Catalogue no. 12-002-XIE

The Research Data Centres Information and Technical Bulletin

Spring 2004, vol.1 no.1

Statistics Canada Statistique Canada

Canada

How to obtain more information

Specific inquiries about this product and related statistics or services should be directed to: Research Data Centres Program, Statistics Canada, Ottawa, Ontario, K1A 0T6 (telephone: 1 800 263-1136).

For information on the wide range of data available from Statistics Canada, you can contact us by calling one of our toll-free numbers. You can also contact us by e-mail or by visiting our Web site.

National inquiries line	**1 800 263-1136**
National telecommunications device for the hearing impaired	**1 800 363-7629**
Depository Services Program inquiries	**1 800 700-1033**
Fax line for Depository Services Program	**1 800 889-9734**
E-mail inquiries	**infostats@statcan.ca**
Web site	**www.statcan.ca**

Ordering and subscription information

This product, Catalogue no. 12-002-XIE, is available on Internet free. Users can obtain single issues at http://www.statcan.ca/cgi-bin/downpub/freepub.cgi.

Standards of service to the public

Statistics Canada is committed to serving its clients in a prompt, reliable and courteous manner and in the official language of their choice. To this end, the Agency has developed standards of service which its employees observe in serving its clients. To obtain a copy of these service standards, please contact Statistics Canada toll free at 1 800 263-1136.

Statistics Canada
Research Data Centres Program

The Research Data Centres Information and Technical Bulletin

Spring 2004, vol.1 no.1

Published by authority of the Minister responsible for Statistics Canada

April 2004

Catalogue no. 12-002-XIE

Frequency: semi-annual

ISSN: 1710-2197

Ottawa

Cette publication est disponible en français (n^{o} 12-002-XIF au catalogue)

Note of appreciation

Canada owes the success of its statistical system to a long-standing partnership between Statistics Canada, the citizens of Canada, its businesses, governments and other institutions. Accurate and timely statistical information could not be produced without their continued cooperation and goodwill.

About the Information and Technical Bulletin

The Research Data Centres Information and technical bulletin is a forum for current and prospective RDC users to exchange practical information and techniques for analyzing datasets available at the centres. The bulletin will be published twice per year, in the spring and fall. Additional special issues on timely topics may also be released on an occasional basis.

Aims:

The main aims of the bulletin are:

- to advance and disseminate knowledge surrounding Statistics Canada's data;
- to exchange ideas among the Research Data Centre user community;
- to support new users of the RDC program; and
- to provide an additional means through which RDC users and subject matter experts and divisions within Statistics Canada can communicate.

Content:

The Information and technical bulletin is interested in receiving articles and notes that will add value to the quality of research produced at the Statistics Canada Research Data Centres and provide methodological support to RDC users.

Topics include, but are not limited to:

- data analysis and modeling;
- data management;
- best or ineffective statistical, computational, and scientific practices;
- data content;
- implications of questionnaire wording;
- comparisons of data sets;
- reviews on methodologies and their applications;
- problem-solving analytical techniques; and
- explanations of innovative tools using Research Data Centre (RDC) surveys and relevant software.

Those interested in submitting an article to the Information and technical bulletin are asked to refer to the Instructions for authors.

The editors and authors would like to thank the reviewers for their valuable comments.

Table of contents

Constructing survival analysis models with independent variables that change over time: Application using data from the Survey of Labour and Income Dynamics

Denis Gonthier

Abstract

This article provides a practical example of the development of a survival analysis model. It begins with an overview of the software tool that was used, SAS. The next section examines the construction of a longitudinal file and the challenges that may present. Of particular interest are explanatory variables that do not have a constant value over time. An example of a practical application is provided to illustrate the survival approach. The example consists of an analysis based on data from the Survey of Labour and Income Dynamics (SLID), specifically data from panel 1 between January 1993 and December 1998. Survey information in vector form is used to develop a Cox semi-parametric model. Comments are provided on a sample computer program. The way in which the program handles the main variables is also discussed. The last section contains a brief description of the results of a relatively simple model.

Introduction

In this paper we will present a practical application of survival analysis using information in vector form from a Statistics Canada longitudinal survey. Hence it is primarily a technical description, which may help researchers who have to deal with independent variables that change over time. Note that we will not be considering the design effect in complex surveys, which presents a special challenge in the context of survival analysis methods. In the model presented, we will be using only a scaled weight.

We will begin with a brief introduction on the software tool we used: SAS and its PROC PHREG. This will be followed by an explanation of the approach required when the explanatory variables do not have a constant value over time. Finally, we will describe a sample application based on data from the Survey of Labour and Income Dynamics (SLID), specifically data from panel 1 between January 1993 and December 1998.

II. Survival analysis and SAS's PROC PHREG

In the construction of a longitudinal analysis file containing information such as a weekly vector of labour market activity status, it may be helpful to use a program such as SAS. SAS has certain iterative functions that simplify the management of such data, as we will see below. SAS also has procedures that can be used to construct proportional risk models. In this case, we will be employing the Cox semi-parametric model. One of the advantages of the Cox model is that, in contrast to other methods such as the accelerated failure time model, the form of the underlying survival function does not have to be specified. The SAS procedure for this model is called PHREG. It should be noted that PHREG does not estimate the variance based on the survey design. SUDAAN could take the design effect into account in survival analysis using the bootstrap weights produced

by Statistics Canada, but it cannot handle independent variables that change over time. Since that is the kind of variables we have, and since they are in the form of vectors, we decided to use SAS.

With PHREG, there are relatively few constraints on the data configuration. This is not so for all survival analysis applications. One constraint is that the variables representing response categories must be redefined by creating a set of dichotomous variables (the reference category will be omitted in the model).

The PHREG procedure can develop simple models with independent variables that do not change over time. Examples of this are covariables such as mother tongue and sex. In such cases, it is sufficient to define the duration variable (dependent variable), define the end-of-episode status (an event or censorship occurs) and incorporate the fixed independent variables into the model.

Using the procedure is more complicated when the explanatory variables do not have a constant value over time. This is discussed in the next section.

III. Using independent variables that change over time

Variables that change over time can be introduced in the PHREG procedure using programming elements such as the IF, THEN operators and the ARRAY instruction that defines vectors. These programming elements are often used in SAS's DATA step, but they can also be used in the PHREG procedure.[1]

The value of the duration variable can be used to define an independent variable that changes over time. For example, PHREG can be programmed so that explanatory variable X takes the value 0 as long as the duration is less than two years (the unit of analysis time is the year), and changes to 1 when the duration is greater than or equal to two years.

If the analysis period is divided into a large number of intervals (for example, if an explanatory variable is measured every month for many months), it is useful to employ vectors to identify which element should be used based on the value of the duration variable. Then the ARRAY instruction is used to define the vectors. We provide an example in the next section.

This approach is made possible by the fact that the PHREG procedure continually recomputes the values of the explanatory variables, which can change with the duration.

[1] See SAS OnlineDOC, V8 - SAS/STAT - SAS/STAT User's Guide – The PHREG Procedure – Syntax – Programming Statements.

IV. Example based on the Survey of Labour and Income Dynamics

We will fit the Cox semi-parametric model to the data from the Survey of Labour and Income Dynamics (SLID) to measure the effect that selected explanatory variables have on the risk of changing provinces.[2] Hence interprovincial migration is the event observed in the model. A logistic regression could have been used to model migration, but the Cox regression was chosen because events are dated in the SLID. We can determine not only when a move occurs but also what the values of the independent variables are when it occurs (with a varying degree of precision).

Only residence episodes that began at some point in the six years during which SLID panel 1 was observed will be considered. This criterion ensures that there no episodes "open to the left".[3] The sample consists of respondents aged 16 to 69 who moved to another province between January 1993 and December 1998.

The model contains independent variables that change over time but differ in chronological precision. Activity status (employed, unemployed or not in the labour force) is measured weekly. Full-time student status is measured monthly. Renter/owner status is measured annually. The values of the independent variables are not always known at the time of migration (the specific date of which is known). We will therefore use the status at the closest point in time prior to the event.[4]

The dependent variable in the Cox model is the duration of the residence episode. It is calculated from the date on which the episode began, and we elected to measure it in years (including fractions of years because the exact dates of the events are known). This variable, known as DURATION, is the elapsed time between the beginning and end of the episode. Since all episodes began at some point in the six-year period, it follows that the DURATION variable will have a value somewhere between 0 and 6 (years). The STATUS variable indicates whether the episode came to an end because the event in question (migration) occurred or because there was censorship. Censorship may be due to the fact that the survey ended without the event occurring.[5]

The ARRAY procedure is used to define vectors of independent variables. For example, a vector called ETUD is created which consists of 72 dichotomous variables for monthly full-time student status during the six-year period considered. To determine which element of the vector will be used, we use as our reference index the MONTH variable, whose value changes with the time elapsed since the beginning of the survey. This time is given by the sum of ECART and DURATION, where ECART is the elapsed time (in years) between the beginning of the survey and the beginning of the episode.[6] Since the survey began on January 1, 1993, the ECART variable will

[2] The example given here relates to a research project currently being carried out in conjunction with Jacques Ledent, a research professor at the *Institut national de la recherche scientifique*.

[3] Such episodes would have started before the survey began, but their start dates would be unknown, as would the length of time elapsed before the observation period.

[4] It may be necessary to allow for the possibility that the independent variables would have a deferred effect, i.e., that a certain amount of time would elapse between the appearance of a particular condition and the effect that it produces. We will not factor such a deferred effect into our model. See Paul Allison's book *Survival Analysis Using the SAS® System: A Practical Guide*, which discusses this issue in the chapter on the Cox regression.

[5] It is possible that there was a further episode, in the case of a respondent who migrated again (the population being studied is composed of people who migrated once during the survey).

[6] In the current example, episodes that started before the beginning of the survey are excluded.

have the value 3.5 for an episode that began on July 1, 1996. Note that the survey records the exact date of the last move.

The value of the MONTH variable changes when the Cox model re-evaluates the dependent variable DURATION (while the value of ECART is fixed). And the model will seek the n^{th} appropriate variable in the ETUD vector. The approach is the same for the set of variables for renter/owner status, which has six annual values. The approach is also the same for the set of variables for activity status, which is measured 53 times a year, for a total of over 300 weekly values over the full length of the observation window. Activity status has three categories. The EMPLOYED and UNEMPLOYED categories will be used in the Cox model, and the reference group will consist of people not in the labour force. A sample SAS program using the PHREG procedure with independent variables that change over time is presented below. There are five independent variables, one continuous (age at the beginning of the episode) and four dichotomous. Where necessary, the measurement in years of the time elapsed since the beginning of the survey (ECART+ DURATION) is converted into months or weeks.

```
01  proc phreg data=test;
02    model duration*status(1)= age owner student employed unemployed;
03    freq poids / notruncate;
04    array prop{6} owner1-owner6;
05      year=ceil(ecart+duration);       /*  YEAR is a whole number varying between 1 and 6  */
06      owner=prop{year};
07    array etud{72} etudes1-etudes72;
08      month=ceil(12*(ecart+duration));   /* MONTH is a whole number varying between 1 and 72  */
09      student=etud{month};
10    array occup{318} occup1-occup318;
11    array chom{318} chomeur1-chomeur318;
12      week=ceil(53*(ecart+duration)); /* WEEK is a whole number varying between 1 and 318    */
13      employed=occup{week};
14      unemployed=chom{week};
15  run;
```

- Line 1 calls the PHREG procedure and the data file.
- Line 2 defines the model, where DURATION is the dependent variable (episode duration in years) and STATUS indicates whether the episode ended with migration or censorship (for censorship, STATUS=1).
- In line 3, FREQ is used to weight the data. The POIDS variable contains a weight that averages 1 for the population being studied (scaled weight). Weighting is used to offset the unequal selection probabilities of SLID respondents. The NOTRUNCATE option ensures that no decimal places are lost.
- Line 4 defines an ownership status vector, which changes annually and has six values (since the survey tracks respondents for six years).
- To select which of the six years the event occurs in, we compute an index in line 5 (the YEAR variable). This variable is a whole number (the CEIL operator rounds the value up to the next highest whole number).

- This index is used in line 6 to select one of the six ownership status values (owner or non-owner).
- The same logic applies in lines 7 to 9. First, a full-time student status vector (dichotomous variable) is defined. Then the total duration in years is multiplied by 12 to convert it to months (line 8).
- The same steps are repeated in lines 10 to 14, with the duration in years being multiplied by 53 to convert it to weeks (line 12).

As an illustration of how PHREG computes the values of independent variables that change over time, the table below shows what happens in the case of two fictitious observations when the compilation of the dependent variable (DURATION) reaches a value of 3.0 years.

	DURATION	DURATION + ECART [7]	MONTH	STUDENT	YEAR	OWNER
Case 1	3.0	3.5	42	Value=ETUDE42	4	Value=OWNER4
Case 2	3.0	4.5	54	Value=ETUDE54	5	Value=OWNER5

When the characteristics of the population at risk are selected for a duration of exactly three years, the values of the independent variables correspond to each respondent's individual timeline. We can use DURATION and ECART to find the values for the months in which the respondents reach an episode duration of three years: the 42nd month for case 1, and the 54th month for case 2. For a variable whose value is known only to the nearest month (such as student status), it makes sense to use the status for the month preceding the migration to ensure that we have a measurement of the characteristic prior to the event.

[7] Case 1's episode began on July 1, 1993 (ECART=0.5), and Case 2's episode began on July 1, 1994 (ECART=1.5).

The result produced by PHREG is shown below.

The PHREG Procedure

Model Information

Data Set	WORK.TEST
Dependent Variable	duree
Censoring Variable	statut
Censoring Value(s)	1
Frequency Variable	poids
Ties Handling	BRESLOW

Summary of the Number of Event and Censored Values

Total	Event	Censored	Percent Censored
1417.0	221.7	1195.3	84.35

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2878.360	2783.734
AIC	2878.360	2793.734
SBC	2878.360	2810.741

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	94.6255	5	<.0001
Score	96.2342	5	<.0001
Wald	90.9854	5	<.0001

The PHREG Procedure

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
age	1	-0.06097	0.00936	42.4367	<.0001	0.941
owner	1	-0.07004	0.14619	0.2295	0.6319	0.932
student	1	-0.02101	0.23297	0.0081	0.9281	0.979
employed	1	-0.18221	0.19365	0.8854	0.3467	0.833
unemployed	1	0.95681	0.23281	16.8912	<.0001	2.603

In this SAS output, we see that there are 1,417 observations consisting of residence episodes for respondents who migrated during the survey. There are 222 events, which are new migrations. The risk of changing provinces (for those who migrated between 1993 and 1998) is shown in the "hazard ratio" column at the bottom of the table. The table also shows that for each additional year of age, the risk of migrating falls by nearly 5% and that this figure is very significant (significance level of less than 0.0001). It indicates as well that the unemployed members of this subpopulation of recent migrants are much more likely to change provinces (hazard ratio of 2.6) than the reference category (not in the labour force). Once again, this result is very significant. Such is not the case for the model's other independent variables. It is important to note that the model used does not compensate for the design effect. Consequently, the results must be interpreted with caution. Similarly, the model does not consider the impact of moves involving the members of the same household. This point would have to be taken into account in any thorough study of the subject.

V. Conclusion

It is possible to get the most longitudinal information out of a survey such as the SLID by using variables with different time scales. This enhances the potential of techniques such as survival analysis in situations where not all the survey data have an exact measure of date.

Producing efficient data files using Stat/Transfer

By Leslie-Anne Keown

Abstract

Large data sets present several challenges to researchers, particularly to less experienced researchers. One of the most time-consuming and frustrating activities for beginning researchers, who do not have experience with large datasets, can be pruning or parsing the data set to only the variables and sample of interest. The production of an 'efficient' data file can assist in the increased performance of hardware and software as well as reducing frustration for the researcher. One way of producing a parsed efficient data file using a program called Stat/Transfer is presented.

Introduction

Often the data sets within the RDC are quite large and the set of variables that researchers need to work with is more compact. The experienced researcher often realizes the utility of building a subset of the data that contains only the variables of interest for that analysis and has a number of methods that have developed through their research experience to do this. The less experienced researcher may not have the experience with a particular software to do this easily and it can be a source of frustration that carries over into the rest of the analysis process. This level of frustration is commonly seen amongst many users of the RDCs', especially graduate students and research assistants.

