The Stata Journal

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press. Stata and Mata are registered trademarks of StataCorp LP.

The Stata Journal publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those teaching, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the Stata Journal, including information for authors, see the web page

http://www.stata-journal.com

The Stata Journal is indexed and abstracted in the following:

- Science Citation Index Expanded (also known as SciSearch®)
- CompuMath Citation Index®

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates:

Subscriptions mailed to US and Canadian addresses:

3-year subscription (includes printed and electronic copy)	$165
2-year subscription (includes printed and electronic copy)	$115
1-year subscription (includes printed and electronic copy)	$ 59
1-year student subscription (includes printed and electronic copy)	$ 35
1-year university library subscription (includes printed and electronic copy)	$ 75
1-year institutional subscription (includes printed and electronic copy)	$175

Subscriptions mailed to other countries:

3-year subscription (includes printed and electronic copy)	$240
2-year subscription (includes printed and electronic copy)	$165
1-year subscription (includes printed and electronic copy)	$ 84
3-year subscription (electronic only)	$160
1-year student subscription (includes printed and electronic copy)	$ 59
1-year university library subscription (includes printed and electronic copy)	$ 95
1-year institutional subscription (includes printed and electronic copy)	$195

Back issues of the Stata Journal may be ordered online at

http://www.stata.com/bookstore/sj.html

The Stata Journal is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the Stata Journal, StataCorp, 4905 Lakeway Drive, College Station TX 77845, USA, or email sj@stata.com.

Volume 7 Number 4 2007

The Stata Journal

The Stata Journal (2007)
7, Number 4, pp. 445–464

Multiple imputation of missing values: further update of ice, with an emphasis on interval censoring

Patrick Royston
Cancer and Statistical Methodology Groups
MRC Clinical Trials Unit
London UK

Abstract. Multiple imputation of missing data continues to be a topic of considerable interest and importance to applied researchers. In this article, the `ice` package for multiple imputation is further updated. Special attention in this article is paid to imputing interval-censored observations, and a suggestion to use imputation of right-censored survival data to elucidate covariate effects graphically.

Keywords: st0067_3, ice, uvis, micombine, ice_reformat, multiple imputation, interval censoring, visualization, censored survival data

1 Introduction

Royston (2004) introduced `mvis`, an implementation for Stata of MICE, a method of multiple multivariate imputation of missing values under missing-at-random (MAR) assumptions. In a second article, Royston (2005a) described `ice`, an upgrade incorporating various improvements and changes to the software based on personal experience, discussion with colleagues, and user requests. An update of `ice` was described by Royston (2005b), and this article presents a further update. The changes are less substantial than before but nevertheless, I feel, are important enough to warrant a paper. I will focus particularly on the new `interval()` option for imputing interval-censored observations. This option may be used with covariates recorded only in categories (such as stated income in surveys or a different application) to impute the *missing* part of left-, interval-, or right-censored time-to-event observations.

The current `ice` system consists of three ado-files: `ice`, `uvis`, and `micombine`. Previous components `mijoin` and `misplit` are out of date and have been removed. This is the final release of `micombine`, since a related article (Carlin, Galati, and Royston 2008) describes a new ado-file, `mim`, which replaces `micombine` and has more facilities.

Finally, another ado-file, `ice_reformat`, is included in the present release for backward compatibility of data files. It converts `.dta` files created by earlier releases of `ice` to the format required by `mim`.

2 Syntax

`ice` *mainvarlist* [*if*] [*in*] [*weight*] [, `boot`[(*varlist*)] `cc(`*ccvarlist*`)` `cmd(`*cmdlist*`)` `cycles(`*#*`)` `dropmissing` `dryrun` `eq(`*eqlist*`)` `genmiss(`*string*`)` `id(`*string*`)` `m(`*#*`)` `interval(`*intlist*`)` `match`[(*varlist*)] `noconstant` `nopp` `noshoweq` `nowarning` `on(`*varlist*`)` `orderasis` `passive(`*passivelist*`)` `replace` `saving(`*filename* [, `replace`]`)` `seed(`*#*`)` `substitute(`*sublist*`)` `trace(`*filename*`)`]

`uvis` *regression_cmd* {*yvar* | *llvar ulvar*} *xvarlisti* [*if*] [*in*] [*weight*], `gen(`*newvarname*`)` [`boot` `match` `noconstant` `nopp` `replace` `seed(`*#*`)`]

where *regression_cmd* may be `intreg`, `logistic`, `logit`, `mlogit`, `ologit`, or `regress`. All weight types supported by *regression_cmd* are allowed. *llvar ulvar* are required with `uvis intreg`.

`micombine` *regression_cmd* [*yvar*] [*covarlist*] [*if*] [*in*] [*weight*] [, `br` `detail` `eform(`*string*`)` `genxb(`*newvarname*`)` `impid(`*varname*`)` `lrr` `noconstant` `obsid(`*varname*`)` `svy`[(*svy_options*)] *regression_cmd_options*]

where *regression_cmd* includes `clogit`, `cnreg`, `glm`, `logistic`, `logit`, `mlogit`, `nbreg`, `ologit`, `oprobit`, `poisson`, `probit`, `qreg`, `regress`, `rreg`, `stcox`, `streg`, or `xtgee`. Other *regression_cmd*s will work but not all have been tested by the author. All weight types supported by *regression_cmd* are allowed.

`ice_reformat` *filename*, `replace`

3 What is new?

The principal changes to `ice` (version 1.4.4), `uvis` (version 1.2.7), and `micombine` (version 1.1.6) compared with the November 2005 release (Royston 2005b) (versions 1.1.1, 1.1.0, and 1.1.0, respectively) are as follows:

1. `ice` now checks for perfect prediction of the outcome when logistic regression (`logit`, `logistic`, `ologit`, `mlogit`) is used to impute a binary, ordered or unordered categorical variable. If perfect prediction is found, `ice` and `uvis` work with a modified type of logistic regression command. The dataset is extended by several pseudo-observations in such a way that nonperfect prediction results and the estimated β regression coefficient and its SE are finite. This approach guarantees sensible imputations in such cases. Treatment of the perfect prediction bug can be suppressed by using the `nopp` option of `ice` or `uvis`.

2. An `interval()` option has been added to `ice`. This option is the key change and its functionality is the main topic of the present article.
3. The imputation and observation indicator variables have been changed from `_j` and `_i` to `_mj` and `_mi`.
4. The original data, including missing values, are output by `ice` to the file of imputations, indexed by `_mj = 0`.
5. `ice`'s `substitute()` option has been improved by making it imply `passive()` for the relevant variables. This saves typing and reduces the chance of making a mistake in the specification.
6. `dropmissing`, `orderasis`, and `nowarning` options have been added to `ice`.
7. A `nopp` option has been added to `ice` and `uvis`.
8. The `using` *filename* syntax has been replaced with a `saving(`*filename*[`, replace`]`)` option. The old syntax still works but is undocumented.
9. The help file for `ice`/`uvis` has been modernized.
10. `svy` commands for Stata 8 and 9 are now supported by `micombine`.
11. `uvis` supports imputation of interval-censored variables with the `uvis intreg` syntax.
12. `ice_reformat` replaces *filename* with a new version of the data, with the following changes:
    a. `_i` and `_j` are renamed to `_mi` and `_mj`, respectively.
    b. The contents of characteristic `char _dta[mi_id]` are changed from `_i` to `_mi`.

4 Options

Only new or changed options are described.

Options for ice

`dropmissing` is a feature designed to save memory when using the file of imputed data created by `ice`. It omits from *filename* all observations that are not in the estimation sample, that is, for which either (i) they are filtered out by `if` or `in`, or a nonpositive weight, or (ii) the values of all variables in *mainvarlist* are missing. This option provides a clean analysis file of imputations, with no missing values. The observations not in the estimation sample are also omitted from the original data, stored as the imputation indexed by `_mj==0` in *filename*.

`interval(`*intlist*`)` imputes interval-censored variables. An interval-censored value is one that is known to lie in an interval $[a, b]$, where a and b are finite and $a \leq b$; in $(-\infty, b]$; or in $[a, \infty)$. When either terminal is infinite, we have left or right censoring, respectively. *intlist* has the syntax *varname*: *llvar ulvar* $\big[$ `,` *varname*: *llvar ulvar* ... $\big]$, where each *varname* is an interval-censored variable, each *llvar* contains the lower bound (a) for *varname* and each *ulvar* contains the upper bound (b) for *varname* (or a missing value to represent $\pm\infty$). The supplied values of *varname* are irrelevant because they will be replaced anyway; it is only required that *varname* exist. Observations with *llvar* missing and *ulvar* present are left-censored for *varname*. Observations with *llvar* present and *ulvar* missing are right-censored for *varname*. Observations with *llvar* = *ulvar* are complete, and no imputation is done for them. Observations with both *llvar* and *ulvar* missing are imputed assuming an uncensored normal distribution.

`nopp` suppresses treatment of the perfect prediction bug.

`nowarning` suppresses warning messages.

`orderasis` enters the variables in *mainvarlist* into the MICE algorithm in the order given. The default is to order them according to the number of missing values; the variable with the least missingness gets imputed first and so on.

`saving(`*filename* $\big[$ `, replace` $\big]$`)` saves the imputations to *filename*. `replace` allows *filename* to be overwritten with new data. Unless `dryrun` has been specified, `saving()` is required.

4.1 Options for uvis

`nopp` suppresses treatment of the perfect prediction bug.

4.2 Options for micombine

`svy`$\big[$ `(`*svy_options*`)` $\big]$ (Stata 9) performs survey regression. The prefix `svy:` is placed before *regression_cmd*. If *svy_options* are supplied then `,` *svy_options* is placed between `svy` and the colon. The data must be `svyset` before this option is used and before `ice` is used to impute missing values. That the data have been `svyset` is inherited by the file of imputations created by `ice`.

`svy` (Stata 8) performs survey regression. The prefix `svy` is placed before *regression_cmd*, so that for example `micombine regress` ... `, svy` is interpreted as `svy regress` .... Options for survey regression are included as options to `micombine`. The data must be `svyset` before the `svy` option is used. This must be done before `ice` is used to impute missing values. That the data have been `svyset` is inherited by the file of imputations created by `ice`.

5 Interval censoring

5.1 Introduction

A value x is said to be *interval-censored* on $[a, b]$ if x is known to lie between a and b but its exact value is not known. An example is a sample survey in which respondents are asked to indicate an income range (e.g., \$0–\$5,000, \$5,001–\$10,000) but not their precise income. In clinical medicine it is not uncommon for continuous or ordinal values to be recorded only in categories. In node-positive breast cancer, for example, the most important prognostic variable, the number of positive lymph nodes (say, `nodes`), is sometimes converted to the lymph node stage (`nstage`), coded as 0 for node negative (`nodes` = 0), 1 for 1–3, 2 for 4–9, and 3 for 10+ nodes. A dataset compiled from different centers could even contain a mixture of `nodes` and `nstage` values, depending on local practice.

Interval-censored data includes some important special cases. For example, with right censoring (e.g., *time-to-event* data), a datum x may be completely observed, in which case, $a = b = x$, or known to be at least x_0, in which case, $a = x_0$ and $b = +\infty$ and x is right-censored. A datum left-censored at x_0 has $a = -\infty$ and $b = x_0$. In the `nodes` example, observations in `nstage` category 0 are exact, whereas those in categories 1 and 2 are interval-censored and those in category 3 are right-censored.

Sometimes, for example, for modeling or descriptive purposes, the continuous values underlying an interval-censored variable need to be imputed. For example, `nodes` is the most powerful predictor of outcome in primary breast cancer. If `nstage` is recorded for some patients and `nodes` for others, the most informative analysis of the dataset may require imputation of exact value of `nodes` for cases with only `nstage` known. One may also be faced with imputing missing values of `nodes`/`nstage`.

An interesting application of imputing interval-censored observations is with time-to-event (e.g., survival) data. Visualization of survival times and other explorations of the data may be more easily achieved with the censored observations replaced with imputed values. I will illustrate this scenario in some detail in section 6.

5.2 The model

In `ice`, imputation of interval-censored observations is based on the assumption that the underlying (*latent*) continuous variable is normally distributed. The Stata command `intreg` is used to estimate the mean and variance of this distribution, based on the interval-censored (doubly truncated) values and on covariates comprising the imputation model. It is assumed that the underlying continuous variable follows a truncated normal distribution in the observed categories. To help make the modeling more realistic, the software allows the imposition of an absolute lower and/or upper limit on the imputed values. This is implemented by truncation of the normal distribution at the specified value(s).

Figure 1 shows the principle of imputation sampling here. For example, an observation of x is known to lie in $[1, 3]$ and a continuous value is sampled from the shaded density. This density takes into account the mean and SD of the underlying normal distribution (bell-shaped curve). These parameters are estimated by `intreg` from the covariates comprising the imputation model. To ensure that the imputations are proper, the parameter values actually used are drawn from their estimated posterior predictive distribution, as is routinely done by `ice`.

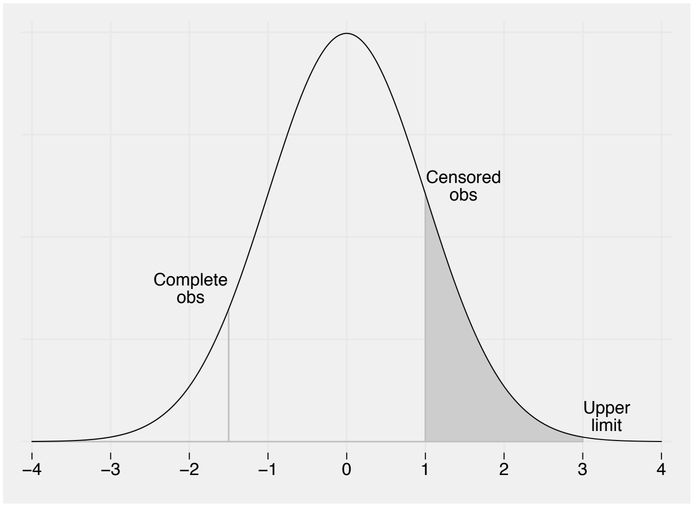

Figure 1: Interval censoring and the normal density function. The gray area indicates an observation that lies somewhere between 1 and 3. `ice` with the `interval()` option would sample from the density corresponding to the gray area.

5.3 In practice

To perform imputation with the `interval()` option, `ice` requires two variables: *ll* containing the lower boundary a for each observation of a variable x, and *ul*, containing the upper boundary, b. Each value of *ll* and *ul* may be missing or nonmissing, but *ll* must never exceed *ul*. Missing values of *ll* indicate left-censored observations; of *ul*, right-censored observations; and of both variables, truly missing observations.

The normality assumption must be plausible for the procedure to be successful in the sense of generating imputations with a realistic distribution. When the variable in question is intrinsically positive and positively skewed, a log transformation is often advantageous since the imputed values are guaranteed to be positive after back-transformation (exponentiation). If a subset of exactly observed values is available, an approximate transformation to normality can often be found by power transformation followed by a normal plot of the transformed variable (`qnorm` command). One is looking for approximate linearity of the normal plot. If the variable has zeros, a common practice is to add 1 before seeking such a transformation.

Variables that are integer-valued (e.g., `nodes`) and interval-censored (e.g., `nstage`) present a further challenge. Clearly the distribution of the underlying latent variable is not really continuous, but such an assumption is a convenient fiction. The case can be handled by judicious rounding. Consider `nstage`. Recalling that the categories 1–3 of `nstage` represent `nodes` values of 1–3, 4–9, and 10+, one might assign the values 1, 4, and 10 to *ll* and 3, 9, and "." (missing) to *ul*. However, with this scheme the imputed continuous values will have gaps in the intervals $(3, 4)$ and $(9, 10)$. A better scheme is to pretend that an observation of k nodes is really an underlying continuous value in the range $(k - 0.5, k + 0.5)$ and specify *ll* as 0.5, 3.5, and 9.5, and *ul* as 3.5, 9.5, and missing. The final step in such a scheme is to round the continuous imputed values to the nearest integer. By making the lower limit of the lowest group 0.5, we are guaranteed that imputed values will not be less than 1 after rounding.

If the variable requires a preliminary transformation to achieve approximate normality, the extra steps of pretransforming *ll* and *ul* and posttransforming them back to the original scale after imputation must be performed. In the `nodes` example, rounding to integers would be the final step.

5.4 Example

Preliminaries

I will illustrate the `nodes` example with real data. Consider the variable `x5` (`nodes`, number of positive lymph nodes) in the breast cancer dataset `brcaex.dta` analyzed by Royston (2004). The distribution of `x5` takes the integers 1, 2, ..., and has coefficient of skewness $\sqrt{b_1} = 2.9$, which is large. More than 25% of the values are 1.

Figure 2 shows two normal plots of `x5`.

(*Continued on next page*)

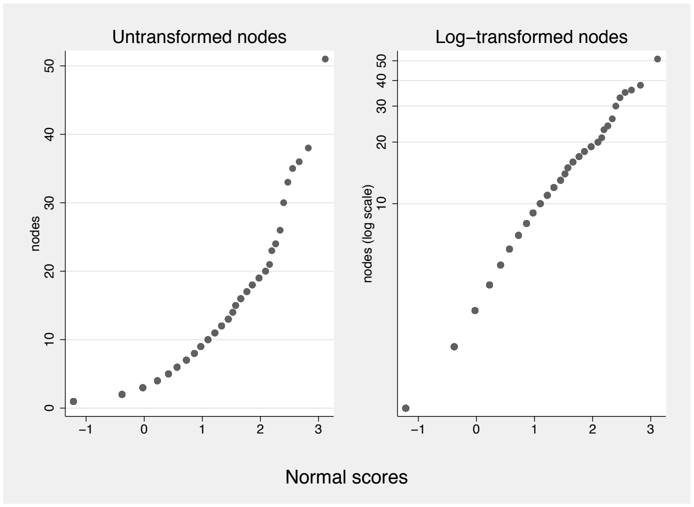

Figure 2: Normal Q–Q plots of untransformed (left panel) and log-transformed (right panel) `nodes`

However, these are not the usual normal plots. Instead, I have created the normal scores variable, `z`, corresponding to `x5` by using the ado-file `nscore` provided with this article. The syntax used is simply `nscore z = x5`. The difference between `nscore` and the factory-supplied program `qnorm` is that `nscore` averages normal scores corresponding to ties in the source variable. This greatly facilitates a visual assessment of linearity, because each horizontal sequence of markers representing tied values is removed from the normal plot (i.e., scatterplot of `x5` against `z`) and replaced with one point. If desired the multiplicity of these points can be indicated by weighting the plot by the number of values at each point.

Clearly untransformed `x5` is far from normal, but log `x5` is reasonably normal (it has $\sqrt{b_1} = 0.3$). Further refinement could be achieved, for example, by adding a constant to `x5` before transformation and tuning the constant to make the normal plot as linear as possible, but this is not really necessary.

Suppose that we did not have the raw values of `x5` but have only the `nstage` categorization. Assessing normality is obviously more difficult now. However, provided we have at least a reasonable idea of the mean of `x5` in each category, perhaps from other datasets, we can get some idea of whether a log transformation makes the data more normal. We replace each category value (1, 2, or 3) with our estimate of the category mean. Here we know the category means: 1.7, 5.9, and 15.2. In reality we might estimate them as the category midpoints (2 and 6.5 for categories 1 and 2) and make an informed guess, say, 14 for category 3. A simple measure of normality (equivalent, in fact, to the Shapiro–Francia statistic) is the correlation coefficient between the mean (or log mean) category values and the category normal equivalent deviates (NEDs). As before, the NEDs are computed by `nscore` and averaged over tied values, here giving just three distinct values: −0.72, 0.54, and 1.55. The resulting correlations are 0.9768 for the untransformed and 0.9985 for the log-transformed means. The log transformation is therefore favored.

Imputation

I will now illustrate how to use `ice` to impute plausible values of `x5` from `nstage` as discussed above. A preliminary multivariable analysis showed that `nstage` is associated with `x3` (tumor size) and `x4a`/`x4b` (dummy variables for tumor grade 1/2/3), so these two variables are included in the imputation model. I added one minor modification: instead of allowing imputed numbers of nodes to be unlimited, I restricted them to a maximum of 55 (the maximum in the original data being 51). Limiting the range of imputed values is often sensible. Stata code to create $m = 10$ imputations is as follows:

```
. gen llnodes = log(0.5*(nstage==1) + 3.5*(nstage==2) + 9.5*(nstage==3))
. gen lunodes = log(3.5*(nstage==1) + 9.5*(nstage==2) +  55*(nstage==3))
. gen lnodes = .
(686 missing values generated)
. ice lnodes llnodes lunodes x3 x4a x4b, saving(nodesimp)
 > m(10) interval(lnodes: llnodes lunodes)
   #missing |
     values |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |        686      100.00      100.00
------------+-----------------------------------
      Total |        686      100.00
   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------
     lnodes | intreg  | x3 x4a x4b
    llnodes |         | [Lower bound for lnodes]
    lunodes |         | [Upper bound for lnodes]
         x3 |         | [No missing data in estimation sample]
        x4a |         | [No missing data in estimation sample]
        x4b |         | [No missing data in estimation sample]
------------------------------------------------------------------------
Imputing
[Only 1 variable to be imputed, therefore no cycling needed.]
1..2..3..4..5..6..7..8..9..10..file nodesimp.dta saved)
. use nodesimp, clear
(German breast cancer data)
. gen int nodes = round(exp(lnodes), 1)
(686 missing values generated)
```

`ice` reports 686 occurrences of 1 missing value because we initially assigned all values of `lnodes` to missing.

Figure 3 compares the imputed `nodes` values with the known values of `x5` in the first imputation (`_mj==1`).

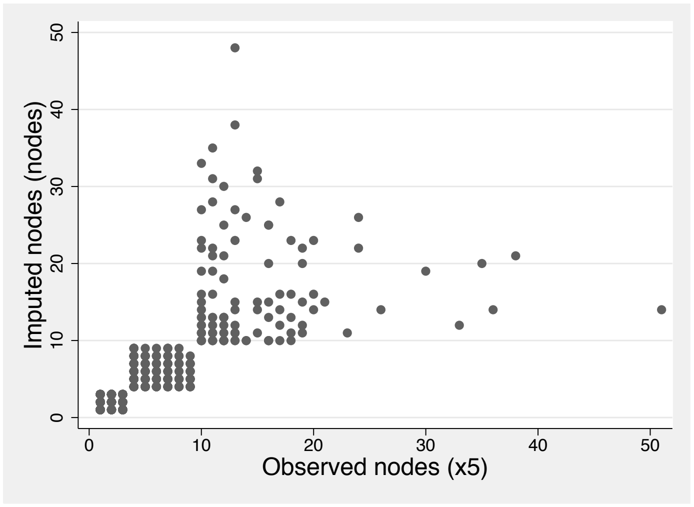

Figure 3: Imputation of interval-censored values of `x5`; comparison of original with imputed values in imputation 1

Because the imputation model does not explain much of the variation, there is considerable uncertainty in the imputations and hence scatter. The Spearman rank correlation between `nodes` and `x5` is between 0.82 and 0.84 across the imputations. Nevertheless, the imputation seems to have done a good job. Using Rubin's rules for combining estimates across imputations, the mean (SE) of `nodes` is 5.18(0.23) and of `x5` (the gold standard) is 5.01(0.21). The bias in the mean is negligible. The mean (SE) of the regression coefficient in a univariate Cox model on log(`nodes`) is 0.556(0.068), compared with 0.543(0.063) for log(`x5`).

6 Imputing right-censored survival data

6.1 Why bother?

As Royston, Parmar, and Altman (2008) discuss and illustrate, with censored survival data it is difficult to visualize and therefore to understand the distribution of the time-to-event outcome variable in relation to covariates. The Cox model, for example, is conceived in terms of hazard ratios, but these are rather indirectly related to differences in survival times. In a clinical trial, the experimental treatment may exhibit a substantial reduction in risk of an event compared with the control arm, as evidenced by a hazard ratio of, say, 0.7. The corresponding Kaplan–Meier survival curves for the two arms may look impressively separated in a plot. However, the actual distributions of time to event may overlap considerably. A scatterplot of these times will go a long way to correcting an overoptimistic impression of the effectiveness of the treatment. Judicious imputation of the right-censored times to event can provide the analyst with a tool that greatly assists inspection of such distributions and hence allows a more realistic assessment of the effect of a treatment at the level of individual survival times. Similar comments apply to the effects of prognostic variables.

6.2 Quantile–quantile plot of censored survival times

In primary breast cancer, time-to-disease recurrence is approximately lognormally distributed (Royston 2001). The marginal distribution of time to event may be assessed in a modified version of a normal quantile–quantile plot. Let $t_{(1)} \leq \cdots \leq t_{(n)}$ be the ordered survival or censoring times of n individuals with estimated survival probabilities (obtained by the Kaplan–Meier or some other suitable method) $S_1 \geq \cdots \geq S_n$. Write $z_j = -\Phi^{-1}(S_j)$ [in Stata, use the `invnormal()` function for $\Phi^{-1}(\cdot)$]. A scatterplot of the $t_{(j)}$ against the z_j is a normal quantile–quantile plot for censored data. Figure 4 shows such a plot for the breast cancer example dataset.

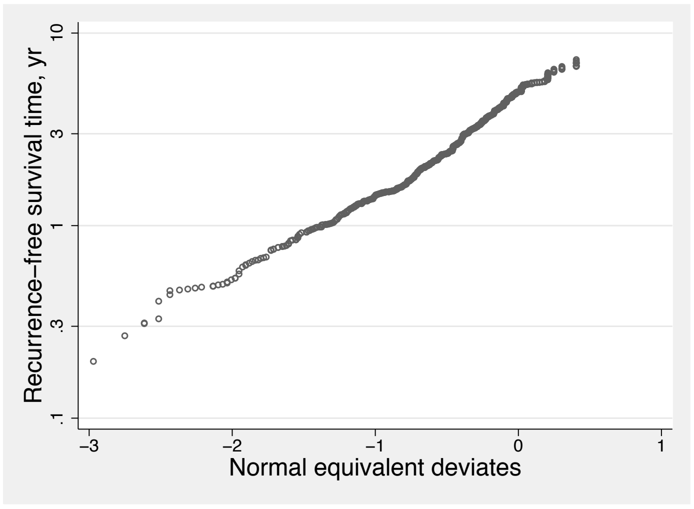

Figure 4: Normal quantile–quantile plot of censored recurrence-free survival time (RFS) data. Vertical axis is a log scale. Linearity suggests that the time-to-event is approximately lognormally distributed.

The times have been plotted on a log scale. The relationship is roughly linear, supporting a lognormal distribution as a reasonable approximation.

6.3 Doing the imputations

The right-censored times can be imputed by using `ice` with the `interval()` option. First, an imputation model is needed to allow for the possible effects of covariates. Because we are working with a lognormal distribution, a sensible approach is to build a multivariable model by using some type of censored-normal regression of the log times on prognostic factors in the dataset. First, let us consider what may be a reasonable upper limit for the imputed survival times. The lognormal distribution is longtailed. Unless we are careful, we may find ourselves creating implausible imputed times (e.g., a recurrence-free survival time of 300 years). We get around this problem by specifying the upper limit of time to be something realistic, for example, 90 minus the age of the patient (`x1`) at entry to the study. (All patients were well under 90 years of age at entry.) We can then use Stata's `mfp` command with `intreg` to find a predictive model based on fractional polynomial transformation of the influential continuous predictors, where needed:

```
. stset rectime censrec, scale(365.25) // time in years
. gen lnt = ln(_t)
. gen ll = lnt
. gen ul = cond(_d==0, ln(90-x1), lnt)
. mfp intreg ll ul x1 x2 x3 x4a x4b x5 x6 x7 hormon, select(.05) df(x5:2)
  (output omitted)
```

The selected model has the following variables (with power(s) in parentheses, when transformed—power 0 meaning log): `x1` $(-1,-1)$, `x4a`, `x5` (0), `x6` (0), and `hormon`. The residual SD (parameter `sigma`) is reported as 0.842. The variance explained by the model may be estimated as $R^2 = 1 - \mathrm{var}(y|\mathbf{x})/\mathrm{var}(y)$ and here is $1 - 0.842^2/0.976^2$ or about 26%. The value of $\mathrm{var}(y) = 0.976^2$ was found by running `intreg` with no covariates (i.e., `intreg ll ul`). The reported value of `sigma` is 0.976.

We now use this imputation model with `ice` to create 10 imputed datasets. The variables `ll` and `ul` are needed again:

```
. gen lnt = ln(_t)
. gen ll = lnt
. gen ul = cond(_d==0, ln(90-x1), lnt)
. fracgen x1 -1 -1
-> gen double x1_1 = X^-1
-> gen double x1_2 = X^-1*ln(X)
   (where: X = x1/10)
. fracgen x5 0
-> gen double x5_1 = ln(X)
   (where: X = x5/10)
. fracgen x6 0
-> gen double x6_1 = ln(X)
   (where: X = (x6+1)/1000)
```

```
. ice lnt ll ul x1_1 x1_2 x4a x5_1 x6_1 hormon, saving(brcaexi, replace) m(10)
> interval(lnt:ll ul)
  #missing |
    values |      Freq.     Percent        Cum.
-----------+-----------------------------------
         0 |        686      100.00      100.00
-----------+-----------------------------------
     Total |        686      100.00
   Variable | Command | Prediction equation
------------+---------+-------------------------------------------------------
        lnt | intreg  | x1_1 x1_2 x4a x5_1 x6_1 hormon
         ll |         | [Lower bound for lnt]
         ul |         | [Upper bound for lnt]
       x1_1 |         | [No missing data in estimation sample]
       x1_2 |         | [No missing data in estimation sample]
        x4a |         | [No missing data in estimation sample]
       x5_1 |         | [No missing data in estimation sample]
       x6_1 |         | [No missing data in estimation sample]
     hormon |         | [No missing data in estimation sample]
------------------------------------------------------------------------------

Imputing
[Only 1 variable to be imputed, therefore no cycling needed.]
1..2..3..4..5..6..7..8..9..10..file brcaexi.dta saved
```

6.4 Plots using the imputed data

Let us now consider visualizing the effect of hormonal treatment (hormon) on recurrence-free survival time. Figure 5 shows a Kaplan–Meier plot of the original, censored time variable according to hormon treatment status.

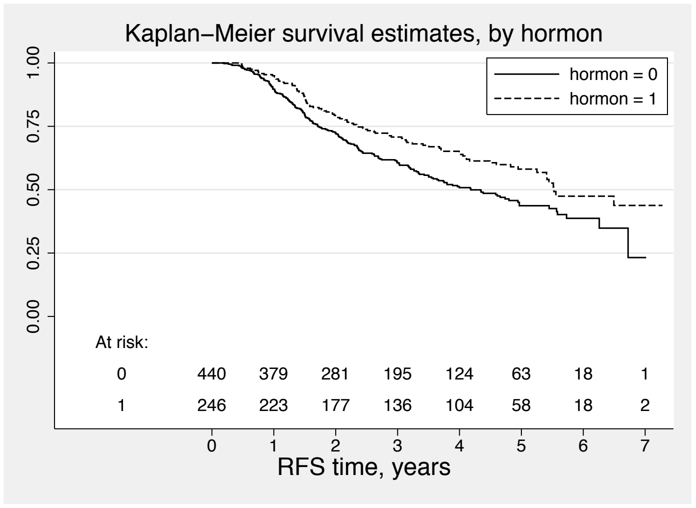

Figure 5: Kaplan–Meier plot of recurrence-free survival time according to hormonal treatment status (hormon)

There is visible white space between the curves, suggesting a large difference in survival. The parameter estimate for `hormon` in the original `intreg` model (adjusted for other predictors) is 0.27 (SE 0.08), suggesting that the treatment increases log RFS time on average by 0.27 or RFS time by about 31%.

Figure 6 is a dot plot of the observed and imputed RFS time in the first imputation (results for the other 9 imputations are similar).

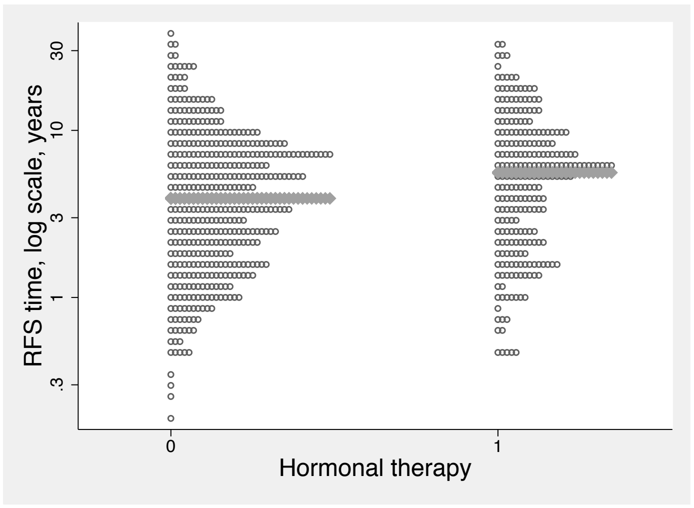

Figure 6: Comparison of time to RFS event for the patients untreated or treated with hormonal therapy (`hormon`) for the first imputation of the RFS time. Horizontal lines show the medians. The vertical scale of the dot plot is logarithmic.

The large degree of overlap between the two survival time distributions is now obvious. The therapy certainly has some effect but is not a miracle cure.

Figure 7 shows a smoothed scatterplot of the relationship between log RFS time and the strongest predictor (number of positive lymph nodes, `x5`) in the first imputation.

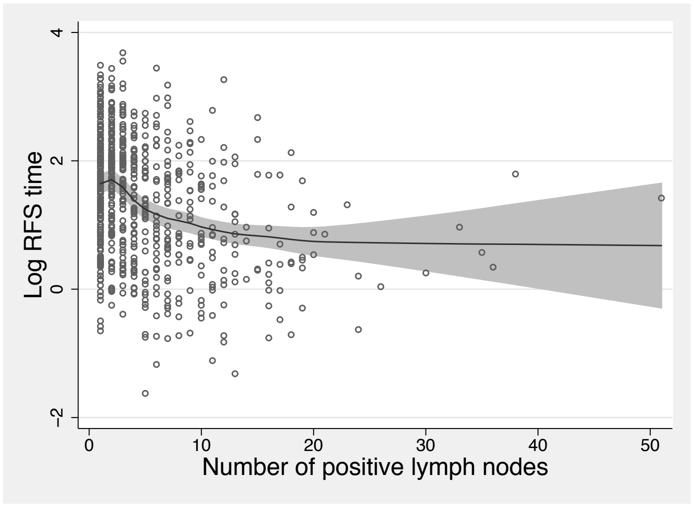

Figure 7: Relation between log RFS time and number of positive lymph nodes (`x5`) in the first imputation, with running-line smooth and 95% pointwise confidence interval

The smoothing was done by using a running-line smoother (Sasieni, Royston, and Cox 2005). A clear nonlinear relationship is present, but there is considerable random variation around the regression line. The Spearman correlation between time and `x5` is −0.31.

6.5 To model or not to model?

Having obtained m complete imputed datasets and having seen the advantages of really getting to grips with the times to event at the individual patient level, it is tempting to try to build new models with the imputed data. First, the parameters of the imputation model are faithfully reproduced (apart from minor random variation) in the multiply imputed dataset. Because `intreg` assumes a truncated normal distribution on the log survival times, it is appropriate to use `regress` on log t followed by application of Rubin's rules (Rubin 1987) to estimate the parameters of the original imputation model in the imputed data. The original (`intreg`) and reestimated (`regress`) parameters are shown in table 1.

(*Continued on next page*)

Table 1: Comparison of regression coefficients and their standard errors for the `intreg` model on the original data and the `regress` model on the imputed data

Predictor	Original (`intreg`)			Imputed (`regress`)		
	$\widehat{\beta}$	SE	$\widehat{\beta}$/SE	$\widehat{\beta}$	SE	$\widehat{\beta}$/SE
`x1`$^{-2}$	−9.65	2.38	−4.05	−9.51	2.50	−3.81
`x1`$^{-0.5}$	17.53	4.62	3.79	17.15	4.70	3.65
`x4a`	−0.294	0.127	−2.31	−0.284	0.124	−2.28
ln `x5`	−0.328	0.039	−8.41	−0.328	0.040	−8.24
ln (`x6`+1)	0.126	0.020	6.32	0.125	0.021	6.02
`hormon`	0.270	0.079	3.41	0.260	0.077	3.36
`_cons`	−1.99	1.04	−1.91	−1.90	1.06	−1.80

Apart from a small amount of random variation, the parameter estimates from the two models are identical; the SEs are usually slightly larger and the $\widehat{\beta}$/SE values slighter smaller in the imputed data. This is as expected, because the imputation involves the injection of random variation, and with only $m = 10$ imputations, a little information is inevitably lost. As m is increased, the similarity of the $\widehat{\beta}$s and of the SEs increases (data not shown).

The imputed dataset faithfully reproduces the characteristics assumed in the original model on which the imputations are based. We assumed a truncated lognormal distribution for the log survival times with certain parameters and functional forms for the effects of covariates, which is what we got.

Going beyond the imputation model may cause problems, however. For example, it is known that there is an interaction between hormonal treatment (`hormon`) and estrogen receptor status (`x7`). Royston and Sauerbrei (2004) showed that the interaction can be adequately modeled by the product term `hormon`×(`x7`+1)$^{-0.5}$. Let us call this interaction variable `x7h`. Suppose that we extended the original `intreg` model by including the terms `hormon` and (`x7`+1)$^{-0.5}$ (i.e., the main effects for the interaction) and `x7h`, estimated the parameters, and then reestimated them by using `micombine regress` or `mim: regress` in the imputed dataset. Table 2 shows the resulting parameter estimates.

Table 2: Comparison of regression coefficients and their standard errors for the `intreg` model on the original data and the `regress` model on the imputed data. The interaction between `x7` and `hormon` is examined.

Predictor	Original (`intreg`)			Imputed (`regress`)		
	$\widehat{\beta}$	SE	$\widehat{\beta}$/SE	$\widehat{\beta}$	SE	$\widehat{\beta}$/SE
`hormon`	0.433	0.108	3.99	0.370	0.113	3.29
$(\texttt{x7}+1)^{-0.5}$	0.113	0.174	0.65	0.089	0.182	0.49
`x7h`	−0.554	0.252	−2.20	−0.386	0.257	−1.50

In the original `intreg` model, `x7h` is significant at the $P = 0.02$ level, whereas in the `regress` model in the imputed dataset, we have $P = 0.13$; the corresponding $\widehat{\beta}$ is reduced in magnitude from −0.55 to −0.39. Imputing using a wrong (or rather, incomplete) model has introduced a nontrivial amount of bias into the estimated interaction between `hormon` and `x7`.

Of course, such a finding is neither surprising nor specific to this situation. An inadequate imputation model can always induce bias of this sort; hence, the generally accepted advice is to use a large imputation model rather than a parsimonious one and to include interactions when necessary. We went against such advice here by building the imputation model with selection of variables and functions at the 5% significance level and not considering interactions at all.

Nevertheless, there is certainly a question as to whether one should include interactions or other higher-order terms in the imputation model. Generally, the issue is how to strike a satisfactory balance between a sufficiently comprehensive imputation model and the possibility of instability due to a grossly overfitted model. In the current example, we already knew from earlier work that an interaction existed, but usually such prior information will not be available. Developing a satisfactory imputation model is still an open issue in the practical analysis of multiple imputed datasets.

With right-censored survival times, a pragmatic approach may be to use imputation simply as a tool to explore the implications of a model fitted to the original data in more detail, as we have done here with the `intreg` approach. For example, the availability of scatterplot smoothers for the imputed data makes it easier to get a feel for the relationships within the data and to look for lack of fit. Nevertheless, to make this process safer and more informative, it is probably sensible to start with a rather larger imputation model. Here we could have included all the available predictors in the `intreg` model and perhaps allowed `mfp` to detect and model nonlinearity at a more relaxed significance level, such as 0.2. We could have also included in the model the interaction between $(\texttt{x7}+1)^{-0.5}$ and `hormon`.

6.6 Incompatibility between imputation and substantive models

Suppose that, having obtained m imputations as described above, we had contemplated doing not ordinary regression but Cox regression on the imputed dataset. Let us compare the regression coefficients of a Cox model estimated on the original and imputed datasets. The imputation model assumes one type of error structure (linear regression on log time) whereas the Cox model assumes another (a proportional hazards model). What effect does this incompatibility have on the $\widehat{\beta}$s?

