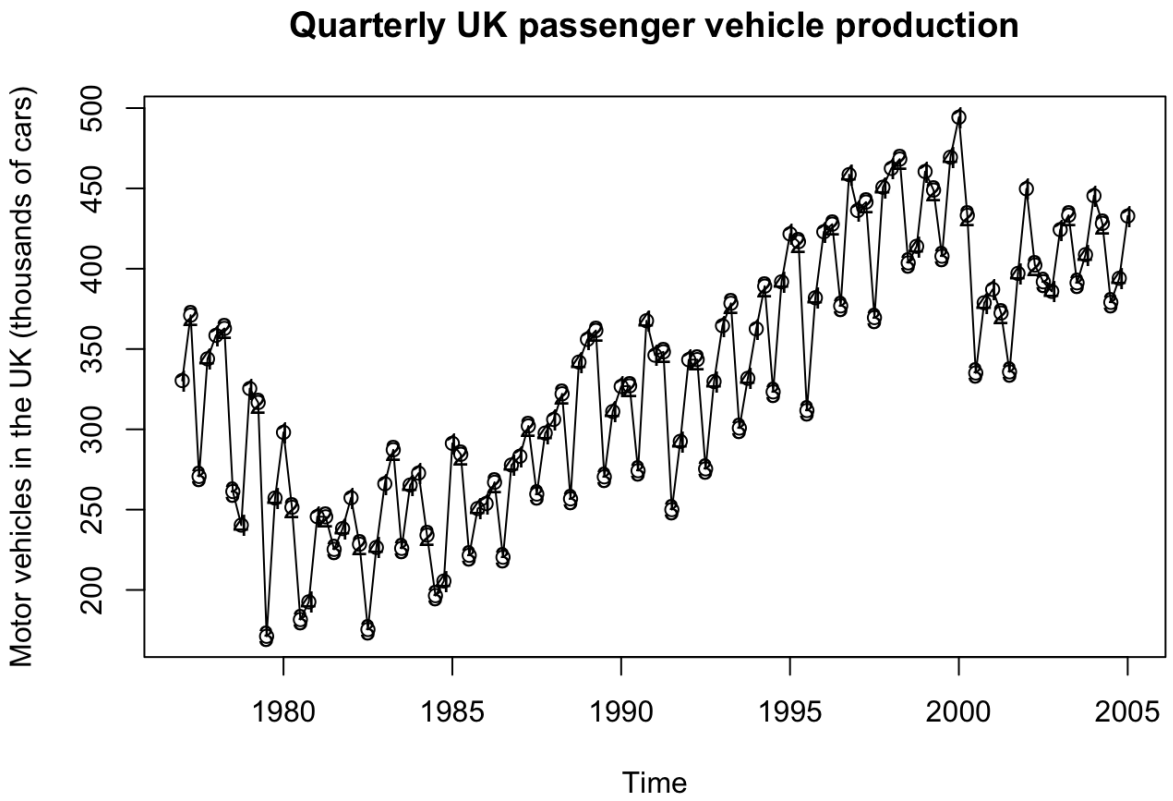
The next example is about 113 quarterly observations of passenger motor vehicle production in the UK (thousands of cars) for the first quarter of 1977 through the first quarter of 2005. For this time series, there is a constant variation around a changing level. This implies the existence of a moving average pattern. There is no trend that one would want to forecast as continuing into the future. However, there is a possibility of a seasonal pattern. The variance is not significantly changing through the time and there also an intervention around 2000.

```
data(ukcars)
plot(ukcars,ylab='Motor vehicles in the UK (thousands of cars)',
     xlab='Time',type='o', main = "Quarterly UK passenger vehicle
   production")
```



Sometimes it would be hard to infer about the existence of the seasonality over the time series plot with a visual inspection. In such cases, we can label data points over the time series plot (Cryer and Chan, 208). For example, if we use labels for months, we can look for whether or not the same months appear in a similar pattern. This idea is illustrated over the passenger motor vehicle production series below:

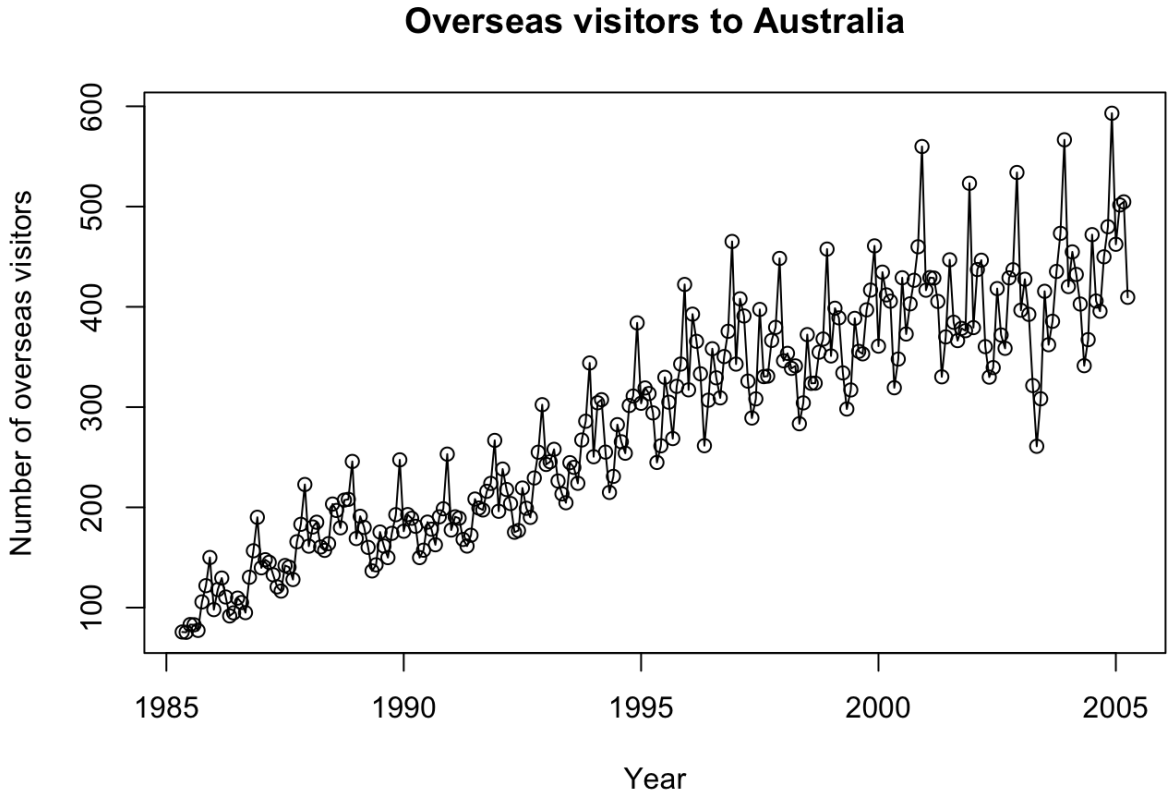
```
plot(ukcars,ylab='Motor vehicles in the UK (thousands of cars)',
     xlab='Time',type='o', main = "Quarterly UK passenger vehicle
   production")
points(y=ukcars,x=time(ukcars), pch=as.vector(season(ukcars)))
```



Lower numbers of productions were generally observed in the third quarter while higher production numbers were observed in the second or fourth quarter until the intervention point. Under the effect of the intervention, which would be a policy change or a financial crisis, the highest numbers were observed in the first quarter after the intervention point. Notice that it was not possible to observe this change without displaying the time series plot with labels on it. In the modelling phase, we need to take this intervention into account, which we will do in Module 4.

The time series plot for 240 monthly observations of the number of short term overseas visitors to Australia from May 1985 to April 2005 is displayed below. There is a definite seasonal pattern in this time series, and the variation increases as the level of the time series increases. It is not possible to tell visually whether the increase is due to an increase in the seasonal fluctuations or is caused by some other factors. While there is an upward drift, it might not be a good idea to forecast it as continuing into the future. There is no sign of an intervention.

```
data(visitors)
plot(visitors,ylab='Number of overseas visitors',xlab='Year',
     type='o', main = "Overseas visitors to Australia")
```



From these few examples, it is clear that there is frequently a need for forecasting that takes into account trend, seasonality, and other features of the data. Specifically, we will focus on the situation where we observe a time series $Y_1, \ldots, Y_n$, and we wish to forecast a future observation at time $n + s$.

Model-Building Strategy

It is not a straightforward task to find appropriate models for time series data. There are several components that a data scientist should deal with. Throughout the course we will follow a multistage model-building strategy with the following three steps:

- **descriptive look**
- **model specification (or identification)**,
- **model fitting**,
- **model diagnostics**, and
- **forecasting**.

In model specification (or identification), the classes of time series models are selected that may be appropriate for a given observed series. In choosing a model, we shall attempt to adhere to the **principle of parsimony**; that is, the model used should require the smallest number of parameters that will adequately represent the time series.

> Albert Einstein is quoted in Parzen (1982, p. 68) as remarking that “everything should be made as simple as possible but not simpler.”