In addition to producing a more compact file, the researcher may also wish to work on only a portion of the total sample of a particular survey – for example, only persons over the age of 18. Furthermore, researchers may wish to transfer the data file from one particular statistical program for use in another (i.e. convert a file from SPSS for use in STATA). Depending on the software being used, producing a dataset that is economical in regards to both physical size and breadth of data (parsing) can be both time-consuming and frustrating. This article proposes to suggest one alternative to creating a more compact data set regardless of the software that will be used to conduct the analysis. This method involves using Stat/Transfer. All examples will be from Version 6.1 of Stat/Transfer but the method should work the same regardless of the version being used.

II. Limitations of this article

There are a number of issues that will not be covered here for the sake of producing a piece that is relatively easy to follow and is geared to helping the less experienced researcher get to analysis as quickly and efficiently as possible. Two of these issues that readers may wish to make further enquiries about are the use of the command files available in Stat/Transfer and optimization types.

There is a programming language in Stat/Transfer that allows the user to use syntax to conduct the processes discussed in this article and other procedures as well. Although this will not be covered here, researchers may wish to refer to the Stat/Transfer manual to consider this option as

they become more familiar with the procedures described below. The use of syntax or command files can be very helpful in keeping logs of what one has done, as well as replicating procedures with the same or different datasets.

Researchers should review the use of optimization types before attempting to transfer files. Optimization is used by Stat/Transfer to produce the smallest possible data file. To do this Stat/Transfer uses a number of variable types to retain the integrity of the original data. In some cases, particularly when changing data from one program format into another, difficulties can arise. For instance, variable names may be truncated or the wrong variable type may result (the usual problem here is between 'float' and 'double'). Thus, the results of all transfers should be checked within the software being used for analysis in order to ensure that variables have retained their structure and that the transfer has not altered the data file in other ways.

III. Practical Example and Method

The basic procedure used is to load the large, unparsed dataset into Stat/Transfer, select the new file name for the parsed file and then to select both the variables and subsample of interest. This procedure will be illustrated using the Public Use Microdata file for the GSS Cycle 13.

The GSS Cycle 13 is on the topic of personal risk and victimization. There are two distinct files – the main file and an incident file. We will only be concerned with the main file in this example. The topic areas covered in the GSS Cycle 13 included perceptions and concern with personal safety, criminal victimization screening, emotional and/or financial abuse by a partner, violence in common-law and marital relationships, emotional and financial abuse of seniors by children and caregivers, and a variety of demographic and classification variables. The main file has over 600 variables. In addition, the sample was increased for Cycle 13 and consequently the sample size is 25,876 observations. The target population was all Canadians over the age of 15, with residents of the territories and full-time residents of institutions being excluded (Statistics Canada 2000a, Statistics Canada 2000b, Statistics Canada 2000c, Statistics Canada 2001).

Let us assume that we wish to conduct an analysis on perceptions of the criminal justice system. We are particularly interested in the perception of males over the age of eighteen. We hypothesize that perceptions of crime and risk, prior victimization and demographic variables will be important factors that need to be include in the analysis. Consequently, we identify from the documentation that we will require the following variables:

a) Unique identifier – RECID
b) Weight – WGHT_PER
c) Perceptions of the criminal justice system – A10a TO A10e, A11a to A11d, A13a to A13b, A14a, A14b
d) Perceptions of crime and safety – A1 to A9
e) Victimization in the last 12 months – MSVIC, MSVIC_X, MSPER, MSPER_X, TOTVIC, TOTVIC_X
f) Demographics – AGEC, SEX, MARSTAT, PRV, URIND, VISMIN, EDU10, NAICS16, INCM, INCMHSD.

In addition to create the sub-sample of interest we will use age greater than 17 (AGEC>17) and only males (SEX=1). Finally, a SPSS format will be selected for the dataset, since the analysis will be conducted in SPSS, however, any format can be chosen to match the users analytical software package of choice.

IV. Setting up the transfer

The first step is to open Stat/Transfer and to name both the file we are starting with (the full file) and the file we will be saving (the parsed file). The place we need to enter information is under the Transfer Tab. This is usually the tab that the program presents as its opening screen. Then select the "Input File Type" of the file we wish to parse down (the full microdata file), in our example the format is SPSS Data file. Also, under the top "File Specification" bar enter the location and name of the file we will be creating; this can easily be done by browsing for the file.

For the file that will be created the steps are the same; select your output file type (again in this example SPSS is used) and select the location of your new data set. Figure 1 shows Stat/Transfer once the data file names have been entered.

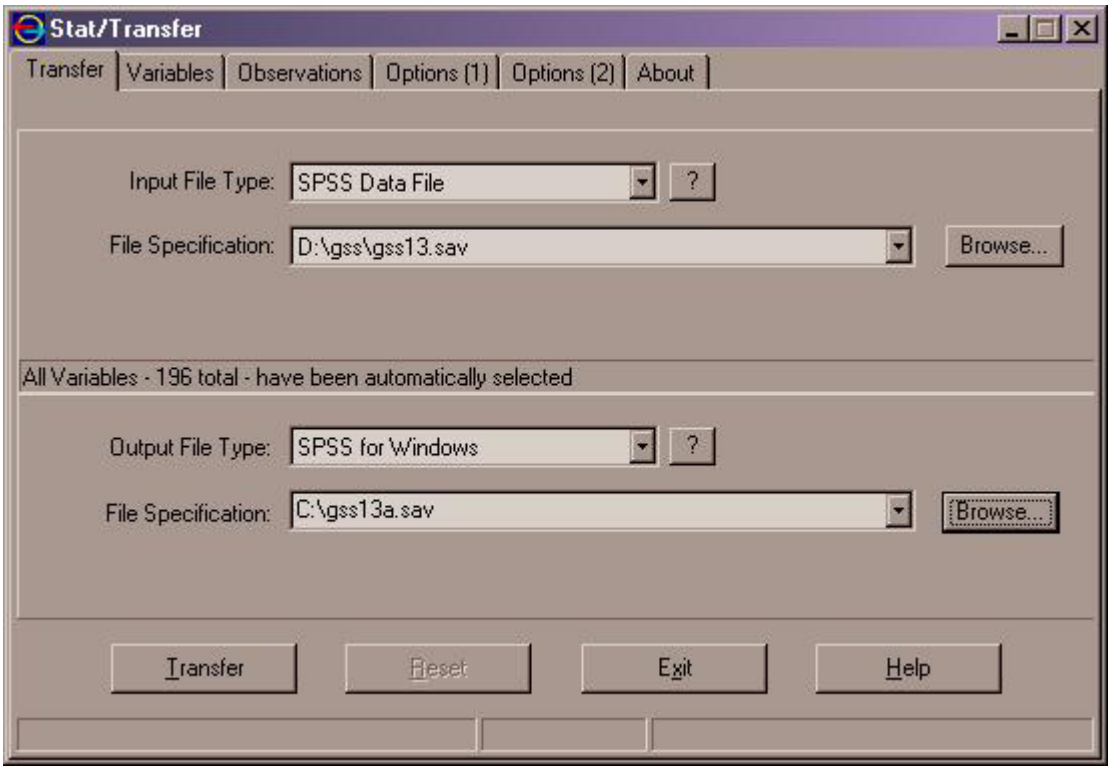


Figure 1

V. Selecting the variables

The next step is to pick the variables that we wish to keep. This involves a number of short steps. First, we need to change to the Variables Tab and then to 'unselect all' variables. This removes all variables from the pre-selected list and allows us to pick only the variables we want. Once we have done this, the screen should appear similar to Figure 2.

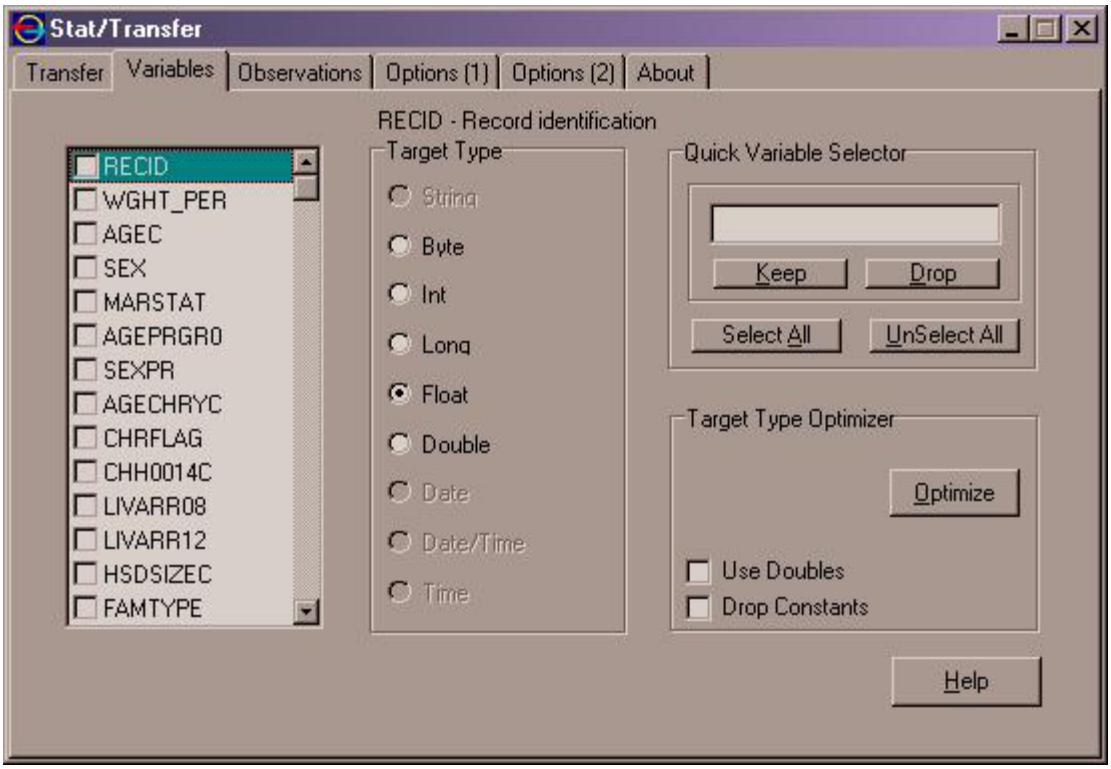


Figure 2

Now we are ready to select the variables of interest. Variables can broadly be divided into two groups – those that have unique names (i.e. RECID) and those that have some commonality in the variable name (i.e. A10a to A10e).

There are two methods of handling those variables that have unique names. First, one may individually select the variables of interest by clicking in the box beside the variable name on the left hand side of the screen. One may alternatively select individual variables by typing the variable name into the box under 'Quick Variable Selector' on the right hand side of the screen. Either one of these methods of selecting variables would be an appropriate technique for the following categories of variables suggested previously– Unique identifier, weight, and demographics.

The process outlined above would be cumbersome for the remaining variables in our example (perceptions of the criminal justice system, perceptions of crime and safety, and victimization in the last twelve months). These variables have commonalties within their names that

can make selecting them much quicker. In the quick selector box the portion of the variable name that contains the commonality is entered, followed by a "wildcard". Wildcards are symbols that denote similar properties of names. For instance to pick all variables that contain A10 in their names, one would type 'A10*' and then select keep. Alternatively we could type 'A1*' and this would keep all variables from A10 to A19. Then we could remove the selection of the A19 variables by typing 'A19*' and then clicking drop. Single characters are denoted by "?" and so to select the variables A1 to A9 we would type 'A?' . To select the victimization variables one would type 'MS*' and 'TOTV*'.

VI. Restricting the sample

Having selected all the variables of interest, it now remains to select the subsample of interest. This is done under the Observations Tab. In the dialogue box, one types in the inclusion criteria. For the GSS example, we would type ' where AGEC > 17 & SEX =1'. The screen should then appear as in Figure 3.

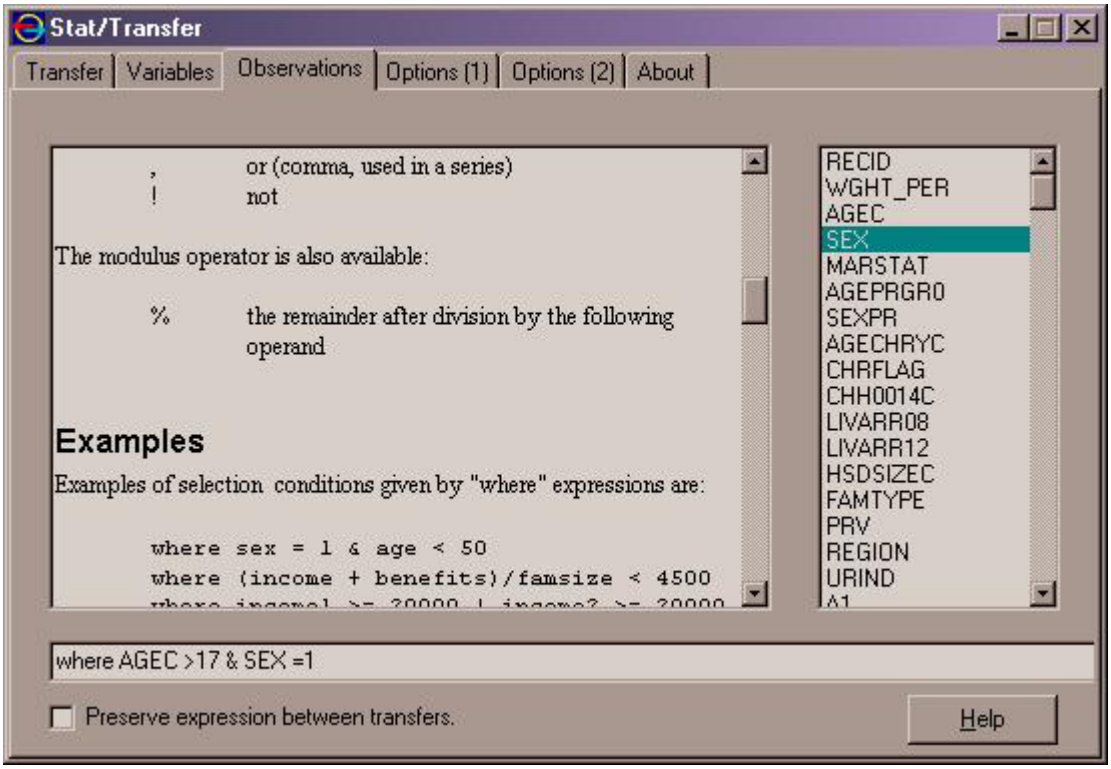


Figure 3

VII. Conclusion

All that remains to be done is to perform the actual transfer and check the results. To perform the transfer, click on the Transfer Tab and verify that nothing has changed about the filenames that were initially entered on this screen. In addition, one can verify the number of

variables to be transferred by observing the notation between the Input and Output File Types fields, in our example, 34 out of 196 variables, have been selected.

After checking that the right number of variables has been specified, click on the Transfer box. The file parsing will now be done and the new file will be saved in the location specified in the output file specification. When the transfer is complete, the word 'finished' will be at the bottom of the screen and the number of observations will be noted. Since we have selected a subsample, this number should be smaller than the actual N of the original sample (the original sample had 25,876 observations, and our new file has 11,088; this is confirmed in Figure 4).

