**AR Models**

Suppose we have a time series xt for t = 1, …, n. Could we use the regression model of

xt = β0 + β1t + εt,

where εt ~ independent N(0,σ2) for it? Yes, but stated confidence levels and type I error rates will likely be incorrect. Thus, inferences can be incorrect. The reason is the likely dependence in the time series.

While one may be able to find a set of explanatory variables that help to de-trend a response variable series so it appears that white noise is leftover (i.e., it looks like the error terms are independent), this is often not possible.

Instead, autoregressive integrated moving average models (ARIMA) are used for time series data. These models were first developed by Box and Jenkins (1970). We have already touched on all parts of this type of model:

* AR: Autoregressive
* MA: Moving average
* I: Integrated (closely related to differencing)

Often, people will refer to ARIMA models as an ARIMA process as well. This refers to how xt comes about through a linear process. Both “model” and “process” will be used interchangeably.

Autoregressive models – AR(p)

An autoregressive model uses past observations of xt to predict future observations. Specifically, the present value of xt is explained by a linear function of p past values of xt-1, …, xt-p.

Example: AR(1) from earlier

xt = 0.7xt-1 + wt where wt ~ independent N(0,1) for t = 1, …, n.

Therefore,

x2 = 0.7x1 + w2

x3 = 0.7x2 + w3

Future values may be “forecasted” by past values using

xn+1 = 0.7xn

More on this in later in the section.

An autoregressive model of order p, denoted as AR(p), is

xt = ϕ1xt-1 + ϕ2xt-2 + … + ϕpxt-p + wt

where

ϕ1, ϕ2, …, ϕp are parameters and

wt ~ ind. (0,) for t = 1,…, n (i.e., white noise)

Notes:

* Some textbooks use “*φ*” as an autoregressive parameter. Both ϕ and *φ* are the “phi” Greek letter.
* To find parameter estimates later in this section, we will assume wt ~ independent N(0,).
* Without loss of generality (WLOG), the mean of xt, μ, will be assumed to be 0 when we write out a general model. HOWEVER, μ ≠ 0 in most applications! The WLOG part is here because one can simply replace xt with xt – μ. The end result is an autoregressive model of

xt - μ = ϕ1(xt-1-μ) + ϕ2(xt-2-μ) + … + ϕp(xt-p-μ) + wt

⇔ xt = μ(1-ϕ1-ϕ2-…-ϕp) + ϕ1xt-1 + ϕ2xt-2+ … + ϕpxt-p + wt

⇔ xt = α + ϕ1xt-1 + ϕ2xt-2+ … + ϕpxt-p + wt

where α = μ(1-ϕ1-ϕ2-…-ϕp). The α does not affect the dependence structure among the xt. This is why the common convention is to exclude the parameter when introducing these models.

**When we estimate the model, we will almost always include an estimate of α.**

* The AR(p) model written in vector form is

xt = **ϕ**′**x**t-1 + wt

where

**ϕ** = (ϕ1, ϕ2, …, ϕp)′

**x**t-1 = (xt-1, xt-2, …, xt-p)′

* The AR(p) model written in backshift notation is

(1-ϕ1B-ϕ2B2-…-ϕpBp)xt = wt

and

ϕ(B)xt = wt

where ϕ(B) = (1-ϕ1B-ϕ2B2-…-ϕpBp) is called the autoregressive operator. We will be using this notation throughout the course.

* The model can be re-expressed as a linear combination of wt’s by “iterating backwards”. For example, an AR(1) model can be represented as:

xt = ϕxt-1 + wt (NOTE: the 1 on ϕ1 was dropped)

= ϕ(ϕxt-2 + wt-1) + wt = ϕ2xt-2 + ϕwt-1 + wt

= ϕ2(ϕxt-3 + wt-2) + ϕwt-1 + wt = ϕ3xt-3+ϕ2wt-2+ϕwt-1+wt

= 

provided that |ϕ|<1 and variance of xt is bounded.

