1.Akaike-Information-Criterion-and-Model-Quality

2020年5月23日 星期六　　下午8:23

T.S. dataset – Process – model – PACF → p

Regression = 2D – plot

no – numeric measurement – R^2

_715a8285 70906ee...

AR(2) process simulation ✓ order ✓ coef – arima.sim – generate data

✗ coef – estimate – arima

✗ order ✗ coef – estimate – arima – SSE, AIC

Measuring the Quality of a Model

At this point we can do a lot! We understand Autoregressive Processes and Moving Average Processes. We can even, for example using Yule-Walker, estimate the coefficients in an Autoregressive Process. To be honest, we'll usually rely on software for estimating coefficients in models, but it's nice to know how to obtain these techniques anyway because it deepens our general understanding.

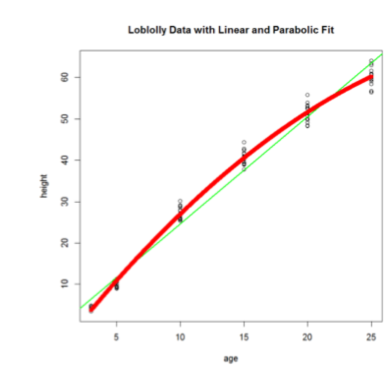
Moving forward, an important issue in modeling is deciding *which* model we believe describes the process which generated our data set. This is a more subtle question than it appears at first glance. Time Series data collected in the field doesn't come labelled with the generating process! Unless we have deep subject-matter knowledge, we have to work to determine a good model for our data.

As a concrete example, suppose you have time series data and you believe an *AR(p)* stochastic process is a good model. We've used the Partial Autocorrelation Function (PACF) to make a guess at the best value for *p,* the order of the process. Unless the series is quite long and therefore we have a lot of data from which to estimate the PACF coefficients, this is a pretty subjective way to proceed. Also, we will soon look at processes with both *AR(p)* and *MA(q)* components (as well as trend and seasonality), called *ARMA* models. We will even consider data sets exhibiting a trend, bringing us to integrated moving average, or *ARIMA* models, and finally data sets with a periodic component or seasonality, called *SARIMA* models. A quick glance at the *ACF* or *PACF* won't be enough to judge the quality of a model or to determine the proper values for *p* and *q* and the other estimated quantities.

Our conundrum is similar to what we find in Regression, so let's first explore these ideas on familiar territory. Take a look at our old friend, the Loblolly Pine data. If we fit a straight line model (green) we have a decent fit, but a parabola (red) looks even better. (There is some heteroscedasticity in that the groups of data points develop increasing spread over time, but we'll ignore that for the moment.) We don't always have the luxury of plotting and then choosing (think of a model with several input variables), so we would like to develop a numerical measure of quality. This will help us choose between candidate models.

there is a trend: taller

variability ↑ right



We've discussed the coefficient of determination before (often thought of as percentage of explained variability), and this provides a mechanism for preferring a quadratic to a linear fit even without plotting and "eye-balling". Our R printout calls this *Multiple R-squared.*

Linear Fit:　　*Multiple R-squared: 0.9799,*　*Adjusted R-squared: 0.9797*

Quadratic Fit:　　*Multiple R-squared: 0.9934,*　*Adjusted R-squared: 0.9932*

Recall that the multiple R^2 value gives the proportion of explained variance, while the corresponding adjusted term R_a^2 makes each variable "pay a tax" to enter the model. There are other measures of quality used in linear regression, such as the Akaike Information Criterion (AIC), but the adjusted R_a^2 coefficient is probably the most common and regularly appears in the default printout of computer software.

pay a penalty – higher order terms of model

order: 1→2 justify

2→3 take dataset

enhance in variability

worth 3rd order

Measuring the Quality of a Time Series Model

There are a variety of ways to judge the quality of a time series model. We'll discuss two common ways: *SSE* and *AIC*. And, rather than deal with the complexities of a natural process, which can be quite messy to model, let's develop our ideas by generating some data from an *AR(p)* process so that we already know the order of the model. Then we work to understand how our measures behave.

