SIMULATING and ESTIMATING ARMA MODELS

Goals:

- Learn how to use \texttt{R} to simulate AR, MA, and ARMA processes.
- Learn how to estimate ARMA and its essential characteristics.

Assignments:

1. Simulate causal and invertible ARMA(2,2) model of your choice with a fixed Normal iid noise (simulate and store the noise), of length 50.

2. Estimate the four model parameters, acf, acvf, compare them with the respective theoretical values/functions. Compare the estimated residuals with the true ones, find the correlation between the two sets. Discuss the quality of the estimation.

3. Repeat the analysis of 1-2 increasing the length of the time series. Find such a length $N_0$ that the four parameters are estimated with less than 5% relative error.

4. Estimate the model parameters assuming that the true order of the model is ARMA(1,1). Compare the quality of estimation, using the variance of the estimated residuals and comparing the estimated and true residuals. Discuss the results.

5. Use \texttt{R} to estimate the variance of an ARMA(1,1) process

$$X_t = Z_t + 0.5Z_{t-1} + 0.9X_{t-1},$$

where $Z_t$ is a standard white noise.

Report: All assignments require a printed report. Your goal is to support your conclusions using theoretical and/or practical arguments using \texttt{R} (do not forget about nice graphs.)

Due date: Thursday, November 3, 2016 in class.

\texttt{R} commands used in these handouts are collected in R-files \texttt{Lab3.R}, which is available on the course Web site.
1. Introduction

ARMA models play an important role in time series analysis; mainly because of their flexibility in fitting a given autocorrelation function. It can be shown that one can find an ARMA model that will exactly match an arbitrary number of the first values of any acf. In particular, if the acf of a time series has only a finite number of non-zero values, it can be fit exactly with a MA(q) process, and if the pacf of a time series has only a finite number of non-zero values, it can be fit exactly by an AR(p) process.

The first step in studying the techniques of ARMA modeling is to learn how to simulate ARMA time series, how to find acf and pacf of a given ARMA model, and how to find an MA representation of a given ARMA model. These topics are considered in Sections 2-7.

The next step is to estimate ARMA parameters from the observed time series; this is discussed in Sections 8-9.

2. Simulating MA(q)

In R, there are two ways of modeling MA(q) processes. The first method is based on the command

```R
X<-filter(Z,filter,sides=1)
```

It uses the following model representation:

\[ X_t = \theta_1 Z_t + \theta_2 Z_{t-1} + \ldots + \theta_q Z_{t-q}. \]

**Inputs:**

- \(Z\) is the time series of innovations; it should be a white noise in order to obtain a proper MA model. Caution: \(R\) does not check whether \(Z\) is a white noise and will work with any time series!

- \(filter\) is a vector with MA coefficients. For example, \(filter=c(1,2)\) corresponds to a MA(1) process
  \[ X_t = Z_t + 2Z_{t-1}. \]

  \texttt{sides = 1} for MA models.

**Output** is an object of the time series class. No check is made whether the resulting MA model is invertible.

An alternative method of MA simulation will be considered in Section 4 below.

3. Simulating AR(p)

In R, there are two ways of modeling AR(p) processes. The first method is based on the command

```R
X<-filter(Z,filter,method='r')
```
It uses the following model representation:

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

**Inputs:**

- \( Z \) is the time series of innovations; it should be a white noise in order to obtain a proper AR model. Caution: \( Z \) does not check whether \( Z \) is a white noise and will work with any time series!
- \( \text{filter} \) is a vector with regression coefficients. For example, \( \text{filter} = c(0.9) \) corresponds to an AR(1) process \( X_t = Z_t + 0.9X_{t-1}. \)
- \( \text{method} = \text{r} \) for AR models.

**Output** is an object of the time series class. No check is made whether the resulting AR model is causal, the output may diverge if it is not.

An alternative method of AR(\( p \)) simulation is considered in Section 4 below.

### 4. Simulating ARMA(p,q)

The most general way of simulating ARMA models is implemented in the command

\[ \text{X<-arima.sim}(n, \text{model}, \ldots) \]

It uses the following model representation:

\[ X_t = \phi_1 X_{t-1} + \ldots + \phi_p X_p + Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q}. \]

**Basic inputs:**

- \( n \) is the length of the output time series
- \( \text{model} \) is a list with model parameters. Notice that here we explicitly assume a unit coefficient by \( Z_t \).

For example,

\[ \text{X<-arima.sim}(n=100, \text{list}(\text{ar}=c(0.9), \text{ma}=c(0.2))) \]

produces an ARMA(1,1) model \( X_t = 0.9X_{t-1} + Z_t + 0.2Z_{t-1}. \)

**Output** is an object of the time series class. No check is made whether the MA part of the model is invertible. The AR part is checked for stationarity and causality, an error message will appear if an attempt is made to simulate a non-causal model.

The command \( \text{arima.sim} \) allows several other inputs that can be very useful in applied modeling:
Additional inputs

sd is the standard deviation of the normal white noise (sd=1 by default)
rand.gen is the name of a function used to generate innovations (rand.gen = rnorm by default)
innov is a time series of innovations (useful to model several models with the same white noise)
arima.sim can be used to generate MA models (by omitting ar part), AR models (by omitting ma part), or white noises (by submitting an empty list of parameters).