To see this, note that the model can be rewritten as

(1-ϕB)xt = wt

⇒ 

⇔ xt = (1+ϕB+ϕ2B2+…)wt = 

using the sum of an infinite series. Remember that . Writing the model as a linear combination of the wt’s is going to be VERY useful for future work!

* The mean and autocovariance function for an AR(1) stationary model can be found easily by using the above representation.

E(xt) = 

γ(h) = Cov(xt, xt+h)

= E(xtxt+h) - E(xt)E(xt+h)

= E(xtxt+h) - 0 = E(xtxt+h) assuming μ =0 .

Then

γ(h) = 

= E[(wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…)
 (wt+h+ϕwt+h-1+ϕ2wt+h-2+ϕ3wt+h-3+…)]

If h = 0,

γ(0) = E[(wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…)2]

= 

=  =

I used these general results that are taught in a mathematical statistics course:

* + - E() = Var(Y1) + E(Y1)2
		- Var(a1Y1 + a2Y2) =  + 

for independent random variables Y1 and Y2 and constants a1 and a2

If h=1,

γ(1) = E[(wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…)

 (wt+1+ϕwt+ϕ2wt-1+ϕ3wt-2+…)]

= E[wt+1(wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…) +

ϕ(wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…)2]

= E[wt+1] × E[wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…] + ϕ

= 0 × [0 + ϕ0 + ϕ20 + …] +  = 

I used wt+1 being independent of all of the w’s in E[wt+ϕwt-1+ϕ2wt-2+ϕ3wt-3+…] in the above result.

In general for h ≥ 0, .

Also, the ACF is ρ(h) = γ(h)/γ(0) = ϕh.

One can also go back to in the notes and use the results of a linear process there. Below is part of this page restated,

In general, a linear process can be defined as

 with  and

 wt ~ ind. N(0,).

It can be shown that  for h ≥ 0.

In our case, we have ψ0 = 1, ψ1 = ϕ, ψ2 = ϕ2, ψ3 = ϕ3, … . Therefore, . This results in





Question: What if μ ≠ 0? How would this change E(xt) and γ(h)?

Example: AR(1) with ϕ = 0.7 and ϕ =- 0.7 (ar1\_sim.R)

The purpose of this example is to show what observed values from an AR(1) process look like for t = 1, …, 100 using  = 100. Pay close attention to the differences between ϕ = 0.7 and ϕ = -0.7. Questions to think about are:

* Why are some plots more or less “choppy” (“jagged”)?
* What would happen to the plots if |ϕ| was closer to 0 or 1?

Note that ρ(h) = γ(h)/γ(0) = ϕh, which leads to autocorrelations of

|  |  |  |
| --- | --- | --- |
| h | ϕ = 0.7 | ϕ = -0.7 |
| 1 | 0.7 | -0.7 |
| 2 | 0.49 | 0.49 |
| 3 | 0.34 | -0.43 |
|  |  |  |

For the plot below, the model is xt = 0.7xt-1 + wt and
(1 – 0.7B)xt = wt.



For the plot below, the model is xt = -0.7xt-1 + wt and
(1 + 0.7B)xt = wt.



R Code (use ar = c(-0.7) for the second plot):

set.seed(7181)

x <- arima.sim(model = list(ar = c(0.7)), n = 100, rand.gen

 = rnorm, sd = 10)

plot(x = x, ylab = expression(x[t]), xlab = "t", type =

 "l", col = c("red"), main =

 expression(paste(x[t] == 0.7\*x[t-1] + w[t], " where ",

 w[t], "~ ind. N(0,100)")), panel.first=grid(col =

 "gray", lty = "dotted"))

points(x = x, pch = 20, col = "blue")

I could just use ar = 0.7, but included c() because it will be needed when we have p > 1.

Be very careful in specifying these models in R!

* The model can be written as xt = ϕxt-1 + wt and (1 – ϕB)xt = wt. Notice that the first plot uses ϕ = +0.7 and the second plot uses ϕ = -0.7. The autoregressive operator can confuse matters in deciding whether the numerical value of ϕ is positive or negative.
* Software packages and textbooks are **not** consistent in their definitions of the positive and negative values. For example, some books may denote the AR operator to be (1 + ϕB) instead of (1 – ϕB).

