**Forecasting – Point Estimates**

Predict (forecast) future values of a time series, xn+1, xn+2, … based on x1, …, xn.

For m time points into the future, the “minimum mean square error predictor” of xn+m is = E(xn+m|xn,xn-1,…, x1). What does this mean?

E(xn+m|xn,xn-1,…, x1) is a conditional expectation. It denotes the expected value of x at m time points into the future, conditional on the time series observed.

E(xn+m|xn,xn-1,…, x1) is the value of g(x1,..,xn) that minimizes E[xn+m – g(x1,..,xn)]2

Instead of using this as a predictor, we will use an approximation  = E(xn+m|xn,xn-1,…,x1,…). What is the difference?

 is based on an infinite past and  is based on a finite past (which is what we actually have).

For notational convenience, I will denote xn, xn-1, …, x1,… as In. This symbolizes all INFORMATION up to time point n from an INFINITE past.

Example: AR(1)

The model is xt = ϕ1xt-1 + wt where wt ~ independent (0,). Taking into account the observed data of x1, …, xn, this also essentially means w1, …, wn are observed.

Suppose we have observations up to time n. We want to forecast future values for time n + m (m > 0). Thus, we want xn+m = ϕ1xn+m-1 + wn+m.

Let m = 1. Then the “forecasted value at time n + 1 given information up to time n” is

 = E(xn+1|In)

= E(ϕ1xn + wn+1 |In)

= E(ϕ1xn |In) + E(wn+1 |In)

= ϕ1E(xn |In) + 0 because wn+1 is unobserved and   
 the wn+1 ~ ind. (0, )

= ϕ1xn  because the expectation is found CONDITIONAL on knowing xn, xn-1,…, x1,…; i.e., we know what xn is!

Let m = 2. Then

 = E(xn+2|In)

= E(ϕ1xn+1 + wn+2 |In)

= E(ϕ1xn+1 |In) + E(wn+2 |In)

= ϕ1E(xn+1 |In) + 0 because wn+2 is unobserved and

the wn+2 ~ (0, )

= ϕ1 because this was found for m = 1.

 can be further written as ϕ1 = 

In summary,

|  |  |
| --- | --- |
| m |  |
| 1 | ϕ1xn |
| 2 | ϕ1 |
| 3 | ϕ1 |
| 4 | ϕ1 |
|  |  |

Because the parameters are generally not known, they are replaced with their estimates. Thus,  = ,  = , … .

It would be more notationally correct to refer to this as  = ,  = , …, but I chose to follow the notational convention of most textbooks on time series.

Question: What happens to the forecast as m → ∞?

The forecast goes to 0 because E(xt) = 0.

What if μ ≠ 0. Then xt = μ(1-ϕ1) + ϕ1xt-1 + wt where μ(1-ϕ1) = α is a constant term. Then

|  |  |
| --- | --- |
| m |  |
| 1 | μ(1-ϕ1) + ϕ1xn |
| 2 | μ(1-ϕ1) + ϕ1 |
| 3 | μ(1-ϕ1) + ϕ1 |
|  |  |

Example: MA(1)

Suppose we have observations up to time n and we want to forecast future values for time n + m (m > 0).

xt = θ1wt-1 + wt where wt ~ independent (0,)

Note that at time n + m, xn+m = θ1wn+m-1 + wn+m.

Let m = 1. Then

 = E(xn+1|In)

= E(θ1wn+wn+1|In)

= θ1E(wn|In) + E(wn+1|In)

= θ1wn + 0 because wn+1 is unobserved with

wn+1 ~ (0, ); also xn = ϕ1xn-1 + wn has been observed

= θ1wn

Let m = 2. Then

 = E(xn+2|In)

= E(θ1wn+1+wn+2|In)

= θ1E(wn+1|In) + E(wn+2|In)

= θ1×0 + 0 because wn+1 and wn+2 are unobserved

N(0, ) random variables

= 0

In summary,

|  |  |
| --- | --- |
| m |  |
| 1 | θ1wn |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
|  |  |

Notice how quickly the forecasted value becomes the mean of the series, 0. Of course, you can also have other MA(q) models with a non-zero mean too.

