Estimation of AR models

- Recall that the AR(\(p\)) model is defined by the equation

\[ X_t = \sum_{j=1}^{p} \phi_j X_{t-j} + \epsilon_t \]

where \(\epsilon_t\) are assumed independent and following a \(N(0, \sigma^2)\) distribution.

- Assume \(p\) is known and define \(\phi = (\phi_1, \phi_2, \phi_3, \ldots, \phi_p)'\), the vector of model coefficients.

- Given the data \(x_1, x_2, x_3, \ldots, x_n\), we want to estimate \((\phi, \sigma^2)\).
Method of moments

- Recall that Yule Walker’s equations establish that,

\[ \rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \ldots + \phi_p \rho_{k-p} \]

- If we write these equations for \( k = 1, 2, \ldots, p \), we obtain a system for \( \phi_1, \phi_2, \ldots, \phi_p \).

- We can solve this system with an estimator for the autocorrelation \( \rho_k \).

- For example, we could use the sample autocorrelation \( \hat{\rho}_k = r_k \) and then solve the \( p \) equations for \( \phi \).

- This method is implemented in R using the function `ar`.

  # For example try
  a = ar(x, method='yule walker')
list(a)
a$ar
# this gives the MOM estimate of ph
# for your data

**Maximum likelihood estimation**

- For MLE, first we need to find the likelihood function for the AR model.

- Since the AR process has a Markovian structure, the joint density of the data is given by the expression

\[
p(x_1, x_2, \ldots, x_n | \phi, \sigma^2) = p(x_1, x_2, \ldots, x_p | \phi, \sigma^2) \prod_{t=p+1}^{n} p(x_t | x_{t-1}, \ldots, x_{t-p}, \phi, \sigma^2)
\]
- Assume the first \( p \) observations (initial values) \((x_1, x_2, \ldots, x_p)\) are completely known.

- Then the model likelihood is defined by ignoring \( p(x_1, x_2, \ldots, x_p) \) from the above expression, i.e.

\[
p(x_1, x_2, \ldots, x_n | \phi, \sigma^2) \propto \prod_{t=p+1}^{n} p(x_t | x_{t-1}, \ldots, x_{t-p}, \phi, \sigma^2)
\]

- What is \( p(x_t | x_{t-1}, \ldots, x_{t-p}, \phi, \sigma^2) \)? From the AR model definition, \( x_t \) can be seen as the response of a linear regression with “regressors” \( x_{t-1}, x_{t-2}, x_{t-3}, \ldots, x_{t-p} \), then

\[
p(x_t | x_{t-1}, \ldots, x_{t-p}, \phi, \sigma^2) = N(x_t | f_i' \phi, \sigma^2)
\]

where \( f_i' = (x_{t-1}, x_{t-2}, x_{t-3}, \ldots, x_{t-p}) \).
• Define $F$ to be a matrix with rows $f'_i; i = p+1... n$ and $x = (x_{p+1}, x_{p+2}, \ldots, x_n)$.

• The likelihood function of the AR model conditional on the initial values is a multivariate Normal of dimension $n - p$ with mean $F\phi$ and covariance $\sigma^2 I_{(n-p)\times(n-p)}$, i.e. $N(x|F\phi, \sigma^2 I_{(n-p)\times(n-p)})$.

• The Maximum Likelihood Estimator (MLE) for $(\phi, \sigma^2)$ is

\[
\hat{\phi} = (F'F)^{-1}F'x
\]

\[
s^2 = R/(n - p)
\]

where $R = (x - F\phi)'(x - F\phi)$.

• For an unbiased estimator of $\sigma^2$, we use

\[
s^2_1 = R/(n - 2p)
\]
The MLE is also given by the “ar” function available in R/Splus.

```r
a = ar(x, method = "mle")
a$ar  # AR coefficients
a$var  # AR variance
```

- All these results are valid if we ignore the uncertainty due to the initial values.

- The complete likelihood of the model considers the extra part, \( p(x_1, x_2, \ldots, x_p | \phi, \sigma^2) \), which is a complicated function of the parameters.

- For the complete likelihood we require numerical methods (Newton-Raphson) to obtain the MLE of the AR model.

