Requirements for contributed gretl packages: an update and clarification

Allin Cottrell

September 2014

This note presents a statement of the basic requirements for a `gfn` package in good standing, as currently understood. In setting out these criteria I don't mean to "point the finger" at the writers of any packages contributed to date. We were very *laissez faire* at first, and it has only become apparent over time what the sensible requirements are. That said, here's the (short) list.

1 Help text

This must give a clear (if brief) account of what the package does, and also must explain clearly what each parameter does, for each public function, insofar as explanation is reasonably required. (A boolean `verbose` parameter probably doesn't need much if any comment, but most parameters do need comment.)

2 Sample script

This is crucial. The sample script *must* work "out of the box" on all platforms, and *must not* take too long to execute.

The sample script is what curious users are likely to run if they just want to see what a package does and check that it's not broken. It's what we package-checkers want to run for the same reasons, but also in the process of regression-testing new gretl release candidates. It's important that a new gretl release doesn't break existing packages, but we can't assess that if a package's sample script is broken in the first place.

Here are the key things to watch out for in relation to sample scripts:

- **Include yourself**: Right at the top, the sample script must `include` the `gfn` file in question. This will never do any harm, and is needed when the script is run "from scratch", without the package being already loaded. The name of the `gfn` file should be given without any added path, and without quotation marks, as in

  ```
  include mypackage.gfn
  ```

- **Dataset**: If the package requires that a dataset be in place the sample script *must* arrange for this in a portable manner. The options are as follows.

  1. Open a data file that's supplied with the gretl distribution (that is, under the Gretl, Greene or Ramanathan tabs in the built-in datafile browser). But if none of the supplied data files are suitable, then
  2. construct an artificial dataset using the `nulldata` command and gretl's random-number generation facilities, or

3. specify a downloaded data file using the `http` prefix with the `open` command.

In the case of artificial data, the script should include a `set seed` command so that the results are reproducible. In the case of downloaded data the URL should be reasonably stable, not something that's likely to disappear or be moved before long.

In *no case* should a datafile be specified with a full path (`/usr/share/gretl/...` or `C:\Program Files\gretl\...`, or anything of the sort). This is obviously not portable, and is never necessary when opening a supplied data file, given gretl's path-searching capability.

- **Execution time**: Some packages carry out Monte Carlo analyses and/or bootstrapping and we all know that such procedures are inherently time consuming. Nonetheless, a sample script should execute on current hardware in a reasonably short time—preferably less than 15 seconds and certainly less than a minute. Otherwise both casual users and testers will lose patience. If this means that only a "toy" example can be run, that's OK. The author can add comments to the script saying that this is just an illustration, serious use requires many more iterations. And/or one can add a more "realistic" invocation of the function(s), commented out, with a statement such as "Uncomment this for a real test".
- **Commenting out**: In some cases an author may wish to indicate alternative ways of calling his or her package. That's fine, but if an alternative call requires a dataset other than the one opened by the script it must be commented out; we don't want any lines in the sample script that will generate errors when the script is called "as is".

The basic point here is that the intent of the sample script in a gfn package is not just "a rough idea of how you might call this package", or "something that ran OK for the author on some machine at some time", but something that will run for any user of gretl on any platform, without modification, provided only that their gretl installation satisfies the stated version requirement of the package.