Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System
  - Strengthening Tools

A SAS User's Guide to Stata

This guide is intended to introduce a SAS user to Stata and to be a cheat sheet when writing Stata code.

- Basic Conceptual Differences Between **SAS** and **Stata**.
- Stata's Windows Mode
- **SAS** code matched to **Stata** code
- **SAS** functions matched to **Stata** functions
- How to submit **Stata** do-files in batch mode

Tools to transfer data between SAS and Stata

(all require that both SAS and Stata are able to be invoked on the same computer)

- SAVASTATA: SAS macro to save a SAS dataset as a Stata dataset.
- SAVASAS: Stata command to save a Stata dataset as a SAS dataset.
- USESAS: Stata command to use a SAS dataset while in Stata.
- USESTATA: See the example in the STATA WRAPPER page that shows you how to create a SAS dataset from a Stata dataset.
- SAVAS (c-shell) script: Invoked at a UNIX/Linux prompt makes Stata datafile copies of SAS datafiles or SAS datafile copies of Stata datafiles.
- STATA_WRAPPER: SAS macro to run Stata code from within SAS.
- SASWRAPPER: Stata command to run a SAS program from within Stata.

SAS macros inspired by Stata programs

- CHAR2FMT: SAS macro to encode character variables.
- RUNQUIT: SAS macro to stop SAS from processing more code when an error occurs without ending your SAS session.
- ISID: SAS macro to test if variable(s) uniquely identify each observation in a dataset.
- MINIME: SAS macro to reset variable lengths to minimize the SAS dataset.
- WHICH: Find out what SAS macro SAS will invoke.
- REN_STEM: SAS macro to rename variables.
- TYPE: SAS macro to print a text file to your SAS output window.

Stata programs

- ADOEDIT: Stata command to edit an ado-file in Stata's do-file editor
- ETIME: Stata command to keep track of elapsed time
- IFWINS: Stata command to subset dataset by if exp before subsetting by in range
- LOOKFORIT: Stata command to search variable names and variable labels for some string or strings
- LOOKFOR_ALL: Searches for variables/patterns in Stata data files (.dta)
- LSTRFUN: Modify long local macros that contain strings
- MKPROFILE: Stata command to create or edit your profile.do
- NSPLIT: Stata command to split numeric variables with integer values into new smaller numeric variables
- RCD: Recursively traverse directories and execute any command in or for each one of them

Tools to transfer data between SAS matrix catalog files and SAS datasets

These SAS macros use **PROC IML** and are a great way to migrate a SAS matrix catalog file between operating systems since SAS catalog files can only be opened in the operating system they were created on.

- MAT2DAT: SAS macro to create SAS datasets from a SAS matrix catalog file
- DAT2MAT: SAS macro to create a SAS matrix catalog file from SAS datasets

SAS code matched to Stata code

How to "Do" in Stata What You Know How to "Program" in SAS Stata code matched to SAS code as closely as it can be. Use your browser's search tool to find the SAS code for which you want the Stata equivalent.

Note: Stata commands are partially underlined to show the minimal characters that need to be typed for Stata to recognize that command.

SAS

SAS is distinct capitalization (all its symbols or mnemonic keywords such as

```
a
```

```
b
```

```
cmd
```

For many situations in SAS order does not matter:

```
cmd var1 var2
```

```
cmd var1
var2
```

```
/* this is a comment */
/* this is also a comment
```

```
/* this is a comment */
/* this is also a comment
```

```
/* this is a comment */
/* this is also a comment
```

Range of values:

```
if a1 in (a, b) and a2 IN
```

```
if var_a1 in(1,2,3,4,5,6,7,8,9)
```

or a list of character values:

```
if stata_var1 IN( "AL", "AK", "AR", "AZ", "CA", "CO",
```

Referencing multiple variables at a time:

In the SAS language variables are not allowed in the order shown:

```
var1 var2 var3 var4 var5
```

This is a SAS rule even from 1:

```
var1 -num- var5
```

In SAS, this means: list variables that are positionally between var1 and var5 inclusive, regardless of their names:

```
var1 var2
```

In the same vein:

```
var1 var2 var3 var4 var5
```

no matter the positions of the variables are in the dataset.

Using a colon selects variables conditioning the name prefix:

```
var1 var2 var3 var4 var5
```

Local macros:

```
var1 var2 var3 variable varying var_1
```

To save the contents of the log window across Output windows, go to log window, right click, select "Save Contents" and save the file. Output will be saved in the file "log.log". Files are automatically generated for you. You can also use PROC PRINTTO to write the log to a file. Output is saved in the current directory unless a full path is specified. Output is saved in the files listed in the output window. Use the "Save" button to save the output in the current directory.

```
proc printto log= "C:\data\log.log" par
proc printto log=log;
proc print data= sashelp.class;
run;
/* specifying nothing sends all writing to the terminal (log
proc printto;
run;
```

```
libname lh "c:\data\";
proc print
lh.mydatasetlibrary;
run;
```

Looping in SAS is

```
data data;
do i= 1 to 10;
output;
end;
run;
```

```
proc contents;
run;
```

In selected variables

```
proc contents data= in.sdmv2 (keep= id date gender age income);
run;
```

```
proc means; var;
run;
```

In selected variables

```
proc means
var age income;
run;
```

```
proc univariate;
var age income;
run;
```

```
proc freq
table var1;
run;
```

```
proc freq
table var1;
run;
```

A series of 1-way tables

```
proc freq;
tables var1 var2;
run;
```

A 2-way table

```
proc freq;
tables var1*var2;
run;
```

```
proc print; var;
```

Selected variables in the order

```
proc print
var var1 var2 var3;
```

In selected variables and a limited range of observations

```
proc print data= in; var id income 1 obs 20;
run;
```

Create a system variable with a default length of 8 bytes

```
data= life
```

Create a system variable with the minimum allowable length (2 bytes)

```
length var1 $ var2= life
```

```
proc print
var var1 var2 var3 var4 var5;
run;
proc freq
table var1 var2 var3 var4 var5;
run;
proc means
var var1 var2 var3 var4 var5;
run;
proc univariate
var var1 var2 var3 var4 var5;
run;
```

