DTU Statistical Consulting Center, DSCC
DTU Dataanalyse
Danmarks Tekniske Universitet

file:marima.doc.tex

Multivariate Time Series Estimation using marima

A time series program in R

by

Henrik Spliid

Version 1.2, November 2015

Henrik Spliid, professor emeritus
DTU, Bygning 324,
Danmarks Tekniske Universitet
2800 Lyngby

www.imm.dtu/DSCC
Telefon : +45 4525 3362
E-mail :hspl@dtu.dk

Contents

1 Introduction

1.1 The basic multivariate ARMA(p,q) model

Let t denote (discrete) time. Consider a k-variate random vector y_t of *observations* and, correspondingly, a k-variate random vector, u_t, of *unknown innovations*:

$$y_t = \begin{pmatrix} y_{1,t} \\ y_{2,t} \\ \cdots \\ y_{k,t} \end{pmatrix} \quad , \text{ and } \; u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \cdots \\ u_{k,t} \end{pmatrix} \quad , \quad t = \{1, 2, \ldots, N\} \tag{1}$$

Further suppose that the random vector y_t is generated through the model

$$y_t + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} = u_t + \theta_1 u_{t-1} + \cdots + \theta_q u_{t-q} \tag{2}$$

where the coefficient matrices $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ all are of dimension $k \times k$.

Generally, it is assumed that the series u_t is without autocorrelation, but the individual elements (coordinates) need not be, for example, uncorrelated. The covariance matrix of u_t will be referred to as Var(u_t) $= \Sigma_u$, and it does not depend on t.

The lefthand side of equation (2) is called the *autoregressive* or AR part of the model, while the righthand side is called the *moving average* or MA part of the model. p is the order of the AR part, and q is the order of the MA part. The model is called the $ARMA(p, q)$ model.

1.2 Organisation of time series

If the matrix of the k-dimensional observations is called y, then y should be organised as a $k \times n$ matrix. For example, it can be initialised using NA:

```
y <- matrix(NA, nrow=k, ncol=N).
```

2 Operator form of the ARMA(p,q) model

2.1 Matrix polynomials used in marima

Define the *k-variate backwards shift operator* B such that if B is multiplied on a time indexed k-variate random variable, the result is to be interpreted as the variable lagged one timestep. Thus, in general, lagging r time steps is accomplished using:

$$B^r y_t = y_{t-r}$$

Introduce the operator B into model (2) which then can be written as

$$(I + \phi_1 B + \cdots + \phi_p B^p) y_t = (I + \theta_1 B + \cdots + \theta_q B^q) u_t \quad ,$$

where I is the $k \times k$ unity matrix. Also, define the *matrix polynomials* $\phi(B) = I + \phi_1 B + \cdots + \phi_p B^p$ and $\theta(B) = I + \theta_1 B + \cdots + \theta_q B^q$. This leads to the general multivariate ARMA(p,q) model in operator form

$$\phi(B) y_t = \theta(B) u_t \quad , \tag{3}$$

Generally the averages of the variables in y_t are subtracted before the model estimation. When reconstructing or forecasting the measured series analysed by `marima`, the averages of the original data can be reintroduced.

2.2 Averages and their representation in the arma-model

Suppose that the vector y_t has been (for example) means-adjusted, such that $y_t = v_t - \eta$, where v_t represents the original measurements, and η is the (estimated) vector of averages: $E\{v_t\} = \eta$. Suppose now, that $\phi(B) y_t = \theta(B) u_t$ or, equivalently:

$$\phi(B)(v_t - \eta) = \theta(B) u_t$$

$$\phi(B) v_t = \mu + \theta(B) u_t \quad ; \quad \mu = \phi(B)\eta$$

$$\mu = \left[\sum_{i=0}^{p} \phi_i \right] \eta \tag{4}$$

Note that equation (4) applies for any transformation of the form $y_t = z_t - \eta$.

2.3 Inverse matrix polynomials and alternative forms

It is convenient to be able to write model (3) in the following form:

$$y_t = u_t + \psi_1 u_{t-1} + \cdots + \psi_\ell u_{t-\ell} + \cdots = \psi(B) u_t \tag{5}$$

Given the model (3), the model (5) can be determined if we are able to calculate the *left inverse polynomial* $\phi^{-1}(B)$ of the polynomial $\phi(B)$, such that

$$\phi^{-1}(B)\phi(B) = I \tag{6}$$

In general, if $\phi(B)$ is a finite order polynomial, the inverse, $\phi^{-1}(B)$ is of infinite order.

Pre-multiplying with $\phi^{-1}(B)$ on both sides of the equals sign in model (3) gives

$$y_t = \phi^{-1}(B)\theta(B) U_t = \psi(B) u_t \tag{7}$$

This form is called the *random shock* form, and, generally (if the model includes a nonzero AR term), the new polynomial $\psi(B)$ is of infinite length with decreasing coefficients, such that $\psi(\ell) \to 0$ for $\ell \to \infty$. If, more precisely, $\sum_{i=0}^{\infty} \psi(z)$ converges for all $|z| \leq 1$ the model (7) is said to be *stationary.*

Similarly if $\theta^{-1}(B)$ is the left inverse of $\theta(B)$, we may pre-multiply with $\theta^{-1}(B)$ on both sides of the equals sign in model (3). This gives the socalled *inverse form*:

$$\pi(B)y_t = u_t \tag{8}$$

where $\pi(B) = \theta^{-1}(B)\phi(B)$. If, similarly, $\sum_{i=0}^{\infty} \pi(z)$ converges for all $|z| \leq 1$ the model (8) is said to be *invertible.*

3 Some marima concepts in R

3.1 Organisation of matrix polynomials in R

A p-order matrix polynomial, such as $\phi(B)$ above, is represented by an `array(.)` with dimension $k \times k \times (1+p)$ holding the matrix-coefficients of the polynomial, $\{\phi_0, \phi_1, \phi_2, \ldots, \phi_p\}$, in that $\phi_0 = I$. The array can, for example, be initialised with NAs using

```
dependent  <- paste("y", c(1:k))
regressor  <- paste("r", c(1:k))
order      <- paste("o", c(0:p))
phi        <-
array(data=NA, dim=c(k,k,(1+p)), dimnames=list(dependent,regressor,order))
```

in which case you will get convenient labels on the polynomial.

In the ℓ'th coefficient matrix, the element $\phi_{i,j,\ell}$ (that is the i'th row and the j'th column), for example, represents the influence (regression) from variable j, lagged ℓ time units, on the present variable i (being the dependent variable, so to speak).

3.2 Simple operations for matrix polynomials using R

As noted, the 0'th order coefficient matrix of `phi`, ϕ_0, (that is `phi[,,1]`), is the $k \times k$ unity matrix.

The *marima* package includes the basic routines for inverting and multiplying matrix polynomials, namely `pol.inv` and `pol.mul`.

The left inverse of `phi` is computed as, say, `inv.phi <- pol.inv(phi, L)` which will result in an array of dimension $k \times k \times (1+L)$ holding the $k \times k$ unity matrix followed by the first `L` matrix coefficients of the left inverse of `phi`. If the leading term of `phi` is *not* the unity matrix the computations performed by `pol.inv` will be carried out with a leading unity matrix inserted. The resulting inverse polynomial will include the proper leading unity matrix followed by the first L matrix coefficients.

The product of two matrix polynomials, $\phi(B)$ and $\theta(B)$, is computed as `pol.mul(phi,theta,L)` which will result in an array of dimension $k \times k \times (1+L)$ holding the $k \times k$ unity matrix followed by the first `L` matrix coefficients of the product $\phi(B)\theta(B)$. If the leading terms of `phi` and `theta` are *not* unity matrices the computations will be carried out with leading unity matrices inserted. The resulting product polynomial will include the proper leading unity matrix followed by the first L matrix coefficients of $\phi(B)\theta(B)$.