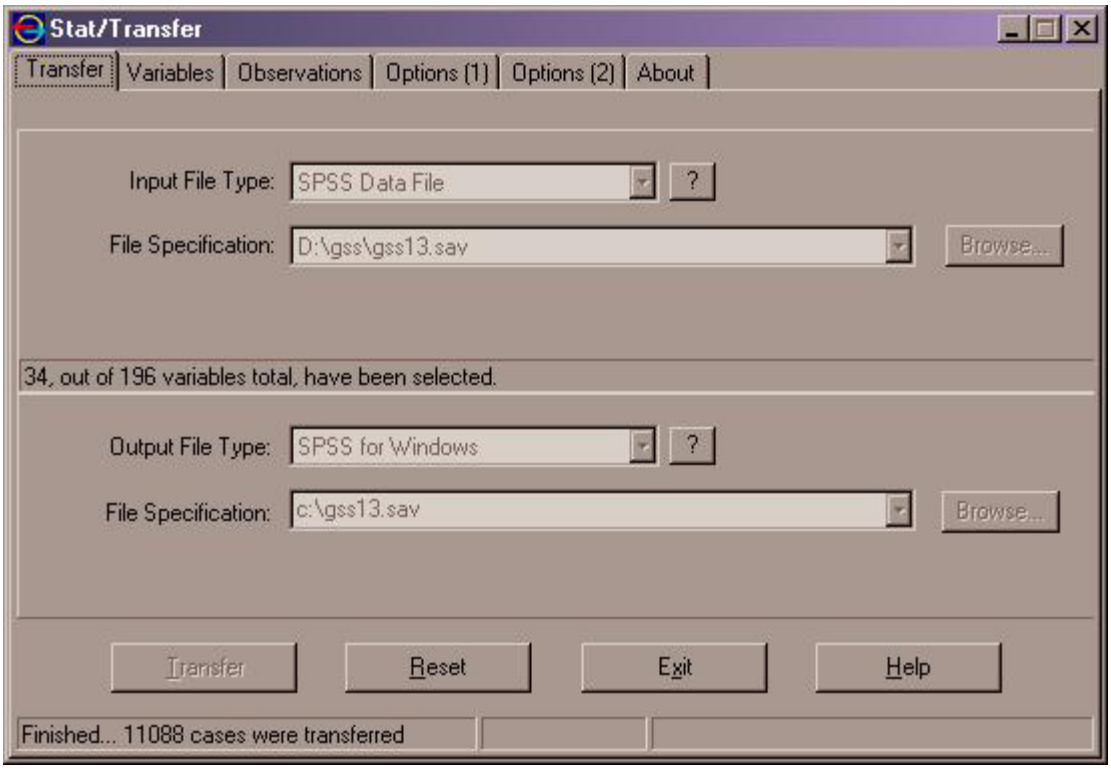


Figure 4

Finally, to check that the file has been parsed correctly, the file should be opened in the statistical software of the users choice and exploratory frequencies and cross-tabs run to verify totals and the correct transfer of variable information. The parsed file may then be used in future analysis and the creation of this parsed file should increase the efficiency of both the hardware and the software.

References

Statistics Canada. 2000a. "The 1999 General Social Survey - Cycle 13 Victimization : Public use microdata file documentation and user's guide." Statistics Canada, Ottawa.

—. 2000b. "The 1999 General Social Survey - Cycle 13 Victimization :Questionnaire Package." Statistics Canada, Ottawa.

—. 2000c. "General Social Survey 1999 - Cycle 13 Victimization [public use microdata file]." Ottawa: Statistics Canada Housing Family and Social Statistics Division.

—. 2001. "A profile of Canadian Victimization: results of the 1999 General Social Survey." Statistics Canada, Ottawa.

Bootstrapping made easy: A Stata ADO File

By Emmanuelle Piérard, Neil Buckley, and James Chowhan

Abstract

This note introduces a Stata command that calculates variance estimates using bootstrap weights. The "bswreg" command is compatible with a wide variety of regression analytical techniques and datasets. This program has been tested and compared against the regression analytical techniques available in bootvare_v20.sas to verify accuracy. National Population Health Survey Cycle 4 data are used for these comparisons. This program provides researchers with an easy and flexible tool that was not previously available.

Introduction

This note introduces a Stata program that calculates variance estimates using bootstrap weights. The main motivation for creating this program was to develop an easy to use and flexible tool within Stata, which can be employed with the bootstrap weights that are made available with most of Statistic Canada's micro-data sets. The use of these bootstrap weights allows researchers to make use of complex survey design information and calculate reliable variance estimates, while preserving the confidentiality of respondents [Yeo et al., 1999].

The program is compatible with a wide variety of regression techniques. This program builds on the linear and logistic regressions that were introduced in "Bootvar" to include a variety of regression techniques.[1] This note discusses the programs unique features, presents the strengths and weaknesses of the program, and describes a simple test used to verify the accuracy of this new Stata program relative to BOOTVARE_V20.SAS.

II. Standard bootstrap

Most of Statistics Canada's surveys use a complex design to draw a representative sample from the population of interest. The resulting micro-data sets are available with bootstrap weights that can be used to account for the complex survey design. The use of these bootstrap weights allows researchers to calculate reliable variance estimates. The bootstrap variance estimator for $\hat{\theta}$, used in this program, is given by [Yeo et al., 1999; 3]:

$$v_B\left(\hat{\theta}\right) = \frac{1}{B}\sum_b \left(\hat{\theta}^*_{(b)} - \hat{\theta}^*_{(.)}\right)^2 \qquad (1)$$

where $\hat{\theta}^*_{(.)} = \left(1/B\right)\sum_b \hat{\theta}^*_{(b)}$,

[1] "The Bootvar program is available in both SAS and SPSS formats. It is made up of a macro that computes variances for totals, ratios, and differences between ratios, and for linear and logistic regression. The Bootvar program is provided with bootstrap weights and a document explaining how to modify and use the program to suit user's needs." [Statistics Canada, 2002; 39]

and $\hat{\theta}$ is an estimate of interest, and B is the number of bootstrap samples.

III. How to

The Stata program is easy to use: simply copy the "bswreg.ado" and "bswreg.hlp" files, which are described in Appendix I, to your Stata ADO folder[2], and then employ the program by using the following syntax command:

bswreg depvar [varlist] weighttype=full_sample_weight [if exp] [in range],
cmd(STATA_regression_command) [cmdops(options_for_regression_command)]
bsweights(bootstrap_weights_varlist) [level(integer)] [bsci] [saving(path_and_filename[,replace])];

For example, suppose a researcher wishes to run an ordinary least squares regression of height on a list of provincial dummies, education dummies, age, and gender using the National Population Health Survey (NPHS) Cycle 4[3]; they want to save the bootstrap output table as a data file in memory (including bootstrap coefficients, standard errors and other inference statistics); and they choose to use all of the 500 available bootstrap weights. The command code line would be as follows:[4]

bswreg height nfld pei ns nb qc on man sask alb lshs someps ugrad
agesq age gender [pw=wt60lf], cmd(reg) bsweights(bsw1-bsw500)
level(95) bsci saving(c:\temp\bswdata.dta, replace); (2)

This command assumes that the appropriate bootstrap weights and the data-file have already been merged accurately by the appropriate unique identifier (in this example, NPHS Cycle 4, where the unique identifiers are realukey and personid). This program does not require the bootstrap weights to have any naming scheme. Further, the bswreg command allows for the use of options. The program has several options available:

cmd: specifies the Stata regression command to bootstrap. This is a required option. The following regression commands have been tested explicitly: regress, logit, probit, tobit, ologit, oprobit, biprobit, mlogit, qreg, glm, intreg, boxcox, (basically any single stage estimation technique should work with this program) and non-twostage "xt" commands that support weights.

[2] Type the command "adopath" at the Stata command prompt for a list of ado directory paths in which to place this program. Further, the researcher will need to "set" the "matsize" and "memory" size to levels appropriate to the computer and dataset that they are using.

[3] The NPHS Cycle 4 (2000-2001) Longitudinal Master File sample is reduced from 17,276 to 12,439 by only including respondents in cycles 1 to 4 and records without missing observations. The regression's dependent variable is height, this variable is a scale that standardizes the metric and imperial systems. For height, a value of 50 on the scale is equivalent to 5'0" (60 inches) (151.1 to 153.6 cm), and an increase of one in the scale is equivalent to an inch. The provincial dummy variables include: nfld pei ns nb qc on man sask alb, where bc is the omitted province. The education variables include: less than high school (lshs), high school graduates (hsgrad--omitted), some post-secondary (someps), and university graduates (ugrad). The age variables include age and age-squared. The gender variable is equal to 1 if male and 2 if female.

[4] The results from this regress are presented in Appendix II.

bsweights: specifies a variable list of the bootstrap weight names. This is a required option. For instance, if your bootstrap weights are named bsw1 to bsw500, you could specify the option as bsweights(bsw1-bsw500).

cmdops: specifies the options you wish to use on the Stata regression command provided in cmd(). Some options are useful and others are meaningless in a bootstrap weighting context. For instance, if you wish to run the REGRESS command with no constant then use the cmd(regress) cmdops(noconstant) options. Options like robust are meaningless in this context since the command computes bootstrap weighted standard errors not robust ones.

level: specifies the confidence level, in percent, for confidence intervals. The default is level(95).

bsci: specifies that the confidence intervals be calculated from the raw bootstrapped distribution of coefficients rather than using the standard formula based on the bootstrapped standard error and the normal distribution.[5]

saving: saves the bootstrap statistics in a separate Stata dataset file that can later be loaded and used by other .DO and .ADO files. If you do not specify an extension, .dta will be assumed. Include the replace option to overwrite an existing file.

IV. Unique features

The bswreg command provides researchers with a flexible tool that was not previously available. Reliable variance estimates can be generated to accompany analytical techniques from ordinary least squares, probits, quantile regressions to random-effects tobit models when used congruently with the bswreg command to produce design-based variance estimates.

The command has a "built-in" help feature that provides basic assistance on predefined search topics. By typing "help bswreg" at the command prompt, the researcher will display a description of the procedure, a list of outputted variables, and a list of example code that employ the bswreg command.

Due to the breadth of the analytical techniques that are compatible with the bswreg command, a semi-sophisticated error-resolving algorithm is required. Specifically, for estimation techniques that require the model to iterate toward convergence, there is the possibility that the model will not converge for every set of bootstrap weights selected. Therefore, the command bswreg has been designed to default to the actual number of successful bootstrap procedures, thereby avoiding errors, such as convergence errors.[6]

[5] This option is provided for users that may have a theoretical reason for employing the confidence intervals derived from the bootstrapped distribution of coefficients.

[6] For example, suppose 500 bootstrap sample weights have been selected to run an iterative procedure using maximum likelihood (random-effects or population-averaged logit models) and x regressions fail to converge due to the nature of x bootstrap sample weights, then 500-x bootstrap sample weighted regressions were successful, and are used to generate the bootstrap variance estimator. BSWREG output provides a count of the number of successful iterations completed.

This program is also designed to deal with bootstrap regressions that fail. There are two main examples where this could take place. First, when a zero bootstrap weight corresponds perfectly with a small sample size on a discrete variable, the result is that the variable is dropped. Once a variable is dropped the regression output from this particular bootstrap sample will not have an identical number of variables as all other "successful" bootstrap samples. This is problematic for the calculation of the variance estimates, and as a result these bootstrap samples are dropped. Second, in a case where the sample would be very small, it is possible that some of the bootstrap samples weight a majority of the sampled observations with a weight of zero. The resulting sample could be too small to perform the estimation procedure, and again this sample would be removed from the bootstrapping procedure. This results in the total number of bootstrap regressions reported at the end of the program being smaller than the original number specified in the bswreg command.

The output generated at the end of the "bswreg" procedure includes the total bootstraps completed, name of variable, coefficient estimate, and the bootstrap standard error of coefficient, z-stat, p-value, lower and upper 95% confidence intervals (by default) assuming a normal distribution, and the option "bsci" produces lower and upper 95% confidence intervals that are generated using the raw bootstrap distribution itself. In terms of the output generated, the main distinguishing feature of this program relative to other programs, is that it has the option to produce two sets of confidence intervals: the first assumes a normal distribution and the second uses the raw bootstrap-distribution.

Any command that requests multiple regressions by a variable that parses the data will work with bswreg command. For example, the "by var_name:" prefix to a regression command will work in the bswreg framework. Continuing the previous example, suppose the researcher wanted to run OLS by gender separately, wanted to only use the bootstrap weights bsw100 to bsw400, a confidence interval of 99%, and the bootstrap distribution, the syntax would be as follows:

bysort gender: bswreg height nfld pei ns nb qc on man sask alb
lshs someps ugrad agesq age [pw=wt60lf], cmd(reg) (3)
bsw(bsw100-bsw400) level(99) bsci;

The bswreg command focuses on reliable variance estimation and related statistics and does not calculate any ancillary statistics. Ancillary statistics are not calculated due to the potential breadth of estimates that could be considered, given the scope of the compatibility of this program. For example, odds ratios for the logit regression, or the change of probability for dummy variables for the probit procedures, are omitted.

The program can be used to calculate various summary statistics such as frequencies, means, and ratios. See Appendix III for examples of how these statistics can be calculated.

A note with regards to testing hypotheses: the bswreg program stores the coefficients in the standard e(b) matrix and the bootstrapped variance-covariance matrix in e(V). This means that the "test" command can be used directly after "bswreg" to conduct tests. However, all tests will use the asymptotic normal distribution instead of the t-distribution. Thus, all tests run with the "test" command will be WALD style tests based on the chi squared distribution. The program prevents

researchers from running F-tests, which in any event would be incorrect due to the asymptotic nature of the bootstrapping technique.

V. Accuracy tests

This program has been tested to verify its accuracy. The program bootvare_v20.sas was used as the benchmark for all tests and comparisons. The ordinary least squares regression procedure in bootvar was duplicated using the bswreg command. Since bootvar has been specifically designed for the NPHS, a longitudinal dataset including Cycle 4 were employed in the tests. Further, in all tests, all 500 bootstrap weights were used for each procedure. See Appendix II for the results. In addition, frequency, mean, and ratio summary statistics were duplicated using a comparable regression framework--see Appendix III.

The results generated by the two programs are identical for the Ordinary Least Squares example. Further, all of the variance related estimates, such as the standard error, z-stat, p-value, and lower and upper bounds (assuming a normal distribution), were identical. These results indicate that this program has a substantial degree of reliability.

VI. Concluding remarks

This program focuses on reliable variance estimation across a wide range of analytical techniques. The Stata program discussed in this note can be used to calculate variance estimates using bootstrap weights across a wide spectrum of datasets (weights are currently available for NLSCY, NPHS, SLID, and YITS)[7]. The accuracy of this program has been verified through comparability tests against other available analytical programs. Researchers who use Stata now have available for their use a flexible tool that is easy to use and accurate.

[7] The following surveys are core datasets that are available through the Statistics Canada Research Data Centre program: National Longitudinal Survey of Children and Youth (NLSCY), National Population Health Survey (NPHS), Survey of Labour and Income Dynamics (SLID), and Youth in Transition Survey (YITS).

References

Statistics Canada. 2002. "Population Health Surveys Program: National Population Health Survey, Cycle 4 (2000 – 2001), Household Component Longitudinal Documentation." Health Statistics Division. Ottawa.

Yeo, Douglas, Harold Mantel, and Tzen-Ping Liu. 1999. "Bootstrap Variance Estimation For the National Population Health Survey." American Statistical Association: Proceedings of the Survey Research Methods Section. Baltimore, August.

