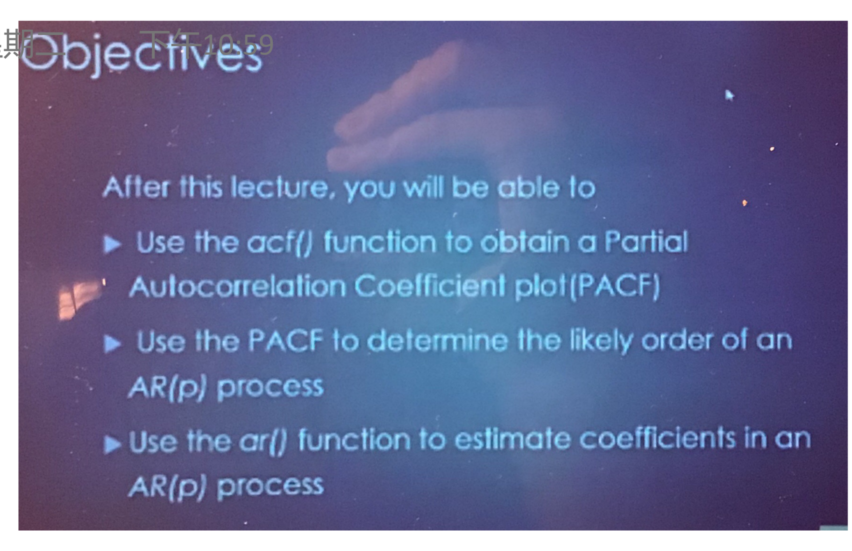
Partial-Autocorrelation-and-the-PACF-First-Examples

2020年5月19日 星期



_3dd83ce2db3af01...

The Partial Autocorrelation Function

Let's suppose you have time series data in front of you, and you have plotted the series as well as plotted the ACF. Unless you have created the data yourself, say with *arima.sim()*, or if you have good knowledge of the physical process that generated your time series, it's not easy to tell from the time series data alone just what the generating process is.

data
x obvious how model it.
model t.s. → stochastic process

Now the ACF is one of our primary tools for characterizing an autoregressive or moving average process. We've already seen how to find the ACF as a formula for *MA(q)* and *AR(p)* processes and learned to predict the characteristic shapes which occur in the ACF for these simple processes. First, the "good news":

A moving average process of order q has an ACF that cuts off after q lags.

So, if you have an *MA()* process with enough terms that you believe the ACF is well estimated, and the ACF cuts off after 4 lags, you can be reasonably sure you have an *MA(4)* process.

How about an *AR(p)* process? That is, if you know you have an *AR(p)* process, can you tell what the *order* of the process is? It would be terrific if we had a plot that would function for an *AR(p)* process the way the ACF does for the *MA(q)* process.

Generating Data and Observing the Partial Autocorrelation Function

More "good news"! As we develop in this lecture, the Partial Autocorrelation Function, or PACF will help us to find the order of an *AR(p)* process. We will develop this graph in the lecture below. First, let's create some data and look at some pictures. AR(p) process

$X_t = Z_t + \phi_1 X_{t-1} + \dots + \phi_p X_{t-p}$

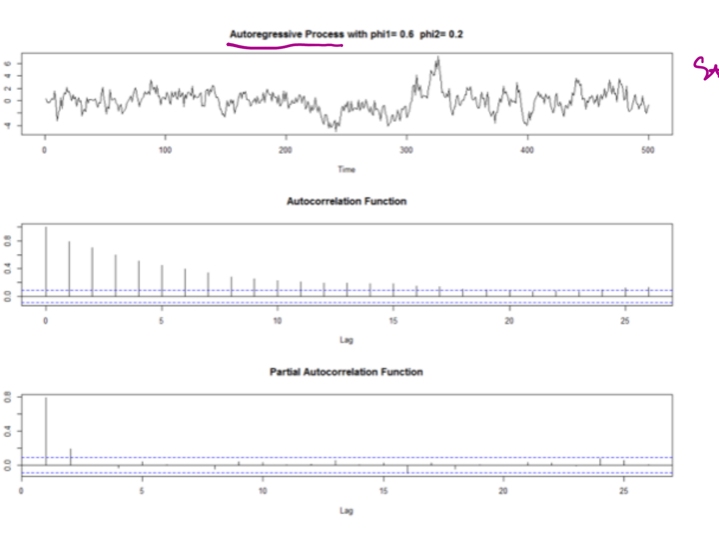
simulation AR(2)
generate data
get rid of all rags stored
nice, organizes
erase blackboard

```
rm(  list=ls( all = TRUE )  )
par(mfrow=c(3,1))

phi.1 = .6; phi.2 = .2; data.ts = arima.sim(n = 500, list(ar = c(phi.1, phi.2)))

plot(data.ts, main=
        paste("Autoregressive Process with phi1=",phi.1," phi2=",phi.2) )
acf(data.ts, main="Autocorrelation Function")
acf(data.ts, type="partial", main="Partial Autocorrelation Function")
```

