The Lagreg package

O. V. Komashko

February 15, 2018

v. 1.0

Abstract

This package is intended to provide a toolbox for regressions with lagged variables, including ARIMA-modelling. Additionally, it contains a tool for automatized analysis of ex-post forecast accuracy for several univariate time series forecasting methods.

Contents

1 Description 2
  1.1 Model selection .......................... 2
    1.1.1 ARMA($X$) .......................... 2
    1.1.2 ARIMA($X$) ......................... 3
    1.1.3 Lag selection in unrestricted distributed lag model .... 3
  1.2 Almon lags (PDL) ....................... 3
  1.3 ARDL models: multipliers and their characteristics ........ 4

2 List of public functions 6
  2.1 Model selection .......................... 6
    2.1.1 auto_arima .......................... 6
    2.1.2 auto_lags ............................ 8
  2.2 Almon lags (PDL) ....................... 8
    2.2.1 pdl .................................. 8
    2.2.2 fcast_pdl ................................ 10
    2.2.3 test_pdl ................................ 10
  2.3 Multipliers and their characteristics ..................... 11
    2.3.1 irf_l .................................. 11
    2.3.2 ltm .................................... 12
    2.3.3 ardl_3 .................................. 12

References 13

Appendix 14

News in v. 1.0 ................................ 14
Versions numbering .......................... 14
Miscellaneous ............................... 14
1 Description

The package provides the following facilities:

- auto-selection of general seasonal ARIMA(\(X\)) models based on different information criteria; the (seasonal) difference order is selected via (seasonal) unit root tests;
- auto-selection of lag length in unrestricted distributed lags models, accounting for autocorrelation; information criteria are used to this end;
- estimation of Almon lag models (PDL) with arbitrary number of pdl terms and zero restrictions at ends; supplied post-estimation facilities include forecasting and a battery of tests;
- computing (cumulative) impulse response functions with standard errors and confidence intervals after the following models:
  - autoregressive distributed lags (ARDL), estimated by ols, or tsls (including --liml and --gmm options);
  - unrestricted distributed lags models, estimated by ols, tsls, or arima;
  - PDL models;
  s.e. and c.i. are computed using delta-method, for ARDL models bootstrapping is also available; plotting is supplied;
- computing long-term multipliers for the models listed in the previous item; for ARDL models the mean lag and the median lag are computed;

1.1 Model selection

1.1.1 ARMA(\(X\))

The auto_arima function is essentially a wrapper over the native arima p d q \{; P D Q\}; depvar \{ indepvars \} command. If "arma" flag is on, d = 0, and D = 0. It finds (seasonal) ARMA(\(X\)) models with minimal AIC, BIC, HQC, or the corrected AIC (\(AIC_c\)); arima p q ; P Q models are estimated with their parameters taken from the following triangles: p and q are such that p + q ≤ max(p + q) (the default is 2), and P + Q ≤ max(P + Q) (the default is 1 for ARMA and 0 for ARMAX). The values of all information criteria are \(gretl\) native ones, except for \(AIC_c\) which is computed as \(^1\)

\[
AIC_c = AIC + \frac{2(p + q + P + Q + k + 1)(p + q + P + Q + k + 2)}{T - p - q - P - Q - k - 2},
\]

where k is the number of exogenous regressors including constant, and T is the (initial) sample size (\(gretl\)’s \(\$T\)). The initial sample size for each model is corrected so as all the models are estimated using samples of equal size.

It is supposed that the order of difference for the series is defined beforehand.

Essentially, auto_arima + "arma" syntax is intended for applying to models with exogenous regressors.

\(^1\)Hyndman and Athanasopoulos (2012), Section 8.6
1.1.2 \textit{ARIMA}(X)

The \texttt{auto_arima} function can deal with general \textit{ARIMA}(X) models. The difference order of the series is selected using KPSS unit root test while the seasonal difference order is selected based on OCSB test.

Essentially, it is intended for applying to models without exogenous regressors.

It is supposed that the selected model will be subsequently estimated by the native \texttt{arima} command to use its rich post-estimation menu.

1.1.3 Lag selection in unrestricted distributed lag model

The \texttt{auto_lags} function is gretl adaptation and "functionalization" of the approach, presented in Hyndman and Athanasopoulos [2012], Section 9.1, Example 9.3. For each lag length less or equal to user-specified \texttt{maxlag} parameter optimal ARMA model with possibly autocorrelated errors is chosen based on AIC, BIC, HQC, or AICc (non seasonal \texttt{auto_arima} with "arma" and \( p + q \leq 2 \) are used inside). Finally, chosen models for each lag length are compared, and the minimal IC model indicates the lag length. It is suggested that the selected model should be estimated afterwards using native \texttt{arima}, or \texttt{ols} commands.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figures/1985_2000_forecasts.png}
\caption{Forecasts plot, produced by \texttt{fcast.pdl}}
\end{figure}

1.2 Almon lags (PDL)

The \texttt{pdl} function estimates the Almon lag, or PDL model\footnote{Almon (1965)}. It can deal with arbitrary number of PDL terms and with full set of end-point restrictions. A part of its code is based on Cottrell (2015).

