Quantitative Macroeconomics

Winter 2023/24

Week 5

Willi Mutschler
willi@mutschler.eu

Version: 1.0
Latest version available on: GitHub

Contents

1. Information Criteria For AR(p)

Consider the following information criteria to estimate the order p of an $AR(p)$ model:

$$AIC(n) = \log \tilde{\sigma}^2(n) + \frac{2}{T^{eff}} n$$
$$SIC(n) = \log \tilde{\sigma}^2(n) + \frac{\log T^{eff}}{T^{eff}} n$$
$$HQC(n) = \log \tilde{\sigma}^2(n) + \frac{2 \log \log T^{eff}}{T^{eff}} n$$

where $\tilde{\sigma}^2$ denotes the ML estimate of the variance term based on the residuals $\hat{u}_t(n)$ of the corresponding estimated $AR(p)$ model. n is the number of estimated parameters and $T^{eff} = T - p^{\max}$, where $p^{\max}$ is the maximum number of lags to consider.

1. Provide intuition between the different criteria. Which one (asymptotically) over- or underestimates the correct order?

2. Write a function `nlag = lagOrderSelectionARp(`y`,`$const$`,`$pmax$`,`$crit$`)` that computes the different order criteria for $p = 1, \ldots, p^{\max}$ using data vector y and possible constant term ($const = 1$) or constant term and linear trend ($const = 2$). $nlag$ should output the recommended lag according to criteria $crit$, which takes a string ($'AIC'$, $'SIC'$ or $'HQC'$) as input value.

3. Load the dataset of the simulated AR(4) process given in the CSV file `AR4.csv`. Which model is preferred according to the order selection criteria?

4. Simulate another stationary AR(4) processes with sample sizes $T = 150$ and redo the previous exercise.

Readings

- Lütkepohl (2004).

2. Portmanteau Test For Residual Autocorrelation

The portmanteau test checks the null hypothesis that there is no remaining residual autocorrelation at lags 1 to h against the alternative that at least one of the autocorrelations is nonzero. In other words, the pair of hypotheses:

$$H_0 : \rho_u(1) = \rho_u(2) = \cdots = \rho_u(h) = 0$$

versus:

$$H_1 : \rho_u(j) \neq 0 \text{ for at least one } j = 1, \ldots, h$$

where $\rho_u(j) = Corr(u_t, u_{t-j})$ denotes an autocorrelation coefficient of the residual series.

Consider the Box-Pierce test statistic Q_h

$$Q_h = T \sum_{j=1}^{h} \hat{\rho}_u^2(j)$$

which has an approximate $\chi^2(h-p)$-distribution if the null hypothesis holds and T is the length of the residual series. The null hypothesis of no residual autocorrelation is rejected for large values of the test statistic.

- Load quarterly data for the price index of US Gross National Product given in `gnpdeflator.csv`. This is a chain-type price index with basis year 2005. The data is seasonally adjusted and spans the period from 1954.Q4 to 2007.Q4.
- Compute the inflation series. That is, take the first difference of the log of gnpdeflator.
- Use the Akaike information criteria to determine the lag length $\hat{p}$.
- Estimate two models with OLS: (i) an $AR(\hat{p})$ model and (ii) an $AR(1)$ model.
- Set $h = \hat{p} + 10$ and compute Q_h as well as the corresponding p-value for both models.
- Comment, based on your findings, whether the residuals are white noise.

Readings:

- Lütkepohl (2004).

3. Bootstrap Confidence Interval For AR(1) Coefficient

Consider the AR(1) model with constant

$$y_t = c + \phi y_{t-1} + u_t$$

for $t = 1, \ldots, T$ with iid error terms u_t and $E(u_t|y_{t-1}) = 0$. Usually, we construct a $(1-\alpha)\%$-confidence interval for ϕ using the normal (or student's t) approximation:

$$\left[\hat{\phi} - z_{\alpha/2} \cdot SE(\hat{\phi});\ \hat{\phi} + z_{1-\alpha/2} \cdot SE(\hat{\phi})\right]$$

with $\hat{\phi}$ denoting the OLS estimate, $SE(\hat{\phi})$ the estimated standard error of ϕ and $z_{\alpha/2}$ the $\alpha/2$ quantile of the standard normal distribution (or t-distribution). If one does not know the asymptotic distribution of a test statistic (or it has a very complicated form), one often relies on a nonparametric simulation approach. To this end, we are going to do a so-called "*bootstrap*", i.e. we recompute the t-statistics a large number of times on artificial data generated from resampled residuals.

