Time Series Forecasting

Most of the discussion so far as focussed on either (a) identifying the time series process or (b) understanding the cyclical variation. But probably the most popular use for time series is to predict the future. Obviously, this is not that easy to do.

The implementation is quite straight-forward, assuming of course that you've correctly identified the process. Suppose you've identified that the process is an AR(1) model:

$X_t = mu + a*(X_{t-1} - mu) + Z_t$
where mu is the mean and Zt is white noise.

The next step is to fit the model and determine a and mu. Suppose you find that
$X_t = 72.5 + .7\ X_{t-1} + Z_t$

The idea is that you use this as a recipe for generating the future observations. You don't know what Zt is because it's random, but it's mean is 0, so a common approach is to pretend that its 0.

So, suppose your last observation was 64.5 (today's high temperature?). Then you predict tomorrow will be 72.5 + .7*(-8) = 66.9. (This model, which predicts that each day will be colder than the previous, is obviously inappropriate for temperature, but play along.)

Obviously, your error for tomorrow's prediction won't be that bad. But suppose you want to predict two days into the future. Now the second day's prediction is based on the "fabricated" value 66.9, and the error is going to be compounded by whatever error there is in that first day. As you can see from this example, the further into the future you predict, the greater the error.

The rate at which the error grows is different for different models. (It's slower for AR(p) when p>1 because then you use more of the past "real" observations.) But if you go far enough into the future, the error variance will approach the variance of the process itself. Which is a little like predicting that tomorrow's temperature will be somewhere between 40 and 90 degrees.

In fact, the error for just one step into the future is the same as the white noise variance, so that a 95% confidence interval is $x_{t+1} +- 1.96 * sigma$
where sigma is the white noise standard deviation.

In general, the variance is
$var(X_{t+j}) = sigma^2[\ 1 + sum(i = 1 \text{ to } j-1)\ psi^2_i]$
The weights, psi, depend on just what sort of model you've fit. For AR(1) models, these weights are $psi_i = a_i$ where a is the parameter fit in the model.

So for the example above, suppose that Var(Zt) = 9, so that SD(Zt) = 3.

This means that our prediction for tomorrow's temp is 66.9 +- 1.96*3 = 66.9 +- 5.88.
But our prediction for two days in advance has a margin of error of 1.96 *sqrt(Var(Xt+2)) =
1.96 * 3 * sqrt([1 + a]) = 1.96 * 3 * sqrt([1+.7]) = 7.67 and
1.96 * sqrt(Var(Xt+3)) = 1.96*3*sqrt([1 + a + a2])= 8.70

For AR(p) models, you calculate the weights recursively:

psi_i = sum(k=1 to p) a_i psi_{i-k}
where psi0 = 1 and psi i = 0 for i < 0.
For an AR(2) model this is
$psi_1 = a_1$

$psi_2 = a_1^2 + a_2$

$psi_3 = a_1(a_1^2 + a_2) + a_2 * a_1$

If you have an MA process (or ARMA), then it's more complicated (but only just) because you must back-calculate the residuals:

Xt = mu + a * Zt + b * Zt-1

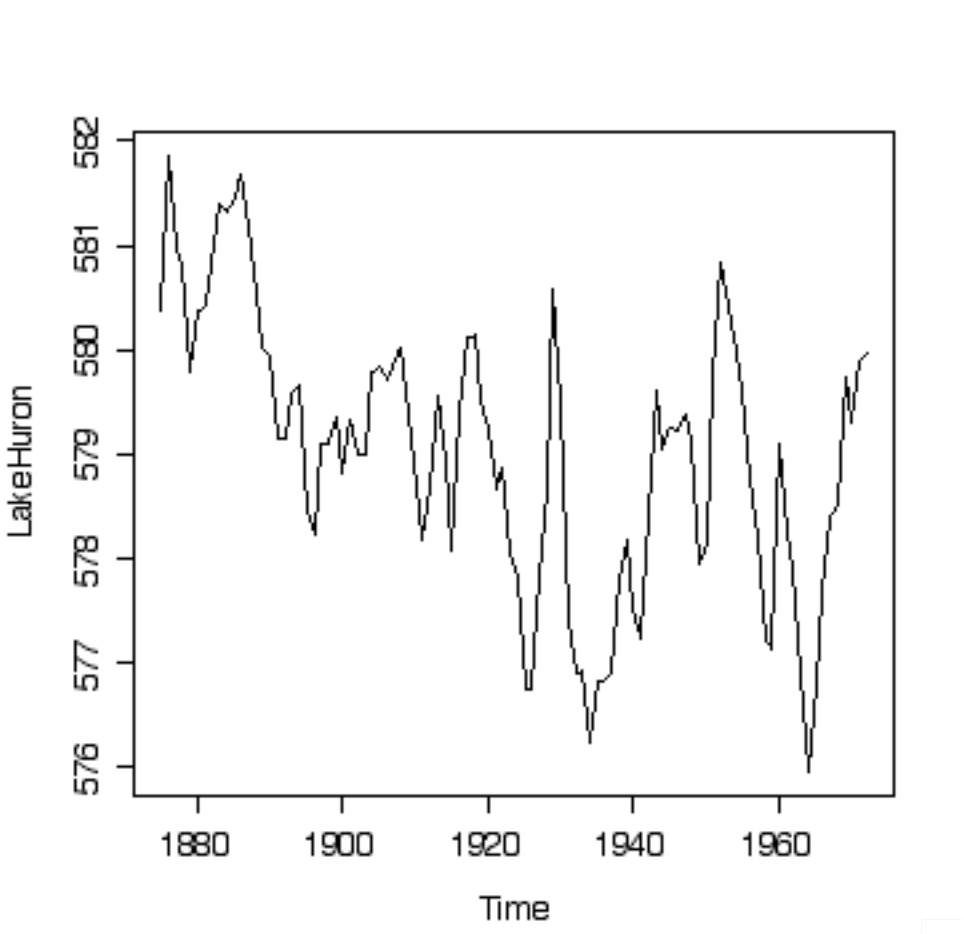
means that if today you saw 64.5, you must determine Zt-1, which means going back to yesterday, seeing what the model predicts for today, and subtracting what it predicts from 64.5 (what you observed), and use this to estimate Zt-1.