Table 3 compares the $\widehat{\beta}$s and shows the percentage bias between the two ways of fitting the Cox model.

Table 3: Parameter estimates for a Cox model on the original data and imputed data assuming an incompatible imputation model

Predictor	$\widehat{\beta}$ in Cox model for		% bias
	Original data	Imputed data	
`x1`$^{-2}$	16.6	15.1	−9
`x1`$^{-0.5}$	−30.1	−29.0	−3
`x4a`	0.497	0.231	−54
ln `x5`	0.508	0.366	−28
ln (`x6`+1)	−0.179	−0.130	−27
`hormon`	−0.390	−0.273	−30

The results show that the incompatibility between the imputation and substantive models induces major bias in most of the estimated βs. The bias is always toward the null (i.e., brings the $\widehat{\beta}$s closer to zero than they should be).

Clearly, there are pitfalls that the user should beware of when contemplating imputing a censored outcome variable. These will also apply (but to a lesser extent, because extrapolation is less likely to be involved) to imputing missing values of a noncensored outcome variable.

7 Final comment

Here I have focused on multiple imputation of interval- or right-censored observations using `ice` and illustrated how judicious use of the `interval()` option may be helpful. I have also pointed out serious pitfalls when the method is used without care to complete a right-censored time-to-event variable. I believe that the user should be wary of literature claims of robustness to misspecification when such a type of imputation is used. For example, Hsu et al. (2007) use proportional hazards models to derive risk scores that help impute interval-censored outcome variables in a nonparametric way. The authors state “In addition to its robustness in this application, the general approach of multiple imputation methods has features that make it attractive. One such feature is that after imputation the data analyst is now free to choose and can easily perform any analysis appropriate for the goals of their study. Conditions for the appropriateness of this philosophy are discussed in Reference [23]". This advice appears to me dangerous—unless the reader carefully consults (and is sufficiently equipped to understand the implications of) Hsu et al. (2007)'s Reference 23 (Meng 1994). I would not, for example, advocate applying linear regression methods to such a multiply imputed dataset, because as far as I understand it, the imputation method implicitly assumes proportional-hazards effects of covariates. The result would be seriously biased regression estimates, as in table 3.

8 Acknowledgment

Ian White suggested and programmed the method used by `ice` and `uvis` to avoid the problem of perfect prediction with the `logit`, `ologit`, and `mlogit` commands.

9 References

Carlin, J. B., J. C. Galati, and P. Royston. 2008. A new framework for managing and analyzing multiply imputed data in Stata. *Stata Journal.* Forthcoming.

Hsu, C., J. M. G. Taylor, S. Murray, and D. Commenges. 2007. Multiple imputation for interval censored data with auxiliary variables. *Statistics in Medicine* 26: 769–781.

Meng, X. L. 1994. Multiple-imputation inferences with uncongenial sources of input (with discussion). *Statistical Science* 9: 538–573.

Royston, P. 2001. The lognormal distribution as a model for survival time in cancer, with an emphasis on prognostic factors. *Statistica Neerlandica* 55: 89–104.

———. 2004. Multiple imputation of missing values. *Stata Journal* 4: 227–241.

———. 2005a. Multiple imputation of missing values: update. *Stata Journal* 5: 188–201.

———. 2005b. Multiple imputation of missing values: update of ice. *Stata Journal* 5: 527–536.

Royston, P., M. K. B. Parmar, and D. G. Altman. 2008. Visualizing length of survival in time-to-event studies: a complement to Kaplan–Meier plots. *Journal of the National Cancer Institute* 100: 92–97.

Royston, P., and W. Sauerbrei. 2004. A new approach to modelling interactions between treatment and continuous covariates in clinical trials by using fractional polynomials. *Statistics in Medicine* 23: 2509–2525.

Rubin, D. B. 1987. *Multiple imputation for non-response in surveys.* New York: Wiley.

Sasieni, P., P. Royston, and N. J. Cox. 2005. Symmetric nearest neighbor linear smoothers. *Stata Journal* 5: 285.

About the author

Patrick Royston is a medical statistician with 30 years of experience, with a strong interest in biostatistical methodology and in statistical computing and algorithms. He works in clinical trials and related research issues in kidney cancer and other cancers. Currently, he is focusing on problems of model building and validation with survival data, including prognostic factors studies; on complex sample size problems in clinical trials with a survival-time endpoint; on writing a book on multivariable regression modeling; and on new trial designs.

The Stata Journal (2007)
7, Number 4, pp. 465–506

Enhanced routines for instrumental variables/generalized method of moments estimation and testing

Christopher F. Baum
Department of Economics
Boston College
Chestnut Hill, MA
baum@bc.edu

Mark E. Schaffer
Economics Department
Heriot–Watt University
Edinburgh, UK
m.e.schaffer@hw.ac.uk

Steven Stillman
Motu Economic Public Policy Research
Wellington, New Zealand
stillman@motu.org.nz

Abstract. We extend our 2003 paper on instrumental variables and generalized method of moments estimation, and we test and describe enhanced routines that address heteroskedasticity- and autocorrelation-consistent standard errors, weak instruments, limited-information maximum likelihood and k-class estimation, tests for endogeneity and Ramsey's regression specification-error test, and autocorrelation tests for instrumental variable estimates and panel-data instrumental variable estimates.

Keywords: st0030_3, ivactest, ivendog, ivhettest, ivreg2, ivreset, overid, ranktest, instrumental variables, weak instruments, GMM, endogeneity, heteroskedasticity, serial correlation, HAC standard errors, LIML, CUE, overidentifying restrictions, Frisch–Waugh–Lovell theorem, RESET, Cumby–Huizinga test

1 Introduction

In Baum, Schaffer, and Stillman (2003), we discussed instrumental variables (IV) estimators in the context of generalized method of moments (GMM) estimation and presented Stata routines for estimation and testing consisting of the `ivreg2` suite. Since that time, those routines have been considerably enhanced and more routines have been added to the suite. This paper presents the analytical underpinnings of both basic IV/GMM estimation and these enhancements and describes the enhanced routines. Some of these features are now also available in Stata 10's `ivregress`, whereas others are not.

The additions include the following:

- Estimation and testing that is robust to and efficient in the presence of arbitrary serial correlation.

- A range of test statistics that allow the user to address the problems of underidentification or weak identification, including statistics that are robust in the presence of heteroskedasticity, autocorrelation, or clustering.

- Three additional IV/GMM estimators: the GMM continuously updated estimator (CUE) of Hansen, Heaton, and Yaron (1996); limited-information maximum likelihood (LIML); and k-class estimators.

- A more intuitive syntax for GMM estimation: the `gmm2s` option requests the two-step feasible efficient GMM (EGMM) estimator, which reduces to standard IV/2SLS if no robust covariance matrix estimator is also requested. The `cue` option requests the continuously updated GMM estimator, which reduces to standard LIML if no robust covariance matrix estimator is also requested.

- A more intuitive syntax for a "GMM distance" or C test of the endogeneity of regressors.

- An option that allows the user to "partial out" regressors: something that is particularly useful when the user has a rank-deficient estimate of the covariance matrix of orthogonality conditions (common with the `cluster()` option and singleton dummy variables).

- Several advanced options, including options that will speed up estimation using `ivreg2` by suppressing the calculation of various checks and statistics.

- A version of Ramsey's regression specification-error test (RESET), `ivreset`, that (unlike official Stata's `ovtest`) is appropriate for use in an IV context.

- A test for autocorrelation in time-series errors, `ivactest`, that (unlike official Stata's `estat bgodfrey`) is appropriate for use in an IV context.

We review the definitions of the method of IV and IV/GMM in the next section to set the stage. The following sections of the paper discuss each of these enhancements in turn. The last two sections provide a summary of `ivreg2` estimation options and syntax diagrams for all programs in the extended `ivreg2` suite.

2 IV and GMM estimation

GMM was introduced in Hansen (1982). It is now a mainstay of both econometric practice and econometrics textbooks. We limit our exposition here to the linear case, which is what `ivreg2` handles. The exposition here draws on Hayashi (2000). For more details and references, see also Baum, Schaffer, and Stillman (2003) and Baum (2006, chap. 8).

2.1 Setup

The equation to be estimated is, in matrix notation,

$$y = X\beta + u \tag{1}$$

with typical row

$$y_i = X_i\beta + u_i$$

The matrix of regressors X is $n \times K$, where n is the number of observations. Some of the regressors are endogenous, so that $E(X_i u_i) \neq 0$.

We partition the set of regressors into $[X_1 \;\; X_2]$, with the K_1 regressors X_1 assumed under the null to be endogenous and the $K_2 \equiv (K - K_1)$ remaining regressors X_2 assumed exogenous, giving us

$$y = [X_1 \;\; X_2][\beta_1' \;\; \beta_2']' + u$$

The set of IV is Z and is $n \times L$. This is the full set of variables that are assumed to be exogenous, i.e., $E(Z_i u_i) = 0$. We partition the instruments into $[Z_1 \; Z_2]$, where the L_1 instruments Z_1 are excluded instruments and the remaining $L_2 \equiv (L - L_1)$ instruments $Z_2 \equiv X_2$ are the included instruments/exogenous regressors:

$$\text{Regressors } X = [X_1 \;\; X_2] = [X_1 \;\; Z_2] = [\text{Endogenous} \;\; \text{Exogenous}]$$

$$\text{Instruments } Z = [Z_1 \; Z_2] = [\text{Excluded} \;\; \text{Included}]$$

The order condition for identification of the equation is $L \geq K$ implying there must be at least as many excluded instruments (L_1) as there are endogenous regressors (K_1) as Z_2 is common to both lists. If $L = K$, the equation is said to be *exactly identified* by the order condition; if $L > K$, the equation is *overidentified*. The order condition is necessary but not sufficient for identification; see section 7 for a full discussion.

2.2 GMM

The assumption that the instruments Z are exogenous can be expressed as $E(Z_i u_i) = 0$. We are considering linear GMM only, and here the L instruments give us a set of L moments

$$g_i(\beta) = Z_i' u_i = Z_i'(y_i - X_i\beta)$$

where g_i is $L \times 1$. The exogeneity of the instruments means that there are L moment conditions, or orthogonality conditions, that will be satisfied at the true value of β:

$$E[g_i(\beta)] = 0$$

Each of the L moment equations corresponds to a sample moment. For some given estimator $\widehat{\beta}$, we can write these L sample moments as

$$\overline{g}(\widehat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} g_i(\widehat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} Z_i'(y_i - X_i\widehat{\beta}) = \frac{1}{n}Z'\widehat{u}$$

The intuition behind GMM is to choose an estimator for β that brings $\overline{g}(\widehat{\beta})$ as close to zero as possible. If the equation to be estimated is exactly identified, so that $L = K$, then we have as many equations—the L moment conditions—as we do unknowns: the K coefficients in $\widehat{\beta}$. Here it is possible to find a $\widehat{\beta}$ that solves $\overline{g}(\widehat{\beta}) = 0$, and this GMM estimator is in fact a special case of the IV estimator as we discuss below.

If the equation is overidentified, however, so that $L > K$, then we have more equations than we do unknowns. Generally, it will not be possible to find a $\widehat{\beta}$ that will set all L sample moment conditions exactly to zero. Here we take an $L \times L$ weighting matrix W and use it to construct a quadratic form in the moment conditions. This gives us the GMM objective function:

$$J(\widehat{\beta}) = n\overline{g}(\widehat{\beta})'W\overline{g}(\widehat{\beta})$$

A GMM estimator for β is the $\widehat{\beta}$ that minimizes $J(\widehat{\beta})$:

$$\widehat{\beta}_{\text{GMM}} \equiv \arg\min_{\widehat{\beta}} J(\widehat{\beta}) = n\overline{g}(\widehat{\beta})'W\overline{g}(\widehat{\beta})$$

Linearly, we are considering, deriving, and solving the K first-order conditions $\{\partial J(\widehat{\beta})\}/(\partial\widehat{\beta}) = 0$ (treating W as a matrix of constants), which yields the GMM estimator:[1]

$$\widehat{\beta}_{\text{GMM}} = (X'ZWZ'X)^{-1}X'ZWZ'y \tag{2}$$

The GMM estimator is consistent for any symmetric positive-definite weighting matrix W, and thus there are as many GMM estimators as there are choices of weighting matrix W. Efficiency is not guaranteed for an arbitrary W, so we refer to the estimator defined in (2) as the possibly *inefficient* GMM estimator.

1. The results of the minimization, and hence the GMM estimator, will be the same for weighting matrices that differ by a constant of proportionality.

We are particularly interested in *efficient* GMM estimators with minimum asymptotic variance. Moreover, for any GMM estimator to be useful, we must be able to conduct inference, and for that we need estimates of the variance of the estimator. Both require estimates of the covariance matrix of orthogonality conditions, a key concept in GMM estimation.

2.3 Inference, efficiency, and the covariance matrix of orthogonality conditions

Denoted by S, the asymptotic covariance matrix of the moment conditions g

$$S = \text{AVar}\{\overline{g}(\beta)\} = \lim_{n\to\infty} \frac{1}{n} E(Z'uu'Z)$$

where S is an $L \times L$ matrix and $\overline{g}(\beta) = (1/n)Z'u$. That is, S is the variance of the limiting distribution of $\sqrt{n}\ \overline{g}$ (Hayashi 2000, 203).

The asymptotic distribution of the possibly inefficient GMM (IGMM) estimator can be written as follows. Let $Q_{XZ} \equiv E(X_i'Z_i)$. The asymptotic variance of the IGMM estimator defined by an arbitrary weighting matrix W is given by

$$V(\widehat{\beta}_{\text{GMM}}) = (Q_{XZ}'WQ_{XZ})^{-1}(Q_{XZ}'WSWQ_{XZ})(Q_{XZ}'WQ_{XZ})^{-1} \tag{3}$$

Under standard assumptions (see Hayashi 2000, 202–203, 209) the IGMM estimator is $\sqrt{n}$-consistent; that is,

$$\sqrt{n}\ (\widehat{\beta}_{\text{GMM}} - \beta) \rightarrow N\{0, V(\widehat{\beta}_{\text{GMM}})\}$$

where $\rightarrow$ denotes convergence in distribution.

Strictly speaking, therefore, we should perform hypothesis tests on $\sqrt{n}\ \widehat{\beta}_{\text{GMM}}$ by using (3) for the variance–covariance matrix. Standard practice, however, is to transform the variance–covariance matrix (3) rather than the coefficient vector (2). This is done by normalizing $V(\widehat{\beta}_{\text{GMM}})$ by $1/n$, so that the variance–covariance matrix reported by statistical packages such as Stata is in fact

$$V\left(\frac{1}{\sqrt{n}}\ \widehat{\beta}_{\text{GMM}}\right) = \frac{1}{n}(Q_{XZ}'WQ_{XZ})^{-1}(Q_{XZ}'WSWQ_{XZ})(Q_{XZ}'WQ_{XZ})^{-1} \tag{4}$$

The EGMM estimator makes use of an estimator with an optimal weighting matrix W, which minimizes the asymptotic variance of the estimator. This is achieved by choosing $W = S^{-1}$. If we substitute this into (2) and (4), we obtain the EGMM estimator

$$\widehat{\beta}_{\text{EGMM}} = (X'ZS^{-1}Z'X)^{-1}X'ZS^{-1}Z'y \tag{5}$$

with asymptotic variance

$$V(\widehat{\beta}_{\mathrm{EGMM}}) = (Q'_{XZ} S^{-1} Q_{XZ})^{-1}$$

Similarly,

$$\sqrt{n}\,(\widehat{\beta}_{\mathrm{EGMM}} - \beta) \rightarrow N\{0, V(\widehat{\beta}_{\mathrm{EGMM}})\}$$

and we perform inference on $\sqrt{n}\,\widehat{\beta}_{\mathrm{EGMM}}$ by using

$$V\left(\frac{1}{\sqrt{n}}\,\widehat{\beta}_{\mathrm{EGMM}}\right) = \frac{1}{n}(Q'_{XZ} S^{-1} Q_{XZ})^{-1} \tag{6}$$

as the variance–covariance matrix for $\widehat{\beta}_{\mathrm{EGMM}}$.

Obtaining an estimate of Q_{XZ} is straightforward: we simply use the sample analog

$$\frac{1}{n}\sum_{i=1}^{n} X'_i Z_i = \frac{1}{n} X' Z$$

If we have an estimate of S, therefore, we can conduct asymptotically correct inference for any GMM estimator, efficient or inefficient.

An estimate of S also makes the EGMM estimator a feasible estimator. In two-step feasible EGMM estimation an estimate of S is obtained in the first step, and we calculate the estimator and its asymptotic variance by using (5) and (6) in the second step.

2.4 Estimating the covariance matrix of orthogonality conditions

The first-step estimation of the matrix S requires the residuals of a consistent GMM estimator $\widetilde{\beta}$. Efficiency is not required in the first step of two-step GMM estimation, which simplifies the task considerably. But to obtain an estimate of S, we must make some further assumptions.

We illustrate this by using the case of independent but possibly heteroskedastic disturbances. If the errors are independent, $E(g_i g'_j) = 0$ for $i \neq j$, and so

$$S = \mathrm{AVar}(\bar{g}) = E(g_i g'_i) = E(u_i^2 Z'_i Z_i)$$

This matrix can be consistently estimated by an Eicker–Huber–White robust covariance estimator

$$\widehat{S} = \frac{1}{n}\sum_{i=1}^{n} \widehat{u}_i^2 Z_i' Z_i = \frac{1}{n}(Z'\widehat{\Omega}Z) \tag{7}$$

where $\widehat{\Omega}$ is the diagonal matrix of squared residuals $\widehat{u}_i^2$ from $\widetilde{\beta}$, the consistent but not necessarily efficient first-step GMM estimator. In the `ivreg2` implementation of two-step EGMM, the first-step estimator is $\widehat{\beta}_{\rm IV}$, the IV estimator.

The resulting estimate $\widehat{S}$ can be used to conduct consistent inference for the first-step estimator using (3), or it can be used to obtain and conduct inference for the EGMM estimator using (5) and (6).

In the next section, we discuss how the two-step GMM estimator can be applied when the errors are serially correlated.

2.5 Using ivreg2 for GMM estimation

The `ivreg2` command is included in the electronic supplement to this issue. The latest version of `ivreg2` can also be downloaded from the SSC archive with the command `ssc describe ivreg2`. We summarize the command's syntax and options in sections 11 and 12, respectively. The commands below illustrate how to use `ivreg2` to obtain the coefficient and variance–covariance estimators discussed above. The example uses the dataset provided in Wooldridge (2003).

The first command requests a standard IV/2SLS estimator and a variance–covariance matrix, which assumes conditionally homoskedastic and independent errors. In this case, IV/2SLS is the EGMM estimator. The second requests the IV/2SLS estimator and a variance–covariance estimator that is robust to heteroskedasticity based on an estimate of $\widehat{S}$ as in (7); here IV/2SLS is an IGMM estimator. The third command requests the two-step feasible EGMM estimator and corresponding variance–covariance matrix. $\widehat{S}$ is again based on (7). The fourth command is equivalent to the first, illustrating that the two-step EGMM estimator reduces to two-stage least squares when the disturbance is assumed to be independently and identically distributed (i.i.d.) and S can be consistently estimated by a classical nonrobust covariance matrix estimator.

1. `ivreg2 lwage exper expersq (educ=age kidslt6 kidsge6)`
2. `ivreg2 lwage exper expersq (educ=age kidslt6 kidsge6), robust`
3. `ivreg2 lwage exper expersq (educ=age kidslt6 kidsge6), gmm2s robust`
4. `ivreg2 lwage exper expersq (educ=age kidslt6 kidsge6), gmm2s`

3 GMM and HAC standard errors

Equation (7) illustrates how the asymptotic covariance matrix of the GMM estimator could be derived in the presence of conditional heteroskedasticity. We now further extend the estimator to handle the case of nonindependent errors in a time-series context. We correspondingly change our notation so that observations are indexed by t and s rather than i. In the presence of serial correlation, $E(g_t g_s') \neq 0, t \neq s$. To derive consistent estimates of S, we define $\Gamma_j = E(g_t g_{t-j}')$ as the autocovariance matrix for lag j. We may then write the long-run covariance matrix

$$S = \text{AVar}(\bar{g}) = \Gamma_0 + \sum_{j=1}^{\infty} (\Gamma_j + \Gamma_j') \tag{8}$$

which may be seen as a generalization of (7), with $\Gamma_0 = E(g_i g_i')$ and

$$\Gamma_j = E(g_t g_{t-j}'),\ j = \pm 1, \pm 2, \ldots$$

As g_t is defined as the product of Z_t and u_t, the autocovariance matrices may be expressed as

$$\Gamma_j = E(u_t u_{t-j} Z_t' Z_{t-j})$$

As usual, we replace the u_t, u_{t-j} by consistent residuals from first-stage estimation to compute the sample autocovariance matrices $\widehat{\Gamma}_j$, defined as

$$\widehat{\Gamma}_j = \frac{1}{n} \sum_{t=1}^{n-j} \widehat{g}_t \widehat{g}_{t-j} = \frac{1}{n} \sum_{t=1}^{n-j} Z_t' \widehat{u}_t \widehat{u}_{t-j} Z_{t-j}$$

We obviously do not have an infinite number of sample autocovariances to insert into the infinite sum in (8). Less obviously, we also cannot simply insert all the autocovariances from 1 to n, because this would imply that the number of sample orthogonality conditions $\widehat{g}_i$ is going to infinity with the sample size, which precludes obtaining a consistent estimate of S.[2] The autocovariances must converge to zero asymptotically as n increases.

The usual way this is handled in practice is for the summation to be truncated at a specified lag q. Thus the S matrix can be estimated by

$$\widehat{S} = \widehat{\Gamma}_0 + \sum_{j=1}^{q} \kappa \left(\frac{j}{q_n} \right) (\widehat{\Gamma}_j + \widehat{\Gamma}_j')$$

2. Although a consistent estimate cannot be obtained with bandwidth equal to sample size, Hall (2005, 305–310) points out that it is possible to develop an asymptotic framework providing inference about the parameters.

where u_t, u_{t-j} are replaced by consistent estimates from first-stage estimation. The *kernel* function, $\kappa(j/q_n)$, applies appropriate weights to the terms of the summation, with q_n defined as the *bandwidth* of the kernel (possibly as a function of n).[3] In many kernels, consistency is obtained by having the weight fall to zero after a certain number of lags.

The best-known approach to this problem in econometrics is that of Newey and West (1987b), which generates $\widehat{S}$ by using the Bartlett kernel function and a user-specified value of q. For the Bartlett kernel, $\kappa(\cdot) = (1 - j/q_n)$ if $j \leq q_n - 1$, 0 otherwise. These estimates are said to be heteroskedasticity- and autocorrelation-consistent (HAC), as they incorporate the standard sandwich formula (7) in computing Γ_0.

HAC estimates can be calculated by using `ivreg2` with the `robust` and `bw()` options with the kernel function's bandwidth (the `bw()` option) set to q.[4] The bandwidth may also be chosen optimally by specifying `bw(auto)` by using the automatic bandwidth selection criterion of Newey and West (1994).[5,6] By default, `ivreg2` uses the Bartlett kernel function.[7] If the equation contains endogenous regressors, these options will cause the IV estimates to be HAC. If the equation is overidentified and the `robust`, `gmm2s`, and `bw()` options are specified, the resulting GMM estimates will be both HAC and more efficient than those produced by IV.

The Newey–West (Bartlett kernel function) specification is only one of many feasible HAC estimators of the covariance matrix. Andrews (1991) shows that in the class of positive semidefinite kernels, the rate of convergence of $\widehat{S} \rightarrow S$ depends on the choice of kernel and bandwidth. The Bartlett kernel's performance is bettered by those in a subset of this class, including the quadratic spectral kernel. Accordingly, `ivreg2` provides a menu of kernel choices, including (abbreviations in parentheses): quadratic spectral (`qua` or `qs`), truncated (`tru`), Parzen (`par`), Tukey–Hanning (`thann`), Tukey–Hamming (`thamm`), Daniell (`dan`), and Tent (`ten`). For the Bartlett, Parzen, and Tukey–Hanning/Hamming kernels, the number of lags used to construct the kernel estimate equals the bandwidth (`bw()`) minus one.[8] If the kernels above are used with `bw(1)`, no lags are used and `ivreg2` will report the usual Eicker–Huber–White "sandwich" heteroskedasticity-robust variance estimates. Most, but not all, of these kernels guarantee that the estimated $\widehat{S}$ is positive definite and therefore always invertible; the truncated kernel, for example, was proposed in the early literature in this area but is now rarely used because it can generate an noninvertible $\widehat{S}$. For a survey covering various kernel estimators and their properties, see Cushing and McGarvey (1999) and Hall (2005, 75–86).

3. For more detail on this GMM estimator, see Hayashi (2000, 406–417).
4. For the special case of ordinary least squares (OLS), Newey–West standard errors are available from [TS] **newey** with the maximum lag $(q-1)$ specified by `newey`'s `lag()` option.
5. This implementation is identical to that provided by Stata's `ivregress` command; see [R] **ivregress**.
6. Automatic bandwidth selection is only available for the Bartlett, Parzen, and quadratic spectral kernels; see below.
7. A common choice of bandwidth for the Bartlett kernel function is $T^{1/3}$.
8. A common choice of bandwidth for these kernels is $(q-1) \approx T^{1/4}$ (Greene 2008, 643). A value related to the periodicity of the data (4 for quarterly, 12 for monthly, etc.) is often chosen.

Under conditional homoskedasticity the expression for the autocovariance matrix simplifies:

$$\Gamma_j \quad = \quad E(u_t u_{t-j} Z_t' Z_{t-j}) = E(u_t u_{t-j}) E(Z_t' Z_{t-j})$$

and the calculations of the corresponding kernel estimators also simplify; see Hayashi (2000, 413–414). These estimators may perform better than their heteroskedasticity-robust counterparts in finite samples. If the researcher is satisfied with the assumption of homoskedasticity but wants to deal with autocorrelation of unknown form, the researcher should use the AC correction without the H correction for arbitrary heteroskedasticity by omitting the **robust** option. **ivreg2** allows selection of H, AC, or HAC VCEs by combining the **robust**, **bw()**, and **kernel()** options. Thus both **robust** and **bw()** must be specified to calculate a HAC VCE of the Newey–West type, using the default Bartlett kernel. [9]

To illustrate the use of HAC standard errors, we fit a quarterly time-series model relating the change in the U.S. inflation rate (**D.inf**) to the unemployment rate (**UR**) for 1960q3–1999q4. As instruments, we use the second lag of quarterly GDP growth and the lagged values of the Treasury bill rate, the trade-weighted exchange rate, and the Treasury medium-term bond rate.[10] We first estimate the equation with standard IV under the assumption of i.i.d. errors.

```
. use http://fmwww.bc.edu/ec-p/data/stockwatson/macrodat
. generate inf = 100 * log( CPI / L4.CPI )
(4 missing values generated)
. generate ggdp = 100 * log( GDP / L4.GDP )
(10 missing values generated)
```

9. Stata's official `newey` command (see [TS] **newey**) does not allow gaps in time-series data. As there is no difficulty in computing HAC estimates with gaps in a regularly spaced time series, `ivreg2` handles this case properly.
10. These data accompany Stock and Watson (2003).

```
. ivreg2 D.inf (UR=L2.ggdp L.TBILL L.ER L.TBON)
IV (2SLS) estimation
--------------------

Estimates efficient for homoskedasticity only
Statistics consistent for homoskedasticity only
                                                      Number of obs =      158
                                                      F(  1,   156) =    10.16
                                                      Prob > F      =   0.0017
Total (centered) SS     =  60.04747699                Centered R2   =   0.1914
Total (uncentered) SS   =  60.05149156                Uncentered R2 =   0.1915
Residual SS             =  48.55290564                Root MSE      =    .5543

------------------------------------------------------------------------------
       D.inf |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          UR |   -.155009   .0483252    -3.21   0.001    -.2497246   -.0602933
       _cons |   .9380705   .2942031     3.19   0.001      .361443    1.514698
------------------------------------------------------------------------------
Underidentification test (Anderson canon. corr. LM statistic):          58.656
                                                   Chi-sq(4) P-val =    0.0000
------------------------------------------------------------------------------
Weak identification test (Cragg-Donald Wald F statistic):               22.584
Stock-Yogo weak ID test critical values:  5% maximal IV relative bias    16.85
                                         10% maximal IV relative bias    10.27
                                         20% maximal IV relative bias     6.71
                                         30% maximal IV relative bias     5.34
                                         10% maximal IV size             24.58
                                         15% maximal IV size             13.96
                                         20% maximal IV size             10.26
                                         25% maximal IV size              8.31
Source: Stock-Yogo (2005).  Reproduced by permission.
------------------------------------------------------------------------------
Sargan statistic (overidentification test of all instruments):           5.851
                                                   Chi-sq(3) P-val =    0.1191
------------------------------------------------------------------------------
Instrumented:         UR
Excluded instruments: L2.ggdp L.TBILL L.ER L.TBON
------------------------------------------------------------------------------
```

In these estimates, the negative coefficient on the unemployment rate is consistent with macroeconomic theories of the natural rate. In that context, lowering unemployment below the natural rate will cause an acceleration of price inflation. The Sargan statistic implies that the test of overidentifying restrictions cannot reject its null hypothesis.

An absence of autocorrelation in the error process is unusual in time-series analysis, so we test the equation by using `ivactest`, as discussed in section 10. By using the default value of one lag, we consider whether the error process exhibits AR(1) behavior. The test statistic implies that the errors do not exhibit serial independence:

(*Continued on next page*)

```
. ivactest

Cumby-Huizinga test with H0: errors nonautocorrelated at order 1
Test statistic:  25.909524
Under H0, Chi-sq(1) with p-value:  3.578e-07
```

Given this strong rejection of the null of independence, we reestimate the equation with HAC standard errors, choosing a bandwidth (`bw()`) of 5 (roughly $T^{1/3}$) and the `robust` option. By default, the Bartlett kernel is used, so that these are Newey–West two-step EGMM estimates.

```
. ivreg2 D.inf (UR=L2.ggdp L.TBILL L.ER L.TBON), gmm2s robust bw(5)
2-Step GMM estimation
---------------------

Estimates efficient for arbitrary heteroskedasticity and autocorrelation
Statistics robust to heteroskedasticity and autocorrelation
  kernel=Bartlett; bandwidth=5
  time variable (t):  date
                                                      Number of obs =      158
                                                      F(  1,   156) =     2.46
                                                      Prob > F      =   0.1185
Total (centered) SS     =  60.04747699                Centered R2   =   0.1548
Total (uncentered) SS   =  60.05149156                Uncentered R2 =   0.1548
Residual SS             =  50.75430293                Root MSE      =    .5668

------------------------------------------------------------------------------
             |               Robust
       D.inf |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          UR |  -.1002374   .0634562    -1.58   0.114    -.2246092    .0241344
       _cons |   .5850796    .372403     1.57   0.116     -.144817    1.314976
------------------------------------------------------------------------------
Underidentification test (Kleibergen-Paap rk LM statistic):              7.954
                                                   Chi-sq(4) P-val =    0.0933
------------------------------------------------------------------------------
Weak identification test (Kleibergen-Paap rk Wald F statistic):          7.362
Stock-Yogo weak ID test critical values:  5% maximal IV relative bias    16.85
                                         10% maximal IV relative bias    10.27
                                         20% maximal IV relative bias     6.71
                                         30% maximal IV relative bias     5.34
                                         10% maximal IV size             24.58
                                         15% maximal IV size             13.96
                                         20% maximal IV size             10.26
                                         25% maximal IV size              8.31
Source: Stock-Yogo (2005).  Reproduced by permission.
NB: Critical values are for Cragg-Donald F statistic and i.i.d. errors.
------------------------------------------------------------------------------
Hansen J statistic (overidentification test of all instruments):         3.569
                                                   Chi-sq(3) P-val =    0.3119
------------------------------------------------------------------------------
Instrumented:         UR
Excluded instruments: L2.ggdp L.TBILL L.ER L.TBON
------------------------------------------------------------------------------
```

It appears that by generating HAC estimates of the covariance matrix, the statistical significance of the unemployment rate in this equation is now questioned. One important statistic is also altered: the test for overidentification, denoted as the Sargan test in the former estimates, is on the borderline of rejecting its null hypothesis at the 90% level. When we reestimate the equation with HAC standard errors, various summary statistics are "robustified" as well: here the test of overidentifying restrictions, now denoted Hansen's J. That statistic is now far from rejection of its null, giving us greater confidence that our instrument set is appropriate.

4 CUE, LIML, and k-class estimation

4.1 CUE and LIML

Again consider the two-step feasible EGMM estimator. In the first step, a consistent but IGMM estimator, $\widetilde{\beta}$, is used to estimate S, the covariance matrix of orthogonality conditions. In the second step, the GMM objective function is maximized by using S^{-1} as the weighting matrix. If we write S as a function of the first-step estimator $\widetilde{\beta}$, the minimization problem in the second step of two-step EGMM estimation that defines the estimator is

$$\widehat{\beta}_{2S\text{EGMM}} \equiv \arg\min_{\widehat{\beta}} J(\widehat{\beta}) = n\overline{g}(\widehat{\beta})'\{S(\widetilde{\beta})\}^{-1}\overline{g}(\widehat{\beta})$$

As noted earlier, the second-step minimization treats the weighting matrix $W = \{S(\widetilde{\beta})\}^{-1}$ as a constant matrix. Thus the residuals in the estimate of S are the first-stage residuals defined by $\widetilde{\beta}$, whereas the residuals in the orthogonality conditions $\overline{g}$ are the second-stage residuals defined by $\widehat{\beta}$.

The minimization problem that defines the GMM/CUE of Hansen, Heaton, and Yaron (1996) is, by contrast,

$$\widehat{\beta}_{\text{CUE}} \equiv \arg\min_{\widehat{\beta}} J(\widehat{\beta}) = n\overline{g}(\widehat{\beta})'\{S(\widehat{\beta})\}^{-1}\overline{g}(\widehat{\beta})$$

Here the weighting matrix is a function of the β being estimated. The residuals in S are the same residuals that are in $\overline{g}$, and estimation of S is done simultaneously with the estimation of β. Generally, solving this minimization problem requires numerical methods.

Both the two-step EGMM and CUE/GMM procedures reduce to familiar estimators under linearity and conditional homoskedasticity. Here $S = E(g_i g_i') = E(u_i^2 Z_i' Z_i) = E(u_i^2)E(Z_i' Z_i) = \sigma^2 Q_{ZZ}$. Q_{ZZ} is estimated by its sample counterpart $(1/n)Z'Z$. In two-step EGMM under homoskedasticity, the minimization becomes

$$\widehat{\beta}_{\text{IV}} \equiv \arg\min_{\widehat{\beta}} J(\widehat{\beta}) = \frac{\widehat{u}(\widehat{\beta})' P_Z \widehat{u}(\widehat{\beta})}{\widehat{\sigma}^2} \tag{9}$$

where $\widehat{u}(\widehat{\beta}) \equiv (y - X\widehat{\beta})$ and $P_Z \equiv Z(Z'Z)^{-1}Z'$ is the projection matrix. In the minimization, the error variance $\widehat{\sigma}^2$ is treated as a constant and hence does not require first-step estimation, and the $\widehat{\beta}$ that solves (9) is the IV estimator $\beta_{\rm IV} = (X'P_ZX)^{-1}X'P_Zy$.[11]

With CUE/GMM under conditional homoskedasticity, the estimated error variance is a function of the residuals

$$\widehat{\sigma}^2 = \widehat{u}'(\widehat{\beta})\widehat{u}(\widehat{\beta})/n$$

and the minimization becomes

$$\widehat{\beta}_{\rm LIML} \equiv \arg\min_{\widehat{\beta}} J(\widehat{\beta}) = \frac{\widehat{u}(\widehat{\beta})'P_Z\widehat{u}(\widehat{\beta})}{\widehat{u}(\widehat{\beta})'\widehat{u}(\widehat{\beta})/n} \tag{10}$$

The $\widehat{\beta}$ that solves (10) is defined as the LIML estimator.

Unlike CUE estimators in general, the LIML estimator can be derived analytically and does not require numerical methods. This derivation is the solution to an eigenvalue problem (see Davidson and MacKinnon 1993, 644–649). The LIML estimator was first derived by Anderson and Rubin (1949), who also provided the first test of overidentifying restrictions for estimation of an equation with endogenous regressors. This Anderson–Rubin statistic (not to be confused with the test discussed below under "weak identification") follows naturally from the solution to the eigenvalue problem. If we denote the minimum eigenvalue by λ, the Anderson–Rubin likelihood-ratio test statistic for the validity of the overidentifying restrictions (orthogonality conditions) is $n\ \log(\lambda)$. Since LIML is also an EGMM estimator, the value J of the minimized GMM objective function also provides a test of overidentifying restrictions. The J test of the same overidentifying restrictions is closely related to the Anderson–Rubin test; the minimized value of the LIML GMM objective function is in fact $J = n(1/1 - \lambda)$. Of course, $n\ \log(\lambda) \approx n(1/1 - \lambda)$.

Although CUE and LIML provide no asymptotic efficiency gains over two-step GMM and IV, recent research suggests that their finite-sample performance may be superior. In particular, there is evidence suggesting that CUE and LIML perform better than IV/GMM in the presence of weak instruments (Hahn, Hausman, and Kuersteiner 2004). This is reflected, for example, in the critical values for the Stock–Yogo weak instruments test discussed in section 7.3.[12] The disadvantage of CUE in general is that it requires numerical optimization; LIML does not but does require the often rather strong assumption of i.i.d. disturbances. In `ivreg2`, the `cue` option combined with the `robust`, `cluster()`, and/or `bw()` options generates coefficient estimates that are efficient in the presence of the corresponding deviations from i.i.d. disturbances. Specifying `cue` with no other options is equivalent to the combination of the options `liml` and `coviv` ("covariance-IV"; see below).

11. The error variance $\widehat{\sigma}^2$, required for inference, is calculated at the end using the IV residuals.
12. With one endogenous regressor and four excluded instruments, the critical value for the Cragg–Donald statistic for 10% maximal size distortion is 24.58 in the case of IV but only 5.44 for LIML.

The implementation of the CUE estimator in `ivreg2` uses Stata's `ml` routine to minimize the objective function. The starting values are either IV or two-step EGMM coefficient estimates. These can be overridden with the `cueinit()` option, which takes a matrix of starting values of the coefficient vector β as its argument. The `cueoptions()` option passes its contents to Stata's `ml` command. Estimation with the `cue` option can be slow and problematic when the number of parameters to be estimated is substantial, and it should be used with caution.

4.2 k-class estimators

LIML, IV, and OLS (but not CUE or two-step GMM) are examples of k-class estimators. A k-class estimator can be written as (Davidson and MacKinnon 1993, 649)

$$\beta_k = \{X'(I - kM_Z)X\}^{-1}X'(I - kM_Z)y$$

where M denotes the annihilation matrix $I - P$. LIML is a k-class estimator with k=λ, the LIML eigenvalue; IV is a k-class estimator with k=1; and OLS is a k-class estimator with k=0. Estimators based on other values of k have been proposed. Fuller's modified LIML (available with the `fuller(#)` option) sets $k = \lambda - \{\alpha/(N - L)\}$, where λ is the LIML eigenvalue, L = number of instruments (included and excluded), and the Fuller parameter α is a user-specified positive constant. The value of $\alpha = 1$ has been suggested as a good choice; see Fuller (1977) or Davidson and MacKinnon (1993, 649–650). Nagar's bias-adjusted 2SLS estimator can be obtained with the `kclass(#)` option by setting $k = 1{+}(L - K)/N$, where $(L{-}K)$ is the number of overidentifying restrictions and N is the sample size; see Nagar (1959). Research suggests that both of these k-class estimators have a better finite-sample performance than IV in the presence of weak instruments, although neither estimator is robust to violations of the i.i.d. assumption. `ivreg2` also provides Stock–Yogo critical values for the Fuller version of LIML.

The default covariance matrix reported by `ivreg2` for the LIML and general k-class estimators is (Davidson and MacKinnon 1993, 650):

$$\widehat{\sigma}^2\{X'(I - kM_Z)X\}^{-1}$$

In fact, the usual IV-type covariance matrix

$$\widehat{\sigma}^2\{X'(I - M_Z)X\}^{-1} = \widehat{\sigma}^2(X'P_ZX)^{-1}$$

is also valid and can be obtained with the `coviv` option. With `coviv`, the covariance matrix for LIML and the other general k-class estimators will differ from that for the IV estimator only because the estimate of the error variance $\widehat{\sigma}^2$ will differ.