1. What is a "Bootstrap approximation"? Provide insight into the basic idea and possible applications of this statistical technique.
2. Write a program for the following:
   - Simulate $T = 100$ observations with $c = 1$, $\phi = 0.8$ and errors drawn from e.g. the exponential distribution such that $E(u_t) = 0$.
   - Estimate the model with OLS and calculate the t-statistic $\tau = \frac{\hat{\phi}}{SE(\hat{\phi})}$.
   - Store the OLS residuals in a vector $\hat{u} = (\hat{u}_2, \ldots, \hat{u}_T)'$.
   - Set $B = 10000$ and initialize the output vector $\tau^* = (\tau_1^*, \ldots, \tau_B^*)$.
   - For $b = 1, \ldots, B$:
     - Draw a sample **with replacement** from $\hat{u}$ and save it as $u^* = u_2^*, \ldots, u_T^*$.
     - Initialize an artificial time series y_t^* with T observations and set $y_1^* = y_1$.
     - For $t = 2, \ldots, T$ generate

       $$y_t^* = \hat{c} + \hat{\phi} y_{t-1}^* + u_t^*$$

     - On this artificial dataset estimate an AR(1) model. Denote the estimated OLS coefficient ϕ^* and corresponding estimated standard deviation $SE(\phi^*)$. Store the following t-statistic in your output vector at position b:

       $$\tau^* = \frac{\phi^* - \hat{\phi}}{SE(\phi^*)}$$

   - Sort the output vector such that $\tau_{(1)}^* \leq \cdots \leq \tau_{(B)}^*$.
   - The "*bootstrap approximate*" confidence interval for ϕ is then given by

     $$\left[\hat{\phi} - \tau_{((1-\alpha/2)B)}^* \cdot SE(\hat{\phi});\ \hat{\phi} - \tau_{((\alpha/2)B)}^* \cdot SE(\hat{\phi})\right]$$

     Set $\alpha = 0.05$ and compare this with the normal approximation.
   - Redo the exercise for $T = 30$ and $T = 10000$. Comment on your findings.

Readings

- Kilian and Lütkepohl (2017, Ch. 12)

References

Kilian, Lutz and Helmut Lütkepohl (2017). *Structural Vector Autoregressive Analysis*. Themes in Modern Econometrics. Cambridge: Cambridge University Press. ISBN: 978-1-107-19657-5. URL: https://doi.org/10.1017/9781108164818.

Lütkepohl, Helmut (2004). "Univariate Time Series Analysis". In: *Applied Time Series Econometrics*. Ed. by Helmut Lütkepohl and Markus Krätzig. 1st ed. Cambridge University Press, pp. 8–85. ISBN: 978-0-521-83919-8 978-0-521-54787-1 978-0-511-60688-5. URL: https://doi.org/10.1017/CBO9780511606885.003.

A. Solutions

1 Solution to Information Criteria For AR(p)

1. The first term measures the fit of a model with order p and it is the same for all criteria. This term decreases for increasing order because there is no correction for degrees of freedom in the ML variance estimator.

   The second term penalizes large AR orders. How this term is effectively chosen distinguishes the different criteria.

   The criteria have the following properties:

   - AIC asymptotically overestimates the order with positive probability
   - HQC estimates the order consistently ($plim\ \hat{p} = p$)
   - SIC is even strongly consistent ($\hat{p} \overset{a.s.}{\to} p$) under quite general conditions (e.g. if the actual DGP is a finite-order AR process and the maximum order is larger than the true order).

   In practice, one often relies on the AIC as having too many lags is not as severe as having too few lags. This is especially true in small samples.

   It is important to notice, that for correctly computing the information criteria one needs to use the same sample size across models with different orders. In other words, the number of pre-sample values set aside for estimation is determined by the maximum order $p^{\max}$.