The model fitting consists of finding the best possible estimates of a number of parameters involved in the model using a statistical estimation method such as least squares, maximum likelihood estimation, or Bayesian estimation.

Model diagnostics is concerned with assessing the quality of the model that we have specified and estimated. We try to answer the questions:

- How well does the model fit the data?
- Are the assumptions of the model reasonably well satisfied?

If no inadequacies are found, the modelling may be assumed to be complete, and the model may be used to forecast future values. Otherwise, we choose another model in the light of the inadequacies found; that is, we return to the model specification step. In this way, we cycle through the three steps until an acceptable model is found.

When we combine these steps in an R script, we create an **expert system** which can make the model selection in an automatic way.

Time Series and Stochastic Processes

The sequence of random variables $\{Y_t : t = 0, \pm1, \pm2, \pm3, \dots\}$ is called a *stochastic process* and serves as a model for an observed time series. It is known that the complete probabilistic structure of such a process is determined by the set of distributions of all finite collections of the Y's. Much of the information in these joint distributions can be described in terms of *means*, *variances*, and *covariances*. So, we will focus on the measures mostly based on the first and second moments.

Means, Variances, and Covariances

For a stochastic process $\{Y_t : t = 0, \pm1, \pm2, \pm3, \dots\}$, the *mean function* is defined by

$$\mu_t = E(Y_t), \text{ for } t = 0, \pm1, \pm2, \pm3, \dots$$

Actually, μ_t is just the expected value of the process at time t.

The *autocovariance function*, $\gamma_{t,s}$ is defined as

$$\gamma_{t,s} = Cov(Y_t, Y_s), \text{ for } t, s = 0, \pm1, \pm2, \pm3, \dots$$

where $Cov(Y_t, Y_s) = E[(Y_t - \mu_t)(Y_s - \mu_s)] = E(Y_t Y_s) - \mu_t \mu_s$.

The *autocorrelation function*, $\rho_{t,s}$, is given by

$$\rho_{t,s} = Corr(Y_t, Y_s), \text{ for } t, s = 0, \pm1, \pm2, \pm3, \dots$$

where

$$Corr(Y_t, Y_s) = \frac{Cov(Y_t, Y_s)}{\sqrt{Var(Y_t)Var(Y_s)}} = \frac{\gamma_{t,s}}{\sqrt{\gamma_{t,t}\gamma_{s,s}}}.$$

Please note that

$$\gamma_{t,t} = Cov(Y_t, Y_t) = E[(Y_t - \mu_t)(Y_t - \mu_t)] = E[(Y_t - \mu_t)^2] = E(Y_t^2) - \mu_t^2 = Var(Y_t)$$

and

$$\gamma_{s,s} = Cov(Y_s, Y_s) = E[(Y_s - \mu_s)(Y_s - \mu_s)] = E[(Y_s - \mu_s)^2] = E(Y_s^2) - \mu_s^2 = Var(Y_s).$$

Consequently, $\rho_{t,t} = \rho_{s,s} = 1$.

Because $\gamma_{t,s} = \gamma_{s,t}$, we have $\rho_{t,s} = \rho_{s,t}$. And because $|\gamma_{t,s}| \leq \sqrt{\gamma_{t,t}\gamma_{s,s}}$, we have $|\rho_{t,s}| \leq 1$.