Appendix I

Ado File:

```
*                                    WARNING

* The authors are the owners of all intellectual
* property rights (including copyright) in this software.  Subject to the terms below,
* you are granted a non-exclusive and non-transferable license to use this software.
*
* This software is provided "as-is", and the owner makes no warranty, either express
* or implied, including but not limited to, warranties of merchantability and fitness
* for any particular purpose.  In no event will the owner be liable for any indirect,
* special, consequential or other similar damages.  This agreement will terminate
* automatically without notice to you if you fail to comply with any term of this
* agreement.

* TO CHANGE THE DECIMAL DISPLAY FORMAT OF THE BOOTSTRAPPED OUTPUT SEARCH FOR THE "FORMAT" COMMAND NEAR THE
BOTTOM OF THIS PROGRAM;

program define bswreg, eclass sortpreserve byable(recall)
* August 8th, 2003 Pierard, Buckley, Chowhan

# delimit;
version 7.0;

syntax varlist(numeric) [aweight pweight fweight iweight] [if] [in], cmd(string) [cmdops(string)]
  BSWeights(varlist numeric) [Level(integer 95)] [BSci] [SAVing(string)];
*This sets the touse variable = 1 if observation is in our sample;
marksample touse;
*Error check to make sure a weight was used;
if "`weight'"==""
  {;
  noi di in red "BSWREG error: You must specify a weight!";
  exit;
  };

quietly
{;
*Preserve the original dataset and set parameter values and setup temporary matrices;
preserve;
set more 1;
tempvar esamplevar;
tempname bhat bsVC bsbhat bsbetas;

*The next line runs the wanted regression and checks for errors;
capture `cmd' `varlist' [`weight'`exp'] if `touse' `in', `cmdops';
if _rc ~= 0
   {;
   noi di in red " ";
   noi di in red "Error doing: `cmd' `varlist' [`weight'`exp'] `if' `in', `cmdops'";
   noi di in red " ";
```

```
   noi di in red "The regression command you have typed in resulted in an error, please investigate";
   noi di in red "this error outside of the 'bswreg' program by typing in the regression command
itself";
   noi di in red "with the options you specified.";
   noi di in red " ";
   exit;
   };
*The next line runs the wanted regression and we store the coefficients in a matrix for later use;
`cmd' `varlist' [`weight'`exp'] if `touse' `in', `cmdops';
gen `esamplevar'=e(sample);
*e(b) is a 1x(k+1) coefficient vector if the model has a constant and k is the number of variables
other than the constant;
matrix `bhat'=e(b);
matrix `bsVC'=e(V);
*we store the variable names of the regressors and the number of regressors in local macros;
local _varnames : colfullnames(`bhat');
local _k=colsof(`bhat')-1;
local _k1=`_k'+1;
*Generate concatenated list of placeholder regressor variable names xc1-xck1, later to be turned into
variables;
local _xclist="";
forvalues _i = 1/`_k1'
   {;
   local _xclist `_xclist' _xc`_i';
   };
*We assigned these placeholder variable names to the regressors in the coefficient vector;
matrix colnames `bhat' = `_xclist';
*Each "true estimate of beta" is saved under it's own variable name;
svmat double `bhat', name(col);
matrix colnames `bhat' = `_varnames';

*Realboot is the actual number of successful bootstrap regressions run in case we get any
convergence/regression errors etc., it starts off at the specified number of bootstrap weights;
local _realboot: word count `bsweights';
noi di " ";

*The main bootstrap loop will run with each bootstrap weight in the supplied bsweight varlist and
exit with the matrix named BETAS containing all the bootstraps of our coefficients, a (boot)x(k+1)
dimensional matrix;
local _i 1;
*Start of bootstrap loop;
foreach bswvar of local bsweights
 {;
 *Display notice of number of completed bootstraps every time 50 are completed;
 if mod(`_i',50)==0
  {;
  noi di `_i' " bootstraps completed";
  };
 *Run the regression with the chosen set of bootstrap weights, only use the coefficients if there are
no errors;
 capture `cmd' `varlist' [`weight'=`bswvar'] if `touse' `in', `cmdops';
 if _rc==0
  {;
   *Store coefficients in the bootstrap matrix;
   matrix `bsbhat'=get(_b);
   *bsbhat is a 1x(`k'+1) (row) vector if the model has a constant.  Need to transpose;
   matrix `bsbhat'=`bsbhat'';
   *If we have the proper number of coefficients then add them to the bootstrap matrix, otherwise do
not add them (this most likely arises due to a regressor being dropped due to multicollinearity;
   if rowsof(`bsbhat')==`_k1'
        {;
        *If we are on the first bootstrap then create the bsbetas matrix, otherwise append to it;
        if `_i'==1
           {;
           matrix `bsbetas'=(`bsbhat');
           };
        else
           {;
           matrix `bsbetas'=(`bsbetas',`bsbhat');
           };
        };
```

```
   else
        {;
        matrix drop `bsbhat';
        local _realboot=`_realboot'-1;
        noi di "Bootstrap #`_i' has been dropped for not having the correct number of coefficients";
        };
  };
 else
  {;
  local _realboot=`_realboot'-1;
  noi di "bootstrap #`_i' has been dropped due to an error estimating the regression";
  };
 local _i=`_i'+1;
 };
*End of bootstrap loop;

*All the bootstraps have been completed now calculate the new standard errors and display relevant
statistics;
*We must transpose the matrix to make each row now, then column, a new variable;
matrix `bsbetas'=`bsbetas'';
*Generate concatenated list of colnames, later to be turned into variables;
local _xvlist="";
forvalues _i = 1/`_k1'
   {;
   local _xvlist `_xvlist' _xv`_i';
   };
*Calls each row of the matrix by the name of the independent variable it corresponds to (we call them
_xv'_i' so that they are not mixed up with the "real" variables);
matrix colnames `bsbetas'= `_xvlist';
*Separate each column as a new variable.  The format of the data must be specified.  It renames each
variable by the name of the column;
svmat double `bsbetas', name(col);

*Generate the bootstrapped variance-covariance matrix, you can access this in e(V) after running the
BSWREG ado file;
forvalues _i = 1/`_k1'
 {;
 forvalues _j = 1/`_k1'
  {;
    correlate _xv`_i' _xv`_j', covariance;
    matrix `bsVC'[`_i',`_j'] = ((`_realboot'-1)/`_realboot')*r(cov_12);
  };
 };

*Generate the standard deviation, t-stat, conf. int. etc. for each variable;
tempvar _bsobs _uniqobs _coefnum;
gen `_bsobs'=_n;
forvalues _i = 1/`_k1'
 {;
 sum _xv`_i';
 * Like the SAS bootvar program, we use (boot-1)/boot because variance and standard error have different
denominators;
 gen _sdx`_i'=sqrt(((`_realboot'-1)/`_realboot')*r(Var)) in 1/1;
 gen _t`_i'=_xc`_i'/_sdx`_i' in 1/1;
 gen _abst`_i'=abs(_t`_i') in 1/1;
 gen _p`_i'=2*norm((-1)*_abst`_i') in 1/1;
if "`bsci'"==""
 {;
   gen _low`level'`_i'=_xc`_i'-invnorm(1-((1-(`level'/100))/2))*_sdx`_i';
   gen _high`level'`_i'=_xc`_i'+invnorm(1-((1-(`level'/100))/2))*_sdx`_i';
 };
 if "`bsci'"=="bsci"
   {;
   sort _xv`_i';
   local _obslow= max(1,round(((1-(`level'/100))/2)*`_realboot',1));
   local _obshigh= max(1,round((1-((1-(`level'/100))/2))*`_realboot',1));
   local _obslow2= _xv`_i'[`_obslow'];
   local _obshigh2= _xv`_i'[`_obshigh'];
   sort `_bsobs';
   gen _low`level'`_i'= `_obslow2' in 1/1;
   gen _high`level'`_i'= `_obshigh2' in 1/1;
```

```
   };
 };

*Assign each coefficient its true regressor name stored at the beginning of this program;
local _i=1;
foreach _curname in `_varnames'
 {;
 gen str10 _xname`_i'="`_curname'";
 local _i=`_i'+1;
 };

*Reshape the data so that the bootstrapped stats can be displayed easily, and then display the
results;
keep _xname* _xc* _sdx* _t* _p* _low`level'* _high`level'*;
drop if _n>1;
gen `_uniqobs'=1;

reshape long _xname _xc _sdx _t _p _low`level' _high`level', i(`_uniqobs') j(`_coefnum');
*The %9.4f tells stata to display the bootstrapped results to 6 decimals using 15 numbers total --
this can be changed to suit tastes;
format _xc _sdx _t _p _low`level' _high`level' %11.6f;

*creates nice labels for variables
label var _xname "Name of variable";
ren _xname Var_name;
label var _xc "Coefficient estimate";
ren _xc Coef;
label var _sdx "Bootstrap standard error of coefficient";
ren _sdx BS_se;
label var _t "Bootstrap z-statistic";
ren _t BS_zstat;
label var _p "Bootstrap p-value";
ren _p BS_pvalue;
if "`bsci'"==""
 {;
 label var _low`level' "Bootstrap lower 95% confidence interval assuming a normal distribution";
 label var _high`level' "Bootstrap upper 95% confidence interval assuming a normal distribution";
 };
if "`bsci'"=="bsci"
 {;
 label var _low`level' "Bootstrap lower 95% confidence interval using bootstrap sample distribution";
 label var _high`level' "Bootstrap upper 95% confidence interval using bootstrap sample
distribution";
 };
ren _low`level' BS_cilow`level';
ren _high`level' BS_ciup`level';

*Display RESULTS!;
noi display in green "Results from BSWREG";
noi display in green "--------------------";
noi display in green " ";
if "`bsci'"=="bsci"
 {;
 noi display in green "* The confidence intervals below are based on the bootstrapped distribution";
 };
else noi display in green "* The confidence intervals below are based on the normal distribution";
noi display in green " ";
noi list Var_name Coef BS_se BS_zstat BS_pvalue BS_cilow`level' BS_ciup`level', nodisplay noobs;
noi di " ";
noi di "Total bootstraps completed: `_realboot'";

*Set the eclass variables like the coefficients and the variance-covariance matrix into their
appropriate matrices so that F-tests and the like can be run;
*If you wish the TEST command to produce F-tests after the BSWREG command then add ",
dof(`_realboot')" to the line below;
estimates post `bhat' `bsVC';

*Save the bootstrap raw data is the "SAVING" option has been used;
if "`saving'"~=""
 {;
  drop _*;
```

```
  save "`saving'", `replace';
 };

*Restore the original dataset
restore;

};
end;
```

BSWREG Help File

```
{smcl}
{* 8August2003 Pierard/Buckley/Chowhan}
{hline}
help for {hi:BSWREG}
{hline}
{title:BSWREG - uses bootstrap weights to calculate standard errors in models involving complex
survey data.}

{p 8 13}{cmd:bswreg} depvar [varlist] {it:weighttype}={it:full_sample_weight} [{cmd:if} {it:exp}]
[{cmd:in} {it:range}][{cmd:,} {cmd:cmd(}{it:STATA_regression_command}{cmd:)}
[{cmd:cmdops(}{it:options_for_regression_command}{cmd:)}]
  {cmdab:bsw:eights(}{it:bootstrap_weights_varlist}{cmd:)} [{cmd:level(}{it:integer}{cmd:)}]
[{cmd:bsci}] [{cmdab:sav:ing(}{it:path_and_filename}[{cmd:,replace}]{cmd:)}];

{p} {cmd:cmd()} and {cmd:bsweights()} are required options for the {cmd:BSWREG} command.
{p} {cmd:by ...: } and {cmd:bysort ...:} can be used with {cmd:BSWREG}. See help {help by}.
{p} {cmd:aweight}s, {cmd:fweight}s, {cmd:iweight}s, and {cmd:pweight}s are allowed as long as the
      given regression command is compatible with them. See help {help weights}.
{p} As {cmd:BSWREG} is an eclass STATA program, it provides STATA with the {cmd:e(b)} coefficient
vector and the {cmd:e(V)} bootstrapped variance-covariance matrix.
    The {cmd:test} command can be used immediately following the {cmd:BSWREG} command to conduct Wald
tests based on the chi-squared distribution.

{inp:The software is provided "as-is" and the authors are not responsible for any misuse.}
{title:Description}
(used to calculate regression statistics using Statistics Canada's bootstrap weights)

{p}{cmd:bswreg} runs a number of regressions, each with a particular bootstrap
weight so that bootstrapped standard errors on the coefficients can be calculated
and displayed. Use of bootstrap weights is recommended for calculating reliable
standard errors, confidence intervals etc. on data from complex
household surveys.

The user provides the names of the bootstrap weights to the {cmd:BSWREG} command
in the {cmdab:bsw:eights(varlist)} option. You must already have the appropriate
bootstrap weights merged into your datafile for this command file to work.  NPHS
merges on REALUKEY and SLID merges on PERSONID. Below is a sample .DO file that
merges NPHS bootstrap weights into a datafile named data.dta:

{inp:use data.dta, replace"}
{inp:sort realukey"}
{inp:save data.dta, replace"}
{inp:use bootstrap/sas_bs_wt_1_4.dta, replace"}
{inp:destring realukey, replace"}
{inp:sort realukey"}
{inp:merge realukey using data.dta"}
{inp:keep if _merge==3"}

{title:Options}

{p 0 4}{cmd:cmd(}{it:STATA_regression_command}{cmd:)} specifies the Stata regression command to
bootstrap. This is a {cmd:required} option. "regress", "probit" and "logit" are a few possibilities.

{p 0 4}{cmd:bsweights(}{it:varlist}{cmd:)} specifies a variable list of the bootstrap weight names.
This is a {cmd:required} option. For instance, if your bootstrap weights are named bsw1 to bsw500,
you may wish to use the
{cmd:bsweights(bsw1-bsw500)} option.
```

```
{p 0 4}{cmd:cmdops(}{it:options_for_regression_command}{cmd:)} specifies the options you wish to use
on the Stata regression command provided in {cmd:cmd()}. Some options are useful and others are
meaningless in a bootstrap weighting context.
For instance, if you wish to run the REGRESS command with no constant then use the {cmd:cmd(regress)
cmdops(noconstant)} options.  Options like {cmd:robust} are meaningless in this context since the
command computes bootstrap weighted
standard errors not robust ones.

{p 0 4}{cmd:level(}{it:integer}{cmd:)} specifies the confidence level, in percent,
for confidence intervals.  The default is {cmd:level(95)}. See help {help level}.

{p 0 4}{cmd:bsci} specifies that the confidence intervals be calculated from the raw bootstrapped
distribution of coefficients rather than using the standard formula
based on the bootstrapped standard error and the normal distribution.

{p 0 4}{cmd:saving(}{it:filename}[{cmd:,replace}]{cmd:)} saves the bootstrap statistics in a
separate Stata dataset file that can later be loaded and used by other .DO and .ADO files.
If you do not specify an extension, {cmd:.dta} will
be assumed. Include the {cmd:,replace} option to overwrite an existing file.

{title:Outputed variables}

{inp: Var_name:}  This is the STATA variable name of the regressor.
{inp: Coef:}      This is the coefficient from the specified regression.
{inp: BS_se:}     This is the new standard error of the coefficient,
            calculated using bootstrap weights.
{inp: BS_zstat:}  This is the new z-stat of the coefficient,
            calculated as the coefficient divided by the bootstrapped standard error.
{inp: BS_pvalue:} This is the new p-value of the coefficient,
            calculated using the z-statistic.
{inp: BS_cilow95n:} This is the lower (level)% confidence interval around the coefficient
            using the bootstrapped std. error.
{inp: BS_ciup95n:} This is the upper (level)% confidence interval around the coefficient
            using the bootstrapped std. error.

{title:Examples}

{p 8 12}{inp:. bswreg income education rural [aw=wt] if married==1, cmd(regress) bsw(bsw1-bsw500)}

{p 8 12}{inp:. bswreg employed education rural [aw=wt66], cmd(probit) bsw(bsw50-bsw100)}

{p 8 12}{inp:. bysort maritalstatus: bswreg income education rural [aw=wt], cmd(reg) bsw(bsw1-
bsw500)}
  {inp:cmdops(noconstant) level(99) bsci saving(c:\data\bsw1.dta,replace)}
```

Appendix II

BSWREG Results:

```
. reg height  nfld pei ns nb qc  on man sask alb lshs  someps ugrad agesq age gender [pw=wt60lf];
(sum of wgt is   2.6597e+07)

Regression with robust standard errors                Number of obs =   12439
                                                      F( 15, 12423) =  279.22
                                                      Prob > F      =  0.0000
                                                      R-squared     =  0.4753
                                                      Root MSE      =   4.208
------------------------------------------------------------------------------
             |               Robust
      height |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        nfld |  -.2922724   .2449279    -1.19   0.233    -.7723689    .1878242
         pei |  -.3285292   .2671396    -1.23   0.219    -.8521642    .1951058
          ns |  -.5414279   .2420961    -2.24   0.025    -1.015974   -.0668821
          nb |  -.5146936   .2386633    -2.16   0.031    -.9825106   -.0468766
          qc |  -.7923805   .1886131    -4.20   0.000    -1.162091   -.4226696
          on |  -.4070276   .1911941    -2.13   0.033    -.7817977   -.0322575
         man |   .2473519   .2449153     1.01   0.313      -.23272    .7274238
        sask |   .1010543   .2473725     0.41   0.683    -.3838343    .5859428
         alb |   .2328238   .2169931     1.07   0.283    -.1925163    .6581638
        lshs |   3.759076   .1975814    19.03   0.000     3.371786    4.146366
      someps |   3.506946   .1773476    19.77   0.000     3.159318    3.854575
       ugrad |   3.155798   .1650738    19.12   0.000     2.832228    3.479368
       agesq |  -.0047157   .0001675   -28.16   0.000     -.005044   -.0043874
         age |    .465882   .0161484    28.85   0.000     .4342285    .4975354
      gender |  -4.695284   .1128784   -41.60   0.000    -4.916543   -4.474025
       _cons |   50.90797    .503058   101.20   0.000      49.9219    51.89404
------------------------------------------------------------------------------

. bswreg height  nfld pei ns nb qc  on man sask alb lshs  someps ugrad agesq age gender [pw=wt60lf],
cmd(reg) bsw(bsw1-bsw500)  level(95) bsci saving(c:\temp\bswdata.dta, replace);

50 bootstraps completed
100 bootstraps completed
150 bootstraps completed
200 bootstraps completed
250 bootstraps completed
300 bootstraps completed
350 bootstraps completed
400 bootstraps completed
450 bootstraps completed
500 bootstraps completed
Results from BSWREG
-------------------

* The confidence intervals below are based on the bootstrapped distribution

  Var_name         Coef        BS_se     BS_zstat    BS_pvalue   BS_cilow95    BS_ciup95
      nfld    -0.292272     0.211569    -1.381455     0.167139    -0.708804     0.145642
       pei    -0.328529     0.241579    -1.359923     0.173854    -0.780092     0.131531
        ns    -0.541428     0.206423    -2.622899     0.008719    -0.968447    -0.145737
        nb    -0.514694     0.231676    -2.221613     0.026309    -1.042424    -0.094749
        qc    -0.792381     0.155897    -5.082725     0.000000    -1.079814    -0.477167
        on    -0.407028     0.161284    -2.523667     0.011614    -0.711807    -0.081808
       man     0.247352     0.203860     1.213342     0.224999    -0.162946     0.623552
      sask     0.101054     0.218886     0.461676     0.644314    -0.360283     0.520983
       alb     0.232824     0.193489     1.203289     0.228864    -0.173770     0.640249
      lshs     3.759076     0.171806    21.879725     0.000000     3.417291     4.105172
    someps     3.506946     0.173803    20.177696     0.000000     3.194609     3.866022
     ugrad     3.155798     0.158046    19.967600     0.000000     2.859345     3.466744
     agesq    -0.004716     0.000158   -29.844374     0.000000    -0.005035    -0.004413
       age     0.465882     0.015283    30.483185     0.000000     0.437043     0.496086
    gender    -4.695284     0.095502   -49.164188     0.000000    -4.879501    -4.502970
     _cons    50.907968     0.435441   116.911263     0.000000    50.092201    51.765503
 Total bootstraps completed: 500
```

