**Estimation – Maximum Likelihood**

Estimate the parameters (ϕ’s, θ’s, μ, and ) of an ARIMA model. A ^ is placed above parameters to denote estimated quantities.

Parameter estimation for ARIMA models:

1. Method of moments
2. Unconditional least squares (ULS)
3. Conditional least squares (CLS)
4. Maximum likelihood estimation (MLE)

Notes:

* Method of moments can provide initial estimates for iterative procedures
* ULS, CLS, and MLE are iterative methods
* ULS and CLS are approximations to MLE
* Which estimation method to use? From Box, Jenkins, and Reinsel’s textbook:

Generally, the conditional and unconditional least squares estimators serve as satisfactory approximations to the maximum likelihood estimator for large sample sizes. However, simulation evidence suggests a preference for the maximum likelihood estimator for small or moderate sample sizes, especially if the moving average operator has a root close to the boundary of the invertibility region.

Likelihood function

If you need a review of maximum likelihood estimation, please see my separate videos and notes from a mathematical statistics course. These notes show mathematical derivations for a few examples and show how to use Sage (a symbolic mathematical software package) to perform these derivations as well.

Consider the following ARMA(p,q) model.

xt - ϕ1xt-1 - … - ϕpxt-p = wt + θ1wt-1 + … + θqwt-q

where wt ~ ind. N(0,) for t = 1, …, n. Comments:

* If we had an ARIMA model instead, let xt = (1-B)dvt for a nonstationary in the mean process vt.
* The normal distribution assumption is needed for these maximum likelihood estimation methods.
* If we do not assume E(xt) = μ = 0, then

(xt-μ) - ϕ1(xt-1-μ) - … - ϕp(xt-p-μ)

 = wt + θ1wt-1 + … + θqwt-q

⇒ xt - ϕ1xt-1 - … - ϕpxt-p = α + wt + θ1wt-1 + … + θqwt-q

with α = μ(1 - ϕ1 - … - ϕp)

This can be rewritten as

wt = xt - ϕ1xt-1 - … - ϕpxt-p

- θ1wt-1 - … - θqwt-q - α (1)

Conditional on past xt values, the likelihood can then be written in the form



where **w**=(w1,…,wn)′, **ϕ**=(ϕ1,…,ϕp)′, and **θ**=(θ1,…,θq)′ are vectors of parameters. Additional details are provided in Shumway and Stoffer’s section on estimation.

The log likelihood function can be found by (**ϕ**,**θ**,μ,|**w**) = log[]. Values of the parameters that maximize this equation are the maximum likelihood estimates (MLEs).

To find parameter estimates, iterative numerical methods must be used to find these estimates. A common method often used is the Newton-Raphson method.

Newton-Raphson method

Let (ϕ1, …, ϕp, θ1, …, θq, , μ)′ be rewritten as a vector of β’s: **β** = (β1,…,βk)′ where k = p + q + 2.

Let  denote the first partial derivative taken with respect to βi. Let  =  be a k1 vector of these first partial derivatives.

Let  be the second partial derivative taken with respect to βi and βj. Let  be a kk matrix of these second partial derivatives and assume it is nonsingular. This matrix is often referred to as a “Hessian” matrix.

Iterative estimates of **β** can be found using the following equation:

After recording the video: The “-“ sign in the equation below was shown mistakenly in the video as a “+” sign.

 for g = 1, 2, …

Remember that  is a k×1 vector. The iteration process stops when these ’s converge to . This is said to happen when  for some small number ε > 0 (there are other definitions of convergence).

Covariance matrix for the estimators

The covariance matrix is  (inverse of the observed Fisher information matrix). The estimated covariance matrix will have the form



Let  be a vector of the maximum likelihood estimators. Then  is approximately distributed as  for large n. See Ferguson’s textbook “A Course in Large Sample Theory” for more information on “standard” maximum likelihood techniques.

Note that hypothesis tests for Ho:βi = βi0 can be done using a Wald statistic:



where  can be found from the ith diagonal element of - and Z has an approximate N(0,1) distribution for a large sample under the null hypothesis.

Some textbooks may examine the asymptotic probability distributions (this is the distribution when n→∞) in more detail than we need for this class. A PhD-level statistics course on asymptotics is required prior to going through these details. For those without this background, you can read through it by simply interpreting

* AN(μ, σ2) to mean the statistic has an approximate normal distribution if the sample size is large
*  to mean ~ (i.e., “distributed as”) if the sample size is large

For example,



means  that is approximately distributed as  for large n.

