Common Hansl Commands

Erik Johnson[*]

This document provides a brief summary of commands to use in the Hansl scripting language that is part of Gretl that are likely to come up quite a bit as you write scripts. For complete documentation, consult either the Gretl User's Guide, the Hansl Primer, or simply type `help [command]` in the Gretl console to see the internal help file. This file assumes that you have either already worked through my Gretl tutorial (including the Hansl portions) and therefore have a limited understanding of coding in Gretl or have a familiarity with coding in a different language.

General notes

When writing Hansl scripts, there are two comment characters. You can either use a `#` to denote the rest of the line of code is a comment. However, if you would like to write a multi-line comment without putting a `#` at the beginning of each line, you can begin the comment with `/*` and then end the comment with a `*/`.

Secondly, Gretl commands often have many options underneath the primary command. All options for a command are specified after a double dash (`--`) that is then immediately followed by the option name and potentially a value for the option. See the notes below on the `summary` or `smpl` commands for some specific examples, though nearly all Gretl commands have options.

Finally, Gretl is entirely based on matrices and therefore many items can be accessed by referencing their location in a matrix. The location in a matrix is defined by using square brackets (i.e. [,]) after the matrix name and writing the row of the number first and column number second separated by a comma: matrixname[row, column].

For instance, after Gretl calculates summary statistics for a variable, all of the numbers that are displayed are stored in a matrix called `$result` and you can then access those numbers by making

[*]Department of Economics, Carthage College

your own matrix or scalar that takes its values from that matrix. If I wanted to save all of the entries in the first column of the matrix, I would simply write `matrix foo=$result[,1]`. Here, the lack of a number before the comma indicates I want all of the rows of the matrix and the "1" indicates I want the first column. If you want a subset of rows or columns, use the `:` character. For instance to get only the second row and the second through fourth column I would write `matrix foo=$result[2,2:4]`. If I simply wanted one number, I would probably store it as a scalar (not a matrix) and write `scalar foo=$result[2,2]` to get the entry in the second row and second column.

Common commands

- `append`: adds more observations to a data set. You can think of this as simply putting more observations at the end/beginning of a spreadsheet. This command will fail without any useful error messages if the data set you are appending does not have **exactly** the same variables as the data set you are appending the data to.
- `discrete`: declares a variable or list of variables as discrete (takes on only a limited number of integer values. Type `discrete v1 v2 v3` to make **v1**, **v2**, and **v3** discrete variables.
- `dummify`: makes a complete set of dummy (indicator) variables for any categorical variable. The new variables have names of **Dx_1, Dx_2, ...** where **x** is the name of your variable. Your variable should be declared as a discrete variable to Gretl before using the this command. (See `setinfo` or `discrete`.) To make two dummy variables for a variable called **sex** that takes on two values, type `dummify sex`. You would then have two new variables in your data set called **Dsex_1** and **Dsex_2**.
- `genr`: makes new variables using combinations of other variables (multiplying, adding, etc.) for example `genr series v3=v1*v2` or using algebraic functions for example `genr series lnv1 = ln(v1)`. Here the `series` portion of the command tells Gretl that the result should be a variable.
- `join`: allows you to add data to an existing data set from another, outside data set. It is often easiest, though not always necessary, for the outside data set to also be in Gretl format. The `join` command encompasses many different ways of adding data to your data set and making sure each observation in your current data set is assigned the correct value from the

other data set.

In order to make sure each observation is matched with the correct variable you will need a "key" variable in each data set. The key is the variable that is common both both data sets that the computer will look at to assign the value of the new variable to each observation. For details about how to join data sets, consult Chapter 7 in the Gretl User's Manual.

- `loops`: allows you to do almost the exact same command multiple times with some small variation. When you find yourself typing nearly the same thing many times in a row, this is a good sign that you should use a loop. There are multiple types of loops that you can use in Gretl. For a good description of how to write loops, consult Chapter 12 in the Gretl User's Manual.

- `open`: loads either a Gretl data file or data file of another format that Gretl can read such as .csv, .txt, .dta. The open command will clear the data file that is currently loaded into Gretl. (Example: `open mydata.gdt`)

- `rename`: changes the name of a variable. To use the command type
  ```
  rename oldname newname
  ```

- `replace`: replaces the values of a particular variable, matrix, or scalar with another value. For instance if you wanted to change all observations that are currently have a value of 0 to a value of 2, you would type `replace [varname] = replace([varname],0,2)`

- `setinfo`: controls the description information about a variable. You can use it to change the descriptive label, mark it as discrete, or change what is displayed when you graph it. For a listing of all of the options type `help setinfo` in the Gretl console.

- `smpl`: controls what portion of the data Gretl will currently examine. You can use logical conditions (see Gretl tutorial for more on this) or return to the full sample (`smpl --full`).

- `store`: saves your data set in your preferred format. The default format is a Gretl format but can also be used to save it in other formats for use by other programs. The default action is to overwrite any dataset with the same name. (Example: `store mydata.gdt`)

- `stringify`: provides the ability to assign meaning to categorical variables. Since all data are stored as numbers, often it is useful to assign labels to those numbers to when looking at the data. For instance, if you have a variable that is the sex of the person who answered a survey, each sex will have a different number. However, when you examine the data you will only see the numbers and not something like "male", "female". In order to make the numbers meaningful, use the stringify command to add labels. The categorical variables should be numbered sequentially between 1 and the number of categories of your variable. In order to add the labels you will need to make an array of strings (non-numbers) using the `strings` and `defarray` command. Then you will use stringify command to assign the labels. If I have a variable called **sex** where 1's are males and 2's are females the following two commands would assign labels to the 1's and 2's.

```
strings sex_label = defarray("Male", "Female")
stringify(sex, sex_label)
```

Note that if females were 1's and males were 2's you would need to switch the order of the label in the `defarray` command.

- `summary`: displays descriptive statistics for the list of variables directly after the summary command. You can either get all descriptive statistics (no option after the command) or basic summary statistics by using the `--simple` option. (Example: `summary variable1 variable2 --simple`)
- `varlist`: lists all variables (or matrices or scalars) in the current data set. `varlist --series` returns all variables, `varlist --scalar` returns all scalars, etc.