Gretl: Retrospect, Design and Prospect

Allin Cottrell

Department of Economics, Wake Forest University.
cottrell@wfu.edu

Abstract. In this paper I will give a brief overview of the history of gretl's development, comment on some issues relating to gretl's overall design, and set out some thoughts on gretl's future.

1 Retrospect

Gretl's "modern history" began in September, 2001, when the gretl code was first imported to CVS at `sourceforge.net`. The program's "pre-history" goes back to the mid-1990s. I will briefly describe this background.

1.1 Econometrics software on DOS and Windows 3.0

I began teaching econometrics at Wake Forest University in 1989. I had taught a related course, Business Statistics, at Elon College in North Carolina over the previous few years, and had tried using various statistical packages—RATS and PcGive in particular. Both of these were powerful programs but neither was particularly user-friendly for undergraduates with little computing background. "EZ-RATS" was a brave attempt at user-friendliness but not a great success: it regular crashed and lost my students' work. When I started at Wake Forest I tried a different approach, using Ramanathan's (1989) textbook, which came with its own DOS software, Ecslib (later known as ESL). Ecslib offered a limited range of estimators—OLS, TSLS, and a few FGLS variants—but it was stable, free (as in beer, to users of the textbook) and easy to learn.

Over the first half of the 1990s Microsoft Windows 3.0 (released in May, 1990) became increasingly popular and DOS software came to look dated and unfriendly. I decided to learn to program in Microsoft's Visual Basic. This was not a pretty language, but it did make for easy construction of a Graphical User Interface (GUI). Also in the early '90s I was introduced to TEX and LATEX by a computer-scientist friend, and I used Visual Basic to write GUI "front-ends" for both Ramanathan's ESL and what was at the time the best free implementation of TEX for the PC, Eberhard Mattes' emTEX.[1]

[1] Archive item: the web page for my emTEX front-end is still viewable at `http://www.wfu.edu/economics/ftp/emtexgi.html`.

Since it will be of some relevance in the sequel, let me define a "front-end". I mean a GUI program whose *raison d'être* is to make it easier for a user to interact with a command-driven (or CLI, "Command line interface") program, that is, a program that is driven either by commands typed at an interactive prompt or by a "script" of commands previously written to file. A front-end supplies an apparatus of dialog boxes, buttons, drop-down lists and so on, by means of which it enables the user to formulate a request to the CLI program. The front end then

1. translates the user's request into commands intelligible to the CLI program;
2. feeds these commands to the CLI program;
3. retrieves the output from the CLI program; and
4. displays the output to the user in a "window" of some sort.

From the user's point of view the attraction of such a front-end is that it obviates the need to master the command-line vocabulary and syntax of the underlying CLI program. From the programmer's point of view, it can be an interesting intellectual challenge to take one's knowledge of the CLI program and parlay it into an easy-to-use interface. This offers the same sort of reward as teaching: the satisfaction of taking something difficult and making it as clear and simple as possible.

Anyway, gretl's first precursor was ESLWIN, a Windows front-end for Ramanathan's DOS program ESL.

1.2 Linux comes on the scene

In the mid-1990s Wake Forest University set out an ambitious *Plan for the Class of 2000* which would put it among the most "wired" universities in the US. This involved distributing IBM ThinkPads to all students and faculty,[2] and a team of IBM people came to campus to discuss the plan. Windows 3.11 had reached the end of the road and Windows 95 was about to appear, but IBM's OS/2 could still (just about) be presented as a credible alternative, and the IBM guys gave out copies of OS/2 to faculty members who were willing to give it a try. I tried it but didn't like it much. But in carrying out the experiment with OS/2 I found that it wasn't really all that hard to install a parallel operating system, and that made me think of installing Linux, about which I had been hearing good things.

Linux was very much to my liking from the start, and I have used it almost exclusively since 1995. Since I didn't want to "waste time" using any OS other than Linux, but was still using Ramanathan's ESL with my students, I asked Ramu if he'd be willing to give me a copy of the source code for ESL so that

[2] This was long before IBM sold its ThinkPad business to Lenovo.