Example: AR(1) with ϕ1 = 0.7, μ = 0, and  = 1 (fit\_AR1.R, AR1.0.7.txt)

This example fits a model using maximum likelihood estimation to the AR(1) simulated data examined earlier.

> ar1 <- read.table(file = "AR1.0.7.txt", header = TRUE, sep

 = "")

> head(ar1)

 t x

1 1 0.04172680

2 2 0.37190682

3 3 -0.18545185

4 4 -1.38297422

5 5 -2.87593652

6 6 -2.60017605

> x <- ar1$x

> mod.fit <- arima(x = x, order = c(1, 0, 0), method =

 "CSS-ML", include.mean = TRUE)

> #summary(mod.fit) #Not helpful

> mod.fit

Call:

arima(x = x, order = c(1, 0, 0), include.mean = TRUE, method = "CSS-ML")

Coefficients:

 ar1 intercept

 0.6854 -0.4322

s.e. 0.0730 0.3602

sigma^2 estimated as 1.336: log likelihood = -156.68, aic = 319.36

> names(mod.fit)

 [1] "coef" "sigma2" "var.coef" "mask"

 [5] "loglik" "aic" "arma" "residuals"

 [9] "call" "series" "code" "n.cond"

[13] "nobs" "model"

> # Estimated phi1 and mu

> mod.fit$coef

 ar1 intercept

 0.6853698 -0.4322225

> # Estimated sigma^2

> mod.fit$sigma

[1] 1.335638

> # Covariance matrix

> mod.fit$var.coef

 ar1 intercept

ar1 0.005324151 0.001518125

intercept 0.001518125 0.129723806

> #Test statistic for Ho: phi1 = 0 vs. Ha: phi1 <> 0

> z <- mod.fit$coef[1]/sqrt(mod.fit$var.coef[1,1])

> z

 ar1

9.392902

> # p-value

> 2\*(1-pnorm(q = abs(z), mean = 0, sd = 1))

ar1

 0

> #Confidence intervals - uses confint.default()

> confint(mod.fit, level = 0.95)

 2.5 % 97.5 %

ar1 0.5423576 0.8283821

intercept -1.1381464 0.2737015

Notes:

* arima() finds the estimated ARIMA model. Examine the syntax used!
* The estimated model is

(1 – 0.6854B)xt = -0.4322(1 – 0.6854) + wt

⇔ (1 – 0.6854B)xt = -0.1360 + wt

where  = -0.4322 and  = 1.336. Equivalently, this can be written as

xt = -0.1360 + 0.6854xt-1 + wt

Compare these estimates to what was obtained earlier with the methods of moments.

* Note the include.mean = TRUE argument in arima() that allows one to estimate μ. This is the default for ARIMA models with d = 0. R’s estimate for the mean is listed in the output as “intercept”. This may lead you to think that  is being estimated instead. Be careful!
* There is another way to estimate μ through the xreg option. This option will be more important later, but it is instructive now to see how it can be used here,

> arima(x = x, order = c(1, 0, 0), method = "CSS-ML",

 include.mean = FALSE, xreg = rep(x = 1, times =

 length(x)))

Call:

arima(x = x, order = c(1, 0, 0), xreg = rep(x = 1, times = length(x)), include.mean = FALSE, method = "CSS-ML")

Coefficients:

 ar1 rep(x = 1, times = length(x))

 0.6854 -0.4322

s.e. 0.0730 0.3602

sigma^2 estimated as 1.336: log likelihood = -156.68, aic = 319.36

The rep(x = 1, times = length(x)) code creates a vector of 1’s with a length of 100. This tells R that there is a time series variable of all 1’s being used to predict xt.

* Notice that 
* Examine the components that can be extracted from the mod.fit list.
* To test if Ho:ϕ1 = 0 vs. Ha:ϕ1 ≠ 0, use the test statistic



which results in a p-value of ≈ 0. Thus, ϕ1 ≠ 0 as would be expected!

* Confidence intervals are found using confint().
* The iterative parameter estimation method used is maximum likelihood estimation. To find initial values of the parameter estimates for this iterative method, conditional sums of squares estimation is used. The method = "CSS-ML" argument is specified in arima() to implement the parameter estimation method. Note that this is actually the default so excluding the method option will result in the same estimation method.
* The maximum likelihood estimation part is carried out by theoptim()function in R (arima() calls this function). This is a very general function that can be used to find values which maximize a function. Finer control of the optimization can then be done through specifying the optim.control = list()option in arima(). For example, information about the iterative history can be specified by using trace = 1 and controlling the number of iterations can be done through maxitoption.