Equation (7) can be carried out using, for example, `psi<-pol.mul(pol.inv(phi,L), theta, L)` giving the unity matrix of order $k \times k$ followed by the first `L` matrix terms of the $\psi(.)$ polynomium. The array `psi` will have dimension $k \times k \times (1 + L)$.

Using, for example, `pol.mul(pol.inv(phi,L=5), phi, L=5)` will result in a unity matrix followed by 5 (zero-) matrices, all matrices having dimension $k \times k$, provided that `phi` is a proper matrix polynomial of order $L \geq 1$. You may try:

```
phi<-array(rnorm(48), dim=(c(4,4,3))) # no leading unity matrix
                                      # generated. Not necessary.
inv.phi<-pol.inv(phi, L=5)
prod.phi<-pol.mul(inv.phi, phi, L=5)
Order<-pol.order(prod.phi, digits=12)
prod.phi<-prod.phi[,,(Order+1)]
prod.phi                             # print result
```

4 Differencing multivariate time series

4.1 Differencing operators

4.1.1 Single time step differencing

The polynomial $\nabla(B) = (I - B)$ can be used to difference the time series one time step, for example

$$z_t = \nabla(B)y_t = y_t - y_{t-1} \tag{9}$$

If, for example, $y_t = \{y_{1,t}, y_{2,t}\}^T$ is bivariate then the ∇ polynomial for differencing both variables once is

$$\nabla(B) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}$$

The inverse of $\nabla(B)$ is called $\nabla^{-1}(B)$, and it corresponds to the (infinite) sum $\nabla^{-1}(B) = I + B + B^2 + \ldots$, so that

$$z_t = \nabla^{-1}(B)y_t = y_t + y_{t-1} + y_{t-2} + \ldots \tag{10}$$

Sometimes it may be necessary to difference a time series twice. This is done by using $\nabla(B)$ twice i.e. $z_t = \nabla(B)\nabla(B)y_t = y_t - 2y_{t-1} + y_{t-2}$.

4.1.2 Seasonal differencing

The polynomial $\nabla(B^s) = (I - B^s)$ is used if a seasonal differencing with seasonality s is wanted:

$$z_t = \nabla(B^s)y_t = y_t - y_{t-s} \tag{11}$$

The polynomial for s timesteps seasonal differencing is

$$\nabla(B^s) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} B^s & 0 \\ 0 & B^s \end{pmatrix}$$

4.1.3 Mixed differencing

When a multivariate time series is at hand, it may be necessary to difference the individual series differently.

Suppose again, that $y_t = \{y_{1,t}, y_{2,t}\}^T$ is bivariate, and that we want to difference over time periods $s = \{s_1, s_2\}$. Then we may define, a little more generally,

$$\nabla(B^s) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} B^{s_1} & 0 \\ 0 & B^{s_2} \end{pmatrix}$$

and

$$\nabla(B^s) \begin{pmatrix} y_{1,t} \\ y_{2,t} \end{pmatrix} = \begin{pmatrix} y_{1,t} \\ y_{2,t} \end{pmatrix} - \begin{pmatrix} y_{1,t-s_1} \\ y_{2,t-s_2} \end{pmatrix}$$

The routine `define.dif(...)` can perform mixed differencing of a multivariate series. The routine `define.sum(...)` does the reverse, that is summing a multivariate series. See the following section (4.2).

4.2 Differencing and summing in marima

4.2.1 Differencing

If differencing (seasonal or other) is to be used, a function `define.dif` is available. The function performs the differencing wanted and generates the corresponding autoregressive representation, which later can be used when forecasting the original time series in the not-differenced form.

The user specifies a differencing pattern in the form of a matrix, called `difference`, with 2 rows and `m` columns, where `m` is number of differencing operations to be performed. Each column in `difference` specifies 2 numbers: 1) number of the variable to be differenced, and 2) differencing length (S) for that variable.

As an example, consider $y_t = \{y_{1,t}, y_{2,t}\}^T$, and suppose it is wanted to difference $y_{1,t}$ ordinarily ($S = 1$) twice, and to difference $y_{2,t}$ once with seasonality $S = 12$. Then the differencing pattern is specified as:

`difference` $= \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 12 \end{pmatrix}$ or in R-code: `difference <- matrix(c(1,1,1,1,2,12), nrow=2)`.

Output from `define.dif` is the differenced time series, the autoregressive representation of the differencing and the averages of the variables in the time series (which are subtracted from the variables before the differencing is performed).

When analysing the resulting time series with `marima` the first values should be disregarded in the usual way. In the above example the first 12 values should be left out, because the first 12 values are obtained by differencing in relation to previous (unknown) values (taken to be equal to the average of the variable in question). You may try

```
y          <- matrix(rnorm(48), nrow=2)
difference <- matrix(c(1,1,1,1,2,12), nrow=2)
```

```
Y           <- define.dif(y, difference=difference)
names(Y)
y.dif    <- Y$y.dif
y.lost   <- Y$y.lost
dif.poly <- Y$dif.poly
averages <- Y$averages
```

4.2.2 Aggregated model

The differencing polynomium based on the above example, `difference` $= \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 12 \end{pmatrix}$, for a bivariate series can be written

$$\nabla(B) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} B + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} B^2 - \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} B^{12}$$

The differenced series is called z_t, and $z_t = \nabla(B)y_t$. Suppose now that z_t is analysed by `marima` and that the estimated model for z_t is $\widehat{\phi}(B)z_t = \widehat{\theta}(B)u_t$. The estimated aggregated (nonstationary) model for the observed time series, y_t, is then $\widehat{\phi}(B)\nabla(B)y_t = \widehat{\theta}(B)u_t$.

The estimated ar-polynomium, $\widehat{\phi}(B)$ (for z_t), is returned by `marima` and saved in, say, `..$ar.estimates`, and the differencing polynomium, $\nabla(B)$, is returned from the differencing function `define.dif` and saved in, say, `..$dif.poly`.

The estimated aggregated ar-polynomium is then calculated as
`ar.aggregated <- pol.mul(..$ar.estimates, ..$dif.poly, L=...)`.

4.3 Analysis of non-stationary models

Non-stationarity is often handled (see Hamilton (1994)) by means of differencing by which one or more unit roots are removed from the autoregressive part of the arma model. A simple example of a unit root appears in the model $y_t = y_{t-1} + \epsilon_t$, where ϵ_t represents some process (as opposed to $y_t = \phi y_{t-1} + \epsilon_t$ where $|\phi| < 1$). Similarly, if $y_t = y_{t-s} + \epsilon_t$, the non-stationarity corresponds to a seasonal period unit root for the time difference s.

These simple types of non-stationarity are handled by differencing the timeseries before it is analysed. In the first example we use $\nabla(B) = 1-B$ and $\nabla(B)y_t = y_t - y_{t-1} = \epsilon_t$. In the second example $\nabla_s(B) = 1-b^s$ and $\nabla_s(B)y_t = y_t - y_{t-s} = \epsilon_t$.

As described in the above, this can be accomplished by differencing the time series properly (by means of the routine `define.dif`) before analysing it with `marima`. Suppose `y` is the k-variate time series, and suppose the differencing wanted is given in the same way as described at page 7, where, for example,
`difference=matrix(c(1,1,2,1,3,12),nrow=2)`
(meaning single time step differencing for variables 1 and 2 , and 12 time steps differencing for variable 3). The proper *R*-code could be something like:

```
dy <- define.dif(y,difference=difference)

# Now the object 'dy' contains dy$y.dif, dy$y.lost and dy$dif.poly
# and dy$averages (the averages of the original time series).
# Here dy$dif.poly is the ar-representation of the differencing
# polynomial, and dy$y.dif is the differenced time series. Note
# that, in general, dy$y.dif is shorter than the original
# time series, y. Now use, for example:

estimate = marima(dy$y.dif, ar=c(...),ma=c(...), etc...)

# and finally:

ar.aggregated <- pol.mul( estimate$ar.estimate, dy$dif.poly,
            L = ( max(ar) + dim( dif.poly)[3] ) )

# will yield the aggregated (non-stationary) ar-part of the model wanted.
```

4.3.1 The full aggregated model

Suppose the above procedure is being used, and $\nabla(B)$ is used for differencing the time series y_t. Further suppose that a model for $\nabla(B)y_t = z_t$ is derived by means of `marima`. Any mean vector of y_t, μ_y, will not affect z_t since, in general, $\nabla(B)(y_y - \mu_y)\nabla(B)y_t$. If some of the variables in y_t are not differenced, the averages of these variables will, of course, be retained in z_t.