5. Theoretical ACF and PACF computation

allows to compute the theoretical values of the autocorrelation function for stationary ARMA processes (do not confuse this with estimation of the acf from a given time series). This is done with the command (mind the capital letters)

> Acf<-ARMAacf(ar=...,ma=...,lag.max=r)

Example:

> Acf<-ARMAacf(ar=c(.2),ma=c(.3,.2),lag.max=4)

computes the first five (for lags 0,1,2,3,4) values of the acf of the process

\[ X_t = 0.2X_{t-1} + Z_t + 0.3Z_{t-1} + 0.2Z_{t-2}, \]

which is \( \rho(0)=1 \), \( \rho(1)=0.498 \), \( \rho(2)=0.248 \), \( \rho(3)=0.050 \), and \( \rho(4)=0.010 \).

You also can compute the theoretical partial autocorrelation function using the command

> Acf<-ARMAacf(ar=...,ma=...,lag.max=r,pacf=TRUE)

6. MA(\( \infty \)) representation of ARMA model

A moving average representation of an ARMA model,

\[ X_t = \sum_{k=0}^{\infty} \psi_k Z_{t-k}, \]

is calculated by the command

> ARMAtoMA(ar=..., ma=..., lag.max=r)

Notice that the calculated coefficients do not include the first 1, for example:

> P<-ARMAtoMA(ar=0.9, lag.max=2)
results in $p=(0.9, 0.81)$. Also, there is no check for causality, so the resulting series may diverge.

7. Variance computations

Notably, the command ARMAtoMA() can be used to compute the variance of a particular ARMA model. For that we compute a long enough series $p$ of MA coefficients (say, $r=100$), and then sum their squares plus 1:

- $\text{sum}(P^2)+1$

Example:

- `P<-ARMAtoMA(ar=0.9, lag.max=100)`
- `V<-sum(P^2)+1`
- `V`
  - `[1] 5.263158`

We know that the theoretical variance for AR(1) model with coefficient $\varphi$ and white noise of unit variance is given by

$$V = \frac{1}{1-\varphi^2} = \frac{1}{1-0.9^2} = 5.263158.$$  

Thus, the numerical approach gives a very accurate approximation.

8. ARMA estimation

Estimation of the ARMA parameters is done with help the command

- `arima(X, order = c(p, d, q)).`

We will discuss here a simple use of his command, and will proceed with more advanced applications in the next Labs.

The triplet $(p,d,q)$ specifies the order of the model. In this Lab will only vary the AR order $p$ and the MA order $q$, keeping the order of differencing $d=0$. Also, we will only apply the estimation procedure to the true ARMA time series with known values of $p$ and $q$.

9. ACF, ACVF, and PACF estimation

Estimation of ACF, ACVF, and PACF for any time series (not necessarily ARMA) is done using the command

```r
> acf(x,lag.max = r,type = c("correlation", "covariance", "partial"),plot = TRUE)
```
Appendix 1: Lab3.R

```r
#=================================#
#             STAT 758            #
#          ARMA modeling          #
#=================================#

# White noises used in simulations
#=================================
Z1 <- rnorm(100)
Z2 <- rnorm(100, mean=10, sd=2)

# MA simulation by filter()
#===========================
X <- filter(Z1, rep(.1, 10))
X <- filter(Z2, c(.6, .3, .1))

# Plot time series
#========================
par(bg='yellow')
plot(X, col='blue', type='o', pch=0)
grid()

# AR simulation by filter()
#===================================
X <- filter(Z1, rep(.1, 10), method='r')
X <- filter(Z2, c(.6, .3, .1), method='r')

# ARMA simulation using arima.sim()
#===================================
X <- arima.sim(n=1000, list(ar=c(0.9), ma=c(0.2)))  # ARMA(1,1)
X <- arima.sim(n=1000, list(ar=c(0.9)))  # AR(1)
X <- arima.sim(n=1000, list(ma=c(0.2)))  # MA(1)

X <- arima.sim(n=1000, list(ma=c(0.2)), sd=10)  # MA(1) with WN(0,10)
X <- arima.sim(n=1000, list(ar=c(0.9)), innov=Z1)  # AR(1) with given WN Z1
X <- arima.sim(n=1000, list(ar=c(0.9)), rand.gen=rt, df=2)  # AR(1) with mild
long-tail (Student df=2)
X <- arima.sim(n=1000, list(ar=c(0.9)), rand.gen=runif)  # AR(1) with U[0,1]

# Plot time series
#========================
par(bg='yellow')
plot(X, col='blue', type='o', pch=0)
grid()
```
# ACF and PACF computations (theoretical)
#========================================================
A<-ARMAacf(ar=c(.2),ma=c(.3,.2),lag.max=20)
P<-ARMAacf(ar=c(.2),ma=c(.3,.2),lag.max=20,pacf=TRUE)

# MA representation for ARMA
#==================================
P<-ARMAtoMA(ar=0.9, lag.max=5)
P<-ARMAtoMA(ar=c(0.9,.1),ma=c(1,2,3), lag.max=5)

# Variance computation
#==================================
P<-ARMAtoMA(ar=0.9, lag.max=100)
sum(P^2)+1

# ARMA estimation
#==================================
L<-100
Z<-rnorm(L)
X<-arima.sim(n=L,list(ar=0.5),innov=Z)
est<-arima(X,order=c(1,0,0))
plot(est$residuals,Z)

# ACF, PACF estimation
#==================================
X<-arima.sim(n=2000,list(ar=0.9)) # AR(1) model
acf(X,lag.max = 20,type = c("partial"),plot = TRUE,lwd=2,col='blue')
points(seq(0,20),ARMAacf(ar=0.9,lag.max=20),pch=19,col='green') # adds theoretical values
#points(seq(1,20),ARMAacf(ar=0.9,lag.max=20,pacf=TRUE),pch=19,col='green')