2. progs/matlab/lagOrderSelectionARp.m

```
function nlag = lagOrderSelectionARp(y,const,pmax,crit)
% function nlag = lagOrderSelectionARp(y,const,pmax,crit)
% ------------------------------------------------------------------------
% Perform and display lag order selection tests for AR(p) model, i.e.
% Akaike, Schwartz and Hannan-Quinn information criteria
% ------------------------------------------------------------------------
% INPUTS
%   - y     : data vector [periods x 1]
%   - const : 1 constant, 2 constant+linear trend. [scalar]
%   - pmax  : number of maximum lags to consider. [scalar]
%   - crit  : criteria to compute lag order selection;
%             possible values: 'AIC', 'SIC', 'HQC'
% ------------------------------------------------------------------------
% OUTPUTS
%   - nlag  : number of lags recommended by the selected information crit
% ------------------------------------------------------------------------
% Willi Mutschler, November 2, 2021
% willi@mutschler.eu
% ------------------------------------------------------------------------

%% Construct regressor matrix and dependent variable
% number of presample values set aside for estimation is determined by pmax
T = size(y,1);      % sample size
T_eff = T-pmax;     % effective sample size used for all estimations, i.e.
                    % number of presample values set aside for estimation
                    % is determined by the maximum order pmax
Y = lagmatrix(y,1:pmax);
y = y((pmax+1):T,:);
YMAX = Y((pmax+1):T,:);
if const == 1 % constant
```

```
    YMAX = [ones(T_eff,1) YMAX];
elseif const == 2 % constant and time trend
    YMAX = [ones(T_eff,1) transpose((pmax+1):T) YMAX];
end

%% Compute information criteria
INFO_CRIT = nan(pmax,1);         % initialize
for p=1:pmax
    n = const+p;                 % number of freely estimated parameters
    YY = YMAX(:,1:n);            % data used in estimation
        thetahat = (YY'*YY)\(YY'*y); % OLS and ML estimator
    uhat = y - YY*thetahat;      % both OLS and ML residuals
    sigma2u = uhat'*uhat/T_eff;  % ML estimate of variance of errors
    %sigma2u = uhat'*uhat/(T_eff-n);  % OLS estimate of variance of errors

    if strcmp(crit,'AIC')     % Akaike
        INFO_CRIT(p,:) = log(sigma2u) + 2/T_eff*n;
    elseif strcmp(crit,'SIC') % Schwartz
        INFO_CRIT(p,:) = log(sigma2u) + log(T_eff)/T_eff*n;
    elseif strcmp(crit,'HQC') % Hannan-Quinn
        INFO_CRIT(p,:) = log(sigma2u) + 2*log(log(T_eff))/T_eff*n;
    end
end

%% Store results and find minimal value of criteria
results = [transpose(1:pmax) INFO_CRIT];
nlag = find(INFO_CRIT == min(INFO_CRIT));
[~,idx] = sort(results(:,2));
results = results(idx,:);

%% Display summary of results
fprintf('************************************************************\n');
fprintf('*** OPTIMAL ENDOGENOUS LAGS FROM %s INFORMATION CRITERIA ***\n',crit);
fprintf('************************************************************\n');
disp(array2table(results,'VariableNames',{'Lag',crit}))
fprintf('  Optimal number of lags (searched up to %d lags):\n',pmax);
fprintf('  %s Info Criterion:    %d\n',crit,nlag);
fprintf('\n');
```

3./4. progs/matlab/AR4LagSelection.m

```
% ------------------------------------------------------------------------
% Determine lag order of AR(4) model with constant on simulated data using
% information criteria
% ------------------------------------------------------------------------
% Willi Mutschler, November 16, 2021
% willi@mutschler.eu
% ------------------------------------------------------------------------

clearvars; clc; close all;

AR4 = importdata('../../data/AR4.csv');
nlag_AIC = lagOrderSelectionARp(AR4.data,1,8,'AIC');
```

```
nlag_SIC = lagOrderSelectionARp(AR4.data,1,8,'SIC');
nlag_HQC = lagOrderSelectionARp(AR4.data,1,8,'HQC');

c = 1; phi=[0.51; -0.1; 0.06; -0.22]; sigma=0.03; p=4; T=200+p;
y=zeros(T,1);
for t=(p+1):T
    y(t)= c + phi(1)*y(t-1) + phi(2)*y(t-2) + phi(3)*y(t-3) + phi(4)*y(t-4) +
        randn()*sigma;
end
y = y(51:200); % get rid of initial 50 observations (so-called burnin phase in
    simulations)
nlag_AIC = lagOrderSelectionARp(y,1,8,'AIC');
nlag_SIC = lagOrderSelectionARp(y,1,8,'SIC');
nlag_HQC = lagOrderSelectionARp(y,1,8,'HQC');
```