Example of an **if-then-else**

```
if a1 in (a,b) or b1 IN( stata );
if a1 in (a,b) and b1 IN( stata );
if a1 in (a,b) and b1 IN( stata )
if a1 in (a,b) and b1 IN( stata )
else stata = 1;
```

```
proc print; var;
```

Selected variables in the order

```
proc print
var var1 var2 var3 var4 var5;
```

In selected variables and a limited range of observations

```
proc print data= in; var id income 1 obs 20;
run;
```

Create a system variable with a default length of 8 bytes

```
data= life
```

Create a system variable with the minimum allowable length (2 bytes)

```
length var1 $ var2= life
```

```
proc freq
table var1 var2 var3 var4 var5;
run;
proc means
var var1 var2 var3 var4 var5;
run;
proc univariate
var var1 var2 var3 var4 var5;
run;
```

Example of an **if-then-else**

```
if a1 in (a,b) or b1 IN( stata );
if a1 in (a,b) and b1 IN( stata )
if a1 in (a,b) and b1 IN( stata )
else stata = 1;
```

Stata's equivalent of the **if-then-else**

```
if a1 in (a,b) or b1 IN( stata ) {
if a1 in (a,b) and b1 IN( stata ) {
if a1 in (a,b) and b1 IN( stata ) {
else {
stata = 1
}
```

Stata's equivalent of the **if-then-else** in a single line of code:

```
if a1 in (a,b) or b1 IN( stata ) ; else stata = 1;
```