The differenced time series $z_t = \nabla(B)y_t$ is supposed to have mean $E(z_t) = \mu_z$ which is estimatet by the average of z_t (as $\widehat{\mu}_z$) and taken out before the analysis made in `marima`

Then the estimated model will be:

$$\widehat{\phi}(B)(z_t - \widehat{\mu}_z) = \widehat{\phi}(B)\nabla(B)(y_t - \widehat{\mu}_y) - \widehat{\phi}(B)\widehat{\mu}_z = \widehat{\theta}(B)u_t$$

or

$$\widehat{\phi}(B)\nabla(B)y_t - \widehat{C} = \widehat{\theta}(B)u_t, \text{ where } \widehat{\mathrm{C}} = \sum_{\mathrm{i}=0}^{\mathrm{p}} \widehat{\phi}_{\mathrm{i}}\widehat{\mu}_{\mathrm{z}} \tag{12}$$

It is seen that the model constant $\widehat{C}$ is estimated from the averages of the variables in z_t and the estimated autoresssive part of the arma model for z_t. The averages of those variables in y_t that are differenced do not influence $\widehat{C}$.

The residuals corresponding to the estimated model (12) and the estimated constant, $\widehat{C}$, will appear in the `marima` object based on the analysis of $z_t = \nabla(B)y_t$.

5 Long term lagging

Sometimes it is impractical to apply autoregression with very high order or seasonal dependence with very long seasonality. Instead it may be more convenient to apply lagging in combination with regression modeling.

5.1 Principles of lagging

Consider the bivariate time series $y_t = \{y_{1,t}, y_{2,t}\}^T$, and suppose it is wanted to model an autocorrelation over a relatively long time period, say S, from $y_{1,t}$, such that the present values in y_t depend on a previous value S time units ago. As an example a new variable generated from $y_{1,t}$, is introduced by lagging $S-1$ time steps as $y_{3,t} = y_{1,t-(S-1)}$, and expanding the original time series with the new variable, $Y_{3,t}$, to:

$$y_t = \begin{pmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{pmatrix} = \begin{pmatrix} y_{1,t} \\ y_{2,t} \\ y_{1,t-(S-1)} \end{pmatrix} \tag{13}$$

In theory one may model this new three-variate series as a usual three-dimensional series. However, the autoregression is generally considered to unidirectional, such that the present values depend only on previous values, but not vice versa. Then, both $y_{1,t}$ and $y_S 2, t$ are assumed to depend linearly on $y_{t,3}$.

The first order autoregression term in the standard `marima` model for the original variables and the introduced lagged variable could then have the following appearance:

$$\phi_1 y_{t-1} = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{pmatrix} \tag{14}$$

where $y_{3,t-1} = y_{1,t-S}$.

Note that, if the introduced lagged variable, y_3, is lagged $S-1$ positions, then the (first order) regression variable in the regression equation for $y_{t,1}$ and $y_{t,2}$ in the example will be lagged further one position to totally S positions. Therefore, if seasonality S is to be modelled, the new variable can be introduced by lagging $S-1$ positions.

Most often no model is specified for the lagged variable(s), and it (they) is (are) used as pure regression variable(s), although this is not a strict requirement. Anyway, a reasonable model specification including the lagged variable(s) should be restricted, so that the original series depends on the lagged variable(s), but not the other way around. Definition of such models including e.g. regression variables is conveniently made using the function `define.model`, see section (6).

Finally, the user must make sure, that there is created a complete time series when lagged variables are defined, created and added to the original time series. If the function `season.lagging` is used for introducing lagged variables this is automatically taken care of.

5.2 Creating lagged variables

A function called `season.lagging` can be used: `Y <- season.lagging(y,lagging)` where `y` is the original k×N time series and `lagging` is the lagging pattern wanted. The new (expanded) series which later can be handled in `marima` is found by, say, `y.lag<-Y$y.lag` .

The input parameter `lagging` is a matrix with 3 rows and `m` columns, where m is the number of (new) lagged variables. Each column in `lagging` contains 3 numbers: 1) number of one of the original variables, 2) number of new varible (>k), and 3) number of positions the new variable is lagged, namely $(S-1)$ positions for seasonality S.

Example: Suppose y_t is bivariate, and we want to generate two new lagged variables, no. 3 and 4 based on variables 1 and 2 by lagging. Suppose seasonalities $S_1 = 6$ and $S_2 = 12$ are the seasonalities that we want to model. Then, variable 3 and 4 are to be generated from the variables 1 and 2 in the original time series, $y_{1,t}$, by lagging 6-1 and 12-1 time units, respectively:

$$\texttt{lagging} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ \text{6-1} & \text{12-1} \end{pmatrix}$$ or in R-code: `lagging <- matrix(c(1,3,6-1,2,4,12-1), nrow=3)`.

This lagging pattern says that "Use variable 1 and create variable 3 as lagged (6-1) positions. Use variable 2 and create variable 4 as lagged (12-1) positions".

The new time series will be a four-variate time series, and it will initially have the following appearance:

$$\begin{array}{lllllllll}
y_{1,1} & \dots & y_{1,N-1} & y_{1,N} & \text{NA} & \dots & \text{NA} & \text{NA} & \text{NA} & \dots \\
y_{2,1} & \dots & y_{2,N-1} & y_{2,N} & \text{NA} & \dots & \text{NA} & \text{NA} & \text{NA} & \dots \\
\text{NA} & \dots & y_{1,N-1-5} & y_{1,N-1-4} & y_{1,N-1-3} & \cdots & y_{1,N} & y_{1,1} & y_{1,2} & \cdots \\
\text{NA} & \dots & y_{2,N-1-11} & y_{2,N-1-10} & y_{2,N-1-9} & \cdots & y_{2,N-1-5} & y_{2,N-1-4} & \cdots & y_{2,N}
\end{array}$$

The new series will have leading NA's for the generated lagged variables, and it will be expanded with the most present values of these variables. All variables, except the one with the longest lagging period, will be expanded with the first observations repeatedly, until all lagged variables have the same length as the variable based on the longest lagging period (variable 4 in the example).

Estimation by `marima` can be performed only for the part of the new series which is complete, that is the new series without the first observations and the future observations containing NA's.

The part of the new expanded series which is complete is saved (by `season.lagging`) in a matrix called `y.lag`. The first unusable observations are saved in `y.lost`, and similarly the *future* observations are saved in `y.future`.

The 'future' values of the expanded series containing the most present values of the lagged variables are retained in order to enable forecasting using the lagged observations and an estimated model. You may try

```
y       <- matrix(rnorm(48), nrow=2)
lagging <- matrix(c(1,3,6-1,2,4,12-1), nrow=3)
Y       <- season.lagging(y, lagging=lagging)
names(Y)
y.lagged  <- Y$y.lagged
y.lost    <- Y$y.lost
y.future  <- Y$y.future
```

6 Model selection in marima

6.1 Defining the marima model

Defining models in `marima` is done by creating 0/1-indicator arrays corresponding to the ar-part and the ma-part of the model. These arrays are organised the same way as the model polynomials wanted. Suppose the ar-indicator array is called `ar.pattern`. Then the value at position `ar.pattern[i,j,`ℓ`]` is the indicator for the $\{i, j\}$'th element in the lag=ℓ ar-parameter matrix $\phi_\ell = \{\phi_{i,j}\}_\ell$.