Because the parameters are generally not known, they are replaced with their estimates. Thus,  = .

What is ? The answer comes from the residuals!

How are residuals found? These are symbolically denoted as  for t = 1, …, n.

Now, wt = xt - θ1wt-1. Let w0 = 0 (remember mean is 0 for white noise). Then

w1 = x1 - θ1w0 = x1 ⇒  = x1

w2 = x2 - θ1w1 ⇒ 

wn = xn - θ1wn-1 ⇒ 

More complicated models follow the same process. See Shumway and Stoffer’s textbook for a ARMA(1,1) example. Alternate methods also include backcasting so that one does not necessarily start xt and wt at fixed constant values (like 0) when t < 1.

Example: ARIMA(1,1,1)

Suppose we have observations up to time n and we want to forecast future values for time n + m (m > 0).

(1-ϕ1B)(1-B)xt = (1+θ1B)wt where wt ~ independent (0,). This can be rewritten as xt = (1+ϕ1)xt-1 - ϕ1xt-2 + θ1wt-1 + wt. At time n + m, we have

xn+m = (1+ϕ1)xn+m-1 - ϕ1xn+m-2 + θ1wn+m-1 + wn+m

Let m = 1. Then

 = E(xn+1|In)

= E[(1+ϕ1)xn - ϕ1xn-1 + θ1wn + wn+1|In]

= (1+ϕ1)E(xn|In) - ϕ1E(xn-1|In) + θ1E(wn|In)

+ E(wn+1|In)

= (1+ϕ1)xn - ϕ1xn-1 + θ1wn + 0

= (1+ϕ1)xn - ϕ1xn-1 + θ1wn

Let m = 2. Then

 = E(xn+2|In)

= E[(1+ϕ1)xn+1 - ϕ1xn + θ1wn+1 + wn+2|In]

= (1+ϕ1)E(xn+1|In) - ϕ1E(xn|In) + θ1E(wn+1|In)

+ E(wn+2|In)

= (1+ϕ1) - ϕ1xn + θ10 + 0

= (1+ϕ1) - ϕ1xn

In summary,

|  |  |
| --- | --- |
| m |  |
| 1 | (1+ϕ1)xn - ϕ1xn-1 + θ1wn |
| 2 | (1+ϕ1) - ϕ1xn |
| 3 | (1+ϕ1) - ϕ1 |
| 4 | (1+ϕ1) - ϕ1 |
|  |  |

Because the parameters are generally not known, they are replaced with their estimates. Also,  replaces wn.

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

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

= "")

> x <- ar1$x

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

> 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

> #Covariance matrix

> mod.fit$var.coef

ar1 intercept

ar1 0.005324151 0.001518125

intercept 0.001518125 0.129723806

> #######################################################

> # Forecasting

> #Notice class of mod.fit is "Arima". Therefore,

# generic functions, like predict, will actually

> # call predict.Arima().

> class(mod.fit)

[1] "Arima"

> #Forecasts 5 time periods into the future

> fore.mod <- predict(object = mod.fit, n.ahead = 5, se.fit

= TRUE)

> fore.mod

$pred

Time Series:

Start = 101

End = 105

Frequency = 1

[1] 1.26014875 0.72767770 0.36273810 0.11261952

[5] -0.05880421

$se

Time Series:

Start = 101

End = 105

Frequency = 1

[1] 1.155698 1.401082 1.502576 1.547956 1.568820

> #x\_100

> x[100]

[1] 2.037059

Estimated model:

(1 – 0.6854B)xt = -0.1360 + wt where  =1.336.

Equivalently,

xt = -0.1360 + 0.6854xt-1 + wt

Forecasts:

| m |  |
| --- | --- |
| 1 | +x100 = -0.1360+0.6854×2.0371 = 1.2602 |
| 2 | + = -0.1360+0.6854×1.2602 = 0.7277 |
| 3 | + = -0.1360+0.6854×0.7277 = 0.3628 |

Notice the syntax used in the predict() function! Calculation of the standard errors and confidence intervals for xn+m will be discussed later.