Bootvare_v20 Results:

The REG Procedure
Dependent Variable: height
Weight: WT60LF

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	15	426030656	28402044	750.15	<.0001
Error	12423	470359719	37862		
Corrected Total	12438	896390375			

Root MSE	194.58162	R-Square	0.4753
Dependent Mean	55.33538	Adj R-Sq	0.4746
Coeff Var	351.64051		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	50.90797	0.21433	237.52	<.0001
nfld	1	-0.29227	0.30121	-0.97	0.3319
pei	1	-0.32853	0.56080	-0.59	0.5580
ns	1	-0.54143	0.23678	-2.29	0.0222
nb	1	-0.51469	0.25727	-2.00	0.0455
qc	1	-0.79238	0.13134	-6.03	<.0001
on	1	-0.40703	0.12280	-3.31	0.0009
man	1	0.24735	0.22511	1.10	0.2719
sask	1	0.10105	0.23464	0.43	0.6667
alb	1	0.23282	0.16058	1.45	0.1471
lshs	1	3.75908	0.11369	33.06	<.0001
someps	1	3.50695	0.11898	29.48	<.0001
ugrad	1	3.15580	0.11630	27.14	<.0001
agesq	1	-0.00472	0.00009228	-51.10	<.0001
age	1	0.46588	0.00821	56.78	<.0001
gender	1	-4.69528	0.07566	-62.06	<.0001

Variance estimation using 500 bootstraps for a Regression

Dependent variable: height

Obs	beta	bhat	bs_var	bs_sd	bs_cv	ci195	ciu95
1	Intercept	50.9080	0.18961	0.43544	0.86	50.0545	51.7614
2	nfld	-0.2923	0.04476	0.21157	72.39	-0.7069	0.1224
3	pei	-0.3285	0.05836	0.24158	73.53	-0.8020	0.1450
4	ns	-0.5414	0.04261	0.20642	38.13	-0.9460	-0.1368
5	nb	-0.5147	0.05367	0.23168	45.01	-0.9688	-0.0606
6	qc	-0.7924	0.02430	0.15590	19.67	-1.0979	-0.4868
7	on	-0.4070	0.02601	0.16128	39.62	-0.7231	-0.0909
8	man	0.2474	0.04156	0.20386	82.42	-0.1522	0.6469
9	sask	0.1011	0.04791	0.21889	216.60	-0.3280	0.5301
10	alb	0.2328	0.03744	0.19349	83.11	-0.1464	0.6121
11	lshs	3.7591	0.02952	0.17181	4.57	3.4223	4.0958
12	someps	3.5069	0.03021	0.17380	4.96	3.1663	3.8476
13	ugrad	3.1558	0.02498	0.15805	5.01	2.8460	3.4656
14	agesq	-0.0047	0.00000	0.00016	3.35	-0.0050	-0.0044
15	age	0.4659	0.00023	0.01528	3.28	0.4359	0.4958
16	gender	-4.6953	0.00912	0.09550	2.03	-4.8825	-4.5081

Appendix III

Frequency Tables:

```
. tab nfld [aw=wt60lf]

       nfld |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 | 12215.0502       98.20       98.20
          1 | 223.949797        1.80      100.00
------------+-----------------------------------
      Total |      12439      100.00


. reg nfld [aw=wt60lf]
(sum of wgt is   2.6597e+07)

      Source |       SS       df       MS              Number of obs =   12439
-------------+------------------------------           F(  0, 12438) =    0.00
       Model |        0.00     0           .           Prob > F      =       .
    Residual |   219.91784 12438  .017681126           R-squared     =  0.0000
-------------+------------------------------           Adj R-squared =  0.0000
       Total |   219.91784 12438  .017681126           Root MSE      =  .13297

------------------------------------------------------------------------------
        nfld |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   .0180038   .0011922    15.10   0.000     .0156669    .0203408
------------------------------------------------------------------------------


. bswreg nfld [aw=wt60lf] , cmd(reg) bsw(bsw1-bsw500)
Results from BSWREG

* The confidence intervals below are based on the normal distribution

  Var_name         Coef        BS_se     BS_zstat    BS_pvalue   BS_cilow95    BS_ciup95
     _cons     0.018004     0.000460    39.113152     0.000000     0.017102     0.018906

Total bootstraps completed: 500

**********************************************************************************************


Bootvare_v20
Variance estimation using 500 bootstraps for Totals and Ratios

Obs    type     var1    var2           yhat          bs_sd    bs_cv          ci195          ciu95
 1     Ratio    nfld    count          1.80           0.05     2.56           1.71           1.89

where count is equal to the population size.
```

Means:

```
. bysort gender: means age [aw=wt60lf]
-------------------------------------------------------------------------------
-> gender = 1
    Variable |    Type         Obs        Mean       [95% Conf. Interval]
-------------+-----------------------------------------------------------------
         age | Arithmetic    5597    38.74955        38.23552   39.26358
             |  Geometric    5597    32.88785        32.35452   33.42998
             |   Harmonic    5597    26.32428        25.78611   26.88539
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-> gender = 2
    Variable |    Type         Obs        Mean       [95% Conf. Interval]
-------------+-----------------------------------------------------------------
         age | Arithmetic    6842     40.6122        40.12048   41.10393
             |  Geometric    6842    34.30807        33.79512   34.82881
             |   Harmonic    6842    27.15345        26.63015   27.69773
-------------------------------------------------------------------------------



. bysort gender: reg age [aw=wt60lf]
-------------------------------------------------------------------------------
-> gender = 1
(sum of wgt is   1.3123e+07)

      Source |       SS       df       MS              Number of obs =    5597
-------------+------------------------------           F(  0,  5596) =    0.00
       Model |        0.00     0           .           Prob > F      =       .
    Residual |  2153437.42  5596  384.817267           R-squared     =  0.0000
-------------+------------------------------           Adj R-squared =  0.0000
       Total |  2153437.42  5596  384.817267           Root MSE      =  19.617

------------------------------------------------------------------------------
         age |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   38.74955   .2622102   147.78   0.000     38.23552    39.26358
------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-> gender = 2
(sum of wgt is   1.3474e+07)

      Source |       SS       df       MS              Number of obs =    6842
-------------+------------------------------           F(  0,  6841) =    0.00
       Model |        0.00     0           .           Prob > F      =       .
    Residual |  2945099.01  6841  430.507092           R-squared     =  0.0000
-------------+------------------------------           Adj R-squared =  0.0000
       Total |  2945099.01  6841  430.507092           Root MSE      =  20.749

------------------------------------------------------------------------------
         age |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |    40.6122   .2508411   161.90   0.000     40.12048    41.10393
------------------------------------------------------------------------------
```

```
. bysort gender: bswreg age [aw=wt60lf] , cmd(reg) bsw(bsw1-bsw500)
_____________________________________________________________________________
-> gender = 1
Results from BSWREG

* The confidence intervals below are based on the normal distribution

  Var_name         Coef        BS_se     BS_zstat    BS_pvalue   BS_cilow95    BS_ciup95
     _cons    38.749551     0.137592   281.626648     0.000000    38.479877    39.019226

Total bootstraps completed: 500
_____________________________________________________________________________
-> gender = 2
Results from BSWREG

* The confidence intervals below are based on the normal distribution

  Var_name         Coef        BS_se     BS_zstat    BS_pvalue   BS_cilow95    BS_ciup95
     _cons    40.612205     0.127663   318.121521     0.000000    40.361992    40.862419

Total bootstraps completed: 500

*******************************************************************************************
```

SAS — Proc Reg Output

```
gender=1 ------------------------------------------------------
                                        Dependent Variable: age
                                            Weight: WT60LF
                                          Parameter Estimates

                                       Parameter       Standard
                         Variable  DF      Estimate          Error    t Value    Pr > |t|
                         Intercept  1      38.74955        0.26221     147.78      <.0001
gender=2 ------------------------------------------------------
                                        Dependent Variable: age
                                            Weight: WT60LF
                                          Parameter Estimates

                                       Parameter       Standard
                         Variable  DF      Estimate          Error    t Value    Pr > |t|
                         Intercept  1      40.61220        0.25084     161.90      <.0001
```

Bootvare_v20

```
Variance estimation using 500 bootstraps for a Regression
Dependent variable: age
Obs    gender      beta         bhat      bs_var      bs_sd     bs_cv     cil95      ciu95
1        1      Intercept    38.7496    0.018932    0.13759     0.36    38.4799    39.0192
2        2      Intercept    40.6122    0.016298    0.12766     0.31    40.3620    40.8624
```

Ratios:

```
. gen ah=age/height

. svyratio age/height [pw=wt60lf]

Survey ratio estimation

pweight:  wt60lf                                  Number of obs    =     12439
Strata:   <one>                                   Number of strata =         1
PSU:      <observations>                          Number of PSUs   =     12439
                                                  Population size  =  26596782

------------------------------------------------------------------------------
      Ratio       |   Estimate    Std. Err.   [95% Conf. Interval]        Deff
------------------+-----------------------------------------------------------
     age/height   |   .7173197    .0041964    .7090941    .7255453    1.752756
------------------------------------------------------------------------------



. bswreg age height [pw=wt60lf], cmd(svyratio) bsw(bsw1-bsw500)

50 bootstraps completed
100 bootstraps completed
150 bootstraps completed
200 bootstraps completed
250 bootstraps completed
300 bootstraps completed
350 bootstraps completed
400 bootstraps completed
450 bootstraps completed
500 bootstraps completed
Results from BSWREG
-------------------

* The confidence intervals below are based on the normal distribution


  Var_name         Coef        BS_se     BS_zstat    BS_pvalue   BS_cilow95    BS_ciup95
age:height     0.717320     0.001743   411.545410     0.000000     0.713903     0.720736

Total bootstraps completed: 500

*******************************************************************************************
```

Bootvare_v20

```
Variance estimation using 500 bootstraps for Totals and Ratios

Obs    type     var1     var2           yhat          bs_sd    bs_cv          cil95          ciu95
 1     Ratio    age      height         71.73          0.17     0.24          71.39          72.07
```

Variance estimation with plausible value achievement data: Two STATA programs for use with the YITS/PISA data.

By Darren Lauzon

Abstract

Use of the plausible value achievement data available in YITS/PISA introduces variability due to measurement error over and above that due to sampling variance induced by the survey design. Two programs are offered: The first estimates standard errors and confidence limits for estimates produced by various STATA model estimation commands when plausible value dependent variables are considered. The second produces kernel density estimates and corresponding confidence limits for plausible value data. Program code is included in an appendix.

Introduction

Complex survey design is a common feature of most data sets available at the Research Data Centres (RDCs). The Youth in Transition Survey/Programme for International Student Assessment (YITS/PISA) adds the additional complexity of plausible value (PV) achievement data. Plausible values are a recent innovation in item response theory and are increasingly used with tests of student achievement to estimate population parameters, such as mean performance or population regression coefficients, and have desirable properties over other available estimation approaches, particularly in tests with relatively few items per individual. The reader is referred to Mislevey, 1991 and the documentation to the 1995 TIMSS (Gonsalez, Smith and Sibberns, 1998) for a detailed discussion. This article outlines the use of plausible values and discusses two STATA programs that apply appropriate methods to compute standard errors for estimates produced by several STATA model estimation commands and a modified version of the kernel density command using plausible value achievement data as the dependent variable.

II. Using plausible values

Let θ be any population parameter (e.g., a mean, a vector of regression coefficients) and let the statistic $\hat{\theta}_j$ be an estimate of that parameter using the *j*th plausible value. For example, the Ordinary Least Squares (OLS) estimator of the population regression coefficients for a (column) vector of conditioning variables x_i for the *i*th student would be $\left(\sum_i x_i x_i'\right)^{-1} \sum_i x_i y_{ij}$ where y_{ij} is the value of the *j*th plausible value for individual *i*. Though it is possible to report this statistic, the estimate may vary notably if plausible value variables other than the *j*th one are chosen. It is therefore recommended to report the average estimate, $\hat{\theta} = (1/J)\sum_j \hat{\theta}_j$, J being the number of plausible value draws (5 in the YITS/PISA data). It should be noted of course that in the case of a linear estimator like the sample mean or the OLS estimator, this is equivalent to estimating the statistic using the average of the J plausible values as the dependent variable. It is still necessary,

however, to calculate the $\hat{\theta}_j$ values in order to compute the measurement error contribution to the total variance so there is no real advantage to averaging the plausible values in the case of linear estimators.

The total variance of $\hat{\theta}$ is estimated by the mean of the sampling variance estimates for the set of J estimates $\hat{\theta}_j$, and a second component based on the variability among the estimates $\hat{\theta}_j$:

$$\text{tot var}(\hat{\theta}) = (1/J)\sum_j \text{var}(\hat{\theta}_j) + (1+1/J)(1/(J-1))\sum_j (\hat{\theta}_j - \hat{\theta})^2 \qquad (1)$$

III. The sampling variance for YITS/PISA data

The sampling variance estimator that is used in the total variance calculation should account for the survey design. In Canada as well as other OECD participant countries, a Balanced Repeated Replication (BRR) design was used. The PISA data set contains 80 BRR replicate weights for all countries including Canada. In Canada, however, many more schools were sampled than in other OECD countries. Though variance strata and psu groupings were established in order to calculate the replicate weights to facilitate comparison of Canadian results to other participant countries, the 80 replicates may be too few given the much larger number of schools in the Canadian sample. Thus a set of 1000 bootstrap weights is also available and it is recommended that these be used in analysis of the PISA data that is not intended for comparison with any participant countries. The PISA was integrated with the first panel of the YITS (15 year-old cohort) and PISA participants also received the YITS questionnaires (for students and their parents). It is possible to link responses to the YITS questionnaires to the PISA data set, in which case, the bootstrap weights should be used for variance calculation. The programs described in this article were originally designed for the BRR case, but now allow the user to use the bootstrap method.

IV. pvpisa

This STATA command will compute the estimated coefficient vector and variance-covariance matrix based on the above considerations for all STATA's e-class (estimation) commands. The command syntax is as follows:

```
pvpisa varlist [weight] [if exp] [in range], cmd(string) pv(varlist) rw(varlist)
fays(real) [cmdops(string) brr uset dof(real) level(integer)]
```

`varlist`: This is a STATA-style varlist of *independent* variables that you want to condition on. Do not include, as is done in other estimation commands, any dependent variable first.

`weights`: pweights would be the relevant choice. All weight types are allowed to avoid errors generated by specific STATA commands that do not support pweights. Enter here the *full-sample*

weight, `w_fstuwt` in the YITS/PISA reading file.[1] If you do not specify a weight, the program will still run, but the results will be meaningless.