The value 1 (one) indicates that a parameter is to be estimated at that position. The value 0 indicates that the parameter corresponding to that position is 0. The function `define.model` can be used for setting up these indicator arrays properly.

Examples of the use of `define.model` are obtained with `library(marima); example(define.model)`.

6.2 Estimation and identification of a marima model

As described by Spliid (1983), the estimation of the model is done with a pseudo-regression method. In the present implementation the *R*-procedure `lm(...)` is used for doing the regression calculations. This choice enables `marima` to utilize the `step(...)` procedure in order to search for a good (reduced) model in a stepwise manner.

The key parameter in `step` is the k-factor used in Akaike's criterion (where k=2 is Akaike's suggestion). The k-factor is set when calling `marima` by specifying the input parameter `penalty` to a suitable k-value (around 1 or 2, for example).

After having estimated the `marima`-model as defined with the use of, for example, `define.model`, the value `penalty=0` causes `marima` to do no search for a (reduced) model. If `penalty=2` the usual AIC is used to identify a reduced model and in a stepwise manner.

This is repeated a few times and from then on, `marima` iterates on the selected model until (hopefully) convergence.

Experience has shown that a first choice `penalty=1` often results in a model which gives a good overview of which coefficients are the most important ones and which are less important. Approximate F-test values for the individual parameter estimates are given in the output object, and they serve the same purpose.

7 Case study

7.1 Australian firearms legislation

Baker & McPhedran (2007) discuss the effect of the Australian firearms legislation of (implemented) 1997 on death rates. Four different (maybe related?) death rates (firearm suicides, firearm homicides, other suicides and other homicides) are considered. Baker & McPhedran analysed the data by conventional univariate ARIMA models with separate models for each of the four death rates. All models estimated were univariate arma(1,1) models, i.e. arma models of order (ar=1,ma=1) and without differerencing.

Here, we shall illustrate the use af `marima` along the same lines, although it is by no means claimed that the results obtained are optimal or represent the best analysis of these data.

7.1.1 Data

The data for the study can be accessed using, for example,
`library(marima); data(austr); all.data <- austr` .

The data.frame `austr` has the following appearance, in that the last 10 lines correspond to not observed future values:

```
  Year  suic.fire homi.fire  suic.other homi.other leg  acc.leg
 1 1915 4.031636  0.5215052    9.166456   1.303763   0       0
 2 1916 3.702076  0.4248284    7.970589   1.416094   0       0
 3 1917 3.056176  0.4250311    7.104091   1.052458   0       0
 4 1918 3.280707  0.4771938    6.621064   1.312283   0       0
 5 1919 2.984728  0.8280212    7.529215   1.309429   0       0
 .   .   .           .           .          .        .       .
 .   .   .           .           .          .        .       .
80 1994 2.4027240 0.2744370    9.297252   1.3385800  0       0
81 1995 2.2023310 0.3209428   10.397440   1.4829770  0       0
82 1996 2.1025940 0.5406671   10.900720   1.1632530  1       1
83 1997 1.7982930 0.4050209   12.901270   1.3284680  1       2
84 1998 1.2024840 0.2885961   13.099060   1.2345500  1       3
85 1999 1.4002010 0.3275942   11.698280   1.4847410  1       4
86 2000 1.2008320 0.3132606   11.099870   1.3365790  1       5
87 2001 1.2980830 0.2575562   11.301570   1.3392920  1       6
88 2002 1.0997420 0.2138386   10.702110   1.4052250  1       7
89 2003 1.0013770 0.1861856   10.099310   1.3334910  1       8
90 2004 0.8361743 0.1592713    9.591119   1.1497400  1       9
91 2005 NA        NA          NA         NA          1      10
92 2006 NA        NA          NA         NA          1      11
 .   .   .         .           .          .          .       .
 .   .   .         .           .          .          .       .
100 2014 NA       NA          NA         NA          1      19
```

It is noted that the data.frame is organised columnwise, while the time series generally should be organised rowwise (see 1.2). This is (if needed) taken care of in `marima` such that the datamatrix is transposed if the number of rows is larger than the number of columns.

The column `leg` indicates whether legislation has been imposed or not (1 or 0). The column `acc.leg` accumulates the legislation.

Note, that `leg` is set to 1 already in 1996. This is because the first effect of `leg` will be for the year *after* 1996 (namely 1997). This is a general feature in time series models where present values depend on previous values. The first year where `leg` can (is believed to) have an effect is therefore 1997.

7.2 Analysis of the four-variate time series

We will estimate the four univariate the models for the four death rates for the period from 1915 to 1996 (both included) as discussed by Baker & McPhedran. In order to define the model the procedure `define.model` is used, and subsequently `marima` is called using the data from the period 1915 to 1996:

```
rm(list=ls())
library(marima)
data(austr)
old.data <- t(austr)[,1:83]
ar<-c(1)
ma<-c(1)
# Define the proper model:
Model1   <- define.model(kvar=7, ar=ar, ma=ma, rem.var=c(1,6,7), indep=c(2:5))
# Now call marima:
Marima1  <- marima(old.data,means=1,
 ar.pattern=Model1$ar.pattern, ma.pattern=Model1$ma.pattern,
             Check=FALSE, Plot=FALSE, penalty=0.0)
short.form(Marima1$ar.estimates, leading=FALSE) # print estimates
short.form(Marima1$ma.estimates, leading=FALSE)
# Can be check'ed using:
  arima(x = old.data[2, ], order = c(1,0,1))
# arima(x = old.data[3, ], order = c(1,0,1))
# arima(x = old.data[4, ], order = c(1,0,1))
# arima(x = old.data[5, ], order = c(1,0,1))
```

Using `define.model` the variables in the data which are irrelevant for the analyses are taken out, `rem.var=c(1,6,7)`, and `indep=c(2:5)` results in the variables 2, 3, 4 and 5 being analysed independently. The estimated model is as follows:

```
 > short.form(Marima1$ar.estimates, leading=FALSE)
, , Lag=0 (unity matrix, not printed here (leading=FALSE))

, , Lag=1

   x1=y1  x2=y2  x3=y3  x4=y4  x5=y5 x6=y6 x7=y7
y1     0  0.0    0.0    0.0    0.0     0     0
y2     0 -0.7932 0.0    0.0    0.0     0     0
y3     0  0.0   -0.7848 0.0    0.0     0     0
y4     0  0.0    0.0   -0.887  0.0     0     0
y5     0  0.0    0.0    0.0   -0.9809  0     0
y6     0  0.0    0.0    0.0    0.0     0     0
y7     0  0.0    0.0    0.0    0.0     0     0

> short.form(Marima1$ma.estimates, leading=FALSE)
, , Lag=0 (unity matrix, not printed here (leading=FALSE))

, , Lag=1

   x1=y1   x2=y2   x3=y3  x4=y4   x5=y5 x6=y6 x7=y7
y1     0  0.0      0.0    0.0     0.0        0     0
y2     0 -0.1317   0.0    0.0     0.0        0     0
y3     0  0.0     -0.4162 0.0     0.0        0     0
y4     0  0.0      0.0    0.0163  0.0        0     0
```

```
y5      0  0.0      0.0     0.0     -0.7104     0      0
y6      0  0.0      0.0     0.0      0.0        0      0
y7      0  0.0      0.0     0.0      0.0        0      0
```

Further statistics are saved in the object `Marima1`. For example the covariance matrix of the residuals (`Marima1$cov.u`):

```
> round(Marima1$resid.cov[2:5,2:5], 4)
       u2     u3      u4      u5
u2 0.1083 0.0161  0.0354  0.0112
u3 0.0161 0.0146  0.0041  0.0020
u4 0.0354 0.0041  0.6582 -0.0041
u5 0.0112 0.0020 -0.0041  0.0224
```