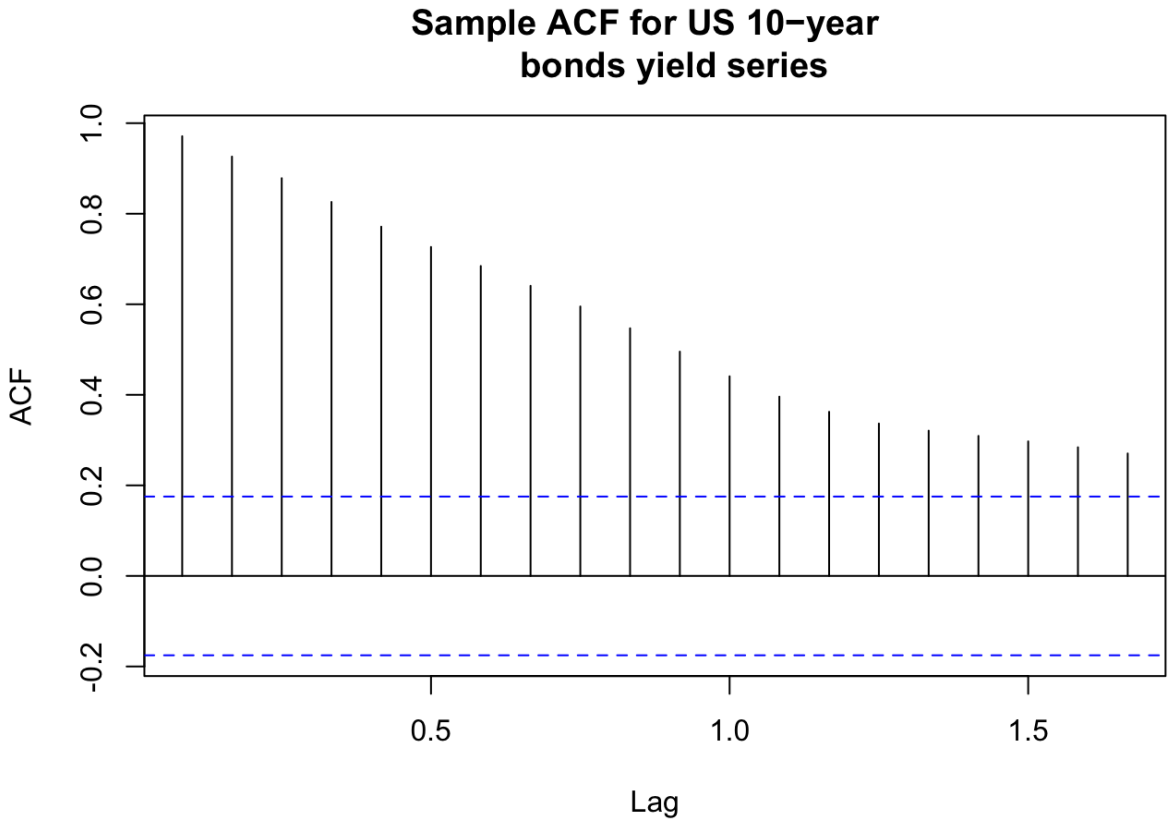
Values of $\rho_{t,s}$ near ± 1 indicate strong (linear) dependence, whereas values near zero indicate weak (linear) dependence. If $\rho_{t,s} = 0$, we say that Y_t and Y_s are **uncorrelated**.

Autocorrelation function

Autocorrelation measures the correlation between two values of the same variable at two different times such as Y_t and Y_{t+k}. Proceeding over the lags $k = 1, 2, \ldots$, we calculate autocorrelations for each lag and display by the sample ACF plot.

The following is the sample Autocorrelation Function (ACF) plot for US bonds series.

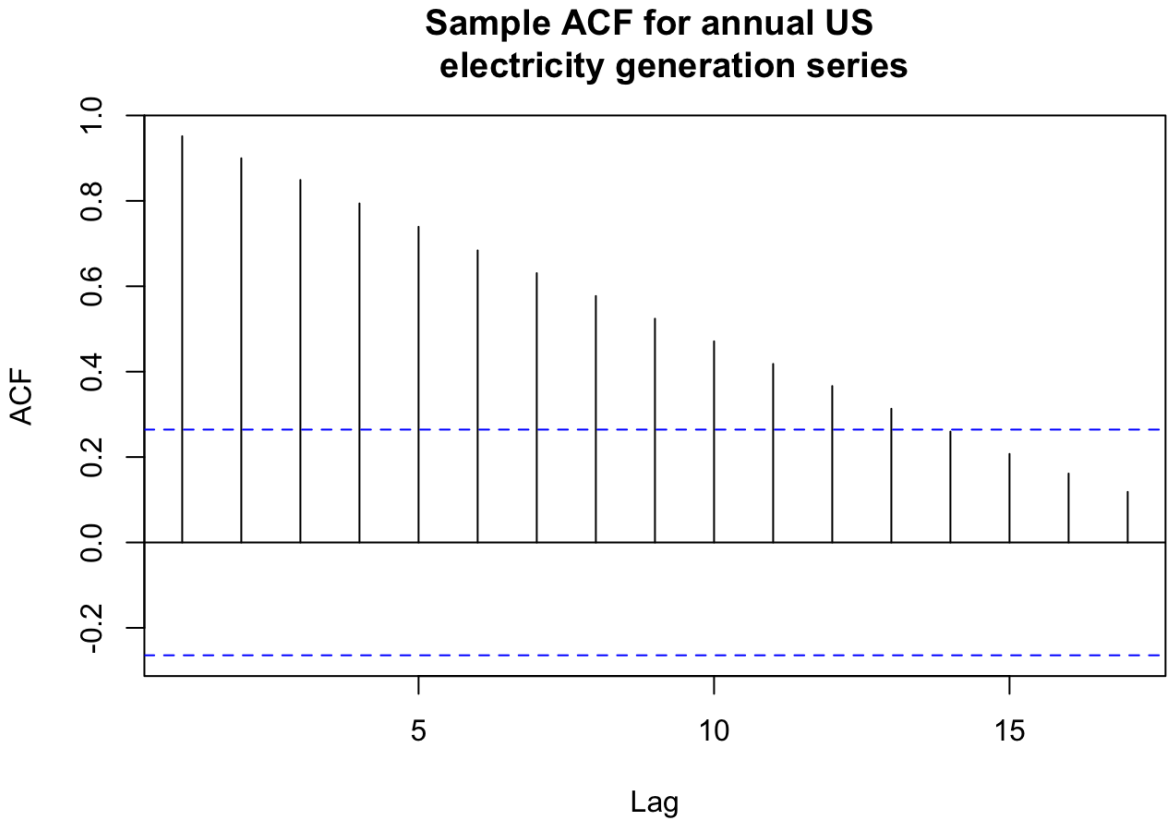
```
acf(bonds, main="Sample ACF for US 10-year
    bonds yield series")
```



