* Definitions

1. A time series (ts) is a collection of observations made sequentially in time.
2. A sequence of ts observations obtained at regular time intervals is called a regular time series (rts).

* Getting ready for Splus

1. Start Splus by typing

   Splus_tutorial.

2. Open a graphic window.

3. Set up your graphic window for multiple plots by typing

   >par (mfrow = c(m,n)) [to plot m x n = mn graphs on your graphic window]

* Time series analysis

1. Preliminary analysis of ts data

   (a.) **Time series plots**

      (i.) Suppose that d contains some regular ts data.

      >d <- c(48, 71, 50, 46, 40, 45, 18, 48, 32, 23, 22, 62, 31, 24, 40, 47, 54, 60, 54, 44, 59, 71, 64, 41)

      >ts.plot(rts(d))
Further,

(ii.) Suppose that the data are annual and starts from 1948. Now use the following sequence of commands

> d1 <- rts (d, start = 1948)
> ts.plot (d1, xlab = ‘’year’’)
> title (‘’Time Series plot of My Data’’, ylab = ‘’My value’’)

(iii.) Suppose that the data are monthly and starts from March 1948. Then use

> d2 <- rts (d, start = c (1948, 3), freq = 12, units = ‘’months’’)
> ts.plot (d2)
> title (‘’’, ylab = ‘’’)

(b.) Use the method of least squares (l.s.) to fit the l.s. trend line to the data in d

> t <- 1 : 24 (since d contains 24 values)
> l <- lm (d ~ t) [or >lsfit (t,d)]
> summary (l) [or > coef (l)]

(c.) Estimation of trend values


(d.) Plot of the trend line together with the ts plot

> ts.plot (rts (d), rts (dh))

or

> ts.plot (rts (d))
> abline (l)

Note:

The first command produces a better plot.

(e.) Plot of the detrended series

> ts.plot (rts(d - dh))

or

> g1<- rts(d - dh, start = 1948)
> ts.plot(g1)
or
> g2 <- rts(d - dh, start = c(1948, 3), freq = 12, units = 'months')
> ts.plot(g2)

2. Moving Averages (ma) of the data in \( d \)

Let \( l \) be the length (or span) of the required ma.

(a.) **When \( l \) is odd**

\[
f_1 > \text{filter} \ (d, \text{rep} \ (1/l, \ 1))
\]

or
\[
> \text{filter} \ (d, \text{c} \ (1/l, \ 1/l, \ldots, \ 1/l))
\]

(there are \( l \) times of \( 1/l \))

when \( l \) is odd the output will have \( n - (l - 1) \) observations with \( (l - 1)/2 \),
\text{NA} (not available) points at each end.

Example: For \( l = 5 \),
\[
> \text{filter} \ (d, \text{rep} \ (1/5, \ 5))
\]

or
\[
> \text{filter} \ (d, \text{c} (1/5, \ 1/5, \ 1/5, \ 1/5, \ 1/5))
\]

(b.) **When \( l \) is even**

In this case we use a centred ma with weights \( (1/2l, 1/l, \ldots, 1/l, 1/2l) \)
(there are \( l - 1 \) times of \( 1/l \) between the first and the last \( 1/2l \))

\[
f_2 > \text{filter} \ (d, \text{c} \ (1/2l, \ 1/l, \ldots, \ 1/l, \ 1/2l))
\]

Example: For \( l = 4 \)

the weights are \( (1/8, 1/4, 1/4, 1/4, 1/8) \). The command is:
\[
> \text{filter} \ (d, \text{c} \ (1/8, \ 1/4, \ 1/4, \ 1/4, \ 1/8))
\]
Further,
In many practical problems we need to produce the ts plot together with ma’s. The command is:

```r
> ts.plot (rts(d), rts(f1), rts(f2))
```
This will produce 2 ma’s calculated above on the ts plot of data.
(3 graphs together on the same plot)

**Label of graphs**

If you need to label each graph by a different symbol, then use:

```r
> ts.plot (rts(d), rts(f1), rts(f2), lty = c(1, 2, 3))
> legend (x, y, c('data', 'ma, l= 5', 'ma, l = 4'),
        lty = c(1, 2, 3))
```

**Legend**

(i.) $x, y$: location of the top left corner of the rectangle in which to put the legend.

(ii.) To change the size of the rectangle, replace $x$ by $c(x_1, x_2)$ and $y$ by $c(y_1, y_2)$, where $(x_1, y_1)$ is the top left corner and $(x_2, y_2)$ is the one opposite such that $x_2 > x_1$ & $y_2 < y_1$.