Stata's equivalent of the **if-then-else** in a single line of code:

```
if a1 in (a,b) or b1 IN( stata ) ; else stata = 1;
```

```
proc contents
```

```
proc contents
```

```
proc means; var;
```

```
proc means
var var1 var2 var3 var4 var5;
```

```
proc means
var var1 var2 var3 var4 var5;
```

```
proc freq
table var1;
```

```
proc freq
table var1;
```

```
proc freq;
tables var1;
```

```
proc freq;
tables var1 var2;
```

```
proc freq;
tables var1*var2;
```

```
proc freq;
tables var1*var2;
```

```
proc freq;
tables var1*
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Basic Conceptual Differences Between SAS and Stata

To learn Stata you will nearly have to unlearn SAS. Think of this page as the "door" to the Stata mindset.

SAS	Stata
There is only one observation in memory at a time. Typically, you can only access the values of variables on the current observation.	The whole dataset is in memory at a time. You can access the values of variables in any observation in the whole dataset. For this reason the **RETAIN** statement makes no sense in Stata. But the norm is to the variable value of the current observation is used.

A series of statements modify the dataset one observation at a time.	One command at a time modifies the dataset one observation at a time, then the next command, etc.
	Think of each command as a data step with only one statement in it:
	`data mydata;`
	`set mydata;`
	`some statement;`
	`run;`

In general, you never need to worry about how much memory SAS will require during your session.	If you are using a previous version of Stata to version 12, then Stata requires the user to be aware of how much of the computer's memory will be required to load the dataset into memory. Stata's default setting for memory may not be set to a large enough setting. A quick rule of thumb would be to set Stata's memory to 20% more than the size of the dataset. If the dataset is smaller than 30 megabytes simply set the memory to 40 megs.
	`set memory 40m`
	To make this setting to be your default every time, you can do one of two things, but not both:
	1) Use the "`permanent`" option:
	`set memory 40m, permanent`
	2) Or click the Stata icon, right click the icon, choose properties, and edit the target field to look something like this:
	`C:\Stata\stata.exe /m40`
	"`/m40`" sets the memory to 40 megabytes.
	In Linux/UNIX you would have to assign an alias to call Stata that would look something like this:
	`alias xstata='xstata -m40'`

For numeric variables, missing values (.) are the smallest values.	For numeric variables, missing values (.) are the LARGEST values. Starting in version 8, Stata offers special missing values `.a` through `.z` where (.) is less than `.a` and `.z` is the largest value.
Using capital or lower case while programming does not matter. For example variable `Age` is the same variable `age` or `aGe`, etc.	Case matters. Lower case is most often used. For example, variables `Age`, `age` and `aGe` are all different variables, albeit not very good ones. All Stata commands are in lower case.

The end-of-line delimiter is the semi-colon (;).
`if oldvar1 = 2 and oldvar2 = 1 then newvar = 1;`
	The default end-of-line delimiter is the carriage return, though you can set the end-of-line delimiter to be the semi-colon with the "#delimit ;" command.
	Since the carriage return tells Stata where the command ends, the "#delimit ;" command allows you to write a command that spans many lines in your do-file. "#delimit cr"returns Stata to using carriage returns.
	Starting with Stata 8, you can just use 3 forward slashes at the end of the line to tell Stata to continue to the next line before finishing the command:
	`generate newvar= 1 if oldvar1 == 2 ///`
	`& oldvar2 == 1`

Conditions come first and then something happens. For example: `if age>10 then age1=1;`	Conditions come at the end of a command. For example: `generate age1= 1 if age > 10 & age < .` Remember to consider that the variable `age` may equal missing for some cases and that missing values are greater than any other number.
All statements have to be spelled out completely and usually spelled correctly for SAS to recognize the statement.	Stata recognizes a command when it is spelled out enough for Stata to understand what command you intend. For example, the command generate can be typed only as `gen`. In Stata documentation you will notice commands with the first few letters underlined which indicates what the minimum characters needs to be typed.
If you want to keep a record of your SAS session in a log file, you save your log window when you are finished with your SAS session.	If you want to keep a record of your Stata session in a log file, Stata requires you to decide to do that at the beginning of the session you want to log. You can control when you are logging to the same file or to different files, but you can only log to one file at a time. In Stata batch mode, a log file is automatically created just like it is with SAS in batch mode.
If you want to create a file that contains a series of SAS statements (a SAS program), create a normal ASCII text file with the file extension "`.sas`" like "`my_prog.sas`". An ASCII text file is a basic text file that can be opened with any text editing software like WordPad, NotePad, pico, vi, etc.	If you want to create a file that contains a series of Stata commands (a Stata do-file), create a normal ASCII text file with the file extension "`.do`" like "`my_dofile.do`".

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Stata's Windows Mode

If you are not familiar with Stata's interactive mode, launch Stata and take a moment to have a look at Stata's layout.

- Stata's **Results** window is a mix of SAS's log and output windows. The results window reprints the command and Stata's reaction to that command.

- The **Stata Command** window is almost like SAS's program editor except that it is designed for the user to submit only one command at a time.

  Helpful tip: to scroll back to previously submitted commands use the "Page Up" key.

- The **Variables** window lists the variable names and labels of the variables in the data set that is currently in memory. If you click on a variable it will appear in the command line.

- The **Review** window lists recently submitted commands from the command window. If you click on a previous command, it will appear in the command window. If you double click it will be executed.

  Helpful tip: If you want to save the commands in the review window, perhaps because you did not start logging your Stata session, click to the left of the word "Review" in the title of the window and select "Save Review Contents". This will save the commands in the order they were submitted in the form of a do-file (`*.do`) which is equivalent to a SAS program (`*.sas`) file.

- The **Stata Do-file Editor** (in versions previous to Stata 10 click the icon of an open envelope. starting in Stata 10 the icon is a notepad) is like SAS's program editor. Here you can submit a single command or several lines of code at a time. To submit commands from the do-file editor click the icon in the tool bar that looks like a piece of paper with lines on it and an arrow pointing down.

- Stata also offers a **Data Editor** and a **Data Browser** just like SAS. To access them there are icons in the tool bar that look like spread sheets. The icon for the data browser is the one with the magnifying glass.

- The **Stata Viewer** is for viewing existing log files as well as help files. To launch the viewer click the icon in the tool bar that is a picture of an eye.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

SAS Functions Matched to Stata Functions

SAS functions SAS Institute's Online SAS documentation	Stata functions
`byte()` lets you use ASCII code to represent characters.	char()
`ceil()` returns the smallest integer that is greater than or equal to the value of the number or numeric variable.	ceil()
`compbl()` replaces multiple blank spaces with just one in a character string.	itrim()
`countw()` counts how many words are in a character string. The default delimiter is not just a blank space.	wordcount()
`day()` returns what the day of the month (1-31) is from the numeric date variable.	day()
`floor()` returns the largest integer that is less than or equal to the value of the number or numeric variable.	floor()
`in()` is true if the value of the preceding variable has the value of at least one value listed.	inlist()
`index()` returns the first position of a character in a character string. `find()` is similar.	strpos()
`indexw()` returns the first position of the first character of a word in a character string. `findw()` is similar.	There is no equivalent Stata function, but regexm() can at least be used to test if the word exists in a string.

`input()` creates numeric data from character data.
`input()` can be used like Stata's date():
`acquisition_date= "25Apr2001";`
`new_date= input(acquisition_date, date9.);`
`format new_date date9.;`
	real() but the destring command is likely a better solution.
	`// Stata 10 or higher`
	`date(acquisition_date, "DMY")`
	`// earlier Stata versions`
	`date(acquisition_date, "dmy")`
	real() cannot have formats specified like SAS's `input()`.

`int()` returns the integer value of the number or numeric variable.	int()
`left()` removes leading blank spaces (on the left-hand side) of a character variable.	ltrim()
`length()` returns how long a string is.	length()
`lowcase()` REPLACE ALL CHARACTERS IN A STRING with lowercase characters.	lower()
`max()` returns the largest value in a list of numeric variables.	max()
`mean()` returns the average of the arguments.	egen's `rowmean()`
`mdy()` creates a numeric date variable from month, day and year variables.	mdy()
`min()` returns the smallest value in a list of numeric variables.	min()
`missing()` is true (returns the value 1) if the argument/variable has a missing value. `nmiss()` can be used like Stata's missing() to check if any of a set of variables have missing values but it can only handle numeric variables. Starting in SAS 9.2 `cmiss()` in that it counts how many character or numeric variables listed have missing values.	missing()
`mod()` returns the remainder from the division of the first argument by the second argument.	mod()
`month()` returns the month of the year from a numeric date variable.	month()
`propcase()` replaces the first letter of a word With A Capital Letter and the following letters to lowercase.	proper()

`prxchange()` lets you use regular to match patterns in strings and replace them with whatever.	regexr()
	regexs() lets you use the group variables created in regexm()
	`// change "Smith, John" into "John Smith"`
	`replace name= regexs(2) + regexs(1) ///`
	`if regexm(name,"(.*),(.*)")`

`prxposn()` returns the character string captured by a group in regular expression. `prxposn()` requires a regular expression id variable.
`** create at most 3 groups from characters`
`* that were separated by at least 1 space: *;`
`reg_id= prxparse("/([^ ]+) ([^ ]+) ([^ ]+)/");`
`if prxmatch(reg_id, my_char_var) then do;`
`word1= prxposn(reg_id, 1, my_char_var);`
`word2= prxposn(reg_id, 2, my_char_var);`
`word3= prxposn(reg_id, 3, my_char_var);`
`end;`
	regexs() but the split command is easier to use and is more powerful in that it can create as many new string variables that are needed.

`put()` creates character data from numeric data.	string() but the tostring command is likely a better solution.
`ranuni()` returns a random number between 0 and 1.	runiform() starting in Stata 10 or `uniform()` in previous versions.
`right()` removes trailing blank spaces (on the right-hand side) of a character variable.	rtrim()
`round()` returns the closest integer value of the number or numeric variable.	round()
`scan()` return the nth word in a string.	word()
`strip()` and `trim()` remove both leading and trailing blanks but `strip()` will leave no spaces if the character value has only spaces where `trim()` will leave 1 space.	trim()
`substr()` returns a substring of characters from a string.	substr()
`translate()` replaces all occurrences of a character in a string with whatever. You cannot specify to only replace the first N occurrences like you can with Stata's subinstr().	subinstr()

`tranwrd()` replaces all occurrences of a substring of characters in a string with whatever. You cannot specify to only replace the first N occurrences of the string nor only replace a word like you can with Stata's subinword(). Using `tranwrd()` like so:
`string_var= "Good Partnership";`
`tranwrd(string_var,"art","");`
will change `string_var` to `"Good P nership"`. Stata's subinword() would not modify `string_var` because `string_var` did not have the string `art` as a word (starting the string and followed by a space or ending the string and being preceded by a space or being surrounded by spaces).
	subinword()

`upcase()` replaces all characters in a string WITH UPPERCASE CHARACTERS.	upper()
`year()` returns the four digit year (century year) from a numeric date variable.	year()

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

How to Submit a Do-file in Batch Mode

The benefits of batch mode are many: 1. Your code is processed faster. 2. Your log and output windows are automatically saved in *.log and *.lst files. 3. Every time you submit your program you know you are starting off with a clean slate. 4. All your friends will think you're cool.

In Stata your results window is automatically saved to an ASCII text log file. You can choose to have your results saved in a `*.smcl` file if you use the `log using` command in your do-file and specify a filename that does not have the extension of "`.log`". The smcl file documents the results window down to the colors the text is printed in. To do this Stata uses smcl tags which are a lot like html tags. The only useful way to view this file is to use Stata's viewer. If you specify that you want your file to be saved as a `*.log` file, then it will be saved as a normal ASCII text file without the smcl tags.

SAS	Stata
In Linux/UNIX at a prompt type: `sas myprogram.sas`	In Linux/UNIX at a prompt type: `stata -b do mydofile.do`

In Windows right-click on the program file listed in the folder that you want to submit and choose "**Batch Submit**".	In Windows you right-click on the do-file listed in the folder that you want to submit and choose "batch" if it is available. If it is not listed, you can add the following to your folder options file types listing for files that end in "`.do`" if you are using Windows XP:
	`C:\Stata\[whatever your Stata executable file is named].exe /m40 /b do "%1"`
	
	"`/m40`" sets the memory to 40 megabytes.
	
	If you are using a more current version of Windows you can setup a "Stata batch" icon in your SendTo directory: `C:\Users\MyUserId\AppData\Roaming\Microsoft\Windows\SendTo` . The AppData directory is likely to be a hidden directory. Create a shortcut to the Stata executable you want to run and then edit the shortcut by right-clicking on it and choose Properties.&nbps; Then have the Target box have the same settings as shown above but without the `"%1"`
	`C:\Stata\[whatever your Stata executable file is named].exe /m40 /b do`
	
	Then edit the "Start in" box to be whatever directory you would prefer to have as the current directory for your Stata batch session. Then when you want to run a do-file in batch mode, you need to right-click on the file and chose SendTo and then click on the name of the shortcut you created.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
        - SAVASTATA
        - Why use SAVASTATA?
        - SAVASAS
        - USESAS
        - SAVAS (c-shell) script
        - STATA_WRAPPER
        - SASWRAPPER
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Tools to transfer data between SAS and Stata

SAVASTATA
SAS macro to save a SAS dataset as a Stata dataset

Why use SAVASTATA?

SAVASAS
Stata command to save Stata datasets as SAS datatsets.

USESAS
Stata command to load SAS datasets into Stata. Once your data is in Stata it is a Stata dataset and should be treated as such. Check out Stata's nearly equivalent command fdause that allows you to use SAS transport/xport datasets.

SAVAS (c-shell) script
Save SAS datasets as Stata datasets/Save Stata datasets as SAS datasets. Based on the file extensions savas... makes SAS (version 6.09 or later) datafile copies of Stata (any version up to and including the version of Stata being run) datafiles. or: makes Stata (version 7 or later) datafile copies of SAS (version 6.09 or later) datafiles.

STATA_WRAPPER
SAS macro to run Stata code from within SAS

SASWRAPPER
Stata command to run a SAS program from within Stata

Research
- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
        - SAVASTATA
        - Why use SAVASTATA?
        - SAVASAS
        - USESAS
        - SAVAS (cshell) script
        - STATA_WRAPPER
        - SASWRAPPER
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

SAVASTATA

SAS macro to save a SAS dataset as a Stata dataset

If you want to use **SAVASTATA**, then feel free to download load it here:

Download savastata.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `savastata.sas` file as a plain text file not an htm/html file.

The best way to download **SAVASTATA** is to use Stata's command ssc install to install usesas because usesas uses **SAVASTATA** to load a SAS dataset into Stata. The `savastata.sas` file will be saved in whatever directory Stata saves downloaded files that begin with the letter "s". The Stata command findfile will help you find the file `savastata.sas`.

Why use SAVASTATA?

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About SAVASTATA

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 20Oct2003
Last updated: 28Mar2011

Save the most recently created SAS dataset in the WORK library to a Stata dataset