Because we have a slowly decreasing pattern in ACF, we infer that there is a trend in the series which dominates the series correlation properties of the series.

The sample ACF is displayed for the annual US net electricity generation below.

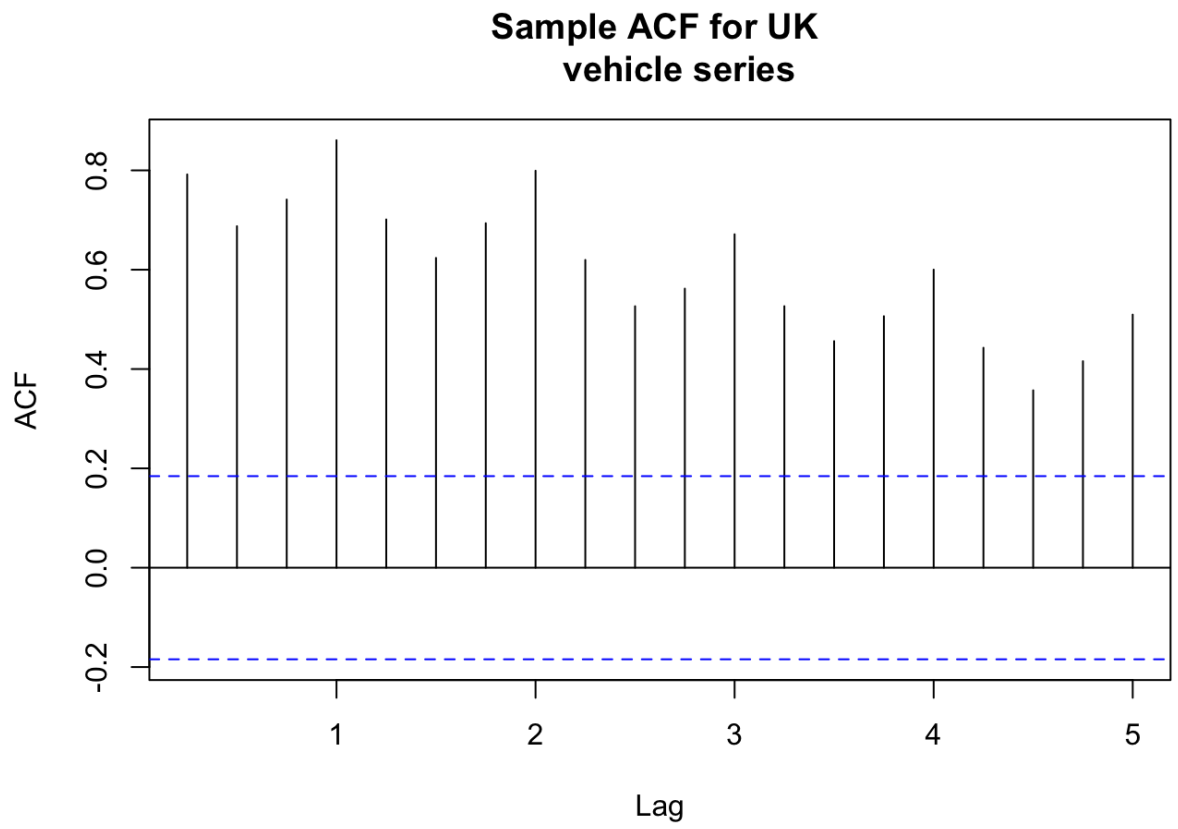
```
acf(usnetelec, main="Sample ACF for annual US
    electricity generation series")
```



As in the bonds series, we can infer the existence of a trend in the series. To observe further serial correlation characteristics of these series we need to remove the trend and then display sample ACF again.