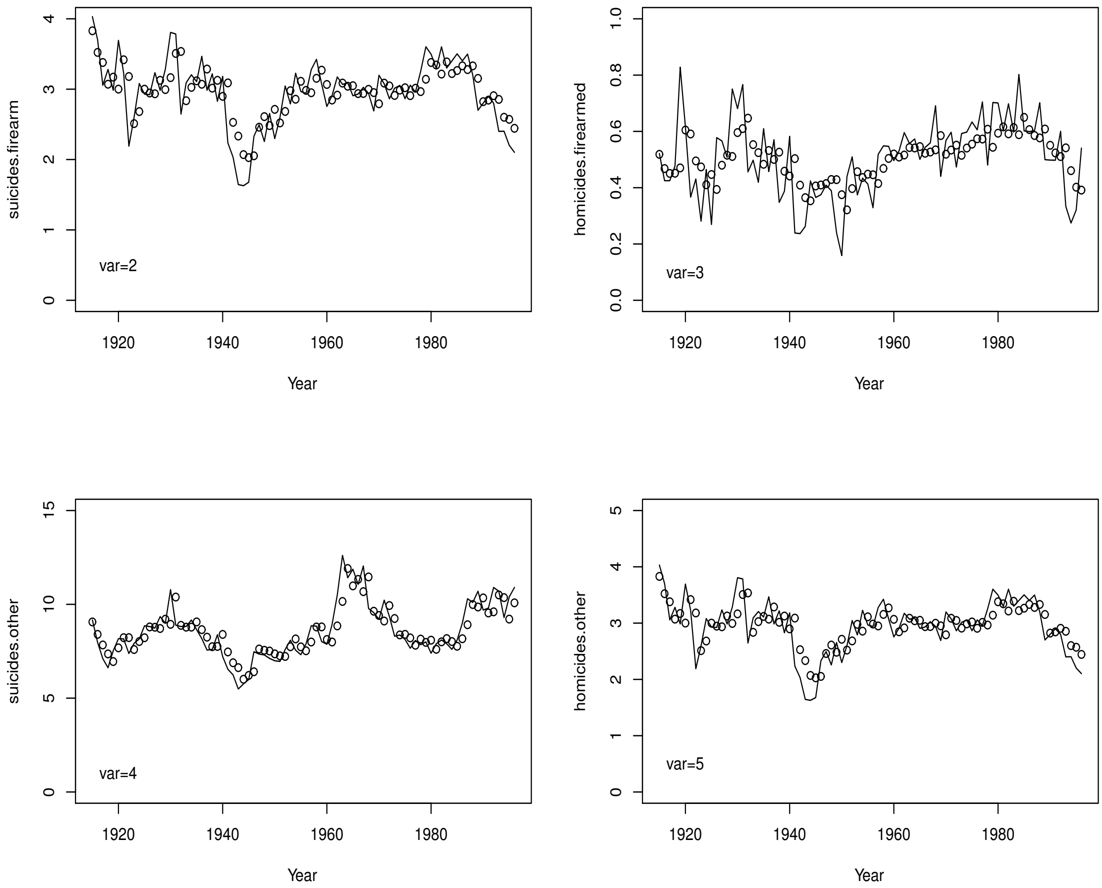
and the covariance matrix of the original variables (`Marima1$cov.y`):

```
> round(Marima1$data.cov[2:5,2:5], 4)
       y2     y3     y4     y5
y2 0.2348 0.0425 0.1110 0.0296
y3 0.0425 0.0199 0.0552 0.0097
y4 0.1110 0.0552 2.3522 0.0790
y5 0.0296 0.0097 0.0790 0.0573
```

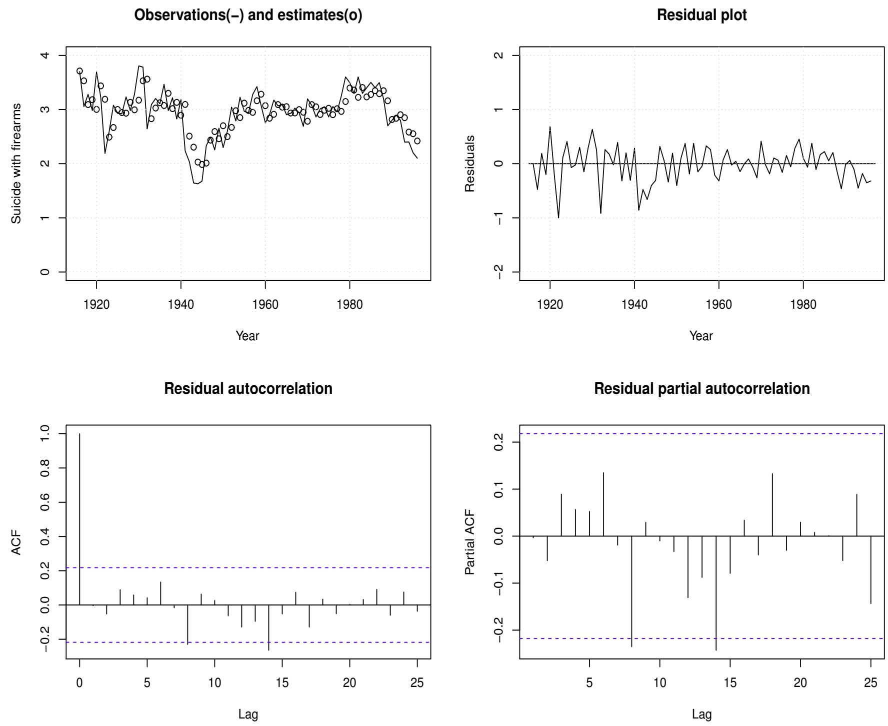
The multiple correlations for the 4 variables are:

```
> round(1-(diag(Marima1$resid.cov[2:5,2:5])/diag(Marima1$data.cov[2:5,2:5])),2)
  u2   u3   u4   u5
0.54 0.27 0.72 0.61
```

The data (lines) and the predictions (circles) are shown in the following plots (1915-1996):



Some relevant model control plots corresponding to the estimated model for the (most) relevant variable (suicides using firearms) are shown below:



We may now estimate the general four-variate arma(1,1) model. The only modification in comparison with the above analysis is the model (Model2) definition statement where `indep=c(2:5)` is taken out (`indep=NULL`).

```
ar<-c(1)
ma<-c(1)
Model2   <- define.model(kvar=7, ar=ar, ma=ma, rem.var=c(1,6,7), indep=NULL)
Marima2  <- marima(old.data, means=1, ar.pattern=Model2$ar.pattern,
     ma.pattern=Model2$ma.pattern, Check=FALSE, Plot=FALSE, penalty=0)


> short.form(Marima2$ar.estimates,leading=FALSE)
, , Lag=1

   x1=y1   x2=y2   x3=y3   x4=y4   x5=y5 x6=y6 x7=y7
y1     0  0.0     0.0     0.0     0.0        0     0
y2     0 -0.5916 -0.8260  0.0545 -0.0583     0     0
y3     0 -0.0933 -0.0868 -0.0034 -0.0861     0     0
y4     0  1.2875 -4.1046 -0.8282 -0.6410     0     0
y5     0  0.1222 -0.6610  0.0164 -0.9458     0     0
y6     0  0.0     0.0     0.0     0.0        0     0
y7     0  0.0     0.0     0.0     0.0        0     0

> short.form(Marima2$ma.estimates,leading=FALSE)
, , Lag=1

   x1=y1   x2=y2   x3=y3  x4=y4   x5=y5 x6=y6 x7=y7
y1     0  0.0     0.0    0.0     0.0       0     0
y2     0  0.0378 -0.1077 0.1750 -0.2258     0     0
```