4.3 Example of CUE/LIML estimation

We illustrate the use of CUE/LIML estimation using the same equation we used in our discussion of HAC standard errors.

```
. ivreg2 D.inf (UR=L2.ggdp L.TBILL L.ER L.TBON ), cue robust bw(5)
initial:       neg GMM obj function -J =  -3.285175
rescale:       neg GMM obj function -J = -2.8716146
Iteration 0:   neg GMM obj function -J = -2.8716146
Iteration 1:   neg GMM obj function -J =  -2.793201
Iteration 2:   neg GMM obj function -J = -2.7931805
Iteration 3:   neg GMM obj function -J = -2.7931798
Iteration 4:   neg GMM obj function -J = -2.7931798
CUE estimation
──────────────

Estimates efficient for arbitrary heteroskedasticity and autocorrelation
Statistics robust to heteroskedasticity and autocorrelation
  kernel=Bartlett; bandwidth=5
  time variable (t):  date
                                                      Number of obs =      158
                                                      F(  1,   156) =     0.55
                                                      Prob > F      =   0.4577
Total (centered) SS     =  60.04747699                Centered R2   =   0.0901
Total (uncentered) SS   =  60.05149156                Uncentered R2 =   0.0901
Residual SS             =   54.6384785                Root MSE      =    .5881

──────────────────────────────────────────────────────────────────────────────
             │               Robust
       D.inf │      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
          UR │  -.0483119   .0644743    -0.75   0.454    -.1746792    .0780555
       _cons │   .2978451   .3804607     0.78   0.434    -.4478442    1.043534
──────────────────────────────────────────────────────────────────────────────
Underidentification test (Kleibergen-Paap rk LM statistic):              7.954
                                                   Chi-sq(4) P-val =    0.0933
──────────────────────────────────────────────────────────────────────────────
Weak identification test (Kleibergen-Paap rk Wald F statistic):          7.362
Stock-Yogo weak ID test critical values: 10% maximal LIML size           5.44
                                         15% maximal LIML size           3.87
                                         20% maximal LIML size           3.30
                                         25% maximal LIML size           2.98
Source: Stock-Yogo (2005).  Reproduced by permission.
NB: Critical values are for Cragg-Donald F statistic and i.i.d. errors.
──────────────────────────────────────────────────────────────────────────────
Hansen J statistic (overidentification test of all instruments):         2.793
                                                   Chi-sq(3) P-val =    0.4246
──────────────────────────────────────────────────────────────────────────────
Instrumented:         UR
Excluded instruments: L2.ggdp L.TBILL L.ER L.TBON
──────────────────────────────────────────────────────────────────────────────
```

When this estimator is used, the magnitude of the point estimate of the `UR` coefficient falls yet farther, and it is no longer significantly different from zero at any reasonable level of significance.

5 GMM distance tests of endogeneity and exogeneity

The value J of the GMM objective function evaluated at the EGMM estimator $\widehat{\beta}_{\text{EGMM}}$ is distributed as χ^2 with $(L-K)$ degrees of freedom under the null hypothesis that the full set of orthogonality conditions are valid. This is variously known as the *Sargan statistic*, *Hansen J statistic*, *Sargan–Hansen J test*, or simply a test of overidentifying restrictions.[13]

A C or GMM distance test can be used to test the validity of a subset of orthogonality conditions. Say that the investigator wants to test the validity of L_B orthogonality conditions. Denote J as the value of the GMM objective function for the EGMM estimator that uses the full set of orthogonality conditions and J_A as the value of the EGMM estimator that uses only the $L_A = L - L_B$ orthogonality conditions that the investigator is not questioning. Then under the null that the L_B suspect orthogonality conditions are actually satisfied, the test statistic $(J - J_A) \sim \chi^2$ with L_B degrees of freedom. If the $\widehat{S}$ matrix from the estimation using the full set of orthogonality conditions is used to calculate both GMM estimators, the test statistic is guaranteed to be nonnegative in finite samples.

Baum, Schaffer, and Stillman (2003) discuss how ivreg2's orthog() option can be used to conduct a C test of the exogeneity of one or more regressors or instruments. To recapitulate, the orthog() option takes as its argument the list of exogenous variables Z_B whose exogeneity is called into question. If the exogenous variable being tested is an instrument, the EGMM estimator that does not use the corresponding orthogonality condition simply drops the instrument. This is illustrated in the following pair of estimations where the second regression is the estimation implied by the orthog() option in the first:

```
. ivreg2 y x1 x2 (x3 = z1 z2 z3 z4), orthog(z4)
. ivreg2 y x1 x2 (x3 = z1 z2 z3)
```

If the exogenous variable that is being tested is a regressor, the efficient GMM estimator that does not use the corresponding orthogonality condition treats the regressor as endogenous, as below; again, the second estimation is implied by the use of orthog() in the former equation:

```
. ivreg2 y x1 x2 (x3 = z1 z2 z3 z4), orthog(x2)
. ivreg2 y x1 (x2 x3 = z1 z2 z3)
```

Sometimes the researcher wants to test whether an endogenous regressor can be treated as exogenous. This is commonly termed an "endogeneity test", but as we discussed in our earlier paper (Baum, Schaffer, and Stillman 2003, 24–27), it is equivalent to estimating the same regression but treating the regressor as exogenous, and then testing the corresponding orthogonality condition using the orthog() option. Although the

13. If the test statistic is required for an IGMM estimator (e.g., an overidentifying restrictions test for the IV estimator that is robust to heteroskedasticity), ivreg2 reports the J statistic for the corresponding EGMM estimator; see Baum, Schaffer, and Stillman (2003). This J statistic is identical to that produced by estat overid following official Stata's ivregress gmm command.

procedure described there is appropriate, it is not intuitive. To address this, we have added a new `endogtest()` option to `ivreg2` to conduct endogeneity tests of one or more endogenous regressors. Under the null hypothesis that the specified endogenous regressors can actually be treated as exogenous, the test statistic is distributed as χ^2 with degrees of freedom equal to the number of regressors tested. Thus, in the following estimation,

```
. ivreg2 y x1 x2 (x3 = z1 z2 z3 z4), endogtest(x3)
```

the test statistic reported for the endogeneity of `x3` is numerically equal to the test statistic reported for the `orthog()` option in

```
. ivreg2 y x1 x2 x3 ( = z1 z2 z3 z4), orthog(x3)
```

The `endogtest()` option is both easier to understand and more convenient to use.

Under conditional homoskedasticity, this endogeneity test statistic is numerically equal to a Hausman test statistic; see Hayashi (2000, 233–234) and Baum, Schaffer, and Stillman (2003, 19–22). The endogeneity test statistic can also be calculated after `ivregress` or `ivreg2` by the command `ivendog`. Unlike the Durbin–Wu–Hausman versions of the endogeneity test reported by `ivendog`, the `endogtest()` option of `ivreg2` can report test statistics that are robust to various violations of conditional homoskedasticity. The `ivendog` option unavailable in `ivreg2` is the Wu–Hausman F-test version of the endogeneity test.

To illustrate this option, we use a dataset provided in Wooldridge (2003). We estimate the log of females' wages as a function of the worker's experience, experience-squared, and years of education. If the education variable is considered endogenous, it is instrumented with the worker's age and counts of the number of preschool children and older children in the household. We test whether the `educ` variable need be considered endogenous in this equation with the `endogtest()` option:

```
. use http://fmwww.bc.edu/ec-p/data/wooldridge/mroz.dta
. ivreg2 lwage exper expersq (educ=age kidslt6 kidsge6), endogtest(educ)
IV (2SLS) estimation
--------------------

Estimates efficient for homoskedasticity only
Statistics consistent for homoskedasticity only
                                                      Number of obs =      428
                                                      F(  3,   424) =     7.49
                                                      Prob > F      =   0.0001
Total (centered) SS     =  223.3274513                Centered R2   =   0.1556
Total (uncentered) SS   =   829.594813                Uncentered R2 =   0.7727
Residual SS             =  188.5780571                Root MSE      =    .6638

------------------------------------------------------------------------------
       lwage |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        educ |   .0964002   .0814278     1.18   0.236    -.0631952    .2559957
       exper |    .042193   .0138831     3.04   0.002     .0149827    .0694033
     expersq |  -.0008323   .0004204    -1.98   0.048    -.0016563   -8.33e-06
       _cons |  -.3848718   1.011551    -0.38   0.704    -2.367476    1.597732
------------------------------------------------------------------------------
Underidentification test (Anderson canon. corr. LM statistic):          12.816
                                                   Chi-sq(3) P-val =    0.0051
------------------------------------------------------------------------------
Weak identification test (Cragg-Donald Wald F statistic):                4.342
Stock-Yogo weak ID test critical values:  5% maximal IV relative bias    13.91
                                         10% maximal IV relative bias     9.08
                                         20% maximal IV relative bias     6.46
                                         30% maximal IV relative bias     5.39
                                         10% maximal IV size             22.30
                                         15% maximal IV size             12.83
                                         20% maximal IV size              9.54
                                         25% maximal IV size              7.80
Source: Stock-Yogo (2005).  Reproduced by permission.
------------------------------------------------------------------------------
Sargan statistic (overidentification test of all instruments):           0.702
                                                   Chi-sq(2) P-val =    0.7042
-endog- option:
Endogeneity test of endogenous regressors:                               0.019
                                                   Chi-sq(1) P-val =    0.8899
Regressors tested:    educ
------------------------------------------------------------------------------
Instrumented:         educ
Included instruments: exper expersq
Excluded instruments: age kidslt6 kidsge6
------------------------------------------------------------------------------
```

In this context, we estimate the equation treating `educ` as endogenous, and merely name it in the `endogtest()` varlist to perform the C (GMM distance) test. The test cannot reject its null that `educ` may be treated as exogenous. In contrast, we may calculate this same test statistic with the earlier `orthog()` option:

```
. ivreg2 lwage exper expersq educ (=age kidslt6 kidsge6), orthog(educ)
```

By using `orthog()`, we again list `educ` in the option's varlist, but we must estimate the equation with that variable treated as exogenous: an equivalent but perhaps a less intuitive way to perform the test.

6 The FWL theorem and a rank-deficient S matrix

According to the Frisch–Waugh–Lovell (FWL) theorem (Frisch and Waugh 1933, Lovell 1963), the coefficients estimated for a regression in which some exogenous regressors, say, X_{2A}, are partialled out from the dependent variable y; the endogenous regressors X_1; the other exogenous regressors X_{2B}; and the excluded instruments Z_1 will be the same as the coefficients estimated for the original model for certain estimators. That is, if we denote a partialled-out variable with a tilde so that $\widetilde{y} \equiv M_{2A}y$, the coefficients estimated for the partialled-out version of the model

$$\widetilde{y} = [\widetilde{X}_1 \;\; \widetilde{X}_{2B}][\beta_1' \;\; \beta_{2B}']' + \widetilde{u}$$

with instruments $\widetilde{Z}_1$ and $\widetilde{X}_{2B}$ will be the same as the shared coefficients fitted for the original model

$$y = [X_1 \;\; X_2][\beta_1' \;\; \beta_2']' + u$$

with instruments Z_1 and X_2. It is even possible to partial-out the full set of included exogenous variables X_2, so that the partialled-out version of the model becomes

$$\widetilde{y} = \widetilde{X}_1\beta_1 + \widetilde{u}$$

with no exogenous regressors and only excluded instruments $\widetilde{Z}_1$, and the estimated $\widehat{\beta}_1$ will be the same as that obtained when estimating the full set of regressors.

The FWL theorem is implemented in `ivreg2` by the new `partial(`*varlist*`)` option, which requests that the exogenous regressors in the *varlist* should be partialled out from all the other variables (other regressors and excluded instruments) in the estimation. If the equation includes a constant, it is automatically partialled out as well.

The `partial()` option is most useful when the covariance matrix of orthogonality conditions S is not of full rank. When this is the case, EGMM and overidentification tests are infeasible as the optimal GMM weighting matrix $W = S^{-1}$ cannot be calculated. Sometimes partialling-out enough exogenous regressors can make the covariance matrix of the remaining orthogonality conditions full rank, and EGMM becomes feasible.

The invariance of the estimation results to partialling-out applies to one- and two-step estimators such as OLS, IV, LIML, and two-step GMM, but not to CUE or to GMM iterated more than two steps. The reason is that the latter estimators update the estimated S matrix. An updated S implies different estimates of the coefficients on the partialled-out variables, which imply different residuals, which in turn produce a different estimated S. Intuitively, partialling-out uses OLS estimates of the coefficients on the partialled-out variables to generate the S matrix, whereas CUE would use more efficient heteroskedastic OLS (HOLS) estimates.[14] Partialling-out exogenous regressors that are not of interest may still be desirable with CUE estimation, however, because reducing the number of parameters estimated makes the CUE numerical optimization faster and more reliable.

14. We are grateful to Manuel Arellano for helpful discussions on this point. For information on HOLS, see Baum, Schaffer, and Stillman (2003).

One common case calling for partialling-out arises when using `cluster()` and the number of clusters is less than L, the number of (exogenous regressors + excluded instruments). This causes the matrix S to be rank deficient (Baum, Schaffer, and Stillman 2003, 9–10). The problem can be addressed by using `partial()` to remove enough exogenous regressors for S to have full rank. A similar problem arises if a robust covariance matrix is requested when the regressors include a variable that is a singleton dummy, i.e., a variable with one value of 1 and $(N-1)$ values of zero or vice versa. The singleton dummy causes the robust covariance matrix estimator to be less than full rank. Here partialling-out the variable with the singleton dummy solves the problem.

The `partial()` option has two limitations: it cannot be used with time-series operators, and postestimation command `predict` can be used only to generate residuals.

7 Underidentification, weak identification, and instrument relevance

7.1 Identification and the rank condition

For (1) to be estimable, it must be *identified*. The order condition $L \geq K$ is necessary but not sufficient; the rank condition must also be satisfied. The rank condition states that the matrix $Q_{XZ} \equiv E(X_i' Z_i)$ is of full column rank, i.e., Q_{XZ} must have rank K. Since $X_2 \equiv Z_2$, we can simplify by partialling them out from X_1 and Z_1, and the rank condition becomes $\rho(Q_{\widetilde{X}_1 \widetilde{Z}_1}) = K_1$. There are several ways of interpreting this condition.

One interpretation is in terms of *correlations*: the excluded instruments must be correlated with the endogenous regressors. In the simplest possible case of an endogenous regressor, an excluded instrument, and partialling-out any exogenous regressors including the constant, $L_1 = K_1 = 1$ and $Q_{\widetilde{X}_1 \widetilde{Z}_1}$ is a scalar. As the constant has been partialled-out, $E(X_i) = E(Z_i) = 0$ and $Q_{\widetilde{X}_1 \widetilde{Z}_1}$ is a covariance. The rank condition in this simple case requires that the correlation or covariance between $\widetilde{X}_1$ and $\widetilde{Z}_1$ is nonzero.

This interpretation can be extended to the general case of $L_1, K_1 \geq 1$ by using *canonical correlations* (Anderson 1984, ch. 12; Hall, Rudebusch, and Wilcox 1996, 287; [MV] **canon**). The canonical correlations r_i between $\widetilde{X}_1$ and $\widetilde{Z}_1$, $i = 1, \ldots, K_1$ represent the correlations between linear combinations of the K_1 columns of $\widetilde{X}_1$ and linear combinations of the L_1 columns of $\widetilde{Z}_1$.[15] In the special case of $L_1 = K_1 = 1$ (and partialling-out the constant), the canonical correlation between $\widetilde{X}_1$ and $\widetilde{Z}_1$ is the usual Pearson correlation coefficient. In the slightly more general case of $L_1 \geq 1$ and $K_1 = 1$, the canonical correlation between $\widetilde{X}_1$ and $\widetilde{Z}_1$ is simply R: the square root of R^2 in a regression of $\widetilde{X}$ on $\widetilde{Z}$. In the general case of $L_1, K_1 \geq 1$, the squared canonical corre-

15. As $X_2 \equiv Z_2$, these variables are perfectly correlated with each other. The canonical correlations between X and Z before partialling out would also include the $L_2 \equiv K_2$ correlations that are equal to unity.

lations may be calculated as the eigenvalues of $(\widetilde{X}_1'\widetilde{X}_1)^{-1}(\widetilde{X}_1'\widetilde{Z}_1)(\widetilde{Z}_1'\widetilde{Z}_1)^{-1}(\widetilde{Z}_1'\widetilde{X}_1)$. The rank condition can then be interpreted as the requirement that all K_1 of the canonical correlations must be significantly different from zero. If one or more of the canonical correlations is zero, the model is *underidentified* or *unidentified*.

An alternative and useful interpretation of the rank condition is to use the reduced form. Write the set of reduced-form (first stage) equations for the regressors X as

$$X = Z\Pi + v$$

By using our partitioning of X and Z, we can rewrite this as

$$X_1 = [Z_1 \;\; Z_2]\,[\Pi_{11}' \;\; \Pi_{12}']' + v_1 \tag{11}$$

$$X_2 = [Z_1 \;\; Z_2]\,[\Pi_{21}' \;\; \Pi_{22}']' + v_2$$

The equation for X_2 is not interesting because $X_2 \equiv Z_2$ follows that $\Pi_{21} = 0$ and $\Pi_{22} = I$. The rank condition for identification comes from the equation for the endogenous regressors X_1. The $L \times K_1$ matrix Π_{11} must be of full column rank ($\rho(\Pi_{11}) = K_1$). If $\rho(\Pi_{11}) < K_1$, the model is again unidentified.

The consequence of utilizing excluded instruments that are uncorrelated with the endogenous regressors is increased bias in the estimated IV coefficients (Hahn and Hausman 2002) and worsening of the large-sample approximations to the finite-sample distributions. Here the bias of the IV estimator is the same as that of the OLS estimator and IV becomes inconsistent (ibid.). Here instrumenting only aggravates the problem, as IV and OLS share the same bias but IV has a larger mean squared error (MSE) by virtue of its larger variance. Serious problems also arise if the correlations between the excluded instruments and endogenous regressors are nonzero but "weak". Standard IV/GMM methods of estimating β_1 suffer from serious finite sample bias problems and alternative methods should be considered.

In the rest of this section, we show how to use `ivreg2` to conduct tests for underidentification and weak identification and how `ivreg2` provides a procedure for inference that is robust to weak identification.

7.2 Testing for underidentification and instrument redundancy

Of course, we do not observe the true Q_{XZ} or Π_{11} matrices; these matrices must be estimated. Testing whether or not the rank condition is satisfied therefore amounts to testing the rank of a matrix. Do the data enable the researcher to reject the null hypothesis that the equation is underidentified, i.e., that $\rho(\widehat{\Pi}_{11}) = (K_1 - 1)$, or, equivalently, $\rho(\widehat{Q}_{\widetilde{X}\widetilde{Z}}) = (K_1 - 1)$? Rejection of the null implies full rank and identification; failure to reject the null implies the matrix is rank-deficient and the equation is underidentified.

If the reduced-form errors v are i.i.d., two approaches are available for testing the rank of $Q_{\widetilde{X}\widetilde{Z}}$: Anderson's (1951) canonical correlations test and the related test of Cragg and Donald (1993). In Anderson's approach, $H_0\colon \rho(\widehat{Q}_{\widetilde{X}\widetilde{Z}}) = (K_1 - 1)$ is equivalent to the null hypothesis that the smallest canonical correlation r_{K_1} is zero. A large sample test statistic for this is simply $nr^2_{K_1}$. Under the null, the test statistic is distributed χ^2 with $(L - K + 1)$ degrees of freedom, so that it may be calculated even for an exactly identified equation. A failure to reject the null hypothesis suggests that the model is unidentified. Not surprisingly given its "$N \times R^2$" form this test can be interpreted as an LM test.[16]

The Cragg–Donald (1993) statistic is an alternative and closely related test for the rank of a matrix that can also be used to test for underidentification. Whereas the Anderson test is an LM test, the Cragg–Donald test is a Wald test, also derived from an eigenvalue problem. Poskitt and Skeels (2002) show that in fact the Cragg–Donald test statistic can be stated in terms of canonical correlations as $nr^2_{K_1}/(1 - r^2_{K_1})$ (see Poskitt and Skeels 2002, 17). It is also distributed as $\chi^2(L - K + 1)$.

Both these tests require the assumption of i.i.d. errors and hence are reported if `ivreg2` is invoked without the `robust`, `cluster()`, or `bw()` options. The Anderson LM χ^2 statistic is reported by `ivreg2` in the main regression output whereas both the Anderson LM and Cragg–Donald Wald χ^2 statistics are reported when the `first` option is specified.

If the errors are heteroskedastic or serially correlated, the Anderson and Cragg–Donald statistics are not valid. This is an important shortcoming, because these violations of the i.i.d. assumption would typically be expected to cause the null of underidentification to be rejected too often. Researchers would face the danger of interpreting a rejection of the null as evidence of a well-specified model that is adequately identified, when in fact it was both underidentified and misspecified.

Recently, several robust statistics for testing the rank of a matrix have been proposed. Kleibergen and Paap (2006) have proposed the rk statistic for this purpose. Their rk test statistic is reported by `ivreg2` if the user requests any sort of robust covariance estimator. The LM version of the Kleibergen–Paap rk statistic can be considered as a generalization of the Anderson canonical correlation rank statistic to the non-i.i.d. case. Similarly, the Wald version of the rk statistic reduces to the Cragg–Donald statistic when the errors are i.i.d. The rk test is implemented in Stata by the `ranktest` command of Kleibergen and Schaffer (2007), which `ivreg2` uses to calculate the rk statistic. If `ivreg2` is invoked with the `robust`, `bw()`, or `cluster()` options, the tests of underidentification reported by `ivreg2` are based on the rk statistic and will be correspondingly robust to heteroskedasticity, autocorrelation, or clustering. For a full discussion of the rk statistic, see Kleibergen and Paap (2006).

16. Earlier versions of `ivreg2` reported an LR version of this test, where the test statistic is $-n\log(1 - r^2_{K_1})$. This LR test has the same asymptotic distribution as the LM form. See Anderson (1984, 497–498).

In the special case of one endogenous regressor, the Anderson, Cragg–Donald, and Kleibergen–Paap statistics reduce to familiar statistics available from OLS estimation of the single reduced-form equation with an appropriate choice of VCE estimator. Thus the Cragg–Donald Wald statistic can be calculated by estimating (11) and testing the joint significance of the coefficients Π_{11} on the excluded instruments Z_1 by using a standard Wald test and a traditional nonrobust covariance estimator. The Anderson LM statistic can be obtained by calculating an LM test of the same joint hypothesis.[17] The Kleibergen–Paap rk statistics can be obtained by performing the same tests with the desired robust covariance estimator. For example, estimating (11) using OLS and testing the joint significance of Z_1 using a heteroskedasticity-robust covariance estimator yields the heteroskedastic-robust Kleibergen–Paap rk Wald statistic.[18]

The same framework may also be used to test a set of instruments for *redundancy* as shown by Breusch et al. (1999). In an overidentified context with $L \geq K$, if some of the instruments are redundant then the large-sample efficiency of the estimation is not improved by including them. It is well known, moreover, that using several instruments or moment conditions can cause the estimator to have poor finite-sample performance. Dropping redundant instruments may therefore lead to more reliable estimation.

The intuition behind a test for instrument redundancy is straightforward. As above, assume that we have partialled-out any exogenous regressors X_2. Partition the excluded instruments $\widetilde{Z}_1$ into $[\,\widetilde{Z}_{1A} \;\; \widetilde{Z}_{1B}\,]$, where $\widetilde{Z}_{1B}$ is the set of possibly redundant instruments after X_2 has been partialled-out. Breusch et al. (1999, 106) show that the redundancy of $\widetilde{Z}_{1B}$ can be stated in several ways: (a) $\text{plim}(1/n)\widetilde{Z}'_{1B}M_{\widetilde{Z}_{1A}}\widetilde{X}_1 = 0$; (b) the correlations between $\widetilde{Z}_{1B}$ and $\widetilde{X}_1$ (given $\widetilde{Z}_{1A}$) are zero; (c) in a regression of $\widetilde{X}_1$ on the full set of excluded instruments $\widetilde{Z}_1$, the coefficients on $\widetilde{Z}_{1B}$ are zero. It is easy to see that the FWL theorem can be used to restate this last condition without the partialling-out of X_2: (d) in a regression of X_1 on the full set of included and excluded instruments Z, i.e., the reduced form (11), the coefficients on Z_{1B} are zero. As Hall and Peixe (2003) point out, redundancy is a conditional concept. Z_{1B} either is or is not redundant conditional on Z_{1A}.

The above suggests a straightforward test of redundancy: simply estimate (11) using OLS and test the significance of Z_{1B} by using a large-sample LM, Wald, or LR test. For example, the redundancy test proposed by Hall and Peixe (2003) is the LR version of this test. These test statistics are all distributed as χ^2 with degrees of freedom equal to the number of endogenous regressors times the number of instruments tested. As usual, implementing this test is easy for the case of an endogenous variable, as only

17. This can be done simply in Stata using `ivreg2` by estimating (11) with only Z_2 as regressors, Z_1 as excluded instruments, and an empty list of endogenous regressors. The Sargan statistic reported by `ivreg2` will be the Anderson LM statistic. See Baum, Schaffer, and Stillman (2003) for further discussion.

18. See the online help for `ranktest` for examples. These test statistics are large-sample χ^2 tests and can be obtained from OLS regression using `ivreg2`. Stata's `regress` command reports finite-sample t tests. Also the robust rk LM statistic can be obtained as described in the preceding footnote. Invoke `ivreg2` with X_1 as the dependent variable, Z_2 as regressors, Z_1 as excluded instruments and no endogenous regressors. With the `robust` option, the reported Hansen J statistic is the robust rk statistic.

an OLS estimation is necessary. The tests of the coefficients can be made robust to various violations of i.i.d. errors in the usual way. However, this procedure is more laborious (though still straightforward) if $K_1 > 1$ as it is then necessary to jointly estimate multiple reduced-form equations.

Fortunately, a simpler procedure is available that will generate numerically equivalent test statistics for redundancy. Define a matrix $\breve{X}$ as X with both X_2 and Z_{1A} partialled-out. Then condition (a) can be restated as (e) $\text{plim}(1/n)\breve{Z}_{1B}'\breve{X}_1 = 0$ or (f) that the correlations between $\breve{Z}_{1B}$ and $\breve{X}_1$ (given Z_{1A} and Z_2) are zero. The redundancy of Z_{1B} can be evaluated by using the `ranktest` command to test the null hypothesis that the rank of $Q_{\breve{X}\breve{Z}}$ is zero. Rejection of the null indicates that the instruments are not redundant. The LM version of the Anderson canonical correlations test is reported if the user indicates that the errors are i.i.d. Here the LM test statistic is n times the sum of the squared canonical correlations between $\breve{Z}_{1B}$ and $\breve{X}_1$. If the user estimates the equation with `robust`, `bw()`, or `cluster()`, an LM version of the Kleibergen–Paap rk statistic is reported that is correspondingly robust to heteroskedasticity, autocorrelation, or clustering.

7.3 Testing for weak identification

The *weak-instruments* problem arises when the correlations between the endogenous regressors and the excluded instruments are nonzero but small. In the past 10–15 years, much attention in the econometrics literature has been devoted to this topic. What is surprising is that, as Bound, Jaeger, and Baker (1995), Staiger and Stock (1997), and others have shown, the weak-instruments problem can arise even when the correlations between X and Z are significant at conventional levels (5% or 1%) and the researcher is using a large sample. For more detailed discussion of the weak-instruments problem, see Staiger and Stock (1997), Stock, Wright, and Yogo (2002), or Dufour (2003). Thus rejecting the null of underidentification using the tests in the previous section and conventional significance levels is not enough; you must call for other methods.

One approach that has been advanced by Stock and Yogo (2005) is to test for the presence of weak instruments. The difference between this approach and the aforementioned underidentification tests is not in the basic statistic used, but in the finite-sample adjustments and critical values and in the null hypothesis being tested. Moreover, the critical values for a weak-instruments test are different for different estimators because the estimators are not affected to the same degree by weak instruments. Specifically, the LIML and CUE estimators are more robust to the presence of weak instruments than are IV and two-step GMM.

The test statistic proposed by Stock and Yogo (2005) is the F-statistic form of the Cragg and Donald (1993) statistic, $\{(N-L)/L_2\}\{r^2_{K_1}/(1-r^2_{K_1})\}$. `ivreg2` will report this statistic for an estimation that assumes i.i.d. disturbances. The null hypothesis being tested is that the estimator is weakly identified in the sense that it is subject to bias that the investigator finds unacceptably large. The Stock–Yogo weak-instruments tests come in two types: maximal relative bias and maximal size, where the null is that the instruments do not suffer from the specified bias. Rejection of their null hypothesis represents the absence of a weak-instruments problem. The first type is based on the ratio of the bias of the estimator to the bias of OLS. The null is that instruments are weak, where weak instruments are defined as instruments that can lead to an asymptotic relative bias greater than some value b. Because this test uses the finite-sample distribution of the IV estimator, it cannot be calculated in certain cases. This is because the m^{th} moment of the IV estimator exists if and only if $m < (L - K + 1)$.[19]

The second type of the Stock–Yogo tests is based on the performance of the Wald test statistic for β_1. Under weak identification, the Wald test rejects too often. The test statistic is based on the rejection rate r (10%, 20%, etc.) that the researcher is willing to tolerate if the true rejection rate should be the standard 5%. Weak instruments are defined as instruments that will lead to a rejection rate of r when the true rejection rate is 5%.

Stock and Yogo (2005) have tabulated critical values for their two weak-identification tests for the IV estimator, the LIML estimator, and Fuller's modified LIML estimator. The weak-instruments bias in the IV estimator is larger than that of the LIML estimators, and hence the critical values for the null that instruments are weak are also larger. The Stock–Yogo critical values are available for a range of possible circumstances (up to 3 endogenous regressors and 100 excluded instruments).

The weak-identification test that uses the Cragg–Donald F statistic, like the corresponding underidentification test, requires an assumption of i.i.d. errors. This is a potentially serious problem, for the same reason as given earlier: if the test statistic is large simply because the disturbances are not i.i.d., the researcher will commit a type-I error and incorrectly conclude that the model is adequately identified.

If the user specifies the `robust`, `cluster()`, or `bw()` options in `ivreg2`, the reported weak-instruments test statistic is a Wald F statistic based on the Kleibergen–Paap rk statistic. We are not aware of any studies on testing for weak instruments in the presence of non-i.i.d. errors. In our view, however, the use of the rk Wald statistic, as the robust analog of the Cragg–Donald statistic, is a sensible choice and clearly superior to the use of the latter in the presence of heteroskedasticity, autocorrelation, or clustering. We suggest, however, that when using the rk statistic to test for weak identification, users either apply with caution the critical values compiled by Stock and Yogo (2005) for the i.i.d. case or refer to the older "rule of thumb" of Staiger and Stock (1997), which says that the F statistic should be at least 10 for weak identification not to be considered a problem.

`ivreg2` will report in the main regression output the relevant Stock and Yogo (2005) critical values for IV, LIML, and Fuller-LIML estimates if they are available. The reported test statistic will be the Cragg–Donald statistic if the traditional covariance estimator is used or the rk statistic if a robust covariance estimator is requested. If the user requests two-step GMM estimation, `ivreg2` will report an rk statistic and the IV critical values. If the user requests the CUE estimator, `ivreg2` will report an rk statistic and

19. See Davidson and MacKinnon (1993, 221–222).

the LIML critical values. The justification for this is that IV and LIML are special cases of two-step GMM and CUE, respectively. The similarities carry over to weak instruments: the literature suggests that IV and two-step GMM are less robust to weak instruments than LIML and CUE. However, users of `ivreg2` may again wish to exercise some caution in applying the Stock–Yogo critical values in these cases.

7.4 Inference robust to weak identification: the Anderson–Rubin test

The first-stage `ivreg2` output also includes the Anderson and Rubin (1949) test of the significance of the endogenous regressors in the structural equation being estimated (not to be confused with the Anderson and Rubin (1949) overidentification test discussed earlier). In the form reported by `ivreg2`, the null hypothesis tested is that the coefficients β_1 of the endogenous regressors X_1 in the structural equation are jointly equal to zero. It is easily extended to testing the equality of the coefficients of X_1 to other values, but this is not supported explicitly by `ivreg2`; see the next section for further discussion.

The development of this Anderson and Rubin (1949) test is straightforward. Substitute the reduced-form expression (11) for the endogenous regressors X_1 into the main equation of the model

$$y = X\beta + u = X_1\beta_1 + Z_2\beta_2 + u = ([Z_1 \;\; Z_2]\,[\Pi_{11}' \;\; \Pi_{12}']' + v_1)\beta_1 + Z_2\beta_2 + u$$

and rearrange to obtain

$$y = Z_1\Pi_{11}\beta_1 + Z_2(\Pi_{12}\beta_1 + \beta_2) + (v_1\beta_1 + u)$$

Now consider estimating a reduced-form equation for y with the full set of instruments as regressors:

$$y = Z_1\gamma_1 + Z_2\gamma_2 + \eta$$

If the null $H_0: \beta_1 = 0$ is correct, $\Pi_{11}\beta_1 = 0$, and therefore $\gamma_1 = 0$. Thus the Anderson and Rubin (1949) test of the null $H_0: \beta_1 = 0$ is obtained by estimating the reduced form for y and testing that the coefficients γ_1 of the excluded instruments Z_1 are jointly equal to zero. If we fail to reject $\gamma_1 = 0$, then we also fail to reject $\beta_1 = 0$.

The Anderson–Rubin statistic is robust to the presence of weak instruments. As instruments become weak, the elements of Π_{11} become smaller and hence so does $\Pi_{11}\beta_1$: the null $H_0: \gamma_1 = 0$ is less likely to be rejected. That is, as instruments become weak, the power of the test declines, an intuitively appealing feature: weak instruments come at a price. `ivreg2` reports both the χ^2 version of the Anderson–Rubin statistic (distributed with L_1 degrees of freedom) and the F statistic version of the test. `ivreg2` also reports the closely related Stock and Wright (2000) S statistic. The S statistic tests the same null hypothesis as the Anderson–Rubin statistic and has the same distribution under the null. It is given by the value of the CUE objective function (with the exogenous regressors partialled out). Whereas the Anderson–Rubin statistic provides a Wald test, the S statistic provides an LM or GMM distance test of the same hypothesis.

More importantly, if the model is fitted with a robust covariance matrix estimator, both the Anderson–Rubin statistic and the S statistic reported by `ivreg2` are correspondingly robust. See Dufour (2003) and Chernozhukov and Hansen (2005) for further discussion of the Anderson–Rubin approach. For related alternative test statistics that are also robust to weak instruments (but not violations of the i.i.d. assumption), see the `condivreg` and `condtest` commands available from Moreira and Poi (2003) and Mikusheva and Poi (2006).

7.5 An example of estimation with weak instruments using ivreg2

We illustrate the weak-instruments problem with a variation on a log wage equation illustrated in Hayashi (2000). The explanatory variables are `s` (completed years of schooling), `expr` (years of work experience), `tenure` in the current job in years, `rns` (a dummy for residency in the southern U.S.), `smsa` (a dummy for urban workers), the worker's `iq` score, and a set of year dummies. Instruments include the worker's `age` and `mrt` (marital status: 1 = married) as instruments.

```
. use http://www.stata-press.com/data/imeus/griliches, clear
(Wages of Very Young Men, Zvi Griliches, J.Pol.Ec. 1976)

. ivreg2 lw s expr tenure rns smsa _I* (iq = age mrt), ffirst robust
> redundant(mrt)

Summary results for first-stage regressions
───────────────────────────────────────────

Variable    | Shea Partial R2 |   Partial R2    |  F(  2,   744)    P-value
iq          |     0.0073      |     0.0073      |        2.93       0.0539

NB: first-stage F-stat heteroskedasticity-robust

Underidentification tests
Ho: matrix of reduced form coefficients has rank=K1-1 (underidentified)
Ha: matrix has rank=K1 (identified)
Kleibergen-Paap rk LM statistic             Chi-sq(2)=5.90     P-val=0.0524
Kleibergen-Paap rk Wald statistic           Chi-sq(2)=5.98     P-val=0.0504

Weak identification test
Ho: equation is weakly identified
Kleibergen-Paap Wald rk F statistic                 2.93
See main output for Cragg-Donald weak id test critical values

Weak-instrument-robust inference
Tests of joint significance of endogenous regressors B1 in main equation
Ho: B1=0 and overidentifying restrictions are valid
Anderson-Rubin Wald test     F(2,744)= 46.95     P-val=0.0000
Anderson-Rubin Wald test     Chi-sq(2)=95.66     P-val=0.0000
Stock-Wright LM S statistic  Chi-sq(2)=69.37     P-val=0.0000

NB: Underidentification, weak identification and weak-identification-robust
    test statistics heteroskedasticity-robust

Number of observations              N  =        758
Number of regressors                K  =         13
Number of instruments               L  =         14
Number of excluded instruments      L1 =          2
```

```
IV (2SLS) estimation
--------------------

Estimates efficient for homoskedasticity only
Statistics robust to heteroskedasticity
                                                      Number of obs =      758
                                                      F( 12,   745) =     4.42
                                                      Prob > F      =   0.0000
Total (centered) SS     =  139.2861498                Centered R2   =  -6.4195
Total (uncentered) SS   =  24652.24662                Uncentered R2 =   0.9581
Residual SS             =  1033.432656                Root MSE      =    1.168

------------------------------------------------------------------------------
             |               Robust
          lw |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          iq |  -.0948902   .0418904    -2.27   0.024    -.1769939   -.0127865
           s |   .3397121   .1183267     2.87   0.004     .1077959    .5716282
        expr |   -.006604   .0292551    -0.23   0.821    -.0639429     .050735
      tenure |   .0848854   .0306682     2.77   0.006     .0247768     .144994
         rns |  -.3769393   .1559971    -2.42   0.016     -.682688   -.0711906
        smsa |   .2181191   .1031119     2.12   0.034     .0160236    .4202146
   _Iyear_67 |   .0077748   .1663252     0.05   0.963    -.3182166    .3337662
   _Iyear_68 |   .0377993   .1523585     0.25   0.804    -.2608179    .3364165
   _Iyear_69 |   .3347027   .1637992     2.04   0.041     .0136622    .6557432
   _Iyear_70 |   .6286425   .2468458     2.55   0.011     .1448336    1.112451
   _Iyear_71 |   .4446099   .1861877     2.39   0.017     .0796887     .809531
   _Iyear_73 |    .439027   .1668657     2.63   0.009     .1119763    .7660778
       _cons |   10.55096   2.781762     3.79   0.000     5.098812    16.00312
------------------------------------------------------------------------------
Underidentification test (Kleibergen-Paap rk LM statistic):              5.897
                                                   Chi-sq(2) P-val =    0.0524
-redundant- option:
IV redundancy test (LM test of redundancy of specified instruments):     0.002
                                                   Chi-sq(1) P-val =    0.9665
Instruments tested:   mrt
------------------------------------------------------------------------------
Weak identification test (Kleibergen-Paap rk Wald F statistic):          2.932
Stock-Yogo weak ID test critical values: 10% maximal IV size             19.93
                                         15% maximal IV size             11.59
                                         20% maximal IV size              8.75
                                         25% maximal IV size              7.25
Source: Stock-Yogo (2005).  Reproduced by permission.
NB: Critical values are for Cragg-Donald F statistic and i.i.d. errors.
------------------------------------------------------------------------------
Hansen J statistic (overidentification test of all instruments):         1.564
                                                   Chi-sq(1) P-val =    0.2111
------------------------------------------------------------------------------
Instrumented:         iq
Included instruments: s expr tenure rns smsa _Iyear_67 _Iyear_68 _Iyear_69
                      _Iyear_70 _Iyear_71 _Iyear_73
Excluded instruments: age mrt
------------------------------------------------------------------------------
```

In the first-stage regression results, the Kleibergen–Paap underidentification LM and Wald tests fail to reject their null hypotheses at the 95% level, suggesting that even for overidentification with the order condition, the instruments may be inadequate to identify the equation. The Anderson–Rubin Wald test and Stock–Wright LM test readily reject their null hypothesis and indicate that the endogenous regressors are relevant. However, given that those null hypotheses are joint tests of irrelevant regressors and appropriate overidentifying restrictions, the evidence is not so promising. In the main equation output, the `redundant(mrt)` option indicates that `mrt` provides no useful information to identify the equation. This equation may be exactly identified at best.