For the case with no end-point restrictions the function makes the standard PDL data transformation, then it calls \texttt{ols}.\footnote{Almon (1965)}
Provided end-point restrictions are applied, it transforms the restrictions to the form suitable to call the `restrict` block with `--full` option. The post-estimation menu (provided by `test_pdl`) is relied upon `ols` in the first case and upon `restrict` block in the second one. The `fcast_pdl` function corresponds to `fcast --out-of-sample` used after e.g., `ols` estimated model. Figure 1 shows a forecasts plot produced by `fcast_pdl`.

![Figure 1: Cumulative irf plot, produced by `fcast_pdl`.](image1)

### 1.3 ARDL models: multipliers and their characteristics

We consider simple univariate autoregressive distributed lag models of the form

\[ \Gamma(L)y_t = \alpha_0 + B(L)x_t + z'_t\alpha_t + \varepsilon_t \]  

(1)

where \( \Gamma(L) \) and \( B(L) \) are polynomials in lag operator, the constant term in the former is 1. Variables in \( z' \) are treated as exogenous. The long-term multiplier of \( x \) on \( y \) is \( \Gamma(1)/B(1) \). The elements of the impulse response function (irf) of \( y \) with respect to \( x \), \( \delta_i \), are determined as coefficients in the Maclaurin series expansion of \( \Gamma(L)/B(L) \):

\[ \Gamma(u)/B(u) = \sum_{i=0}^{\infty} \delta_i u^i \]  

(2)

Under the well-known conditions for consistent estimation, gretl routinely deals with model (1) using `ols`, or, in the presence of autocorrelation (hence, endogeneity), using `tsls`. It means, all we need is supplying facilities to compute irf’s, long-term multipliers, mean, and median lags, provided the model estimates are obtained via `gretl` standard means.

The long-term multiplier and its standard errors are computed via `ltm` function.
The \texttt{irf.l} function computes the irf and the cumulative irf for model (1). Figure 2 illustrates irf plots produced by \texttt{irf.l}.

The mean lag makes sense provided no $\delta_i$’s have different signs. The definition is

$$\text{meanlag} = \frac{\sum_{i=0}^{\infty} i\delta_i}{\sum_{i=0}^{\infty} \delta_i}$$

From equation (2) we have:

$$\left. \frac{d}{du} \left( \frac{\Gamma(u)}{B(u)} \right) \right|_{u=1} = \sum_{i=0}^{\infty} i\delta_i$$

Hence,

$$\text{meanlag} = \left. \frac{d}{du} \left( \frac{\Gamma(u)}{B(u)} \right) \right|_{u=1} / \left( \frac{\Gamma(1)}{B(1)} \right)$$

(3)

The \texttt{ardl.3} function computes mean lag, based on equation (3). It is also outputs median lag and long-term multiplier.
2 List of public functions

2.1 Model selection

2.1.1 auto_arima

auto_arima (series y, string allspec[null], list x[null])

Return type: bundle

This function finds (seasonal) ARIMA$(X)$ models with minimal $AIC$, $BIC$, $HQC$, or the corrected $AIC$ ($AIC_c$). Difference order ($d$), and seasonal difference order ($D$) can be inputted, or auto-selected based on KPSS test and OCSB test correspondingly. Then it search on the set of $ARIMA(p, d, q; P, D, Q)$ models with theirs parameters taken from the following triangles: $p$ and $q$ are such that $p + q \leq \max(p + q)$ (the default is 2), and $P + Q \leq \max(P + Q)$. The function arguments are:

1. series y: the dependent variable;

2. string allspec[null]: miscellaneous parameters; allspec has the following structure:
   
   allspec = "arma, ft, x = regressors, mpq = integer,\n   mPQ = integer, d = integer, D = integer, --options"

   Each component can be missed; they can appear in arbitrary order.