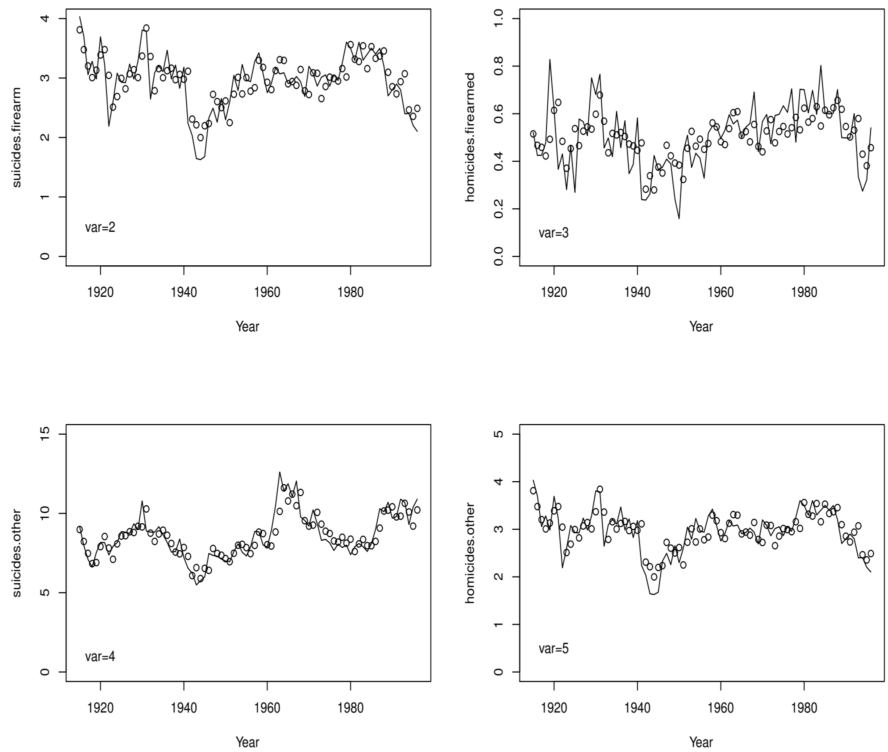
```
y3      0 -0.0146  0.2216 0.0415 -0.0097     0     0
y4      0  1.0700 -2.3897 0.0523 -0.2860     0     0
y5      0  0.0885 -0.5220 0.0308 -0.6557     0     0
y6      0  0.0     0.0    0.0     0.0        0     0
y7      0  0.0     0.0    0.0     0.0        0     0
```

In order to evaluate the improvement in taking into account the correlations between the four variables we may compute

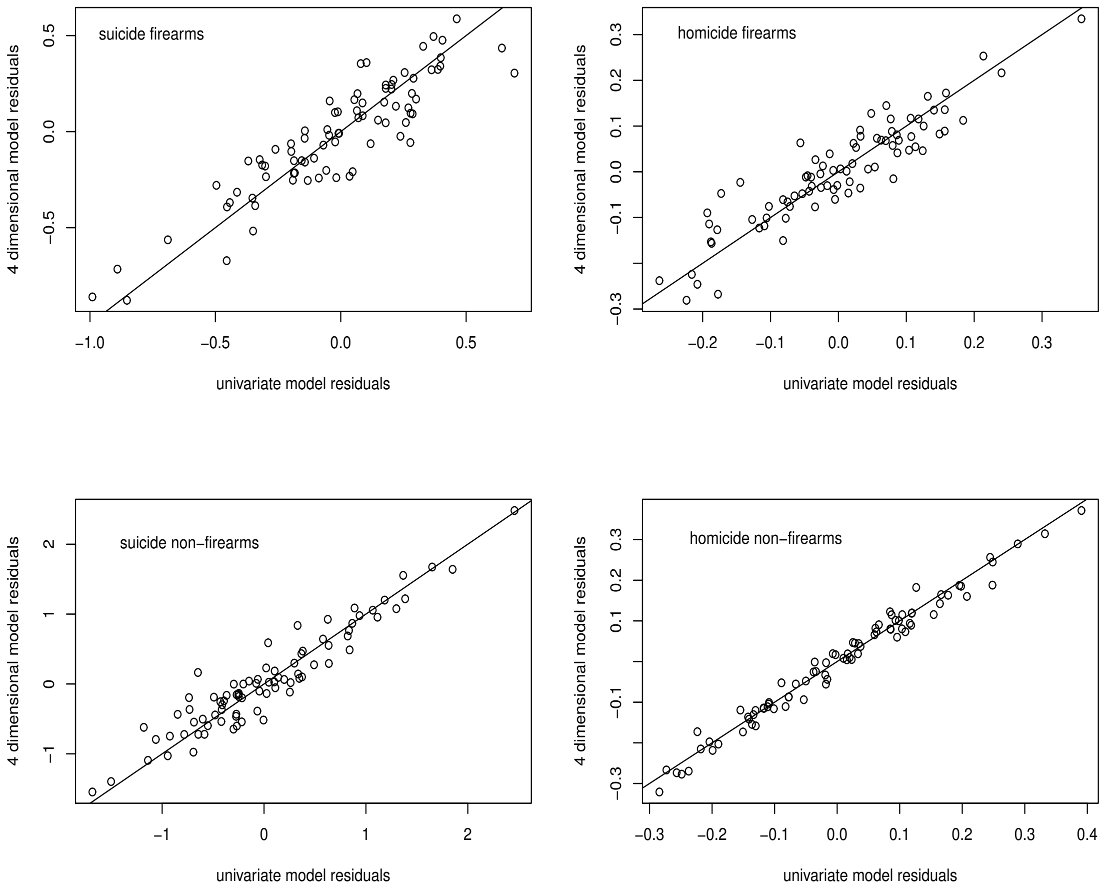
```
>  round(diag(Marima2$resid.cov/Marima1$resid.cov)[2:5], 2)
  u2   u3   u4   u5
0.84 0.89 0.85 1.00
```

so that, for example, the residual variance of the predictions for the first variable (suicides by firearms) estimated by the 4-dimensional model is (only) 84% of the corresponding residual variance for the 4-independent variables model. For the fourth variable (homicides by firearms) there is practically no improvement using the 4-dimensional model.

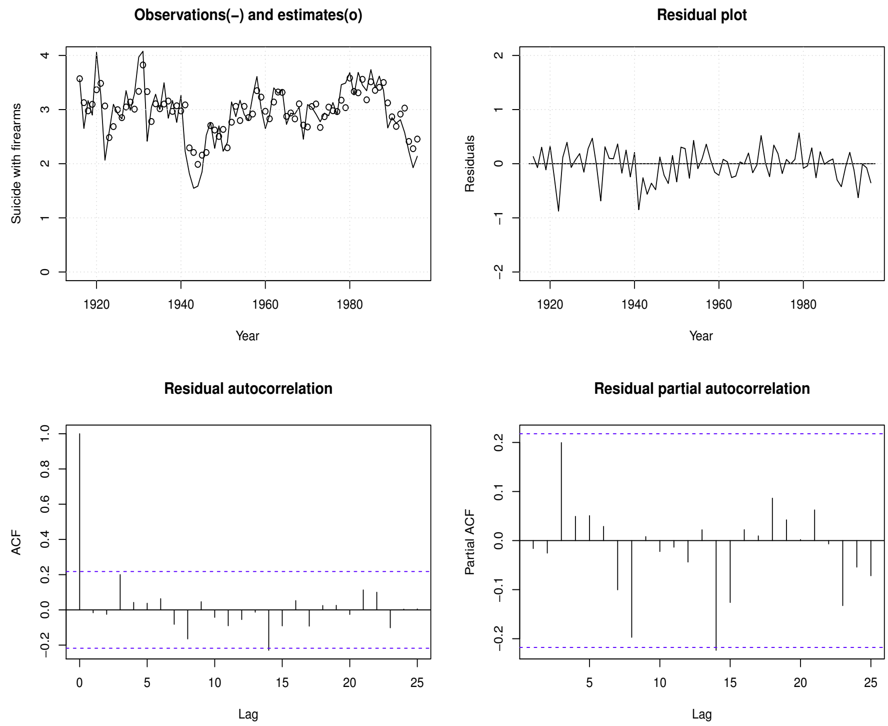
The observations and the predictions for all four varibles are shown below
(lines=predictions, points=data).