7.6 The relationship between inference robust to weak identification and overidentification tests

The Anderson–Rubin test that is robust to weak identification (and its related alternatives) relies heavily on the orthogonality of the excluded instruments Z_1. If the orthogonality conditions are violated, the Anderson–Rubin test will tend to reject the null $H_0: \beta_1 = 0$ even if the true $\beta_1 = 0$. The reason is easy to see: if Z_1 is correlated with the disturbance u, it will therefore also be correlated with the reduced-form error η, and so the estimated $\widehat{\gamma}_1$ will be biased away from zero even if in reality $\beta_1 = 0$.

Generally, in a test of overidentification, the maintained hypothesis is that the model is identified, so that a rejection means rejecting the orthogonality conditions. In the β_1 test that is robust to weak identification, the maintained hypothesis is that the instruments are valid, so that a rejection means rejecting the null that β_1 equals the hypothesized value.

This relationship between weak identification and overidentification tests can be stated precisely in the case of CUE or LIML estimation. We have been careful in the above to state that the two Anderson–Rubin tests should not be confused, but they are, in a sense, based on the same statistic. Assume that the exogenous regressors X_2, if any, have been partialled-out so that $\beta_1 \equiv \beta$. The value of the CUE/GMM objective function at $\widehat{\beta}_{CUE}$ provides a test of the orthogonality conditions; the LIML LR version of this test is the Anderson–Rubin overidentifying restrictions test. The value of the CUE/GMM objective function at some other, hypothesized $\widetilde{\beta}$ provides a test $H_0: \beta = \widetilde{\beta}$. This is the Stock and Wright (2000) S statistic, which is a Lagrange multiplier (LM) version of the Anderson–Rubin weak-instruments-robust test.

This can be illustrated using the Hayashi–Griliches example below. We assume conditional homoskedasticity and estimate using LIML. The Anderson–Rubin LR overidentification statistic (distributed with one degree of freedom) is small, as is the Sargan–Hansen J statistic, suggesting that the orthogonality conditions are valid:

```
. use http://www.stata-press.com/data/imeus/griliches, clear
(Wages of Very Young Men, Zvi Griliches, J.Pol.Ec. 1976)
. qui ivreg2 lw s expr tenure rns smsa _I* (iq = age mrt),
> partial(s expr tenure rns smsa _I*) liml
. di e(arubin)
1.1263807
. di e(j)
1.1255442
```

The Anderson–Rubin test of $H_0 : \beta_{IQ} = 0$ is calculated automatically by `ivreg2` with the `ffirst` option and is equivalent to estimating the reduced form for `lw` and testing the joint significance of the excluded instruments `age` and `mrt`:

```
. qui ivreg2 lw s expr tenure rns smsa _I* (iq = age mrt), liml ffirst
. di e(archi2)
89.313862
. qui ivreg2 lw s expr tenure rns smsa _I* age mrt
. test age mrt
 ( 1)  age = 0
 ( 2)  mrt = 0
           chi2(  2) =   89.31
         Prob > chi2 =    0.0000
```

The Stock–Wright S statistic is an LM or GMM distance test of the same hypothesis. This LM version of the Anderson–Rubin Wald test of `age` and `mrt` using the reduced-form estimation above is asymptotically equivalent to an LM test of the same hypothesis, available by using `ivreg2` and specifying these as excluded instruments (see Baum, Schaffer, and Stillman 2003 for further discussion). It is this LM version of the Anderson–Rubin weak-instruments-robust test that is numerically identical to the value of the GMM objective function at the hypothesized value $\beta_{IQ} = 0$:

```
. qui ivreg2 lw s expr tenure rns smsa _I* (=age mrt)
. di e(j)
79.899445
. mat b[1,1]=0
. qui ivreg2 lw s expr tenure rns smsa _I* (iq = age mrt),
> partial(s expr tenure rns smsa _I*) b0(b)
. di e(j)
79.899445
```

For $J(\beta_0)$ to be the appropriate test statistic, it is necessary for the exogenous regressors to be partialled out with the `partial()` option.

7.7 More first-stage options

To aid in the diagnosis of weak instruments, the `savefirst` option requests that the individual first-stage regressions be saved for later access by using the `estimates` command; see [R] **estimates**. If saved, they can also be displayed using `first` or `ffirst` and the `ivreg2` replay syntax. The regressions are saved with the prefix `_ivreg2_` unless the user specifies an alternative prefix with the `savefprefix(`*prefix*`)` option. The saved estimation results may be made the active set with `estimates restore`, allowing commands such as `test`, `lincom`, and `testparm` to be used.

The `rf` option requests that the reduced-form estimation of the equation be displayed. The `saverf` option requests that the reduced-form estimation is saved for later access by using the `estimates` command. If saved, it can also be displayed by using the `rf` and `ivreg2` replay syntax. The regression is saved with the prefix `_ivreg2_` unless the user specifies an alternative prefix with the `saverfprefix(`*prefix*`)` option.

8 Advanced ivreg2 options

Two options are available for speeding `ivreg2` execution. `nocollin` specifies that the collinearity checks not be performed. This option should be used with caution. `noid` suspends calculating and reporting of the underidentification and weak identification statistics in the main output.

The `b0(`*matrix*`)` option allows the user to specify that the GMM objective function, J, should be calculated for an arbitrary parameter vector. The parameter vector must be given as a matrix with appropriate row and column labels. The `b0()` option is most useful if the user wishes to conduct a weak-instruments-robust test of $H_0 : \beta_1 = b_0$, where b_0 is specified by the user. For example, in the illustration given in section 7.6, the null hypothesis that the coefficient on `iq` is 0.05 can be tested simply by replacing the line `mat b=J(1,1,0)` with `mat b=J(1,1,0.05)`. A heteroskedastic-robust S statistic can be obtained by specifying `robust` along with `b0(b)`. To construct a weak-instruments-robust confidence interval, the user can simply conduct a grid search over the relevant range for β_1.[20]

Two options have been added to `ivreg2` for special handling of the GMM estimation process. The `wmatrix(`*matrix*`)` option allows the user to specify a weighting matrix rather than computing the optimal weighting matrix. Estimation with the `wmatrix()` option yields a possibly inefficient GMM estimator. `ivreg2` will use this inefficient estimator as the first-step GMM estimator in two-step EGMM when combined with the `gmm2s` option; otherwise, `ivreg2` reports this IGMM estimator.

The `smatrix(`*matrix*`)` option allows the user to directly specify the matrix `S`, the covariance matrix of orthogonality conditions. `ivreg2` will use this matrix in the calculation of the variance–covariance matrix of the estimator, the J statistic, and if the `gmm2s` option is specified, the two-step EGMM coefficients. The `smatrix()` option can be useful for guaranteeing a positive test statistic in user-specified GMM-distance tests as described in section 5.

As Ahn (1997) shows, Hansen's J test has an LM interpretation but can also be calculated as the result of a Wald test. This is an application of the Newey and West (1987a) results on the equivalence of LM, Wald, and GMM distance tests. In the context of an overidentified model, the J statistic will be identical to a Wald χ^2 test statistic from an exactly identified model in which more instruments are included as regressors as long as the same estimate of S is used in both estimated equations. As an example:

```
. use http://www.stata-press.com/data/imeus/griliches, clear
(Wages of Very Young Men, Zvi Griliches, J.Pol.Ec. 1976)
. qui ivreg2 lw (iq=med kww age), gmm2s
. di e(sargan)
102.10909
```

20. It is important to note that an Anderson–Rubin confidence region need not be finite nor connected. The test provided in `condivreg` (Moreira and Poi 2003, Mikusheva and Poi 2006) is uniformly most powerful in the situation where there is one endogenous regressor and i.i.d. errors. The Anderson–Rubin test provided by `ivreg2` is a simple and preferable alternative when errors are not i.i.d. or there is more than one endogenous regressor.

```
. mat S0 = e(S)
. qui ivreg2 lw med age (iq=kww), gmm2s smatrix(S0)
. test med age
 ( 1)  med = 0
 ( 2)  age = 0
           chi2(  2) =  102.11
         Prob > chi2 =    0.0000
. qui ivreg2 lw kww age (iq=med), gmm2s smatrix(S0)
. test kww age
 ( 1)  kww = 0
 ( 2)  age = 0
           chi2(  2) =  102.11
         Prob > chi2 =    0.0000
. qui ivreg2 lw med kww (iq=age), gmm2s smatrix(S0)
. test med kww
 ( 1)  med = 0
 ( 2)  kww = 0
           chi2(  2) =  102.11
         Prob > chi2 =    0.0000
```

9 RESET in the IV context

The `ivreset` command performs various flavors of RESET as adapted by Pesaran and Taylor (1999) and Pagan and Hall (1983) for IV estimation. RESET is sometimes called an omitted-variables test (as in official Stata's `ovtest`) but probably is best interpreted as a test of neglected nonlinearities in the choice of functional form (Wooldridge 2002, 124–125). Under the null hypothesis that there are no neglected nonlinearities, the residuals should be uncorrelated with low-order polynomials in $\widehat{y}$, where the $\widehat{y}$'s are predicted values of the dependent variable. In the `ivreset` implementation of the test, an equation of the form $y = X\beta + Y\gamma + v$ is estimated by IV, where the Ys are powers of $\widehat{y}$, the fitted value of the dependent variable y. Under the null hypothesis that there are no neglected nonlinearities and the equation is otherwise well specified, γ should not be significantly different from zero.

As Pesaran and Taylor (1999) and Pagan and Hall (1983) point out, however, RESET for an IV regression cannot use the standard IV predicted values $\widehat{y} \equiv X\widehat{\beta}$ because X includes endogenous regressors that are correlated with u. Instead, RESET must be implemented using "forecast values" of y that are functions of the instruments (exogenous variables) only. In the Pagan–Hall version of the test, the forecast values $\widehat{y}$ are the reduced-form predicted values of y, i.e., the predicted values from a regression of y on the instruments Z. In the Pesaran–Taylor version of the test, the forecast values $\widehat{y}$ are the "optimal forecast" values. The optimal forecast (predictor) $\widehat{y}$ is defined as $\widehat{X}\widehat{\beta}$, where $\widehat{\beta}$ is the IV estimate of the coefficients and $\widehat{X} \equiv [Z\widehat{\Pi}\ Z_2]$, i.e., the reduced-form predicted values of the endogenous regressors plus the exogenous regressors. If the equation is exactly identified, the optimal forecasts and reduced-form forecasts coincide, and the Pesaran–Taylor and Pagan–Hall tests are identical.

The `ivreset` test types vary according to the polynomial terms (square, cube, fourth power of $\widehat{y}$), the choice of forecast values (Pesaran–Taylor optimal forecasts or Pagan–Hall reduced-form forecasts), test statistic (Wald or GMM-distance), and large- versus small-sample statistic (χ^2 or F statistic). The test statistic is distributed with degrees of freedom equal to the number of polynomial terms. The default is the Pesaran–Taylor version using the square of the optimal forecast of y and a χ^2 Wald statistic with one degree of freedom.

If the original `ivreg2` estimation was heteroskedasticity-robust, cluster–robust, AC, or HAC, the reported RESET will be as well. The `ivreset` command can also be used after OLS regression with `regress` (see [R] **regress**) or `ivreg2` when there are no endogenous regressors. Then either a standard RESET using fitted values of y or a robust test corresponding to the specification of the original regression is reported.

We illustrate use of `ivreset` using a model fitted to the Griliches data:

```
. use http://fmwww.bc.edu/ec-p/data/hayashi/griliches76.dta
(Wages of Very Young Men, Zvi Griliches, J.Pol.Ec. 1976)
. quietly ivreg2 lw s expr tenure rns smsa (iq=med kww), robust
. ivreset
Ramsey/Pesaran-Taylor RESET test
Test uses square of fitted value of y (X-hat*beta-hat)
Ho: E(y|X) is linear in X
Wald test statistic:              Chi-sq(1) =  4.53   P-value = 0.0332
Test is heteroskedastic-robust
. ivreset, poly(4) rf small
Ramsey/Pagan-Hall RESET test
Test uses square, cube and 4th power of reduced form prediction of y
Ho: E(y|X) is linear in X
Wald test statistic:              F(3,748) =  1.72    P-value = 0.1616
Test is heteroskedastic-robust
```

The first `ivreset` takes all the defaults and corresponds to a second-order polynomial in $\widehat{y}$ with the Pesaran–Smith optimal forecast and a Wald χ^2 test statistic that rejects the null at better than 95%. The second uses a fourth-order polynomial and requests the Pagan–Hall reduced-form forecast with a Wald F statistic, falling short of the 90% level of significance.

10 A test for autocorrelated errors in the IV context

The `ivactest` command performs the Cumby and Huizinga (1992) generalization of a test proposed by Sargan (1988) for serial independence of the regression errors, which in turn generalizes the test proposed by Breusch and Godfrey (`estat bgodfrey`) applicable to OLS regressions. Sargan's extension of the Breusch–Godfrey test to the IV context, the serial correlation (SC) test, is described as a "general misspecification chi-squared statistic" by Pesaran and Taylor (1999, 260). The SC test statistic is based on the residuals of the IV regression and its conventional VCE. Cumby and Huizinga extend Sargan's test to cases in which the IV VCE was estimated as heteroskedasticity-robust, autocorrelation-robust, or HAC.

Cumby and Huizinga (1992) state that the null hypothesis of the test is "that the regression error is a moving average of known order $q \geq 0$ against the general alternative that autocorrelations of the regression error are nonzero at lags greater than q. The test ... is thus general enough to test the hypothesis that the regression error has no serial correlation ($q = 0$) or the null hypothesis that serial correlation in the regression error exists, but dies out at a known finite lag ($q > 0$)" (p. 185).

The Cumby–Huizinga test is especially attractive because it can be used in three frequently encountered cases where alternatives such as the Box–Pierce test ([TS] **wntestq**), Durbin's h test (`estat durbinalt`), and the Breusch–Godfrey test (`estat bgodfrey`) are not applicable. One of these cases is the presence of endogenous regressors, which renders each of these tests invalid. A second case involves the overlapping data commonly encountered in financial markets where the observation interval is shorter than the holding period, which requires the estimation of the induced moving-average (MA) process. The Cumby–Huizinga test avoids estimation of the MA process by using only the sample autocorrelations of the residuals and a consistent estimate of their asymptotic covariance matrix. The third case involves conditional heteroskedasticity of the regression error term, which is also handled without difficulty by the Cumby–Huizinga test.

If the prior estimation command estimated a VCE under the assumption of i.i.d. errors, the Cumby–Huizinga statistic becomes the Breusch–Godfrey statistic for the same number of autocorrelations and will return the same result as `estat bgodfrey`. That special case of the test was that proposed by Sargan in an unpublished working paper in 1976 (reprinted in Sargan 1988).

Two parameters may be specified in `ivactest`: `s()`, the number of lag orders to be tested, and `q()`, the lowest lag order to be tested.[21] By default, `ivactest` takes `s=1` and `q=0` and produces a test for AR(1). A test for AR(p) may be produced with `s=`p. Under the null hypothesis of serial independence for lags $q - (q + s)$, the Cumby–Huizinga test statistic is distributed χ^2 with s degrees of freedom.

We illustrated the use of `ivactest` in section 3.

11 Syntax

These syntax diagrams describe all the programs in the `ivreg2` suite, including those that have not been substantially modified since their documentation in Baum, Schaffer, and Stillman (2003).

21. If the previous command estimated a VCE under the assumption of i.i.d. errors, `q()` must be 0.

```
ivreg2 depvar [varlist1] (varlist2=varlist_iv) [if] [in] [weight] [, gmm gmm2s
   bw(#|auto) kernel(string) liml fuller(#) kclass(#) coviv cue
   cueinit(matrix) cueoptions(string) b0(matrix) robust cluster(varname)
   orthog(varlist_ex) endogtest(varlist_en) redundant(varlist_ex)
   partial(varlist_ex) small noconstant smatrix(matrix) wmatrix(matrix)
   first ffirst savefirst savefprefix(string) rf saverf
   saverfprefix(string) nocollin noid level(#) noheader nofooter
   eform(string) depname(varname) plus]

overid [, chi2 dfr f all depvar(varname)]

ivhettest [varlist][, ivlev ivsq fitlev fitsq ph phnorm nr2 bpg all]

ivendog [varlist]

ivreset [, polynomial(#) rform cstat small]

ivactest [, s(#) q(#)]
```

12 A summary of ivreg2 estimation options

The version of `ivreg2` accompanying this paper uses a different syntax for specifying the type of estimator to be employed. In previous versions of the software (Baum, Schaffer, and Stillman 2003; 2004; 2005), the `gmm` option implied a heteroskedasticity-robust estimator. When the `gmm` option was combined with the `bw()` option, estimates were autocorrelation-robust but not heteroskedasticity-robust. This version of `ivreg2` uses a new taxonomy of estimation options, summarized below. The `gmm2s` option by itself produces the IV/2SLS estimator, as described in section 2.5. One of the options—`robust`, `cluster()`, or `bw()`—must be added to generate two-step EGMM estimates.

Table 1 summarizes the estimator and the properties of its point and interval estimates for each combination of estimation options.

Table 1: Summary of `ivreg2` estimation options

Estimator option	Covariance matrix option(s): (none)	Covariance matrix option(s): `robust`, `cluster()`, `bw()`, `kernel()`
(none)	IV/2SLS SEs consistent under homoskedasticity	IV/2SLS with robust SEs
`liml`	LIML SEs consistent under homoskedasticity	LIML with robust SEs
`gmm2s`	IV/2SLS SEs consistent under homoskedasticity	Two-step GMM with robust SEs
`cue`	LIML SEs consistent under homoskedasticity	CUE/GMM with robust SEs
`kclass()`	k-class estimator SEs consistent under homoskedasticity	k-class estimator with robust SEs
`wmatrix()`	possibly IGMM SEs consistent under homoskedasticity	IGMM with robust SEs
`gmm2s` + `wmatrix()`	Two-step GMM with user-specified first step SEs consistent under homoskedasticity	Two-step GMM with robust SEs

12.1 ivreg2 versus ivregress

Stata's official `ivregress` command, available in Stata 10 and later, provides an LIML and GMM estimator in addition to two-stage least squares. The GMM estimator can produce HAC estimates, as discussed in section 3 but cannot produce AC estimates. The `ivregress` command does not support the general k-class estimator nor GMM/CUE but provides an "iterative GMM" estimator. Overidentification tests and first-stage statistics are available as `estat` subcommands. `ivreg2`'s ability to partial-out regressors with the `partial()` option is not available in `ivregress`.

Several tests performed by `ivreg2` are also not available with `ivregress`. These include the GMM distance tests of endogeneity/exogeneity discussed in section 5, the general underidentification/weak-identification test of Kleibergen and Paap (2006) discussed in section 7, and tests that permit inference robust to weak instruments. In diagnosing potentially weak instruments, `ivreg2`'s ability to save the first-stage regressions is also unique.

The default behavior of `ivregress gmm` with the `vce(robust)` option produces coefficients that match those of `ivreg2, gmm2s robust` but with different standard errors. Whereas `ivreg2` uses the expression for the VCE of the efficient GMM estimator (6), `ivregress gmm` calculates the VCE as if the estimator was not efficient using (4) from a new estimate of the asymptotic covariance matrix S based on the second-step GMM residuals. To replicate this behavior of `ivregress gmm` and generate identical standard errors, the `wmatrix()` option of `ivreg2` can be used

```
. webuse abdata, clear
. qui ivreg2 n (w = k ys), gmm2s robust
. mat W_GMM2S = e(W)
. ivregress gmm n (w =k ys), vce(robust)
. ivreg2 n (w = k ys), wmatrix(W_GMM2S) robust
```

Both programs' methodology for calculation of the variance–covariance matrix yield consistent estimates.

13 Acknowledgments

We are grateful to many members of the Stata user community who have assisted in the identification of useful features in the `ivreg2` suite and helped identify problems with the programs. We thank (without implicating) Austin Nichols for his suggestions on this draft, Frank Kleibergen for discussions about testing for identification, and Manuel Arellano and Graham Elliott for helpful discussions of GMM/CUE. Some portions of the discussion of weak instruments in section 7.3 are taken from chapter 8 of Baum (2006).

14 References

Ahn, S. C. 1997. Orthogonality tests in linear models. *Oxford Bulletin of Economics and Statistics* 59: 183–186.

Anderson, T. W. 1951. Estimating linear restrictions on regression coefficients for multivariate normal distributions. *Annals of Mathematical Statistics* 22: 327–351.

———. 1984. *Introduction to Multivariate Statistical Analysis*. 2nd ed. New York: Wiley.

Anderson, T. W., and H. Rubin. 1949. Estimation of the parameters of a single equation in a complete system of stochastic equations. *Annals of Mathematical Statistics* 20: 46–63.

Andrews, D. W. K. 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* 59: 817–858.

Baum, C. F. 2006. *An Introduction to Modern Econometrics Using Stata*. College Station, TX: Stata Press.

Baum, C. F., M. E. Schaffer, and S. Stillman. 2003. Instrumental variables and GMM: Estimation and testing. *Stata Journal* 3: 1–31.

———. 2004. Software updates: Instrumental variables and GMM: Estimation and testing. *Stata Journal* 4: 224.

———. 2005. Software updates: Instrumental variables and GMM: Estimation and testing. *Stata Journal* 5: 607.

Bound, J., D. Jaeger, and R. Baker. 1995. Problems with instrumental variable estimation when the correlation between the instruments and the endogenous explanatory variables is weak. *Journal of the American Statistical Association* 90: 443–450.

Breusch, T., H. Qian, P. Schmidt, and D. Wyhowski. 1999. Redundancy of moment conditions. *Journal of Econometrics* 91: 89–111.

Chernozhukov, V., and C. Hansen. 2005. The reduced form: A simple approach to inference with weak instruments. Working Paper, University of Chicago, Graduate School of Business.

Cragg, J. G., and S. G. Donald. 1993. Testing identifiability and specification in instrumental variables models. *Econometric Theory* 9: 222–240.

Cumby, R. E., and J. Huizinga. 1992. Testing the autocorrelation structure of disturbances in ordinary least squares and instrumental variables regressions. *Econometrica* 60: 185–195.

Cushing, M., and M. McGarvey. 1999. Covariance matrix estimation. In *Generalized Methods of Moments Estimation*, ed. L. Matyas. Cambridge: Cambridge University Press.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics.* 2nd ed. New York: Oxford University Press.

Dufour, J.-M. 2003. Identification, weak instruments and statistical inference in econometrics. Working Paper 2003s-49, CIRANO.

Frisch, R., and F. V. Waugh. 1933. Partial time regressions as compared with individual trends. *Econometrica* 1: 387–401.

Fuller, W. A. 1977. Some properties of a modification of the limited information estimator. *Econometrica* 45: 939–53.

Greene, W. H. 2008. *Econometric Analysis.* 6th ed. Upper Saddle River, NJ: Prentice Hall.

Hahn, J., and J. Hausman. 2002. Notes on bias in estimators for simultaneous equation models. *Economics Letters* 75: 237–41.

Hahn, J., J. Hausman, and G. Kuersteiner. 2004. Estimation with weak instruments: Accuracy of higher-order bias and MSE approximations. *Econometrics Journal* 7: 272–306.

Hall, A. R. 2005. *Generalized Method of Moments*. Oxford: Oxford University Press.

Hall, A. R., and F. P. M. Peixe. 2003. A consistent method for the selection of relevant instruments. *Econometric Reviews* 22: 269–287.

Hall, A. R., G. D. Rudebusch, and D. W. Wilcox. 1996. Judging instrument relevance in instrumental variables estimation. *International Economic Review* 37: 283–298.

Hansen, L. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* 50: 1029–1054.

Hansen, L., J. Heaton, and A. Yaron. 1996. Finite sample properties of some alternative GMM estimators. *Journal of Business and Economic Statistics* 14: 262–280.

Hayashi, F. 2000. *Econometrics*. Princeton, NJ: Princeton University Press.

Kleibergen, F., and R. Paap. 2006. Generalized reduced rank tests using the singular-value decomposition. *Journal of Econometrics* 127: 97–126.

Kleibergen, F., and M. Schaffer. 2007. ranktest: Stata module to test the rank of a matrix using the Kleibergen–Paap rk statistic. Boston College Department of Economics, Statistical Software Components S456865. Downloadable from http://ideas.repec.org/c/boc/bocode/s456865.html.

Lovell, M. 1963. Seasonal adjustment of economic time series. *Journal of the American Statistical Association* 58: 993–1010.

Mikusheva, A., and B. P. Poi. 2006. Tests and confidence sets with correct size when instruments are potentially weak. *Stata Journal* 6: 335–347.

Moreira, M. J., and B. P. Poi. 2003. Implementing tests with correct size in the simultaneous equations model. *Stata Journal* 3: 57–70.

Nagar, A. L. 1959. The bias and moment matrix of the general k-class estimators of the parameters in simultaneous equations. *Econometrica* 27: 575–595.

Newey, W. K., and K. D. West. 1987a. Hypothesis testing with efficient method of moments estimation. *International Economic Review* 28: 777–787.

———. 1987b. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica* 55: 703–708.

———. 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653.

Pagan, A. R., and D. Hall. 1983. Diagnostic tests as residual analysis. *Econometric Reviews* 2: 159–218.

Pesaran, M. H., and L. W. Taylor. 1999. Diagnostics for IV regressions. *Oxford Bulletin of Economics & Statistics* 61: 255–281.

Poskitt, D., and C. Skeels. 2002. Assessing instrumental variable relevance: An alternative measure and some exact finite sample theory. Technical report, Monash University and University of Melbourne.

Sargan, J. 1988. Testing for misspecification after estimation using instrumental variables. In *Contributions to econometrics: John Denis Sargan*, ed. E. Maasoumi, vol. 1. Cambridge: Cambridge University Press.

Staiger, D., and J. H. Stock. 1997. Instrumental variables regression with weak instruments. *Econometrica* 65: 557–86.

Stock, J., and M. Watson. 2003. *Introduction to Econometrics*. Reading, MA: Addison–Wesley.

Stock, J. H., and J. H. Wright. 2000. GMM with weak identification. *Econometrica* 68: 1055–1096.

Stock, J. H., J. H. Wright, and M. Yogo. 2002. A survey of weak instruments and weak identification in generalized method of moments. *Journal of Business and Economic Statistics* 20: 518–529.

Stock, J. H., and M. Yogo. 2005. Testing for weak instruments in linear IV regression. In *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, ed. D. W. K. Andrews and J. H. Stock, 80–108. Cambridge: Cambridge University Press.

Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

———. 2003. *Introductory Econometrics: A Modern Approach*. 2nd ed. New York: Thomson Learning.

About the authors

Christopher F. Baum is associate professor of economics at Boston College. He is an associate editor of *Computational Economics*, the *Stata Journal*, and the *Journal of Statistical Software*. Baum founded and manages the Boston College Statistical Software Components (SSC) archive at RePEc (*http://repec.org*). His recent research has focused on the effects of uncertainty on international trade flows, bank lending behavior, and firms' cash holdings.

Mark E. Schaffer is professor of economics and Director of the Centre for Economic Reform and Transformation (CERT) at Heriot–Watt University, Edinburgh, Scotland. He is also a Research Fellow at the Centre for Economic Policy Research (CEPR), the Institute for the Study of Labor (IZA), and the William Davidson Institute. His research interests include various aspects of firm and household behavior in the transition countries of Eastern Europe, the former USSR, and East Asia.

Steven Stillman is a senior fellow at Motu Economic and Public Policy Research Trust. He is also an affiliated research fellow at the Institute for the Study of Labour (IZA), the Centre

for Research and Analysis of Migration (CReAM), and the William Davidson Institute. Recent work includes analyzing the effect of suffering injuries on future labor market outcomes in New Zealand, the impact of migration to New Zealand on the income and health of Pacific Islanders, and the effect of economic shocks in Russia on household consumption, individual nutrition, and overall living standards.

The Stata Journal (2007)
7, Number 4, pp. 507–541

Causal inference with observational data

Austin Nichols
Urban Institute
Washington, DC
austinnichols@gmail.com

Abstract. Problems with inferring causal relationships from nonexperimental data are briefly reviewed, and four broad classes of methods designed to allow estimation of and inference about causal parameters are described: panel regression, matching or reweighting, instrumental variables, and regression discontinuity. Practical examples are offered, and discussion focuses on checking required assumptions to the extent possible.

Keywords: st0136, xtreg, psmatch2, nnmatch, ivreg, ivreg2, ivregress, rd, lpoly, xtoverid, ranktest, causal inference, match, matching, reweighting, propensity score, panel, instrumental variables, excluded instrument, weak identification, regression, discontinuity, local polynomial

1 Introduction

Identifying the causal impact of some variables, X^T, on y is difficult in the best of circumstances, but faces seemingly insurmountable problems in observational data, where X^T is not manipulable by the researcher and cannot be randomly assigned. Nevertheless, estimating such an impact or "treatment effect" is the goal of much research, even much research that carefully states all findings in terms of associations rather than causal effects. I will call the variables X^T the "treatment" or treatment variables, and the term simply denotes variables of interest—they need not be binary (0/1) nor have any medical or agricultural application.

Experimental research designs offer the most plausibly unbiased estimates, but experiments are frequently infeasible due to cost or moral objections—no one proposes to randomly assign smoking to individuals to assess health risks or to randomly assign marital status to parents so as to measure the impacts on their children. Four types of quasiexperimental research designs offering approaches to causal inference using observational data are discussed below in rough order of increasing internal validity (Shadish, Cook, and Campbell 2002):

- Ordinary regression and panel methods
- Matching and reweighting estimators
- Instrumental variables (IV) and related methods
- Regression discontinuity (RD) designs

Each has strengths and weaknesses discussed below. In practice, the data often dictate the method, but it is incumbent upon the researcher to discuss and check (insofar as possible) the assumptions that allow causal inference with these models, and to qualify conclusions appropriately. Checking those assumptions is the focus of this paper.

A short summary of these methods and their properties is in order before we proceed. To eliminate bias, the regression and panel methods typically require confounding variables either to be measured directly or to be invariant along at least one dimension in the data, e.g., invariant over time. The matching and reweighting estimators require that selection of treatment X^T depend only on observable variables, both a stronger and weaker condition. IV methods require extra variables that affect X^T but not outcomes directly and throw away some information in X^T to get less efficient and biased estimates that are, however, consistent (i.e., approximately unbiased in sufficiently large samples). RD methods require that treatment X^T exhibit a discontinuous jump at a particular value (the "cutoff") of an observed assignment variable and provide estimates of the effect of X^T for individuals with exactly that value of the assignment variable. To get plausibly unbiased estimates, one must either give up some efficiency or generalizability (or both, especially for IV and RD) or make strong assumptions about the process determining X^T.

1.1 Identifying a causal effect

Consider an example to fix ideas. Suppose that for people suffering from depression, the impact of mental health treatment on work is positive. However, those who seek mental health treatment (or seek more of it) are less likely to work, even conditional on all other observable characteristics, because their depression is more severe (in ways not measured by any data we can see). As a result, we estimate the impact of treatment on work, incorrectly, as being negative.

A classic example of an identification problem is the effect of college on earnings (Card 1999, 2001). College is surely nonrandomly assigned, and there are various important unobserved factors, including the alternatives available to individuals, their time preferences, the prices and quality of college options, academic achievement (often "ability" in economics parlance), and access to credit. Suppose that college graduates earn 60 and others earn 40 on average. One simple (implausible but instructive) story might be that college has no real effect on productivity or earnings, but those who pass a test S that grants entry to college have productivity of 60 on average and go to college. Even in the absence of college, they would earn 60 if they could signal (see Spence 1973) productivity to employers by another means (e.g., by merely reporting the result of test S). Here extending college to a few people who failed test S would not improve their productivity at all and might not affect their earnings (if employers observed the result of test S).

If we could see the outcome for each case when treated and not treated (assuming a single binary treatment X^T) or an outcome y for each possible level of X^T, we could calculate the treatment effect for each individual i and compute an average. Of course, this is not possible as each gets some level of X^T or some history of X^T in a panel setting. Thus we must compare individuals i and j with different X^T to estimate an average treatment effect (ATE). When X^T is nonrandomly assigned, we have no guarantee that individuals i and j are comparable in their response to treatment or what their outcome would have been given another X^T, even on average. The notion of "potential outcomes" (Rubin 1974) is known as the *Rubin causal model*. Holland (1986) provided the classic exposition of this now dominant theoretical framework for causal inference, and Rubin (1990) clarified the debt that the Rubin causal model owes to Neyman (1923) and Fisher (1918, 1925).

In all the models discussed in this paper, we assume that the effect of treatment is on individual observations and does not spill over onto other units. This is called the stable-unit-treatment-value assumption by Rubin (1986). Often, this may be only approximately true, e.g., the effect of a college education is not only on the earnings of the recipient, since each worker participates in a labor market with other graduates and nongraduates.

What is the most common concern about observational data? If X^T is correlated with some other variable X^U that also has a causal impact on y, but we do not measure X^U, we might assess the impact of X^T as negative even though its true impact is positive. Sign reversal is an extreme case, sometimes called *Simpson's paradox*, though it is not a paradox and Simpson (1951) pointed out the possibility long after Yule (1903). More generally, the estimate of the impact of X^T may be biased and inconsistent when X^T is nonrandomly assigned. That is, even if the sign of the estimated impact is not the opposite of the true impact, our estimate need not be near the true causal impact on average, nor approach it asymptotically. This central problem is usually called *omitted-variable bias* or *selection bias* (here selection refers to the nonrandom selection of X^T, not selection on the dependent variable as in `heckman` and related models).

1.2 Sources of bias and inconsistency

The selection bias (or omitted-variable bias) in an ordinary regression arises from endogeneity (a regressor is said to be endogenous if it is correlated with the error), a condition that also occurs if the explanatory variable is measured with error or in a system of "simultaneous equations" (e.g., suppose that work also has a causal impact on mental health or higher earnings cause increases in education; in this case, it is not clear what impact, if any, our single-equation regressions identify).

Often a suspected type of endogeneity can be reformulated as a case of omitted variables, perhaps with an unobservable (as opposed to merely unobserved) omitted variable, about which we can nonetheless make some predictions from theory to sign the likely bias.

The formula for omitted-variable bias in linear regression is instructive. With a true model

$$y = \beta_0 + X^T\beta_T + X^U\beta_U + \varepsilon$$

where we regress y on X^T but leave out X^U (for example, because we cannot observe it), the estimate of β_T has bias

$$E(\widehat{\beta}_T) - \beta_T = \delta\beta_U$$

where δ is the coefficient of an auxiliary regression of X^U on X^T (or the matrix of coefficients of stacked regressions when X^U is a matrix containing multiple variables) so the bias is proportional to the correlation of X^U and X^T and to the effect of X^U (the omitted variables) on y.

In nonlinear models, such as a `probit` or `logit` regression, the estimate will be biased and inconsistent even when X^T and X^U are uncorrelated, though Wooldridge (2002, 471) demonstrates that some quantities of interest may still be identified under additional assumptions.

1.3 Sensitivity testing

Manski (1995) demonstrates how a causal effect can be bounded under very unrestrictive assumptions and then how the bounds can be narrowed under more restrictive parametric assumptions. Given how sensitive the quasiexperimental methods are to assumptions (selection on observables, exclusion restrictions, exchangeability, etc.), some kind of sensitivity testing is in order no matter what method is used. Rosenbaum (2002) provides a comprehensive treatment of formal sensitivity testing under various parametric assumptions.

Lee (2005) advocates another useful method of bounding treatment effects, which was used in Leibbrandt, Levinsohn, and McCrary (2005).

1.4 Systems of equations

Some of the techniques discussed here to address selection bias are also used in the simultaneous-equations setting. The literature on structural equations models is extensive, and a system of equations may encode a complicated conceptual causal model, with many "causal arrows" drawn to and from many variables. The present exercise of identifying the causal impact of some limited set of variables X^T on a single outcome y can be seen as restricting our attention in such a complicated system to just one equation, and identifying just some subset of causal effects.

For example, in a simplified supply-and-demand system:

$$\mathrm{lnQ}_{\mathrm{supply}} = e_s \mathrm{lnP} + a\mathrm{TransportCost} + \varepsilon_s$$

$$\mathrm{lnQ}_{\mathrm{demand}} = e_d \mathrm{lnP} + b\mathrm{Income} + \varepsilon_d$$

where price (lnP) is endogenously determined by a market-clearing condition $\mathrm{lnQ}_{\mathrm{supply}} = \mathrm{lnQ}_{\mathrm{demand}}$, our present enterprise limits us to identifying only the demand elasticity e_d using factors that shift supply to identify exogenous shifts in price faced by consumers (exogenous relative to the second equation's error ε_d), or identifying only the supply elasticity e_s using factors that shift demand to identify exogenous shifts in price faced by firms (exogenous relative to the first equation's error ε_s).

See [R] **reg3** for alternative approaches that can simultaneously identify parameters in multiple equations, and Heckman and Vytlacil (2004) and Goldberger and Duncan (1973) for more detail.

1.5 ATE

In an experimental setting, typically the only two quantities to be estimated are the sample ATE or the population ATE—both estimated with a difference in averages across treatment groups (equal in expectation to the mean of individual treatment effects over the full sample). In a quasiexperimental setting, several other ATEs are commonly estimated: the ATE on the treated, the ATE on the untreated or control group, and a variety of local ATEs (LATE)—local to some range of values or some subpopulation. One can imagine constructing at least 2^N different ATE estimates in a sample of N observations, restricting attention to two possible weights for each observation. Allowing a variety of weights and specifications leads to infinitely many LATE estimators, not all of which would be sensible.

For many decision problems, a highly relevant effect estimate is the marginal treatment effect (MTE), either the ATE for the marginal treated case—the expected treatment effect for the case that would get treatment with a small expansion of the availability of treatment—or the average effect of a small increase in a continuous treatment variable. Measures of comparable MTEs for several options can be used to decide where a marginal dollar (or metaphorical marginal dollar, including any opportunity costs and currency translations) should be spent. In other words, with finite resources, we care more about budget-neutral improvements in effectiveness than the effect of a unit increase in treatment, so we can choose among treatment options with equal cost. Quasiexperimental methods, especially IV and RD, often estimate such MTEs directly.

If the effect of a treatment X^T varies across individuals (i.e., it is not the case that $\beta_i = \beta$ for all i), the ATE for different subpopulations will differ. We should expect different consistent estimators to converge to different quantities. This problem is larger than the selection-bias issue. Even in the absence of endogenous selection of X^T (but possibly with some correlation between X_i^T and β_i, itself now properly regarded as a random variable) in a linear model, ordinary least squares (OLS) will not, in general, be consistent for the average over all i of individual effects β_i. Only with strong distributional assumptions can we proceed; e.g., if we assume β_i is normally distributed then the ATE may be consistently estimated by `xtmixed` or `xtrc`, or if we assume X^T is normally distributed then the ATE may be consistently estimated by OLS.

2 Regression and panel methods

If an omitted variable can be measured or proxied by another variable, an ordinary regression may yield an unbiased estimate. The most efficient estimates (ignoring issues around weights or nonindependent errors) are produced by OLS when it is unbiased. The measurement error entailed in a proxy for an unobservable, however, could actually exacerbate bias, rather than reduce it. One is usually concerned that cases with differing X^T may also differ in other ways, even conditional on all other observables X^C ("control" variables). Nonetheless, a sequence of ordinary regressions that add or drop variables can be instructive as to the nature of various forms of omitted-variable bias in the available data.

A complete discussion of panel methods would not fit in any one book, much less this article. However, the idea can be illuminated with one short example using linear regression.

Suppose that our theory dictates a model is of the form

$$y = \beta_0 + X^T\beta_T + X^U\beta_U + \varepsilon$$

where we do not observe X^U. The omitted variables X^U vary only across groups, where group membership is indexed by i, so a representative observation can be written as

$$y_{it} = \beta_0 + X_{it}^T\beta_T + u_i + \varepsilon_{it}$$

where $u_i = X_i^U\beta_U$. Then we can eliminate the bias arising from omission of X^U by differencing

$$y_{it} - y_{is} = (X_{it}^T - X_{is}^T)\beta_T + (\varepsilon_{it} - \varepsilon_{is})$$

using various definitions of s.

The idea of using panel methods to identify a causal impact is to use an individual panel i as its own control group, by including information from multiple points in time. The second dimension of the data indexed by t need not be time, but it is a convenient viewpoint.