PACF

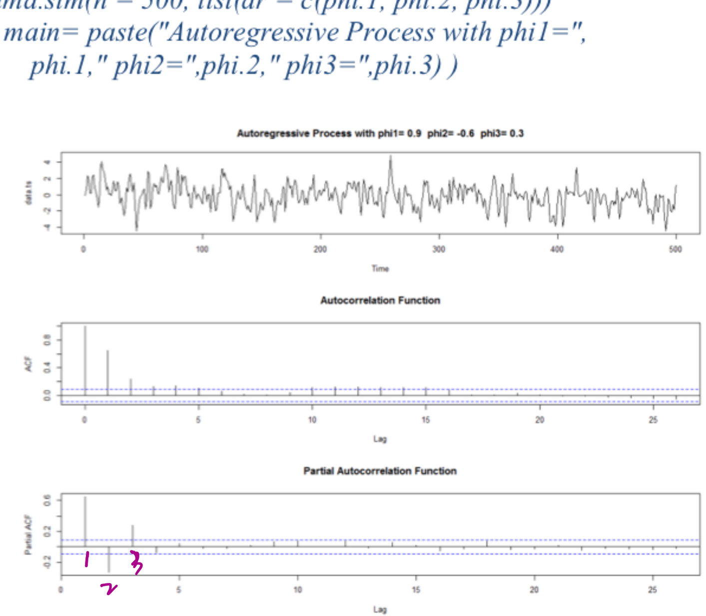


You should run the code several times. You will observe that, while the actual time series itself changes from simulation to simulation, the ACF and PACF are relatively constant. Now run the code again, this time with the following two lines substituted into the obvious place:

simulation AR(3)
coefficients

```
phi.1 = .9; phi.2 = -.6; phi.3 = .3;
data.ts = arima.sim(n = 500, list(ar = c(phi.1, phi.2, phi.3)))
plot(data.ts, main= paste("Autoregressive Process with phi1=",
             phi.1," phi2=",phi.2," phi3=",phi.3) )
```

– random → stationary



You can continue to play with the code, changing the number of terms and the coefficients used in your simulations. Do you have any conjectures?

Partial Autocorrelation Function and the Beveridge Wheat Price Data Set

Here is a classic data set in Time Series Analysis, the good old Beveridge Wheat Price Index. You can get the numbers at the Time Series Data Library…just visit the website:

http://datamarket.com/data/list/?q=provider:tsdl

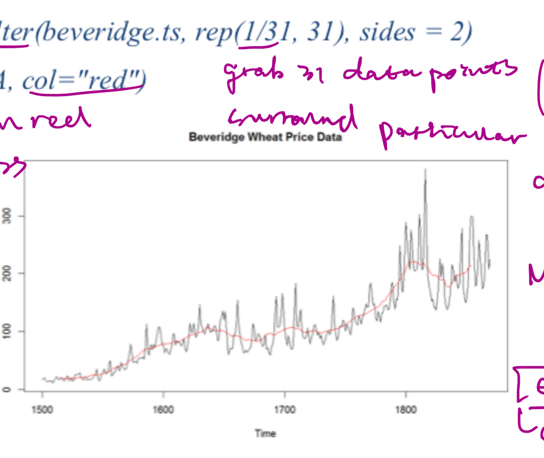
The data set originally appeared in a paper called "Weather and Harvest Cycles" (Beveridge, 1921) and has also been discussed in subsequent papers critiquing the original analysis, for example (Sargan, 1953) and (Granger & Hughes, 1971). While we note that there are issues with Beveridge's analysis, we present the data as a nice illustration of the topic at hand.

Can you follow along on this code? I have downloaded the data into a text file, removing some header information. Next, I create a time series with the data in the second column (the actual prices, starting in the year 1500) and create a "filter" with the simple moving average that uses 15 data points on either side of a given year to introduce some smoothing. We plot the original series and the smoothed series on the same axes.

create t.s.
extract 2nd col.

```
beveridge      = read.table("beveridge.txt", header=TRUE)
beveridge.ts   = ts(beveridge[,2], start=1500)
plot( beveridge.ts, ylab="price", main="Beveridge Wheat Price Data")
beveridge.MA = filter(beveridge.ts, rep(1/31, 31), sides = 2)
lines(beveridge.MA, col="red")
```

start at year 1500
grab 31 data points
15 up
15 down
surround particular data point at any time
along representation 31 data point
superimpose in red
in MA process