To simulate data from a higher order AR(p), use the following type of syntax in the model option:

list(ar = c(0.7, -0.4))

This specifies ϕ1 = 0.7 and ϕ2 = -0.4.

R has a nice function that allows you to see the ACF for an AR (or ARMA) process better. The ARMAacf() allows one to plot the actual ACF for an AR (and ARMA) model.

Example: AR(1) with ϕ1 = 0.7 and ϕ1 = -0.7 (ar1\_sim.R)

When ϕ1 = 0.7, here are the results of the ARMAacf()function. Notice the use of phi1 instead of just phi in the main argument of the plot() function. Using phi produces φ instead of ϕ.

> round(ARMAacf(ar = c(0.7), lag.max = 20), 4)

 0 1 2 3 4 5 6 7 8 9 10

1.0000 0.7000 0.4900 0.3430 0.2401 0.1681 0.1176 0.0824 0.0576 0.0404 0.0282

 11 12 13 14 15 16 17 18 19 20

0.0198 0.0138 0.0097 0.0068 0.0047 0.0033 0.0023 0.0016 0.0011 0.0008

> plot(y = ARMAacf(ar = c(0.7), lag.max = 20), x = 0:20,

 type = "h", ylim = c(-1,1), xlab = "h", ylab =

 expression(rho(h)), main = expression(paste("ACF for

 AR(1) with ", phi1[1] == 0.7)))

> abline(h = 0)



Using ϕ1 = -0.7 (code omitted),



The type = "h" argument in the plot()tells R to plot vertical lines from a y-axis value of 0 value up to the autocorrelations.

Be careful with the plotting of the ACF, suppose the following code was used instead:

> plot(x = ARMAacf(ar = c(0.7), lag.max = 20), type =

 "h", ylim = c(-1,1), xlab = "h", ylab =

 expression(rho(h)), main = expression(paste("ACF

 for AR(1) with ", phi1[1] == 0.7)))

> abline(h = 0)



In this case, there is no y = part and the plot above is drawn incorrectly with ρ(1) = 1, ρ(2) = 0.7 (everything is shifted one lag). The reason why this happens is that ARMAacf() gives output that starts at h = 0, but R does not have a way to adjust the x-axis scale to adjust for it.

Example: AR(1) with ϕ1 = 1 (no program)

For the AR(1) process to be stationary, the following condition must be satisfied: |ϕ1| < 1. To see what happens when it is not satisfied, consider a sample generated from an AR(1) process with ϕ1 = 1. Below is a plot.



The process no longer appears to be stationary in the mean.

Below is another set of observations generated from an AR(1) process.



This series also appears come from a non-stationary process.

Other examples can be constructed for |ϕ1|≥1.

Causal process

Note that the AR(1) process with |ϕ1| ≥ 1 can be rewritten as a stationary “future-dependent” series. See Shumway and Stoffer’s “Explosive AR Models and Causality” example. Because we will not know the “future” values in actual applications, this representation will not help us!

When a process is stationary AND does not depend on the future, the process is “causal”. These are the type of processes that we will be interested in.

A more formal definition of a causal process will be given later.

Writing higher-order casual models as 

Higher-order models will have constraints on ϕ1, ϕ2, … , ϕp to insure a model is causal. We will examine these constraints later. For now, assume we have a causal model.

In a previous example, the AR(1) process was rewritten as

(1-ϕB)xt = wt

⇒ 

⇔ xt = (1+ϕB+ϕ2B2+…)wt = 

When the process is a higher order AR(p), we can do the same thing! In general, consider the AR(p) process below:

(1-ϕ1B-ϕ2B2-…-ϕpBp)xt = wt

⇔ ϕ(B)xt = wt

⇔ xt = [1/ϕ(B)]wt = ψ(B)wt

where ψ(B) = 1+Bψ1+B2ψ2+…, and 1/ϕ(B) = ψ(B). Be careful with this definition of ψ(B) because there are +, not –, values separating the terms. Also, note that ψ(B) is an infinite series.