Components:

- arma: if present, only ARMA$(X)$ are considered; gives the same effect as setting $d = 0$ and $D = 0$, see below;

- ft: governs the use of the constant in models with $d = 0$ and no exogenous regressors, see rules to deal with constant below;

- x = regressors: names of exogenous variables, e.g. x = const x, see two ways to input exogenous regressors below;

- d = integer: ARIMA difference order, e.g. $d = 1$; if missing, is selected using KPSS test at 0.05;

- D = integer: ARIMA seasonal difference order, e.g. $D = 1$; if missing, is selected using OCSB test at 0.05; instead of setting $d = 0$ and $D = 0$ we can use arma flag to consider ARMA models only;

- mpq = integer: defines triangle for searching $p$ and $q$; is set to 2 if missing;

- mPQ = integer: defines triangle for searching $P$ and $Q$; if missing, it is set to 1 for pure ARIMA models and is set to 0 for ARIMAX: this mean that by default non-seasonal models are considered in the presence of exogenous regressors; the reason is as follows it is common to study dependencies using seasonally adjusted data;

- --options: options to pass to arima, e.g.

   --x-12-arima --y-diff-only; to suppress printing, add --quiet (-q) the same way as it is used in native gretl commands;
3. list x[null]: independent variables (optional);

rules to deal with constant
for ARIMA: if \(d = 0\) (inputted, or detected) constant is included; to exclude it use `auto_arima(d, d, lg); "d = 0, --nc")
if \(d = 1\) (inputted, or detected) constant is excluded; to include constant use 'ft' flag: auto_arima(lg, "D = 1, ft")

for ARIMAX: as for the native arima command, to exclude/include constant use `auto_arima(y, "x = x1 x2")/auto_arima(y, "x = 0 x1 x2")

two ways to input exogenous regressors
- auto_arima(y, "arma, x = 0 x1 x2")
- list x = 0 x1 x2
  auto_arima(y, "arma", x)

  If list argument to auto_arima is inputted; \(x = \ldots\)' slot of allspec parameter is ignored; a user can choose the most convenient way for herself

A typical usage without exogenous regressors is

bundle baari = auto_arima(y)
string command = baari.aiccmodel
@command

A typical usage for ARMAX is

bundle baarx = auto_arima(y, "arma", xlist)
string command = baarx.aiccmodel
@command

Output bundle (b) contents

b.aicmod (string): AIC-selected model as quoted full-syntax arima/ols command, e.g. "arima 0 1 1; 0 1 1; lg --nc"

b.bicmod (string): BIC-selected model

b.hqcmod (string): HQC-selected model

b.aiccmod (string): \(AIC_c\)-selected model

b.ic (matrix): example output:

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>P</th>
<th>Q</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>aic</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-483.39</td>
</tr>
<tr>
<td>bic</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-474.77</td>
</tr>
<tr>
<td>hqc</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-479.89</td>
</tr>
<tr>
<td>aicc</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1.0000</td>
<td>-483.30</td>
</tr>
</tbody>
</table>

b.table (matrix): traces results for all models in triangles described at the top of the entry.
2.1.2 auto_lags

auto_lags(series y, series x, list z[null], scalar maxlag[NA], bool isx12[0])

Return type: bundle
Auto-selection of lag length in unrestricted distributed lags models, accounting for autocorrelation based on minimization of information criterion (AIC, BIC, HQC, or AICc).
The function arguments are:

1. series y: the dependent variable
2. series x: the regressor to select lag length
3. list z[null]: possible other regressors including const
4. scalar maxlag[NA]: maximal lag length for x, NA → xmax(2,$pd); for example, if maxlag = 4, models with 4, 3, 2, 1, and 0 (i.e. only x itself is included into the list of regressors) lags of x are considered. For each of these 5 regressors specifications optimal autocorrelation structure is detected using auto_arima. At this stage we would have 5 models. The last step is select lag length which corresponds to model with minimal IC.
5. bool isx12[0]: whether to call arima inside with "--x-12-arima" option.

Output bundle (b) contents:
models (aicmod, bicmod, hqcmod, aiccmod): these bundle elements are strings which represents the selected models, e.g. "ols d_u 0 g(0 to -2)", or "arima 1 0; d_u 0 g(0 to -2)" the four elements corresponds to 4 information criteria mentioned above;
icpq: 4 by 4 matrix, its contents should be clear from the following example:

<table>
<thead>
<tr>
<th></th>
<th>lag</th>
<th>ic</th>
<th>p</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>aic</td>
<td>2.0000</td>
<td>-41.047</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>bic</td>
<td>2.0000</td>
<td>-29.137</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>hqc</td>
<td>2.0000</td>
<td>-36.272</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>aicc</td>
<td>2.0000</td>
<td>-40.514</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

2.2 Almon lags (PDL)

2.2.1 pdl

pdl(list y, list x, matrix pqe, list X[null], string opt[null])

Return type: bundle
Estimates PDL model with arbitrary number of pdl terms. End-point restrictions can be imposed for left, right, or both ends.
The function arguments are:

1. **list y**: the dependent variable (it should contain only one variable); list is applied for output bundle to have ylist element to make better correspondence with **ols** extractors;

2. **list x**: regressor(s) to have **PDL** lags;

3. **matrix pqe**: matrix of **PDL** specifications, one row for each element of x; each row has 3 elements: maximal lag(p), polynomial order(q), end restrictions(e).
   