I could build a Linux version. He kindly said Yes. Around this time I had a sabbatical semester and spent much of the time learning the C programming language. ESL was written in C, and it didn't take much effort to get it running on Linux.

My first attempt at a GUI econometrics program on Linux was a re-write of ESLWIN using the GUI scripting language Tcl/Tk, namely TkESL: again, a "front-end" for a command-line program. This was workable but I soon felt the need for something better. Front-ends are inherently limited. The external relation between the GUI apparatus and the underlying command-processor is a problem—there's always the possibility of a disconnect when translating from GUI objects to commands, then translating back from text output to GUI display. The smallest change in the CLI program can wreak havoc. Besides, the mechanism is inherently inefficient: too much parsing and re-parsing is required,[3] and for each command, or batch of commands, passed to the CLI program, that program must be run from scratch, which always involves costs of initialization. All of the burden of "remembering the state" is placed on the GUI wrapper.

1.3 Enter GTK

The graphical image manipulation program GIMP—initially written by Spencer Kimball and Peter Mattis when they were graduate students at Berkeley—was one of the first "modern" open-source GUI programs to emerge, and it quickly became a flagship product for the Free Software movement.[4] Mattis had originally used Motif—a proprietary graphical interface toolkit for unix-type systems—for GIMP, but by the time of the 0.6x series he was "really fed up with Motif" and decided to write his own toolkits, which he called gtk and gdk for the Gimp Tool Kit and the Gimp Drawing Kit. By version 0.99 of GIMP (1997) this had evolved into GTK+, and as the historian on `www.gimp.org` relates, "Some developers got the crazy idea that it was a great toolkit and should be used in everything." GTK+ became the basis for the Gnome desktop, and (in a smaller way) it also became the basis for the gretl GUI.

Previously available GUI toolkits for unix/Linux were proprietary and/or very "low-level" and difficult to program. GTK+ introduced a new paradigm

[3] I have no expertise in biology, but I enjoy reading popular science. Over the years I've often been somewhat puzzled by accounts of various sorts of "transcription" of information at sub-cellular level: isn't there more transcription going on than is strictly required? But having programmed GUI front-ends I think I now understand what's happening. Evolution is a hack—a brilliant hack, but a hack nonetheless—and as such it partakes of the same sort of hackery as a GUI front-end, where information that is "well understood" at point A cannot be communicated directly to point B, but must be coded up for re-parsing at B!

[4] The first public release of GIMP was version 0.54 (January 1996). See `http://www.gimp.org/about/ancient_history.html`.

and it was quickly apparent that this was the wave of the future. In the late '90s I first experimented by coding gstar, a GTK+ front end for the "starchart" program (written by Alan Paeth and Craig Counterman),[5] and then began making a proper econometrics GUI using GTK+.

2 Design

Ramanathan's ESL was an all-in-one command-line program.[6] If gretl was to be more than a front-end for that program, the first task was to take the basic econometric code and put it into the form of a library, preferably a "shared" one of the modern sort. The next step was to reconstitute a working command-line program linked against the library, and check that it produced the same results as the original ESL. And the step after that would be to write a GUI client program for the same library—not just an external "front end" but an integrated program.

I'll spare you the details of this process, and just note that it was a great learning experience. I remember my excitement when gretlcli plus libgretl first churned out a set of OLS estimates that checked out correctly against ESL.

The GUI took longer, of course, but eventually it fell into place too. When I started with GTK+ it was clearly a good way to go for the Linux platform, but I began to wonder if I'd have to learn Windows programming if I ever wanted to produce a similar GUI for Microsoft Windows (e.g. for my students). Fortunately this was not so. Thanks to Tor Lillqvist's efforts in porting GTK+ to Windows (originally because he wanted to port the GIMP), programmers working on Linux can now create Windows versions of their GTK+ programs with ease.