The sample ACF for passenger motor vehicle production in the UK series is as follows:

```
acf(ukcars, main="Sample ACF for UK
    vehicle series")
```



In addition to the trend, we observe the existence of seasonality due to the wave pattern within the slowly decreasing pattern in ACF in this series. Also, the number of lags on x-axis shows the periods rather than each time point.

The Random Walk

Let $e_1, e_2, \ldots$ be a sequence of independent, identically distributed random variables each with zero mean and variance σ_e^2. The observed time series, $\{Y_t : t = 1, 2, \ldots\}$, is constructed as follows:

$$\begin{aligned} Y_1 &= e_1 \\ Y_2 &= e_1 + e_2 \\ &\vdots \\ Y_t &= e_1 + e_2 + \cdots + e_t \end{aligned}$$

Apparently, this system of equations can be written as

$$Y_t = Y_{t-1} + e_t$$

with *initial condition* $Y_1 = e_1$.

> If the e's are interpreted as the sizes of the "steps" taken (forward or backward) along a number line, then Y_t is the position of the **random walker** at time t.

We obtain the mean and variance of the random walk process as follows:

$$\mu_t = E(Y_t) = E(e_1 + e_2 + \cdots + e_t) = E(e_1) + E(e_2) + \cdots + E(e_t) = 0 + 0 + \cdots + 0.$$

So, $\mu_t = 0$ for all t. As for the variance,

$$Var(Y_t) = Var(e_1 + e_2 + \cdots + e_t) = Var(e_1) + Var(e_2) + \cdots + Var(e_t) = \sigma_e^2 + \sigma_e^2 + \cdots + \sigma_e^2.$$

So we have a linearly increasing variance $Var(Y_t) = t\sigma_e^2$ with time.

For $1 \leq t \leq s$, we have the following autocorrelation function,

$$\gamma_{t,s} = t\sigma_e^2.$$

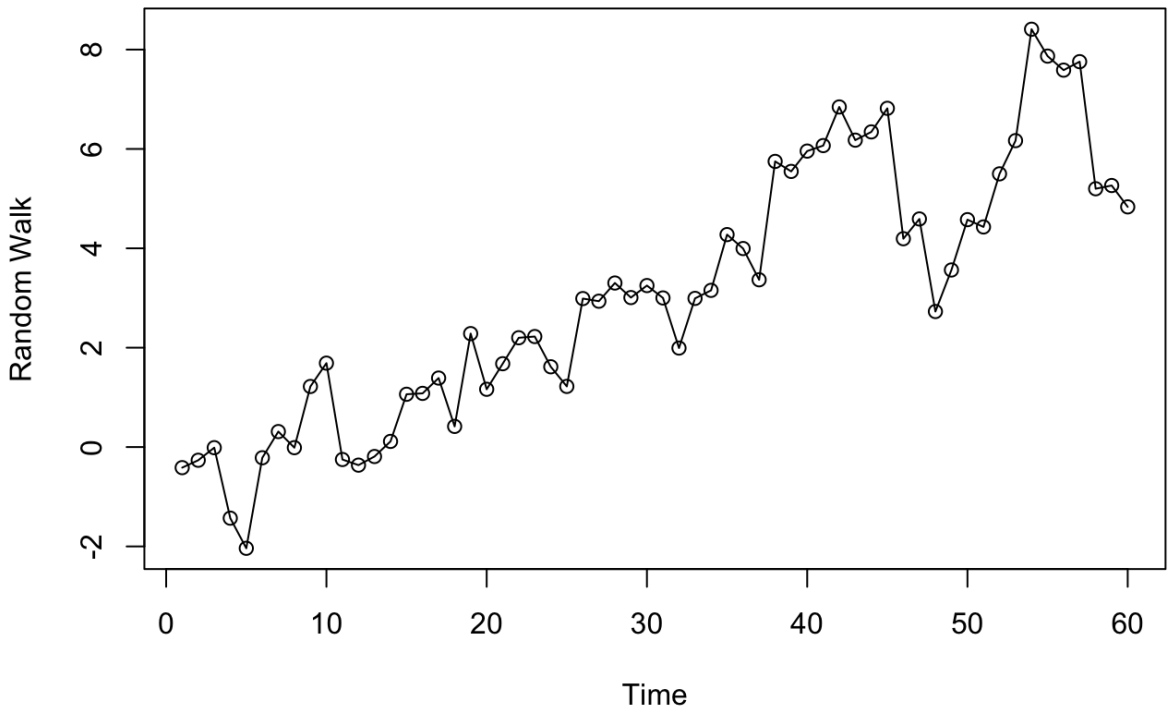
Autocorrelation function is straightforwardly obtained as follows:

$$\rho_{t,s} = \sqrt{\frac{t}{s}}$$

for $1 \leq t \leq s$. For example, we have the following autocorrelation values $\rho_{1,2} = 0.707$, $\rho_{8,9} = 0.943$, $\rho_{24,25} = 0.980$, and $\rho_{1,25} = 0.200$. The neighboring time points are more correlated than those distant from each other.

