Stata Group Comparison (GC) ado files

Purpose: The GC-programs simplifies the comparison of predicted probabilities across groups in binary logit models. Binary probit will be added if there is demand. The programs are a wrapper for SPost's (Long and Freese 2005)[i] **`prvalue`** to make it simpler to compare predicted probabilities across groups in binary logit and (eventually) binary probit.

Date/version: 2009-05-26 version 0.5.x

Author: Scott Long

Location: Currently in the **`scottlong`** Stata package (**`findit scottlong`** to install).

Overview

When comparing groups in regression models for binary, ordinal, or nominal outcomes, tests of the equality of coefficients across groups (e.g., $B_{x,m}=B_{x,f}$) are invalid since they confound group differences in unexplained variation in outcomes and the group differences in the magnitude of the effects of the variable on the outcome (see Allison 1999). Long (2005; 2009) shows that tests of the equality of predicted probabilities from the two groups are not affected by the identification problem that confounds effect size and unexplained variation. He proposes testing the equality of predicted probabilities across groups. This approach requires software to estimate the model (e.g., **`logit`**, **`probit`**), predictions of the probabilities (e.g., **`predict`**, **`prvalue`**), and tests of the equality of the predictions (e.g., **`prvalue`**). While Long's recommendations can be implemented with the SPost **`prvalue`** command, with even moderately complex models this can be complex and error prone. The GC commands are a wrapper for **`prvalue`** to make comparisons simpler.

Overview of how the GC commands work

The GC commands:

1) Creates group indicator variables such as white=1 if white, else 0; black = 1 if black, else 0.

2) Creates interactions between group indicators and the RHS (right-hand-side) variables in the model (e.g., wx1 = white * x1; bx1 = black * x1).

3) Creates a global macros with the names of the RHS variables in a both a non-interactive model in which group is included only as a dummy variable and in a fully interactive model in which the coefficient for each RHS can differ by group.

4) It creates global strings setting variable to values (e.g., white=0 wx1=0 black=1 bx1=11.2) that are used to compute predicted probabilities by passing these strings to **`prvalue`** using either **`gcprvalue`** or **`gcprgen`**.

5. Creates tables with results and variables with predictions that can be plotted.

Quick summary of each GC command

1) **gcsetup** indicates the non-group RHS variables that might be in models that are later estimated; names the group variable that already exists in the dataset (e.g., **grpvar(male)**) that is coded 0 for group 0 and 1 for group 1; provide names for group indicators variables (e.g., **gr0var(f0)** creates f0 that is 1 if a person is equal to 0 for the group variables, in this case male==0). It also lets you assign labels to the two groups. Here is a sample command for a single RHS variable article with groups defined by variable male:

```
gcsetup articles, details grpvar(male) gr0var(f0) ///
  gr0label(women) gr1var(m1) gr1label(men)
```

2) **gcmodel** specifies the non-group RHS variables to include in the model. It creates globals with the names of the variables to include in the interactive and noninteractive models. For example:

```
gcmodel articles, label(M1_articles) details
```

indicates that the RHS variable is articles, the model is named M1_articles. The details options prints which globals are created. For example the global rhsnoint contains "male articles". It also makes it simple to include squared and cubed terms in the model and will automatically determine the values for these variables given the value set for the linear term. Multiple models can be specified after using a single **gcsetup**.

3) Estimates the logit model. You can use the globals created by **gcmodel** to specify the model. For example:

```
logit tenure $rhsnoint
```

is equivalent to entering the command:

```
logit tenure male articles
```

For complex models, this allows you to easily specify very complex models.

4) **gcprvalue** uses the globals created by **gcspecify** and **gcmodel** and the results of the model just estimated to run a pair of **prvalue**'s that will compare the predictions for the two groups. For example, the command:

```
gcprvalue, x(articles=15) matrix(model1)
```

is equivalent to running a pair of **prvalue, save** and **prvalue, dif** commands to compare the groups. By using the same matrix in repeated calls of **gcprvalue**, you can save the results to a matrix that is printed with the print command or by using the **gcresults** command.

5) **gcresults** simply prints the results that have been collected to a matrix by repeated uses of **gcprvalue**.

6) **gcprgen** computes predicted probabilities for both groups as one variable varies. It also computes the group difference in the probabilities along with a confidence interval for those differences. For example,

```
gcprgen articles, from(0) to(50) gap(2) gen(m1)
```

computes predictions as articles increases from 0 to 50 in steps of 2. The resulting predictions are stored in variables that have the stem **m1**.

7) There are several utilities that the user will not need to use directly unless there are problems.

a) **gcwhich** is a utility that indicates which versions of the GC commands are being used. If you have problems with the GC commands which are still being developed, be sure to report the output of **gcwhich** if you report any problems.

b) **gcreset** is used by various programs to reset globals that have been created.

c) **gcglobals** is a utility that lists the values of global variables created by GC commands.

d) **gc_xischeck** is a utility called by other programs to verify that globals are defined as needed.

e) **gc_setcheck** is a utility to verify that **gcsetup** has been run.

Example

See the gcpaper*.do do-files for the example used in Long (2009).

References

Allison, PD (1999)

Long, JS (200x)

Long, JS (2009)

Long, JS and J Freese (2005) xxx

History

2009-05-21: gc suite from summer 2008 was too complex. Began 0.5.x version with simpler commands and syntax.

[i] Online in Stata, enter: **findit spost9_ado**. See http://www.indiana.edu/~jslsoc/spost.htm for details.