3. **Differencing.**

This is a very powerful tool of removing trend and/or periodicities of a homogeneous nonstationary time series.

(a.) **lag 1 differencing**

```r
> diff(d)
```

(b.) **lag l differencing**

```r
> dif <- diff(d, lag = 1)
```
Note:

dif will contain only $n - l$ number of observations. If you want to arrange the output in the correct time order (and the same length as in $d$), then use

```r
diff <- c(rep(NA, l), diff(d, lag = 1))
```

4. Visualising correlation of the data in $d$

(a.) **Lagged scatter plots**

```r
> lag.plot (d, lags = m, layout = c(a, b))
```

This produces $m$ scatter plots of pairs of values $(x_t, x_{t+k})$ for $k = 1, 2, ..., m$. `layout = c(a,b)` is to set up these $m = ab$ plots in a suitable way as an $aXb$ matrix.

As an example,

```r
> lag.plot (d, lags = 6, layout = c(3, 2))
```

produces 6 scatter plots of $(x_t, x_{t+k})$ for $k = 1, ..., 6$ and set up these 6 plots as a 3X2 matrix.

Note:

These lagged scatter plots can be used to detect the correlation at various lags.

(b.) **Autocorrelation plot (acf)**

```r
> acf (d)
```

Note:

you can specify the number of lags of the acf plot using

```r
> acf (d, lag.max = number)
```

Example:

If you need to see the acf values up to lag 30, then use

```r
> cf <- acf (d, lag.max = 30)
> cf
```
(c.) Partial Autocorrelation plot (pacf)

\[ \text{acf}(d, \text{lag.max} = \text{number}, \text{type} = "\text{partial}") \]

Example:

\[ \text{acf}(d, \text{lag.max} = 30, \text{type} = "\text{partial}") \]

produces the pacf plot of data up to lag 30.

5. Simulation of AR, ARMA and ARIMA processes

Let \( \{X_t\} \) be a stationary time series.

**Note:** \( X_t \) may contain the \( d \)th \((d \geq 1)\) difference of the original data if it is non-stationary. There is no constant term in the model. Thus the mean is 0.

**Notation:** A zero mean ARMA\((p,q)\) model is given by

\[ X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = Z_t - \theta_1 Z_{t-1} - \cdots - \theta_q Z_{t-q}, \]

where \( Z_t \sim WN(0, \sigma^2) \). i.e. the \( Z_t \) are uncorrelated, and have zero mean and variance \( \sigma^2 \).

**Simulation of pure AR processes**

A recursive filter can be used to simulate a pure AR process of the form

\[ X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + Z_t. \]

**Recursive filters**

\[ v \leftarrow \text{rnorm (number)} \] # number: specify the required length of the series.

\[ \text{ar} \leftarrow \text{filter}(v, \text{c(phi 1, phi 2,..., phi p)}, \"r\", \text{init = rnorm(p)}) \]

Example:

\[ v \leftarrow \text{rnorm (200)} \]

\[ \text{arsim} \leftarrow \text{filter}(v, \text{c(0.4, 0.5)}, \"r\", \text{init=rnorm(2)}) \]
generates 200 values from
\[ x_t = 0.4x_{t-1} - 0.5x_{t-2} + Z_t \]
using Graussian innovations.

**Simulation of general ARIMA processes**

```r
> arima.sim (number, model = list (ndiff = number, 
ar = c(phi 1, phi 2,..., phi p), ma = c(theta 1,theta 2,...,theta q)))
```

Example:

```r
>x <- arima.sim (500, model = list (ndiff=1, ar = c(0.4, 0.5), ma = c(0.7, 0.1)))
```
generates 500 values from the ARIMA(2,1,2) process
\[ x_t - 0.4x_{t-1} + 0.5x_{t-2} = Z_t - 0.7Z_{t-1} - 0.1Z_{t-2} \]
using Graussian innovations.

**Notes:**

(i.) The innovation vector \( Z_t \) can be replaced by another vector or by another random generator.

(ii.) There are number of optional arguments in arima.sim command. Please type `?arima.sim` for more details.

(iii.) arima.sim command can be used to simulate pure AR or pure MA processes. For example,

```r
>x <- arima.sim (300, model = list (ar = c(0.4, 0.5)))
```
generates 300 values from the AR(2) process
\[ x_t - 0.4x_{t-1} + 0.5x_{t-2} = Z_t \]
using Graussian innovations.

