The lagreg package

O. V. Komashko

April 2, 2016

Abstract

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

Contents

1 Description

The package provides the following facilities:

- auto-selection of seasonal $ARMA$ models based on different information criteria;
- auto-selection of lag length in unrestricted distributed lags models, accounting for autocorrelation; information criteria are used to this end;

- Almon lag models (*PDL*) with arbitrary number of pdl terms and zero restrictions at ends;
- computing (cumulative) impulse response functions after autoregressive distributed lags (*ARDL*), estimated by `ols`, or `tsls` (including `--liml` and `--gmm` options); bootstrapping confidence intervals and plotting are also supplied;
- automatized accessing of forecast accuracy for sequential forecasts based on shifting informational base for several models, such as `arima p d q; P D Q; y [options]`, or `ols y 0 time seasonals() [options]`.

1.1 Seasonal $ARMA(X)$ model selection

The `auto_arma` function is essentially a wrapper over the native `arima p d q [ ; P D Q ]; depvar [ indepvars ]` command, calling it with `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` is called with its parameters taken from the following triangles: `p` and `q` are such that `p + q` $\leqslant$ `max(p+q)` (the default is 2), and `P + Q` $\leqslant$ `max(P+Q)` (the default is 1). The values of all information criteria are `gretl` native ones, except for AIC_c which is computed as[1]

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

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 of the series is defined beforehand using existing `gretl` commands or function packages.

The function is intentionally supplied with "scanty" interface, since it is supposed that the selected model will be subsequently estimated by the native `arima` command to use its rich post-estimation menu.

1.2 Lag selection in unrestricted distributed lag model

The `auto_lags` 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 `maxlag` optimal $ARMA$ model for possible autocorrelated errors is chosen via the same criteria (non seasonal `auto_arma` with `p + q` $\leqslant$ `2` is used inside). It is suggested that the selected model should be estimated afterwards using the native `arima` command.

1.3 Almon lags

The `m_almon_reg` function for estimation of the Almon lag model[2] is conceived as a supplement to `almonreg` package[3] by Allin Cottrell: while `m_almon_reg` has

[1] Hyndman and Athanasopoulos [2012], Section 8.6
[2] Almon [1965]
[3] Cottrell [2015]

no GUI, which is supplied with `almonreg`, it can deal with arbitrary number of PDL terms and with full set of end-point restrictions. A part of the package uses modified code by Allin Cottrell[4].

For the case with no end-point restrictions the function makes the standard PDL data transformation, then it calls `ols`.

Provided end-point restrictions, it transforms the restrictions to the form suitable to call `restrict` block with the `--full` option. The post-estimation menu is relied upon `ols` in the first case and upon `restrict` block in the second one.

1.4 ARDL models

We consider simple univariate autoregressive distributed lag models of the form

$$\Gamma(L)y_t = \alpha_0 + \mathrm{B}(L)x_t + \boldsymbol{z}_t'\boldsymbol{\alpha} + \varepsilon_t \ , \tag{1}$$

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

$$\frac{\mathrm{B}(u)}{\Gamma(u)} = \sum_{i=0}^{\infty} \delta_i u^i \tag{2}$$

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 `nlConfint` from `waldTest` package[5].

The `ardl_irf_ci` and `ardl_cirf_ci` functions compute the irf and the cumulative irf for model (1), respectively. The model (1) is estimated inside of these functions using `ols`, or `tsls` if the list of instruments is inputted to obtain the estimates of the coefficients of $\mathrm{B}(L)$ and $\Gamma(L)$. The recursive algorithm to compute irf's is based on the source code in Pfaff [2008]. It was tested against Wolfram Alpha$^{\mathrm{TM}}$. After irf's are computed, bootstrapping confidence intervals becomes a routine exercise in applying `resample`.