Then 1 = ϕ(B) × ψ(B)

= (1-ϕ1B-ϕ2B2-…-ϕpBp)(1+ψ1B+ψ2B2+…)

To find the values of the ψj’s in terms of the ϕi’s, we can equate the coefficients of the B’s on both sides of the equality (note the left side has 1×B0 + 0×B1 + 0×B2 +…)

1 = (1-ϕ1B-ϕ2B2-…-ϕpBp)(1+ψ1B+ψ2B2+…)

⇔1 = 1 - ϕ1B - ϕ2B2 - ϕ3B3 - …

+ ψ1B - ψ1ϕ1B2 - ψ1ϕ2B3 - ψ1ϕ3B4 - …

+ ψ2B2 - ψ2ϕ1B3 - ψ2ϕ2B4 - ψ2ϕ3B5 - …

+ ψ3B3 - ψ3ϕ1B4 - ψ3ϕ2B5 - ψ3ϕ3B6 - …

⇔1 = 1 + (ψ1-ϕ1)B+ (ψ2 - ψ1ϕ1-ϕ2)B2

 + (ψ3 - ψ2ϕ1 - ψ1ϕ2 - ϕ3)B3 + …

Then equating coefficients:

B0: 1=1

B1: ψ1-ϕ1 = 0 ⇒ ϕ1=ψ1

B2: ψ2 - ψ1ϕ1-ϕ2 = 0 ⇒ 

B3: ψ3 - ψ2ϕ1 - ψ1ϕ2 - ϕ3 = 0 ⇒



In general, .

Example: AR(2) process

For an AR(2) process, this means that



Then writing the ψ’s in terms of the ϕ’s:





Equating powers of B implies that

B1: (ψ1-ϕ1)=0 ⇒ ψ1 = ϕ1

B2: (ψ2-ϕ1ψ1-ϕ2)=0 ⇒ ψ2 = ϕ1ψ1+ϕ2 = +ϕ2

B3: (ψ3-ϕ1ψ2-ϕ2ψ1)=0 ⇒ ψ3 = ϕ1ψ2+ϕ2ψ1 = +ϕ2ϕ1

In general, ψj = ψj-1ϕ1 + ψj-2ϕ2 for j ≥ 2 where ψ0 = 1.

For the special case of ϕ2 = 0 (i.e., AR(1)), ψj =  and

 for |ϕ1|<1

R has a nice function that allows you to make this conversion a little easier. The ARMAtoMA() converts an ARMA model to a “MA model”. For now, one can think of this like how we converted the AR(1) or AR(2) model to a model with an infinite number of wt random variables.

Example: AR(1) with ϕ1 = 0.7 (ar1\_sim.R)

Using ARMAtoMA() produces,

> ARMAtoMA(ar = c(0.7), lag.max = 20)

[1] 0.7000 0.4900 0.3430 0.2401 0.1681 0.1176 0.0824 0.0576

[9] 0.0404 0.0282 0.0198 0.0138 0.0097 0.0068 0.0047 0.0033

[17] 0.0023 0.0016 0.0011 0.0008

These are the ψ1, ψ2, …, ψ20 values when ϕ1 = 0.7.

One could also use this function for higher order AR models. For example,

> #Example for AR(2)

> round(ARMAtoMA(ar = c(0.7, -0.4), lag.max = 20), 4)

[1] 0.7000 0.0900 -0.2170 -0.1879 -0.0447 0.0438 0.0486

[8] 0.0165 -0.0079 -0.0121 -0.0053 0.0011 0.0029 0.0016

[15] -0.0001 -0.0007 -0.0005 0.0000 0.0001 0.0001

Remember that

ψ1 = ϕ1 = 0.7

ψ2 = ϕ1ψ1+ϕ2 = +ϕ2 = 0.72 – 0.4 = 0.09.