A fixed-effects (FE) model such as `xtreg, fe` effectively subtracts the within-i mean values of each variable, so, for example, $\overline{X}_i^T = 1/N_i \sum_{s=1}^{N_i} X_{is}^T$, and the model

$$y_{it} - \overline{y}_i = (X_{it}^T - \overline{X}_i^T)\beta_T + (\varepsilon_{it} - \overline{\varepsilon}_i)$$

can be estimated with OLS. This is also called the "within estimator" and is equivalent to a regression that includes an indicator variable for each panel i, allowing for a different intercept term for each panel.

An alternative to the FE model is to use the first difference (FD), i.e., $s = (t-1)$ or

$$y_{it} - y_{i(t-1)} = (X_{it}^T - X_{i(t-1)}^T)\beta_T + (\varepsilon_{it} - \varepsilon_{i(t-1)})$$

which is `regress d.y d.x` in `tsset` data or `xtivreg2 y x, fd` (Schaffer and Stillman 2007), which offers more standard error (SE) corrections beyond `cluster()` and `robust`.

A third option is to use the long difference (LD), keeping only two observations per group. For a balanced panel, if $t = b$ is the last observation and $t = a$ is the first, the model is

$$y_{ib} - y_{ia} = (X_{ib}^T - X_{ia}^T)\beta_T + (\varepsilon_{ib} - \varepsilon_{ia})$$

producing only one observation per group (the difference of the first and last observations).

Figure 1 shows the interpretation of these three types of estimates by showing one panel's contribution to the estimated effect of an indicator variable that equals one for all $t > 3$ (t in 0, ..., 10) and equals zero elsewhere—e.g., a policy that comes into effect at some point in time (at $t = 4$ in the example). The FE estimate compares the mean outcomes before and after, the FD estimate compares the outcome just prior to and just after the change in policy, and the LD estimate compares outcomes well before and well after the change in policy.

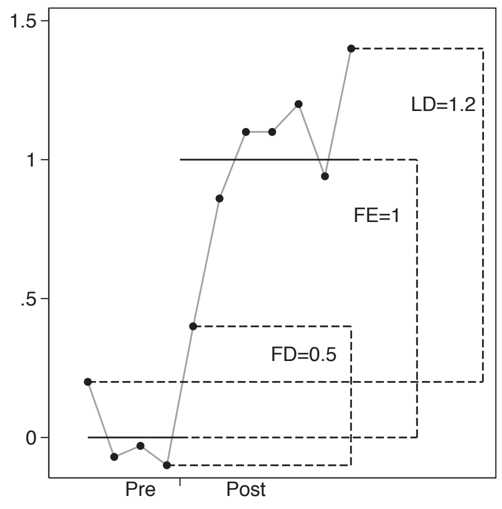

Figure 1: One panel's contributions to FE/FD/LD estimates

Clearly, one must impose some assumptions on the speed with which X^T affects y or have some evidence as to the right time frame for estimation. This type of choice comes up frequently when stock prices are supposed to have adjusted to some news, especially given the frequency of data available; economists believe the new information is capitalized in prices, but not instantaneously. Taking a difference in stock prices between 3 p.m. and 3:01 p.m. is inappropriate but taking a difference over a year is clearly inappropriate as well, because new information arrives continuously.

In panel models, one must usually think carefully about within-panel trends and the frequency of measurement. (We cannot usually obtain consistent estimates of within-panel trends for the same reason that we cannot usually obtain consistent estimates of

FE: the number of parameters increases linearly in the number of panels, N.) Baum (2006) discussed some filtering techniques to get different frequency "signals" from noisy data. A simple method used in Baker, Benjamin, and Stanger (1999) is often attractive, because it offers an easy way to decompose any variable X_t into two orthogonal components: a high-frequency component $(X_t - X_{t-1})/2$ and a low-frequency component $(X_t + X_{t-1})/2$ that together sum to X_t.

A simple example of all three (FE, FD, and LD) is

```
webuse grunfeld
xtreg inv ks, fe vce(cluster company)
regress d.inv d.ks, vce(cluster company)
summarize time, meanonly
generate t=time if time==r(min) | time==r(max)
tsset company t
regress d.inv d.ks, vce(cluster company)
```

Clearly, different assumptions about the error process apply in each case, in addition to assumptions about the speed with which X^T affects y. The FD and LD models require an ordered t index (such as time). The `vce(cluster` *clustvar*`)` option used above should be considered nearly *de rigeur* in panel models to allow for errors that may be correlated within group and not identically distributed across groups. The performance of the cluster–robust estimator is good with 50 or more clusters, or fewer if the clusters are large and balanced (Nichols and Schaffer 2007). For LD, the `vce(cluster` *clustvar*`)` option is equivalent to the `vce(robust)` option, because each group is represented by one observation.

Having eliminated bias due to unobservable heterogeneity across i units, it is often tempting to difference or demean again. It is common to include indicator variables for t in FE models, for example,

```
webuse grunfeld
quietly tabulate year, generate(d)
xtreg inv ks d*, fe vce(cluster company)
```

The above commands create a two-way FE model. If individuals, i, are observed in different settings, j—for example, students who attend various schools or workers who reside in various locales over time—we can also include indicator variables for j in an FE model. Thus we can consider various n-way FE models, though models with large numbers of dimensions for FE may rapidly become unstable or computationally challenging to fit.

The LD, FD, and FE estimators use none of the cross-sectional differences across groups (individuals), i, which can lead to lower efficiency (relative to an estimator that exploits cross-sectional variation). They also drop any variables that do not vary over t within i, so the coefficients on some variables of interest may not be estimated with these methods.

The random-effects estimator (RE) available with `xtreg` exploits cross-sectional variation and reports coefficients on variables that do not vary over t within i, but it requires strong assumptions about error terms that are often violated in practice. Particularly, for RE to be unbiased in situations where FE is unbiased, we must assume that u_i is uncorrelated with X^T_{it} (which contradicts our starting point above, where we worried about a X^U correlated with X^T). There is no direct test of this assumption about an unobservable disturbance term, but **hausman** and **xtoverid** (Schaffer and Stillman 2006) offer a test that the coefficients estimated in both the RE and FE models are the same, e.g.,

```
ssc install xtoverid
webuse grunfeld
egen ik=max(ks*(year==1935)), by(company)
xtreg inv ks ik, re vce(cluster company)
xtoverid
```

where a rejection casts doubt on whether RE is unbiased when FE is biased.

Other xt commands, such as **xtmixed** (see [XT] **xtmixed**) and **xthtaylor** (see [XT] **xthtaylor**), offer a variety of other panel methods that generally make further assumptions about the distribution of disturbances and sources of endogeneity. Typically, there is a tradeoff between improved efficiency bought by making assumptions about the data-generating process versus robustness to various violations of assumptions. See also Griliches and Hausman (1986) for more considerations related to all the above panel methods. Rothstein (2007) offers a useful applied examination of identifying assumptions in FE models and correlated RE models.

Generally, panel methods eliminate the bias because of some unobserved factors and not others. Considering the FE, FD, and LD models, it is often hard to believe that all the selection on unobservables is because of time-invariant factors. Other panel models often require unpalatable distributional assumptions.

3 Matching estimators

For one discrete set of treatments, X^T, we want to compare means or proportions much as we would in an experimental setting. We may be able to include indicators and interactions for factors (in X^C) that affect selection into the treatment group (say, defined by $X^T = 1$), to estimate the impact of treatment within groups of identical X^C using a fully saturated regression. There are also matching estimators (Cochran and Rubin 1973; Stuart and Rubin 2007) that compare observations with X^C by pairing observations that are close by some metric (see also Imai and van Dyk 2004). A set of alternative approaches involve reweighting so the joint or marginal distributions of X^C are identical for different groups.

Matching or reweighting approaches can give consistent estimates of a huge variety of ATEs, but only under the assumptions that the selection process depends on observables and that the model used to match or reweight is a good one. Often we push the problems associated with observational data from estimating the effect of X^T on y down onto estimating the effect of X^C on X^T. For this reason, estimates based on reweighting or matching are unlikely to convince someone unconvinced by OLS results. Selection on observables is not the type of selection most critics have in mind.

3.1 Nearest-neighbor matching

Nearest-neighbor matching pairs observations in the treatment and control groups and computes the difference in outcome y for each pair and then the mean difference across pairs. The Stata command `nnmatch` was described by Abadie et al. (2004). Imbens (2004) covered details of nearest-neighbor matching methods. The downside to nearest-neighbor matching is that it can be computationally intensive, and bootstrapped SEs are infeasible owing to the discontinuous nature of matching (Abadie and Imbens 2006).

3.2 Propensity-score matching

Propensity-score matching essentially estimates each individual's propensity to receive a binary treatment (with a `probit` or `logit`) as a function of observables and matches individuals with similar propensities. As Rosenbaum and Rubin (1983) showed, if the propensity was known for each case, it would incorporate all the information about selection, and propensity-score matching could achieve optimal efficiency and consistency. In practice, the propensity must be estimated and selection is not only on observables, so the estimator will be both biased and inefficient.

Morgan and Harding (2006) provide an excellent overview of practical and theoretical issues in matching and comparisons of nearest-neighbor matching and propensity-score matching. Their expositions of different types of propensity-score matching and simulations showing when it performs badly are particularly helpful. Stuart and Rubin (2007) offer a more formal but equally helpful discussion of best practices in matching.

Typically, one treatment case is matched to several control cases, but one-to-one matching is also common and may be preferred (Glazerman, Levy, and Myers 2003). One Stata command `psmatch2` (Leuven and Sianesi 2003) is available from the Statistical Software Components (SSC) archive (`ssc describe psmatch2`) and has a useful help file. There is another useful Stata command `pscore` (Becker and Ichino 2002; `findit pscore` in Stata). `psmatch2` will perform one-to-one (nearest neighbor or within caliper, with or without replacement), k-nearest neighbors, radius, kernel, local linear regression, and Mahalanobis matching.

Propensity-score methods typically assume a common support; i.e., the range of propensities to be treated is the same for treated and control cases, even if the density functions have different shapes. In practice, it is rare that the ranges of estimated propensity scores are the same for both the treatment and control groups, but they do nearly always overlap. Generalizations about treatment effects should probably be limited to the smallest connected area of common support.

Often a density estimate below some threshold greater than zero defines the end of common support; see Heckman, Ichimura, and Todd (1997) for more discussion. This is because the common support is the range where both densities are nonzero, but the estimated propensity scores take on a finite number of values. Thus the empirical densities will be zero almost everywhere. Generally, we need to use a kernel density estimator like `kdensity` to obtain smooth estimated densities of the propensity score for both treatment and control groups, but then areas of zero density will have positive density estimates. Thus some small value f_0 is redefined to be effectively zero, and the smallest connected range of estimated propensity scores $\widehat{\lambda}$ with $\widehat{f}(\widehat{\lambda}) \geq f_0$ for both treatment and control groups is used in the analaysis, and observations outside this range are discarded.

Regardless of whether the estimation or extrapolation of estimates is limited to a range of propensities or ranges of X^C variables, the analyst should present evidence on how the treatment and control groups differ and on which subpopulation is being studied. The standard graph here is an overlay of kernel density estimates of propensity scores for treatment and control groups. This is easy to create in Stata with `twoway kdensity`.

3.3 Sensitivity testing

Matching estimators have perhaps the most detailed literature on formal sensitivity testing. Rosenbaum (2002) bounds on treatment effects may be constructed by using `psmatch2` and `rbounds`, a user-written command by DiPrete and Gangl (2004), who compare Rosenbaum bounds in a matching model with IV estimates. `sensatt` by Nannicini (2006) and `mhbounds` by Becker and Caliendo (2007) are also Stata programs for sensitivity testing in matching models.

3.4 Reweighting

The propensity score can also be used to reweight treatment and control groups so the distribution of X^C looks the same in both groups. The basic idea is to use a `probit` or `logit` regression of treatment on X^C to estimate the conditional probability $\widehat{\lambda}$ of being in the treatment group and to use the odds $\widehat{\lambda}/(1-\widehat{\lambda})$ as a weight. This is like inverting the test of randomization used in experimental designs to make the group status look as if it were randomly assigned.

As Morgan and Harding (2006) point out, all the matching estimators can also be thought of various reweighting schemes whereby treatment and control observations are reweighted to allow causal inference on the difference in means. A treatment case i matched to k cases in an interval, or k-nearest neighbors, contributes $y_i - k^{-1}\sum_1^k y_j$ to the estimate of a treatment effect. One could easily rewrite the estimate of a treatment effect as a weighted-mean difference.

The reweighting approach leads to a whole class of weighted least-squares estimators and is connected to techniques described by DiNardo, Fortin, and Lemieux (1996), Autor, Katz, and Kerney (2005), Leibbrandt, Levinsohn, and McCrary (2005), and Machado and Mata (2005). These techniques are related to various decomposition techniques in Blinder (1973), Oaxaca (1973), Yun (2004, 2005a,b), Gomulka and Stern (1990), and Juhn, Murphy, and Pierce (1991, 1993). DiNardo (2002) usefully outlines some connections between propensity-score methods and the decomposition techniques.

The `dfl` (Azevedo 2005), `oaxaca` (Jann 2005b), and `jmpierce` (Jann 2005a) commands available from the SSC archive are useful for the latter. The decomposition techniques seek to attribute observed differences in an outcome y both to differences in X^C variables and differences in the associations between X^C variables and y. They are most useful for comparing two distributions where the binary variable defining the group to which an observation belongs is properly considered exogenous, e.g., sex or calendar year. See also Rubin (1986).

The reweighting approach is particularly useful in combining matching-type estimators with other methods, e.g., FE regression. After constructing weights $w = \widehat{\lambda}/(1-\widehat{\lambda})$ (or the product of weights $w = w_0\widehat{\lambda}/(1-\widehat{\lambda})$, where w_0 is an existing weight on the data used in the construction of $\widehat{\lambda}$) that equalize the distributions of X^C, other commands can be run on the reweighted data, e.g., `areg` for a FE estimator.

3.5 Examples

Imagine the outcome is wage and the treatment variable is union membership. One can reweight union members to have distributions of education, age, race/ethnicity, and other job and demographic characteristics equivalent to nonunion workers (or a subset of nonunion workers). One could compare otherwise identical persons within occupation and industry cells by using a regression approach or `nnmatch` with exact matching on some characteristics. An example comparing several regressions with propensity-score matching is

```
ssc install psmatch2
webuse nlswork
xi i.race i.ind i.occ
local x "union coll age ten not_s c_city south nev_m _I*"
regress ln_w union
regress ln_w `x´
generate u=uniform()
sort u
psmatch2 `x´, out(ln_w) ate
twoway kdensity _ps if _tr || kdensity _ps if !_tr
generate w=_ps/(1-_ps)
regress ln_w `x´ [pw=w] if _ps<.3
regress ln_w `x´ [pw=w]
```

The estimated union wage premium is about 13% in a regression but about 15% in the matching estimate of the average benefit to union workers (the ATE on the treated) and about 10% on average for everyone (the ATE). The reweighted regressions give different estimates: for the more than 70% of individuals who are unlikely to be unionized (propensity under 30%), the wage premium is about 9%, and for the full sample, it is about 18%.

Arguably none of these estimates of wage premiums correspond to a readily specified thought experiment, such as "what is the estimated effect on wages of being in a union for a randomly chosen individual?" (the ATE) or "what is the estimated effect on wages of being in a union for an individual just on the margin of being in a union or not?" (the

LATE). DiNardo and Lee (2002) offer a much more convincing set of causal estimates of the LATE by using an RD design (see below).

We could also have estimated the wage premium of a college education by switching `coll` and `union` in the above syntax (to find a wage premium of 25% in a regression or 27% using `psmatch2`). We could use data from Card (1995a,b) on education and wages to find a college wage premium of 29% using a regression or 30% using `psmatch2`.

```
use http://fmwww.bc.edu/ec-p/data/wooldridge/card
generate byte coll=educ>15
local x "coll age exper* smsa* south mar black reg662-reg669"
regress lw `x´
psmatch2 `x´, out(lw) ate
```

We return to this example in the next section.

4 Instrumental variables

An alternative to panel methods and matching estimators is to find another set of variables Z correlated with X^T but not correlated with the error term, e.g., e in

$$y = X^T\beta_T + X^C\beta_C + e$$

so Z must satisfy $E(Z'e) = 0$ and $E(Z'X^T) \neq 0$. The variables Z are called *excluded instruments*, and a class of IV methods can then be used to consistently estimate an impact of X^T on y.

Various interpretations of the IV estimate have been advanced, typically as the LATE (Angrist, Imbens, and Rubin 1996), meaning the effect of X^T on y for those who are induced by their level of Z to have higher X^T. For the college-graduate example, this might be the average gain $E_i\{y_i(t) - y_i(0)\}$ over all those i in the treatment group with $Z = 1$ (where Z might be "lived close to a college" or "received a Pell grant"), arising from an increase from $X^T = 0$ to $X^T = t$ in treatment, i.e., the wage premium due to college averaged over those who were induced to go to college by Z.

The IV estimators are generally only as good as the excluded instruments used, so naturally criticisms of the predictors in a standard regression model become criticisms of the excluded instruments in an IV model.

Also, the IV estimators are biased, but consistent, and are much less efficient than OLS. Thus failure to reject the null should not be taken as acceptance of the alternative. That is, one should never compare the IV estimate with only a zero effect; other plausible values should be compared as well, including the OLS estimate. Some other common pitfalls discussed below include improper exclusion restrictions (addressed with overidentification tests) and weak identification (addressed with diagnostics and robust inference).

Since IV estimators are biased in finite samples, they are justified only for large samples. Nelson and Startz (1990) showed how strange the finite sample behavior of an

IV estimator can be. Bound, Jaeger, and Baker (1995) showed that even large samples of millions of observations are insufficient for asymptotic justifications to apply in the presence of weak instruments (see also Stock and Yogo 2005).

4.1 Key assumptions

Because IV can lead one astray if any of the assumptions is violated, anyone using an IV estimator should conduct and report tests of the following:

- instrument validity (overidentification or `overid` tests)
- endogeneity
- identification
- presence of weak instruments
- misspecification of functional form (e.g., RESET)

Further discussion and suggestions on what to do when a test is failed appear in the relevant sections below.

4.2 Forms of IV

The standard IV estimator in a model

$$y = X^T\beta_T + X^C\beta_C + e$$

where we have Z satisfying $E(Z'e)=0$ and $E(Z'X^T)\neq 0$ is

$$\widehat{\beta}^{\rm IV} = \begin{pmatrix} \widehat{\beta}_T^{\rm IV} \\ \\ \widehat{\beta}_C^{\rm IV} \end{pmatrix} = (X'P_ZX)^{-1}X'P_Zy$$

(ignoring weights), where $X=(X^TX^C)$ and P_Z is the projection matrix $Z_a(Z_a'Z_a)^{-1}Z_a'$ with $Z_a=(ZX^C)$. We use the component of X^T along Z, which is exogenous, as the only source of variation in X^T that we use to estimate the effect on y.

These estimates are easily obtained in Stata 6–9 with the syntax `ivreg y xc* (xt* = z*)`, where `xc*` are all exogenous "included instruments" X^C and `xt*` are endogenous variables X^T. In Stata 10, the syntax is `ivregress 2sls y xc* (xt* = z*)`. For Stata 9 and later, the `ivreg2` command (Baum, Schaffer, and Stillman 2007) would be typed as

```
ssc install ivreg2
ivreg2 y xc* (xt* = z*)
```

Example data for using these commands can be easily generated, e.g.,

```
use http://fmwww.bc.edu/ec-p/data/wooldridge/card, clear
rename lw y
rename nearc4 z
rename educ xt
rename exper xc
```

The standard IV estimator is equivalent to two forms of two-stage estimators. The first, which gave rise to the moniker *two-stage least squares* (2SLS), has you regress X^T on X^C and Z, predict $\widehat{X}_T$, and then regress y on $\widehat{X}_T$ and X^C. The coefficient on $\widehat{X}_T$ is $\widehat{\beta}_T^{\mathrm{IV}}$, so

```
foreach xt of varlist xt* {
        regress `xt´ xc* z*
        predict `xt´_hat
}
regress y xt*_hat xc*
```

will give the same estimates as the above IV commands. However, the reported SEs will be wrong as Stata will use $\widehat{X}_T$ rather than X^T to compute them. Even though IV is not implemented in these two stages, the conceptual model of these first-stage and second-stage regressions is pervasive, and the properties of said first-stage regressions are central to the section on identification and weak instruments below.

The second two-stage estimator that generates identical estimates is a *control-function approach*. Regress each variable in X^T on the other variables in X^T, X^C, and Z to predict the errors $\widehat{v}_T = X^T - \widehat{X}_T$ and then regress y on X^T, $\widehat{v}_T$, and X^C. You will find that the coefficient on X^T is $\widehat{\beta}_T^{\mathrm{IV}}$, and tests of significance on each $\widehat{v}_T$ are tests of endogeneity of each X^T. Thus

```
capture drop *_hat
unab xt: xt*
foreach v of loc xt {
        local otht: list xt-v
        regress `v´ xc* z* `otht´
        predict v_`xt´, resid
}
regress y xt* xc* v_*
```

will give the IV estimates, though again the standard errors will be wrong. However, the tests of endogeneity (given by the reported p-values on variables `v_*` above) will be correct. A similar approach works for nonlinear models such as `probit` or `poisson` (`help ivprobit` and `findit ivpois` for relevant commands). The tests of endogeneity in nonlinear models given by the control-function approach are also robust (see, for example, Wooldridge 2002, 474 or 665).

The third two-stage version of the IV strategy, which applies for one endogenous variable and one excluded instrument, is sometimes called the *Wald estimator*. First, regress X^T on X^C and Z (let $\widehat{\pi}$ be the estimated coefficient on Z) and then regress y on Z and X^C (let $\widehat{\gamma}$ be the estimated coefficient on Z). The ratio of coefficients on Z ($\widehat{\gamma}/\widehat{\pi}$) is $\widehat{\beta}_{\mathrm{IV}}$, so

```
regress xt z xc*
local p=_b[z]
regress y z xc*
local g=_b[z]
display `g´/`p´
```

will give the same estimate as the IV command `ivreg2 y xc* (xt=z)`. The regression of y on Z and X^C is sometimes called the *reduced-form regression*. This name is often applied to other regressions, so I will avoid using the term.

The generalized method of moments, limited-information maximum likelihood, and continuously updated estimation and generalized method of moments forms of IV are discussed at length in Baum, Schaffer, and Stillman (2007). Various implementations are available with the `ivregress` and `ivreg2` commands. Some forms of IV may be expressed as k-class estimation, available from `ivreg2`, and there are many other forms of IV models, including official Stata commands, such as `ivprobit`, `treatreg`, and `ivtobit`, and user-written additions, such as `qvf` (Hardin, Schmiediche, and Carroll 2003), `jive` (Poi 2006), and `ivpois` (on SSC).

4.3 Finding excluded instruments

The hard part of IV is finding a suitable Z matrix. The excluded instruments in Z have to be strongly correlated with the endogenous X^T and uncorrelated with the unobservable error e. However, the problem we want to solve is that the endogenous X^T is correlated with the unobservable error e. A good story is the crucial element in any plausible IV specification. We must believe that Z is strongly correlated with the endogenous X^T but has no direct impact on y (is uncorrelated with the unobservable error e), because the assumptions are not directly testable. However, the tests discussed in the following sections can help support a convincing story and should be reported anyways.

Generally, specification search in the first-stage regressions of X^T on some Z does not bias estimates or inference nor does using generated regressors. However, it is easy to produce counterexamples to this general rule. For example, taking $Z = X^T + \nu$, where ν is a small random error, will produce strong identification diagnostics—and might pass overidentification tests described in the next section—but will not improve estimates (and could lead to substantially less accurate inference).

If some Z are weak instruments, then regressing X^T on Z to get $\widehat{X}_T$ and using $\widehat{X}_T$ as the excluded instruments in an IV regression of y on X^T and X^C will likewise produce strong identification diagnostics but will not improve estimates or inference. Hall, Rudebusch, and Wilcox (1996) reported that choosing instruments based on measures of the strength of identification could actually increase bias and size distortions.

4.4 Exclusion restrictions in IV

The exclusion restrictions $E(Z'e) = 0$ cannot be directly tested, but if there are more excluded instruments than endogenous regressors, an overidentification (overid) test is feasible and the result should be reported. If there are exactly as many excluded instruments as endogenous regressors, the equation is *exactly identified*, and no overid test is feasible.

However, if Z is truly exogenous, it is likely also true that $E(W'e) = 0$, where W contains Z, squares, and cross products of Z. Thus there is always a feasible overid test by using an augmented set of excluded instruments, though $E(W'e) = 0$ is a stronger condition than $E(Z'e) = 0$. For example, if you have two good excluded instruments, you might multiply them together and square each to produce five excluded instruments. Testing the three extra overid restrictions is like Ramsey's regression specification-error (RESET) test of excluded instruments. Interactions of Z and X^C may also be good candidates for excluded instruments. For reasons discussed below, adding excluded instruments haphazardly is a bad idea, and with many weak instruments, limited-information maximum likelihood or continuously updated estimation is preferred to standard IV/2SLS.

Baum, Schaffer, and Stillman (2007) discuss the implementation of overid tests in `ivreg2` (see also `overid` from Baum et al. 2006). Passing the overid test (i.e., failing to reject the null of zero correlation) is neither necessary nor sufficient for instrument validity, $E(Z'e) = 0$, but rejecting the null in an overid test should lead you to reconsider your IV strategy and perhaps to look for different excluded instruments.

4.5 Tests of endogeneity

Even if we have an excluded instrument that satisfies $E(Z'e) = 0$, there is no guarantee that $E(X^{T\prime}\varepsilon) \neq 0$ as we have been assuming. If $E(X^{T\prime}\varepsilon) = 0$, we prefer ordinary regression to IV. Thus we should test the null that $E(X^{T\prime}\varepsilon) = 0$ (a test of endogeneity), though this test requires instrument validity, $E(Z'e) = 0$, so it should follow any feasible overid tests.

Baum, Schaffer, and Stillman (2007) describe several methods to test the endogeneity of a variable in X^T, including the `endog()` option of `ivreg2` and the standalone `ivendog` command (both available from SSC archive, with excellent help files). Section 4.2 also shows how the control function form of IV can be used to test endogeneity of a variable in X^T.

4.6 Identification and weak instruments

This is the second of the two crucial assumptions and presents problems of various sizes in almost all IV specifications. The extent to which $E(Z'X^T) \neq 0$ determines the strength of identification. Baum, Schaffer, and Stillman (2007) describe tests of identification, which amount to tests of the rank of $E(Z'X^T)$. These rank tests address the concern that a number of excluded instruments may generate exogenous variation in one endogenous variable and be uncorrelated with another endogenous variable, so the equation is not identified even though it satisfies the order condition (the number of excluded instruments is at least as great as the number of endogenous variables).

For example, if we have two endogenous variables X_1 and X_2 and three excluded instruments, all three excluded instruments may be correlated with X_1 and not with X_2. The identification tests look at the least partial correlation, or the minimum eigenvalue of the Cragg–Donald statistic (**?**), for example, and measures of whether at least one endogenous variable has no correlation with the excluded instruments.

Even if we reject the null of underidentification and conclude $E(Z'X^T) \neq 0$, we can still face a "weak-instruments" problem if some elements of $E(Z'X^T)$ are close to zero.

Even if we have an excluded instrument that satisfies $E(Z'e) = 0$, there is no guarantee that $E(Z'X^T) \neq 0$. The IV estimate is always biased but is less biased than OLS to the extent that identification is strong. In the limit of weak instruments, there would be no improvement over OLS for bias and the bias would be 100% of OLS. In the other limit, the bias would be 0% of the OLS bias (though this would require that the correlation between X^T and Z be perfect, which is impossible since X^T is endogenous and Z is exogenous). In applications, you would like to know where you are on that spectrum, even if only approximately.

There is also a distortion in the size of hypothesis tests. If you believe that you are incorrectly rejecting a null hypothesis about 5% of the time (i.e., you have chosen a size $\alpha = 0.05$), you may actually face a size of 10% or 20% or more.

Stock and Yogo (2005) reported rule-of-thumb critical values to measure the extent of both of these problems. Their table 1 shows the value of a statistic measuring the predictive power of the excluded instruments that will imply a limit of the bias to some percentage of OLS. For two endogenous variables and three excluded instruments ($n = 2$, $K_2 = 5$), the minimum value to limit the bias to 20% of OLS is 5.91. `ivreg2` reports these values as *Stock–Yogo weak ID test critical values*: one set for various percentages of "maximal IV relative bias" (largest bias relative to OLS) and one set for "maximal IV size" (the largest size of a nominal 5% test).

The key point is that all IV and IV-type specifications can suffer from bias and size distortions, not to mention inefficiency and sometimes failures of exclusion restrictions. The Stock and Yogo (2005) approach measures how strong identification is in your sample, and `ranktest` (Kleibergen and Schaffer 2007) offers a similar statistic for cases where errors are not assumed to be independently and identically distributed. Neither provides solutions in the event that weak instruments appear to be a problem. A further limitation is that these identification statistics only apply to the linear case, not the nonlinear analogs, including those estimated with generalized linear models. In practice, researchers should report the identification statistics for the closest linear analog; i.e., run `ivreg2` and report the output alongside the output from `ivprobit`, `ivpois`, etc.

If you suspect weak instruments may be producing large bias or size distortions, you have several options. You can find better excluded instruments, possibly by transforming your existing instruments. You can use limited-information maximum likelihood or continuously updated estimation, which are more robust to many weak instruments than standard IV. Perhaps best of all, you can conduct inference that is robust to weak instruments: with one endogenous variable, use `condivreg` (Mikusheva and Poi 2006), or with more than one, use tests described by Anderson and Rubin (1949) and Baum, Schaffer, and Stillman (2007, sec. 7.4 and 8).

4.7 Functional form tests in IV

As Baum, Schaffer, and Stillman (2007, sec. 9) and Wooldridge (2002, 125) discuss, the RESET test regressing residuals on predicted y and powers thereof is properly a test of a linearity assumption or a test of functional-form restrictions. `ivreset` performs the IV version of the test in Stata. A more informative specification check is the graphical version of RESET: predict $\widehat{X}_T$ after the first-stage regressions, compute forecasts $\widehat{y} = X^T \widehat{\beta}_T^{\mathrm{IV}} + X^C \widehat{\beta}_C$ and $\widehat{y}_f = \widehat{X}_T \widehat{\beta}_T^{\mathrm{IV}} + X^C \widehat{\beta}_C$, and graph a scatterplot of the residuals $\widehat{\varepsilon} = y - \widehat{y}$ against $\widehat{y}_f$. Any unmodeled nonlinearities may be apparent as a pattern in the scatterplot.

4.8 Standard errors in IV

The largest issue in IV estimation is often that the variance of the estimator is much larger than ordinary regression. Just as with ordinary regression, the SEs are asymptotically valid for inference under the restrictive assumptions that the disturbances are independently and identically distributed. Getting SEs robust to various violations of these assumptions is easily accomplished by using the `ivreg2` command (Baum, Schaffer, and Stillman 2007). Many other commands fitting IV models offer no equivalent robust SE estimates, but it may be possible to assess the size and direction of SE corrections by using the nearest linear analog in the spirit of using estimated design effects in the survey regression context.

4.9 Inference in IV

Assuming that we have computed consistent SEs and the best IV estimate we can by using a good set of Z and X^C variables, there remains the question of how we interpret the estimates and tests. Typically, IV identifies a particular LATE, namely the effect of an increase in X^T due to an increase in Z. If X^T were college and Z were an exogenous source of financial aid, then the IV estimate of the effect of X^T on wages would be the college wage premium for those who were induced to attend college by being eligible for the marginally more generous aid package.

Angrist and Krueger (1991) estimated the effect of education on earnings by using compulsory schooling laws as a justification for using quarter of birth dummies as instruments. Even if the critiques of Bound, Jaeger, and Baker (1995) did not apply, the identified effect would be for an increase in education due to being forced to remain in school a few months more. That is, the measured wage effect of another year of education is roughly for the eleventh grade and only for those who would have dropped out if not for compulsory schooling laws.

Sometimes a LATE of this form is exactly the estimate desired. If, however, we cannot reject that the IV estimate differs from the OLS estimate or the IV confidence region includes the OLS confidence region, we may not have improved estimates but merely produced noisier ones. Only where the IV estimate differs can we hope to ascertain the nature of selection bias.

4.10 Examples

We can use the data from Card (1995a,b) to estimate the impact of education on wages, where nearness to a college is used as a source of exogenous variation in educational attainment:

```
use http://fmwww.bc.edu/ec-p/data/wooldridge/card
local x "exper* smsa* south mar black reg662-reg669"
regress lw educ `x´
ivreg2 lw `x´ (educ=nearc2 nearc4), first endog(educ)
ivreg2 lw `x´ (educ=nearc2 nearc4), gmm
ivreg2 lw `x´ (educ=nearc2 nearc4), liml
```

The return to another year of education is found to be about 7% by using ordinary regression or 16% or 17% by using IV methods. The Sargan statistic fails to reject that excluded instruments are valid, the test of endogeneity is marginally significant (giving different results at the 95% and 90% levels), and the Anderson–Rubin and Stock–Wright tests of identification strongly reject that the model is underidentified.

The test for weak instruments is the F test on the excluded instruments in the first-stage regression, which at 7.49 with a p-value of 0.0006 seems to indicate that the excluded instruments influence educational attainment, but the size of Wald tests on `educ`, which we specify as 5%, might be roughly 25%. To construct an Anderson–Rubin confidence interval, we can type

```
generate y=.
foreach beta in .069 .0695 .07 .36 .365 .37 {
        quietly replace y=lw-`beta´*educ
        quietly regress y `x´ nearc2 nearc4
        display as res "Test of beta=" `beta´
        test nearc2 nearc4
}
```

This gives a confidence interval of (.07, .37); see Nichols (2006, 18) and Baum, Schaffer, and Stillman (2007, 30). Thus the IV confidence region includes the OLS estimate and nearly includes the OLS confidence interval, so the evidence on selection bias is weak. Still, if we accept the exclusion restrictions as valid, the evidence does not support a story where omitting ability (causing both increased wages and increased education) leads to positive bias. If anything, the bias seems likely to be negative, perhaps due to unobserved heterogeneity in discount rates or credit market failures. In the latter case, the omitted factor may be a social or economic disadvantage observable by lenders.

A similar set of conclusions apply if we model the education response as a binary treatment, college:

```
generate byte coll=educ>15
regress lw coll `x´
treatreg lw `x´, treat(coll=nearc2 nearc4)
ivreg2 lw `x´ (coll=nearc2 nearc4), first endog(coll)
ivreg2 lw `x´ (coll=nearc2 nearc4), gmm
ivreg2 lw `x´ (coll=nearc2 nearc4), liml
```

These regressions also indicate that the OLS estimate may be biased downward, but the OLS confidence interval is contained in the `treatreg` and IV confidence intervals. Thus we cannot conclude much with confidence.

5 RD designs

The idea of the RD design is to exploit an observable discontinuity in the level of treatment related to an assignment variable Z, so the level of treatment X^T jumps discontinuously at some value of Z, called the cutoff. Let Z_0 denote the cutoff. In the neighborhood of Z_0, under some often plausible assumptions, a discontinuous jump in the outcome y can be attributed to the change in the level of treatment. Near Z_0, the level of treatment can be treated as if it is randomly assigned. For this reason, the RD design is generally regarded as having the greatest internal validity of the quasiexperimental estimators.

Examples include share of votes received in a U.S. Congressional election by the Democratic candidate as Z, which induces a clear discontinuity in X^T, the probability of a Democrat occupying office the following term, and X^T may affect various outcomes y, if Democratic and Republican candidates actually differ in close races (Lee 2001). DiNardo and Lee (2002) use the share of votes received for a union as Z, and unions may affect the survival of a firm (but do not seem to). They point out that the union wage premium, y, can be consistently estimated only if survival is not affected (no differential attrition around Z_0), and they find negligibly small effects of unions on wages.

The standard treatment of RD is Hahn, Todd, and van der Klaauw (2001), who clarify the link to IV methods. Recent working papers by Imbens and Lemieux (2007) and McCrary (2007) focus on some important practical issues related to RD designs.

Many authors stress a distinction between "sharp" and "fuzzy" RD. In sharp RD designs, the level of treatment rises from zero to one at Z_0, as in the case where treatment is having a Democratic representative in the U.S. Congress or establishing a union, and a winning vote share defines Z_0. In fuzzy RD designs, the level of treatment increases discontinuously, or the probability of treatment increases discontinuously, but not from zero to one. Thus we may want to deflate by the increase in X^T at Z_0 in constructing our estimate of the causal impact of a one-unit change in X^T.

In sharp RD designs, the jump in y at Z_0 is the estimate of the causal impact of X^T. In a fuzzy RD design, the jump in y divided by the jump in X^T at Z_0 is the local Wald estimate (equivalent to a local IV estimate) of the causal impact. The local Wald estimate reduces to the jump in y at Z_0 in a sharp RD design as the jump in X^T is one, so the distinction between fuzzy and sharp RD is not that sharp. Some authors, e.g., Shadish, Cook, and Campbell (2002, 229), seem to characterize as fuzzy RD a wider class of problems, where the cutoff itself may not be sharply defined. However, without a true discontinuity, there can be no RD. The fuzziness in fuzzy RD arises only from probabilistic assignment of X^T in the neighborhood of Z_0.

5.1 Key assumptions and tests

The assumptions that allow us to infer a causal effect on y because of an abrupt change in X^T at Z_0 are the change in X^T at Z_0 is truly discontinuous, Z is observed without error (Lee and Card 2006), y is a continuous function of Z at Z_0 in the absence of treatment (for individuals), and that individuals are not sorted across Z_0 in their responsiveness to treatment. None of these assumptions can be directly tested, but there are diagnostic tests that should always be used.

The first is to test the null that no discontinuity in treatment occurs at Z_0, since without identifying a jump in X^T we will be unable to identify the causal impact of said jump. The second is to test that there are no other extraneous discontinuities in X^T or y away from Z_0, as this would call into question whether the functions would be smooth through Z_0 in the absence of treatment. The third and fourth test that predetermined characteristics and the density of Z exhibit no jump at Z_0, since these call into question the exchangeability of observations on either side of Z_0. Then the estimate itself usually supplies a test that the treatment effect is nonzero (y jumps at Z_0 because X^T jumps at Z_0).

Abusing notation somewhat so that Δ is an estimate of the discontinuous jump in a variable, we can enumerate these tests as

- (T1) $\Delta X^T(Z_0) \neq 0$
- (T2) $\Delta X^T(Z \neq Z_0) = 0$ and $\Delta y(Z \neq Z_0) = 0$
- (T3) $\Delta X^C(Z_0) = 0$
- (T4) $\Delta f(Z_0) = 0$
- (T5) $\Delta y(Z_0) \neq 0$ or $\left(\frac{\Delta y(Z_0)}{\Delta X^T(Z_0)}\right) \neq 0$

5.2 Methodological choices

Estimating the size of a discontinuous jump can be accomplished by comparing means in small bins of Z to the left and right of Z_0 or with a regression of various powers of Z, an indicator D for $Z > Z_0$, and interactions of all Z terms with D (estimating a polynomial in Z on both sides of Z_0, and comparing the intercepts at Z_0). However, since the goal is to compute an effect at precisely one point (Z_0) using only the closest observations, the standard approach is to use local linear regression, which minimizes bias (Fan and Gibels 1996). In Stata 10, this is done with the `lpoly` command; users of previous Stata versions can use `locpoly` (Gutierrez, Linhart, and Pitblado 2003).

Having chosen to use local linear regression, other key issues are the choice of bandwidth and kernel. Various techniques are available for choosing bandwidths (see e.g., Fan and Gibels 1996, Stone 1974, 1977), and the triangle kernel has good properties in the RD context, due to being boundary optimal (Cheng, Jianqing, and Marron 1997).

There are several rule-of-thumb bandwidth choosers and cross-validation techniques for automating bandwidth choice, but none is foolproof. McCrary (2007) contains a useful discussion of bandwidth choice and claims that there is no substitute for visual inspection comparing the local polynomial smooth with the pattern in a scatterplot. Because different bandwidth choices can produce different estimates, the researcher should report at least three estimates as an informal sensitivity test: one using the preferred bandwidth, one using twice the preferred bandwidth, and another using half the preferred bandwidth.