2.1 Design schema

Gretl's design schema is shown in Figure 1. The command-line and GUI clients are in a sense at par as clients of libgretl (although of course the GUI program is a great deal more complicated). There are two other elements in the picture.

First, we have implemented several of the less commonly used features in gretl as dynamically loadable modules ("plugins"). The basic idea here was to avoid "bloat" in libgretl itself and hold down the memory footprint of the main library. This may now seem over-scrupulous given the amount of RAM to be found in today's PCs. You may blame my thrifty Scottish upbringing if you wish.

[5] See `http://ricardo.ecn.wfu.edu/~cottrell/gstar/`.

[6] Although the 'L' in ESL stood for "Library", ESL did not provide a library in the technical sense.

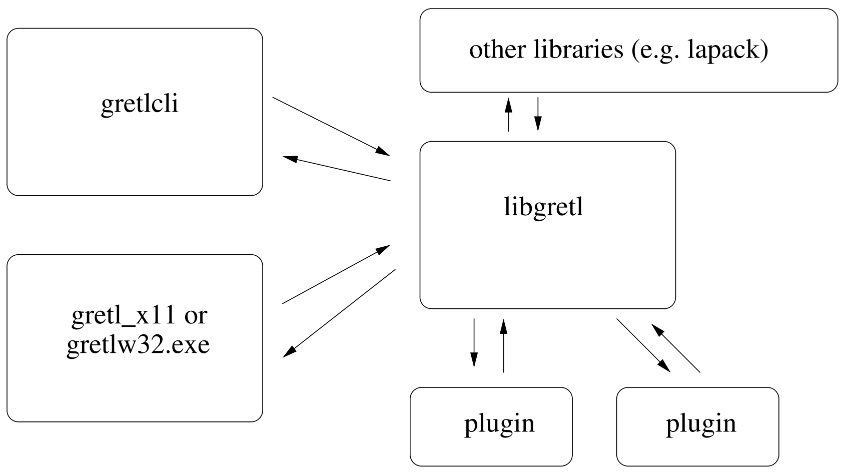

Fig. 1. Schema of gretl's design

Second, gretl relies on various third-party libraries to support its functionality. Probably the most basic of these are libxml2 (since gretl's data files and session files are stored in XML) and Lapack.

Lapack was introduced as a gretl dependency in version 1.0.8 of gretl. This followed the review of gretl by Baiocchi and Distaso (2003). These authors were basically enthusiastic about gretl, but drew attention to some issues of numerical precision. Up till this point all regression calculations in gretl had used the Cholesky decomposition code inherited from ESL. Cholesky decomposition is efficient and accurate for "reasonable" data, but breaks down on very highly collinear data. To ensure accurate results for a data matrix arbitrarily close to singularity it was clear that we needed QR and/or SVD methods, and rather than attempt to code these from scratch we decided to use what is in effect the gold standard for linear computation, Lapack.

I've heard that some people have expressed surprise that gretl doesn't make use of GSL, the Gnu Scientific Library. When I was first thinking about drawing on a third-party library for numerical computation I did consider GSL, but at that time it seemed to me that GSL was relatively immature; it appeared to replicate some but not all of Lapack's functionality, and was much less well tested than the latter. Things have moved on since then, and there may be a case for revisiting this issue, but I can't say I'm sorry to have chosen Lapack. The Fortran interface is awkward at first but it doesn't take long to learn, and we now have a substantial suite of `gretl_matrix` functions that offer C wrappers for the Lapack APIs. And Lapack development continues, witness the recent release of Lapack 3.2.1 and the xblas library with extended precision BLAS functions.

2.2 Other third-party libraries

It's not necessary to enumerate all the additional libraries that gretl either requires or uses optionally (from libfftw to gtksourceview), but a few issues may be worth mentioning.