2 Solution to Portmanteau Test For Residual Autocorrelation

progs/matlab/portmanteauTest.m

```
% ------------------------------------------------------------------------
% Comparison of Portmanteau Tests For Residual Autocorrelation on inflation
% series for (i) AR(phat) where phat is determined by the AIC criteria and
% (ii) AR(1) model
% ------------------------------------------------------------------------
% Willi Mutschler, January 2018
% willi@mutschler.eu
% ------------------------------------------------------------------------

clearvars; clc; close all;
gnpdeflator = importdata('../../data/gnpdeflator.csv');
% computation of  inflation series
price_index = gnpdeflator.data(:,3);
infl = log(price_index(2:end,:))- log(price_index(1:(end-1),:));

pmax = 12;                                      % set maximum number of lags
const = 0;                                      % include constant?
alph = 0.05;                                    % significance level
phat = lagOrderSelectionARp(infl,const,pmax,'AIC'); % compute criteria

OLSAR_phat = ARpOLS(infl,phat,const,alph);    % estimate AR(phat)
OLSAR_1    = ARpOLS(infl,1,const,alph);       % estimate AR(1)

% Compute portmanteu statistic
h = phat+10;          % maximum number of lags
u_p = OLSAR_phat.resid;
u_1 = OLSAR_1.resid;
T_p = size(u_p,1);
T_1 = size(u_1,1);
% initialize output vectors
rho_p = nan(1,h);
rho_1 = nan(1,h);
% compute variances
gam_p = 1/T_p*(u_p' *u_p);
gam_1 = 1/T_1*(u_1' *u_1);
% compute autocorrelations
for j=1:h
    rho_p(1,j) = 1/((T_p-j)*gam_p)*(u_p(1+j:T_p,:)'*u_p(1:T_p-j,:));
    rho_1(1,j) = 1/((T_1-j)*gam_1)*(u_1(1+j:T_1,:)'*u_1(1:T_1-j,:));
end
% compute test statistic
Q_p = T_p*sum(rho_p.*rho_p);
Q_1 = T_1*sum(rho_1.*rho_1);

% compute critical values and p values from chi2 distribution
Qpcrit_phat = chi2inv(1-0.05,h-phat);
Qpcrit_1 = chi2inv(1-0.05,h-1);
Qpval_phat = chi2cdf(Q_p,h-phat, "upper");
Qpval_1 = chi2cdf(Q_1,h-1, "upper");

```

```
% compare critical values and p values for AR(phat) and AR(1) model
fprintf('\nPORTMANTEAU TEST\n')
fprintf('H0: No remaining residual autocorrelation\n')
fprintf('\nTest Statistic > Critical Value\n')
disp(array2table([Q_p>Qpcrit_phat Q_1>Qpcrit_1],'VariableNames',{'AR(phat)','AR(1)'},'
    RowNames',{'Reject H0'}));
fprintf('\np-values\n')
disp(array2table([Qpval_phat Qpval_1],'VariableNames',{'AR(phat)','AR(1)'},'RowNames'
    ,{'p value of test'}));
```

The Null hypothesis of **no remaining residual autocorrelation** can be rejected for the $AR(1)$ model, but not for the $AR(\hat{p})$ model. This implies that the residuals in the $AR(1)$ model are NOT white noise, while the residuals in the $AR(\hat{p})$ model are LIKELY to be white noise. In sum, both the information criteria as well as the Portmanteau test favor the $AR(\hat{p})$ model over the $AR(1)$ model.