`[if]` and `[in]`: Use these as with any other STATA command to subset the data.

The options are as follows:
`cmd()`: Here you type the STATA command you want to execute: e.g., `tobit`

`cmdops()`: Here you type all the options you would normally specify with the command, e.g. `noconstant` for `regress`. Bear in mind that some of these options will not make sense when used with a replicate variance calculation scheme. It is up to the user to choose the options appropriately. Only options that affect the estimates are relevant. Variance related options like "`robust`" and "`cluster()`" will have no effect because the covariance matrices computed by the estimation command are not used by `pvpisa`.

`brr`: If `brr` is specified, the BRR method is used and the BRR weights should be given in `rw()`. If `brr` is not specified, the bootstrap is assumed and the weights in `rw()` should be the bootstrap weights.

`pv(varlist)`: This is the list of variables that contain the plausible values, the dependent variables for the analysis. The PISA PV variables for overall reading achievement are `pv1read pv2read pv3read pv4read pv5read`.

`rw(varlist)`: This is the list of replicate weight variables. The BRR weights for the PISA reading data would be entered as: `w_fstr1-w_fstr80`. The bootstrap weights, if the `brr` option is not specified, would be `BPSR1-BPSR1000`.

`fays(real)`: Enter here the constant for use with Fay's variant of the BRR method. The default is 0.5 for the YITS/PISA data. This option is relevant only if `brr` is specified.
`uset`: This option requests that confidence limits and p-values be calculated using student's t-distribution with degrees of freedom given by `dof()`. The default is to use the standard normal distribution.

`dof(real)`: Use this if you specify `uset`. This is the degrees of freedom to use in the calculation of confidence limits and p-values. See the last section for details concerning the choice of degrees of freedom. The default is 80. This option has no effect if `uset` is not specified.

`Level(integer)`: This is just the standard STATA option that requests the level for constructing confidence intervals. The default is the 95% interval.[2]

[1] At this time many of STATA's panel data estimation routines (so-called "`xt`" commands) do not accept weights, or only accept "group" weights, the group typically being the unit of analysis such as a person. Thus `pvpisa` cannot be used with YITS/PISA for many of the traditional panel data commands. I am looking into developing separate commands that will take the student weights for these estimators. Note, however, that the fixed and random effects estimators can be estimated (after transforming the data) by simple OLS, thus making it possible to use `pvpisa`. See Green (2001) for the necessary transformations.

Note that the output contains only the estimated coefficients, standard errors, t- or z-statistics, p-values and confidence limits. Extra output, such as the MSE or R-squared statistic, is not provided. No likelihood estimates are given in the case of MLE commands. These statistics should be calculated accounting for the survey design. STATA's `test` and `predict` commands can be used after `pvpisa` as with any estimation command. You can also get the coefficient vector using `e(b)` and the coefficient covariance matrix using `e(V)` after estimation if you wish to use `pvpisa` in a program.

At the time of writing, `pvpisa` was tested with the following single equation commands:[3]

```
Regress     Ologit
tobit       Mlogit
qreg
probit
logit
oprobit
```

Example 1:

This example uses the YITS/PISA reading data to regress reading achievement on total hours of instructional time per year (`tothrs`) and student-reported class size (`st28q01`) for students in Alberta.

```
pvpisa tothrs st28q01 [pweight = w_fstuwt] if prov == 8, cmd("regress")
pv(pv1read pv2read pv3read pv4read pv5read) rw(w_fstr1-w_fstr80) brr uset
fays(0.5) level(90)
```

Here is the output.

```
Command Summary: regress pv1read pv2read pv3read pv4read pv5read tothrs st28q01 if prov == 8 ,
confidence level is 90
BRR specified: BRR weights assumed and BRR method used
estimates for pv1read complete
estimates for pv2read complete
estimates for pv3read complete
estimates for pv4read complete
estimates for pv5read complete
There are 5 plausible values
There are 80 replciates
The confidence value is 90
t distribution requested degrees of freedom 80
------------------------------------------------------------------------------
p.v.         |      Coef.   Std. Err.      t    P>|t|     [90% Conf. Interval]
-------------+----------------------------------------------------------------
tothrs       |   .0028195   .0213395     0.13   0.895    -.0326921    .0383311
st28q01      |   2.520135   .3262991     7.72   0.000     1.977133    3.063137
_cons        |   485.8485   22.92675    21.19   0.000     447.6955    524.0015
------------------------------------------------------------------------------
```

Example 2:

[2] Note, `level` can be typically specified as an option on most e-class commands. If you specify `level` in the `cmdops()` option, it will have no effect on the confidence limits estimated by `pvpisa`. You must use the `level` option here for the desired effect.

[3] Multiple equation commands are being considered for two-stage least squares, seemingly unrelated regression, the Heckman selection model and the bivariate probit.

This example estimates the coefficients of an ordered probit model to estimate the probability of being in one of five possible proficiency levels in the reading test.

```
local j 1
while `j' <= 5 {
      g y`j' = .
      qui {
      replace y`j' = 0 if pv`j'read <= 334.75
      replace y`j' = 1 if pv`j'read > 334.75 & pv`j'read <= 407.67
      replace y`j' = 2 if pv`j'read > 407.67 & pv`j'read <= 480.18
      replace y`j' = 3 if pv`j'read > 480.18 & pv`j'read <= 552.89
      replace y`j' = 4 if pv`j'read > 552.89 & pv`j'read <= 625.61
      replace y`j' = 5 if pv`j'read > 625.61
      }
      local j = `j' + 1
      }

pvpisa tothrs st28q01 [pweight = w_fstuwt], cmd("oprobit") pv(y1-y5) rw(w_fstr1-
w_fstr80) brr uset fays(0.5)
```

Here is the output.

```
Command Summary: oprobit y1 y2 y3 y4 y5 tothrs st28q01  ,
confidence level is 95
BRR specified: BRR weights assumed and BRR method used
estimates for y1 complete
estimates for y2 complete
estimates for y3 complete
estimates for y4 complete
estimates for y5 complete
There are 5 plausible values
There are 80 replciates
The confidence value is 95
t distribution requested degrees of freedom 80
------------------------------------------------------------------------------
p.v.         |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
tothrs       |  -.0001164   .0000907    -1.28   0.203    -.0002969    .0000642
st28q01      |   .0285507   .0022075    12.93   0.000     .0241576    .0329439
_cut1        |  -1.478013   .1153145   -12.82   0.000    -1.707496   -1.24853
_cut2        |  -.7588495   .1155712    -6.57   0.000    -.9888435   -.5288555
_cut3        |  -.0304934   .1113955    -0.27   0.785    -.2521775    .1911908
_cut4        |   .7283558   .1127004     6.46   0.000     .5040748    .9526368
_cut5        |   1.574229   .1106444    14.23   0.000     1.354039    1.794418
------------------------------------------------------------------------------
```

V. kpvpisa

This program uses a modified version of STATA's `kdensity` command to compute the kernel density estimator of the population achievement density. The modification allows the use of iweights, which are rescaled to sum to 1 prior to estimation. For survey data, the iweights feature is more appropriate.

The kernel density estimator has been recommended in the past for exploratory data analysis (see the book by Silverman, 1986). Now, the kernel density estimator is increasingly used in non-parametric regression analysis (Yatchew, 1998; Blundel and Duncan 1998; DiNardo and Tobias, 2001). The kernel estimator is also used in matching studies (Smith and Todd, 2001). A recent area

of application has been the analysis of wage and earnings inequality (DiNardo, Fortin and Lemieux, 1996; Daly and Valleta, 2000). The kernel density estimator, with survey weights w_i normalized to sum to 1 is defined as:

$$\hat{f}(y_0) = \sum_i \frac{w_i}{h} K\left(\frac{y_0 - y_i}{h}\right) \qquad (2)$$

Here, the function is estimated at the point y_0. The constant h is the bandwidth or smoothing constant and is chosen by the analyst. K is the kernel function. This function places decreasing weight on data that are further away from the point y_0. The kernel density estimator is a generalization of the familiar histogram, the histogram being a special case of the kernel density estimator with a suitably chosen kernel function. An excellent and accessible discussion of the kernel density estimator is in DiNardo and Tobais (2001).

Like any other estimator based on data from a sample survey, the point estimator $\hat{f}(y_0)$ has sampling variation. There is currently open discussion of variance estimation with the kernel density estimator. The bootstrap has been used in many applications. Buskirk and Lohr (2003) discuss the asymptotic properties of the sample weighted kernel density estimator and derive confidence limits based on asymptotic normality conditions. The program `kpvpisa` simply applies equation (2) using the `kdensity` command and computes confidence limits based on either the BRR weights or the bootstrap weights. The total variance, including the measurement error associated with the use of plausible values is incorporated as indicated in equation (1). The syntax is as follows:

```
kpvpisa varlist [weight] [if exp] [in range], at(varname) rw(varlist) fays(real)
[brr uset dof(real) bandwidth(real) kernel(string) level(integer)]
```

`varlist:` Enter the list of plausible value variable names.

`weight:` The program takes aweights, fweights or iweights. Aweights and fweights are allowed to correspond to the original `kdensity` command. The modified command that `kpvpisa` calls, however, allows iweights, which are scaled to sum to one. Users should therefore use the iweights. Enter the full-sample weight here. For the PISA reading data, this is `w_fstuwt`. Note that the results will be meaningless if the weight variable is omitted.

`[if]` and `[in]:` These are standard STATA options to subset data and use the standard syntax.

`at(varlist):` The `at()` option of kdensity *must* be used because you need to average the density estimates across the J plausible values and be able to plot the results. Enter the name of a variable that contains the values at which to estimate the density. An example is given below. The P values of the `at()` variable should be in the first P observations of the data set, P being the number of points at which the density is estimated. Use `sort` if this is not the case.

`rw(varlist)`**:** Here you type the names of the replicate weight variables. TheBRR weights for overall reading achievement are `w_fstrwt1-wfstrwt80`. The bootstrap weights for overall reading achievement are `BPSR1-BPSR1000`.

`brr`**:** Use this option for the BRR method. The replicate weights in `rw()` should be the BRR weights. The default is the bootstrap method in which case the weights in `rw()` should be the bootstrap weights.

`uset`**:** Use this option to request confidence limits be based on the student's t- distribution. The default is to use the standard normal. There is no option as yet to use the bootstrap distribution in the case that `brr` is not specified.

`dof(real)`**:** Use this option if `uset` is specified. This allows the user to set the degrees of freedom. The default is to use the formula listed in Appendix A, which takes into account the use of plausible values. This option has no effect if `uset` is not specified.

`fays(real)`**:** Enter Fay's constant. For PISA this is 0.5.

`bandwidth(real)` : This is the bandwidth *h* that is used to smooth the density estimate. If this is omitted, the default is the "rule of thumb" estimator described by Silverman (1986).

`kernel(string)`**:** Enter here the option for the kernel function (see `kdensity`) in the STATA manual for the options. The default is the Gaussian (normal) kernel.

`level(integer)`**:** This is the standard STATA option for specifying the level of significance for the confidence intervals. This level will be used to estimate the confidence bounds for the density estimate.

`kpvpisa` places the following variables in your dataset. These are placed in the first *P* observations, *P* being the number of values for the `at()` variable.

`dpv`**:** the average of the J plausible value density estimates.

`vdpv`: the estimated total variance of dpv.

`dpvlo`**:** the lower confidence limit for the density estimate dpv.

`dpvhi`**:** the upper confidence limit for the density estimate dpv.

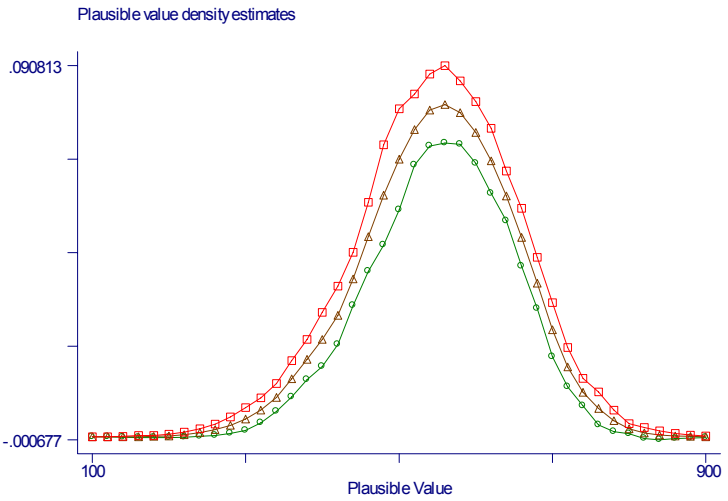
`kpvpisa` will graph `dpvlo`, `dpv` and `dpvhi` against the variable chosen for the `at()` option to give the shape of the density function and the margin of error.

Example 3:

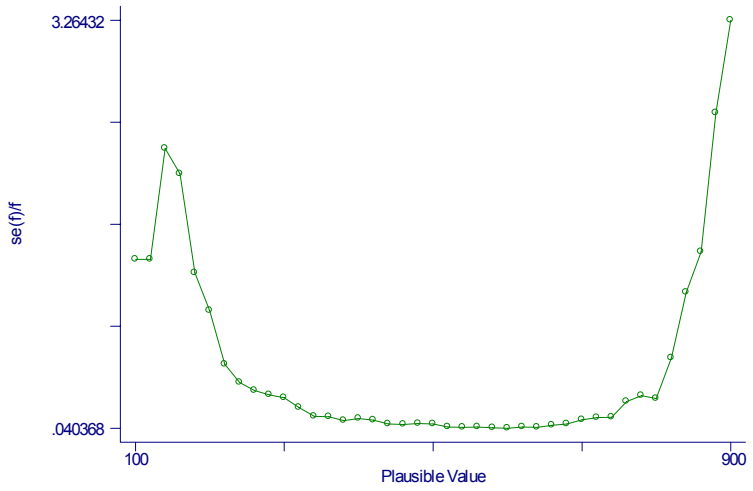
This command estimates the reading achievement density for Alberta

```
kpvpisa pv1read pv2read pv3read pv4read pv5read [aweight = w_fstuwt] if prov==8,
at(xvalue) rw(w_fstr1-w_fstr80) brr uset fays(0.5) level(95)
```

The following are the estimated density and confidence limits:



From the graph above it would appear that the density is least accurately estimated near the mode of the distribution, where the confidence bounds are widest. This pattern occurs because it is around the mode that the density estimates are the largest and therefore the width of the confidence interval, which is a function of the estimated standard error, is greatest. However, a different perspective emerges if we examine the standard error of the density relative to the density itself. The next figure shows a plot of $se(\hat{f})/\hat{f}$, the estimated coefficient of variation (CV). Here we see the estimated standard error of the density relative to the magnitude of the density estimate is largest at the tails of the distribution, as one would expect.



