The PTconf package

Sven Schreiber

for gretl, this version: 0.9x (October 2016)

This function package is based on my paper "The estimation uncertainty of permanent-transitory decompositions in co-integrated systems", forthcoming in *Econometric Reviews*, DOI: 10.1080/07474938.2016.1235257, see also: http://www.tandfonline.com/doi/full/10.1080/07474938.2016.1235257. Please cite the paper if you use this package for research.

"PTconf" stands for *P*ermanent-*T*ransitory *Conf*idence.

1 Possible usages

The main usages are demonstrated in the example script that comes with the package (but in a different order).

1.1 Defaults only, no bootstrap (GUI or script)

The first, easiest, and fastest possibility is to use the *PTdefaults()* function. This is also the function that you see (without knowing it) when you execute the *PTconf* package in gretl's graphical interface.[1] This function takes the same arguments as gretl's *vecm* command, namely the lag order p, the cointegration rank r, and an existing list of variables *endo*. It then retrieves both the GG (Gonzalo-Granger) and SWP (Stock-Watson-Proietti) transitory components of all variables (the point estimates in each period) under the names <*nameofvar*>_*GG* and <*nameofvar*>_*SWP*, contained in the gretl list that is returned. Furthermore the limits of asymptotically justified confidence intervals with nominal 95% coverage are also retrieved, under the names <*nameofvar*>_*GGlow*, <*nameofvar*>_*GGup*, <*nameofvar*>_*SWPlow*, <*nameofvar*>_*SWPup*. Bear in mind that in smaller samples (and with an estimated cointegration matrix as it is done here) the true 95% confidence intervals may be much wider than these nominal-asymptotic ones.

[1] Enabling GUI access to the more advanced features described below is on the to-do list for version 1.0.

The next two arguments are optional and only needed when you want to produce the default plots. The switch *plottrans* activates it and at the same time determines whether the GG (1) or SWP (2) components are plotted. The following list *which* may specify a subset of the full variable list. As before, only 95% asymptotic intervals are used in this quick-and-easy usage.

1.2 Full control (script only)

A more advanced but still fairly easy (hopefully!) usage is to specify more options, as follows.

1. You first call the *setupPT()* function, where apart from the (first three) parameters of the *PTdefaults()* function you can enter the number of bootstrap draws (or choose 0 to skip the bootstrap) and the significance level (i.e., 1 minus the desired coverage). You get a gretl bundle back which you have to pass on (in pointer form, e.g. *&myPTb* if you choose the name *myPTb*) as the first argument to each of the following functions.

   (a) Note that only a single significance level (a scalar) can be given to *setupPT()*. If you want to get results for several significance levels at once, you need to change the *sigs* member of the bundle that *setupPT()* returns to a suitable vector, see the example script for a demo.[2]

   (b) Another bundle member that is interesting at this stage is *restrBeta*. If you do nothing, then the cointegration matrix β (dimension $n \times r$) is unrestricted and estimated by the Johansen procedure. If you know β a priori you can specify it by creating/defining *restrBeta*. Again, see the example script for a demo.

2. Then optionally you can call *printPToptions()* to get a summary of what options are stored in the bundle so far.

3. The next step would be to call the *putPTconf()* function which computes the transitory components along with the confidence intervals, including running a bootstrap if that was specified. The result is that the main bundle then contains several arrays of matrices, e.g. when the bundle is called *b*:

   b.amDeltaGG b.amDeltaSWP b.amBootnaiveGG b.amBootnaiveSWP b.amBootHallGG b.amBootHallSWP

[2] Do not run different setups with different scalar significance levels instead, because that would trigger separate bootstraps.

Each of these arrays contains n matrices, one for each variable. Each matrix has $T + p$ rows (the sample size with initial values) and has twice the number of columns as the number of chosen significance levels. For example, for a single significance level 5% the columns would correspond to the 2.5% and 97.5% quantiles; for two significance levels 5% and 10% the columns would refer to: 2.5%, 5%, 95%, 97.5%, always ordered ascending.

