Quantitative Macroeconomics

Winter 2023/24

Week 4

Willi Mutschler
willi@mutschler.eu

Version: 1.0
Latest version available on: GitHub

Contents

1. Ordinary Least Squares Estimation Of AR(p)

Consider an AR(p) model with a constant and linear term:

$$y_t = c + d \cdot t + \phi_1 y_{t-1} + \cdot + \phi_p y_{t-p} + u_t = Y'_{t-1}\theta + u_t$$

where $Y_{t-1} = (1, t, y_{t-1}, \dots, y_{t-p})$ and $u_t \sim WN(0, \sigma_u^2)$. The ordinary least-squares (OLS) estimator of $\theta = (c, d, \phi_1, \dots, \phi_p)$ is

$$\hat{\theta} = \left(\sum_{t=1}^{T} Y_{t-1} Y'_{t-1} \right)^{-1} \sum_{t=1}^{T} Y_{t-1} y_t$$

Under the assumptions of stationarity and other standard regularity conditions one can derive that

$$\sqrt{T}(\hat{\theta} - \theta) \overset{d}{\to} \tilde{U} \sim N \left(0, \sigma_u^2 \; plim \left(T^{-1} \sum_{t=1}^{T} Y_{t-1} Y'_{t-1} \right)^{-1} \right)$$

The residual variance may be estimated consistently by

$$\hat{\sigma}_u^2 = \frac{1}{T - length(\theta)} \sum_{t=1}^{T} \hat{u}_t^2$$

where $\hat{u}_t = y_t - Y'_{t-1}\hat{\theta}$ are the OLS residuals.

1. Write a `function OLS = ARpOLS(`y`,`p`,`$const$`,`α`)` that takes as inputs a data vector y and number of lags p. The input *const* is 1 if there is a constant in the model, 2 if there is a constant and a linear trend. The function outputs a structure OLS, which contains the OLS estimates of θ, its standard errors, t-statistics and p-values given significance value α, as well as the OLS estimate of σ_u.

2. Load simulated data for an AR(4) model given in the CSV file `AR4.csv`. Estimate an AR(4) model with a constant term using your `ARpOLS` function.

Readings

- Lütkepohl (2004)

2. Maximum Likelihood Estimation Of Gaussian AR(p)

Consider an AR(p) model with a constant and linear trend:

$$y_t = c + d \cdot t + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + u_t = Y_{t-1}\theta + u_t$$

where $Y_{t-1} = (1, t, y_{t-1}, \ldots, y_{t-p})$ is the matrix of regressors, $\theta = (c, d, \phi_1, \ldots, \phi_p)$ the parameter vector and the error terms u_t are white noise and normally distributed, i.e. $u_t \sim N(0, \sigma_u^2)$ and $E[u_t u_s] = 0$ for $t \neq s$. If the sample distribution is known to have probability density function $f(y_1, \ldots, y_T)$, an estimation with Maximum Likelihood (ML) is possible. To this end, we decompose the joint distribution by

$$f(y_1, \ldots, y_T|\theta, \sigma_u^2) = f_1(y_1|\theta, \sigma_u^2) \times f_2(y_2|y_1, \theta, \sigma_u^2) \times \cdots \times f_T(y_T|y_{T-1}, \ldots, y_1, \theta, \sigma_u^2)$$

Then the log-likelihood is

$$\log f(y_1, \ldots, y_T|\theta, \sigma_u^2) = \sum_{t=1}^{T} \log f_t(y_t|y_{t-1}, \ldots, y_1, \theta, \sigma_u^2)$$

Let's denote the values that maximize the log-likelihood as $\tilde{\theta}$ and $\tilde{\sigma}_u^2$. ML estimators have (under general assumptions) an asymptotic normal distribution

$$\sqrt{T}\begin{pmatrix} \tilde{\theta} - \theta \\ \tilde{\sigma}_u^2 - \sigma_u^2 \end{pmatrix} \xrightarrow{d} U \sim N(0, I_a(\theta, \sigma_u^2)^{-1})$$

where $I_a(\theta, \sigma_u)$ is the asymptotic information matrix. Recall that the asymptotic information matrix is the limit of minus the expectation of the Hessian of the log-likelihood divided by the sample size.