The following shows a simulated random walk:

```
data(rwalk) # rwalk contains a simulated random walk
plot(rwalk,type='o',ylab='Random Walk')
```



A Moving Average

Suppose that $\{Y_t\}$ is constructed as

$$Y_t = \frac{e_t + e_{t-1}}{2}$$

where e's are i.i.d. with zero mean and variance σ_e^2. Consequently, we have the following:

$$\mu_t = 0$$
$$Var(Y_t) = 0.5\sigma_e^2$$
$$Cov(Y_t, Y_{t-1}) = 0.25\sigma_e^2 \text{ and } Cov(Y_t, Y_{t-k}) = 0 \text{ for } k > 1, \text{ and}$$
$$\rho_{t,s} = 0.5 \text{ and } \rho_{t,t-k} = 0 \text{ for } k > 1.$$

Values of Y precisely one-time unit apart have exactly the same correlation no matter where they occur in time and, more generally, $\rho_{t,t-k}$ is the same for all values of t. This leads us to the important concept of stationarity.

Stationarity

The basic idea of stationarity is that the probability laws that govern the behaviour of the process do not change over time. The assumption of stationarity allows us to make statistical inferences about the structure of a stochastic process on the basis of an observed record of that process. Specifically, a process $\{Y_t\}$ is said to be strictly stationary if the joint distribution of $Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}$ is the same as the joint distribution of $Y_{t_1-k}, Y_{t_2-k}, \ldots, Y_{t_n-k}$ for all choices of time points $t_1, t_2, \ldots, t_n$ and all choices of time lag k.

It then follows that both the mean and variance functions are constant for all time: $E(Y_t) = E(Y_{t-k})$ and $Var(Y_t) = Var(Y_{t-k})$ for all t and k.

The covariance between Y_t and Y_s depends on time only through the time difference $|t - s|$ and not otherwise on the actual times t and s.

Thus, for a stationary process, to simplify our notation, we can write

$$\gamma_k = Cov(Y_t, Y_{t-k}) \text{ and } \rho_k = Corr(Y_t, Y_{t-k}) = \frac{\gamma_k}{\gamma_0}.$$

General properties of a stationary process are

$$\gamma_0 = Var(Y_t)$$
$$\gamma_k = \gamma_{-k}$$
$$|\gamma_k| \leq \gamma_0$$

and correspondingly,

$$\rho_0 = 1$$
$$\rho_k = \rho_{-k}$$
$$|\rho_k| \leq 1.$$

If a process is strictly stationary and has finite variance, then the covariance function must depend only on the time lag. A definition that is similar to that of strict stationarity but is mathematically weaker is the following: A stochastic process $\{Y_t\}$ is said to be weakly (or second-order) stationary if

1. The mean function is constant over time, and
2. $\gamma_{t,t-k} = \gamma_{0,k}$ for all time t and lag k.

In this course, the term stationary when used alone will always refer to this weaker form of stationarity.

White Noise

One of the important and mostly used stationary processes is called **white noise** process. A white noise process is actually a sequence of i.i.d. random variables $\{e_t\}$. Many useful processes can be constructed from white noise.

For a white noise process, $\mu_t = E(e_t)$ is constant and $\gamma_k = Var(e_t)$ for $k = 0$ and $\gamma_k = 0$ for $k \neq 0$. Thus, we can write $\rho_k = 1$ for $k = 0$ and $\rho_k = 0$ for $k \neq 0$. We usually assume that the white noise process has mean zero and denote $Var(e_t)$ by σ_e^2.

The moving average also defines a stationary process. For a moving average process, $Y_t = (e_t + e_{t-1})/2$, we have $\rho_k = 1$ for $k = 0$, $\rho_k = 0.5$ for $|k| = 1$, and $\rho_k = 0$ for $|k| \geq 2$.

Unit root tests

In some circumstances, we need to decide whether or not a process is stationary. A statistical way of evaluating stationarity of a process is to use a *unit root test*. In unit root tests the most crucial part is to specify the null hypothesis correctly. So, we need to characterize the trend properties of the series. An appropriate null hypothesis should reflect whether there is an increasing or decreasing trend. Test procedures use this information to specify the deterministic terms in the regressions used for testing and this influences the asymptotic properties of unit root test statistics.