If you call *putPTconf()* with the optional integer argument you can reset the number of bootstrap draws that was previously specified. That probably makes sense only when you want to run the same setup several times with a different number of bootstrap replications.[3]

4. To transform the results into series in your current workfile you can use the *getPTtranslist()* function which returns a list. In the *vars* list argument specify which variables you are interested in, and in the string argument *kind* you can include either "GG" or "SWP" or both. The *details* argument ranges from 0 to 2 and controls how much of the settings is included in the names of the produced series. For example, apart from the point estimate detail==0 will give just <*nameofvar*>*_GGup* and <*nameofvar*>_GGlow. What this contains is determined by the *conftype* parameter; 0 for the asymptotic, 1 for the direct bootstrap, or 2 for the Hall bootstrap series. Only the first of the specified significance levels is picked here.

   The choice detail==1 would change this to:
   <*nameofvar*>*_r*<*rank*>*p*<*lagorder*>*GG*<*quantile*>, for example *y_r1p4GG025* for a variable "y" with a single cointegration relation, 4 lags, and the 2.5% quantile of the distribution. Only one of the Hall bootstrap or the asymptotic delta-method series pairs are retrieved, but here for all specified significance levels.

   Finally, detail==2 also adds the suffix *_asymp*, *_bDir*<*bootreps*>, or *_bHall*<*bootreps*> to the variable name, as it retrieves all those variants. If you want to compare the asymptotic method and the bootstrap (without manually going into the arrays of matrices described above, that is), this is the only way.

Now that the series are in your workfile, you can plot them or do anything you want. If you want the permanent component, make use of the additive definition $x_t = x_t^{perma} + x_t^{trans}$ to construct it as the difference of the observed variable and the calculated transitory component (and similarly for the uncertainty of the permanent component).

[3] However, a more efficient way would then probably be to run only one bootstrap with the maximum number of draws, and to analyze a lower number by using only a subset of the draws. Anyway, it's only an *optional* parameter.

Finally, if you want to plot some of the confidence intervals of the transitory components, there is the *plotPTtrans()* function. The optional arguments are: The list argument *which* may specify a subset of the variables, with *transtype* you choose between GG (1) or SWP (2), and *conftype* works as in the *getPTtranslist()* function.

2 Public functions

(To learn about the further private functions, simply check out the source code and the comments therein.)

- setupPT, returns *bundle*
  int p[1::2] "lag order (levels)", int r[1::1] "cointegration rank", const list endo "endog. variables list", int bootreps[0::0] "bootstrap draws (0 to skip)", scalar sig[0:1:0.05] "significance level", int verbosity[0::0] "how much echo and feedback to give"
- printPToptions, returns nothing
  arguments: bundle *b
- putPTconf, returns nothing (adds stuff to the bundle)
  arguments: bundle *b, int newbreps[-1::-1]
- getPTtranslist, returns *list*
  arguments: bundle *b, const list vars, string kind, int details[0:2:0], int conftype[0:2:0]
- PTdefaults, returns *list*
  arguments: int p[1::2] "lag order (levels)", int r[1::1] "cointegration rank", const list endo "endog. variables list", int plottrans[0:2:0] "plot trans. components?" {"none", "GG", "SWP"}, const list which[null] "which variables to plot")
- plotPTtrans, returns nothing
  arguments: bundle *b, const list which[null], int transtype[1:2:2] "GG or SWP?" {"GG", "SWP"}, int conftype[0:2:0] {"delta", "direct bootstrap", "Hall bootstrap"})

3 Limitations and further plans

No exogenous variables yet except the unrestricted constant. This means neither seasonal dummies, nor a linear trend (apart from the induced drift term in the non-stationary directions), nor changing the constant to be restricted.

Partial restrictions of β are not allowed yet.

If you need these features please make yourselves heard; preferably on the gretl mailing list, but a private mail to the address given in the package will also be fine. The more demand there is, the more motivation for additional features...