To find irf, we need to subset coefficients on $\Gamma(L)$ and $\mathrm{B}(L)$ from the vector of estimated model parameters (`$coeff`). This procedure is automatized by using several string functions, including additional private functions from the package. For the subsetting to be correct, all lags should have only `gretl` standard names for lags, i.e. lags of `somevar` should be `somevar_1`, etc.

The mean lag makes sense provided no δ_i's have different signs. The usual

[4] *ibid.*

[5] Komashko [2015]

definition is

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

From equation (2) we have:

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

Hence,

$$meanlag = \frac{d}{du}\left(\frac{\mathrm{B}(u)}{\Gamma(u)}\right)\Big|_{u=1} \Big/ \left(\frac{\mathrm{B}(1)}{\Gamma(1)}\right) \tag{3}$$

The `ardl_memelag` function computes median lags, based on equation (3). It uses a private function to compute the derivative in (3) analytically. Besides, this function returns median lag and the long term multiplier.

1.5 Ex-post forecast accuracy assessment

The `fcast_acc` function is intended to be used alongside with the native `gretl` estimation commands provided the following conditions:

- the estimated model is univariate;
- the `fcast` methods are available for the estimated models;
- out-of-sample values from the right-hand side of the model equation are supplied by `gretl` automatically, as for `arima`, or `ols` with only trend and the seasonal dummies as regressors.

The work of the function splits naturally into two parts: (i) obtaining a vector of errors; (ii) calling `fcstats` to output accuracy measures. Further details are given in Subsection 2.5.

2 List of public functions

2.1 auto_arma

```
auto_arma(series y, list x[null],int pq[1::2], int PQ[0::1],
          string opt[null],bool prn[0],bool prntest[1])
```

Return type: `matrix`

This is a very nice function[6]. The function arguments are:

1. `series y`: the dependent variable;
2. `list x[null]`: independent variables (optional);
3. `int pq[1::2]`: max(p+q), where p and q are nonseasonal ARMA parameters;
4. `int int PQ[0::1]`: max(P+Q), where P and Q are seasonal ARMA parameters; if it is set to zero only nonseasonal models are considered;
5. `string opt[null]`: option(s) to pass for `arima`, e.g. "`--x-12-arima`"
6. `bool prn[0]`: whether to print criteria for all models; the default is "no";
7. `bool prntest[1]`: whether to print Ljung-Box Q' for the best models; the default is "yes".

The function finds (seasonal) $ARMA(X)$ models with minimal AIC, BIC, HQC, and AIC_c within the set of models such that `p + q` $\leqslant$ `pq` and `P + Q` $\leqslant$ `PQ`. Output is a matrix with 4 rows, corresponding to 4 criteria. A row consist of $ARMA$ specification: `p q P Q` (`p q` only for non- seasonal models) and the value of corresponding criterion.

2.2 auto_lags

```
auto_lags(scalar maxlag, series y, series x, int crit[1:4:4],
          list z[null], bool cons[1])
```

Return type: `matrix`

The function arguments are:

1. `scalar maxlag`: maximal lag order of lagged x;
2. `series y`: the dependent variable;
3. `series x`: the regressor to select the length of lags of;

[6] The author has absolutely no strength of will to delete this sentence, inherited from the template file.

4. `int crit[1:4:4]`: information criterion to minimize; can be AIC, BIC, HQC, or AIC_c;

5. `list z[null]`: other regressors (optionnal);

6. `bool cons[1]`: whether to include `constant` in regression; the default is "yes".

Selects number of lags in unrestricted distributed lag model via minimizing user-specified information criterion (one of the four options). For each lag length less or equal to `maxlag` optimal $ARMA$ model for possible autocorrelated errors is chosen via the same criterion (non-seasonal `auto_arma` with `p + q` $\leqslant 2$ is used inside).