We can include constant to capture if the series is stationary but has a nonzero mean, deterministic trend to capture if the series stationary with a deterministic trend or we can include both of them to capture if the series stationary with a deterministic trend and has a nonzero mean. Notice that we need to have some knowledge about the existence of these cases before putting this information for testing.

Another issue with the unit root tests is the selection of lag length which specifies the regression model used for testing. If the lag length is too small then the remaining autocorrelations in the errors will create a bias in the test. If it is too large then there will be power issues with the test. There are several approaches in the literature to specify the lag length. The following approaches are two of them

- For an autoregressive model first and then use the optimal number of lags from that model.
- Use the following to calculate the lag length:

$$k = \lceil 12(N/100)^{1/4} \rceil$$

where $\lceil \cdot \rceil$ is the integer part of the inner expression and N is the number of observations in the series.

Commonly used unit root tests are augmented Dickey-Fuller (ADF) and Phillips-Perron (PP) tests. The ADF and PP tests differ from each other in the way deal with serial correlation and heteroskedasticity in the errors. The ADF test can be applied using `adf()` function from `tseries` package and the PP test can be applied using `PP.test()` function from `stats` basic package.

The following code chunk applies ADF test to US 10–year bonds series with the first approach to determine the lag length.

```
library(tseries)
k = ar(bonds)$order
print(k)
```

```
## [1] 2
```

```
adf.test(bonds, k = k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  bonds
## Dickey-Fuller = -3.1598, Lag order = 2, p-value = 0.09783
## alternative hypothesis: stationary
```

The following code chunk implements the test with the second approach to determine the value of lag length.

```
k = trunc(12*((length(bonds)/100)^(1/4)))
print(k)
```

```
## [1] 12
```

```
adf.test(bonds, k = k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  bonds
## Dickey-Fuller = -3.2208, Lag order = 12, p-value = 0.08766
## alternative hypothesis: stationary
```

The following code chunk implements the test with the default value of lag length, which is $k = \lceil (N-1)^{1/3} \rceil$.

```
test  = adf.test(bonds)
test$parameter
```

```
## Lag order
##         4
```

```
print(test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  bonds
## Dickey-Fuller = -3.409, Lag order = 4, p-value = 0.05631
## alternative hypothesis: stationary
```

As it can be observed from the test results, the ADF test is very sensitive to the selection of lag length. From all versions of the test, we can conclude that the series is **nonstationary** at 5% level of significance.

The following code chunk implements the PP test for the bonds series. For the PP test, we have two options for the lag length. If `lshort` parameter is set to `TRUE` we get $k = \lceil 4 * (N/100)^{1/4}) \rceil$, otherwise we get $k = \lceil 12 * (N/100)^{1/4}) \rceil$.

```
PP.test(bonds, lshort = TRUE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  bonds
## Dickey-Fuller = -3.6346, Truncation lag parameter = 4, p-value
=
## 0.03304
```

```
PP.test(bonds, lshort = FALSE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  bonds
## Dickey-Fuller = -3.7285, Truncation lag parameter = 12, p-value
=
## 0.02476
```

According to the PP test, US bonds series is **stationary** at 5% level and the PP test is sensitive to the selection of lag length as well.

Forecasting Methods and Models

A *forecast* is a prediction of the value at a future time period and a *forecasting method* is an algorithm that provides a point forecast. On the other hand, a statistical model provides a stochastic data generating a process that may be used to produce an entire probability distribution for a future time period $n + s$. A point forecast can then be obtained easily by taking the mean (or median) of the probability distribution. A model also allows the computation of prediction (forecast) intervals with a given level of confidence. We use the notation $\hat{Y}_{n+h|n}$ to denote a point forecast of Y_{n+h} using the information available at time n.

We can use model structure and coefficient estimates to produce forecasts for any kinds of models. We will cover dynamic linear regression models for time series data, exponential smoothing, and state space models in this course. So, let's have a brief look at these models.

Summary

In this module, we introduced a special data type called time series data. We use simple visualisations to plot time series data for descriptive purposes. Then we focused on the fundamentals of time series analysis such as mean function, autocovariance function, autocorrelation, and partial autocorrelation functions. We illustrated these concepts with the basic processes: the random walk, white noise, and a simple moving average. The fundamental concept of stationarity is introduced and two-unit root tests to detect nonstationarity and some issues around the unit root tests are discussed.

References

- Cryer, J.D., Chan, K.S. (2008). Time Series Analysis With Applications in R, Springer.
- Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008). Forecasting with exponential smoothing: the state space approach (http://www.exponentialsmoothing.net), Springer-Verlag.