We'll use the trusty *arima.sim()* which, of course, simulates *a*uto*r*egressive *i*ntegrated *m*oving *a*verage models.

We'll start with something simple and simulate the process

$$X_t = Z_t + .7X_{t-1} - .2X_{t-2}$$

We start by creating a "list" to specify the parameters as well as include the coefficients of the process: *list(order = c(2,0,0), ar =c(0.7, -.2)).* Generally speaking, as we will explore in greater detail later this week, by order we mean that we describe the process in terms of its autoregressive order (2), the order of any differencing (0), and then the order of the moving average part (0).

Our code will therefore be:

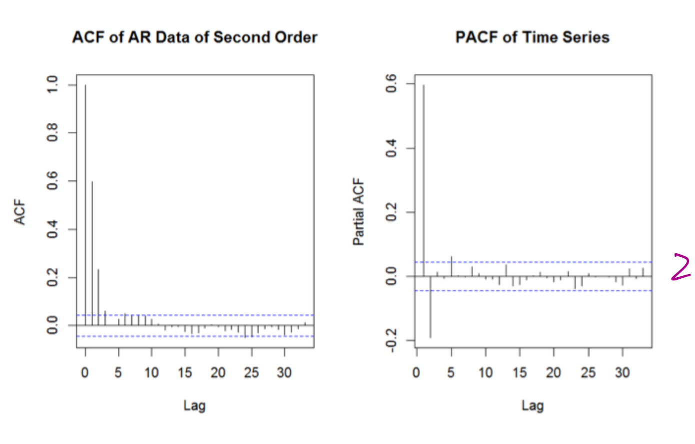
```
rm(list=ls(all=TRUE))
set.seed(43)            #Roman conquest of Britain
data = arima.sim( list(order = c(2,0,0), ar =c( 0.7, -.2)), n = 2000)
```

2000 data points – large dataset.

Now plot our ACF and PACF (you can plot the time series data itself as well):

```
par(mfrow=c(1,2))
acf(data, main="ACF of AR Data of Second Order")
acf(data, type="partial", main="PACF of Time Series")
```

Look at the PACF and feel satisfied that this is consistent with a second order model (some spikes will rise past our dotted line by chance…you can run a much longer time series to see the spikes settle in).



It may feel a little artificial to estimate the coefficients from the time series data (since we already know what they are), but we are exploring the behavior of our estimation process and, in general, we won't know the coefficients of a process and will need to estimate them from the data alone.

Rather than estimate the coefficients by writing our own routines, we can invoke a very useful command called *arima()*. This command will estimate the parameters of the model for you as long as you tell it the order of the process. For example, I ran this on our data and found:

```
arima(data, order=c(2,0,0), include.mean=FALSE )

Coefficients:
          ar1       ar2
       0.7111   -0.1912
s.e.   0.0219    0.0220

sigma^2 estimated as 0.9985:  log likelihood = -2836.64,   aic = 5679.27
```

guess p value

AIC - educated guess

These estimates compare quite favorably to our established values $\phi_1 = .7, \phi_2 = -.2$.

Again, unless you already happen to know that your *AR(p)* process is of a certain order, it is hard to look at the *ACF and PACF* and really say with assurance what the order of the process is. I ran the command

```
arima(data, order=c(p,0,0) , include.mean=F)
```

for various values of the order *p* and produced the following table. So should you! The AIC value is part of the standard print out. I computed the SSE values as *sum(resid(m)^2).* More on that later!

Order of AR process, p	1	2	3	4	5
coefficients	0.5970	0.7111 -0.1912	0.7136 -0.2003 0.0128	0.7137 -0.2016 0.0176 -0.0066	0.7141 -0.2027 0.0302 -0.0515 0.0629
AIC	5751.732	5679.274	5680.945	5682.857	5676.917
SSE	2072.832	1997.007	1996.678	1996.590	1988.660

drop

If you'd like some code, here it is!