3 Solution to Bootstrap Confidence Interval For AR(1) Coefficient

progs/matlab/bootstrapCIAR1.m

```
% ------------------------------------------------------------------------
% Percentile-t-Bootstrap confidence interval for the AR(1) coefficient
% compared to the asymptotic confidence interval
% ------------------------------------------------------------------------
% Willi Mutschler, November 14, 2023
% willi@mutschler.eu
% ------------------------------------------------------------------------

clearvars; clc; close all;

% Set options
B = 10000;    % number of bootstrap repetitions (experiment with different values!)
T = 100;      % set sample size (experiment with different values!)
burnin = 100; % number of observations to discard
alph = 0.05;  % confidence level

% generate true data from AR(1)
u = exprnd(1,T+burnin,1)-1; % draw from exponential distribution
                            % and subtract expectation to get E(u)=0
y = nan(T+burnin,1);        % initialize data vector
c = 1;                      % AR(1) constant
phi = 0.8;                  % AR(1) coefficient
y(1) = c/(1-phi);           % initialize with mean
for t=2:(T+burnin)
    y(t) = c + phi*y(t-1) + u(t); % generate from AR(1)
end
y = y(burnin+1:end); % discard burnin observations

% OLS estimation and t-statistic on true data
OLS        = ARpOLS(y,1,1,alph);
uhat       = OLS.resid;
chat       = OLS.thetahat(1);
phihat     = OLS.thetahat(2);
sig_phihat = OLS.sd_thetahat(2);
sig_uhat   = OLS.siguhat;
tau        = phihat/sig_phihat;

% Percentile-t-Bootstrap
taustar_parametric     = nan(B,1); % initialize t statistics output vector
taustar_non_parametric = nan(B,1); % initialize t statistics output vector

for b=1:B
    % draw with replacement
    ustar_parametric = randn(length(uhat),1).*sig_uhat;
    ustar_non_parametric = datasample(uhat,length(uhat),'Replace',true); % requires
        statistics toolbox
    ystar_parametric     = nan(T,1);  % initialize artificial data vectors
    ystar_non_parametric = nan(T,1);  % initialize artificial data vectors
    ystar_parametric(1,:)     = y(1); % intialize the first observation with real data
    ystar_non_parametric(1,:) = y(1); % intialize the first observation with real data
```

```
    for t=2:size(y,1)
        % generate artificial data from AR(1)
        ystar_parametric(t,1)     = chat + phihat*ystar_parametric(t-1)     +
            ustar_parametric(t-1,1);
        ystar_non_parametric(t,1) = chat + phihat*ystar_non_parametric(t-1) +
            ustar_non_parametric(t-1,1);
    end
    % same OLS estimation and t-statistic on artificial data
    OLSstar_parametric          = ARpOLS(ystar_parametric,1,1,alph);
    OLSstar_non_parametric      = ARpOLS(ystar_non_parametric,1,1,alph);
    phistar_parametric          = OLSstar_parametric.thetahat(2);
    phistar_non_parametric      = OLSstar_non_parametric.thetahat(2);
    sig_phistar_parametric      = OLSstar_parametric.sd_thetahat(2);
    sig_phistar_non_parametric  = OLSstar_non_parametric.sd_thetahat(2);
    taustar_parametric(b,1)     = (phistar_parametric-phihat)/sig_phistar_parametric;
    taustar_non_parametric(b,1) = (phistar_non_parametric-phihat)/
        sig_phistar_non_parametric;
end
% bootstrap distribution
taustar_parametric        = sort(taustar_parametric);                          %
    sort output vector to access quantiles
taustar_non_parametric    = sort(taustar_non_parametric);                      %
    sort output vector to access quantiles
Lower_Boot_parametric     = phihat-taustar_parametric((1-alph/2)*B)*sig_phihat;     %
    lower bound for bootstrap CI (parametric)
Upper_Boot_parametric     = phihat-taustar_parametric(    alph/2*B)*sig_phihat;     %
    upper bound for bootstrap CI (parametric)
Lower_Boot_non_parametric = phihat-taustar_non_parametric((1-alph/2)*B)*sig_phihat; %
    lower bound for bootstrap CI (non-parametric)
Upper_Boot_non_parametric = phihat-taustar_non_parametric(    alph/2*B)*sig_phihat; %
    upper bound for bootstrap CI (non-parametric)
% asymptotic distribution
z = norminv(1-alph/2,0,1);          % 1-alph/2 quanitle of standard normal
    distribution
Lower_Approx = phihat-z*sig_phihat; % lower bound for approx CI
Upper_Approx = phihat+z*sig_phihat; % upper bound for approx CI
table([Lower_Approx;Lower_Boot_parametric;Lower_Boot_non_parametric],...
      [Upper_Approx;Upper_Boot_parametric;Upper_Boot_non_parametric],...
      'RowNames',{'Approx' 'Parametric' 'Non-parametric'},'VariableNames',{'Lower' '
          Upper'})

x = -5:0.1:5;
figure('name','Distribution of taustar');
sgtitle('taustar')
subplot(1,2,1)
  histogram(taustar_parametric,'Normalization','pdf');
  hold on;
  plot(x,normpdf(x,0,1));
  title('Parametric');
  legend('Parametric','Standard Normal')
  hold off;
subplot(1,2,2)
```

```
91  histogram(taustar_non_parametric,'Normalization','pdf');
92  hold on;
93  plot(x,normpdf(x,0,1));
94  title('Non-Parametric');
95  legend('Non-Parametric','Standard Normal')
96  hold off;
```

For large T the bootstrap CI are almost identical to the asymptotic CIs, for small T they are narrower.

No efficiency gain from imposing parametric assumptions even when that assumption is true, whereas imposing the wrong parametric structure tends to undermine the accuracy of the bootstrap inference.

$\hookrightarrow$ Nonparametric approach is typically strictly preferred in practice.