$$I_a(\theta, \sigma_u^2) = \lim_{T\to\infty} -\frac{1}{T} E \begin{pmatrix} \frac{\partial^2 \log l}{\partial \theta^2} & \frac{\partial^2 \log l}{\partial \theta \partial \sigma_u^2} \\ \frac{\partial^2 \log l}{\partial \sigma_u^2 \partial \theta} & \frac{\partial^2 \log l}{\partial (\sigma_u^2)^2} \end{pmatrix}$$

1. First consider the case of $p = 1$
   a) Derive the exact log-likelihood function for the $AR(1)$ model with $|\theta| < 1$ and $d = 0$:

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

   b) Why do we often look at the log-likelihood function instead of the actual likelihood function?
   c) Regard the value of the first observation as deterministic or, equivalently, note that its contribution to the log-likelihood disappears asymptotically. Maximize analytically the conditional log-likelihood to get the ML estimators for θ and σ_u. Compare these to the OLS estimators.

2. Now consider the general $AR(p)$ model.
   a) Write a function `logLikeARpNorm`$(x,y,p,const)$ that computes the value of the log-likelihood conditional on the first p observations of a Gaussian $AR(p)$ model, i.e.

   $$\log l(\theta, \sigma_u) = -\frac{T-p}{2}\log(2\pi) - \frac{T-p}{2}\log(\sigma_u^2) - \sum_{t=p+1}^{T} \frac{u_t^2}{2\sigma_u^2}$$

   where $x = (\theta', \sigma_u)'$, y denotes the data vector, p the number of lags and $const$ is equal to 1 if there is a constant, and equal to 2 if there is a constant and linear trend in the model.
   b) Write a `function ML = ARpML`$(y,p,const,\alpha)$ that takes as inputs a data vector y, number of lags p and $const = 1$ if the model has a constant term or $const = 2$ if the model has a constant term and linear trend. α denotes the significance level. The function computes

- the maximum likelihood estimates of an AR(p) model by numerically minimizing the negative conditional log-likelihood function using e.g. `fminunc`
- the standard errors by means of the asymptotic covariance matrix, i.e. the inverse of the hessian of the negative log-likelihood function (hint: gradient-based optimizers also output the hessian)

Save all results into a structure "ML" containing the estimates of θ, its standard errors, t-statistics and p-values as well as the ML estimate of σ_u.

c) Load simulated data given in the CSV file `AR4.csv` and estimate an AR(4) model with constant term. Compare your results with the OLS estimators from the previous exercise.

Readings

- Lütkepohl (2004).

3. Maximum Likelihood Estimation Of Laplace AR(p)

Consider the AR(1) model with constant

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

Assume that the error terms u_t are i.i.d. Laplace distributed with known density

$$f_{u_t}(u) = \frac{1}{2} \exp\left(-|u|\right)$$

Note that for the above parametrization of the Laplace distribution we have that $E(u_t) = 0$ and $Var(u_t) = 2$, so we are only interested in estimating c and ϕ and not the standard deviation of u_t as it is fixed.

1. Derive the log-likelihood function conditional on the first observation.
2. Write a function that calculates the conditional log-likelihood of c and ϕ.
3. Load the dataset given in the CSV file `LaPlace.csv`. Numerically find the maximum likelihood estimates of c and ϕ by minimizing the negative conditional log-likelihood function.
4. Compare your results with the maximum likelihood estimate under the assumption of Gaussianity. That is, redo the estimation by minimizing the negative Gaussian log-likelihood function.

