The Stata Journal

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 who teach, 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®

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 (2008)
8, Number 1, pp. 134–136

Stata tip 56: Writing parameterized text files

Rosa Gini
Regional Agency for Public Health of Tuscany
Florence, Italy
rosa.gini@arsanita.toscana.it

Stata includes several commands for text file manipulation. A good example is the `copy` command ([D] **copy**). Typing

```
. copy filename1 filename2
```

simply copies `filename1` to `filename2`, regardless of its content.

Often when dealing with text files, you need greater flexibility. Stata can also read and write text files using the `file` suite of commands ([P] **file**). Thereby, you can rewrite a text file by first reading and then writing it with modifications.

```
local filetarget "filename2"
local filesource "filename1"
local appendreplace "replace" /* append or replace */
tempname target source
file open `target´ using `filetarget´, write `appendreplace´ text
file open `source´ using `filesource´, read text
file read `source´ textline
while r(eof) == 0 {
        file write `target´ `"`textline´"´ _n
        file read `source´ textline
}
file close `source´
file close `target´
```

A notable feature of this second way of copying text files is that you can append files to existing files. Even more importantly, you need not copy the file character for character. While rewriting, Stata may substitute the values of local or global macros that have been defined. This allows users to work with a template and produce text with elements substituted for each occasion. Such files may be called "parameterized", as they contain elements constant within a document but variable from document to document.

As an example of the many applications of this simple device, consider the needs of those who periodically access big databases to produce standard reports. The structure of the database is fixed; hence, access will be by a cascade of queries. The queries will be the same every time except for some parameters that change, such as the date (e.g., month, quarter, or year). Stata can access such databases without intermediaries, as SQL code for the queries can be stored in a text file with global macros and be rewritten and executed periodically using the `odbc` command ([D] **odbc**). A file `query_para.sql` might contain the following simple SQL parameterized code:

```
CREATE TABLE ${year}_AMI AS
SELECT H.ID, H.CODE_PATIENT, H.YEAR, H.SEX, H.AGE
FROM HOSPITALIZATIONS  H, PATHOLOGIES  H_PATHOL
WHERE H.ID=H_PATHOL.ID AND H_PATHOL.DIAGNOSIS="410" AND ${conditions} AND
> H.YEAR=${year} AND H_PATHOL.YEAR=${year};

CREATE INDEX ${year}_ID ON ${year}_AMI (CODE_PATIENT)
TABLESPACE epidemiology;
ANALYZE TABLE ${year}_AMI compute statistics;

CREATE TABLE ${year}_MORTALITY AS
SELECT DISTINCT CASES.CODE_PATIENT, MOR.DEATH_DATE
FROM ${year}_AMI CASES, MORTALITY  MOR
WHERE MOR.CODE_PATIENT=CASES.CODE_PATIENT;
```

This sequence of queries looks for patients hospitalized for AMI (acute myocardial infarction, or heart attack) in a given year and then links the list of patients to the mortality records to obtain data on survival. As the list of patients may be very long, the code computes an index to perform better linkage.

The following code is a template for one Stata session. For example, substitute any `connect_options` desired for `odbc`.

```
/* set parameters */
global year = 2005
global conditions "H.AGE>64"
/* rewrite text */
local filetarget "query.sql"
local filesource "query_para.sql"
local appendreplace "replace" /* append or replace */
tempname target source
file open `target´ using `filetarget´, write `appendreplace´ text
file open `source´ using `filesource´, read text
local i = 1
file read `source´ textline
while r(eof) == 0 {
        file write `target´ `"`textline´"´ _n
        local ++i
        file read `source´ textline
}
file close `source´
file close `target´
/* execute queries */
odbc sqlfile("query.sql"), dsn("DataSourceName") [connect_options]
/* load and save generated tables */
foreach table in AMI MORTALITY {
        odbc load table("${year}_`table´"), clear [connect_options]
        save ${year}_`table´, replace
}
```

The code will make Stata

1. Write a text file of actual (nonparameterized) SQL code, where `${year}` is substituted by 2005 and `${conditions}` is substituted by `"H.AGE>64"`.

2. Execute the SQL code via `odbc`. This may take some time. If an SQL client is available, a practical alternative is to make Stata call that client and ask it

to execute the SQL using the **`shell`** command ([D] **shell**). This will make the execution of the queries independent of the Stata session.

3. Load the generated tables.

4. Save each of the generated tables as a Stata `.dta` file for later analysis.