Below are plots of the observed and the forecasts.

After recording the video: Replace the above sentence with “Below are the residuals.”

> #Residuals

> names(mod.fit)

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

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

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

[13] "nobs" "model"

> mod.fit$residuals

Time Series:

Start = 1

End = 100

Frequency = 1

[1] 0.34512757 0.47929876 -0.30435533 -1.11988088

[5] -1.79209750 -0.49310572 0.81405525 0.42879914

EDITED

[97] -0.10984115 0.69886850 -0.26732184 2.62425181

> # Last residual - as.numeric() removes a leftover label

> as.numeric(x[100] - mod.fit$coef[2]\*(1-mod.fit$coef[1]) –

mod.fit$coef[1]\*x[99])

[1] 2.624252

We can add to the plot of the time series the forecasts t = n, …, m. For visual display, sometimes it is interesting to add the corresponding predicted values for t = 1, …, n. What are these predicted values? A simple computational way to find these in R is use a result from a regression course:

residual = observed – predicted

which leads to

predicted = observed – residual

Similar to there being multiple ways to find residuals, there are multiple ways to find predicted values. Below is how I used predicted = observed – residual and created the corresponding plot.

> #Predicted values for t = 1, ..., 100

> x - mod.fit$residuals

Time Series:

Start = 1

End = 100

Frequency = 1

[1] -0.30340077 -0.10739194 0.11890348 -0.26309333

[5] -1.08383902 -2.10707034 -1.91807243 -0.89265030

EDITED

[97] -1.25434524 -1.07096242 -0.39101218 -0.58719250

> #Add the forecasts into the first plot with C.I.s

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

"o", col = "red", lwd = 1, pch = 20, main =

expression(paste("Data simulated from AR(1): ", x[t]

== 0.7\*x[t-1] + w[t], " where ", w[t], "~N(0,1)")) ,

panel.first = grid(col = "gray", lty = "dotted"), xlim

= c(1, 105))

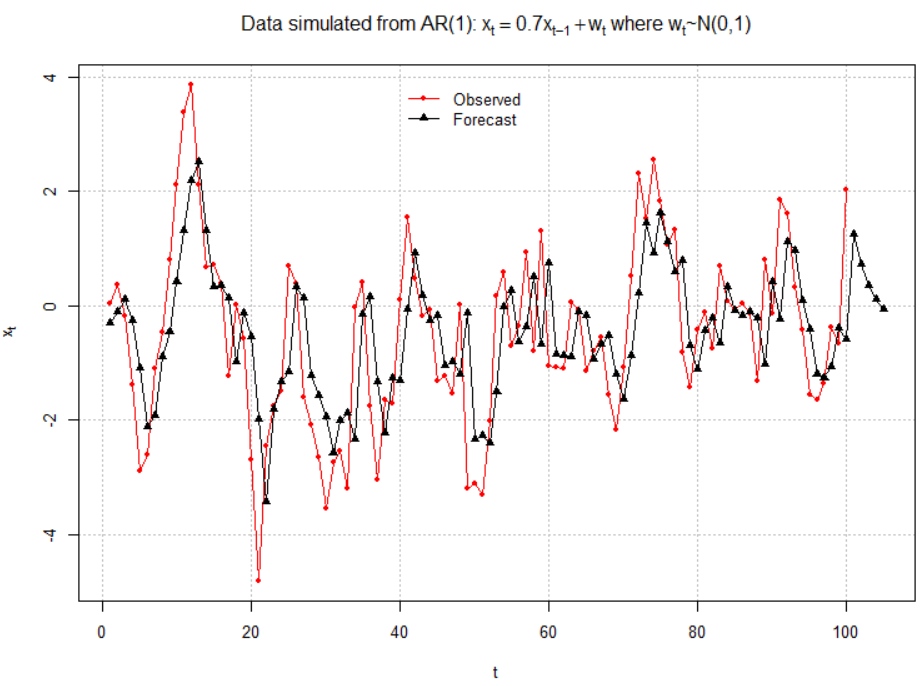
> lines(x = c(x - mod.fit$residuals, fore.mod$pred), lwd

= 1, col = "black", type = "o", pch = 17)

> legend(locator(1), legend = c("Observed", "Forecast"),

lty = c("solid", "solid"), col = c("red", "black"), pch

= c(20, 17), bty = "n")



The x-axis limits were changed in the plot()function to allow for the forecasts for t = 101, …, 105 to be shown. Notice how I put these predicted values together into one vector for the first lines()function call.