Example 4:

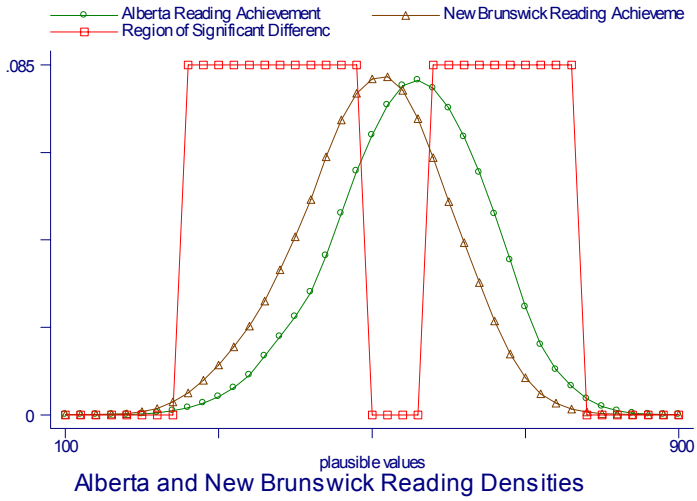
This do file illustrates a typical use of `kpvpisa`. Here we estimate the reading achievement density for two provinces: Alberta and New Brunswick. We of course would like to know to what degree the difference is *not* due to chance. We answer this by comparing the confidence limits for each density estimate. If the limits do not overlap, then the difference is statistically significant.[4] The do file produces a plot of the Alberta and New Brunswick densities and an indicator function that is non-zero when the two achievement probabilities are significantly different from each other.

```
clear
set memory 50m
set matsize 800
set logtype text

use "pisadata.dta"

      * create 50 xvalue variable at which densities are estimated
      local i 100
      local count 1
      g xvalue = . in 1/50
      while `i' <= 900 {
            qui replace xvalue = `i' in `count'
            local I = `I' + 20
            local count = `count' + 1
      }

kpvpisa pv1read pv2read pv3read pv4read pv5read [aweight = w_fstuwt] if prov==8,
at(xvalue) rw(w_fstr1-w_fstr80) fays(0.5) level(95)

g ab_d = dpv
```

[4] Note that no Bonferroni adjustment is made in this case, because I am simply plotting locations where the densities are significantly different *at that point*. I do not make global statements like: "The densities differ at one or more locations, therefore the densities are different." That is, these individual tests are not used to say that one density is significantly different than the other.

```
g ab_lo = dpvlo
g ab_hi = dpvhi

drop dpv dpvlo dpvhi vdpv

kpvpisa pv1read pv2read pv3read pv4read pv5read [aweight = w_fstuwt] if prov==3,
at(xvalue) rw(w_fstr1-w_fstr80) fays(0.5) level(95)

g nb_d = dpv
g nb_lo = dpvlo
g nb_hi = dpvhi

g byte I = ab_lo > nb_hi | nb_lo > ab_hi

g scale = 0.085*I

label variable xvalue "plausible values"
label variable ab_d "Alberta Reading Achievement"
label variable nb_d "New Brunswick Reading Achievement"
label variable scale "Region of Significant Difference"
graph ab_d nb_d scale xvalue, c(lll) ti("Alberta and New Brunswick Reading
Densities")
```

The plot is:



VI. A Final Word

The programs `pvpisa` and `kpvpisa` are provided to the RDC community as "beta" versions in the hope that they will be helpful. The programs were written for STATA version 7 and have not been tested on version 8. A note on the use of the t- distribution for inference is warranted. For analysis based on the full PISA/YITS sample there is not likely to be much difference between the choice of the t or standard normal distribution for inference. If the user is interested in using the t distribution, the choice of degrees of freedom needs to be made carefully. Ideally, degrees of

freedom should take into account the use of plausible values. I implement this with `kpvpisa` but not with `pvpisa` (see Appendix A). Also, the degrees of freedom will generally be lower if the user subsets data in such a way that there are strata with no psus in the sub-sample on which inference is based. This would occur if analysis were restricted to one province for example. In such cases, it is recommended that users use the `uset` and `dof()` options to explore the sensitivity of their results to changes in the degrees of freedom. For more accurate calculation of degrees of freedom, the user is advised to contact the manager of the survey data they are using. With respect to the bootstrap option, I do not yet give the option of using the bootstrap distribution for computing confidence intervals in this version as is done in other applications such as the STATA programs described elsewhere in this issue.

References

Adams, R and M. Wu (2002) *PISA 2000 Technical Report*, Paris, OECD

Blundel R. and A Duncan (1998) "Kernel Regression in Empirical Microeconomics", *Journal of Human Resources* 33(1):62-87

Buskirk, T. D. and S. Lohr (2003) "Asymptotic Properties of Kernel Density Estimation with Complex Survey Data" Arizona State University technical paper.

Daly, M. C. and R. G. Valletta (2000) "Inequality and Poverty in the United States: The Effects of Changing Family Behaviour and Rising Wage Dispersion" Working Paper, Federal Reserve Bank of San Francisco.

DiNardo, J. and J. L. Tobias (2001) "Nonparametric Density and Regression Estimation", *Journal of Economic Perspectives* 13(4):11-28

DiNardo, J., N. M. Fortin and T. Lemieux (1996) "Labor Market Institutions and the Distribution of Wages 1973-1992: A Semiparametric Approach, *Econometrica,* 64(5):1001-1044

Gonsalez, Smith and Sibberns (1998) *User Guide for the TIMSS International Database: Final Year of Secondary School, 1995 Assessment*, International Association for the Evaluation of Education Achievement and TIMSS International Study Centre, Boston College

Green, William H. (2001) *Econometric Analysis 4th Edition*, Prentice-Hall, Upper Saddle River, New Jersey.

Mislevey, R. J. (1991) "Randomization based inference about examinees in the estimation of item parameters" *Psychometrika* 56:177-196

Silverman, B. (1986) *Density Estimation for Statistics and Data Analysis*, London: Chapman and Hall.

Smith, J. and Todd, P. (2001) "Does Matching Overcome LaLonde's Critique of Nonexperimental Estimators?" Unpublished working paper, University of Maryland.

Yatchew, A. (1998) "Nonparametric Regression Techniques in Economics" *Journal of Economic Literature* 36:669-721

Appendix A

Calculation of degrees of freedom accounting for plausible values:

This formula is implemented in `kpvpisa` but not in `pvpisa`. The reason is that Stata's `estimates post` command was used and this only allows scalar degrees of freedom, not one that varies with the parameters estimated. Here, though, is the formula that is used in `kpvpisa`. Given

$$\text{tot var}(\hat{\theta}) = (1/J)\sum_j \text{var}(\hat{\theta}_j) + (1+1/J)(1/(J-1))\sum_j (\hat{\theta}_j - \hat{\theta})^2 ,$$

define $f_J = (1+1/J) * B_J / tot\,\text{var}$,where $B_J = (1/(J-1))\sum_j (\hat{\theta}_j - \hat{\theta})^2$. f_J is just the proportion of the total variance attributable to measurement error owing to the use of plausible values. Let *d* be the degrees of freedom that would prevail if θ_n the latent ability parameter for student *n* (estimated by using the plausible values) had been observed. For PISA data this value varies by country and is a maximum of 80. The degrees of freedom calculation is

$$dof = \frac{1}{\frac{f_J^2}{J-1} + \frac{(1-f_J)^2}{d}} .^{5}$$

PLEASE NOTE:

This data product is provided "as-is", and Statistics Canada makes no warranty, either express or implied, including but not limited to, warranties of merchantability and fitness for a particular purpose. In no event will Statistics Canada be liable for any direct, special, indirect, consequential or other damages however caused.

[5] This formula and a discussion is provided in the 2000 PISA Technical Report, Adams and Wu (2002).

Appendix B

pvpisa

```
program define pvpisa, eclass byable(recall)
        version 7.0

      syntax varlist(numeric) [pweight fweight aweight iweight] [if] [in], cmd(string) [cmdops(string)] pv(varlist numeric) rw(varlist numeric)
fays(real) [brr uset dof(integer 80) level(integer 95)]

        di in gr "Command Summary: `cmd' `pv' `varlist' `if' `in', `cmdops'"
        di in gr "confidence level is `level'"

        if "`brr'" != "" {

                di in ye "BRR specified: BRR weights assumed and BRR method used"

                /* compute sampling variance for `ith' plausible value */
                local i 1 /* i indexes plausible values */
                foreach pvname in `pv' {

                        /* compute full sample estimate of beta */
                qui `cmd' `pvname' `varlist' [`weight'`exp'] `if' `in', `cmdops'
                        matrix b`i'0 = e(b)'

                        matrix vbrr`i' = (b`i'0 - b`i'0)*(b`i'0 - b`i'0)' /*This is a quick cheat way to get a KxK
                                                                                0 matrix. That is, initialize the cov
                                                                                matrix to 0. */

                        local reps 1 /* reps indexes the replicates */
                        /*start BRR - accumulate sum of squares using the `i'th plausible values */
                        foreach wname in `rw' {
                                qui `cmd' `pvname' `varlist' `if' `in' [`weight' = `wname'], `cmdops'
                                matrix vbrr`i' = vbrr`i' + (e(b)' - b`i'0)*(e(b)' - b`i'0)'
                                local reps = `reps' + 1
                        }
                        matrix vbrr`i' = (1/(`reps'*(1 - `fays')^2))*vbrr`i'               /* compute the fays BRR variance-covariance matrix */
                        di in gr "estimates for `pvname' complete"
                local i = `i' + 1

                }

                /* ensure counters are set properly */
              local i = `i' - 1
                local reps = `reps' - 1

                /* Begin computing Variance and confidence limits of estimates taking into account plausible value imputation and survey desing */
                /* compute average pv estiamte */
                local z 1
                matrix b0 = J(rowsof(b10),1, 0)
                while `z' <= `i' {
                        matrix b0 = b0 + (1/`i')*b`z'0
                        local z = `z' + 1
                }

                /* compute elements for total VCE */
                /* U is average VCE of plausible value VCEs */
```

```
local z 1
matrix U = J(rowsof(vbrrl), colsof(vbrrl),0)
local z 1
while `z' <= `i' {
        matrix U = U + (1/`i')*(vbrr`z')
        local z = `z' + 1
}

/* B is VCE of plausible value estiamtes */
matrix B = J(rowsof(vbrrl), colsof(vbrrl), 0)
local z 1
while `z' <= `i' {
        local ii 1      /* These lines are meant to prevent a division by 0 error
                           that would occur if only 1 plausible value were used.
                           this enables the user to run BRR on non plausible value
                           d.v.s simply by listing the d.v. in the pv() option. */
        if `i' - 1 != 0 {
                local ii = (1/(`i'-1))
        }
        matrix B = B + `ii'*(b`z'0 - b0)*(b`z'0 - b0)'
local z = `z' + 1
}

/* estimate total variance covariance matrix */
matrix vbrrpv = U + (1 + (1/`i'))*B


matrix b0 = b0'

di in green "There are `i' plausible values"
di in green "There are `reps' replciates"
di in green "The confidence value is `level'"

if "`uset'" != "" {
        di in ye "t distribution requested degrees of freedom `dof'"
        estimates post b0 vbrrpv, dep("p.v.") dof(`dof')
}
else estimates post b0 vbrrpv, dep("p.v.")
estimates display, level(`level')

} /* end execution of BRR routine */

if "`brr'" == "" {

        di in ye "BRR not specified: Bootstrap weights assumed and Bootstrap method used"

        /* compute sampling variance for `ith' plausible value */
        local i 1 /* i indexes plausible values */

        foreach pvname in `pv' {

                /* compute full sample estimate of beta */
        qui `cmd' `pvname' `varlist' [`weight'`exp'] `if' `in', `cmdops'
                matrix b`i'0 = e(b)'


                matrix b`i'avg = b`i'0 - b`i'0    /* initialize average beta vector to zero */
                local reps 1
```

```
/*compute average bs beta */

    foreach wname in `rw' {
    qui `cmd' `pvname' `varlist' [`weight'= `wname'] `if' `in', `cmdops'
          matrix b`i'avg = b`i'avg + e(b)'
    local reps = `reps' + 1
    }
    matrix b`i'avg = (1/`reps')*b`i'avg

    matrix vbrr`i' = (b`i'0 - b`i'0)*(b`i'0 - b`i'0)' /*This is a quick cheat way to get a KxK
                                                        0 matrix. That is, initialize the cov
                                                          matrix to 0. */

    local reps 1 /* reps indexes the replicates */
    /*start BRR - accumulate sum of squares using the `i'th plausible values */
    foreach wname in `rw' {
          qui `cmd' `pvname' `varlist' `if' `in' [`weight' = `wname'], `cmdops'
          matrix vbrr`i' = vbrr`i' + (e(b)' - b`i'avg)*(e(b)' - b`i'avg)'
    local reps = `reps' + 1
    }
    matrix vbrr`i' = (1/(`reps'-1))*vbrr`i'          /* compute the bootstrap variance-covariance matrix */
     di in gr "estimates for `pvname' complete"
local i = `i' + 1

}

/* ensure counters are set properly */
local i = `i' - 1
  local reps = `reps' - 1

/* Begin computing Variance and confidence limits of estimates taking into account plausible value imputation and survey desing */
/* compute average pv estiamte */
local z 1
matrix b0 = J(rowsof(b10),1, 0)

while `z' <= `i' {
      matrix b0 = b0 + (1/`i')*b`z'0
local z = `z' + 1
}

/* compute elements for total VCE */
/* U is average VCE of plausible value VCEs */

local z 1
matrix U = J(rowsof(vbrr1), colsof(vbrr1),0)
local z 1

while `z' <= `i' {
      matrix U = U + (1/`i')*(vbrr`z')
local z = `z' + 1
}

/* B is VCE of plausible value estiamtes */
matrix B = J(rowsof(vbrr1), colsof(vbrr1), 0)
local z 1
while `z' <= `i' {
      local ii 1       /* These lines are meant to prevent a division by 0 error
                                that would occur if only 1 plausible value were used.
                                this enables the user to run BRR on non plausible value
```

```
    if `i' - 1 != 0 {
            local ii = (1/(`i'-1))
        }
        matrix B = B + `ii'*(b`z'0 - b0)*(b`z'0 - b0)'
    local z = `z' + 1
    }

    /* estimate total variance covariance matrix */
    matrix vbrrpv = U + (1 + (1/`i'))*B

    matrix b0 = b0'

    di in green "There are `i' plausible values"
    di in green "there are `reps' replciates"
    di in green "The confidence value is `level'"

    if "`uset'" != "" {
        di in ye "t distribution requested degrees of freedom `dof'"
        estimates post b0 vbrrpv, dep("p.v.") dof(`dof')
    }
    else estimates post b0 vbrrpv, dep("p.v.")
    estimates display, level(`level')

}

end
```

kpvpisa.ado

```
program define kpvpisa, byable(recall)
        version 7.0
        syntax varlist(numeric) [iweight fweight aweight] [if] [in],  at(varname numeric)   [ brr uset dof(integer 80) bandwidth(real 10)]
    [kernel(string)] rw(varlist numeric) fays(real) [level(integer 90)]

        di in green "Confidence level is: " "`level'%"

        label variable `at' "Plausible Value"

        * Obtain the users choice of kernel function and write it to macro K
        local K " "
        if "`kernel'" != " " {
            local K = "`kernel'"
        }

        * Obtain the users choice of bandwidth and enter it as the width option in macro h
        local h " "
        if `bandwidth' != 10  {
            local h = "width(`bandwidth')"
        }

        *count number of replicates
        local reps 0
        foreach rep in `rw' {
            local reps = `reps' + 1
        }

        *count number of plausible values
        local M 0
        foreach m in `varlist' {
            local M = `M' + 1
        }

        *do full sample densities
        qui g dpv = 0
        di in gr "computing full sample density estimates:"
        local j 1
        foreach pv in `varlist' {
            di in gr "command summary: kdens_ `pv' [`weight'`exp'] `if' `in', at(`at') g(xpoint d`j'0)  `K' `h' nograph "

            tempvar d`j'0
            kdens_ `pv' [`weight'`exp'] `if' `in', at(`at') g(xpoint `d`j'0') nograph
            qui replace `d`j'0' = `d`j'0'*r(scale)  /* The scale return from kdens_ ensures densities itegrate to 1 */
            drop xpoint
            qui replace `d`j'0' = 0.000000000000000`j' if `d`j'0' == 0 /* this line prevents any division by 0 problem later */
            qui replace dpv = dpv + (1/`M')*`d`j'0'
            local j = `j' + 1
        }

        /* compute Ustar, the average sampling variance where sampling variance is either
            the BRR variance or the Bootstrap variance */

        tempvar Ustar
        qui g `Ustar' = 0

        if "`brr'" != "" {
```

```
di in ye "BRR method requested: replicate weights assumed to be BRR"
*do BRR variance esitmates for pv densities
di in gr " "
di in gr  "doing BRR replicates now for:"
local j 1
foreach pv in `varlist' {
        tempvar U`j'
        local k 1
        qui g `U`j'' = 0
        di in gr "`pv'"
        foreach repwt in `rw' {
                tempvar d`j'`k'
                kdens_`pv' [`weight'=`repwt'] `if' `in', at(`at') g(xpoint `d`j'`k'')`K' `h' nograph
                qui replace `d`j'`k'' = `d`j'`k''*r(scale)
                drop xpoint
                qui replace `d`j'`k'' = 0.000000000000000`k' if `d`j'`k'' == 0
                qui replace `U`j'' = `U`j'' + (1/(`reps'*(1-`fays')^2))*((`d`j'`k'' - `d`j'0')^2)
                drop `d`j'`k'' /* drop now to save room */
                local k = `k' + 1
        }
        qui replace `Ustar' = `Ustar' + (1/`M')*`U`j''
        drop `U`j'' /* drop now to save room */
        local j = `j' + 1
}

}
di
if "`brr'" == "" {
        di in ye "BRR not specified: replicate weights assumed to be bootstrap weights"
        tempvar Ustar
        qui g `Ustar' = 0
        local j 1
        foreach pv in `varlist' {
                /* accumulate average density results */
                di in gr "accumulating average density results for pv `j'"
                tempvar davg`j'
                local k 1
                qui g `davg`j'' = 0
                foreach repwt in `rw' {
                        tempvar d`j'`k'
                        kdens_`pv' [`weight'=`repwt'] `if' `in', at(`at') g(xpoint `d`j'`k'')`K' `h' nograph
                        drop xpoint
                        qui replace `d`j'`k'' = `d`j'`k''*r(scale)
                        qui replace `davg`j'' = `davg`j'' + (1/`reps')*`d`j'`k''
                        drop `d`j'`k''
                        local k = `k' + 1
                }
                di in gr "average density for pv `j' done"
                /* now accumulate variance of jth plausible value density estimate */
                di in gr "doing bootstrap variance estitmate for pv `j'"
                tempvar U`j'
                qui g `U`j'' = 0
                local k 1
                foreach repwt in `rw' {
                        tempvar d`j'`k'
                        kdens_`pv' [`weight'=`repwt'] `if' `in', at(`at') g(xpoint `d`j'`k'')`K' `h' nograph
                        qui replace `d`j'`k'' = `d`j'`k''*r(scale)
                        drop xpoint
                        qui replace `d`j'`k'' = 0.000000000000000`k' if `d`j'`k'' == 0
                        qui replace `U`j'' = `U`j'' + (1/(`reps'-1))*((`d`j'`k'' - `davg`j'')^2)
```

```
            drop `d`j''`k'' /* drop now to save room */
            local k = `k' + 1
        }
        qui replace `Ustar' = `Ustar' + (1/`M')*`U`j''
        drop `U`j'' /* drop now to save room */
        local j = `j' + 1
    }
}
di

local j 1
tempvar Bm
qui g `Bm' = 0
while `j' <= `M' {
    qui replace `Bm' = `Bm' + (1/(`M'-1))*(`d`j'0' - dpv)^2
    local j = `j' + 1
}

g vdpv = `Ustar' + (1 + (1/`M'))*`Bm'

*calculate dof.
tempvar fm dof
g `fm'=(1 + (1/`M'))*`Bm'/vdpv
g `dof' = 1/((`fm'^2/(`j'-2)) + (((1-`fm')^2)/(`reps')))

local talpha = (1 - (`level'/100))/2 /* needed because invttail function is pr(T>t) */
local nalpha = 1 - `talpha' /* needed because the invnorm function is pr(Z<z) */
if "`uset'" != "" {
    di in ye "student's t distribution is used to compute confidence interval"
    g dpvlo = dpv - invttail(`dof', `talpha')*sqrt(vdpv)
    g dpvhi = dpv + invttail(`dof', `talpha')*sqrt(vdpv)
}

if "`uset'" == "" {
    di in ye "standard normal distribution is used to compute confidence interval"
    g dpvlo = dpv - invnorm(`nalpha')*sqrt(vdpv)
    g dpvhi = dpv + invnorm(`nalpha')*sqrt(vdpv)
}

graph dpvlo dpv dpvhi `at', c(lll) t1("Plausible value density estimates")

end
```