2.3 m_almon_reg

```
m_almon_reg(series y, list x, matrix pqe, list X[null],
            bool cons[1], string opt[null])
```

Return type: `bundle`

The function arguments are:

1. `series y`: the dependent variable;

2. `list x`: regressors (only names) with PD lags, specified by `pqe`;

3. `matrix pqe`: matrix of PDL specifications: one row for each element of list `x`; the rows are: max lag, pol. order, end restrictions.
   Examples:
   `pqe = {4,2,0}`: PDL with 4 lags, order = 2, no end restrictions;
   `pqe = {4,2,1}`: the same with zero restriction at the left end;
   `pqe[3] = 2` is for the right end zero restriction, and `pqe[3] = 3` is for zero restrictions at the both ends;
   `list x = x1 x2; 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 (optionnal);

5. `bool cons[1]`: whether to include `constant` into regression; the default is "yes";

6. `string opt[null]`: quoted `ols` options e.g. "`--robust`".

Build-in post-estimation options:
With `opt = "--modtest"`, Breusch - Godfrey , normality, White, and RESET tests are performed. You can set several options:
`opt = "--modtest --robust --anova"`. In the case of end restrictions "`--robust`" option is ignored.

External post-estimation options:
Testing of restrictions : `nlwaldtest` from `waldTest`. Forecasting: `predict_y`

from `margeff_el`.

Output bundle content:

`"beta"` vector of all model coefficients in terms of lags of original data.

`"se"` standard errors for all model coefficients

`"vcv"` covariance matrix of all model coefficients. Note, that it doesn't have the full rank, hence we can test multiple restrictions independent in the terms of the coefficients on the transformed data.

`"xnames"` names of regressors with PD lags (strings).

`"starts"` the beginning indexes of coefficients on variables with PDL in the vector of all coefficients. Used for easy subsetting `beta` for plotting irfs.

`"lengths"` the quantity of coefficients on variables with PDL. Used for easy subsetting `beta` for plotting irfs.

`"pqe"` the same as input.

`"gamma"` vector of all model coefficients in terms of transformed variables.

`"uhat"` residuals (series).

`"dwpval"` p-value for the Durbin–Watson test (for the no end-point restrictions case only).

2.4 ARDL

2.4.1 ardl_irf_ci

```
ardl_irf_ci(series y, list x, string xnam, list z[null],
            scalar cl[NA], int nlags[10], int nrepl[1000],
            string tslsopt[null])
```

Return type: `matrix`

The function arguments are:

1. `series y`: the dependent variable;
2. `list x`: the independent variables (including `constant`, if needed);
3. `string xnam`: the name of the regressor with lags;
4. `list z[null]`: instruments (optionnal); including instruments means that `tsls` is used instead of `ols` to estimate the regression;
5. `scalar cl[NA]`: confidence level, `NA = 0.9`;
6. `nlags[10]`: lags number to compute irf

7. `int nrepl[1000]`: the number of bootstrapping replications;
8. `string tslsopt[null]`: `tsls` options, e.g. "`--gmm`";

Returns `(nlags+1)` by 3 matrix, which consists of irf and its confidence bounds.

2.4.2 ardl_cirf_ci

```
ardl_cirf_ci(series y, list x, string xnam, list z[null],
             scalar cl[NA], int nlags[10], int nrepl[1000],
             string tslsopt[null])
```

Return type: `matrix`

The function arguments are:

1. `series y`: the dependent variable;
2. `list x`: the independent variables (including `constant`, if needed);
3. `string xnam`: the name of the regressor with lags;
4. `list z[null]`: instruments (optionnal); including instruments means that `tsls` is used instead of `ols` to estimate the regression;
5. `scalar cl[NA]`: confidence level, `NA = 0.9`;
6. `nlags[10]`: lags number to compute irf
7. `int nrepl[1000]`: the number of bootstrapping replications;
8. `string tslsopt[null]`: `tsls` options, e.g. "`--gmm`";

Returns `(nlags+1)` by 3 matrix, which consists of cumulative irf and its confidence bounds.

