Washington University
Department of Economics

Fall 2009
James Morley

Economics 518B

Homework #1
Simulating an AR(1)
Due: Thursday 9/3

This homework will involve generating output from GAUSS. Please use a different program (e.g., Excel or EViews) for the figures (GAUSS figures are particularly ugly!). Compile all of your results in a word processor. Try to conserve paper in presenting the results. In particular, the figures do not need to be large, although I should be able to see what the units of measurement are.

1. GAUSS Exercise

Type up the attached program in a text editor (don't forget those semicolons at the end of each line!) and save the file as "hw1.gau". You may have to rename the file to make sure it is "hw1.gau" rather than, say, "hw1.gau.txt". Open GAUSS. Go to the file menu. Select "Edit" and find the "hw1.gau" file. Open the file in GAUSS. The program "hw1.gau" will allow you to simulate and plot first-order autoregressive processes for different parameter values. Note the autoregressive parameter phi is initially set to 0.5. Also, be careful in distinguishing "`1`" (i.e., the number one) and "`l`" (i.e., the letter before m).

- Run the program and look at resulting data (hw1.dta) in Excel to see what the resulting series looks like. Repeat this a few times to get a sense of what an AR(1) process with phi=0.5 looks like. Don't print output yet.
- Try different values for phi between –1 and 1, including phi=–0.9 and phi=0.9.
- Try phi equal to one (the random walk case).
- Try phi equal to 1.2 (an explosive case).

a. Generate a plot of the series for the following cases: phi=-0.9, 0, 0.5, 0.9, 1, and 1.2. Comment on any notable features of the various plots. Does your plot for the phi=1 case look more persistent than the phi=0.9 case? What do you find for the phi=1.2 case?
b. Repeat exercise a. for sample size (T) equal to 500.
c. Comment on differences in findings across sample sizes.

Now, find the "/*" below the line that reads "@here@". Delete this "/*" tag and the "*/" tag at the end of the program.

a. Set phi=0.5 and T=50. Run the program. You should get output for both conditional maximum likelihood and exact maximum likelihood. The difference is that conditional MLE treats the first observation as a given constant (log likelihood=0 for the observation) rather than as a random variable drawn from the unconditional distribution of Y. With a low value for phi, you should find that the estimates are similar for both types of MLE (note that c is the intercept from a regression model form used to estimate CMLE, while mu is the unconditional mean directly estimated by EMLE). Report your estimates in table format.