kdens_.ado

```
/* This program modifies the kdensity command supplied with Stata version 7.0
   to allow iweights. The iweights are scaled to sum to one.
   This program is called by kpvpisa
*/
*! version 2.3.6  26jun2000
program define kdens_, rclass
      version 6.0

      syntax varname [if] [in] [fw aw iw] [, /*
            */ Generate(string) N(integer 50) /*
            */ Width(real 0.0) noGRaph noDENsity /*
            */ BIweight COSine EPan GAUss RECtangle PARzen /*
            */ TRIangle Symbol(string) Connect(string) /*
            */ Title(string) AT(varname) NORmal STUd(int 0) * ]

      if "`at'"!="" & `n'!=50 {
            di in red "may not specify both the at() and n() options"
            exit 198
      }

      local ix `"`varlist'"'
      local ixl: variable label `ix'
      if `"`ixl'"'==`""" {
            local ixl "`ix'"
      }

      local gen `"`generat'"'

      local kflag = ( (`"`epan'"' != `""') + (`"`biweigh'"' != `""') + /*
                  */ (`"`triangl'"' != `""') + (`"`gauss'"' != `""') + /*
                  */ (`"`rectang'"' != `""')  + (`"`parzen'"' != `"""))
      if `kflag' > 1 {
            di in red `"only one kernel may be specified"'
            exit 198
      }

      if `"`biweigh'"'      != `""' { local kernel=`"Biweight"'     }
      else if `"`cosine'"'  != `""' { local kernel=`"Cosine"'       }
      else if `"`triangl'"' != `""' { local kernel=`"Triangle"'     }
      else if `"`gauss'"'   != `""' { local kernel=`"Gaussian"'     }
      else if `"`rectang'"' != `""' { local kernel=`"Rectangular"'  }
      else if `"`parzen'"'  != `""' { local kernel=`"Parzen"'       }
      else                          { local kernel=`"Epanechnikov"' }

      marksample use
      qui count if `use'
      if r(N)==0 { error 2000 }

      tokenize `gen'
      local wc : word count `gen'
      if `wc' {
            if `wc' == 1 {
                  if `"`at'"' == `""' {
                        error 198
                  }
                  confirm new var `1'
                  local yl `"`1'"'
```

```
        local xl `"`at'"'
        local nsave 1
    }
    else {
        if `wc' != 2 { error 198 }
        confirm new var `1'
        confirm new var `2'
        local xl  `"`1'"'
        local yl  `"`2'"'
        local nsave 2
    }
}
else {
    if `"`graph'"'!=`""' {
        di in bl /*
*/ `"no graph requested and did not request results be saved; no action taken"'
        exit
    }
    local xl   `"x"'
    local yl   `"density"'
    local nsave 0
}

tempvar d m z y
qui gen double `d'=.
qui gen double `y'=.
qui gen double `z'=.
qui gen double `m'=.

if `"`at'"' != `""' {
    qui count if `at' != .
    local n = r(N)
    qui replace `m' = `at'
    local srtlst : sortedby
    tempvar obssrt
    gen `obssrt' = _n
    sort `m' `obssrt'
}
else {
    if `"`n'"'!= `""' {
        if `n' <= 1 { local n = 50 }
        if `n' > _N {
            local n = _N
            noi di in gr `"(n() set to "' `n' `")"'
        }
    }
}

if "`weight'" != "" {
    tempvar tt
    qui gen double `tt' `exp' if `use'
    qui summ `tt', meanonly
    if "`weight'" == "aweight" {
        qui replace `tt' = `tt'/r(mean)
    }
    if "`weight'" == "iweight" {
        qui replace `tt' = `tt'/r(sum)
    }
```

```
}
else {
        local tt = 1
}

quietly summ `ix' [aweight`exp'] if `use', detail
local nmean = r(mean)
local nsig = r(Var)

tempname wwidth
scalar `wwidth' = `width'
if `wwidth' <= 0.0 {
        scalar `wwidth' = min( sqrt(r(Var)), (r(p75)-r(p25))/1.349)
        scalar `wwidth' = 0.9*`wwidth'/(r(N)^.20)
}

tempname delta wid
scalar `delta' = (r(max)-r(min)+2*`wwidth')/(`n'-1)
scalar `wid'   = r(N) * `wwidth'

if `"`at'"' == `""' {
        qui replace `m' = r(min)-`wwidth'+(_n-1)*`delta' in 1/`n'
}

tempname tmp1 tmp2 tmp3

local i 1
if `"`biweigh'"' != `""' {
        local con1 = .9375
        while `i'<=`n' {
                qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') /*
                        */ if `use'
                qui replace `y'=`tt'*`con1'*(1-(`z')^2)^2 /*
                        */ if abs(round(`z',1e-8))<1
                qui summ `y', meanonly
                qui replace `d'=(r(sum))/`wid' in `i'
                qui replace `y'=.
                local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
}
else if `"`cosine'"' != `""' {
        while `i'<=`n' {
                qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') /*
                        */ if `use'
                qui replace `y'= `tt'*(1+cos(2*_pi*`z')) /*
                        */ if abs(round(`z',1e-8))<0.5
                qui summ `y', meanonly
                qui replace `d'=(r(sum))/`wid' in `i'
                qui replace `y'=.
                local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
}
else if `"`triangl'"' != `""' {
        while `i'<=`n' {
                qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') if `use'
                qui replace `y'= `tt'*(1-abs(`z')) /*
                        */ if abs(round(`z',1e-8))<1
```

```
            qui summ `y', meanonly
            qui replace `d'=(r(sum))/`wid' in `i'
            qui replace `y'=.
            local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
    }
    else if `"`parzen'"' != `""' {
        local con1 = 4/3
        local con2 = 2*`con1'
        while `i'<=`n' {
            qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') if `use'
            qui replace `y'= `con1'-8*(`z')^2+8*abs(`z')^3 /*
                */ if abs(round(`z',1e-8))<=.5
            qui replace `y'= `tt'*`con2'*(1-abs(`z'))^3          /*
                */ if abs(round(`z',1e-8))>.5 & /*
                */ abs(round(`z',1e-8))<1

            qui summ `y', meanonly
            qui replace `d'=(r(sum))/`wid' in `i'
            qui replace `y'=.
            local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
    }
    else if `"`gauss'"' != `""' {
        local con1 = sqrt(2*_pi)
        while `i'<=`n' {
            qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') if `use'
            qui replace `y'= `tt'*exp(-0.5*((`z')^2))/`con1'
            qui summ `y', meanonly
            qui replace `d'=(r(sum))/`wwidth' in `i'
            local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
    }
    else if `"`rectang'"' != `""' {
        while `i'<=`n' {
            qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') if `use'
            qui replace `y'= `tt'*0.5 if abs(round(`z',1e-8))<1
            qui summ `y', meanonly
            qui replace `d'=(r(sum))/`wid' in `i'
            qui replace `y'=.
            local i = `i'+1
        }
        qui replace `d'=0 if `d'==. in 1/`n'
    }
    else {
        local con1 = 3/(4*sqrt(5))
        local con2 = sqrt(5)
        while `i'<=`n' {
            qui replace `z'=(`ix'-`m'[`i'])/(`wwidth') if `use'
            qui replace `y'= `tt'*`con1'*(1-((`z')^2/5)) /*
                */  if abs(round(`z',1e-8))<=`con2'
            qui summ `y', meanonly
            qui replace `d'=(r(sum))/`wid' in `i'
            qui replace `y'=.
            local i = `i'+1
        }
```

```
        qui replace `d'=0 if `d'==. in 1/`n'
    }

    label var `d' `"`yl'"'
    label var `m' `"`ixl'"'

    qui summ `d' in 1/`n', meanonly
    local scale = 1//(`n'*r(mean))

    if `"`density'"' != `""' {
        qui replace `d' = `d'*`scale' in 1/`n'
    }

    if `"`graph'"'==`""' {
        if `"`symbol'"'  == `""' {local symbol `"o"'}
        if `"`connect'"' == `""' {local connect `"l"' }
        if `"`title'"'   == `""' {
            local title `"Kernel Density Estimate"'
        }
        if `"`normal'"' != `""' {
            tempvar znorm
            scalar `tmp1' = 1/sqrt(2*_pi*`nsig')
            scalar `tmp2' = -0.5/`nsig'
            qui gen `znorm' = `tmp1'*exp(`tmp2'*(`m'-`nmean')^2)
            local symbol `"`symbol'i"'
            local connect `"`connect'l"'
            if `"`density'"' != `""' {
                tempvar fz
                qui gen `fz' = sum(`znorm')
                qui replace `znorm' = `znorm'/`fz'[_N]
            }
        }
        if `stud' > 0 {
            tempvar tm
            scalar `tmp1' = exp(lngamma(`stud'+1)/2)) /*
                */ / exp(lngamma(`stud'/2)) /*
                */ * 1/sqrt(`stud'*_pi)
            scalar `tmp2' = (`stud'+1)/2
            scalar `tmp3' = sqrt(`nsig')
            qui gen `tm' = `tmp1' * 1/((1+((`m'-`nmean') /*
                */ / `tmp3' )^2/`stud')^`tmp2')
            local symbol `"`symbol'i"'
            local connect `"`connect'l"'
            tempvar ft
            qui gen `ft' = sum(`tm')
            if `"`density'"' != `""' {
                qui replace `tm' = `tm'/`ft'[_N]
            }
            else {
                qui replace `tm' = `tm'/`ft'[_N]/`scale'
            }
        }
        graph `d' `znorm' `tm' `m', s(`symbol') c(`connect') /*
        */ title(`"`title'"') `options'
    }
    /* double save in S_# and r() */
    ret clear
    ret local kernel `"`kernel'"'
    ret scalar width = `wwidth'
```

```
        /* (sic) */
ret scalar n = `n'
ret scalar scale = `scale'
global S_1 `""`kernel'""'
global S_3 = `wwidth'
global S_2 = `n'
global S_4 = `scale'

if `nsave' == 1 {
        label var `d' `"density: `ixl'""'
        rename `d' `yl'
    }
    else if `nsave' == 2 {
        label var `m' `""`ixl'""'
        label var `d' `"density: `ixl'"'
        rename `d' `yl'
        rename `m' `xl'
    }
    if "`at'" != "" {
        sort `srtlist' `obsst'
    }

end
```

Instructions for authors

The Information and technical bulletin will accept submissions for articles that address methodological or technical topics related to the datasets that are available at the Research Data Centres.

Language of material:

Manuscripts may be submitted in English or French. Accepted submissions will be translated into both official languages for publication.

Length of submissions:

The maximum length of submitted articles should not exceed 20 pages, double-spaced, excluding programs and appendices. In addition to in-depth explanations of technical issues, the bulletin also accepts short (3 page) submissions that provide quick solutions to analytical problems and commentary from fellow researchers about material previously released in the bulletin.

File formats and layout of text:

Manuscripts must be submitted in Microsoft Word (.doc) and may be sent by regular mail on a disk or CD or by email.

Manuscripts must have a cover page showing the names of the authors, their primary institution of affiliation, and the contact information (telephone number, mailing address and e-mail address) of the lead author.

Manuscripts must be prepared in 12pt Times New Roman, double-spaced, with 1-inch (2.5 cm) margins.

Titles should have sentence-case capitalization (e.g., Information and technical bulletin).

Boldface type should only be used for headings. Underlining and italics are not to be used.

Endnotes (please do not use footnotes) and references should be single-spaced and formatted according to the *Statistics Canada Style Guide.*

File formats and layout of tables and charts

Tables and charts must be submitted in Microsoft Excel worksheets (.xls) or in comma-separated value (.csv) format. Each file must be clearly named table1, chart6, etc.

Tables and charts may be sent by regular mail on a disk or CD or by e-mail.

Follow the instructions for formatting tables and graphs in the Statistics Canada Style Guide.

Do not insert tables or charts into the text, but indicate their location in the text by inserting the title, followed by the filename in parentheses, e.g.,

Chart 6. Chocolate consumption by children, Canada, 2000 (chart6)

Mathematical expressions

All mathematical expressions should be set out separate from paragraph text. Equations must be numbered, with the number appearing to the right of the equation flush with the margin.

Style guide

Please follow the Statistics Canada Style Guide in all respects. A copy of the Style Guide is available by contacting the Editorial Committee at the addresses, below:

Addresses for submission

Manuscripts and all correspondence relating to the contents of the Bulletin should be sent to the Editorial Committee

- by email to rdc-cdr@statcan.ca
- or by regular mail to:
  The Editorial Committee, RDC Information and Technical Bulletin
  McMaster Research Data Centre, Statistics Canada
  Mills Library Memorial
  Library Room 217
  1280 Main Street West,
  Hamilton, Ontario L8S 4L6
  Canada

The review process

The editorial committee conducts the initial article review process. Editors may solicit past authors of the Bulletin or subject matter experts to participate in the process. The articles submitted to the Bulletin are reviewed for accuracy, consistency, and quality.

Upon completion of the initial review, the articles undergo both peer and institutional review. Peer reviews are conducted in accordance with Statistics Canada's Policy on the Review of Information Products. Institutional reviews are be conducted by members of senior management within Statistics

Canada in order to ensure that the material does not compromise the Agency's guidelines of standards, or reputation for non-partisanship, objectivity and neutrality.

For more information about the review process, please contact the Editorial Committee at the addresses above.