Below is a zoomed in version of the plot.

> #Zoom in

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

"o", col = "red", lwd = 1, pch = 20, main =

expression(paste("Data simulated from AR(1): ", x[t]

== 0.7\*x[t-1] + w[t], " where ", w[t], "~N(0,1)")) ,

panel.first = grid(col = "gray", lty = "dotted"), xlim =

c(96, 105))

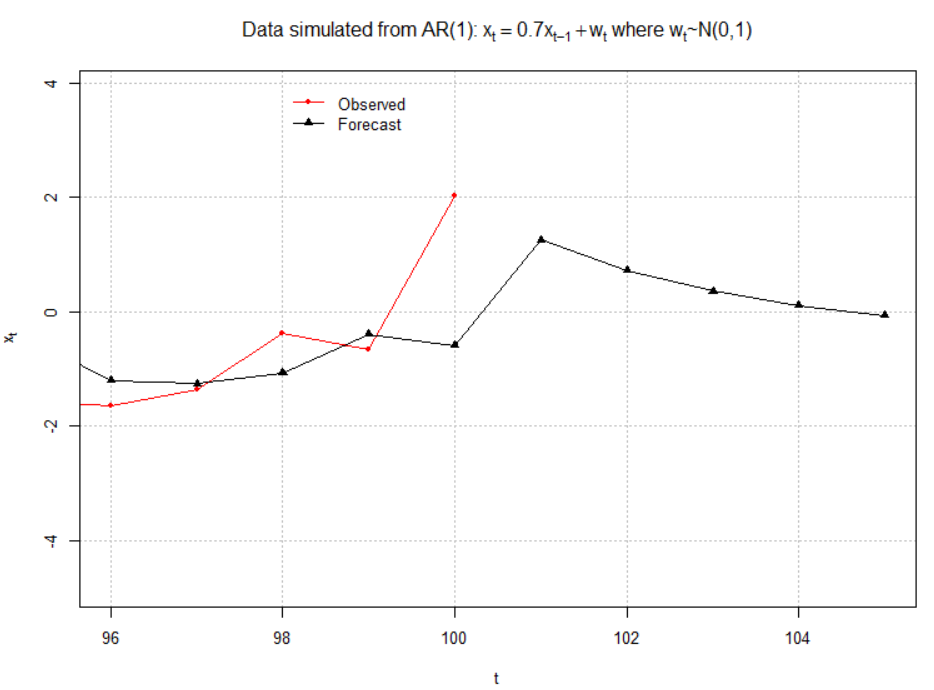
> lines(x = c(x - mod.fit$residuals, fore.mod$pred), lwd

= 1, col = "black", type = "o", pch = 17)

> legend(locator(1), legend = c("Observed", "Forecast",

lty = c("solid", "solid"), col = c("red", "black"), pch =

c(20, 17), bty = "n")



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

> 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

The estimated model is

(1 − 0.6720B)(1 − B)xt = (1+0.4681B)wt with  = 9.56

Equivalently,

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

Forecasts for t = 201, …, 205:

> #Forecasts 5 time periods into the future

> fore.mod <- predict(object = mod.fit, n.ahead = 5, se.fit

= TRUE)

> fore.mod

$pred

Time Series:

Start = 201

End = 205

Frequency = 1

[1] -486.3614 -484.9361 -483.9784 -483.3348 -482.9023

$se

Time Series:

Start = 201

End = 205

Frequency = 1

[1] 3.091673 7.303206 11.578890 15.682551 19.534208

> x[199:200]

[1] -488.2191 -488.4823

> mod.fit$residuals[199:200]

[1] -4.954901 4.908614

We will discuss later how the standard errors and confidence intervals.