b. Change to T=20. Run the program. Report your results. Are the results for both types of MLE still as similar? Comment on what you would expect the role of sample size to be in the relationship between the estimates?
c. Set T=50 and change phi to 0.9. Run the program. Report the results. Are your estimates for both types of MLE as similar for the T=50 and phi=0.5 case. Comment on what you would expect the role of phi to be in the relationship between the estimates?
d. Set phi to 1. Run the program ten times. Report the estimates of phi. What do you notice about them?
e. Set T=500. Run the program ten times. Report the estimates of phi. What do you notice about your estimates of phi? What about the similarity between the estimates for both types of MLE?
f. Experiment with different combinations of true parameters. (You do not need to report results.) For T=50, how do different phi's affect the accuracy of your estimates of the unconditional mean mu (note that mu is not calculated for CMLE unless abs(phi)<1, so only consider values of phi that satisfy this constraint)? What is the intuition behind your result?
g. Find a "real world" data series that looks like an AR(1) with phi=0.5. Plot the real series along side a simulated series for phi=0.5 (with the same sample size as the real world series). Explain in what sense the series "look similar".

```
/* PROGRAM SIMULATES AR(1) DATA
ALSO, CALCULATES MLE FOR AR(1)

3 PROCEDURES
1. Log likelihood for CMLE
2. Log likelihood for full sample MLE
3. Transformation to constrain parameters

PROGRAM DESIGNED BY JAMES MORLEY 1/23/2000
MODIFIED 8/7/2008 */

@INITIAL GAUSS STUFF@
new; @clear memory@
format /m1 /rd 9,6; @set details of output format@


@SIMULATE DATA@
mu=1; @unconditional mean@
phi=0.5; @autoregressive coefficient@
sig_e=0.5; @standard deviation of error term@
T=50; @sample size@

if abs(phi) ge 1;
y0=0; @initial demeaned value for nonstationary case@
else;
y0= sqrt((sig_e^2)/(1-phi^2))*rndn(1,1); @initial demeaned value for
stationary case@
endif;

e=sig_e*rndn(T,1); @Normal random errors with variance sig_e^2@
u=recserar(e,y0,phi); @autoregressive series with zero mean@

y = mu + u; @autoregressive series with mean mu when abs(phi)<1, random
walk with no drift when phi=1@

output file=hw1.dta reset;  @change working directory to wherever you
want this file to go@
output on;
y~mu*ones(T,1); @write the series and unconditional mean to a text file
for use in EViews or Excel@
output off;

@here@
/*

output file=hw1.out reset;
output off;


@ESTIMATE CMLE AR(1) PARAMETERS@
prmtr_in=0|phi|ln(sig_e);
{xout,fout,gout,cout}=QNewton(&lik_fcn1,prmtr_in);
prm_fnl1=trans_cmle(xout);
lik1=-fout;
                hessn0=hessp(&lik_fcn1,xout); @calculate hessian@
                cov0=inv(hessn0); @estimate var-cov for parameters@
            grdn_fnl=gradp(&TRANS_CMLE,xout);
            cov=grdn_fnl*cov0*grdn_fnl';
@delta method to address transformation@
            SD1 =sqrt(diag(cov)); @calculate standard errors@

@ESTIMATE exact AR(1) PARAMETERS@
prmtr_in=mu|0|ln(sig_e);
{xout,fout,gout,cout}=QNewton(&lik_fcn2,prmtr_in);
prm_fnl2=trans_emle(xout);
lik2=-fout;
                hessn0=hessp(&lik_fcn2,xout); @calculate hessian@
                cov0=inv(hessn0); @estimate var-cov for parameters@
            grdn_fnl=gradp(&TRANS_EMLE,xout);
            cov=grdn_fnl*cov0*grdn_fnl';
@delta method to address transformation@
            SD2 =sqrt(diag(cov)); @calculate standard errors@
output on;
"CMLE Estimates (standard errors in parenthesis)";
"c          phi         sig_e";
prm_fnl1';
"(";;sd1';;")";
""; "ll=";;lik1;
"";
if abs(prm_fnl1[2])<1; "The unconditional mean mu=";;prm_fnl1[1]/(1-
prm_fnl1[2]); else; "mu cannot be calculated because abs(phi) ge 1";
endif;

"";

" Exact MLE Estimates (standard errors in parenthesis)";
"mu         phi         sig_e";
prm_fnl2';
"(";;sd2';;")";
""; "ll=";;lik2;
output off;

END; @End of main program@

/*--------------------*/
@PROCEDURE TO CALCULATE CONDITIONAL LIKELIHOOD VALUE FOR AR(1)@

proc lik_fcn1(prmtr1);

     local prmtr,c_hat,phi_hat,sig_hat,j,e_t,llikv;
@local variables for procedure@

     prmtr=trans_cmle(prmtr1); @constrain parameters@

      c_hat=prmtr[1];
      phi_hat=prmtr[2];
      sig_hat=prmtr[3];

@Initialize for CONDITIONAL MLE@
llikv = 0;

@Errors 2 through T needed to calculate CONDITIONAL (treat y[1] as
given) likelihood@
j=2;
```

```
do until j>T;
e_t = y[j] - c_hat - phi_hat*y[j-1];
llikv = llikv -0.5*ln(2*PI*sig_hat^2)-0.5*(e_t^2)/sig_hat^2; @log
likelihood value@
j=j+1;
endo;

retp(-llikv); @return minus llikv since qnewton minimizes function@
endp;

/*--------------------*/
@PROCEDURE TO CALCULATE EXACT LIKELIHOOD VALUE FOR AR(1)@

proc lik_fcn2(prmtr1);

     local prmtr,mu_hat,phi_hat,sig_hat,j,e_t,llikv;
@local variables for procedure@

     prmtr=trans_emle(prmtr1); @constrain parameters@

      mu_hat=prmtr[1];
      phi_hat=prmtr[2];
      sig_hat=prmtr[3];

@Initialize for FULL SAMPLE MLE@
e_t = y[1] - mu_hat;
llikv = -0.5*ln(2*PI*((sig_hat^2)/(1-phi_hat^2)))
-0.5*(e_t^2)/((sig_hat^2)/(1-phi_hat^2));
@log likelihood value@

@Errors 2 through T and log likelihood@
j=2;
do until j>T;
e_t = y[j] - mu_hat - phi_hat*(y[j-1] - mu_hat);
llikv = llikv -0.5*ln(2*PI*sig_hat^2)-0.5*(e_t^2)/sig_hat^2;
@log likelihood value@
j=j+1;
endo;

retp(-llikv); @return minus llikv since optmum minimizes@
endp;

/*--------------------*/
@PROCEDURE TO CONSTRAIN PARAMETERS@
proc trans_cmle(c0);

local c1;

      c1 = c0;
      c1[3]=exp(c0[3]); @constrain variance to be non-negative@

retp(c1);
endp;
/*--------------------*/
```

```
/*--------------------*/
@PROCEDURE TO CONSTRAIN PARAMETERS@
proc trans_emle(c0);

local c1;

      c1 = c0;
      c1[2]=c0[2]/(1+abs(c0[2])); @constrain AR to be stable@
      c1[3]=exp(c0[3]); @constrain variance to be non-negative@

retp(c1);
endp;
/*--------------------*/

*/
```