5.3 (T1) X^T jumps at Z_0

The identifying assumption is that X^T jumps at Z_0 because of some known legal or program-design rules, but we can test that assumption easily enough. The standard approach to computing SEs is to `bootstrap` the local linear regression, which requires wrapping the estimation in a program, for example,

```
program discont, rclass
     version 10
     syntax [varlist(min=2 max=2)] [, *]
     tokenize `varlist´
     tempvar z f0 f1
     quietly generate `z´=0 in 1
     local opt "at(`z´) nogr k(tri) deg(1) `options´"
     lpoly `1´ `2´ if `2´<0, gen(`f0´) `opt´
     lpoly `1´ `2´ if `2´>=0, gen(`f1´) `opt´
     return scalar d=`=`f1´[1]-`f0´[1]´
     display as txt "Estimate: " as res `f1´[1]-`f0´[1]
     ereturn clear
end
```

In the program, the assignment variable Z is assumed to be defined so that the cutoff $Z_0 = 0$ (easily done with one `replace` or `generate` command subtracting Z_0 from Z). The triangle kernel is used and the default bandwidth is chosen by `lpoly`, which is probably suboptimal for this application. The local linear regressions are computed twice: once using observations on one side of the cutoff for $Z < 0$ and once for $Z \geq 0$. The estimate of a jump uses only the predictions at the cutoff $Z_0 = 0$, so these are the only values computed by `lpoly`.

We can easily generate data to use this example program:

```
ssc install rd, replace
net get rd
use votex if i==1
rename lne y
rename win xt
rename d z
foreach v of varlist pop-vet {
     rename `v´ xc_`v´
}
bs: discont y z
```

In a more elaborate version of this program called `rd` (which also supports earlier versions of Stata), available by typing `ssc inst rd` in Stata, the default bandwidth is selected to include at least 30 observations in estimates at both sides of the boundary. Other options are also available. Try `findit bandwidth` to find more sophisticated bandwidth choosers for Stata. The key point is to use the `at()` option of `lpoly` so that the difference in local regression predictions can be computed at Z_0.

A slightly more elaborate version of this program would save local linear regression estimates at a number of points and offer a graph to assess fit:

```
program discont2, rclass
     version 10
     syntax [varlist(min=2 max=2)] [, s(str) Graph *]
     tokenize `varlist´
     tempvar z f0 f1 se0 se1 ub0 ub1 lb0 lb1
     summarize `2´, meanonly
     local N=round(100*(r(max)-r(min)))
     cap set obs `N´
     quietly generate `z´=(_n-1)/100 in 1/50
     quietly replace `z´=-(_n-50)/100 in 51/`N´
     local opt "at(`z´) nogr k(tri) deg(1) `options´"
     lpoly `1´ `2´ if `2´<0, gen(`f0´) se(`se0´) `opt´
     quietly replace `f0´=. if `z´>0
     quietly generate `ub0´=`f0´+1.96*`se0´
     quietly generate `lb0´=`f0´-1.96*`se0´
     lpoly `1´ `2´ if `2´>=0, gen(`f1´) se(`se1´) `opt´
     quietly replace `f1´=. if `z´<0
     quietly generate `ub1´=`f1´+1.96*`se1´
     quietly generate `lb1´=`f1´-1.96*`se1´
     return scalar d=`=`f1´[1]-`f0´[1]´
     return scalar f1=`=`f1´[1]´
     return scalar f0=`=`f0´[1]´
     forvalues i=1/50 {
          return scalar p`i´=`=`f1´[`i´]´
     }
     forvalues i=51/`N´ {
          return scalar n`=`i´-50´=`=`f0´[`i´]´
     }
     display as txt "Estimate: " as res `f1´[1]-`f0´[1]
     if "`graph´"!="" {
          label var `z´ "Assignment Variable"
          local lines "|| line `f0´ `f1´ `z´"
          local a "tw rarea `lb0´ `ub0´ `z´ || rarea `lb1´ `ub1´ `z´"
          `a´ || sc `1´ `2´, mc(gs14) leg(off) sort `lines´
     }
```

```
          if "`s´"!="" {
          rename `z´ `s´`2´
          rename `f0´ `s´`1´0
          rename `lb0´ `s´`1´lb0
          rename `ub0´ `s´`1´ub0
          rename `f1´ `s´`1´1
          rename `lb1´ `s´`1´lb1
          rename `ub1´ `s´`1´ub1
          }
         ereturn clear
end
```

In this version, the local linear regressions are computed at a number of points on either side of the cutoff Z_0 (in the example, the maximum of Z is assumed to be 0.5, so the program uses hundredths as a convenient unit for Z), but the estimate of a jump still uses only the two estimates at Z_0. The `s()` option in the above program saves the local linear regression predictions (and `lpoly` confidence intervals) to new variables that can then be graphed. Graphs of all output are advisable to assess the quality of the fit for each of several bandwidths. This program may also be bootstrapped, although recovering the standard errors around each point estimate from `bootstrap` for graphing the fit is much more work than using the output of `lpoly` as above.

5.4 (T2) y and X^C continuous away from Z_0

Although we need only assume continuity at Z_0 and need no assumption that the outcome and treatment variables are continuous at values of Z away from the cutoff Z_0 (i.e., $\Delta X^T(Z \neq Z_0) = 0$ and $\Delta y(Z \neq Z_0) = 0$), it is reassuring if we fail to reject the null of a zero jump at various values of Z away from the cutoff Z_0 (or reject the null only in 5% of cases or so). Having defined a program `discont`, we can easily randomly choose 100 placebo cutoff points $Z_p \neq Z_0$, without replacement in the example below, and test the continuity of X^T and y at each.

```
by z, sort: generate f=_n>1 if z!=0
generate u=uniform()
sort f u
replace u=(_n<=100)
levelsof z if u, loc(p)
foreach val of local p {
        capture drop newz
        generate newz=z-`val´
        bootstrap r(d), reps(100): discont y znew
        bootstrap r(d), reps(100): discont xt znew
}
```

5.5 (T3) X^C continuous around Z_0

If we can regard an increase in treatment X^T as randomly assigned in the neighborhood of the cutoff Z_0, then predetermined characteristics X^C such as race or sex of treated individuals should not exhibit a discontinuity at the cutoff Z_0. This is equivalent to the standard test of randomization in an experimental design, using a test of the equality of the mean of every variable in X^C across treatment and control groups (see `help hotelling` in Stata), or the logically equivalent test that all the coefficients on X^C in a regression of X^T on X^C are zero. As in the experimental setting, in practice the tests are usually done one at a time with no adjustment for multiple hypothesis testing (see `help _mtest` in Stata).

In the RD setting, this is simply a test that the measured jump in each predetermined X^C is zero at the cutoff Z_0 or $\Delta X^C(Z_0) = 0$ for all X^C. If we fail to reject that the measured jump in X^C is zero, for all X^C, we have more evidence that observations on both sides of the cutoff are exchangeable, at least in some neighborhood of the cutoff, and we can treat them as if they were randomly assigned treatment in that neighborhood.

Having defined the programs `discont` and `discont2`, we can simply type

```
foreach v of varlist xc* {
        bootstrap r(d), reps(100): discont `v´ z
        discont2 `v´ z, s(h)
        scatter `v´ z, mc(gs14) sort || line h`v´0 h`v´1 hz, name(`v´)
        drop hz
}
```

5.6 (T4) Density of Z continuous at cutoff

McCrary (2007) gives an excellent account of a violation of exchangability of observations around the cutoff. If individuals have preferences over treatment and can manipulate assignment, for instance by altering their Z or misreporting it, then individuals close to Z_0 may shift across the boundary. For example, some nonrandomly selected subpopulation of those who are nearly eligible for food stamps may misreport income, whereas those who are eligible do not. This creates a discontinuity in the density of Z at Z_0. McCrary (2007) points out that the absence of a discontinuity in the density of Z at Z_0 is neither necessary nor sufficient for exchangability. However, a failure to reject the null hypothesis, which indicates the jump in the density of Z at Z_0 is zero, is reassuring nonetheless.

McCrary (2007) discussed a test in detail and advocated a bandwidth chooser. We can also adapt our existing program to this purpose by using multiple `kdensity` commands to estimate the density to the left and right of Z_0:

```
kdensity z if z<0, gen(f0) at(z) tri nogr
count f0 if z>=0
replace f0=f0/r(N)*`=_N´/4
kdensity z if z>=0, gen(f1) at(z) tri nogr
count f1 if z<0
replace f1=f1/r(N)*`=_N´/4
generate f=cond(z>=0,f1,f0)
bootstrap r(d), reps(100): discont f z
discont2 f z, s(h) g
```

We could also wrap the `kdensity` estimation inside the program that estimates the jump, so that both are bootstrapped together; this approach is taken by the `rd` command available by typing `ssc inst rd`.

5.7 (T5) Treatment-effect estimator

Having defined the program `discont`, we can type

```
bootstrap r(d), reps(100): discont y z
```

to get an estimate of the treatment effect in a sharp RD setting, where X^T jumps from zero to one at Z_0. For a fuzzy RD design, we want to compute the jump in y scaled by the jump in X^T at Z_0, or the local Wald estimate, for which we need to modify our program to estimate both discontinuities. The program `rd` available by typing `ssc inst rd` does this, but the idea is illustrated in the program below by using the previously defined `discont` program twice.

```
program lwald, rclass
     version 10
     syntax varlist [, w(real .06) ]
     tokenize `varlist´
     display as txt "Numerator"
     discont `1´ `3´, bw(`w´)
     local n=r(d)
     return scalar numerator=`n´
     display as txt "Denominator"
     discont `2´ `3´, s(`sd´) bw(`w´)
     local d=r(d)
     return scalar denominator=`d´
     return scalar lwald=`n´/`d´
     display as txt "Local Wald Estimate:" as res `n´/`d´
     ereturn clear
end
```

This program takes three arguments—the variables y, X^T, and Z—assumes $Z_0 = 0$, and uses a hardwired default bandwidth of 0.06. The default bandwidth selected by `lpoly` is inappropriate for these models, because we do not use a Gaussian kernel and are interested in boundary estimates. The `rd` program from SSC archive is similar to the above; however, it offers more options—particularly with regard to bandwidth selection.

5.8 Examples

Voting examples abound. A novel estimate in Nichols and Rader (2007) measures the effect of electing as a Representative a Democratic incumbent versus a Republican incumbent on a district's receipt of federal grants:

```
ssc install rd
net get rd
use votex if i==1
rd lne d, gr
bs: rd lne d, x(pop-vet)
```

The above estimates that the marginally victorious Democratic incumbent brings 20% less to his home district than a marginally victorious Republican incumbent. However, we cannot reject the null of zero difference. This is true for a variety of bandwidth choices (figure 2 shows the small insignificant effect). The above is a sharp RD design, but the Wald estimator can be used to estimate effect, because the jump in `win` at 50% of vote share is one and dividing by one has no impact on estimates.

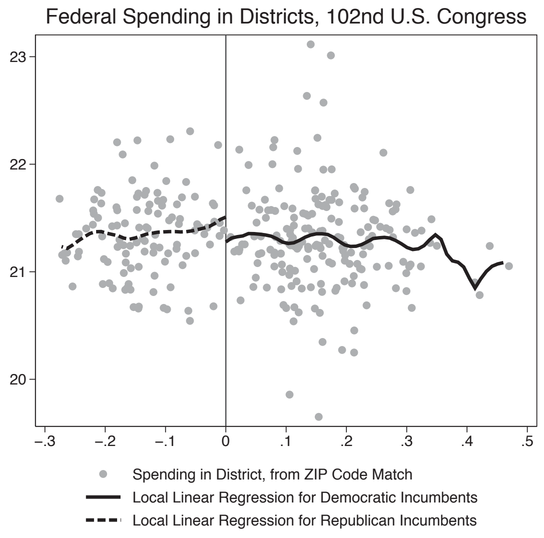

Figure 2: RD example

Many good examples of fuzzy RD designs concern educational policy or interventions (e.g., van der Klaauw 2002 or Ludwig and Miller 2005). Many educational grants are awarded by using deterministic functions of predetermined characteristics, lending themselves to evaluation using RD. For example, some U.S. Department of Education grants to states are awarded to districts with a poverty (or near-poverty) rate above a threshold, as determined by data from a prior Census, which satisfies all of the requirements for RD. The size of the discontinuity in funding may often be insufficient to identify an effect. Often a power analysis is warranted to determine the minimum detectable effect.

Returning to the Card (1995a,b) example of the effect of education on earnings, we can imagine exploiting a discontinuity in the availability of college to residents of certain U.S. states at the state boundary. College applicants who live 4.8 miles and 5 miles from a college may look similar in various observable characteristics, but if a state boundary separates them at 4.9 miles from the college, and the college is a state institution, they may face different probabilities of admission or tuition costs. The data in Card (1995a,b) do not support this strategy, of course, because we would need to know the exact locations of all individuals relative to state boundaries. However, it helps to clarify the assumptions that justify the IV approach. We need to assume that location relative to colleges is randomly sprinkled over potential applicants, which seems questionable (Black 1999), especially when one considers including parental education in the model.

6 Conclusions

Often exploring data using quasiexperimental methods is the only option for estimating a causal effect when experiments are infeasible, and may sometimes be preferred even when an experiment is feasible, particularly if a MTE is of interest. However, the methods can suffer several severe problems when assumptions are violated, even weakly. For this reason, the details of implementation are frequently crucial, and a kind of cookbook or checklist for verifying that essential assumptions are satisfied has been provided above for the interested researcher. As the topics discussed continue to be active research areas, this cookbook should be taken merely as a starting point for further explorations of the applied econometric literature on the relevant subjects.

7 References

Abadie, A., D. Drukker, J. Leber Herr, and G. W. Imbens. 2004. Implementing matching estimators for average treatment effects in Stata. *Stata Journal* 4: 290–311.

Abadie, A., and G. W. Imbens. 2006. On the failure of the bootstrap for matching estimators. NBER Technical Working Paper No. 325.
http://www.nber.org/papers/t0325/.

Anderson, T., and H. Rubin. 1949. Estimation of the parameters of a single equation in a complete system of stochastic equations. *Annals of Mathematical Statistics* 20: 46–63.

Angrist, J. D., G. W. Imbens, and D. B. Rubin. 1996. Identification of causal effects using instrumental variables. *Journal of the American Statistical Association* 91: 444–472.

Angrist, J. D., and A. B. Krueger. 1991. Does compulsory school attendance affect schooling and earnings? *Quarterly Journal of Economics* 106: 979–1014.

Autor, D. H., L. F. Katz, and M. S. Kerney. 2005. Rising wage inequality: The role of composition and prices. NBER Technical Working Paper No. 11628.
http://www.nber.org/papers/w11628/.

Azevedo, J. P. 2005. dfl: Stata module to estimate DiNardo, Fortin, and Lemieux counterfactual kernel density. Statistical Software Components S449001, Boston College Department of Economics. Downloadable from
http://ideas.repec.org/c/boc/bocode/s449001.html.

Baker, M., D. Benjamin, and S. Stanger. 1999. The highs and lows of the minimum wage effect: A time-series cross-section study of the Canadian law. *Journal of Labor Economics* 17: 318–350.

Baum, C. F. 2006. Time-series filtering techniques in Stata. Boston, MA: 5th North American Stata Users Group meetings.
http://www.stata.com/meeting/5nasug/TSFiltering_beamer.pdf.

Baum, C. F., M. Schaffer, and S. Stillman. 2007. Enhanced routines for IV/GMM estimation and testing. *Stata Journal* 7: 465–506.

Baum, C. F., M. Schaffer, S. Stillman, and V. Wiggins. 2006. overid: Stata module to calculate tests of overidentifying restrictions after ivreg, ivreg2, ivprobit, ivtobit, and reg3. Statistical Software Components S396802, Boston College Department of Economics. Downloadable from http://ideas.repec.org/c/boc/bocode/s396802.html.

Becker, S., and M. Caliendo. 2007. Sensitivity analysis for average treatment effects. *Stata Journal* 7: 71–83.

Becker, S. O., and A. Ichino. 2002. Estimation of average treatment effects based on propensity scores. *Stata Journal* 2: 358–377.

Black, S. 1999. Do better schools matter? Parental valuation of elmentary education. *Quarterly Journal of Economics* 114: 577–599.

Blinder, A. S. 1973. Wage discimination: Reduced form and structural estimates. *Journal of Human Resources* 8: 436–455.

Bound, J., D. Jaeger, and R. Baker. 1995. Problems with instrumental variable estimation when the correlation between the instruments and the endogenous explanatory variables is weak. *Journal of the American Statistical Association* 90: 443–450.

Card, D. E. 1995a. Using geographic variation in college proximity to estimate the return to schooling. In *Aspects of Labour Economics: Essays in Honour of John Vanderkamp*, ed. L. Christofides, E. K. Grant, and R. Swindinsky. Toronto, Canada: University of Toronto Press.

———. 1995b. Earnings, schooling, and ability revisited. *Research in Labor Economics* 14: 23–48.

———. 1999. The causal effect of education on earnings. *Handbook of Labor Economics* 3: 1761–1800.

———. 2001. Estimating the return to schooling: Progress on some persistent econometric problems. *Econometrica* 69: 1127–1160.

Cheng, M., F. Jianqing, and J. S. Marron. 1997. On automatic boundary corrections. *Annals of Statistics* 25: 1691–1708.

Cochran, W., and D. B. Rubin. 1973. Controlling bias in observational studies. *Sankhyā* 35: 417–446.

DiNardo, J. 2002. Propensity score reweighting and changes in wage distributions. Working Paper, University of Michigan.
http://www-personal.umich.edu/~jdinardo/bztalk5.pdf.

DiNardo, J., N. M. Fortin, and T. Lemieux. 1996. Labor market institutions and the distribution of wages, 1973–1992: A semiparametric approach. *Econometrica* 64: 1001–1044.

DiNardo, J., and D. Lee. 2002. The impact of unionization on establishment closure: A regression discontinuity analysis of representation elections. NBER Technical Working Paper No. 8993. http://www.nber.org/papers/w8993/.

DiPrete, T., and M. Gangl. 2004. Assessing bias in the estimation of causal effects: Rosenbaum bounds on matching estimators and instrumental variables estimation with imperfect instruments. *Sociological Methodology* 34: 271–310.

Fan, J., and I. Gibels. 1996. *Local Polynomial Modelling and Its Applications*. New York: Chapman & Hall.

Fisher, R. A. 1918. The causes of human variability. *Eugenics Review* 10: 213–220.

———. 1925. *Statistical Methods for Research Workers*. Edinburgh: Oliver & Boyd.

Glazerman, S., D. M. Levy, and D. Myers. 2003. Nonexperimental versus experimental estimates of earnings impacts. *Annals of the American Academy of Political and Social Science* 589: 63–93.

Goldberger, A. S., and O. D. Duncan. 1973. *Structural Equation Models in the Social Sciences*. New York: Seminar Press.

Gomulka, J., and N. Stern. 1990. The employment of married women in the United Kingdom, 1970–1983. *Econometrica* 57: 171–199.

Griliches, Z., and J. A. Hausman. 1986. Errors in variables in panel data. *Journal of Econometrics* 31: 93–118.

Gutierrez, R. G., J. M. Linhart, and J. S. Pitblado. 2003. From the help desk: Local polynomial regression and Stata plugins. *Stata Journal* 3: 412–419.

Hahn, J., P. Todd, and W. van der Klaauw. 2001. Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica* 69: 201–209.

Hall, A. R., G. D. Rudebusch, and D. W. Wilcox. 1996. Judging instrument relevance in instrumental variables estimation. *International Economic Review* 37: 283–298.

Hardin, J. W., H. Schmiediche, and R. J. Carroll. 2003. Instrumental variables, bootstrapping, and generalized linear models. *Stata Journal* 3: 351–360.

Heckman, J., H. Ichimura, and P. Todd. 1997. Matching as an econometric evaluation estimator: Evidence from evaluating a job training program. *Review of Economic Studies* 64: 605–654.

Heckman, J. J., and E. Vytlacil. 2004. Structural equations, treatment effects, and econometric policy evaluation. *Econometrica* 73: 669–738.

Holland, P. W. 1986. Statistics and causal inference. *Journal of the American Statistical Association* 8: 945–960.

Imai, K., and D. A. van Dyk. 2004. Causal inference with general treatment regimes: Generalizing the propensity score. *Journal of the American Statistical Association* 99: 854–866.

Imbens, G. 2004. Nonparametric estimation of average treatment effects under exogeneity: A review. *Review of Economics and Statistics* 86: 4–29.

Imbens, G. W., and T. Lemieux. 2007. Regression discontinuity designs: A guide to Practice. NBER Technical Working Paper No. 13039. http://www.nber.org/papers/w13039/.

Jann, B. 2005a. jmpierce: Stata module to perform Juhn–Murphy–Pierce decomposition. Statistical Software Components S448803, Boston College Department of Economics. Downloadable from http://ideas.repec.org/c/boc/bocode/s448803.html.

———. 2005b. oaxaca: Stata module to compute decompositions of outcome differentials. Statistical Software Components S450604, Boston College Department of Economics. Downloadable from http://ideas.repec.org/c/boc/bocode/s450604.html.

Juhn, C., K. M. Murphy, and B. Pierce. 1991. Accounting for the slowdown in black–white wage convergence. In *Workers and Their Wages: Changing Patterns in the United States*, ed. M. Kosters, 107–143. Washington, DC: American Enterprise Institute.

———. 1993. Wage inequality and the rise in returns to skill. *Journal of Political Economy* 101: 410–442.

Kleibergen, F., and M. Schaffer. 2007. ranktest: Stata module to test the rank of a matrix using the Kleibergen–Paap rk statistic. Boston College Department of Economics, Statistical Software Components S456865. Downloadable from http://ideas.repec.org/c/boc/bocode/s456865.html.

Lee, D. S. 2001. The electoral advantage to incumbency and voters' valuation of politicians' experience: A regression discontinuity analysis of elections to the U.S. House. NBER Technical Working Paper No. 8441. http://www.nber.org/papers/w8441/.

———. 2005. Training, wages, and sample selection: Estimating sharp bounds on treatment effects. NBER Technical Working Paper No. 11721. http://www.nber.org/papers/w11721/.

Lee, D. S., and D. Card. 2006. Regression discontinuity inference with specification error. NBER Technical Working Paper No. 322. http://www.nber.org/papers/t0322/.

Leibbrandt, M., J. Levinsohn, and J. McCrary. 2005. Incomes in South Africa since the fall of apartheid. NBER Technical Working Paper No. 11384. http://www.nber.org/papers/w11384/.

Leuven, E., and B. Sianesi. 2003. psmatch2: Stata module to perform full Mahalanobis and propensity score matching, common support graphing, and covariate imbalance testing. Boston College Department of Economics, Statistical Software Components. Downloadable from http://ideas.repec.org/c/boc/bocode/s432001.html.

Ludwig, J., and D. L. Miller. 2005. Does head start improve children's life chances? Evidence from a regression discontinuity design. NBER Technical Working Paper No. 11702. http://www.nber.org/papers/w11702/.

Machado, J., and J. Mata. 2005. Counterfactual decompositions of changes in wage distributions using quantile regression. *Journal of Applied Econometrics* 20: 445–465.

Manski, C. 1995. *Identification Problems in the Social Sciences*. Cambridge, MA: Harvard University Press.

McCrary, J. 2007. Manipulation of the running variable in the regression discontinuity design: A density test. NBER Technical Working Paper No. 334. http://www.nber.org/papers/t0334/.

Mikusheva, A., and B. P. Poi. 2006. Tests and confidence sets with correct size when instruments are potentially weak. *Stata Journal* 6: 335–347.

Morgan, S. L., and D. J. Harding. 2006. Matching estimators of causal effects: Prospects and pitfalls in theory and practice. *Sociological Methods and Research* 35: 3–60.

Nannicini, T. 2006. sensatt: A simulation-based sensitivity analysis for matching estimators. Boston College Department of Economics, Statistical Software Components. Downloadable from http://ideas.repec.org/c/boc/bocode/s456747.html.

Nelson, C., and R. Startz. 1990. Some further results on the exact small sample properties of the instrumental variable estimator. *Econometrica* 58: 967–976.

Neyman, J. 1923. *Roczniki Nauk Roiniczych* (Annals of Agricultural Sciences) Tom X: 1–51 [In Polish]. Translated as "On the application of probability theory to agricultural experiments. Essay on principles. Section 9," by D. M. Dabrowska and T. P. Speed (Statistical Science 5: 465–472, 1990).

Nichols, A. 2006. Weak instruments: An overview and new techniques. Boston, MA: 5th North American Stata Users Group meetings. http://www.stata.com/meeting/5nasug/wiv.pdf.

Nichols, A., and K. Rader. 2007. Spending in the districts of marginal incumbent victors in the House of Representatives. Unpublished working paper.

Nichols, A., and M. E. Schaffer. 2007. Cluster–robust and GLS corrections. Unpublished working paper.

Oaxaca, R. 1973. Male–female wage differentials in urban labor markets. *International Economic Review* 14: 693–709.

Poi, B. P. 2006. Jackknife instrumental variables estimation in Stata. *Stata Journal* 6: 364–376.

Rosenbaum, P. R. 2002. *Observational Studies*. 2nd ed. New York: Springer.

Rosenbaum, P. R., and D. B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55.

Rothstein, J. 2007. Do value-added models add value? Tracking fixed effects and causal inference. Unpublished working paper.

Rubin, D. B. 1974. Estimating causal effects of treatments in randomised and non-randomised studies. *Journal of Educational Psychology* 66: 688–701.

———. 1986. Statistics and causal inference: Comment: Which ifs have causal answers. *Journal of the American Statistical Association* 81: 961–962.

———. 1990. Comment: Neyman (1923) and causal inference in experiments and observational studies. *Statistical Science* 5: 472–480.

Schaffer, M., and S. Stillman. 2006. xtoverid: Stata module to calculate tests of overidentifying restrictions after xtreg, xtivreg, xtivreg2, and xthtaylor. Statistical Software Components S456779, Boston College Department of Economics. Downloadable from http://ideas.repec.org/c/boc/bocode/s456779.html.

———. 2007. xtivreg2: Stata module to perform extended IV/2SLS, GMM and AC/HAC, LIML, and *k*-class regression for panel-data models. Statistical Software Components S456501, Boston College Department of Economics. Downloadable from http://ideas.repec.org/c/boc/bocode/s456501.html.

Shadish, W. R., T. D. Cook, and D. T. Campbell. 2002. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Boston: Houghton Mifflin.

Simpson, E. H. 1951. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B* 13: 238–241.

Spence, M. 1973. Job market signaling. *Quarterly Journal of Economics* 87: 355–374.

Stock, J. H., and M. Yogo. 2005. Testing for weak instruments in linear IV regression. In *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, ed. D. W. K. Andrews and J. H. Stock, 80–108. Cambridge: Cambridge University Press.

Stone, M. 1974. Cross-validation and multinomial prediction. *Biometrika* 61: 509–515.

———. 1977. Asymptotics for and against cross-validation. *Biometrika* 64: 29–35.

Stuart, E. A., and D. B. Rubin. 2007. Best practices in quasiexperimental designs: Matching methods for causal inference. In *Best Practices in Quantitative Social Science*, ed. J. Osborne. Thousand Oaks, CA: Sage.

van der Klaauw, W. 2002. Estimating the effect of financial aid offers on college enrollment: A regression discontinuity approach. *International Economic Review* 43: 1249–1287.

Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Yule, G. U. 1903. Notes on the theory of association of attributes in statistics. *Biometrika* 2: 275–280.

Yun, M.-S. 2004. Decomposing differences in the first moment. *Economics Letters* 82: 275–280.

———. 2005a. Normalized equation and decomposition analysis: Computation and inference. IZA Discussion Paper No. 1822. ftp://ftp.iza.org/dps/dp1822.pdf.

———. 2005b. A simple solution to the identification problem in detailed wage decompositions. *Economic Inquiry* 43: 766–772.

About the author

Austin Nichols is an economist at the Urban Institute, a nonprofit, nonpartisan think tank. He occasionally teaches statistics and econometrics, and he has used Stata almost daily since 1995. His research interests include poverty, social insurance, tax policy, and demographic outcomes such as fertility, marital status, health, and education.

The Stata Journal (2007)
7, Number 4, pp. 542–555

A generic function evaluator implemented in Mata

Henrik Støvring
Research Unit for General Practice
University of Southern Denmark
Odense, Denmark
hstovring@health.sdu.dk

Abstract.

Background: When implementing new statistical procedures, there is often a need for simple—and yet computationally efficient—ways of numerically evaluating composite distribution functions. If the statistical procedure must support calculations for censored and noncensored cases, those calculations should be carried out using efficient computational implementations of both definite and indefinite integrals (e.g., calculation of tail areas of distribution functions).

Method: We developed a generic function evaluator such that users may specify a function using reverse Polish notation. As its argument the function evaluator takes a matrix of pointers and then applies the rows of this matrix to its internally defined stack of pointers. Accordingly, each row of the argument matrix defines a single operation such as evaluating a function on the current element of the stack, applying an algebraic operation to the two top elements of the stack, or manipulating the stack itself. Defining new composite distribution functions from other (atomic) distribution functions then corresponds to joining two or more function-defining matrices vertically. This approach can further be used to obtain integrals of any defined function. As an example we show how the density and distribution function for the minimum of two Weibull distributed random variables can be numerically evaluated and integrated.

Results: The procedure provides a flexible and extensible framework for implementing numerical evaluation of general, composite distributions. The procedure is numerically relatively efficient, although not optimal.

Keywords: pr0034, rpnfcn(), RPN, Mata

1 Introduction

In applied statistics there is often a need for numerically evaluating probability distributions that are composite in the sense that they arise from two or more random variables, each with their own particular distribution function.

One example is a two-component mixture with the following generic form for its cumulative distribution function (cdf)

$$F(t;p,\theta_1,\theta_2) = p\,F_1(t;\theta_1) + (1-p)F_2(t;\theta_2)$$

where F_1 and F_2 are two cdfs, p is a probability, and θ_1 and θ_2 are parameter vectors.

A second example is the probability density function (pdf) of $Y = \min(X, Z)$, where X and Z are two independent, nonnegative random variables with pdfs f_X and f_Z, respectively, and associated survivor functions S_X and S_Z. It is well known (see, for example, Degroot 1986, 160) that here the pdf for Y is given by (with obvious notation)

$$f_Y(t;\theta_1,\theta_2) = f_X(t;\theta_1)S_Z(t;\theta_2) + S_X(t;\theta_1)f_Z(t;\theta_2) \tag{1}$$

and survivor function given by

$$S_Y(t;\theta_1,\theta_2) = S_X(t;\theta_1)S_Z(t;\theta_2) \tag{2}$$

A third example where the problem becomes more compounded is the so-called forward recurrence distribution given by the following generic form

$$f_R(t;\theta) = \frac{S_T(t;\theta)}{\mu_T(\theta)} \tag{3}$$

where $\mu_T(\theta)$ is the mean of T, a nonnegative random variable with survivor function S_T (Støvring and Vach 2005). Because T is nonnegative, its mean can be written as

$$\mu = \int_0^\infty S_T(s;\theta)\,ds$$

Often no analytical expressions for such integrals of S_T exist, and so numerical evaluation must be applied to obtain this mean, as well as the cdf of R, which is given by

$$F_R(t;\theta) = \mu_T^{-1}(\theta)\int_0^t S_T(s;\theta)ds$$

One straightforward method of implementing these functions as ordinary Mata functions would be to code a new function that first invokes the subfunctions it is made of, and then combines results of these function calls appropriately. This nesting approach implies that we would need to pass the vector of all parameter values, (p,θ_1,θ_2), say, to the new function F, which then must parse them and pass them on to the relevant functions, F_1 and F_2. Although this is possible, it becomes tedious when F_1 or F_2 are themselves composite functions or if we want to allow for different functional shapes of F_1 and F_2. Different shapes effectively require writing a new definition of F for each shape with dedicated and appropriate parsing of relevant parameters. Further, when we need to integrate the new composite function, it is difficult to implement a generic integration function, which can easily accommodate the different structures of the integrand and the varying number of arguments needed. Again, although possible, so far no generic routines exist for solving the problem generally, as far as the author is aware.

To overcome this we suggest and implement a new function evaluator based on reverse Polish notation (RPN). The motivation for using RPN is that it avoids nesting of functions and instead naturally specifies computations sequentially. This has the advantage of allowing function definitions to be framed as matrices of pointers, where each row defines one computational step in the RPN algorithm. These definition matrices can be straightforwardly stacked to construct new composite functions, thus facilitating implementation of complicated pdfs and cdfs. Finally, the strategy allows a general implementation of integration as this can be obtained from sandwiching any integrand between two general, special purpose functions defining the points of evaluation and summarizing the integrand at the evaluated points.

The paper is organized as follows: section 2.1 introduces the principle of RPN before section 2.2 describes how it is implemented in Mata. In sections 2.3 and 2.4, I show how the procedure allows for general numerical integration, both for definite and indefinite integrals. To show the potential of the procedure, section 2.5 gives an example on evaluation of a forward recurrence distribution arising from a random variable, which itself is the minimum of two Weibull distributed random variables. Finally, I discuss the primary advantages and limitations of the suggested procedure in section 3.

2 Methods

2.1 The principle of RPN

Conceptually, RPN is based on first defining two operands and then applying a binary operator (addition, multiplication, etc.) to obtain the result. Although the notation for this principle is less intuitive than the standard infix notation, its main advantage is obliviating the need for parentheses and equation signs; i.e., it essentially specifies computations in one (long) serial sequence instead of the intuitively appealing nesting of the usual notation with parentheses. Anyone with experience in using an RPN calculator has recognized that RPN involves fewer button pushes and allows one to compute almost straight from left to right of any numerical expression, although it needs a slightly different organization of computations than a traditional calculator.

The key step in implementing RPN is to setup an indexed stack with elements containing operands and results. Then one must define functions for manipulating stack elements. These functions roughly fall into three categories: stack manipulation, binary operators, and single element functions. Consider for example the steps involved in the computation of the exponential density given by

$$f(t;\lambda) = \lambda \exp[-\lambda t]$$

With RPN, the computation would be given by the algorithm presented in figure 1.

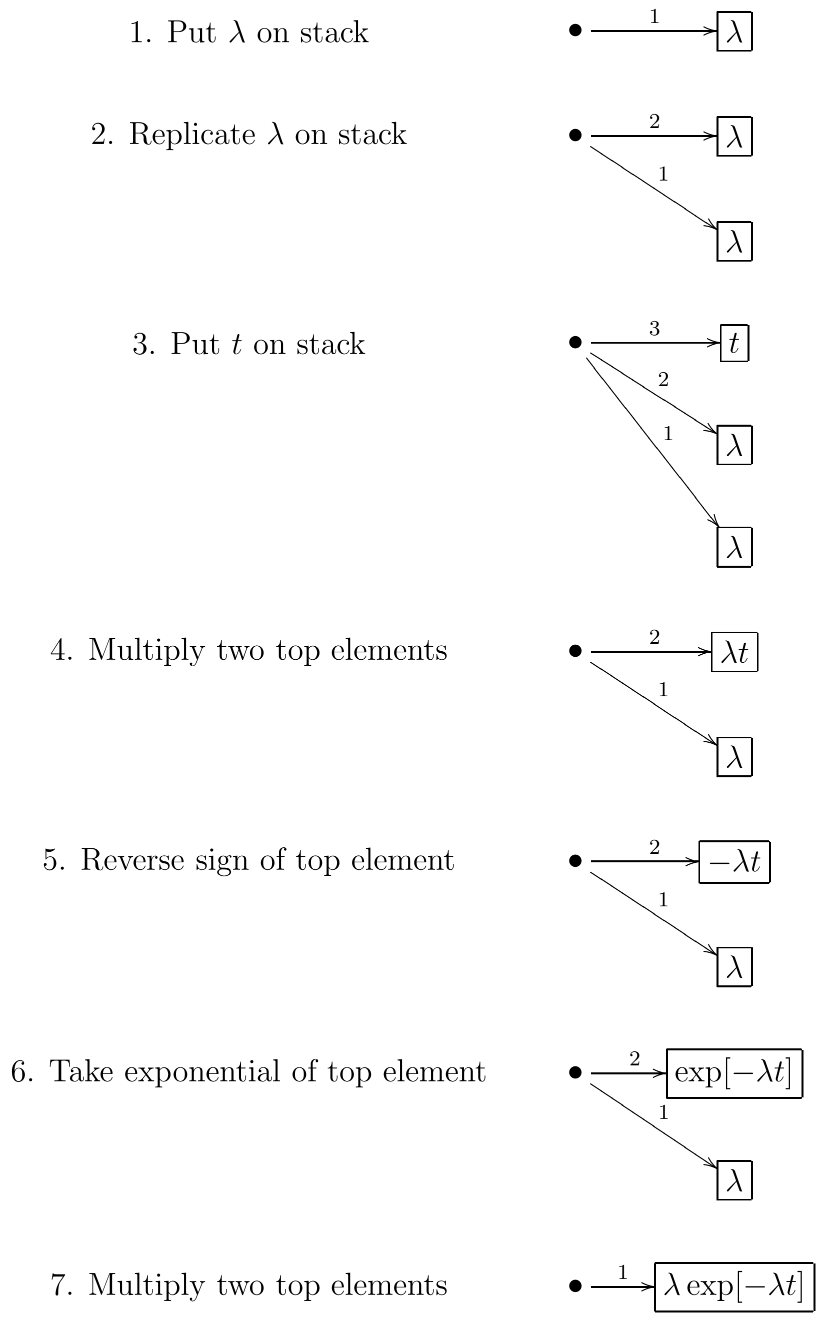
1. Put λ on stack
1
λ
2. Replicate λ on stack
2
1
λ
λ
3. Put t on stack
3
2
1
t
λ
λ
4. Multiply two top elements
2
1
λt
λ
5. Reverse sign of top element
2
1
$-\lambda t$
λ
6. Take exponential of top element
2
1
$\exp[-\lambda t]$
λ
7. Multiply two top elements
1
$\lambda \exp[-\lambda t]$

Figure 1: The steps involved in computing the exponential pdf using RPN. The • with arrows denotes the stack, which is nothing more than an indexed set of pointers, 1, 2, . . . , here denoted by arrows, and the referenced boxes contain the elements of the stack after the indicated action. Operations on elements of the stack place the result of the operation on the stack.

Steps 1–3 are examples of stack manipulation actions: steps 1 and 3 put arguments λ and t, respectively, into the stack, whereas step 2 copies the current top element to a new top element of the stack. Steps 4 and 7 apply the binary operator of multiplication to the two top elements of the stack. Steps 5 and 6 apply a single element function to the top element of the stack and returns the result in the same place.

2.2 Implementing RPN for matrices in Mata

The implementation of an RPN function evaluator in Mata consists of two main ingredients. The first is a function, `rpnfcn()`, which sets up the stack and sequentially applies the individual steps of any algorithm defined in a matrix of pointers in which the rows each define one step in the algorithm. Internally, `rpnfcn()` relies heavily on a helper function, `anyeqpt()`, for determining the type of action to take in each step. The code of `rpnfcn()` and `anyeqpt()` are included in the `rpn` package. The second main ingredient consists of various stack functions of the three types described above. A short description of the basic implemented stack functions is given in table 1. The implemented pdfs and cdfs are only examples and more should be implemented whenever need arises.

The actual parameterizations chosen for the densities are described by the following expressions for the associated survivor functions:

$$\begin{aligned} \text{Exponential:} \quad & S(t;\theta=\beta) = \exp[-e^{\beta}t] \\ \text{Weibull:} \quad & S\{t;\theta=(\beta,\alpha)\} = \exp\left[-(e^{\beta}t)^{\exp(\alpha)}\right] \end{aligned} \tag{4}$$

where Φ is the standard normal cdf. The parameterization is chosen so as to avoid restricting parameters to be positive, which is useful in avoiding boundary problems in estimation procedures.

Table 1: Implemented stack functions. Argument types are internal elements of the stack (*Int*) or external arguments (*Ext*) supplied by a pointer in the controlling algorithm matrix.