Readings

- Lütkepohl (2004)

References

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 Ordinary Least Squares Estimation Of AR(p)

1.

progs/matlab/ARpOLS.m

```matlab
function OLS = ARpOLS(y,p,const,alph)
% OLS = ARpOLS(y,p,const,alph)
% -----------------------------------------------------------------------
% OLS regression of AR(p) model:
% y_t = c + d*t + theta_1*y_{t-1} + ... + theta_p*y_{t-p} + u_t
% with white noise u_t ~ (0,sigma_u)
% -----------------------------------------------------------------------
% INPUT
%   - y       [Tx1]     data vector of dimension T
%   - p       [scalar]  number of lags
%   - const   [scalar]  1 constant; 2 constant and linear trend in model
%       - alpha   [scalar]  significance level for t statistic and p value
% -----------------------------------------------------------------------
% OUTPUT
%       - OLS: structure including estimation results
%     - T_eff          [scalar]       effective sample size used in estimation
%     - thetahat       [(const+p)x1]  estimate of coefficients
%     - sd_thetahat    [(const+p)x1]  estimate of standard error of coefficients
%     - tstat          [(const+p)x1]  t statistics
%     - pvalues        [(const+p)x1]  p values of H_0: thetahat = 0
%     - siguhat        [scalar]       estimate of standard deviation of error term u
%     - theta_ci       [(const+p)x2]  (1-alph)% confidence intervall for theta given significance level alph
%     - resid          [T_eff x 1]    residuals
% -----------------------------------------------------------------------
% Calls
% - lagmatrix.m (Econometrics Toolbox)
% -----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% -----------------------------------------------------------------------

T = size(y,1);              % sample size
T_eff = T-p;                % effective sample size used in estimation
Y = lagmatrix(y,1:p);       % create matrix with lagged variables
if const==1                 % add constant term
    Y = [ones(T,1) Y];
elseif const==2             % add constant term and time trend
    Y = [ones(T,1) transpose(1:T) Y];
end
Y = Y((p+1):end,:);         % get rid of initial p observations
y = y(p+1:end);             % get rid of initial p observations

YtYinv = inv(Y'*Y);
thetahat = YtYinv*(Y'*y); % OLS estimator of coefficients
yhat = Y*thetahat;          % predicted values
uhat = y - yhat;            % residuals
```

```
utu = uhat'*uhat;         % sum of squared residuals

var_uhat = utu/(T_eff-p-const);         % variance of error term
siguhat = sqrt(var_uhat);               % standard deviaiont of error term
var_thetahat = diag(var_uhat*(YtYinv)); % variance of coefficients
sd_thetahat = sqrt(var_thetahat);       % standard error of coefficients

tstat = thetahat./sd_thetahat;          % t-statistics
tcrit = -tinv(alph/2,T_eff-p-const);    % critical value
pvalues = tpdf(tstat,T_eff-p-const);    % p-value
% confidence interval
theta_ci=[thetahat-tcrit.*sd_thetahat, thetahat+tcrit.*sd_thetahat];

% Store into output structure
OLS.T_eff        = T_eff;
OLS.thetahat     = thetahat;
OLS.siguhat      = siguhat;
OLS.sd_thetahat  = sd_thetahat;
OLS.tstat        = tstat;
OLS.pvalues      = pvalues;
OLS.theta_ci     = theta_ci;
OLS.resid        = uhat;
end
```

progs/matlab/AR4OLS.m

```
% -----------------------------------------------------------------------
% Estimation of AR(4) model with constant using OLS on simulated data
% -----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% -----------------------------------------------------------------------

% Housekeeping
clearvars; clc; close all;
% load data
AR4 = importdata("../../data/AR4.csv");
y = AR4.data;                       % vector with data
p = 4;                              % set number of lags
const = 1;                          % model with constant
alph = 0.05;                        % significance level
OLS = ARpOLS(y,p,const,alph);       % estimate model using ARpOLS function

% Display results and compare to true values
TrueVals = [1; 0.51; -0.1; 0.06; -0.22; 0.5];
result = table([OLS.thetahat;OLS.siguhat],TrueVals);
result.Properties.VariableNames = {'OLS_Estimate','True_Values'};
result.Properties.RowNames = {'c','\phi_1','\phi_2','\phi_3','\phi_4','\sigma_u'};
disp(result)
```