**Remark:** To avoid the effect of the initialization, a series longer than the one needed is generated, and the simulated series is taken from the end of the generated series.
6. **Parameter estimation of ARIMA models.**

In this section we consider the maximum likelihood estimation (mle) procedure for estimation of parameter of an ARIMA. This is also called the fitting an ARIMA model for a set of data in \( d \). Notice that we need a prior knowledge of \( p, q \) and \( d \) from the tsplot, the acf and pacf plots.

(a.) **Fitting an \( ARMA(p, q) \) model**

\[
fit1 <- arima.mle(d - mean(d), model = list(order = c(p, o, q)))
\]

(b.) **Fitting an \( ARIMA(p, r, q) \) model, \( r(\neq 0) \)**

\[
fit2 <- arima.mle(d, model = list(order = c(p, r, q)))
\]

**Splus output**

The output contains:
- Estimated ar and ma coefficients in \$model$ar and \$model$ma;
- Variance - covariance matrix of estimates in \$var.coef;
- AIC (Akaike Information Criterion) value in \$aic;
- Maximum value of the loglikelihood in \$loglik;
- Estimate of \( \sigma^2 \) is in \$sigma2.

7. **Diagnostic checking**

This stage of fitting ARIMA models consists of validation the model. The command

\>

produces 3 graphs consisting

(i) the plot of standardized residuals,
(ii) ACF plot of residuals, and
(iii) \( P-values \) for Goodness of fit statistic.
\( \chi^2 \) Goodness of fit test

In Splus the goodness of fit (gof) statistic is calculated by
\[ Q = N \sum_{k=1}^{K} \hat{r}_k^2, \]
where \( N \) is the number of observations, \( K \) is a fixed number, and \( \hat{r}_k \) is the lag \( k \) acf of the residuals via the estimated model. If \( r \) is the number of parameters in the fitted model, then under the null hypothesis of \( H_0 : X_t \sim \text{model} \), \( Q \) is approximately Chi-square (\( \chi^2 \)) random variable on \( K - r \) degrees of freedom.

This portmanteau test is due to Box and Pierce (1970). The corresponding gof statistic and associated \( P \)-values can be found in \( > \text{g} \), where

\[ > \text{g} <- \text{arima.diag} (\text{fit1, gof.lag=k}) \]

\( > \text{g} \)

**note:** In practice \( k \) is between 10 and 30

Plots (i) & (ii) in \( > \text{arima.diag} \) (fit1) are viewed to ensure that the residuals can be thought of as white noise. Large \( P \)-values of the gof statistic (for large lags) support the null hypothesis of \( H_0 : x_t \sim \text{model} \)

**Note:**

A better gof test is available in the ts literature based on the statistic
\[ Q^* = N(N + 2) \sum_{k=1}^{K} \frac{\hat{r}_k^2}{N - k}. \]
Under \( H_0 : X_t \sim \text{model} \), it is known that \( Q^* \) is approximately \( \chi^2 \) random variable on \( K - r \) degrees of freedom (as in \( Q \)).

This test is known as Ljung-Box-Pierce test. (This statistic is not available in Splus. You may perform your own calculations and do this test for a better, and a reliable result.)

8. Forecasting.

Use fit1 or fit2 in section 6.

(i) \( \text{fore1} <- \text{arima.forecast} (\text{d - mean(d), n = number, model = fit1$model}) \)
or
(ii) fore2 <- arima.forecast (d, n=number, model=fit2$model)

Note:
n= number is used to represent the leadtime of the forecasts.
As an example, n=3 gives the first 3 forecast values from the last observation of the data set.

Output

$mean - forecast values
$std.err - standard errors

In (I) we need to add the mean(d) to recover the correct forecast values. That is:
>fore11 <- fore1$mean + mean(d)
>fore11 (this gives the adjusted forecast values)

9. Confidence intervals (CI) of forecast values

Upper (u) and lower (l) limits of CI’s are calculated using:

>ul <- fore11 + Z * fore1 $std.err
>ll <- fore11 - Z * fore1 $std.err

where Z is the appropriate percentile value from the $N(0,1)$ distribution.

Example:
For a symmetric 95% CI, replace Z by qnorm(0.975) or by 1.96

10. Fitted values.

Using the results of fit2, the following command produces the fitted values as one-step-ahead predicted values:

arima.fitted <- arima.filt (d-mean(d),model=fit2$model)