A comparison of the residuals from the univariate models and the 4-dimensional model is shown in the following figure. It is seen that the major differences are for the variable no. 2 (suicides using firearms):



The following plots are the same model control plots as shown above for the 'suicides using firearms' data:



It is seen that except for the pronounced improvement in residual variance (by about 14%) the residual autocorrelations and the partial autocorrelations are not very different from the values based on univariate estimation.

7.3 Estimation of legislation effect

We shall now estimate a regression model in which variables 6 and 7 are acting as a regression variables (use `reg.var=c(6,7)` when calling the model definition procedure `define.model`).

7.3.1 Multivariate model with legislation regression

```
library(marima)
data(austr)
all.data<-t(austr)[,1:90]
ar<-c(1)
ma<-c(1)
Model3   <- define.model(kvar=7, ar=ar, ma=ma, rem.var=c(1), reg.var=c(6,7))
Marima3  <- marima(all.data,means=1, ar.pattern=Model3$ar.pattern,
    ma.pattern=Model3$ma.pattern, Check=FALSE, Plot=FALSE, penalty=0)
```

```
, , Lag=1

> short.form(Marima3$ar.estimates,leading=FALSE)
   x1=y1   x2=y2   x3=y3   x4=y4   x5=y5   x6=y6   x7=y7
y1     0  0.0     0.0     0.0     0.0     0.0     0.0
y2     0 -0.3922 -1.6062  0.0532 -0.0742  0.7155 -0.0163
y3     0 -0.0569 -0.2544 -0.0024 -0.0812  0.0773  0.0107
y4     0  0.8260 -2.7354 -0.7853 -0.5335 -0.9404  0.3087
y5     0  0.1167 -0.6289  0.0140 -0.9501 -0.0257  0.0084
y6     0  0.0     0.0     0.0     0.0     0.0     0.0
y7     0  0.0     0.0     0.0     0.0     0.0     0.0

> short.form(Marima3$ma.estimates,leading=FALSE)
, , Lag=1

   x1=y1  x2=y2   x3=y3  x4=y4   x5=y5 x6=y6 x7=y7
y1     0 0.0     0.0    0.0     0.0        0     0
y2     0 0.2052 -0.8038 0.1557 -0.3363     0     0
y3     0 0.0176  0.0605 0.0374 -0.0181     0     0
y4     0 0.7415 -1.0624 0.0903 -0.0894     0     0
y5     0 0.0879 -0.4930 0.0221 -0.6900     0     0
y6     0 0.0     0.0    0.0     0.0        0     0
y7     0 0.0     0.0    0.0     0.0        0     0
```

One can assess the model coefficients by means of the `Marima3$ar.fvalues` and `Marima3$ma.fvalues` giving:

```
> round(short.form(Marima3$ar.fvalues, leading=FALSE), 2)
, , Lag=1

   x1=y1 x2=y2 x3=y3 x4=y4 x5=y5 x6=y6 x7=y7
y1     0  0.00  0.00  0.00  0.00  0.00  0.00
y2     0  1.70  1.03  1.75  0.11  5.57  0.10
y3     0  0.25  0.18  0.03  0.89  0.46  0.30
y4     0  1.22  0.48 61.98  0.89  1.56  5.66
y5     0  0.61  0.64  0.50 71.42  0.03  0.11
y6     0  0.00  0.00  0.00  0.00  0.00  0.00
y7     0  0.00  0.00  0.00  0.00  0.00  0.00

> short.form(Marima3$ma.fvalues, leading=FALSE)
 , , Lag=1

   x1=y1 x2=y2 x3=y3 x4=y4 x5=y5 x6=y6 x7=y7
y1     0  0.00  0.00  0.00  0.00     0     0
y2     0  0.41  0.25  6.50  1.03     0     0
y3     0  0.02  0.01  2.62  0.02     0     0
y4     0  0.87  0.07  0.35  0.01     0     0
y5     0  0.31  0.39  0.53 17.77     0     0
y6     0  0.00  0.00  0.00  0.00     0     0
y7     0  0.00  0.00  0.00  0.00     0     0
```

7.3.2 Multivariate model with legislation regression, 'penalty' reduced

A model reduction/identification can be performed using the option 'penalty', for example `penalty=1`. We consider all data, including the period where the legislation may have effect.

```
library(marima)
data(austr)
all.data<-t(austr)[,1:90]
ar<-c(1)
ma<-c(1)
Model4   <- define.model(kvar=7, ar=ar, ma=ma, rem.var=1, reg.var=c(6,7))
Marima4  <- marima(all.data, means=1, ar.pattern=Model4$ar.pattern,
     ma.pattern=Model4$ma.pattern, Check=FALSE, Plot=FALSE, penalty=1)
```

The `means=1` declaration (default) ensures that all variables are means adjusted before analysis. It is equivalent to `means=c(1,1,1,1,1,1,1)`.

```
round(short.form(Marima4$ar.estimates,leading=FALSE),4)
, , Lag=1

   x1=y1   x2=y2   x3=y3   x4=y4   x5=y5   x6=y6  x7=y7
y1     0  0.0000  0.0000  0.0000  0.0000  0.0000 0.0000
y2     0 -0.5619 -0.8153  0.0342  0.0000  0.5733 0.0000 (suicide with f.a.)
y3     0 -0.0608 -0.3171  0.0000 -0.0669  0.0966 0.0000 (homicide with f.a.)
y4     0  0.6533 -1.6631 -0.8268 -0.4720 -0.9667 0.3253 (suicide without f.a.)
y5     0  0.0000  0.0000  0.0000 -0.9801  0.0000 0.0000 (homicide without f.a.)
y6     0  0.0000  0.0000  0.0000  0.0000  0.0000 0.0000
y7     0  0.0000  0.0000  0.0000  0.0000  0.0000 0.0000

> round(short.form(Marima4$ma.estimates,leading=FALSE),4)
, , Lag=1

   x1=y1  x2=y2 x3=y3  x4=y4   x5=y5 x6=y6 x7=y7
y1     0 0.0000     0 0.0000  0.0000     0     0
y2     0 0.0000     0 0.1303 -0.2351     0     0
y3     0 0.0000     0 0.0391  0.0000     0     0
y4     0 0.5857     0 0.0000  0.0000     0     0
y5     0 0.0000     0 0.0000 -0.7163     0     0
y6     0 0.0000     0 0.0000  0.0000     0     0
y7     0 0.0000     0 0.0000  0.0000     0     0
```

It is seen that generally many of the regression coefficients for the intervention (x6) and the regression (x7) in the *penalty=1* reduced model are 0 (zero). For variable y2 (suicides with firearms) a constant *decrease* of 0.5733 and no annual *decrease or increase* from 1997 and onwards is found. For variable 3 (homicide with firearms) a small constant *increase* of 0.0966 and practically no annual *change* is found. For variable 4 (suicide without use of firearms) a constant *increase* of 0.9667 and an annual *decrease* of 0.3253 per year is found, but no change of level. For variable 5 (homicide without use of firearms) no effect from the legislation is found.

One might conclude that the level of the rate of suicides using firearms is decreased by about 0.5733 with no annual effect. But suicides without using firearms decreases by about 0.3253 per year after an initial increase of about 0.9667. The rate of homicides (with or without the use of firearms) is generally not affected by the legislation.