One general bit of advice for those advertising a good prediction model: collect enough data so that you can set aside maybe the last 1/3 of the observations. Use this 1/3 to test your ability to predict. You'd be surprised how often this common-sense advice is overlooked. (It's rarely mentioned in textbooks.) But obviously, if your claim is that you can predict the future, you had best demonstrate it on the best data you have.

Example:
Let's reconsider the LakeHuron data. Earlier, we had fit this as an AR(2) via:

```
out <- ar(LakeHuron)
> out$ar
[1]  1.0538249 -0.2667516
```

This routine decided to stop with an AR(2) model since it used the AIC criterion to decide which fit was "best".



There are 98 terms, so let's just use the first 65 to fit the model. The outcome is surprisingly close:

```
> out2$ar
[1]  1.0776872 -0.2591484
> out2$x.mean
[1] 579.1892
```

Xt = = 579.1892 + 1.0776872(Xt-1 - 579.1892) - .259*(Xt-2 - 579.1892) + Zt

Our forcast for the 66th observation, then is

X66 = 579.2 + 1.08(X65 - 579.1892) -.259(X64 - 579.1892) + 0
X66 = 579.2 + 1.08*(577.79 - 579.1892) -.259*(578.18 - 579.1892) + 0
(We get these values from a call to:

```
> newLH[c(64,65)]
[1] 577.79 578.18
```

X66 = 577.9502
We can predict the margin of error in this prediction:

First, we need the SD of the white noise, which we get from

```
> out2$var.pred
[1] 0.4761921
```

Then the margin of error is 1.96 * SD(Xt+1) =
1.96 * sqrt(.476) = 1.35

So we are 95% confident that the next value will be between 577.9502 - 1.35 and 577.9502 + 1.35
or (576.6, 579.3).

The actual value is

```
> LakeHuron[66]
[1] 577.51
```

So we win.

We can write a program to continue this. And of course we can calculate our error for any term in the future. But the real proof will be to compare our predictions to the actual values. One way of doing this is to look at the sum of the squared residuals, which is a general measure of error. This approach is even more powerful if there are two competing methods for prediction. Clearly, the best predictor should win.

Other Approaches

There is a tradition to other approaches that are useful if your only goal is prediction -- and not estimation. Some of these approaches are "non-parametric" in that they don't assume any particular model, but instead are somewhat "seat of the pants" approaches to making good predictions.

Exponential Smoothing

One of these is called "exponential smoothing." it works like this:

Let X(N,1) be the 1st prediction based on a series of length N. In general X(N,q) is the qth prediction after the last in the time series. We'll use upper case X to represent predicted values and upper case x for the observed. The idea is to predict X(N,1) with a weighted sum of the past terms. Because most recent terms are the most influential, our weighted sum should give more weight to those than to the distant past. As a general constraint, these weights must add to 1.

One set of weights that does this is

$c_i = a(1-a)^i$ for i = 0,1,... and 0 < a < 1

So that

X(N,1) = axn + a(1-a)xn-2 +.....

Factor out the a's and we get
X(N,1) = a xN + (1-a)X(N-1,1)

Define the error at time N to be eN = xn - X(N-1,1) (the difference between the observed value and the predicted value.

An algorithm, then is this:
1) Choose a0 arbitrary at some low value, say something between .1 and .3.
2) Let X(1,1) = x1 (so our prediction for X2 is x1)
3) e2 = x2 - X(1,1)
4) X(2,1) = a0 * e2 + X(1,1)
5) e3 = x3 - X(2,1)
and continue up to eN.
6) Compute the sum of the squared e's
7) Repeat process with a1 = a0 + .1
8) Continue until you've generated a set of a's, and then choose the one with the smallest sum of squared e's. the process is not to sensitive to a, and so you can use fairly crude steps.

Box-Jenkins Approach

This method fits an ARIMA(p,d,q) model. The strategy is something like this:
1) Make a correlagram to see if data are stationary. If not, compute new series consisting of first-order differences (dXt = Xt - Xt-1) and repeat.
If the data are seasonal with period s, try d*ds(Xt) as your difference. That is
$d(X_t - X_{t-s} = (X_t - X_{t-1}) - (X_{t-s} - X_{t-(s-1)})$
(A spectral analysis might be useful for help with this.)
2)If you now have non-stationary, non-seasonal data, fit an ARMA.
3)If you now have non-stationary, seasonal data, then we fit a general "seasonal" ARMA as follows:
Examine the correlogram of the difference series produced in 1. Use the partial acf to help determine the order of the AR part (p). (This is the last non-zero term). The correlgram itself can tell you the order of the MA process, q; if there's a sudden drop off to 0, q is the last non-zero term. (If there's no sudden drop off, maybe q = 0.) Further details require more notation and exposition. Refer to Box and Jenkins, (19780), Chpater 9, "Time Series Analysis, forecasting and Control."

Once the model is fit, forcasting proceeds as it did in the beginning of this discussion (for AR and MA models.)