The methods() function is used to see the method functions available for objects of class Arima. For example, there is a vcov.Arima() method function.

> class(mod.fit)

[1] "Arima"

> methods(class = "Arima")

[1] coef logLik predict print tsdiag vcov

> methods(generic.function = vcov)

[1] vcov.aov\* vcov.Arima\* vcov.glm\*

[4] vcov.lm\* vcov.mlm\* vcov.nls\*

[7] vcov.summary.glm\* vcov.summary.lm\*

see '?methods' for accessing help and source code

> # Covariance matrix (2nd way)

> vcov(mod.fit)

 ar1 intercept

ar1 0.005324151 0.001518125

intercept 0.001518125 0.129723806

> getAnywhere(vcov.Arima)

A single object matching ‘vcov.Arima’ was found

It was found in the following places

 registered S3 method for vcov from namespace stats

 namespace:stats

with value

function (object, ...)

object$var.coef

<bytecode: 0x000000001b7ede68>

<environment: namespace:stats>

Example: ARIMA(1,1,1) with ϕ1 = 0.7, θ1 = 0.4,  = 9, n = 200 (arima111\_sim.R, arima111.csv)



Below is the R code used to estimate an ARIMA(1,1,1) model for the data.

> arima111 <- read.csv(file = "arima111.csv")

> x <- arima111$x

> mod.fit <- arima(x = x, order = c(1, 1, 1))

> mod.fit

Call:

arima(x = x, order = c(1, 1, 1))

Coefficients:

 ar1 ma1

 0.6720 0.4681

s.e. 0.0637 0.0904

sigma^2 estimated as 9.558: log likelihood = -507.68, aic = 1021.36

> # Covariance matrix

> mod.fit$var.coef

 ar1 ma1

ar1 0.004060990 -0.003341906

ma1 -0.003341906 0.008175261

> confint(object = mod.fit, level = 0.95)

 2.5 % 97.5 %

ar1 0.5470940 0.7968949

ma1 0.2909019 0.6453306

> # Test statistic for Ho: phi1 = 0 vs. Ha: phi1 <> 0

> z <- mod.fit$coef[1]/sqrt(mod.fit$var.coef[1,1])

> z

 ar1

10.54508

> 2\*(1-pnorm(q = abs(z), mean = 0, sd = 1))

ar1

 0

> # Test statistic for Ho: theta1 = 0 vs. Ha: theta1 <> 0

> z <- mod.fit$coef[2]/sqrt(mod.fit$var.coef[2,2])

> z

 ma1

5.177294

> 2\*(1-pnorm(q = abs(z), mean = 0, sd = 1))

 ma1

2.251268e-07

Notes:

* Notice the syntax used to fit the model using arima().
* No estimate of μ was found. The reason is that the time series data is differenced. Remember that
E(xt - xt-1) = μ - μ = 0. There are times when estimating a “constant” with the rest of the model is still of interest for an ARIMA model. We will discuss these later. The arima()function will default to include.mean = FALSE when d > 0 so you will need to use the xreg option in arima() to include the constant term.
* The estimated model is

(1 − B)(1 − B)1xt= (1 + B)wt

⇔ (1 − 0.6720B)(1 − B)xt = (1 + 0.4681B)wt

⇔ xt = (1 + 0.6720)xt-1 – 0.6720xt-2 + 0.4681wt-1 + wt

with wt ~ ind. N(0,9.56)

* The estimated covariance matrix for  = (, )′ is:



* To test if Ho:ϕ1 = 0 vs. Ha:ϕ1 ≠ 0, use the test statistic



which results in a p-value of ≈ 0. Thus, ϕ1 ≠ 0 as would be expected!

* To test if Ho:θ1 = 0 vs. Ha:θ1 ≠ 0, use the test statistic



which results in a p-value of ≈ 0. Thus, θ1 ≠ 0 as would be expected!

Final notes:

* The iterative methods used to find the parameter estimates may NOT converge. If this happens, try a larger number of iterations. If this does not work, one should NOT use any estimates produced for the model.
* An alternative to using a normal distribution approximation for inference is the bootstrap. R has a few functions in the boot package that allow one to perform these methods.