   Examples:
   - \( \text{pqe} = \{4, 2, 0\} \): **PDL** with 4 lags, polynomial order is 2, no end restrictions;
   - \( \text{pqe} = \{4, 2\} \rightarrow \text{pqe} = \{4, 2, 0\} \);
   - \( \text{pqe} = \{4, 2, 1\} \): the same, but with zero restriction at the left end;
   - \( \text{pqe}[3] = 2 \) (or \( \text{pqe[i, 3]} = 2 \)): is for the right end zero restriction;
   - \( \text{pqe}[3] = 3 \) (or \( \text{pqe[i, 3]} = 3 \)): is for zero restrictions at the both ends;
   - \( \text{list x} = x1 \ x2 \), \( \text{pqe} = \{4, 2, 1; 5, 3, 0\} \) is for regression with **PDL** terms in two variables: \( x1 \) and \( x2 \);

4. **list X[null]**: other regressors (optional); to run pdl-regression with constant term **constant** should be included into \( X \) explicitly as it is with \( \text{xlist} \) used in **ols** command;

5. **string opt[null]**: options to pass to **ols/restrict --full**, e.g. **--quiet**

**Output bundle (b) contents:**

- **extractors**: for example \( \text{b.lnl} \) and \( \text{b.uhat} \) mean the same as \( \$\text{lnl} \) and \( \$\text{uhat} \), correspondingly, etc, etc;

- **supplementary**: used as input to \( \text{fcast.pdl} \) and \( \text{test.pdl} \)

**Post-estimation:**

- **testing restrictions on model coefficients and confidence intervals for functions of coefficients**: it can easily done using \( \text{waldTest.gfn} \) package Komashko (2017b);

- **diagnostic checks**: \( \text{test.pdl} \) is used to this end;

- **out of sample forecasting**: it is realized by \( \text{fcast.pdl} \)
2.2.2 fcast_pdl

fcast_pdl(bundle *bu, int h[1::1], scalar cl[NA], bool plt[1],
           bool prn[1])

Return type: matrix
The function arguments are:

1. bundle *bu: estimated by pdl PDL model bundle;
2. int h[1::1]: horizon;
3. scalar cl[NA]: confidence level, NA → 0.9;
4. bool plt[1]: whether to plot forecasts
5. bool prn[1]: whether to print forecasts

Output matrix contains point forecasts, its standard error, and confidence bounds.
Figure 1 shows a sample plot produced by fcast_pdl.
Example:
#include lagreg.gfn
open green51.gdt --quiet
series g = 100*diff(realgdp)/realgdp
series d_u = diff(unemp)
bundle b = pdl(d_u,g,{4,2,2},const,"-q")
t2 = $t2
dataset addobs 2
g[t2+1] = 0.3
g[t2+2] = 0.4
fcast_pdl(&b,2)

2.2.3 test_pdl

test_pdl(bundle *bu, string test[null])

Return type: matrix
The function arguments are:

1. bundle *bu: estimated by pdl PDL model bundle;
2. string test[null]: test(s) specifications;
   • modtest: let we run
     ols y xlist
     modtest --autocorr 3
     to do the same for PDL model we need
     bundle b = pdl(...)
     test_pdl(&b, "--autocorr 3")
See `modtest` entry in Gretl Command reference for the list of all available tests.

- **test** = "reset|qlrtest|cusum|normtest|chow [--options]" : example: `test = "reset --squares-only --silent"`; see the relevant entries in Gretl Command reference for options lists. Details:
  - `normtest`: since the model residuals series is what to test, the name is omitted, e.g. `test = "normtest --swilk"
  - `chow`: to use `test = "chow dummyvar --dummy"` one needs to create global series `dummyvar` beforehand; its values will be read by `test.pdl` from the global dataset by name

- **test** = null: perform a battery of tests (--autocorr, --normality, --white, --white-nocross, reset, reset --squares-only)

Outputs a matrix of the following form: `{$test, $pvalue}`, or empty matrix if test = null.