One issue is Internet connectivity. For several years we've had a "database server" at Wake Forest University from which gretl users can download database files. More recently we've extended this to traffic in "function package" files, and this month (May 2009) I've added the ability to download from within gretl the packages of data files associated with the textbooks by Wooldridge, Stock and Watson, Verbeek and so on. The code we use to enable such traffic was originally "borrowed" from GNU wget and modified for gretl. It seems to work OK for the most part, but I wonder if we could do better, in terms of robustness and extensibility, by linking against libcurl (see `http://curl.haxx.se/`). One nice thing about linking to a third-party library that is under active development is that one gets bug-fixes and new features "for free"—just sit back and enjoy.

Another issue relates to the reading and writing of files in the PKZIP format. Gretl sessions files are zipped in this format, following the pattern of ODF files; and we now read ODS spreadsheets. Similarly to the borrowing from wget, the gretl code for handling zip archives is adapted from code by Mark Adler *et al* from Info-ZIP (zip version 2.31). Since I made that adaptation, libgsf—the Gnome Structured File library, coded in conjunction with Gnumeric—has become reasonably mature. It still (as of version 1.14.12) does not offer all the functionality of Info-ZIP for handling zipfiles, but while our chunk of zip code is effectively frozen it's likely that libgsf will continue to develop apace, so there may be a case for switching to libgsf at some point.

The third and last point that I'll raise in this context does not concern a library, but a third-party *program* of which we make extensive use, namely gnuplot. In this case gretl plays the role of a "front-end", and I spoke earlier of the inherent limitations of that role. There has been talk in the gnuplot community at various times of making a "libgnuplot" library, which would be very helpful from our point of view, but there doesn't seem to be much momentum behind that move, so far as I can see. Over the years I have from time to time checked out some seemingly promising options for graphing libraries, but have not found anything that offers all the functionality of gnuplot. Meanwhile, although there's no gnuplot library yet, gnuplot—which has been very full-featured for a long time—continues to improve. And we've established good relations with the gnuplot developers, and have gained acceptance for some patches which make gnuplot more "gretl-friendly".

This is one area where things are easier in relation to the gretl packages for Windows and OS X than for gretl on Linux. With the Windows and OS X packages we can ship a build of CVS gnuplot that we know will "do the right thing"—specifically, that it can handle UTF-8 encoding, and will produce high-quality PNG and PDF output using the Pango and Cairo libraries. On Linux, gnuplot is probably already installed, but we can't be sure what version and how well it's configured, so we have to implement a lot of workarounds. There may be a case for making a comprehensive gretl package for Linux that bundles gnuplot, though this rather goes against the grain.

3 Prospect

Gretl began life as a teaching tool—specifically, a tool for teaching econometrics at the undergraduate level—but it has grown far beyond that.

As an index of this growth, let me refer back to 2005, when I visited gretl contributors in Ancona, Torun and Bilbao. I recall a discussion in Bilbao where people were putting forward their wish lists for future developments: these included general purpose MLE functionality, GMM, and a general facility for manipulating matrices. I'll admit that these tasks seemed quite daunting at the time, but there followed a flurry of gretl activity in which all of these things were added and more. After spending time together in person, Jack Lucchetti and I were able to collaborate very effectively in coding some of the more challenging additions. The `mle` command was introduced in gretl 1.5.0 (December 2005) and general matrix functionality in 1.5.1 (March 2006). Version 1.6.0 (September 2006) introduced "native" exact ML estimation of ARMA models using the Kalman filter. GMM followed in 1.6.1 (February 2007), along with the Arellano–Bond dynamic panel estimator.

Over the same period the internationalization of gretl accelerated. The first translations were into French and Spanish (2002), then Italian and Polish (2004). Since then we have added Basque, German, Turkish, Russian, Portuguese, Traditional Chinese and most recently Czech. And these are challenging translations, requiring a firm grasp of technical econometric terminology.

The question arises, what should be gretl's role? What should we be aiming for? Reverting to the discussions of 2005 for a moment, there was some debate at the time on the gretl mailing list as to whether it really made sense for gretl to aim for a substantially higher level of econometric sophistication—the alternative being to concentrate on polishing gretl as a robust and user-friendly teaching tool. De facto, this debate has been resolved in favour of the pursuit of sophistication. I think there's at least one good rationale for this.