**Bayesian analysis of AR(\(p\)) model**
• In the context of linear regression (initial values known), we can use a non-informative prior for the parameters,

\[ p(\phi, \sigma^2) \propto 1/\sigma^2 \]

• Using Bayes theorem, the posterior distribution for \((\phi, \sigma^2)\) is given by:
  - \( \phi \) conditional on \( \sigma^2 \) follows a multivariate Normal \( N(\phi|\hat{\phi}, \sigma^2 (F' F)^{-1}) \).
  - The marginal distribution for \( \sigma^2 \) follows an Inverse Gamma posterior \( IG((n - 2p)/2, R/2) \).
  - The marginal distribution for \( \phi \) follows multivariate t distribution with \( n - 2p \) degrees of freedom and location parameter \( \hat{\phi} \).

• For posterior inference using the complete model
likelihood, we require numerical techniques such as Markov chain Monte Carlo (MCMC) methods.

**Inference on characteristic reciprocal roots**

- For this, we need to find the solutions to the equation $\Phi(B) = 0$ where $\Phi(B)$ is the characteristic polynomial of the AR process.

- We have close form expressions for this characteristic roots if $p = 1, 2$, but if $p > 2$ it becomes really hard to obtain the solutions.

- We can use the R/Splus function `polyroot` to find the roots.

```r
ph = c(2*0.95*cos(0.5), -0.95^2)
ph
```
\begin{verbatim}
[1]  1.667407  -0.902500
polyroot(c(1,-ph))
[1]  0.9237711+0.5046585i  0.9237711-0.5046585i
1/polyroot(c(1,-ph))
[1]  0.8337034-0.4554543i  0.8337034+0.4554543i
# For modulus and frequencies
Mod(1/polyroot(c(1,-ph))))
[1]  0.95  0.95
Arg(1/polyroot(c(1,-ph))))
[1]  -0.5   0.5
2*pi/Arg(1/polyroot(c(1,-ph))))
[1]  -12.56637  12.56637
\end{verbatim}

• Given some estimate \( \hat{\phi} \), we can compute estimates \( \hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_p \) for the reciprocal roots.
• Bayesian inference for roots. Mapping from $\phi$ to “roots” is mathematically intractable (unless $p = 1, 2$).

• In general, there is no close form expression to the posterior distribution of the $\alpha$’s, although we know $(\phi, \sigma^2)$ follow a Normal/Inverse Gamma posterior.

• We will rely on Monte Carlo simulation to study the posterior distribution for the $\alpha$’s.

• Monte Carlo simulation scheme:
  - Simulate a value $\sigma^2$ from an $IG((n - 2p)/2, R/2)$ distribution.
  - Simulate the vector of coefficients $\phi$ from $N(\phi|\hat{\phi}, \sigma^2(F'F)^{-1})$ distribution.
  - With the simulated value for $\phi$ solve the equation
\( \Phi(B) = 0 \). This leads to one generate sample from the posterior distribution of the \( \alpha' \)'s.

- Iterate until we collect M samples and summarize samples.

- This algorithm produces “exact” Monte Carlo samples of a posterior distribution. It does not require convergence monitoring or a burn-in period.

- To simulate a multivariate Normal distribution, we need to use Cholesky’s decomposition (\( \text{chol} \) function in R).

- If \( z \) is a k-dimensional vector that follows a multivariate \( N(z|m,V) \), where \( m \) is the mean and \( V \) is the covariance matrix, this function \( \text{chol} \) allows us to find a matrix \( A \) such that \( V = AA' \).
• To simulate a \( z \) vector, we generate \( y_1, y_2, \ldots, y_k \) iid \( \text{N}(0,1) \) random deviates and make

\[
z = m + Ay
\]

where \( y = (y_1, y_2, \ldots, y_k) \) and \( A \) is the Cholesky’s decomposition of \( V \).