### 2.3 Multipliers and their characteristics

#### 2.3.1 `irf_l`

`irf_l(bundle *b, string opt[null])`

Return type: matrix
The function arguments are:

1. **bundle *b**: estimated ARDL, PDL, or unrestricted distributed lag model bundle (b = `$model`, if `ols`, `arima`, or `tsls` was used to estimate dl model; for PDL models `b` is output of `pdl`);

2. **string opt[null]**: miscellaneous options; the structure is as follows:
   ```
   opt = "cum, noplot, cl = value, x = name, se = boot,\n   lags = integer, repl = integer"
   ```

Each of the seven slots can be omitted. Slots can be arbitrary sorted.

- **cum**: cumulative irf is computed if present, ordinary irf if missing;
- **noplot**: no plot is produced if present;
- **cl = value**: confidence level, e.g. "cl = 0.99", cl is set to 0.9 if missing;
- **x = name**: name of shock variable, the left-most admissible regressor is chosen if missing;
- **se = boot**: for ARDL only; bootstrapped/delta-method standard errors; `se = boot` is the same as `repl = 1000`
- **lags = integer**: for ARDL only; number of lags to compute and plot (for finite lag models all lags are dealt with); if missing, 10 lags is used;
- **repl = integer**: for ARDL only; number of bootstrap replications; if present, `se = boot` is not needed
Example (for ARDL):
opt = "cum, cl = 0.99, x = unemp, lags = 20, repl = 2000"
This should produce cumulative irf for 20 lags with bootstrapped standard errors and 99% confidence bounds based on 2000 replications; the shock variable is supposed to be named "unemp".

Output matrix contains the long term multiplier, its standard error, and confidence bounds.
A sample plot produced by irf_l is shown at Figure 2.

2.3.2 ltm

\[ \text{ltm}(\text{bundle}\ *b, \text{string}\ \text{xnam}[\text{null}]), \text{scalar}\ \text{cl}[\text{NA}] \]  
Return type: matrix  
The function arguments are:  
1. \text{bundle}\ *b: estimated ARDL, PDL, or unrestricted distributed lag model bundle (b = \$model, if ols, arima, or tsls was used to estimate dl model; for PDL models b is output of pdl);  
2. \text{string}\ \text{xnam}[\text{null}]: name of shock variable regressor; if missing the left-most admissible regressor is chosen;  
3. \text{scalar}\ \text{cl}[\text{NA}]: confidence level, NA \rightarrow 0.95  
Output matrix contains the long term multiplier, its standard error, and confidence bounds.

2.3.3 ardl_3

\[ \text{ardl}_3(\text{bundle}\ *b, \text{string}\ \text{xnam}[\text{null}]) \]  
Return type: matrix  
The function arguments are:  
1. \text{bundle}\ *b: estimated ARDL model bundle (b = \$model);  
2. \text{string}\ \text{xnam}[\text{null}]: name of shock variable regressor; if missing the left-most admissible regressor is chosen;  
Returns 3 by 1 matrix; its elements are long term multiplier, mean lag, and median lag.
References


Appendix

News in v. 1.0

New features:

- improved time performance
- intuitive syntax
- `auto_arima` selects general ARIMA models; it uses KPSS test do detect unit roots, and OCSB test to detect seasonal unit roots
- `irf_l` computes and plots (c)irfs for ARDL, PDL, and unrestricted distributed lag models; for all models above it computes delta-method standard errors (as in previous package versions, for ARDL bootstrapped standard errors and confidence bounds can be computed)
- new post-estimation functions for PDL models:
  - `fcast_pdl` computes ant plots forecasts
  - `test_pdl` performs (almost) all tests that can be performed after `ols` command
- `ltm` finds long term multipliers with s.e. and c.i. for all dl models mentioned above

Deprecations and replacements:

- `auto_arma` → `auto_arima` (with "arma" option)
- `m_almon_reg` → `pdl`
- `{ardl_irf_ci, ardl_cirf_ci, ardl_irf}` → `irf_l`
- `ardl_memelag` → `ardl_3`
- `irf_plot` → `irf_l` (it produces plots if "nplot" is not indicated)
- `fcast_acc` has moved to `tsfcst.gfn` package \(^3\) where it is named as `fcast_ac`

`auto_lags` has not been renamed/moved; it has changed syntax only.

Versions numbering

The essential numbering is as follows: `integer.digit`, e.g. 0.7, or 1.0. Subsequent digits carry only technical information on reuploading a version after revision. If you see v. 1.02 on the server, it would mean it was reuploaded twice after some corrections.

In the case of large modifications "jumps" are possible, e.g. 0.7 → 1.0.

Miscellaneous

\(^3\)Komashko (2017a)
About the author:
He is fond of his cat, see Figure A1.

Figure A1: Kosha (*Felis silvestris catus bigener*)