Type	Name	i	Arguments *Int*	Arguments *Ext*	Description
Manipulation	`tostack`	+1		x	Puts x onto stack
	`enter`	+1	e_i		Copies e_i to new top element e_{i+1}
	`swapst`	0	e_i, e_{i-1}		Swap content of e_i and e_{i-1}
	`rotst`	0	$e_1, e_2, \ldots, e_i$		Rotate stack such that e_1 is new top element and all other elements are pushed down one step
Binary operators[a]	`add`	−1	e_i, e_{i-1}		Add e_i and e_{i-1}
	`subtract`	−1	e_i, e_{i-1}		Subtract e_i from e_{i-1}
	`product`	−1	e_i, e_{i-1}		Multiply e_i and e_{i-1}
	`divide`	−1	e_i, e_{i-1}		Divide e_{i-1} with e_i
Single element functions	`fexp`	0	e_i	θ	Exponential pdf
	`Sexp`	0	e_i	θ	Exponential survivor function
	`fwei`	0	e_i	θ	Weibull pdf
	`Swei`	0	e_i	θ	Weibull survivor function
Integration	`MCatnode`	0	e_i	(a, b)	Setup nodes for stratified, antithetic Monte Carlo integration
	`MCatwt`	0	e_i	(a, b)	Setup weights for stratified, antithetic Monte Carlo integration
	`tlMCatnode`	0	e_i	(a)	Setup nodes for stratified, antithetic Monte Carlo indefinite integration
	`tlMCatwt`	0	e_i	(a)	Setup weights for stratified, antithetic Monte Carlo indefinite integration
	`intres`	−1	e_i, e_{i-1}		Row sum of e_{i-1} multiplied elementwise with e_i

[a] All binary operators act elementwise using the Mata colon operators `:+`, `:-`, `:*`, and `:/`, respectively.

Example on basic use

To illustrate how to apply the `rpnfcn()` function, consider computation of the density for the minimum of two Weibull distributed random variables, i.e., the density given in (1) with the Weibull parameterization of (4). The algorithm matrix for this density takes the following form in Mata syntax:

```
. mata
─────────────────────────────────────────────────── mata (type end to exit) ─────
: x = (NULL)

: theta1 = (NULL)

: theta2 = (NULL)

: fweimin = (&tostack(),  &x \
>                 &enter(),    NULL \
>                 &enter(),    NULL \
>                 &fwei(),     &theta1 \
>                 &swapst(),   NULL \
>                 &Swei(),     &theta2 \
>                 &product(),  NULL \
>                 &rotst(),    NULL \
>                 &enter(),    NULL \
>                 &Swei(),     &theta1 \
>                 &swapst(),   NULL \
>                 &fwei(),     &theta2 \
>                 &product(),  NULL \
>                 &add(),      NULL
>                 )

: end
─────────────────────────────────────────────────────────────────────────────────
```

We defined `x`, `theta1`, and `theta2` to be empty matrices as they must exist before defining the algorithm matrix. Their contents can, however, be set after defining the algorithm matrix. The following Mata code fills the matrices and evaluates the function defined in `fweimin`:

```
. mata
─────────────────────────────────────────────────── mata (type end to exit) ─────
: x = (0, 2, 8 \
>      .7, 1, 6)

: theta1 = (-2, 2)

: theta2 = (0, .3)

: rpnfcn(fweimin)
                  1              2              3
    ┌──────────────────────────────────────────────┐
  1 │             0    .1344860648    4.74373e-08  │
  2 │   .6423218484    .4965861523    .0000298698  │
    └──────────────────────────────────────────────┘

: end
─────────────────────────────────────────────────────────────────────────────────
```

2.3 Definite integration with RPN

Suppose that we want to evaluate

$$H(a,b) = \int_a^b h(s)ds$$

Conceptually, numerical integration can be divided into three steps: first, define the points at which to evaluate h, the so-called nodes; second, apply h to these nodes; and third, sum the weighted results of this evaluation, where the weights depend on the choice of nodes. For an example of how to integrate a function with stratified, antithetic Monte Carlo integration, see Ripley (1987, 131–132). Formally, the computation to be done is given by

$$\int_a^b f(s)ds \approx \frac{b-a}{2M} \sum_{j=0}^{M-1} \Big[f\left\{a + \delta(j+u_j)\right\} + f\left\{a + \delta(j+1-u_j)\right\} \Big]$$

where $\delta = (b-a)/M$ and all u_j are uniformly distributed and mutually independent. The nodes here are thus

$$a + \delta(j+u_j) \text{ and } a + \delta(j+1-u_j)$$

all with weights $(b-a)/(2M)$.

Following this, three functions have been implemented to setup a matrix containing the nodes (`MCatnode`), setup a matrix containing the weights (`MCatwt`), and compute the integration result by summing the weighted evaluation results (`intres`), respectively. All three functions are generic in the sense that they are completely independent of the evaluator to be used: `MCatnode` replaces the matrix in the current top element of the stack with a matrix of the relevant nodes. The function to be integrated is then applied to the nodes leaving the result in the top element of the stack, and finally, the weights are computed and the summation function is applied, yielding the result matrix. For algorithm matrix notation, this means sandwiching the rows for the function to be integrated between rows concerning `MCatnode` initially, and—at the bottom—`MCatwt` and `intres`. For this to work, the function to be evaluated must only rely on the single matrix of nodes present in the stack after execution of `MCatnode`.

Example on definite integration

Consider integration of the density for the minimum of two Weibull distributed random variables, `fweimin` defined above. Let the matrix `ab` be an $n \times 2$ matrix, where each row contains integration limits, such that the first column is the lower limit and the second column is the upper limit. Assume that `u` is an $n \times m$ matrix containing uniformly distributed random numbers. For example, the matrices could be given by

```
. set seed 9876
. mata
------------------------------------------------- mata (type end to exit) -----
: ab = (0, 4 \
>       0, 2 \
>       2, 4 \
>       4, 10)
: u = uniform(rows(ab), 30)
: end
-------------------------------------------------------------------------------
```

The algorithm matrix for the integration can now be defined, and the integration performed as follows:

```
. mata
------------------------------------------------- mata (type end to exit) -----
: intfweimin = (&tostack(), &u \
>               &MCatnode(), &ab ) \
>               fweimin[2..rows(fweimin), .] \
>              (&tostack(), &u \
>               &MCatwt(), &ab \
>               &intres(), NULL)
: rpnfcn(intfweimin)
                 1
    +---------------+
  1 |  .9972108522  |
  2 |   .921117737  |
  3 |  .0766576168  |
  4 |  .0014903807  |
    +---------------+

:
: end
-------------------------------------------------------------------------------
```

2.4 Indefinite integration with RPN

Suppose that we want to evaluate

$$S(a) = \int_a^\infty h(s)ds$$

One way to do this numerically is to apply the transformation $s \mapsto e^{-s}$ so that the integral becomes

$$S(a) = \int_0^{\exp(-a)} \frac{h(-\log u)}{u} du$$

Because the integral is now definite, it can be evaluated as above, except that the weights should now take into account the term u^{-1}. For stratified, antithetic sampling the nodes become

$$a - \log\left(\frac{j+u_j}{M}\right) \quad \text{and} \quad a - \log\left(\frac{j+1-u_j}{M}\right)$$

for $j = 0, 1, \ldots, M-1$. The corresponding weights are

$$\frac{1}{2(j+u_j)} \quad \text{and} \quad \frac{1}{2(j+1-u_j)}$$

Example on indefinite integration

Consider tail integration of the density for the minimum of two Weibull distributed random variables, `fweimin` defined above. Let the matrix `a` be an $n \times 1$ vector, where each element is a lower integration limit. Assume that `u` is an $n \times m$ matrix containing uniformly distributed random numbers. For example, the matrices could be given by

```
. set seed 4757
. mata
──────────────────────────────────────────────── mata (type end to exit) ─────
: a = (0\
>      2\
>      4)
: u = uniform(rows(a), 20)
: end
──────────────────────────────────────────────────────────────────────────────
```

The algorithm matrix for the integration can now be defined, and the integration performed as follows:

```
. mata
──────────────────────────────────────────────── mata (type end to exit) ─────
: Sweimin = (&tostack(), &u \
>             &tlMCatnode(), &a ) \
>            fweimin[2..rows(fweimin), .] \
>            (&tostack(), &u \
>             &tlMCatwt(), &a \
>             &intres(), NULL)
: rpnfcn(Sweimin)
                 1
    ┌──────────────┐
  1 │  .9986485593 │
  2 │  .0781537442 │
  3 │  .0014923006 │
    └──────────────┘
:
: end
──────────────────────────────────────────────────────────────────────────────
```

Here it is also possible to evaluate the integral explicitly by using the formula in (2), and so we can validate the results above:

```
. mata
──────────────────────────────────────────────── mata (type end to exit) ───
: Sweimin_exact = (&tostack(), &a \
>                  &enter(), NULL \
>                  &Swei(), &theta1 \
>                  &swapst(), NULL \
>                  &Swei(), &theta2 \
>                  &product(), NULL)

: rpnfcn(Sweimin_exact)
                  1
    ┌───────────────┐
  1 │             1 │
  2 │   .0781648043 │
  3 │   .0014922392 │
    └───────────────┘

: end
────────────────────────────────────────────────────────────────────────────
```

Although the numerical evaluation based on stratified, antithetic sampling leads to results close to their true values, tail integration of a density for a nonnegative random variable all the way from zero should be considered an extreme case, where precision is low. In practical applications, such integrals are of course best replaced by their (known) theoretical value of 1. For slightly larger values of a, the integral is best replaced by a definite integral from a to some large constant b (larger than all elements of a) plus an indefinite one from b. The indefinite integral from b only needs to be computed once and hence allows using many strata (M) for improving precision. One simple choice for b is

$$b = \max(a_i | i = 1, 2, \ldots, n)$$

where a_i's are the elements of a.

2.5 Example on forward recurrence density for minimum of two Weibull distributed random variables

To show some of the potential of the implemented RPN evaluator, let us consider the distribution arising from transforming the minimum of two Weibull distributed random variables into a forward recurrence time. Based on the expressions in (2) and (3) the density for the new random variable R is given by

$$f_R(r) = \frac{S_X(r)S_Z(r)}{\int_0^\infty S_X(s)S_Z(s)\,ds}$$

where S_X and S_Z are the survivor functions of the two original Weibull distributed random variables. This density can be implemented and evaluated using the RPN procedure as follows:

```
. mata
------------------------------------------------- mata (type end to exit) -----
: a0 = J(1, 1, 0)

: u0 = uniform(rows(a0), 20)

: frfweimin_exact = (&tostack(), &x ) \
>                   Sweimin_exact[2..rows(Sweimin_exact), .] \
>                   (&tostack(), &u0 \
>                    &tlMCatnode(), &a0 ) \
>                   Sweimin_exact[2..rows(Sweimin_exact), .] \
>                   (&tostack(), &u0 \
>                    &tlMCatwt(), &a0 \
>                    &intres(), NULL \
>                    &divide(), NULL)

: x
        1    2    3
    +----------------+
  1 |   0    2    8  |
  2 |  .7    1    6  |
    +----------------+

: rpnfcn(frfweimin_exact)
                  1              2              3
    +--------------------------------------------+
  1 |  1.090430679   .0852333006   1.16102e-08   |
  2 |  .5878359215   .4011468755   .0000116703   |
    +--------------------------------------------+

: end
--------------------------------------------------------------------------------
```

The corresponding survivor function can be implemented and evaluated as follows:

```
. mata
------------------------------------------------- mata (type end to exit) -----
: a0 = J(3, 1, 0)

: u0 = uniform(rows(a0), 10)

:
: frSweimin_exact = (&tostack(), &u \
>                    &tlMCatnode(), &a ) \
>                   Sweimin_exact[2..rows(Sweimin_exact), .] \
>                   (&tostack(), &u \
>                    &tlMCatwt(), &a \
>                    &intres(), NULL \
>                    &tostack(), &u0 \
>                    &tlMCatnode(), &a0 ) \
>                   Sweimin_exact[2..rows(Sweimin_exact), .] \
>                   (&tostack(), &u0 \
>                    &tlMCatwt(), &a0 \
>                    &intres(), NULL \
>                    &divide(), NULL)

: a
       1
    +-----+
  1 |  0  |
  2 |  2  |
  3 |  4  |
    +-----+
: rpnfcn(frSweimin_exact)
                 1
    ┌───────────────┐
  1 │  .9985048706  │
  2 │  .0459127621  │
  3 │  .0007040949  │
    └───────────────┘

: end
```

3 Discussion

In this paper we have developed and described a general procedure for numerical evaluation in Mata of complex density and distribution functions and for general integration of such functions. The procedure uses RPN to serialize computations, where individual steps in the computation are specified in rows of a controlling algorithm matrix. The approach allows building complex expressions by simply stacking relevant rows of algorithm matrices, and we have shown how to implement general integration based on stratified, antithetic sampling both for definite and indefinite integrals. The algorithm is numerically relatively efficient as it encourages not copying argument matrices more than needed in any given algorithm, and internally, the result matrix replaces the input matrix whenever possible, helping to minimize the strain on memory.

From one perspective, the procedure does not add anything new: what can be calculated with RPN can also be calculated with the ordinary `infix` notation already present in Mata. The more interesting question is, however, how easy it is to implement a complex function in a systematic way. From that perspective, the main virtue of the RPN implementation is its use of algorithm matrices. These matrices make the function definition—together with its required arguments—explicit, and it straightforwardly allows the function to be reused in even more complex functions without subparsing of arguments. It is possible that this could have also been obtained by clever exploitation of Mata's facility for defining structures, but the author is not aware of existing implementations of this type.

The first limitation of the described RPN procedure is its reliance on specific, named arguments, which all must exist at the time of defining the algorithm matrix. This makes algorithm matrices less universal, although this can in part be overcome with appropriate naming conventions for arguments. The primary objective of the RPN procedure—at least as envisaged by its author—is, however, to be used internally in other user-developed packages, and so the naming of arguments is less problematic.

Second, it is not clear how double integrals are to be handled with the proposed RPN procedure. Although it is often possible to avoid this, situations do arise where double integrals are inevitable. If, for example, no direct analytical expression had existed for the survivor function for the minimum of two Weibull distributed random variables, a double integral would have arisen when considering the survivor function for the associated forward recurrence distribution evaluated in section 2.5.

A further inconvenience of the procedure is that algorithm matrices quickly become complex and hard to read, which hinders debugging. To some degree this is offset by the ability to build complex functions from simpler algorithm matrices, which can be debugged individually, but in some settings the debugging may still prove problematic. Accordingly, the RPN procedure should not be considered a panache but rather its advantages and shortcomings should be carefully weighed against each other when considering how to implement a specific numerical evaluation.

With the present implementation of the RPN procedure, only the most basic stack functions and densities are supplied. For the RPN procedure to become more useful more subfunctions are needed, particularly implementation of other probability distributions and other integration rules.

4 References

Degroot, M. 1986. *Probability and Statistics*. 2nd ed. Reading, MA: Addison–Wesley.

Ripley, B. D. 1987. *Stochastic Simulation*. New York: Wiley.

Støvring, H., and W. Vach. 2005. Estimation of prevalence and incidence based on occurrence of health-related events. *Statistics in Medicine* 24: 3139–3154.

About the author

Henrik Støvring is a senior researcher at the Research Unit for General Practice, University of Southern Denmark, Odense, Denmark.

The Stata Journal (2007)
7, Number 4, pp. 556–570

Mata Matters: Structures

William Gould
StataCorp
College Station, TX
wgould@stata.com

Abstract. Mata is Stata's matrix language. In the Mata Matters column, we show how Mata can be used interactively to solve problems and as a programming language to add new features to Stata. Structures are the subject of this column. Structures are an advanced programming technique that can greatly simplify complicated code.

Keywords: pr0035, Mata, structures, `struct`

1 Introduction

Structures simplify complicated programs, particularly those that involve many subroutines. Using structures makes such programs easier to design, code, and modify. Structures can even make complicated programs easier to use.

A structure is an aggregate of variables under one name, such as

```
struct mystruct {
        real scalar     a
        real scalar     b
}
```

The above is called a structure definition and is shown exactly as you would enter it into Mata. It defines a new *type*, `struct mystruct`, and that new type is conceptually no different from the already existing types of `real`, `complex`, and `string`. Just as one can have `real` scalars, `real` vectors, and `real` matrices in a program, one can have `struct mystruct` scalars, `struct mystruct` vectors, and `struct mystruct` matrices. The function below has all three:

```
function myfunc(...)
{
        struct mystruct scalar  s
        struct mystruct vector  v
        struct mystruct matrix  M

        ...
}
```

Inside `myfunc()`, one accesses the components of the structures `s`, `v`, and `M` by referring to `s.a` and `s.b`, `v[`*i*`].a` and `v[`*i*`].b`, and `M[`*i*,*j*`].a` and `M[`*i*,*j*`].b`.

One can also refer to `s`, `v`, and `M` in their entirety, which is especially useful in making code more readable. In what follows, we pass the entirety of `s`, `v`, and `M` to subroutines `mysub1()`, `mysub2()`, and `mysub3()`:

```
function myfunc(...)
{
        struct mystruct scalar  s
        struct mystruct vector  v
        struct mystruct matrix  M
        ...
        real scalar             x, y, z

        ...
        x = mysub1(s)                           // <- new
        y = mysub2(v)                           // <- new
        z = mysub3(M)                           // <- new
        ...
}
```

For `mysub1()`, we did not code `x = mysub1(s.a, s.b)`; we simply coded `x = mysub1(s)` and that gave the subroutine access to everything in `s`, namely, `s.a` and `s.b`. In our example, the structure contains 2 components, but in a real programming application, `s` might contain 40. Whether 2 or 40, the code for the subroutines reads the same way:

```
real scalar mysub1(struct mystruct scalar s)
{
        ...
}

real scalar mysub2(struct mystruct vector v)
{
        ...
}

real scalar mysub3(struct mystruct matrix M)
{
        ...
}
```

If we need to refer to the components of the structure inside one of the subroutines, we do that in the same way we did in the main program, namely, by coding `s.a` and `s.b` in `mysub1()`, `v[`*i*`].a` and `v[`*i*`].b` in `mysub2()`, `M[`*i*,*j*`].a`, and `M[`*i*,*j*`].b` in `mysub3()`. As with arguments generally, we did not have to name the arguments `s`, `v`, and `M`, the same as in the main program. The component names, however, are fixed. If `mysub1()` had been declared `mysub1(struct mystruct scalar hset)`, then inside `mysub1()`, we would refer to `hset.a` and `hset.b`.

In addition to receiving entire structures as arguments, subroutines can return entire structures. Our main routine might call a subroutine `mysub4()` that returns a `struct mystruct` vector:

```
function myfunc(...)
{
        struct mystruct scalar  s
        struct mystruct vector  v
        struct mystruct matrix  M
        ...
        real scalar                x, y, z

        ...
        x = mysub1(s)
        y = mysub2(v)
        z = mysub3(M)
        ...
        v = mysub4(...)                     // <- new
        ...
}
```

In the new line `v = mysub4(...)`, `mysub4()` returns not just an entire structure, but a vector of entire structures. Subroutine `mysub4()` would be coded

```
struct mystruct vector mysub4(...)
{
        struct mystruct vector   new

        ...
        return(new)
}
```

This ability to package a collection of variables under one name can result in significant code simplification. One can write subroutines that receive a problem and return a solution, where problem and solution are each single variables consisting of many parts. The code for a particular problem might read

```
struct problem {
        ...
}

struct solution {
        ...
}

function main_routine(dep_varname, indep_varnames, touse_varname)
{
        struct problem scalar   p
        struct solution scalar  s

        p = set_up_problem(dep_varname, indep_varnames, touse_varname)
        check_assumptions(p)
        s = get_solution(p)
        display_problem_header(p)
        display_solution(s)
        post_results_to_stata(s)
}
```

Notice how easy the code is to read and therefore to modify.

2 When to use structures

One use of the structures is to implement new element types. For instance, if Mata had not included built-in type `complex`, and we found ourselves needing complex numbers, we could define

```
struct complex {
        real scalar   re, im
}
```

Because structure variables can be declared scalars, vectors, and matrices, with that single definition, we now have complex scalars, complex vectors, and complex matrices. We could now write the necessary complex-number manipulation functions, such as those for complex arithmetic, and we would be on our way to a solution. The advantage of this approach is that when we write the main part of our program, we write our code as if Mata had complex numbers all along. The only difference is that everywhere a real Mata program has `complex`, ours would have `struct complex`.

Another use of structures is to simplify the bookkeeping required in complicated problems, and here structure scalars are the most useful. The way I am using the word complicated—what distinguishes a complicated problem from a simple one—is the amount of information necessary to describe it. Say that the problem is statistical: there might be a vector y, a matrix X, another matrix of exogenous variables Z, a scalar *vcetype* that indicates how a variance matrix is to be calculated, and yet another scalar coding that indicates how y is coded. In solving this complicated problem, we will need to pass various parts of this information to various subroutines. The easy way to do that is to define one structure to hold all the information,

```
struct problemdef {
        real vector     y
        real matrix     X
        real matrix     Z
        real scalar     vcetype
        string scalar   coding
}
```

In our main program, we define a `struct problemdef` scalar, let's call it `pd`, and then we pass `pd` from one subroutine to the next.

One advantage of this coding style is that, if we later discover that we need to add another element to the structure—say, results are to be optionally projected according to matrix P—splicing in the new feature will be easy. We add the new element to the structure definition

```
struct problemdef {
        real vector     y
        real matrix     X
        real matrix     Z
        real matrix     P           // Projection, optional
        real scalar     vcetype
        string scalar   coding
}
```

and recompile. Then we modify only the subroutines that must use `P`, and `P` will be right at our fingertips when we need it.

Mata's `optimize()` function uses this approach. `optimize()` finds solutions for the minimum or maximum of a function and, in the process, uses an internal structure containing 57 components to track problems. You can see the structure definition by typing `viewsource optimize.mata` at the Stata (not Mata) prompt. In the code the structure has the inelegant name `opt__struct`, with two underscores between `opt` and `struct`, but do not let the name disguise the simplification the structure itself introduces. It is worth taking a look.

Mata's `optimize()` function uses scalar structures. Vectors of structures are often useful in programming data-management tasks. For instance, in a program dealing with disk files, a useful structure might be

```
struct filedesc {
        string scalar    filename
        real scalar      creation_date
        string scalar    filetype
        string scalar    path
}
```

The structure is defined for holding information of one file. In the program, one would use a `struct filedesc` vector to hold information on the collection of files.

3 When not to use structures

If your problem can be solved by writing a Mata function without recourse to subroutines, there is no need for structures. Consider, for instance,

```
struct point {
        real scalar  x, y
}

real scalar distance(x0, y0, x1, y1)
{
        struct point scalar  p0, p1

        p0.x = x0 ; p0.y = y0
        p1.x = x1 ; p1.y = y1
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

The structure added nothing except complication, and this program would be better written as

```
real scalar distance(x0, y0, x1, y1)
{
        return(sqrt((x1-x0)^2 + (y1-y0)^2))
}
```

You might argue that a structure could be beneficial had we used the new `struct point` type in the arguments of `distance()`. Function `distance()` could have been written

```
real scalar distance(struct point p0, struct point p1)
{
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

Here you argue that you have reduced the number of arguments from four to two and improved readability. You have a good argument.

The official StataCorp response is that users of your routines should not be required to use structures because structures require a level of programming knowledge that most users do not have. You have included a hurdle that will prevent some users from using your routine. We at StataCorp use structures but only in hidden ways.

4 Use of structures in hidden ways

In using Mata functions written by StataCorp, you have used structures and never noticed. Other programmers may wish to adopt our style. Obviously, we at StataCorp might write a program that, in its internals, defines a structure, passes the structure to various subroutines, and then returns results that were stored in the structure. That is not what I meant when I said you have used structures and never noticed. There are cases where we have returned a structure to you but did not tell you that it was in fact a structure.

Mata's `optimize()` does this. Say that you want to find the value of (p_1, p_2), which maximizes $y = \exp(-p_1^2 - p_2^2 - p_1 p_2 + p_1 - p_2 - 3)$. One way of obtaining the solution is

```
void myfunction(todo, p, y, g, H)
{
        y = exp(-p[1]^2 - p[2]^2 - p[1]*p[2] +
                p[1] - p[2] - 3)
}

S = optimize_init()
optimize_init_evaluator(S, &myfunction())
optimize_init_which(S, "max")
optimize_init_evaluatortype(S, "d0")
optimize_init_params(S, (0,0))
optimize(S)
```

You can read about `optimize()` in the *Mata Reference Manual* under [M-5] **`optimize()`** or by typing `help mata optimize()`. If you try the example, you will find the maximum occurs at $(1, -1)$.

What I want to emphasize here is the role of `S`. In the documentation we will tell you that `S = optimize_init()` is just something you have to do at the start of a problem. We call `S` the problem handle and tell you that, after obtaining `S`, you are to pass `S` to each of the other optimization functions.

We also tell you that if you feel the need to explicitly declare `S`, make it `transmorphic`. We do not tell you that `S` is a structure and that it is the 57-component structure I mentioned earlier. If you list `S`, you will see

```
: S
  0x15f26f0c
```

The value you see may differ, but it certainly does not look like a 57-component structure. Try typing

```
: liststruct(S)
  (output omitted)
```

You will see something different, namely, the 57-element structure.

When we wrote `optimize()`, we did not do anything special to make `S` list itself as 0x15f26f0c; this is just how structures look when you attempt to list them in raw form. The code 0x15f26f0c is the memory address where the structure is stored.

Let's review the calls to the `optimize()` routines, because now knowing that `S` is a structure, we can see how `optimize()` works. To obtain the solution, we suggested you type

```
S = optimize_init()
optimize_init_evaluator(S, &myfunction())
optimize_init_which(S, "max")
optimize_init_evaluatortype(S, "d0")
optimize_init_params(S, (0,0))
optimize(S)
```

(A boldface **S** is used for emphasis.)

When you typed **S** `= optimize_init()`, `optimize_init()` defined a structure, filled it in with default values, and returned it to you. In the subsequent calls, you supplied that same structure as an argument. The various `optimize_init_*(`**S**`, ...)` functions modified the information in **S**. Finally, when you typed `optimize(`**S**`)`, you unknowingly passed all 57 things `optimize()` needed to define the problem and to perform the requested optimization.

There are three benefits of this design.

1. If we at StataCorp later find that we need to track 58 rather than 57 things, we simply add another element to the structure and make the few modifications

necessary to use the new information. We need not tell you about it, except perhaps to mention a new feature.

2. By hiding from you that `S` is a structure, you cannot declare `S` to be a `struct opt__struct` scalar and so cannot access the information in `S`. More importantly, you cannot accidentally modify any of `S`'s components. To us, the programmers of `optimize()`, `S` contains 57 things. To you, the user, it contains only an odd hexadecimal number such as 0x15f26f0c. If `optimize()` fails to work as advertised, both of us can be certain that it is our fault, not yours.

   We use this same approach at StataCorp to protect us from ourselves. We manage a large code base, and we continually struggle to keep projects separated from one another. Our goal is that Stata internally be a well-defined set of independent modules. We want that so that modifications to one module do not require modifications to others, so we can make improvements to one module and not worry about unanticipated, and usually negative, side effects.

   Consider a project to implement a new estimator that uses `optimize()` as a subroutine. The code we write does not include the declaration `struct opt__struct scalar S`. The new project treats `S` as a transmorphic just as we tell you to do. Thus the new code we write cannot reach into what is properly the purview of `optimize()` and that keeps us from building dependencies where none should exist, which is always a temptation.

   By following this rule, if we need to modify `optimize()`, we can lay our hands on all the relevant code simply by searching for the declaration `struct opt__struct`. We can be certain that code not containing the declaration is irrelevant because the code does not have access to the contents, even if it has access to the variable.

3. This brings us to the third benefit, which is that we can modify our code and that will have no implication for yours. We can make changes in the structure, even large ones, and you do not even have to recompile your code that uses `optimize()`.

5 Mechanics

There are several mechanical issues involved in using structures that I need to tell you about. First and foremost, structures must be defined before they are used. In the program

```
struct point {
        real scalar  x, y
}

real scalar distance(struct point p0, struct point p1)
{
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

you cannot reverse the order of the definitions of `struct point` and `real scalar distance()` because the Mata compiler would not know how to interpret `p1.x`, `p0.x`, etc.

Second, although you can usually omit declarations of variables and arguments, you cannot omit them for structures, or more correctly, you cannot omit them if you need to access what is inside them. It would not do to code the above program

```
real scalar distance(p1, p2)
{
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

because then the Mata compiler would not know that `p0` and `p1` are `struct point`s and thus would not know how to interpret `p1.x`, `p0.x`, etc.

The requirement for explicit declarations also applies to structures used as variables within the program. In the program

```
real scalar distance(x0, y0, x1, y1)
{
        struct point scalar  p0, p1           // <- required

        p0.x = x0 ; p0.y = y0
        p1.x = x1 ; p1.y = y1
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

you cannot omit the `struct point scalar p0, p1`.

Finally, do not omit the `scalar` when it applies to structures. Those with C programming instincts will find this rule difficult to remember. The following will not work:

```
real scalar distance(x0, y0, x1, y1)
{
        struct point         p0, p1          // <- problem here

        p0.x = x0 ; p0.y = y0
        p1.x = x1 ; p1.y = y1
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

The function will compile without error, but an error will be issued when the function is executed. The error will arise because `struct point p0, p1` was interpreted as meaning `struct point matrix p0, p1` rather than as a `struct point scalar`. Why substituting `matrix` for `scalar` causes the problem is going to take some explaining, but it should be obvious to you why Mata assumed `matrix` rather than `scalar`. That was based on Mata's standard rule that has nothing to do with structures: omit the element type, and `transmorphic` is assumed; and omit the storage type, and `matrix` is assumed.

There is another Mata rule, also not specific to structures, that you already know but may never have verbalized:

> The entirety of an object must be defined before a component of the object can be defined.

Applying this rule in standard cases means that you cannot define the element `z[1]` before you have defined the entire vector `z` to be, say, 1×3. Similarly, you cannot define the element `Z[1,1]` before you have defined the entire matrix `Z` to be, for example, 2×3. Declaring `real vector z` or `real matrix Z` is not sufficient. This story is going to end that you cannot define `p0.x` before you define `p0`, but it is going to take a while to get there. Let's explore the standard, nonstructure case first. Everything we learn will apply to the structure case.

There are many ways you define `z` to be 1×3 or `Z` to be 2×3; there are so many ways that you may not even be aware that you are defining the entirety of `z` or `Z` because you do not think of it in that way. For `z`: 1×3, you might simply initialize *all of* `z` to contain the values you want:

```
z = (1, 2, 3)
```

For `Z`: 2×3, you could follow the same approach,

```
Z = (1, 2, 3 \ 4, 5, 6)
```

or perhaps you code something that simply results in `Z`'s entire definition, such as

```
Z = pinv(A)
```

Here you have a preexisting matrix `A`: 3×2, and the entirety of `Z` simply arises from calculation. `pinv()` returns the Moore–Penrose inverse.

The point is that defining the entirety of a vector or matrix object is such a common action you do not usually think of it as something special; it is obvious that you have to define `z` or `Z` before you can redefine `z[1]` or `Z[1,1]`. In some programs, however, `z` and `Z` do not define themselves naturally, and then you use built-in function `J()` to begin. You code

```
z = J(1, 3, 0)
```

```
Z = J(2, 3, 0)
```

`J(`r`,` c`,` x`)` returns an $r\times c$ matrix with all elements set to x. Thus `J(1, 3, 0)` returns a 1×3 vector with elements set to 0 and `J(2, 3, 0)` returns a 2×3 matrix with elements similarly set to 0. Then you can proceed to reset the elements,

```
z    = J(1, 3, 0)
z[1] = ...
z[2] = ...
z[3] = ...
```

So what is the effect of including explicit declarations such as `real vector z` and `real matrix Z` in your programs? In the program

```
function example(...)
{
        real vector    z
        real matrix    Z

        ...
}
```

the result is to make `z` be 1×0 and to make `Z` be 0×0 at the outset. The type and general shape are established, but the details have yet to be specified.

Structures work the same way. Let's return to our broken program and understand why it is broken and what we can do about it:

```
real scalar distance(x0, y0, x1, y1)
{
        struct point         p0, p1          // <- problem here

        p0.x = x0 ; p0.y = y0
        p1.x = x1 ; p1.y = y1
        return(sqrt((p1.x-p0.x)^2 + (p1.y-p0.y)^2))
}
```

To remind you, we omitted `scalar` from the `struct point p0, p1` declaration, with the result that `matrix` was assumed. We now know that matrices start their life being 0×0, and matrices of structures are no different from any other kind of matrix. `p0` and `p1` are 0×0. Hence statements such as `p0.x =` ... make no sense, in the same way that `z[1] =` ... makes no sense when `z` is 0×0. There is no first element of `z`, and similarly there is no `x` component of `p0`.

What can we do about it? The easy solution is simply to add the word `scalar` back to the declaration. Then `p0` and `p1` will be 1×1, and our program will work. The other solution would be to set `p0` equal to a 1×1 `struct point`, which is similar to how we solved the previous problems for `z` and `Z`. We called a function, `J()`, that returned a 1×3 real vector and a 2×3 real matrix, respectively. Similarly, we could set `p0` equal to the result from a function that returned a 1×1 `struct point` scalar; that is, a generic solution to our real vector `z` problem was

```
z    = ...
z[1] = ...
z[2] = ...
z[3] = ...
```

In the same way, a generic solution to our structure problem is

```
p0   = ...
p0.x = ...
p0.y = ...
```

The only difference between the two problems is that, in `z =` ..., the dots returned a 1×3 real vector, and in `p0 =` ..., the dots must return a 1×1 `struct point` scalar. I am about to show you how to do that but, before I do, let me emphasize that no rational person would use this solution because including `scalar` on the original declaration is easier. This solution will have a use, however, when we need a vector or matrix of structures. But let's stay with the 1×1 case at first. The solution is

```
p0   = J(1, 1, point())
p0.x = ...
p0.y = ...
```

The solution is `p0 = J(1, 1, point())`, just as `z = J(1, 3, 0)` was a solution to our real vector problem. The differences are that we do not want a 1×3 result, we want a 1×1 result, so the first two arguments to `J()` are 1 and 1; and we do not want a real result, we want a `struct point` result, so the third argument changes from 0, an instance of a real, to `point()`, an instance of a `struct point`. By the way, the solution simplifies to

```
p0   = point()
p0.x = ...
p0.y = ...
```

because `J(1, 1,` *anything*`)` simplifies to *anything*.

Let me explain about `point()`. When you declare a structure such as

```
struct point {
        real scalar  x, y
}
```

in addition to recording the definition, Mata also creates a function of the same name that returns a 1×1 instance of the structure. Thus when we code `p0 = J(1, 1, point())`, or `p0 = point()`, we obtain a 1×1 `struct point` scalar. Doing that defines the entirety of `p0` and then we can define its components.

As I said, including `scalar` on the declaration line would have been easier. In another problem, however, we might need a 1×3 `struct point` vector, and the solution just given generalizes to that case. We declare `struct point vector p` so that `p` is 1×0, and then we code `p = J(1, 3, point())`, just as we coded `z = J(1, 3, 0)` when we wanted a 1×3 real vector.

For vectors and matrices of structures, there is a variation available to the `J(`*r*`,` *c*`, point())` solution. The function `point()`—the function Mata automatically created from our structure definition—allows arguments, and its full syntax is `point(`*r*`,` *c*`)`. The r and c are optional. In full form, the function returns an $r \times c$ `struct point` vector or matrix. So rather than coding code `p = J(1, 3, point())`, we can code `p = point(1, 3)`. It does not matter which we code, although the second will execute a little more quickly.

6 Technical notes

What follows is included to reassure you that all the generalities one would expect are included in Mata's implementation of structures. With the exception of items (1)–(4), the issues addressed below seldom arise.

1. Structures and structure references are fully compiled, the latter into *address* + *offset* form. This means structures can be used without concerns about performance degradation. This also means that if a structure definition is modified, all functions that include explicit declarations of the structure must be recompiled.

2. A structure definition `struct` *whatever* `{ ... }` also results in the automatic creation of new function *whatever*`()`. If you are creating `.mlib` libraries, include *whatever*`()` among the functions saved, and similarly, if you are saving your functions in `.mo` files, be sure to save *whatever*`.mo`. Do this even if your other functions do not include calls to *whatever*`()`. The Mata compiler itself will insert calls to *whatever*`()` to construct structure scalars.

3. Let `w` be a `struct` *whatever* scalar and assume that `w` contains real matrix `X`. Then you can use `w.X` just as you can use any other matrix. In particular, `w.X[`*i*`,`*j*`]` refers to the i, j-element of matrix `w.X`.

4. Let `w` be a `struct` *whatever* vector and assume that `w` contains real matrix `X`. Then you can use `w[`*k*`].X` just as you can use any other matrix. In particular, `w[`*k*`].X[`*i*`,`*j*`]` refers to the i, j-element of matrix `w[`*k*`].X`.

5. Let `w` be a `struct` *whatever* matrix and assume that `w` contains real matrix `X`. Then you can use `w[`*k*`,`*l*`].X` just as you can use any other matrix. In particular, `w[`*k*`,`*l*`].X[`*i*`,`*j*`]` refers to the i, j-element of matrix `w[`*k*`,`*l*`].X`.

6. Let `w` be a `struct` *whatever* vector. Then `w[`*i*`]` is a `struct` *whatever* scalar. `w[`*i*`]` could be passed as an argument to any function expecting a `struct` *whatever* scalar.

7. Let `w` be a `struct` *whatever* matrix. Then `w[`*i*`,`*j*`]` is a `struct` *whatever* scalar. `w[`*i*`,`*j*`]` could be passed as an argument to any function expecting a `struct` *whatever* scalar.

   `w[`*i*`,.]` is a `struct` *whatever* rowvector. `w[`*i*`,.]` could be passed as an argument to any function expecting a `struct` *whatever* rowvector.

   `w[.,`*j*`]` is a `struct` *whatever* colvector. `w[.,`*j*`]` could be passed as an argument to any function expecting a `struct` *whatever* colvector.

8. Structures may contain other structures. For instance,

   ```
   struct point {
           real scalar  x, y
   }

   struct line {
           struct point scalar p0, p1
   }
   ```

   Let `li` be a `struct line` scalar. Then `li.p0` and `li.p1` are `struct point` scalars. `li.p0` and `li.p1` could be passed as arguments to any function expecting a `struct point` scalar.

`li.p0.x` is a real scalar and presumably is the x coordinate of `p0`.

`li.p0.y` is a real scalar and presumably is the y coordinate of `p0`.

9. Structures may contain structures may contain structures, and so on. `a.b.c.d` refers to the component `d` of structure `c` of structure `b` of structure `a`.

10. Structures may not contain themselves.

11. Structure pointers are allowed but are used less in Mata than in languages such as C. Structure pointers are necessary in Mata when you need structures to contain themselves, just as they are in C. Such constructs are commonly used to construct linked lists.

12. Let `w` be a `struct` *whatever* scalar. Then `&w` is a pointer to the scalar, which is to say, a `pointer(struct` *whatever* `scalar)`. `&(w.x)` is a pointer to the component `x` of the structure.

    Let `w` be a `struct` *whatever* vector. Then `&w` is a pointer to the vector, which is to say, a `pointer(struct` *whatever* `vector)`. `&(w[`*i*`])` is a `pointer(struct` *whatever* `scalar)`.

    Let `w` be a `struct` *whatever* matrix. Then `&w` is a pointer to the matrix, which is to say, a `pointer(struct` *whatever* `matrix)`. `&(w[`*i*`,`*j*`])` is a `pointer(struct` *whatever* `scalar)`.

13. `p->x` refers to component `x` of the structure pointed to by `p` and is equivalent to `(*p).x`. `p` must be explicitly declared a `pointer(struct` *whatever*`) scalar`.

14. Pointers to pointers of structures and pointers to pointers to pointers of structures, etc., are allowed. `(*p)->x` refers to component `x` of the structure pointer pointed to by `p` and is equivalent to `(**p).x`. `p` must be explicitly declared a `pointer(pointer(struct` *whatever*`) scalar) scalar`.

15. Be careful to distinguish between a `pointer(struct` *whatever*`) vector) scalar` and `pointer(struct` *whatever*`) scalar) vector`. `(*p)[i]` is appropriate for the first and `(*(p[i]))`, equivalent to `(*p[i])`, for the second. The same applies to matrices.