• If \( V \) is numerically close to a singular matrix, we could use the Singular Value Decomposition of \( V \) (svd) instead of Cholesky’s decomposition.
Identification of AR roots

- The AR model is invariant for different labeling of the $\alpha$’s, since the characteristic polynomial
  \[ \phi(B) = \prod_{i=1}^{p} (1 - \alpha_i B) X_t. \]
- The values of the AR coefficients are $\phi$ are invariant to permutations of the sub-indices for the $\alpha'$s
- For identification, complex reciprocal roots are ordered by modulus ($r'$s) or by frequencies ($\omega'$s)
- If we have $C$ complex pairs of reciprocal roots ordered by modulus then
  \[ \alpha_1 = r_1 \exp(\pm i\omega_1); \alpha_2 = r_2 \exp(\pm i\omega_2); \ldots; \alpha_C = r_C \exp(\pm i\omega_C) \]
  with the condition, $r_1 > r_2 > r_3 \ldots > r_C$. 
• Ordering the roots by frequencies means that \\
  \[ \omega_1 < \omega_2 < \ldots < \omega_C. \]

• For the case of real roots the natural thing to do, is to order them from the smallest to the largest. For \( R \) real roots,
\[ r_1 < r_2 < r_3 \ldots < r_R \]

• **Example** EEG trace of 400 observations. The ACF/PACF of this series suggests the use of an AR model with \( p = 8, 9, \text{ or } 10. \)

• Fitting an AR(10) using R/Splus (it could also been an AR(8) or AR(9) gives the following MOM estimator for the parameters are
\[ \hat{\phi} = (0.27, 0.03, -0.16, -0.18, -0.14, -0.15, -0.23, -0.1, -0.05, -0.11) \]
and $\hat{\sigma}^2 = 3808.58$.

- The reciprocal roots denoted by $(r_i, \omega_i)$ and associated to this $\hat{\phi}$ vector are:
  
  $$(.97, .48); (.8, 2.21); (.75, 2.86); (.75, .99); (.74, 1.48)$$

- No real reciprocal were obtained for this AR fit.

- The MLE (rounded to 2 digits) is $\hat{\phi} = (0.25, 0.04, -0.17, -0.17, -0.13, -0.17, -0.24, -0.11, -0.05, -0.11)$, and $\hat{\sigma}^2 = 3657.47$.

- The unbiased estimate for $\sigma^2$ is $3753.72$.

- The MLE for each reciprocal root in terms of modulus
and frequency:

\[(.97, .48); (.8, 2.22); (.77, 2.85); (.75, 1.47); (.74, .99)\]

```r
a = ar(eeg, aic=F, order=10, method="mle")
ph = a$ar
v = a$var.pred
round(ph, 2)
ar1  ar2  ar3  ar4  ar5  ar6  ar7  ar8
  0.25  0.03 -0.16 -0.17 -0.13 -0.17 -0.24 -0.11
ar9  ar10
  -0.05 -0.11
round(v, 2)
[1] 3609.46
alpha = 1/polyroot(c(1, -ph))
round(m <- Mod(alpha), 2)
```
[1] 0.97 0.80 0.80 0.97 0.74 0.76
 0.75 0.74 0.75 0.76
round(w<-Arg(alpha),2)
  [1] -0.48 -2.22 2.22 0.48 -1.00
-2.85 1.47 1.00 -1.47 2.85
#order by modulus
m=m[w>0]
w=w[w>0]
rev(m[order(m)])
  [1] 0.97 0.79 0.76 0.75 0.74
rev(w[order(m)])
  [1] 0.48 2.22 2.85 1.47 0.99
We now show the results for a Bayesian Analysis of an AR(10) model and based on 5000 Monte Carlo samples. The graphs include:

- Histograms of posterior samples for $\phi$ coefficients and the variance $\sigma^2$
- Histograms of posterior samples for the $\alpha$’s ordered by modulus.
- Histograms of posterior samples for the $\alpha$’s ordered by periods (or frequencies).
- The $\alpha$’s are shown in terms of the pairs $(r_i, \omega_i)$ or $(r_i, \lambda_i); i = 1, 2, \ldots, 5.$
Posterior histograms for $\phi_1 - \phi_6$
Posterior histograms for $\phi_7 - \phi_{10}$ and $\sigma^2$
Posterior histograms ordered by modulus of 