It has occasionally been put to me that maybe I'm not doing my students a service by teaching econometrics using gretl. The argument is that students are better served if they use from the start a software package that they can continue to use as researchers and professionals; by doing so they would be learning marketable skills and avoiding the need to re-learn how to do things with "standard" software. This sort of comment rankled greatly, but I recognize it has some validity. If gretl were a "dead end" it would be relatively difficult to justify its use in teaching, even if it is more user-friendly than the alternatives. Why not use Stata, Eviews or SAS? Just because we're Free Software ideologues, or happen to enjoy messing about with coding?

3.1 Promoting the adoption of gretl

For several reasons, those of us involved in gretl's development would like to see gretl used more widely, both in teaching and in research. We believe that free, open-source software is desirable in its own right, and is particularly desirable in the scientific domain where it ought to be clear precisely how results are obtained (which is not the case with closed-source proprietary software). [Add reference to Talha Yalta's paper?] We also believe that we have an excellent piece of software in gretl and that students would benefit from using it.[7] And we'd like to see the gretl developer community expand, so that we are less reliant on just a few coders and the project can become self-sustaining.[8]

Jack Lucchetti's analysis of downloads of gretl from the SourceForge site (Lucchetti, 2009) shows a rising trend. That's good, but there are factors making it difficult for gretl to achieve a "break through" to a substantially higher level of adoption. Jack mentions some of these; I'll elaborate a little.

One obvious point is that gretl is competing in a tough market. I don't have solid data to back up this claim, but it seems that Stata and Eviews are currently the leading products in the teaching of econometrics. These programs are also widely used in research, and seem to have edged previously popular software such as RATS and Limdep into niche roles. If gretl were a commercial product aiming to break into this market in a big way (and not just to find a niche) we'd be spending a great deal on advertising, would have a booth at the annual meetings of the Allied Social Science Association, and so on. In fact, of course, from the start we had to rely on "osmosis", achieved through, for example, personal contacts and web searches (e.g. people looking specifically for open-source statistical software). However, we now have a factor working in

[7] This opinion is not confined to gretl developers. I have received many, many emails over the years from professors and students around the world, to just this effect.

[8] And, of course, at a personal level, a bit more recognition would not go amiss: those of use who work on gretl make no money out of it—that was not the plan—but we're only human.

our favour, namely the gretl-aware textbooks that have been published in Polish (Kufel, 2007, also available in a Russian edition) and Spanish (Gallastegui, 2005). In addition we have Lee Adkins' gretl-based ebook (2009) and the forthcoming fourth edition of Christopher Dougherty's *Introduction to Econometrics* (Oxford). In this context I wonder if it would be worth exploring the possibility of writing collaboratively an English-language econometrics text that makes use of gretl? Lee Adkins has done a great deal in this direction already, but I'm thinking of something that would not be tied to a specific existing text (Adkins' ebook is designed to accompany Hill, Griffiths and Lim (2008)), and that, hopefully, could be placed with a major publisher.

3.2 Gretl and R

Still under the general topic of the difficulty of breaking into the highly competitive market in econometric software, one special issue arises for gretl. Besides its specific design features, gretl's most notable attribute is obviously that it is open source and free. But gretl is not entering an empty space in that respect: in residence is the highly respected and full-featured GNU R.

It's noteworthy that R achieved a very positive write-up in the *New York Times* earlier this year (Vance, 2009). Hal Varian (now chief economist at Google) is quoted as saying, "The great beauty of R is that you can modify it to do all sorts of things. And you have a lot of prepackaged stuff that's already available, so you're standing on the shoulders of giants." The article notes the increasing adoption of R for data analysis in both academic and commercial contexts and cites Max Kuhn, associate director of nonclinical statistics at Pfizer: "R has really become the second language for people coming out of grad school now, and there's an amazing amount of code being written for it. You can look on the SAS message boards and see there is a proportional downturn in traffic." The rejoinder by a SAS spokesperson, "We have customers who build engines for aircraft. I am happy they are not using freeware when I get on a jet," sounds defensive and out of touch.