```
%savastata( "directory to save Stata dataset", options );
```

Dependency: **SAVASTATA** requires that you have a working copy of SAS and a working copy of Stata Intercooled, SE or MP on your computer to run successfully. If your SAS dataset is small enough **SAVASTATA** may work on Stata Small (Student Version).

Description

SAVASTATA uses the most recently created dataset in the WORK directory, figures out how best to store the data as a Stata dataset, keeps most date formats and other basic formats for numeric variables, checks for invalid Stata variable names, if using or outputting to Stata 6: checks to see if long variable names need to be shortened, prints them to the SAS log and shortens them, writes out the SAS dataset to an ASCII datafile, attempts to preserve all user-defined formats for numeric variables, writes out Stata do-files, and submits them to Stata in batch mode in order to have Stata read the data in and saves the data as a Stata dataset.

If your SAS dataset is using formats that are in a formats catalog, **SAVASTATA** will make an attempt to preserve them as value labels in Stata. Stata does not allow all the variations of user-defined formats that SAS offers.

Restrictions of each version of Stata:

1. Variable names can be no more than 8 characters long: Stata 6
2. Variable names can be up to 32 characters long: Starting with Stata 7
3. String variables can contain a maximum of 80 characters: Stata 6, 7 and 8 Intercooled.
4. String variables can contain a maximum of 244 characters: Stata 7 and 8 SE and Intercooled starting with verison 9.
5. The maximum number of variables is 2,047: Stata Intercooled
6. The maximum number of variables is 32,767: Stata SE and MP starting with version 7
7. Special missings (**.** through **.z**) for numeric data are allowed: Starting with Stata 8

Options

NOTE: Options can be used in any order. You can specify as many as you want or none at all.

The following three options work as they do in Stata:

-replace

If the Stata dataset you want to output already exists, then overwrite it with the dataset generated by **SAVASTATA**.

-old

Outputs a dataset to the previous version of Stata. **SAVASTATA** uses the Stata command saveold when this option is specified. **NOTE**: Stata 9 SE or MP can only save to Stata 7 Intercooled and not to Stata 7 SE.

intercooled

Outputs a Stata Intercooled dataset if using Stata SE or MP. Starting with Stata 10 it only checks that your dataset has no more than 2,047 variables. It does not save the dataset as an intercooled dataset since starting with Stata 10 that is no longer an option.

NOTE: If you do not specify what version of Stata to save in, Stata will save the dataset in the current version.

-check

Creates two check files for the user to compare the SAS input dataset with the Stata output dataset to make sure **SAVASTATA** created the files correctly. This is a comparison that should be done after any datafile is converted to any other type of datafile by any software. The files are created in the same directory as the output Stata datafile and are named starting with the name of the datafile followed by either "`_SAScheck.lst`" or "`_STAScheck.log`". The SAS check file contains output from **PROC MEANS**, **PROC CONTENTS** (by position), and a **PROC PRINT** of the first 5 observations. The Stata check file contains the equivalent output by the Stata commands summarize, describe, and a list of the first 5 observations.

-float

Numeric variables that contain decimals will be stored as **float** instead of the default of **double**. This may result in a loss of precision, but **float** is the default storage type that Stata uses. This will help decrease your file size.

-quotes

Replace double quotes (") occurring in character variables with single quotes (') and replace compound quotes (`" or "' ) occurring in variable labels or formats with single quotes ( `). **SAVASTATA** cannot always process character variables with double quotes or variable labels or formats with compound quotes.

-messy

Puts the intermediary files generated by **SAVASTATA** used to create the Stata dataset in the directory named in the pathname provided in the call to the **SAVASTATA** macro. This is a debugging option.

-ascii

Outputs only the ASCII datafile and does not save your dataset in Stata format. It also turns on **-messy** switch so that the Stata input files are not deleted after your program has run. Use this switch to make your own edits to the input of these data into Stata. Use this option if you do not have Stata on the same computer that you have SAS. The files generated by **SAVASTATA** can be moved to another computer that does have Stata and run there to create the Stata dataset.

The default version of Stata that your ascii file will be set to be used for will be Stata 6 (only allow up to 2,047 variables).

If you want to specify what version of Stata will be used to read the ascii data you can do so like so:

```
%savastata("C:\my_data\", -ascii9);
```

Here are more examples of the -ascii option:

```
%savastata("C:\my_data\", -ascii9SE);     **  (for Stata 9 SE/MP) **;
%savastata("C:\my_data\", -ascii9);       **  (for Stata 9 Intercooled)  **;
%savastata("C:\my_data\", -ascii10MP);    **  (for Stata 10 SE/MP) **;
%savastata("C:\my_data\", -ascii8);       **  (for Stata 8 Intercooled)  **;

%savastata("C:\my_data\", -ascii9.2SE);   **  IS AN INVALID VERSION,
                                          *     just specify: -ascii9SE **;
%savastata("C:\my_data\", -ascii 9.2 SE); **  IT IS INVALID TO USE SPACES,
                                          *     just specify: -ascii9SE **;
```

How to use the SAVASTATA macro:

Using the **SAVASTATA** macro requires that you understand how to use the "%include" SAS statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "C:\SASmacros\", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\savastata.sas";
```

The include statement makes SAS aware of the **SAVASTATA** macro which is in the file `savastata.sas`. To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program.

```
%savastata(C:\mySASdir\, -old);
```

The information inside the parentheses is passed on to the **SAVASTATA** macro. The first string of information is the location you want to save your SAS dataset as a Stata dataset. This is somewhat like a **libname** statement. The second string of information is the options you wish to pass on to the **SAVASTATA** macro. You can use as many options as you like or none at all.

Examples

```
%include "C:\SASmacros\savastata.sas";  ** Include macro once in a SAS session and call it
                                         *  as many times as you like in that session. **;

data work.ToBeStata; ** This makes a copy of the SAS dataset in the in the WORK library. **;
 set in.mySASfile;
run;

options xmin; ** If you use this SAS option in Windows, you will not see all the pop-up windows when SAS invokes St
%savastata(C:\mydata\,);   ** Saves the dataset in the c:\mydata\ directory if it does not
                            *  already exist in that directory. **;

%savastata(C:\mydata\, -replace);   ** Saves the dataset c:\mydata\, overwriting it if
                                     *  it already exists. **;

%savastata(C:\data\, -old);      ** Saves the dataset as the previous version of Stata
                                  *  in c:\data\  directory **;

%savastata("C:\data\", -old -replace);  ** Saves the dataset as the previous version of Stata
                                         *  in c:\data\  directory, overwriting it if it
                                         *  already exists. **;

%savastata(C:\data\, -intercooled);  ** Saves the dataset as the Intercooled version of Stata
                                      *  in C:\data\  directory.  This is only possible if
                                      *  your version of Stata is an SE edition.  Starting with
                                      *  Stata 10 it only checks that your dataset has no more
                                      *  than 2,047 variables. **;


** This is a UNIX/Linux example: **;
%savastata(/project/data/, -old -intercooled); ** Saves the dataset as previous version of
                                                *   Stata Intercooled in the /project/data/
                                                *   directory **;
```

Setting up SAVASTATA

These are instructions to edit the `savastata.sas` file.

NOTE: If you are setting up this macro on your computer for the first time, you can choose which version of Stata you are going to have **SAVASTATA** use. If you do not choose to set the following switches, **SAVASTATA** will figure out what version of Stata you are running for you. This may add a noticeable amount of time to processing so you may want to set these switches to the correct version of Stata. You can easily figure out what version of Stata you are using by looking at the top of your results window in Stata or by typing in the Stata command about at the Stata command line. One advantage of leaving **SAVASTATA** to figure out what version of Stata is being used is that when you upgrade your version of Stata you will not have to update **SAVASTATA**.

```
***  Need to set all if you set any. **;
%let u_ver=8.0; *** version of Stata that is being used. decimal values are okay. *;
                 **  set this if you do not want Stata to be run just figure out what *;
                 **  version of Stata you are using. *;
%let s_ver=8.0; *** version of Stata that data will be saved as. *;
                 **  set this only if you set the u_ver variable. *;
                 **  old option will change this if used **;
%let u_SEver=0; *** 1 if SE/MP version of Stata that is being used. *;
%let s_SEver=0; *** 1 if SE/MP version of Stata that data will be saved as. *;
                 **  intercooled option will change this if used **;
```

NOTE: If you are running **SAVASTATA** on the UNIX/Linux platforms then you need to be able to start a Stata batch job by:

```
$  stata -b do "mydofile.do"
```

If not then change the setting of the **ustata** macro variable.

```
%let ustata=stata;
/* This may work:
%let ustata=/usr/local/stata/stata;  */
```

If you are running **SAVASTATA** on the Windows platform, you may need set the location of the Stata executable file: `stata.exe` in the **SAVASTATA** SAS macro. If you do not know where your `stata.exe` is located, find your Stata short-cut icon, right-click on it, choose "properties", and look in the "target" field. This will show you where the `stata.exe` is located on your hard drive.