\((r_i, \omega_i), i = 1, 2, 3\)
Posterior histograms ordered by modulus of 

\((r_i, \omega_i), i = 4, 5\)
Posterior histograms ordered by modulus of
$(r_i, \lambda_i), i = 1, 2, 3$
Posterior histograms ordered by modulus of 
\((r_i, \lambda_i), i = 4, 5\)
Posterior histograms ordered by wavelength of

\((r_i, \omega_i), i = 1, 2, 3\)
Posterior histograms ordered by wavelength of 

\((r_i, \omega_i), \ i = 4, 5\)
Posterior histograms ordered by wavelength of 
$(r_i, \lambda_i), i = 1, 2, 3$
Posterior histograms ordered by wavelength of $(r_i, \omega_i), i = 4, 5$
Graphs based on residuals obtained at the MLE $\hat{\phi}$
AR(10) spectrum computed at the MLE
• Computing the “approximate” test for white noise for the model residuals gives $T = 17138.13$ and a p-value of $0.839543$.

• The p-value is “large” so we don’t have any evidence against the null hypothesis of the residuals following a white noise process.

• Using this Bayesian approach, we make statements about this process following a stationary series or not.

• For our 5000 samples, we can count have many of this samples produce a characteristic reciprocal root with a modulus greater than one.

• The relative frequency of the event “modulus greater than one” gives us an estimate of the posterior
probability of this EEG series comes from “a non-stationary process”.

• For these 5000 samples, the posterior probability is 0.026.

• In a similar way, we can compute the posterior probability of having at least one real root under the AR(10) process (0.0032).
Code for Bayesian AR estimation

eeg=scan("eeg")

# Building F matrix for AR(10)
n <- length(eeg)
x <- eeg[11:n]

f1 <- eeg[10:(n-1)]; f2 <- eeg[9:(n-2)]
f3 <- eeg[8:(n-3)]; f4 <- eeg[7:(n-4)]
f5 <- eeg[6:(n-5)]; f6 <- eeg[5:(n-6)]
f7 <- eeg[4:(n-7)]; f8 <- eeg[3:(n-8)]
f9 <- eeg[2:(n-9)]; f10 <- eeg[1:(n-10)]

#
Fdes <- cbind(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10)
# F^tF mle and R

```
FF.t <- t(Fdes)%*%Fdes
FF.inv <- solve(t(Fdes)%*%Fdes)
phhat <- FF.inv%*%t(Fdes)%*%x
R <- (x - (Fdes)%*%phhat)
s <- (ssq=sum (R*R))/(n-10)
```

# Compute Cholesky decomposition
```
FF.inv <- 0.5*(FF.inv + t(FF.inv))
FF.inv.cd <- t(chol(FF.inv))
```

# ar 10 function that draws from the
# posterior for (phi,sigma^2)
ar10 <- function (m)
{
    phhatmat <- matrix(phhat,10,m)
    v <- rchisq(m,(n-20))
    v <- ssq/v
    # generation of ph
    norbi <- matrix(rnorm(10*m),ncol=m)
    normv <- t(sqrt(v)*t(norbi))
    ph <- (FF.inv.cd%*%normv) + phhatmat
    #result
    ph <- t(ph)
    return(ph,v)
}
phlist=ar10(5000)
phsim = phlist$ph
sigma2 = phlist$v

coeq = cbind(rep(1, 5000), -phsim)
roots = 1/t(apply(coef, 1, polyroot))
mod = Mod(roots)
om = Arg(roots)

# Ordering by modulus
lam = 2*pi/om
lam[om <= 1e-9 | om >= pi - 1e-9] = 0
om[om <= 1e-9 | om >= pi - 1e-9] = 0
mod[om <= 1e-9 | om >= pi - 1e-9] = 0
#
lm=matrix(NA,nrow=dim(lam)[1],ncol=dim(lam)[2])
md=matrix(NA,nrow=dim(mod)[1],ncol=dim(mod)[2])
w=matrix(NA,nrow=dim(om)[1],ncol=dim(mod)[2])
#
for(i in 1:5000){ind=order(mod[i,]);
    w[i,]=om[i,ind];
    lm[i,]=lam[i,ind];
    md[i,]=mod[i,ind]}

# Some residual analysis
ts.plot(as.ts(R))
hist(R,nclass=30,prob=T,density=-1)
qqnorm(R)
qqline(R)