What does R's success mean for gretl? On the one hand it demonstrates that there is scope for free software to make substantial inroads on the turf of the vendors of proprietary statistical software, which is very encouraging. On the other hand it may be seen as raising the question of whether gretl is really needed. Is there room for gretl alongside R in this domain? The gretl developers are well aware of R's strengths but consider that gretl still has a role to play. Gretl has an intuitive GUI; R does not. While gretl is in general much less comprehensive than R it nonetheless adds some specialized econometric functionality in relation to R. And we have taken pains to make gretl as interoperable with R as possible, so that users can take advantage of the complementarity between the two programs. Nonetheless, this is an area where more thought and more work are required: what exactly should be the relationship between gretl and R?

3.3 Extending gretl

I mentioned earlier the goal that gretl should become self-sustaining, and not overly dependent on the work of a few individuals. In that regard, the way in which the range of packages for R has mushroomed (see Varian's comment above) is very pertinent. In 2006 we introduced a facility to create (and download) "function packages" containing user-contributed code for gretl—code written in the gretl scripting language rather than C. I think this is the right way to go, but although some excellent packages have been contributed it's fair to say that this has not "taken off" to date. This is something we should revisit. There are some awkward aspects of the gretl function packager and we should resolve these. We also need to think about the issue more generally: is there anything we can do specifically to promote the contribution of packages? What can we learn from R?

In closing I'll mention one other aspect of extending gretl and getting it to be better known. In section 2 I spoke about gretl's shared library, libgretl, which was originally created by adapting Ramanathan's ESL code base. The thing is that libgretl in some ways still bears the marks of its origins. Basically, it contains *all* the common code that is needed by both the command-line and the GUI client programs. Some of this code (in particular sections that have been added relatively recently) is quite general and offers a reasonably clean and consistent API—for example the `gretl_matrix` code, the probability distribution code based on Stephen Moshier's `cephes`, the BFGS maximizer, the Kalman filter—while some of it is highly gretl-specific and presents APIs that are unlikely to be comprehensible to anyone who hasn't worked on gretl for years.

One idea for the future then, is to factor out the "private" and "public" components of the current libgretl and to spruce up the APIs of the latter. We'd have, say, `libgretl_priv` and (public) libgretl. It would become easier for new contributors to find their way around the code base, and at the same time third-party developers would have access to a cleaner and more manageable econometrics library for use in their own projects, hence promoting the use of gretl code and gretl's visibility.

Bibliography

[1] ADKINS, L. (2009): *Using gretl for Principles of Econometrics, 3rd edition.* online., Version 1.211, `http://www.learneconometrics.com/gretl/ebook.pdf`

[2] BAIOCCHI, G. AND DISTASO, W. (2003): "GRETL: Econometric software for the GNU generation", *Journal of Applied Econometrics*, 18, pp. 105–10.

[3] GALLASTEGUI, A. F. (2005): *Econometrìa*, Madrid: Pearson Educación.

[4] HILL, R. C., GRIFFITHS, W. E. AND G. C. LIM (2008) *Principles of Econometrics*, 3e, New York: Wiley.

[5] KUFEL, TADEUSZ (2007) *Ekonometria*, 2e, Warsaw: Wydawnictwo Naukowe PWN.

[6] LUCCHETTI, RICCARDO (JACK) (2009), "Who uses gretl? An analysis of the SourceForge download data", this volume.

[7] RAMANATHAN, RAMU (1989), *Introductory Econometrics with Applications*, San Diego: Harcourt Brace Jovanovich.

[8] VANCE, ASHLEE (2009) "Data Analysts Captivated by R's Power", The New York Times, January 6 `http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html` Visited 2009-04-04.