Change what is inside the parentheses to the location of your `stata.exe` file:

```
%let ustata=%nrstr(C:\Stata\stata.exe);  ** This may work. **;
```

In earlier versions of Stata, the Windows Stata executable file is named `wstata.exe`.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
        - SAVASTATA
        - Why use SAVASTATA?
        - SAVASAS
        - USESAS
        - SAVAS (o-shell) script
        - STATA_WRAPPER
        - SASWRAPPER
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Why use SAVASTATA?

SAVASTATA takes advantage of the abilities of SAS and Stata to transfer data from SAS to Stata where other transfer programs do not have SAS or Stata available to them.

- **SAVASTATA** lets SAS create the transfer file and Stata create the binary file where other transfer software have to guess at how these files are constructed. Normally the way to get a Stata datafile copy of a SAS file you have to first save your SAS dataset to your hard drive and the run it through DBMS/Copy or Stat/Transfer. Then you have two files when it's very likely you only need the one Stata dataset. **SAVASTATA** cuts out the "middle-man" and saves only one dataset...the desired Stata dataset. **SAVASTATA** saves disk space and reduces clutter.

- **SAVASTATA** documents in your SAS log how the Stata dataset was created, how many observations and variables it has and what version of Stata it is.

```
NOTE: SAVASTATA has successfully saved the
Stata 11 Intercooled datafile C:\my project\data\mealfreq.dta.   *
Stata reports that the dataset has 200 observations and 11 variables.   *
NOTE:  SAVASTATA took less than a minute to run.   *
```

- **SAVASTATA** makes conversion of user-defined formats to Stata value labels easy. If you have created user-defined formats in your SAS program and associated them to variables in a data step, then **SAVASTATA** will create Stata value labels from them. If SAS knows about the user-defined formats, then **SAVASTATA** will save them as value labels. No need to go into any great effort to save formats to a formats catalog or dataset to save them as value labels.

- **SAVASTATA** by default saves to the current version of Stata that you have on your computer. There is no need to try to decide if you need a "Stata SE doubles" dataset like in DBMS/Copy. By default **SAVASTATA** saves to the version of Stata that you have installed.

- **SAVASTATA** saves your data to variables set to optimum data types.

- **SAVASTATA** can been used on all versions of Windows as well as UNIX and Linux.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
        - SAVASTATA
        - Why use SAVASTATA?
        - SAVASAS
        - USESAS
        - SAVAS (cshell) script
        - STATA_WRAPPER
        - SASWRAPPER
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

SAVASAS

Stata command to save Stata datasets as SAS datatsets.

If you want to use **savasas**, then feel free to use Stata's command

ssc install to download and install **saswrapper**:

```
ssc install savasas , replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About savasas

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Date: 27Oct2003

Last updated: 16Aug2010

Save a Stata dataset as a SAS dataset

```
savasas [varlist] [using filename ] [if exp] [in range] [ , replace formats rename check type(
version_of_sas_data_file ) messy sascode ]
```

Description

savasas saves the Stata dataset in memory as a SAS file. By default the output dataset will have the same name and be in the same directory, but with the "`.sas7bdat`" file name extension and contain all observations and every variable in the Stata datafile. **savasas** can save to other versions of SAS. Version 7/8/9: "`*.sas7bdat`", Version 6: "`*.sas001`" / "`*.sd2`" , SAS 6 Transport/xport: `.xpt`.

The procedure is as follows:

1. **savasas** uses Stata's command fdasave to save the dataset in memory temporarily as a SAS xport datafile.
2. **savasas** writes a SAS input program to load the dataset into SAS and to assign variable names, labels (and formats).
3. **savasas** runs the program in SAS in batch mode to load the xport data

   Note to Windows users: This is where the Windows operating system will pop-up windows that indicate that SAS is running in batch. Do not try to close these windows as that will potentially halt **savasas**.
4. SAS saves the data as the version of SAS being run on by **savasas** unless a other version SAS file type is specified.

The maximum length for a string variable is 244 characters. In such cases, the first 200 characters will be taken and passed on to SAS (this is a limitation of the SAS xport dataset used to transfer data from Stata to SAS).

NOTE **savasas** calls SAS to run the SAS program. This requires the ability to run SAS on your computer. If you do not have a working copy of SAS, use the **sascode** option and copy the SAS program (`mydatafile_infile.sas`) and the xport datafile (`mydatafile_infile.raw`) to a computer that has a working copy of SAS. The SAS program will need to be modified to point to the location of the xport datafile.

Options

replace works like Stata's **replace** in that it allows you to overwrite the SAS datafile if it already exists.

formats specifies to save value labels that exist in the Stata dataset as SAS formats in a SAS formats in a file that will have the same name as the datafile but with the "`.sas7bcat`" file extension. This formats catalog file will be created in the same directory as the SAS datafile. To use this SAS formats catalog file you will need to add code to your program to make SAS aware that this file contains format definitions used by variables in the dataset. Your code should look something like this:

```
libname in "c:\myData";
options fmtsearch=(in.mySASdataset);
```

NOTE SAS user-defined formats have to be 8 characters or less and cannot end in a number. This is a list of format names that SAS reserves and thus cannot be user-defined format names:

```
best, binary, comma, commax, d, date, datetime, dateampm, day, ddmmyy, dollar,
dollarx, downame, e, eurdfdd, eurdfde, strong eurdfdn, eurdfdt, eurdfdwn, eurdfmn,
eurdfmy, eurdfwdx, eurdfwkx, f, float, fract, hex, hhmm, hour, ib, ibr, ieee, julday,
julian, percent, minguo, mmddyy, mmss, mmyy, monname, month, monyr, negparen, nengo,
numx, octal, pd, pdjulg, pdjuli, pib, pibr, pk, pvalue, qtr, qtrr, rb, roman, s370ff,
s370fib, s370fibu, s370fpd, s370fpdu, s370fpib, s370frb, s370frd, s370frdl, s370frds,
s370frdt, s370frdu, sxn, time, timeampm, tod, weekdate, weekdatx, weekday, worddate,
worddatx, wordf, words, year, yen, yymm, yymmdd, yymon, yyq, yyqr, z, zd
```

savasas will make some attempt to rename invalid SAS formats but it would be best for you to rename or drop them yourself before saving to SAS.

rename specifies that any required renaming of the file name and/or variable names is to be done. The **rename** option is only necessary when saving to SAS 6/transport or when variable names are not unique in SAS. When saving to SAS 6/transport **rename** attempts to rename long variable names (more than 8 characters) to be unique by shortening all long variable names to the first 8 characters or up to 7 plus a number. **savasas** displays the list of renamed variables. **rename** also renames the SAS file name when the name provided is not a valid SAS file name.

check specifies that basic stats for both datasets are to be generated to compare the Stata input dataset with the SAS output dataset in order to make sure **savasas** created the files correctly. This is a comparison that should be done after any datafile is converted to any other type of datafile by any software. The SAS file is created in the same directory as the output SAS datafile and is named starting with the name of the datafile followed by "`_SAScheck.lst`" (SAS), for example: "`mydata_SAScheck.lst`" .

type(*version_of_sas_data_file*) specifies the SAS datafile version desired, from the following list:

one of	SAS version datafile
--------	----------
sas	SAS version 9 * the default if no type specified *
sas6	SAS version 6
sasx	SAS version 6 transport/xport

Hint: If you specify the correct file name extension, you do not have to specify **type**. Remember that a SAS Version 9 (`.sas7bdat`) file will be created if no file extension or type specification is made.

SAS version 6 and Transport/xport file restrictions:

1. File names or variable names cannot be longer than 8 characters. **savasas** checks for names that are longer than 8 characters; and, if the rename option is issued, renames them to the first 8 characters or up to 7 plus a number. In addition, it displays the list of renamed variables.
2. Variable labels can be a maximum of 40 characters. **savasas** saves the first 40 characters.

messy specifies that all the intermediary files created by **savasas** during its operation are not to be deleted. The **messy** option prevents **savasas** from cleaning up after it has finished. This option is mostly useful for debugging purposes in order to find out where something went wrong. All intermediary files have a name starting with an underscore (_) followed by the process ID and are located in Stata's temp directory.

sascode specifies that only a SAS xport datafile and an input SAS program are to be created. By default, **savasas** executes all four steps outlined above. The **sascode** option aborts this process after step 2. The user then needs to read in the data *manually* using SAS. **savasas** writes a SAS program (`mydata_infile.sas`) to read in the xport datafile (`mydata.xpt`).

Examples

To save the dataset in memory to the same directory with the same name as the dataset in memory as a "`.sas7bdat`" file, type:

```
. savasas
```

To save the dataset in memory to the current directory with the name mySASdata as a "`.sas7bdat`" file, type:

```
. savasas using mySASdata
```

To save the dataset in memory to c:\data\ directory with the name mySASdata as a "`.sas7bdat`" file and create 2 check files, type:

```
. savasas using "c:\data\mySASdata", check
```

To save only variables between **var1** and **var20** from the dataset in memory to the current directory with the name mySASdata as a "`.sas7bdat`" file and create a "`formats.sas7bcat`" file containing SAS formats of any Stata value labels, replacing an existing file if necessary, type:

```
. savasas gender-g32_b using mySASdata, replace formats
```

More examples:

```
. savasas using mySASdata.sas7bdat, replace

. savasas in 1/30 using mySASdata, type(sas6) check

. savasas using mySASdata, messy type(sasx) rename
```

Setting up savasas

If you are setting up this program on your computer for the first time, you may need to edit the `sasexe.ado` file to set the location of your SAS executable file (`sas.exe`) . If you do not, **savasas** will look in the "usual" locations for it. **savasas** also may need to have the location of the SAS macro savastata.sas set. The `sasexe.ado` file is an ASCII text file and should be saved as such after editing. Stata's do-file editor will do the job.

Acknowledgements

The user-written program **outdat** inspired this program.

Research
CPC's Primary Research Areas
CPC's Signature Research Approaches
Projects
Data
Publications
Tools & Resources
Agent Based Modeling Tools
Biomedical Specimens and Population Research
Data Analysis
Stata Tutorial
Topics in SAS Programming
ML Sampling Weights
SAS Codebook
A SAS User's Guide to Stata
SAS code matched to Stata code
Basic Conceptual Differences Between SAS and Stata
Stata's Windows Mode
SAS Functions Matched to Stata Functions
How to Submit a Do-file in Batch Mode
Tools to transfer data between SAS and Stata
SAVASTATA
Why use SAVASTATA?
SAVASAS
USESAS
SAVAS (oshell) script
STATA_WRAPPER
SASWRAPPER
SAS macros inspired by Stata programs
Stata programs
Tools to transfer data between SAS matrix catalog files and SAS datasets
Data Security Plans
Secure Data Facility
M&E and Health Information System Strengthening Tools

USESAS

Stata command to load SAS datasets into Stata. Once your data is in Stata it is a Stata dataset and should be treated as such. Check out Stata's nearly equivalent command fdause that allows you to use SAS transport/xport datasets.

If you want to use **usesas**, then feel free to use Stata's command

ssc install to download and install **usesas**:

```
ssc install usesas , replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About usesas

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Last updated: 09Dec2010

Use a SAS dataset in Stata 8.1+

```
usesas using filename [ , formats char2lab check clear xport describe keep( variable names) if(SAS if statement) in(
firstobs/lastobs) quotes messy ]
```

Description

usesas loads a SAS datafile into memory. This usually occurs by supplying **usesas** a SAS dataset (`*.sas7bdat, *.sd7, *.sd2, *.sd01, *.xpt`) or an SPSS portable file (`*.por`), but **usesas** can also load a SAS dataset into memory via a SAS program (`*.sas`) that creates a SAS dataset. The last dataset created by the SAS program will be the SAS dataset processed by **usesas**. **usesas** figures out how much memory the SAS dataset will require to be loaded into Stata and sets Stata's memory for you.

usesasindicates that it has finished running by reporting to you how many observations and variables are in your dataset now in memory. For example:

```
Stata reports that the dataset has 200 observations and 11 variables.
```

usesas uses the SAVASTATA SAS macro to create the Stata dataset from the SAS dataset. Stata's net install command will download the SAVASTATA SAS macro for you.