2. **2 Solution to Maximum Likelihood Estimation Of Gaussian AR(p)**

1. Let's first consider the AR(1) model

a) The first observation y_1 is a random variable with mean and variance equal to:

$$E[y_1] = \mu = \frac{c}{1-\theta} \text{ and } Var[y_1] = \frac{\sigma_u^2}{1-\theta^2}$$

Since the errors are Gaussian, y_1 is also Gaussian, i.e. $y_1 \sim N\left(\frac{c}{1-\theta}, \frac{\sigma_u^2}{1-\theta^2}\right)$. The pdf is thus:

$$f_1(y_1|\theta, \sigma_u^2) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma_u^2/(1-\theta^2)}} \exp\left\{-\frac{1}{2}\frac{[y_1 - (c/(1-\theta))]^2}{\sigma_u^2/(1-\theta^2)}\right\}$$

The second observation y_2 conditional on y_1 is given by $y_2 = c + \theta y_1 + u_2$. Conditional on y_1, y_2 is thus the sum of a deterministic term $(c+\theta y_1)$ and the $N(0, \sigma_u^2)$ variable u_2. Hence:

$$y_2|y_1 \sim N(c + \theta y_1, \sigma_u^2)$$

and the pdf is given by:

$$f_2(y_2|y_1, \theta, \sigma_u^2) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left\{-\frac{1}{2}\frac{[y_2 - c - \theta y_1]^2}{\sigma_u^2}\right\}$$

The joint density of observations 1 and 2 is then just:

$$f(y_2, y_1|\theta, \sigma_u^2) = f_2(y_2|y_1, \theta, \sigma_u^2) \cdot f_1(y_1|\theta, \sigma_u^2)$$

In general the value of $y_1, y_2, ..., y_{t-1}$ matter for y_t only through the value y_{t-1} and the density of observation t conditional on the preceding $t-1$ observations is given by

$$f_t(y_t|y_{t-1}, \theta, \sigma_u^2) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left\{-\frac{1}{2}\frac{[y_t - c - \theta y_{t-1}]^2}{\sigma_u^2}\right\}$$

The likelihood of the complete sample can thus be calculated as:

$$f(y_T, y_{T-1}, ..., y_1|\theta, \sigma_u^2) = f_1(y_1|\theta, \sigma_u^2) \cdot \prod_{t=2}^{T} f_t(y_t|y_{t-1}, \theta, \sigma_u^2)$$

The log-likelihood is therefore

$$\begin{aligned}
\log l(\theta, \sigma_u^2) &= \log f_1(y_1|\theta, \sigma_u^2) + \sum_{t=2}^{T} \log f_t(y_t|y_{t-1}, \theta, \sigma_u^2) \\
&= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_u^2/(1-\theta^2)) - \frac{(y_1 - (c/(1-\theta))^2)}{2\sigma_u^2/(1-\theta^2)} \\
&- ((T-1)/2)\log(2\pi) - ((T-1)/2)\log(\sigma_u^2) - \sum_{t=2}^{T} \frac{(y_t - c - \theta y_{t-1})^2}{2\sigma_u^2}
\end{aligned}$$

b) Theoretically it does not matter whether we consider the log-likelihood or the actual likelihood function, as the value that maximizes the likelihood also maximize the log-likelihood, because the log is a monotone transformation. However, it is usually easier to work with sums instead of products theoretically, e.g. the LLN or CT are based on sums. Computationally, working with products is typically impossible as the resulting values very quickly surpass machine precision (they quickly go to $\pm\infty$); working with sums does not have this problem. So from a computational perspective we will exclusively work with the log-likelihood function.

c) Discarding the first observation, the conditional log-likelihood is given by:

$$\log l^c(\theta, \sigma_u^2) = -((T-1)/2)\log(2\pi) - ((T-1)/2)\log(\sigma_u^2) - \sum_{t=2}^{T} \frac{(y_t - c - \theta y_{t-1})^2}{2\sigma_u^2}$$
$$= -((T-1)/2)\log(2\pi) - ((T-1)/2)\log(\sigma_u^2) - \sum_{t=2}^{T} \frac{u_t^2}{2\sigma_u^2}$$