16. Memory management, and in particular the allocating and freeing of memory for structures, is automatic. If you do not use pointers to structures, this is simple and easy and works exactly as you would expect. If you do use pointers to structures, this is still simple and easy for you, but you may be surprised at the features provided by Mata. If you use pointers, the memory associated with a structure is not released until (1) the value of the last pointer pointing to the structure is changed or (2) the pointer itself ceases to exist because the program in which the pointer appears returns. For instance, consider a linked list. Let `*p` be a member of the list and assume `p->next` points to the next member. Assume that `p->next` is the only pointer pointing to `*(p->next)`. Simply coding `p->next = NULL` is sufficient to free `*(p->next)`. You may code this even if freeing `*(p->next)`

implies subsequent structures will need to be freed, such as `*(p->next->next)`, `*(p->next->next->next)`, and so on. Mata handles all of this for you.

7 Conclusion

Structures have two important uses: (1) introduction of new types such as complex (if Mata did not already have them) or quaternions (which Mata does not have), and (2) in programming complicated problems, where complicated means lots of information is required simply to describe what is to be done. In both cases, structures make it easy to share information across functions and make code more readable by hiding details. Hidden structures can also be used to enforce separation of complicated systems into independent modules. There is a cost to learning how to use structures, but those who program complicated tools will find the investment profitable.

About the author

William Gould is President of StataCorp, head of development, and principal architect of Mata.

The Stata Journal (2007)
7, Number 4, pp. 571–581

Speaking Stata: Counting groups, especially panels

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

Abstract. Counting panels, and more generally groups, is sometimes possible in Stata through a reduction command (e.g., `collapse`, `contract`, `statsby`) that produces a smaller dataset or through a tabulation command. Yet there are also many problems, especially with irregular sets of observations for varying times, that do not yield easily to this approach. This column focuses on techniques for answering such questions while maintaining the same data structure. Especially useful are the Stata commands `by:` and `egen` and indicator variables constructed for the purpose. With `by:` we often exploit the fact that subscripts are defined within group, not within dataset. `egen` functions are often used to produce group-level statistics. Tagging each group just once ensures that summaries, including counts, are of groups, not individual observations.

Keywords: dm0033, data management, panels

1 Introduction

Hierarchical or multilevel data are the focus of many statistical problems. Some researchers may even deal with nothing else in their daily statistical work. While modeling methods for such data attract the greatest publicity, many questions arising with multilevel structures also call for basic data management. In this column I focus on a fundamental kind of query. Using the language of panel data, the generic question is, "How many panels . . . ?" Recently in the *Stata Journal*, I emphasized the value of the `count` command (Cox 2007a; see [D] **count**) and explained how to calculate the number of events in time intervals (Cox 2007b). The kind of question discussed here is often even simpler. The challenge is to translate it into Stata terms.

Medical research provides examples of panels that are easy to think about. You might have x-ray records with patient identifier, age of patient, and date of x-ray as variables, or you might have drug prescription records with identifier, drug and date of prescription as variables. Similar panels also arise frequently in social sciences or business. You might have sales records with customer identifier, product purchased, and date of purchase.

These problems are not restricted to datasets indexed by identifiers and dates. Those with daily meteorological data might ask, "In how many years . . . ?" Here years play the same role as panels. Furthermore, no time basis is needed. A dataset on individuals organized by families might lead to questions about how many families satisfy certain criteria, such as having two or more children or being all female.

With such examples in mind, let us identify the kind of problem a little more generally:

1. Observations (records) refer to individuals (x-rays, prescriptions, sales, days, family members).
2. Observations are grouped in some way, commonly into panels (patients, customers), but also into years, families, and so forth. For simplicity, we will discuss panels and trust that it is clear that the ideas apply more generally.
3. No time basis is necessary for the ideas here to apply. When date and/or time is an aspect of the data, there is no assumption that data are regularly spaced in time, or even that this is ideal.
4. The researcher's question is about panels. Each panel corresponds to one or more observations. We have to think, therefore, on two levels: that of the panels and that of the individual observations.

Even with that specification, there remain several ways of approaching such questions in Stata. We can seek to produce a reduced dataset, with one observation per panel, that contains the answers to our question. Commands such as `collapse`, `contract`, and `statsby` (see corresponding entries in [D]) feature strongly in this approach, sometimes after some preprocessing. A tabulation may also be the answer after some preprocessing. Both approaches have several attractions, but neither is the main focus here. Once you realize that such commands offer a solution, the problem is typically almost solved—the main issue being creating the precise syntax.

The focus here is on a different strategy in which we maintain the same data structure. The tools that are repeatedly useful include `by:`, `egen` and indicator variables constructed for the purpose. These are discussed in many Stata tutorials (e.g., Cox 2002a, Cox 2002b), but it can take much practice before they become intuitive.

2 Data structure is the problem, but also the solution

Consider again the issue of data structure. This kind of question is a little tricky because a data structure that is in most ways convenient, and indeed natural, is nevertheless awkward for some purposes.

Imagine, once more, data on x-rays or drug prescriptions. Irregular timing and differing numbers of observations in such panels are facts of life. Thus it would usually be artificial and, indeed, inefficient to `reshape` datasets into wide structures with just one observation for each patient. You could represent the first, second, and so forth of several visits by variables such as `date1`, `date2`, and so on. But you would need as many such variables as there were observations in the panel with most observations.

There would then be many missing values. Worst of all, many questions would still be difficult to answer.

Hence the strategy discussed here keeps the same structure. It is likely to be easier to work with, not only for the questions discussed but for other kinds of questions for the same datasets.

3 A first example: panels of firms

Let us work out a Stata answer to a question about a specific dataset. You can read it in yourself, assuming that you have Internet access:

```
. webuse grunfeld
```

This happens to be a well-behaved panel dataset. Each of 10 companies is observed for each of 20 years. There are no gaps and no missing values. However, we do not require such good behavior, as you will see.

How many companies, and which ones, experienced higher than average growth in `invest` over the period? Clearly, this is a question about companies, but it must be answered from data on individual companies in specific years. Under the terms of our self-denying ordinance, we will not resort to a reduction or `reshape` of the data, nor will we exploit the equal lengths of the panels or their regular timing in any way.

There could be discussion on how best to measure growth, but we choose a simple definition of (last − first) / first.

```
. by company (year), sort: gen invest_change = (invest[_N] - invest[1]) / invest[1]
```

The first Stata lesson here is a reminder of the value of `by:`. One basic tutorial was given in an earlier column (Cox 2002a). Alternatively, check out entries on `by:` indexed in the Stata manual. `by:` is the basic tool for calculations by panels or other groups. Those commands that appear to sidestep it will almost always use it internally.

Even if the data are already sorted in the way you want, it does no harm to spell out a sort order. That will also, on occasion, save you from incorrect results when the data are not sorted as you desire. It is also helpful when looking back at past log files that give session transcripts.

A feature we are exploiting here is that under `by:`, subscripts are interpreted within the groups defined by `by:`. We insisted, as part of the command just given, that the observations are sorted by `year` within blocks of observations for each `company`. That being so, `invest[1]` and `invest[_N]` are values of `invest` for the earliest and latest observations for each company.

You should appreciate two possible problems with this calculation. If invest[1] is ever 0, the calculation will yield a missing result. Similarly, if either first or last value is missing, we will also get a missing result.

Clearly, the advice is to check your data so that you can identify problems such as these. The second problem is sufficiently common that a more general method of dealing with it is worth knowing about. In related form, it will be familiar to Stata programmers.

Construct an indicator variable that captures missingness:

```
. gen byte invest_miss = missing(invest)
```

Using a byte type is not essential unless you are short of memory but is good practice. The new variable will contain 1 whenever invest is missing and 0 otherwise. With this in hand, we can tweak the sort order:

```
. by invest_miss company (year), sort: gen invest_change =
>  (invest[_N] - invest[1]) / invest[1]
```

The sort order is now first by invest_miss, second by company, and third by year. Thus all the observations with missing values of invest have been put on one side in quarantine, where they cannot infect calculations made with the nonmissing values. invest[1] and invest[_N] now refer to the first and last years in each group defined jointly by whether invest is missing and company. Thus we would always use the first and last years for which values are available. Otherwise, missing values of invest would inevitably yield missings for invest_change.

You might not want this. You might say: I want to use data for the first and last years of the study consistently. If the result is missing, I want to know that and then ignore the result. That is a scientific or practical decision for you to make. The point is to know how to tackle the issue in different ways within Stata. With irregular times and a sprinkling of missings, using the first and last nonmissings could easily be the better choice.

With the Grunfeld data all that would make no difference, so we will not pursue the detail further in this example. Your own data may not be so well behaved.

We now have measures of growth, so what about the average? You could reach for summarize. However, that without qualification would summarize 200 observations, and not 10 panels. Although the difference will not bite with equal-length panels, we need a more general method. (Conversely, if you did want summaries over panels to be weighted by number of observations within panel, then summarize on the observations is exactly right: just be sure that you choose it consciously rather than accidentally.)

Inspection of the data with edit or list will confirm what you will have expected: the results for invest_change are identical within each panel. Thus we need to pick just one observation from each panel for further use. You could be quite capricious about this and pick any one haphazardly or use your favorite small integer (the third, or the seventh, or whatever), but clearly none of those methods is general. As panels could be as small as a single observation, there are two general methods of selection when all values are identical within panels: selecting either the first or the last. (The last of one is the same as the first of one.)

If we used the first, the technique once more is to produce an indicator variable:

```
. by company: gen byte tag = _n == 1
```

The new variable, `tag`, will be 1 for the first observation in each panel, for which `_n == 1` is true, and 0 otherwise. The new variable, in particular, will never be missing. The name `tag` may suggest that we are tagging some observations for future study, and it is also a prompt for another way of doing essentially the same:

```
. egen tag = tag(company)
```

That alternative is less typing, but a bit more work for Stata. By the way, producing a `byte` variable is wired in to `egen, tag()`, even if you specify otherwise.

`egen` is a major help in this kind of work, and it pays to be familiar with its possibilities (Cox 2002b; [D] **egen**). These include not only those `egen` functions provided with official Stata, but an even larger number of user-written functions: use `findit` to identify what is available from where. In other panel calculations, researchers frequently want to relate data to summaries over panels or over times, and `egen` is then often the tool of choice. It is also more direct than using `collapse` to produce a set of summaries and then using `merge` to put those summaries back into the original dataset, a route surprisingly often followed.

If we decided to tag the last observation in each panel, the syntax would be

```
. by company: gen byte tag = _n == _N
```

Given such an indicator variable, we now have an easy way to pick precisely one observation from each panel—in situations in which every value of a variable of interest is the same within each panel. Do note that stipulation carefully. We can just add `if tag == 1` to commands as appropriate. In fact, we can do better than that. All we need say is `if tag`. The correspondence is simply this: the logical expression `tag == 1` is true, and thus evaluates to 1, precisely when `tag` itself is nonzero, because the only nonzero (true) value possible for `tag` is 1. (Recall the flag earlier that `tag` cannot be missing, a property also wired into `egen, tag()`. There is more on true and false in Stata in Cox [2002a].)

With such tricks, we are now almost finished with this question. The mean of our growth measure across companies is given by

```
. summarize invest_change if tag
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
invest_cha~e |        10    2.843469    1.758583   .8527977   5.666012
```

We will save this mean in a local macro:

```
. local mean = r(mean)
```

How many companies have higher than average growth?

```
. count if invest_change > `mean´ & tag
  5
```

And which are they?

```
. levelsof company if invest_change > `mean´ & tag
1 3 4 6 8
```

The extra condition `& tag` is redundant in the last statement, but it does no harm.

4 A second example: visits to a clinic

How many patients who visited a clinic in 2005 also visited in 2006? For this kind of calculation, we need an identifier variable, say, `id`, and a date variable, say, `date`, which we will take to be a daily date. The logical condition marking a visit in 2005 may be expressed in various ways. Here are two:

```
if year(date) == 2005
```

and

```
if inrange(date, mdy(1,1,2005), mdy(12,31,2005))
```

The first is pleasingly direct for the question asked. The second is very much worth knowing because of the way it can be modified for many other problems, especially those in which the interval is more irregular, usually for some very compelling external reason.

Some Stata morals deserve comment. Beginning Stata users often work through a problem like this by thinking: I need a variable specifying the year to make progress. A variable for year would indeed make the problem look simpler, but often you can move directly to a solution just by using functions such as `year()`. More generally, scanning the documentation giving the list of functions ([D] **functions**) will often make clear some tools that you have been missing. It is one of the driest parts of the Stata documentation, all characters and no plot, but one of the most useful.

`inrange()` is often overlooked. You could always spell out a range using the component inequalities, but using `inrange()` can often be simpler. What is more, it excludes missings, usually the right thing to do. Otherwise, `inrange()` always includes its end-values, and so specifies closed intervals $[a, b]$. Thus `inrange(1, 1, 7)` and `inrange(7, 1, 7)` both evaluate to 1, meaning true. Further examples are given in Cox (2006).

We could construct indicator variables as before

```
. gen byte in2005 = inrange(date, mdy(1,1,2005), mdy(12,31,2005))
. gen byte in2006 = inrange(date, mdy(1,1,2006), mdy(12,31,2006))
```

which flag on the observation level whether (or not) a visit was in 2005 or 2006. But there is a better way to do it. As we want answers for patients (our panels), all we care about is whether there was any visit in 2005 and also any visit in 2006 by each patient. A cumulative sum given by the `sum()` function will serve well. We can calculate, separately by `id`,

```
. sum(inrange(date, mdy(1,1,2005), mdy(12,31,2005)))
```

and its twin

```
. sum(inrange(date, mdy(1,1,2006), mdy(12,31,2006)))
```

which will get us most of the way toward a solution. Here calls to three distinct functions, `mdy()`, `inrange()`, and `sum()`, are nested. As with elementary algebra, Stata works from the inside outward when given an expression with multiple parentheses.

As a detail on the side, we could calculate the fixed dates and feed them to Stata as constants in local macros

```
local b2005 = mdy(1,1,2005)
local e2005 = mdy(12,31,2005)
local b2006 = mdy(1,1,2006)
local e2006 = mdy(12,31,2006)
```

and so work with

```
. sum(inrange(date, `b2005´, `e2005´))
```

and

```
. sum(inrange(date, `b2006´, `e2006´))
```

That should be less work for Stata, as otherwise the `mdy()` calculations are repeated for each observation. However, I would never do this unless I was writing a program or do-file likely to be used repeatedly. Life is too short to optimize all your code.

Now consider what happens for each panel as either cumulative sum is calculated. Sums produced by `sum()` are always initialized as 0. (Occasionally, you have to fight this, as when the sum of several missings is 0, not missing, and that is not what you want.)

For all dates up to and including 2004, both `inrange()` calls will yield 0. Thus the cumulative sums will remain 0. For each date in 2005, the first `inrange()` call will yield 1, and correspondingly for 2006 and the second `inrange()` call. Any dates in 2007, or later, would also yield 0. Thus the cumulative sums yielded by `sum()` are cumulative counts of visits in the two years.

All the question requires is knowing, for each patient, whether there was any visit in 2005 and also any visit in 2006. We can do most of the work in one command:

```
. by id, sort: gen byte both =
> sum(inrange(date, mdy(1,1,2005), mdy(12,31,2005)))
> &
> sum(inrange(date, mdy(1,1,2006), mdy(12,31,2006)))
```

Note the logical "and" (`&`) operator. The result of the logical expression will be true (1 numerically) if and only if both sums are true (nonzero). With this calculation, cumulative sums can never be negative, as they are initialized as zero and we only ever add 1s or 0s as we work through each panel. Thus a cumulative sum can only be zero (no events of the kind specified) or positive (at least one event of the kind specified).

There is more technique that is useful. The answer we want for each panel will be in the last observation for each panel. We can select that too within the same command:

```
. by id, sort: gen byte both = cond(_n == _N,
>  sum(inrange(date, mdy(1,1,2005), mdy(12,31,2005))) &
>  sum(inrange(date, mdy(1,1,2006), mdy(12,31,2006))), .)
```

The `cond()` function here splits the universe in two. If an observation is the last in its panel, that is `_n == _N`, we get the cumulative sum we want, the final result at the end of the panel. For all other observations in each panel, we just get missing. Stata is obliged to work out the cumulative sums for every observation, but only for the last observation in each panel does it put the result in our new variable `both`.

Once more we are exploiting the fact that under `by:` subscripts are defined within group, defined here by `id`, not within dataset. The `cond()` function here lets us do in one statement what would otherwise require two, or an even more complicated statement. For a tutorial, see Kantor and Cox (2005).

Using `cond()` in this way has a nice downstream effect. The result of the previous command, centered on a logical operation using `&`, could be 0 (when either cumulative sum was zero, or both), 1 (when both cumulative sums are nonzero, meaning positive in this problem), or missing (whenever the observation was not the last in its panel). We can therefore proceed directly to counting panels with visits in both 2005 and 2006:

```
. count if both == 1
```

In other words, the effect of the statement was to put nonmissing results for `both` into just one observation in each panel, as we did earlier, but differently, when we defined a tag variable. So we can count what we want without tagging.

5 An egen solution for visits to a clinic

There is another way to do it, which hides some of the details here, and thus has its attractions. It is less efficient in computing terms. However, you might not care about that, especially whenever your personal time taken to figure out Stata code is far greater than any possible machine time to produce the results.

We can use the `egen` function `total()` to get totals or sums. These are not cumulative sums, however, and so will be constant for whatever group of observations is used. (Before Stata 9, `egen`'s `total()` function was called `sum()`. See [D] **egen**.)

The syntax for `egen, total()` reveals that it will feed on an expression, which can be (much) more complicated than one variable name. This detail is, it seems, frequently overlooked. In revisiting the example, we use the `year()` function as a reminder of that solution. As before, the sum or total of a logical expression yielding 0 or 1 yields the count or frequency of observations in which that expression is true.

```
. by id, sort: egen in2005 = total(year(date) == 2005)
. by id: egen in2006 = total(year(date) == 2006)
. gen byte both = in2005 & in2006
. egen tag = tag(id)
. count if both & tag
```

It is characteristic of `egen` solutions that there is no fancy footwork with subscripts such as `_n` or `_N`. Typically, that is what the `egen` functions do on your behalf but out of sight.

There is a more instructive question. Why does the solution not telescope to

```
. by id, sort: egen both = total(year(date) == 2005 & year(date) == 2006)
. egen tag = tag(id)
. count if both & tag
```

as might be thought? `egen` would evaluate

```
year(date) == 2005 & year(date) == 2006
```

for each observation and give a total of the results for each `id`. However, for any observation it can never be true that the year is 2005 and also 2006. It could be one, or the other, or neither, but it cannot be both. Thus the count produced by `egen, total()` would always be zero with this argument. This problem is also not fixed by using the logical 'or' operator (`|`):

```
. by id, sort: egen either = total(year(date) == 2005 | year(date) == 2006)
```

This command counts visits in either 2005 or 2006, without regard to whether visits were in both 2005 and 2006. The result could easily be of interest, but it is an answer to a different question.

This last detail underlines once more the crucial distinction between what is true or false about individual observations and properties of groups of observations, panels in this example.

6 Other possibilities

Techniques for related problems should now be within sight.

Visits in 2005, but not 2006, and vice versa, or patients in the dataset who visited in neither year, could be identified by looking for the appropriate zero sums.

Problems that require knowing the numbers of visits call for looking directly at the sums.

Problems could easily be more complicated given other conditions that you might want to specify. You might want to know not only about visits in 2005 and 2006 but also particular kinds of outcome. Typically, you would just stipulate extra logical expressions. Such problems need not be much more difficult. It is just a matter of spelling out all the restrictions needed.

Problems involving three or more levels of structure also yield to extensions of the techniques here.

Naturally, many related problems do call for either a reduction to a smaller dataset with one observation for each panel or a tabulation, or both. If we wanted a breakdown of visits by patient and year, a year variable would indeed be needed, but a table might be the answer.

7 Conclusions

Counting panels, and more generally groups, is sometimes possible through a reduction command that produces a smaller dataset or through a tabulation command. Yet there are also many problems, especially with irregular sets of observations for varying times, that do not yield easily to this approach. This column has focused on techniques for answering such questions while maintaining the same data structure. Especially useful are `by:`, `egen`, and indicator variables constructed for the purpose. With `by:` we often exploit the fact that subscripts are defined within group, not within dataset. `egen` functions are often used to produce group-level statistics. Indicator variables containing 0s and 1s yield answers to many questions by simple logic and arithmetic. Tagging each group just once ensures that summaries, including counts, are of groups, not individual observations.

8 Acknowledgments

This column stems from various Statalist threads in October 2007. One solution draws on code posted by Kit Baum in response to a similar question.

9 References

Cox, N. J. 2002a. Speaking Stata: How to move step by: step. *Stata Journal* 2: 86–102.

———. 2002b. Speaking Stata: On getting functions to do the work. *Stata Journal* 2: 411–427.

———. 2006. Stata tip 39: In a list or out? In a range or out? *Stata Journal* 6: 593–595.

———. 2007a. Speaking Stata: Making it count. *Stata Journal* 7: 117–130.

———. 2007b. Stata tip 51: Events in intervals. *Stata Journal* 7: 440–443.

Kantor, D., and N. J. Cox. 2005. Depending on conditions: A tutorial on the `cond()` function. *Stata Journal* 5: 413–420.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He wrote several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*.

The Stata Journal (2007)
7, Number 4, pp. 582–583

Stata tip 52: Generating composite categorical variables

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

If you have two or more categorical variables, you may want to create one composite categorical variable that can take on all the possible joint values. The canonical example for Stata users is given by cross-combinations of `foreign` and `rep78` in the `auto` data. Setting aside missings, `foreign` takes on values of 0 and 1, and `rep78` takes on values of 1, 2, 3, 4, and 5. Hence there are ten possible joint values, which could be 0 and 1, 0 and 2, and so forth. As it happens, only eight occur in the data. If we add the value labels attached to `foreign`, we have Domestic 1, Domestic 2, and so forth.

Writing the values like that raises the question of whether these cross-combinations will be better expressed as string variables or as numeric variables with value labels. On the whole, an integer-valued numeric variable with value labels defined and attached is the best arrangement for any categorical variable, but a string variable may also be convenient, especially if you are producing a kind of composite identifier.

A method often seen is to produce string variables with `tostring` (see [D] **destring**), for example,

```
. tostring foreign rep78, generate(Foreign Rep78)
. gen both = Foreign + Rep78
```

Naturally, there are endless minor variations on this method. A small but useful improvement is to insert a space or other punctuation:

```
. gen both = Foreign + " " + Rep78
```

However, this method is not especially good. `tostring` is really for correcting mistakes, whether attributable to human fault or to some software used before you entered Stata: some variable that should be string is in fact numeric. You need to correct that mistake. `tostring` is a safe way of doing that.

That intended purpose does not stop `tostring` being useful for things for which it was not intended, but there are two specific disadvantages to this method:

1. This method needs two lines, and you can do it in one. That is a little deal.
2. This method could lose information, especially for variables with value labels or with noninteger values. That is, potentially, a big deal.

The second point may suggest using `decode` instead, but my suggestions differ. A better method is to use `egen, group()`. See [D] **egen**.

```
. egen both = group(foreign rep78), label
```

This command produces a new numeric variable, with integer values 1 and above, and value labels defined and attached. Particularly, note the `label` option, which is frequently overlooked.

This method has several advantages:

1. One line.
2. No loss of information. Observations that are identical on the arguments are identical on the results. Value labels are used, not ignored. Distinct noninteger values will also remain distinct.
3. The label is useful—indeed essential—for tables and graphs to make sense.
4. Efficient storage.
5. Extends readily to three or more variables.

Another fairly good method is to use `egen, concat()`.

```
. egen both = concat(foreign rep78), decode p(" ")
```

This command creates a string variable, so it is less efficient for data storage and is less versatile for graphics or modeling. Compared with `tostring`, the advantages are

1. One line.
2. You can mix numeric and string arguments. `concat()` will calculate what is needed.
3. You can use the `decode` option to use value labels on the fly.
4. You can specify punctuation as separator, here a blank.
5. Extends to three or more variables.

The Stata Journal (2007)
7, Number 4, pp. 584–586

Stata tip 53: Where did my p-values go?

Maarten L. Buis
Department of Social Research Methodology
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.buis@fsw.vu.nl

A useful item in the Stata toolkit is the returned result. For example, after most estimation commands, parameter estimates are stored in a matrix `e(b)`. However, these commands do not return the t statistics, p-values, and confidence intervals for those parameter estimates. The aim here is to show how to recover those statistics by using the results that are returned. Consider the following OLS regression:

```
. sysuse auto
(1978 Automobile Data)
. regress price mpg foreign

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =   14.07
       Model |   180261702     2  90130850.8           Prob > F      =  0.0000
    Residual |   454803695    71  6405685.84           R-squared     =  0.2838
-------------+------------------------------           Adj R-squared =  0.2637
       Total |   635065396    73  8699525.97           Root MSE      =  2530.9

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -294.1955   55.69172    -5.28   0.000    -405.2417   -183.1494
     foreign |   1767.292    700.158     2.52   0.014     371.2169    3163.368
       _cons |   11905.42   1158.634    10.28   0.000     9595.164    14215.67
------------------------------------------------------------------------------
```

1 t statistic

The t statistic can be calculated from $t = (\widehat{b} - b)/\text{se}$, where $\widehat{b}$ is the estimated parameter, b is the parameter value under the null hypothesis, and se is the standard error. The null hypothesis is usually that the parameter equals zero; thus we have $t = \widehat{b}/\text{se}$. The t statistic for one parameter (`foreign`) can be calculated by

```
. di _b[foreign]/_se[foreign]
2.5241336
```

All the parameter estimates are also returned in the matrix `e(b)`. A vector of all standard errors is a bit harder to obtain; they are the square roots of the diagonal elements of the matrix `e(V)`. In Mata that vector can be created by typing `diagonal(cholesky(diag(V)))`. Continuing the example, a vector of all t statistics can be computed within Mata by

```
: b = st_matrix("e(b)")´
: V = st_matrix("e(V)")
: se = diagonal(cholesky(diag(V)))
: b :/ se
                  1
    +---------------+
  1 |  -5.282572354 |
  2 |    2.52413358 |
  3 |   10.27538518 |
    +---------------+
```

2 p-value

The p-value can be calculated from $p = 2 * (1 - T(\text{df}, |t|))$, where T is the cumulative distribution function of Student's t distribution, df is the residual degrees of freedom, and $|t|$ is the absolute value of the observed t statistic. The t statistic was calculated before, and the residual degrees of freedom are returned as `e(df_r)`. The absolute value can be calculated by using the `abs()` function, and $(1 - T(\text{df}, t))$ can be calculated by using the `ttail(`df,t`)` function. The calculation is put together as follows:

```
. local t = _b[foreign]/_se[foreign]
. di 2*ttail(e(df_r),abs(`t´))
.01383634
```

Using Mata, the vector of all p-values is then

```
: df = st_numscalar("e(df_r)")
: t = b :/ se
: 2*ttail(df, abs(t))
                 1
    +--------------+
  1 |  1.33307e-06 |
  2 |  .0138363442 |
  3 |  1.08513e-15 |
    +--------------+
```

3 Confidence interval

The lower and upper bounds of the confidence interval can be calculated as $\widehat{b} \pm t_{\alpha/2}\text{se}$, where $t_{\alpha/2}$ is the critical t-value given a significance level $\alpha/2$. This critical value can be calculated by using the `invttail(`df, $\alpha/2$`)` function. The lower and upper bounds of the 95% confidence interval for the parameter of `foreign` are thus given by

```
. di _b[foreign] - invttail(e(df_r),0.025)*_se[foreign]
371.2169
. di _b[foreign] + invttail(e(df_r),0.025)*_se[foreign]
3163.3676
```

(*Continued on next page*)

The vectors of lower and upper bounds for all parameters follow suit in Mata as

```
: b :- invttail(df,0.025):*se, b :+ invttail(df,0.025):*se
                  1              2
    +-------------------------------+
  1 |  -405.2416661   -183.1494001  |
  2 |   371.2169028    3163.367584  |
  3 |     9595.1638    14215.66676  |
    +-------------------------------+
```

4 Models reporting z statistics

If you are using an estimation command that reports z statistics instead of t statistics, the values become

- `_b[foreign]/_se[foreign]` for the z statistic;
- `2*normal(-abs(‘z’))` for the p-value (where the minus sign comes from the fact `normal()` starts with the lower tail of the distribution, whereas `ttail()` starts with the upper tail);
- `_b[foreign] - invnormal(0.975)*_se[foreign]` for the lower bound of the 95% confidence interval, and `_b[foreign] + invnormal(0.975)*_se[foreign]` for the upper bound (.975 is used instead of .025 for the same kind of reason).

5 Further comments

Often it is unnecessary to do these calculations. In particular, if you are interested in creating custom tables of regression-like output the `estimates table` command or the tools developed by Jann (2005, 2007) are much more convenient. Similarly, if the aim is to create graphs of regression output, take a good look at the tools developed by Newson (2003) before attempting to use the methods described here. This tip is for situations in which no such command does what you want.

References

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

———. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Newson, R. 2003. Confidence intervals and p-values for delivery to the end user. *Stata Journal* 3: 245–269.

The Stata Journal (2007)
7, Number 4, pp. 587–589

Stata tip 54: Post your results

Philippe Van Kerm
CEPS/INSTEAD
Differdange, Luxembourg
philippe.vankerm@ceps.lu

The command `post` and its companion commands `postfile` and `postclose` are described in [P] **postfile** as “utilities to assist Stata programmers in performing Monte Carlo type experiments”. That description understates their usefulness, as `post` is one of the most flexible ways to accumulate results and save them for later use in an external file.

Stata output is displayed in the Results window and can be stored in log files. However, browsing log files and selecting particular results can be tedious and inefficient. Fortunately, there are several alternatives, including the use of `file` (see [P] **file**) or the `estimates` suite of commands (see [R] **estimates**), and `post`, the focus here.

Use of `post` is fully described in [P] **postfile**. The steps are in essence:

1. Call `postfile` to initialize the results file: identify the filename, name its variables, and determine their types.
2. Run the analysis and accumulate the results by repeatedly calling `post`. Each call to `post` adds one observation (record or line) to the results file.
3. Close the results file with `postclose`.

`post` is flexible in what it records: e-class, r-class, or s-class results, string or numeric values, locals, constants, etc. Posted results are recorded without disturbing the data in memory. This is particularly neat: it keeps datasets tidy and allows calling multiple files without interfering with the accumulation of results.

This first example uses the `auto` data. We loop over all possible combinations of `foreign` and `rep78` and save average `price` within each group. Estimates are recorded in a new file named `autoinfo.dta`, which is later opened for displaying results with `tabdisp`.

```
. tempname hdle
. postfile `hdle´ foreign rep78 mean using autoinfo
. sysuse auto
(1978 Automobile Data)
. forvalues f=0/1 {
  2.         forvalues r=1/5 {
  3.                 summarize price if foreign==`f´ & rep78==`r´, meanonly
  4.                 post `hdle´ (`f´) (`r´) (r(mean))
  5.         }
  6. }
. postclose `hdle´
. use autoinfo, clear
. label define lf 0 "Domestic car" 1 "Foreign car"
. label values foreign lf
. label variable foreign "Origin of car"
. label variable rep78 "1978 repair record"
. tabdisp rep78 foreign, cell(mean)

─────────────────────────────────────────
1978      │
repair    │         Origin of car
record    │  Domestic car   Foreign car
──────────┼──────────────────────────────
        1 │        4564.5
        2 │      5967.625
        3 │      6607.074      4828.667
        4 │      5881.556      6261.444
        5 │        4204.5      6292.667
─────────────────────────────────────────
```

This example just shows the technique. In fact, for similar problems, the same effect can be produced easily with `statsby` (see [D] **statsby**):

```
. sysuse auto
(1978 Automobile Data)
. statsby mean=r(mean), by(foreign rep78) saving(autoinfo2): summarize price
(running summarize on estimation sample)
  (output omitted)
. use autoinfo2
(statsby: summarize)
. tabdisp rep78 foreign, cell(mean)
  (output omitted)
```

However, `statsby` is too restricted for more elaborate problems. A second example shows computations that store results for each of a series of files, here the numbers of observations and variables. It also demonstrates that graph commands are easily used for displaying results.

```
. tempname hdle
. postfile `hdle´ str20 name str100 label nobs nvar using sysfilesinfo
. sysuse dir
  (output omitted)
. local allfiles "`r(files)´"
. foreach dtafile of local allfiles {
  2.         sysuse `dtafile´, clear
  3.         describe, short
  4.         post `hdle´ ("`dtafile´") (`"`: data label´"´) (r(N)) (r(k))
  5. }
  (output omitted)
. postclose `hdle´
. use sysfilesinfo
. keep if label!=""
(18 observations deleted)
```

```
. replace name = subinstr(name,".dta","",.)
(15 real changes made)
. label variable nobs "Number of observations in dataset"
. label variable nvar "Number of variables in dataset"
. scatter nvar nobs if nobs<250, mlabel(name) mlabpostion(12)
```

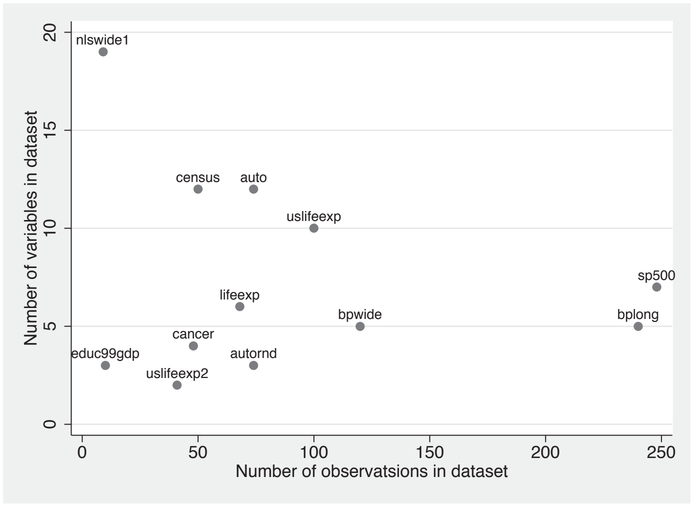

Figure 1: Exploiting posted results

The flexibility of `post` for both collection, when relevant results are posted, and processing, when collected results are analyzed, makes it useful in a broad range of settings, which is different from Monte Carlo simulations.

The Stata Journal (2007)
7, Number 4, pp. 590–592

Stata tip 55: Better axis labeling for time points and time intervals

Nicholas J. Cox
Department of Geography
Durham University
Durham City, UK
n.j.cox@durham.ac.uk

Plots of time-series data show time on one axis, usually the horizontal or x axis. Unless the number of time points is small, axis labels are usually given only for selected times. Users quickly find that Stata's default time axis labels are often not suitable for use in public. In fact, the most suitable labels may not correspond to *any* of the data points. This will arise when it is better to label longer time intervals, rather than any individual times in the dataset.

For example,

```
. webuse turksales
```

reads in 40 quarterly observations for 1990q1 to 1999q4 with a response variable of turkey sales. The default time axis labels with both `line sales t` and `tsline sales` are 1990q1, 1992q3, 1995q1, 1997q3, and 2000q1. These are not good choices for any purpose, even exploration of the data in private.

Label choice is partly a matter of taste, but you might well agree with Stata that labeling every time point would be busy and the result difficult to read. With 40 quarterly values, possible choices include one point per year (10 labels) and one point every other year (5 labels). One possibility is to label every fourth quarter, as that is usually the quarter with highest turkey sales. `summarize` reveals that the times range from 120 to 159 quarters (0 means the first quarter of 1960), so we can type

```
. line sales t, xlabel(123(4)159)
```

Note how we use a *numlist*, `123(4)159`, to avoid spelling out every value. The step length is 4 for four quarters. See [U] **11.1.8 numlist** or `help numlist` for more details of *numlist*s. This graph too would need more work before publication, as the labels are still crowded. The text of the labels (e.g., 1990q4) may or may not be judged suitable, depending partly on the readership for the graph.

However, there is another choice: label time intervals (years) and mark the boundaries between those time intervals by ticks. Consider 1990. The four quarters in Stata's units are 120, 121, 122, and 123. Thus we could put text showing the year at a midpoint of 121.5 and ticks showing year boundaries at 119.5 and 123.5. For all years, we should use the *numlist* idea again with the following command to produce figure 1.

```
. line sales t, xtick(119.5(4)159.5, tlength(*1.5))
> xlabel(121.5(4)157.5, noticks format(%tqCY)) xtitle("")
```

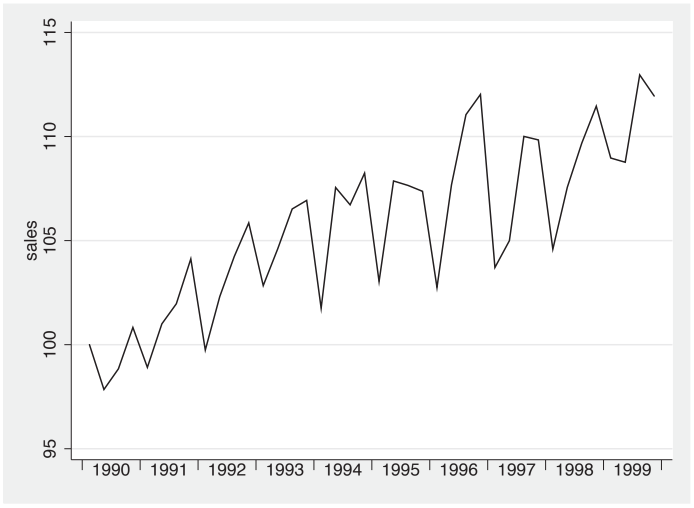

Figure 1: Turkey sales in each quarter. Time axis labels show years (with ticks suppressed) and time axis ticks show year ends.

The most important details here are suppressing the ticks for the axis labels and specifying a format for them. Cosmetic additions include lengthening the ticks compared with the default and suppressing the axis title, which would otherwise be the variable name `t` (or a variable label if it existed). It is usually clear from the labels what is being shown. Other possibilities include changing the text size for the axis label, changing the angle at which the axis label is shown, and suppressing the century by using a format like `%tqY`. Those may not be especially attractive, but nevertheless might be forced upon you by practicalities.

The main idea is clearly more general. The axis labels and the axis ticks need not correspond to each other, and it might be good to have fewer labels than ticks for longer series. Monthly and half-yearly data naturally yield to the same method, but use 12 or 2 and not 4 as the step length. Weekly and daily data are more awkward but still manageable.

If you were producing many similar graphs, you might want to automate this process to some degree. The mental arithmetic might easily be more challenging than in the turkey example. Let us imagine daily data for several years. Thus we could put ticks every January 1 and year labels every July 1. That will be adequate precision in practice. Find the first and last years in your data, if necessary by a command like `gen year = year(date)` followed by `summarize`. Suppose again that the years are 1990–1999. We can put the needed dates in local macros with a loop:

```
. forvalues y = 1990/1999 {
        local jan `jan´ `=mdy(1,1,`y´)´
        local jul `jul´ `=mdy(7,1,`y´)´
}
```

Each time around the loop the daily dates for January 1 and July 1 in each year are calculated on the fly with a call to the `mdy()` function and added to a macro. For more details, see [P] **forvalues** and [P] **macro**, the corresponding help files, or Cox (2002). Once done, the graph command is something like

```
. line whatever date, xlabel(`jul´, format(%tdCY) noticks)
> xtick(`jan´, tlength(*1.5))
```

A key requirement is that the local macros used in the graph command must be visible, by virtue of being in the same interactive session, do-file, or program. That is in essence what `local` means.

Calendar years, meaning here Western calendar years, are clearly not the only possibilities. You could use other boundaries and midpoints for years or other periods defined by other criteria (e.g., academic, financial, fiscal, hydrological, political, religious).

Reference

Cox, N. J. 2002. Speaking Stata: How to face lists with fortitude. *Stata Journal* 2: 202–222.

The Stata Journal (2007)
7, Number 4, p. 593

Software Updates

gr0012_1: Density probability plots. N. J. Cox. *Stata Journal* 5: 259–273.

The program has been updated so that users of Stata 9 and later can use an `addplot()` option.

st0133_1: Fitting mixed logit models by using maximun simulated likelihood. A. R. Hole. *Stata Journal* 7: 388–401.

New features include options for specifying weights (including sampling weights) and for obtaining robust and cluster–robust standard errors. The estimation speed has also been improved by using analytical instead of numerical derivatives when maximizing the simulated log-likelihood function. This change has the side effect of producing somewhat different estimation results compared with the previous version for some datasets and model specifications. The new `numerical` option may be used to replicate estimation results produced with the old version, but it should only be used for that purpose, as it causes the command to run slowly.