With the help of the above output, by-hand calculations of the forecasts are shown below. Note that  was found from the R output.

| m |  |
| --- | --- |
| 1 | (1+)x200 - x199 +   = (1+0.6720)(-488.4823) – (0.6720)× (-488.2191) + 0.4681×4.9086 = -486.36 |
| 2 | (1+) - x200  = (1+0.6720)(-486.36) – (0.6720)× (-488.4823) = -484.93 |
| 3 | (1+) -  = (1+0.6720)(-484.9216) - 0.6720× (-486.3584) = -483.96 |

Below are plots of the forecasts

> #Forecasts with C.I.s

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

"o", col = "red", lwd = 1, pch = 20, main =

expression(paste("ARIMA model: ", (1 - 0.7\*B)\*(1-

B)\*x[t] == (1 + 0.4\*B)\*w[t])), panel.first = grid(col =

"gray", lty = "dotted"), xlim = c(1, 205))

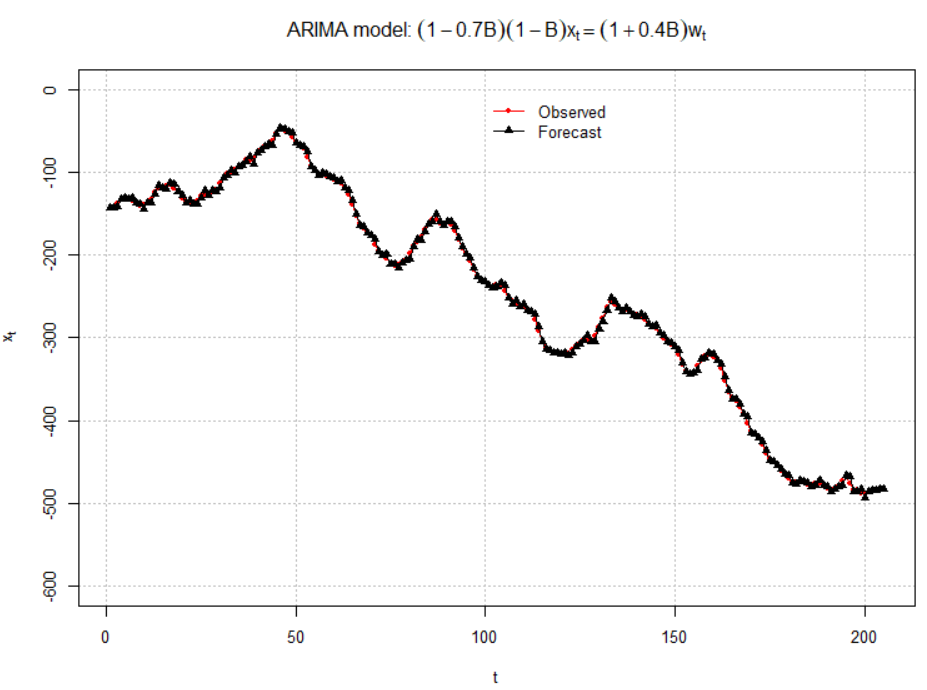
> lines(x = c(x - mod.fit$residuals, fore.mod$pred), lwd

= 1, col = "black", type = "o", pch = 17)

> legend(locator(1), legend = c("Observed", "Forecast"),

lty = c("solid", "solid"), col = c("red", "black"), pch

= c(20, 17), bty = "n")



It is hard to see the observed and forecasted values in the above plot so I zoomed in to create the plot below.

> #Zoom in

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

"o", col = "red", lwd = 1, pch = 20, main =

expression(paste("ARIMA model: ", (1 - 0.7\*B)\*(1-

B)\*x[t] == (1 + 0.4\*B)\*w[t])), panel.first =

grid(col = "gray", lty = "dotted"), xlim = c(196,

205), ylim = c(-540, -440))

> lines(x = c(x - mod.fit$residuals, fore.mod$pred), lwd

= 1, col = "black", type = "o", pch = 17)

> legend(locator(1), legend = c("Observed", "Forecast")

lty = c("solid", "solid"), col = c("red", "black"), pch =

c(20, 17), bty = "n")