Note that the first two sums do not depend on θ; thus, when maximizing $\log l^c(\theta, \sigma_u^2)$ with respect to θ, we are basically minimizing the squared residuals, which will simply yield the OLS estimator. The estimator for the variance, however, is different, as we are dividing the sum of sqared residuals ($\sum_{t=2}^{T} u_t^2$) by $T^{eff} = (T-1)$ when doing ML instead of by $T^{eff} - 1$ when doing OLS. Obviously, for large T this does not matter.

progs/matlab/logLikeARpNorm.m

```matlab
function loglik=logLikeARpNorm(x,y,p,const)
% loglik=logLikeARpNorm(x,y,p,const)
% -------------------------------------------------------------------------
% Computes the conditional log likelihood function of Gaussian AR(p) model:
% y_t = c + d*t + theta_1*y_{t-1} + ... + theta_p*y_{t-p} + u_t
% with u_t ~ N(0,sig_u)
% -------------------------------------------------------------------------
% INPUT
%   - x      [(const+p+1)x1]  vector of coefficients, i.e. [c,d,theta_1,...,theta_p,sig_u]'
%   - y      [Tx1]            data vector of dimension T
%   - p      [scalar]         number of lags
%   - const  [scalar]         1 constant; 2 constant and linear trend in model
% -------------------------------------------------------------------------
% OUTPUT
%       - loglik [double]          value of Gaussian log-likelihood
% -------------------------------------------------------------------------
% Calls
%   - lagmatrix.m (requires Econometrics Toolbox)
% -------------------------------------------------------------------------
% Willi Mutschler, November 14, 2023
% willi@mutschler.eu
% -------------------------------------------------------------------------
penalizedLikelihood = -1e10; % very small number to penalize likelihood, e.g. -Inf

theta = x(1:(const+p)); % [c d theta']
sig_u = x(const+p+1);   % standard deviation of error term sig_u
% make sure sig_u is positive
if sig_u <= 0
    loglik = penalizedLikelihood;
    return                  % this ends the current function call
else
    T = size(y,1);          % sample size
    Y = lagmatrix(y,1:p);   % create matrix with lagged variables
    if const == 1           % add constant
        Y = [ones(T,1) Y];
    elseif const == 2       % add constant and time trend
```

```matlab
        Y = [ones(T,1) transpose(1:T) Y];
    end
    Y = Y((p+1):end,:);      % get rid of initial observations
    y = y(p+1:end);          % get rid of initial observations

    uhat = y - Y*theta;      % ML residuals
    utu = uhat'*uhat;        % ML sum of residuals

    % compute the conditional log likelihood
    loglik = -log(2*pi)*(T-p)/2 - log(sig_u^2)*(T-p)/2 -utu/(2*sig_u^2);
end

% if anything goes wrong set value very small
if isnan(loglik) || isinf(loglik) || ~isreal(loglik)
    loglik = penalizedLikelihood;
end

end % function end
```