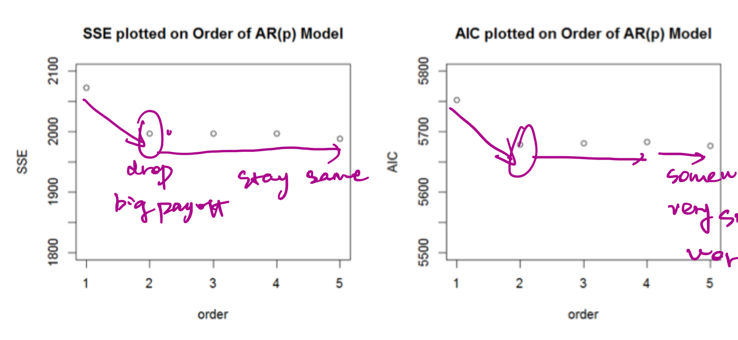
```
SSE=NULL
AIC=NULL
for (p in 1:5) {
     m = arima(data, order=c(p,0,0), include.mean=FALSE )

     SSE[p] = sum(resid(m)^2)
     AIC[p] = m$aic

     print( m$coef )
     print( paste(m$aic, sum(resid(m)^2)) )
}
```

Using the Residual Sum of Squares

The model we are trying to develop

$$X_t = Z_t + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p}$$

attempts to predict the value of the time series at time *t*, call it x_t, by using the values at the *p* preceding time steps. A naïve suggestion would be to try an order, e.g. *p=3*, ① estimate the coefficients for this order, and then ③ determine the residual sum of squares for our ② candidate model. fit a model We could then pick whichever model gives us the lowest aggregate residual sum of squares. We have our 5 models for *p=1* through *p=5*. We can extract the residuals from each, and then look at the sum of the squares of the errors (SSE) for each. When I plot the residual squared terms against the order I obtain the graph on the left.




```
par(mfrow=c(1,2))
order=c(1,2,3,4,5)
plot(SSE~order, main="SSE plotted on Order of AR(p) Model", ylim=c(1800, 2100))
plot(AIC~order, main="AIC plotted on Order of AR(p) Model", ylim=c(5500, 5800))
```

Adding more terms reduces the *SSE*, but the really big drop occurs when we move from *p=1* to *p=2*. After that, the enhancements are really much more modest. There seems to be a tradeoff between added complexity and a meager diminution of the *SSE*. On the right I've also plotted the AIC against the order, as discussed below. Based on these plots we would select a 2nd order model.

Using the AIC

The AIC tries to help you assess the relative quality of several competing models, just like adjusted R^2 in linear regression, by giving *credit for models which reduce the error sum of squares* and at the same time by building in a *penalty for models which bring in too many parameters*. Obviously there's a fair amount of theory behind the exact form of the AIC. This statistic is expressed somewhat differently in various software packages, depending on how they adjust the formula (which is based on maximum likelihood estimation). The version appearing in Akaike's 1974 paper is

as you're flying 30,000 feet.

keep basic idea in your head

↓ SSE

pay penalty

$$AIC = -2 \cdot \log(maximum\ likelihood) + 2 \cdot (number\ of\ parameters\ in\ the\ model)$$

Here is a simple version of a formula for the AIC of a given model with *p* terms:

$$AIC = \underbrace{\log(\hat{\sigma}^2)}_{SSE} + \underbrace{\frac{n + 2 \cdot p}{n}}_{\text{pay penalty} - n}, \quad \text{where} \quad \hat{\sigma}^2 = \frac{SSE}{n}$$

While different authors and software packages use varying forms of the AIC, often eliminating constant terms that are the same for various candidate models. In the end, we are looking to compare the AIC for a variety of candidate models and as long as we can make relative comparisons we are OK. ***We prefer a model with a lower AIC***. In the current case, that means we'd believe a two parameter model is the best (though only narrowly) and we would model our time series as

$$X_t = 0.7111 X_{t-1} + -0.1912\ \ X_{t-2} + Z_t$$