2.4.3 ardl_irf

```
ardl_irf(series y, list x, string xnam, list z[null],
         int nlags[10], string tslsopt[null])
```

Return type: `matrix`

The function arguments are:

1. `series y`: the dependent variable;
2. `list x`: the independent variables (including `constant`, if needed);
3. `string xnam`: the name of the regressor with lags;

4. `list z[null]`: instruments (optionnal); including instruments means that `tsls` is used instead of `ols` to estimate the regression;

5. `nlags[10]`: lags number to compute irf

6. `string tslsopt[null]`: `tsls` options, e.g. "`--gmm`";

Returns `(nlags+1)` by 1 matrix, which consists of irf. Cumulative irf can be computed using `cum`.

2.4.4 ardl_memelag

```
ardl_memelag(series y, list x, string xnam, list z[null],
             string tslsopt[null])
```

Return type: `matrix`

The function arguments are:

1. `series y`: the dependent variable;

2. `list x`: the independent variables (including `constant`, if needed);

3. `string xnam`: the name of the regressor with lags;

4. `list z[null]`: instruments (optionnal); including instruments means that `tsls` is used instead of `ols` to estimate the regression;

5. `string tslsopt[null]`: `tsls` options, e.g. "`--gmm`";

Returns 3 by 1 matrix; its elements are long term multiplier, mean lag, and median lag.

2.5 fcast_acc

```
fcast_acc(string cmd, series y, int res[0], int h[1])
```

Return type: `matrix`

The function arguments are:

1. `string cmd`: quoted `gretl` command with arguments and options; it specifies the model used for forecasting, e.g. "`"arima 0 1 1; 1 0 0; y`";

2. `series y`: the dependent variable;

3. `int res[0]`: the number of initially reserved observations, `0 = [T/5]`;

4. `int h[1]`: horizon of forecasting;

Usage (as in the sample script):

```
open bjg.gdt
fcast_acc ("arima 0 1 1; 1 0 0; y", lg, 24, 12)
fcast_acc ("ols y 0 time seasonals()", lg, 24, 12)
```

The function reserves `res` last observations (24 in the example) above, estimates the model specified by `cmd`, computes `h`-steps ahead forecasts (12 in the example); then it adds observations by 1 to the sample for the model estimation; in the last sample of the loop only `h` last observations are reserved.

Finally, it uses `fcstats` to output accuracy measures for each `h`-steps forecast and for all shifting forecasts.

Output is a matrix with `colnames` as specified in the help file for `fcstats`, whereas the `rawnames` are the last observations used for model estimation, and "`total`" in the last row is for overall accuracy measures.

2.6 Plotting

2.6.1 irf_plot

```
irf_plot(matrix m, bool cumul[0], int gnuplotversion[4::5])
```

Return type: `void`

The function arguments are:

1. `matrix m`: matrix with ether one column (irf only), or with three columns (irf and confidence bounds);
2. `bool cumul[0]`: what to plot: cumulative or simple (the default) irf; it is simply an instruction for printing the plot title;
3. `int gnuplotversion[4::5]`: 5 for `gnuplot` 5.x (the default); 4 for `gnuplot` 4.x.

Note that in the case of `m` having 3 columns the confidence levels should be given as its `colnames`, e.g. "`99%`".

References

Rob Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2012. URL `https://www.otexts.org/book/fpp`.

Shirley Almon. The distributed lag between capital appropriations and expenditures. *Econometrica*, 33(1):178–196, 1965.

Allin Cottrell. Gretl function package almonreg, 2015. URL `http://ricardo.ecn.wfu.edu/gretl/cgi-bin/current_fnfiles/almonreg.zip`.

Oleh Komashko. Gretl function package waldTest, 2015. URL `http://ricardo.ecn.wfu.edu/gretl/cgi-bin/current_fnfiles/waldTest.gfn`.

Bernhard Pfaff. Var, svar and svec models: Implementation within R package vars. *Journal of Statistical Software*, 27(4), 2008. URL `http://www.jstatsoft.org/v27/i04/`.