3. progs/matlab/ARpML.m

```matlab
function ML = ARpML(y,p,const,alph)
% ML = ARpML(y,p,const,alph)
% -----------------------------------------------------------------------
% Maximum Likelihood Estimation of Gaussian AR(p) model:
% y_t = c + d*t + theta_1*y_{t-1} + ... + theta_p*y_{t-p} + u_t
% with u_t ~ iid N(0,sig_u)
% -----------------------------------------------------------------------
% INPUTS
%       - y       [Tx1]      dependent variable vector
%   - p       [scalar]   number of lags
%   - const   [scalar]   0 no constant; 1 constant; 2 constant and linear trend
%   - alph    [scalar]   significance level for t statistic
% -----------------------------------------------------------------------
% OUTPUT
%       - ML: structure including estimation results
%     - T_eff            [scalar]       effective sample size used in estimation
%     - thetatilde       [(const+p)x1]  estimate of coefficients
%     - sd_thetatilde    [(const+p)x1]  estimate of standard error of coefficients
%     - tstat            [(const+p)x1]  t statistics given alph as significance level
%     - pvalues          [(const+p)x1]  p values of H_0: thetahat = 0
%     - sigutilde        [scalar]       estimate of standard deviation of error term u
%     - sd_sigutilde     [scalar]       estimate of standard error of standard deviation of error term u
%     - theta_ci         [(const+p)x2]  (1-alph)% confidence intervall for theta given significance level alph
%     - logl             [double]       value of maximized log likelihood
% -----------------------------------------------------------------------
% CALLS
%    - logLikeARpNorm.m    : Computes log likelihood function of Gaussian AR(p)
```

```
%    - hessian_numerical.m : Computes second order partial derivatives using numerical differentiation
% ----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% ----------------------------------------------------------------------

T = size(y,1);           % sample size

% Optimization with fminunc which finds the minimum of negative log-likelihood
f = @(x) -1*logLikeARpNorm(x,y,p,const); % use function handle to hand over additional parameters and multiply by -1 for negative log-likelihood
% start values
x0 = randn(p+const+1,1); % randomize
x0(end) = abs(x0(end));  % make sure initial sig_u is positive

[x,fval,exitflag,output,grad,hess] = fminunc(f,x0);
% alternatively use hessian_numerical.m that does two-sided finite difference computation of hessian
% [x,fval] = fminunc(f,x0);
hess = reshape(hessian_numerical(f,x),length(x),length(x)); %alternatively use output argument of fminunc

thetatilde = x(1:p+const);          % estimated coefficient values
sigutilde = x(end);                 % estimated standard daviation of error term
V = inv(hess);                      % estimated covariance matrix of coefficients and (log of) standard deviation of error
sd = sqrt(diag(V));                 % estimated standard error vector
sd_thetatilde = sd(1:p+const);      % estimated standard errors of coefficients
sd_sigutilde = sd(end);             % estimated standard error of standard deviation of error term
T_eff = T-p;                        % effective sample size used in estimation
logl  = -fval;                      % value of maximized log likelihood
tstat = thetatilde./sd_thetatilde;  % t-statistic
tcrit = -tinv(alph/2,T_eff-p);      % critical value from t-distribution
pvalues = tpdf(tstat,T_eff-p);      % p-values from t-distribution

% confidence interval for coefficients given signficance level alph
theta_ci=[thetatilde-tcrit.*sd_thetatilde, thetatilde+tcrit.*sd_thetatilde];

% Store into output structure
ML.T_eff          = T_eff;
ML.logl           = logl;
ML.thetatilde     = thetatilde;
ML.sigutilde      = sigutilde;
ML.sd_thetatilde  = sd_thetatilde;
ML.sd_sigutilde   = sd_thetatilde;
ML.tstat          = tstat;
ML.pvalues        = pvalues;
ML.theta_ci       = theta_ci;

end %function end
```

progs/matlab/AR4ML.m

```matlab
% ----------------------------------------------------------------------
% Estimation of AR(4) model with constant using ML on simulated data
% ----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% ----------------------------------------------------------------------

% Housekeeping
clearvars; clc; close all;
% load data
AR4 = importdata("../../data/AR4.csv");
y = AR4.data;
p = 4;                              % set number of lags
const = 1;                          % model with constant
alph = 0.05;                        % significance level
ML = ARpML(y,p,const,alph);         % estimate model using ARpML function

% Display results and compare to true values
TrueVals = [1; 0.51; -0.1; 0.06; -0.22; 0.5];
result = table([ML.thetatilde;ML.sigutilde],TrueVals);
result.Properties.VariableNames = {'ML_Estimate','True_Values'};
result.Properties.RowNames = {'c','\phi_1','\phi_2','\phi_3','\phi_4','\sigma_u'};
disp(result)

% Compare to OLS estimates
OLS = ARpOLS(y,p,const,alph);
disp([TrueVals [OLS.thetahat; OLS.siguhat] [ML.thetatilde; ML.sigutilde]]);
```

The estimates for the coefficients are the same, but slightly different for the standard deviation of the error term.