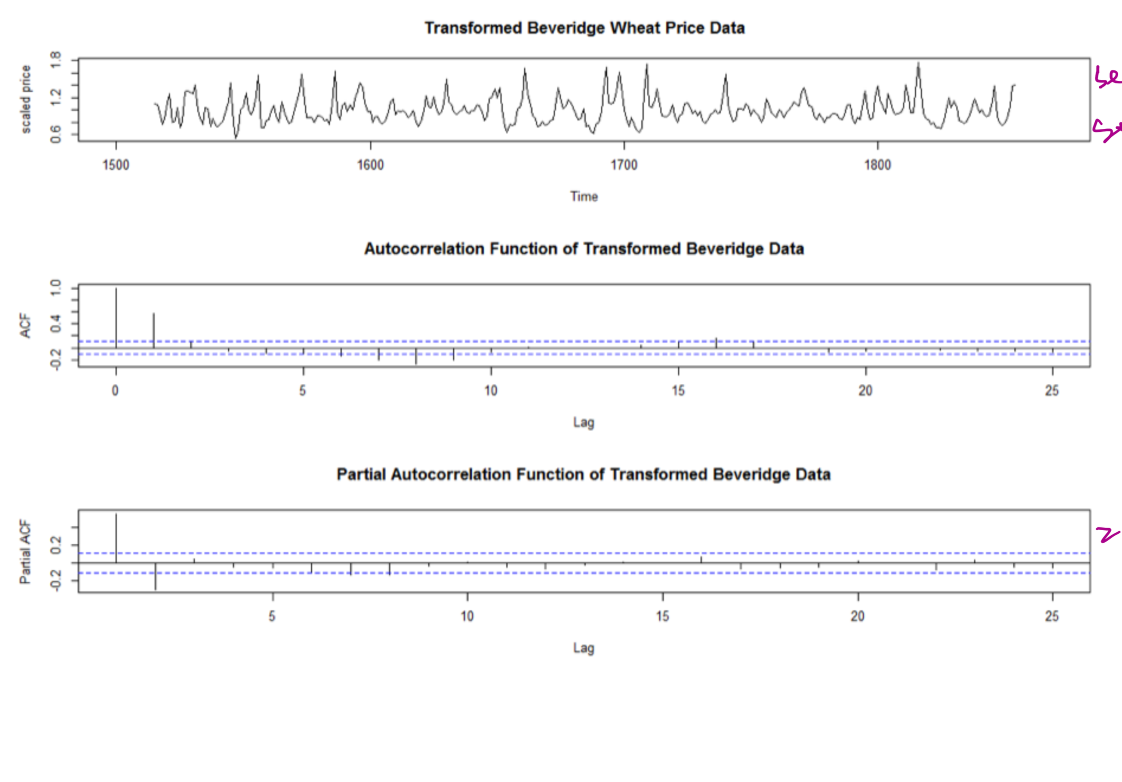


Now Beveridge transformed his data by scaling each data point by its corresponding smoothed value. I've done this with the following lines, and plotted the usual graphs.

```
par(mfrow=c(3,1))
Y =  beveridge.ts/beveridge.MA
plot( Y, ylab="scaled price", main="Transformed Beveridge Wheat Price Data")
acf(na.omit(Y),
        main="Autocorrelation Function of Transformed Beveridge Data")
acf(na.omit(Y), type="partial",
        main="Partial Autocorrelation Function of Transformed Beveridge Data")
```

scale at each data point by MA process
some data – beginning & end – x represented in Y – NA
get rid of data points x meaningful
ACF
– PACF

The *acf()* function doesn't like missing data, so the first and last 15 numbers are ignored or omitted with the clean-up function *na.omit()*. What do you notice about the Partial Autocorrelation function?



We will use the routine *ar()* to estimate the coefficients of our model. In other lectures we describe how to do this; for now, just think of this as similar to a call to *lm()* when we are doing a regression curve fitting. We believe we have a model

$$AR(p)\ process: \qquad X_t = Z_t + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p}$$

We let the *ar()* function find the coefficients for us. We discuss selecting a model using quality criteria such as the Akaike Information Criterion (AIC) in other lectures. As R will tell you, the *ar()* routine will "Fit an autoregressive time series model to the data, by default selecting the complexity by AIC." By complexity we mean how many terms to take, or what the value of *p* is. If we will allow up to 5 terms in our model (a reasonable number) we call

generate coefficients in AR process

```
ar(na.omit(Y), order.max = 5)
```

t.s – estimate coefficients in stochastic process
ar free look for stochastic process param up to 5

And obtain a second order model

```
Coefficients:
      1          2
 0.7239    -0.2957

Order selected 2          sigma^2 estimated as  0.02692
```

= PACF plan

Just to state the obvious at this point:

An autoregressive process of order p, an AR(p), has a PACF that cuts off after p lags.

That's handy!

Now that we know one of the ways we use a PACF, let's discuss the intuition behind the PACF and how to calculate it for a stochastic process and how to estimate it for some given time series data.