In order to asses the significance of the model found one may use the F-values of the ar-part of the estimated model:

```
> round(short.form(Marima4$ar.fvalues, leading=FALSE), 2)
, , Lag=1

   x1=y1 x2=y2 x3=y3  x4=y4  x5=y5 x6=y6 x7=y7
y1     0  0.00  0.00   0.00   0.00  0.00  0.00
y2     0 36.96  6.79   1.43   0.00  8.53  0.00
y3     0  3.04  7.54   0.00   1.44  2.14  0.00
y4     0  5.11  4.55 181.42   1.59  1.88  6.48
y5     0  0.00  0.00   0.00 138.38  0.00  0.00
y6     0  0.00  0.00   0.00   0.00  0.00  0.00
y7     0  0.00  0.00   0.00   0.00  0.00  0.00
```

An F-value=2.85, having 1 and around 90 degrees of freedom (length of time series), corresponds to a p-value≃10% . Therefore, the dependence of the legislation is only highly significant for variables 2 (suicides with firearms) and 4 (homicide with firearms) with p-values below 1% (1-pf(6.48,1,90)≃1.3%).

Further, it is seen that variable 5 does not seem to depend on any of the other variables (2, 3, 4), neither in the autoregressive nor in the moving average part of the model:

$$(y_{5,t} - 1.104) - 0.5619 \cdot (y_{5,t-1} - 1.104) = u_{5,t} - 0.7163 \cdot u_{5,t-1}$$

in that the mean of the observed y_5, 1.104, was subtracted from the observations before the marima-estimation.

One may analyse the identified model a little further by taking out the arma-part of the model, calculate its random shock form with many lags (100 say), and print the 25th term:

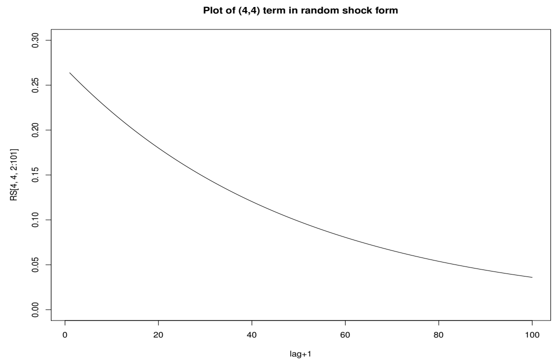
```
 AR<-Marima4$ar.estimates[2:5,2:5,1:2]
 MA<-Marima4$ma.estimates[2:5,2:5,1:2]
 RS<-rand.shock(ar=AR, ma=MA, L=100)
 colnames(RS)<-rownames(RS)<-c(2:5)
# Note that dim(RS)=(4,4,100+1)
> round(RS[,,25+1], 4)
        2       3       4       5
2  0.0085 -0.0008 -0.0047 -0.0385
3  0.0009 -0.0001 -0.0005  0.0130
4 -0.0607  0.0062  0.0339  0.7906
5  0.0000  0.0000  0.0000  0.1628
```

It is seen that, for example, the coefficient [4,4] corresponding to the autoregressive term for 'homicide without use of firearms' is not close to 0. This indicates that the identified model for 'homicide without use of firearms' may be close to being non-stationary. For lag=100 (position 100+1) we find:

```
> round(RS[,,100+1], 6)
         2     3      4        5
2  2.0e-06 0e+00 -1e-06 -0.009376
3  0.0e+00 0e+00  0e+00  0.002780
4 -1.3e-05 1e-06  7e-06  0.181099
5  0.0e+00 0e+00  0e+00  0.036041
```

which, again, demonstrates that the arma-part of the model is close to being non-stationary. For example, the code

```
> plot(RS[4,4,2:101], type="l", xlab="lag+1", ylim=c(0.0,0.30),
+     main="Plot of (4,4) term in random shock form")
```



gives an almost perfectly exponentially, but very slowly, decreasing plot of the (4,4) coefficients in the random shock representation of the arma model.

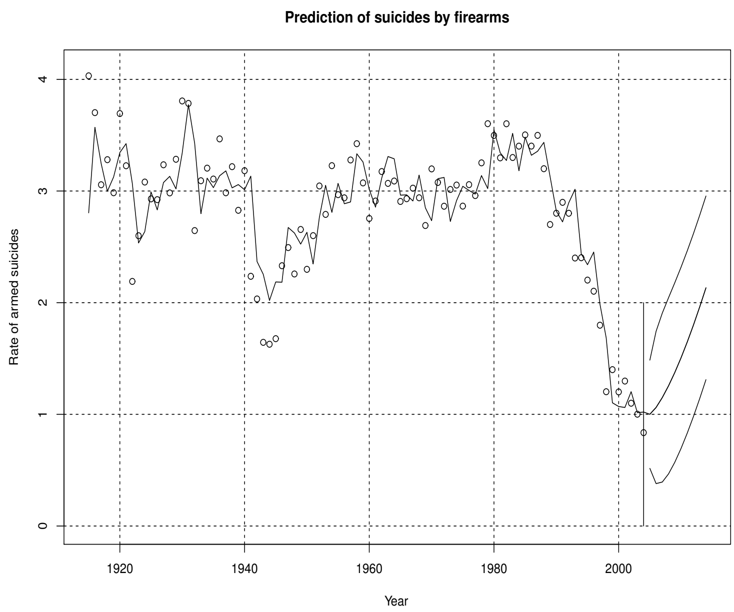
7.4 Prediction of timeseries

The routine called `arma.forecast` is used. We start by estimating our model (as before), and then (using the prediction-prepared data) we use `arma.forecast`, and all output is saved in the object created:

```
library(marima)
# Four variate timeseries of order ARMA(1,1) with intervention/regression:
data(austr)
all.data<-t(austr)
# austr data.frame been prepared so that future values of regression
# are put in the future positions (from no. 91 and onwards (2005-2014)):
Model5   <- define.model(kvar=7, ar=c(1), ma=c(1), rem.var=c(1), reg.var=6:7)
Marima5  <- marima(all.data[,1:90], Model5$ar.pattern, Model5$ma.pattern
      , Check=FALSE, Plot=FALSE, penalty=1)
# call the forecasting function using Marima and the prepared data:
Forecasts <-
    arma.forecast(all.data[,1:100],nstart=90,nstep=10,marima=Marima5)
### From here on the plot is constructed ###
  Year<-series[1,91:100];
  Predict<-Forecasts$forecasts[2,91:100]
  stdv<-sqrt(Forecasts$pred.var[2,2,])
  upper.lim=Predict+stdv*1.645
  lower.lim=Predict-stdv*1.645
  Out<-rbind(Year,Predict,upper.lim,lower.lim)
  print(Out)
  # plot results:
  plot(series[1,1:100], Forecasts$forecasts[2,],type="l", xlab="Year",
  ylab="Rate of armed suicides", main="Prediction of suicides by firearms",
        ylim=c(0.0,4.1))
  lines(series[1,1:90], series[2,1:90], type="p")
  grid(lty=2, lwd=1, col="black")
  Years<-2005:2014
  lines(Years, Predict, type="l")
```

```
lines(Years, upper.lim, type="l")
lines(Years, lower.lim, type="l")
lines(c(2004,2004), c(0,2))
```

The data (o), the 1-step-ahead forecasts (–) and the nstep=10 forecast (–) and a 90% prediction interval for the forecast are shown in the plot below. Note, that the prediction interval is computed from the marima-estimates and without taking the estimation uncertainty into account.



8 References

[1] Baker, J. & McPhedran, S. (2007) Gun Laws and Sudden Death, British Journal of Criminology, 47: pp. 455-469.

[2] Hamilton, J.D. (1994) Time Series Analysis, Princeon University Press, pp. 438-444.

[3] Jenkins, G.M. & Alavi, A.S. (1981) Some Aspects of Modelling and Forecasting Multivariate Time Series, Journal of Time Series Analysis, Vol. 2, no 1.

[4] Madsen, H. (2008) Time Series Analysis, Chapmann & Hall (in particular chapter 9: Multivariate time series).

[5] Reinsel G.C. (2003) Elements of Multivariate Time Series Analysis, Springer Verlag, 2^{nd} ed. pp. 106-114.

[6] Spliid, H. (1983) A Fast Estimation Method for the Vector Autoregressive Moving Average Model with Exogeneous Variables, Journal of the American Statistical Association, Vol.78, no.384.

. o O o .

[end-of-document]