4. 3 Solution to Maximum Likelihood Estimation Of Laplace AR(p)

1. Computation of the conditional expectation and variance:
   - $E[y_t|y_{t-1}] = c + \phi y_{t-1}$
   - $Var[y_t|y_{t-1}] = var(u_t) = 2$

   Hence the conditional density is

$$f_t(y_t|y_{t-1}; c, \phi) = \frac{1}{2} \cdot e^{-|y_t-(c+\phi y_{t-1})|} = \frac{1}{2} \cdot e^{-|u_t|}$$

   The conditional log-likelihood function is therefore given by

$$\log L(y_2, \ldots, y_T; c, \phi) = -(T-1) \cdot \log(2) - \sum_{t=2}^{T} |u_t|$$

2. progs/matlab/logLikeARpLaplace.m

```matlab
function loglik=logLikeARpLaplace(x,y,p,const)
% loglik=logLikeARpLaplace(x,y,p,const)
% -----------------------------------------------------------------------
% Computes the conditional log likelihood function of Laplace AR(p) model:
% y_t = c + d*t + theta_1*y_{t-1} + ... + theta_p*y_{t-p} + u_t
% with u_t ~ Laplace distributed with E(u_t)=0, Var(u_t)=2 (known variance)
% -----------------------------------------------------------------------
% INPUT
%   - x      [(const+p)x1]  vector of coefficients, i.e. [c,d,theta_1,...,theta_p
    ]'
%   - y      [Tx1]          data vector of dimension T
%   - p      [scalar]       number of lags
%   - const  [scalar]       1 constant; 2 constant and linear trend in model
% -----------------------------------------------------------------------
% OUTPUT
%       - loglik  [double]      value of Laplace log-likelihood
% -----------------------------------------------------------------------
% Calls
%   - lagmatrix.m (requires Econometrics Toolbox)
% -----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% -----------------------------------------------------------------------

theta = x;                % AR coefficients
T = size(y,1);            % sample size

Y = lagmatrix(y,1:p);     % create matrix with lagged variables
if const == 1             % add constant
    Y = [ones(T,1) Y];
elseif const == 2         % add constant and time trend
    Y = [ones(T,1) transpose(1:T) Y];
end
Y = Y((p+1):end,:);       % get rid of initial observations
y = y(p+1:end);           % get rid of initial observations

```

```
uhat = y - Y*theta;     % ML residuals

% compute the conditional log likelihood
loglik = -log(2)*(T-p) -sum(abs(uhat));

if isnan(loglik) || isinf(loglik) || ~isreal(loglik)
    loglik = -1e10;     % if anything goes wrong set value very small, can also use -Inf
end

end % function end
```

