The gretl_edit facility

Allin, 2022-08-05

This note describes something relatively new in gretl git: a secondary, specialized GUI executable called `gretl_edit`. This program is built only if `--enable-build-editor` is passed to gretl's `configure` script. Will anyone other than me be interested in using it? We'll see (but Jack has indicated some interest).

As opposed to the regular `gretl` executable, `gretl_edit` starts up with no 'main window', just a script editor window (a tabbed editor, regardless of your choice of tabbing or not in the regular program). This window will be populated by a script if you append its filename on the command line, otherwise it will show a new, untitled script.

Why would anyone want that? Well, sometimes I have use for the full gretl GUI, but in the context of intensive development of new hansl code I sometimes don't; I'd like to use the built-in script editor, but I have no interest in the main window or any of its progeny. All I want to do is edit a script, run it, examine its output, modify the script, run it again... maybe dozens or even hundreds of times.

And in that context I don't want *any* state to be preserved between iterations; each run should be totally *de novo* or things will get messed up.

So that's what `gretl_edit` gives you. The 'Run' button in the script editor window and its keystroke equivalent, `Ctrl+r`, are remapped so that they invoke `gretlcli` to run the current script and capture the output to a GTK window. *No state is preserved in the GUI other than that of the script(s) being edited.* So far as `gretl_edit` is concerned there's no "current dataset", no saved matrices or whatever. Things of that sort exist in the memory space of `gretlcli` while it's executing and vanish thereafter.

When developing complex hansl code I sometimes have a script `S` whose results are known to be good but which is not as ambitious as it might be, and another script `S'` which is more ambitious but at any point might represent a regression relative to `S`. The new program is set up to handle that sort of thing. Regardless of the usual choice of what exactly to do with the output when a script is executed in the GUI, each script gets its own specific output window, whose contents are replaced whenever the script in question is run. The only additional windows that may appear are plots called for in a script, if the relevant commands include the `--output=display` directive.

A couple of new keyboard shortcuts are supported by `gretl_edit`: `Ctrl+o` activates the File dialog for opening a script file, and `Ctrl+Alt+k` kills the `gretlcli` process currently executing a script, if any (just in case of an infinite loop or unexpectedly time-consuming computation).

macOS: For now We're not including `gretl_edit` in our packages for macOS. That's because putting a secondary GUI executable into an Application bundle is contrary to the design of such bundles, and leads to ugly consequences. The secondary program doesn't have its own icon, it just gets the generic "exec" icon; launching `gretl_edit` via its representation inside the bundle spawns a Terminal window (the executable is not recognized as self-contained); and you get a broken top-of-screen menu with a "Quit gretl_edit" item that doesn't work. Maybe we can find a way around this in future.