progs/matlab/ARpMLLaPlace.m

```
function ML = ARpMLLaplace(y,p,const,alph)
% ML = ARpMLLaplace(y,p,const,alph)
% -----------------------------------------------------------------------
% Maximum Likelihood estimation of Laplace AR(p) model:
% y_t = c + d*t + theta_1*y_{t-1} + ... + theta_p*y_{t-p} + u_t
% with u_t ~ Laplace distributed with E(u_t)=0, Var(u_t)=2
% -----------------------------------------------------------------------
% INPUTS
%   - y       [Tx1]     dependent variable vector
%   - p       [scalar]  number of lags
%   - const   [scalar]  0 no constant; 1 constant; 2 constant and linear trend
%   - alph    [scalar]  significance level for t statistic
% -----------------------------------------------------------------------
% OUTPUT
%       - ML: structure including estimation results
%     - T_eff            [scalar]       effective sample size used in estimation
%     - thetatilde       [(const+p)x1]  estimate of coefficients
%     - sd_thetatilde    [(const+p)x1]  estimate of standard error of coefficients
%     - tstat            [(const+p)x1]  t statistics given alph as significance level
%     - pvalues          [(const+p)x1]  p values of H_0: thetahat = 0
%     - theta_ci         [(const+p)x2]  (1-alph)% confidence intervall for theta given significance level alph
%     - logl             [double]       value of maximized log likelihood
% -----------------------------------------------------------------------
% CALLS
%    - logLikeARpLaPlace   : Computes log likelihood function of Laplace AR(p)
%    - hessian_numerical.m : Computes second order partial derivatives using numerical differentiation
% -----------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% -----------------------------------------------------------------------

T = size(y,1);                          % sample size
x0 = randn(p+const,1);                  % randomize start values, note that sig_u is known
%x0 = [1;0.8]; %true values
% Optimization with fminunc which finds the minimum of negative log-likelihood
```

```
fun = @(x)-1*logLikeARpLaplace(x,y,p,const); % use function handle to hand over
                                            % additional parameters to negative
                                            % of logLikeARpLaplace
%[x,fval,exitflag,output,grad,hessian] = fminunc(fun,x0); % the hessian might be badly shaped, better use hessian_numerical.m
[x,fval] = fminunc(fun,x0);
hess = reshape(hessian_numerical(fun,x),length(x),length(x));

thetatilde = x;                     % estimated coefficient values
V = inv(hess);                      % estimated covariance matrix of coefficients
sd_thetatilde = sqrt(diag(V));      % estimated standard error of coefficients
T_eff = T-p;                        % effective sample size used in estimation
logl  = -fval;                      % value of maximized log likelihood
tstat = thetatilde./sd_thetatilde;  % t-statistic
tcrit = -tinv(alph/2,T_eff-p);      % critical value from t-distribution
pvalues = tpdf(tstat,T_eff-p);      % p-values from t-distribution

% confidence interval given signficance level alph
theta_ci=[thetatilde-tcrit.*sd_thetatilde, thetatilde+tcrit.*sd_thetatilde];

% Store into output structure
ML.T_eff          = T_eff;
ML.logl           = logl;
ML.thetatilde     = thetatilde;
ML.sd_thetatilde = sd_thetatilde;
ML.tstat          = tstat;
ML.pvalues        = pvalues;
ML.theta_ci       = theta_ci;

end %function end
```

progs/matlab/AR1MLLaPlace.m

```
% -------------------------------------------------------------------------
% Estimate Laplace AR(1) model with constant and known variance of errors
% with Maximum Likelihood on simulated data
% -------------------------------------------------------------------------
% Willi Mutschler, November 9, 2022
% willi@mutschler.eu
% -------------------------------------------------------------------------

clearvars; clc; close all;                  % housekeeping
y = importdata('../../data/LaPlace.csv');  % load data
y = y.data;                                 % focus on numerical values
p = 1;                                      % set number of lags
const = 1;                                  % model with constant
alph = 0.05;                                % significance level
MLLaPlace = ARpMLLaplace(y,p,const,alph);  % estimate model using ARpMLLaplace

% Display results and compare to true values
TrueVals = [1; 0.8];
result = table(TrueVals,MLLaPlace.thetatilde,MLLaPlace.sd_thetatilde);
result.Properties.VariableNames = {'True_Values','ML_Estimate','STD_ERR'};
```

```
result.Properties.RowNames = {'c','\phi'};
disp(result)

% Compare to Gaussian ML estimates
MLGaussian = ARpML(y,p,const,alph); % estimate model using ARpML
disp([TrueVals MLLaPlace.thetatilde MLGaussian.thetatilde]);
disp([MLLaPlace.sd_thetatilde MLGaussian.sd_thetatilde]);
```

3. Note that the values are very close to each other. Maximizing the Gaussian likelihood, even though the underlying distribution is not Gaussian, is also known as pseudo-maximum likelihood or quasi-maximum likelihood. It usually performs surprisingly well if you cannot pin down the underlying distribution.