NOTE: **usesas** calls SAS to run a SAS program. This requires the ability to run SAS on your computer.

Options

formats specifies to create value labels from SAS user-defined formats that are stored in a SAS formats catalog file that has the same name as the dataset and is in the same directory as the SAS dataset. For example: `MySasData.sas7bcat`. If this file doesn't exist, **usesas** will look for the file `formats.sas7bcat` in the same directory as the dataset.

char2lab specifies to encode long SAS character variables like the Stata command encode. Character variables that are too long for a Stata string variable are maintained in value labels. This is all done with the CHAR2FMT SAS macro.

check specifies that basic stats for both datasets are to be generated to compare the SAS input dataset with the Stata output dataset in order to make sure **usesas** created the files correctly. This is a comparison that should be done after any datafile is converted to any other type of datafile by any software. The SAS file is created in the same directory as the SAS datafile and is named starting with the name of the datafile followed by "`_SAScheck.lst`" (SAS). e.g. "`mydata_SAScheck.lst`"

clear specifies to clear the data currently in memory before running **usesas**.

float specifies that numeric variables that would otherwise be stored as numeric type double be stored with numeric type float. This option should only be used if you are certain you have no integer variables that have more than 7 digits (like an ID variable).

xport specifies that the input dataset is a SAS Transport/Xport dataset. Since there is no standard file extension for SAS Xport datasets, this option is required.

describe makes **usesas** act somewhat like the Stata command describe using. It does not bring the full dataset into memory. Instead it specifies for **usesas** only to load the descriptive information about the using dataset into Stata's memory as a Stata dataset and print it. So, instead of loading the actual dataset into Stata, **usesas** loads the descriptive information (variable names, what type of variables they are, the variable labels and formats associated to the variables) into Stata as a dataset. You can clear the descriptive data out of Stata's memory or use the descriptive data however you like to create variable lists for your actual invocation of **usesas**. This may be helpful for situations where the SAS dataset has more variables than your version of Stata can handle. You can create a variable list from the variable called "name" to create another invocation of **usesas** to read in only the variables you need.

If you do not want to have the **describe** option list the descriptive information of the imported dataset, you can use the option **listnot** with **describe**. The descriptive information will still be loaded into Stata as a Stata dataset. The descriptive data are sorted in the variable order of the using dataset so a variable list for **usesas**could be created like so:

```
. display "`= trim(name[1])'--`= name[2047]'" id--income88
```

which could then be used like so to keep the first 2,047 variables in the using dataset (2,047 is the maximum number of variables that Stata Intercooled can handle):

```
. usesas using "mySASdata.sas7bdat", clear keep(`= trim(name[1])'--`= name[2047]')
```

SAS variable lists using two dashes "--" tells SAS to use the variables that exist positionally between the first variable and the last variable in the using dataset inclusively. Read more about this under the documentation of the **keep** option. The **describe** option makes **usesas** return the following in **r()**:

Scalars:
r(N) number of observations in using dataset
r(k) **number of variables in using dataset**
Macros:
r(varlist) variables in using dataset
r(sortlist)variables by which using data are sorted

The above scalars and macros contain information about the dataset that was described, not information of the dataset of descriptive information that **usesas** loaded into Stata with the **describe** option.

keep allows for a list of variables from the SAS dataset to be read in. This list is used in the SAS code portion of **usesas** so must be written in the SAS varlist style. SAS does not allow for varlists to contain stars/asterisks (`*`) or question marks (`?`). For example:

```
keep(var1-var20)
```

includes only vars that start with "**var**" and end in a number between 1 and 20.

```
keep(var1 var20)
```

includes only vars that start with "**var**" and end in a number between 1 and 20.

```
keep(var1--var20)
```

includes only vars that in the dataset between `var1` and `var20`. This is like Stata's varlist style `var1-var20`.

if allows for a SAS **if** statement to subset the data before it's read in. Any valid SAS style **if** statement will work.

in allows for subsetting the data before it's read in. Use only `#/#` where both numbers are positive, for example `1/30` for the first 30 observations.

quotes specifies that double quotes that exist in string variables are to be replaced with single quotes. Since the data are written out to an ASCII file and then read into Stata, double quotes there are rare instances when double quotes are not allowed inside string variables.

messy specifies that all the intermediary files created by **usesas** during its operation are not to be deleted. The **messy** option prevents **usesas** from cleaning up after it has finished. This option is mostly useful for debugging purposes in order to find out where something went wrong. All intermediary files have a name starting with an underscore (_) followed by the process ID and are located in Stata's temp directory.

Examples

```
. usesas using mySASdata.sas7bdat . usesas using "c:\data\mySASdata.sas7bdat", check . usesas using "mySASdata.xpt",
```

Setting up usesas

NOTE: If you are setting up this program on your computer for the first time, you may need to edit the `sasexe.ado` file to set the location of your SAS executable file (`sas.exe`) . If you do not, **usesas** will try to set it for you. **usesas** also may need to have the location of the SAS macros `savastata.sas` and `char2fmt.sas` set. The `sasexe.ado` file is an ASCII text file and should be saved as such after editing. Stata's do-file editor will do the job.

Research
CPC's Primary Research Areas
CPC's Signature Research Approaches
Projects
Data
Publications
Tools & Resources
Agent Based Modeling Tools
Biomedical Specimens and Population Research
Data Analysis
Stata Tutorial
Topics in SAS Programming
ML Sampling Weights
SAS Cookbook
A SAS User's Guide to Stata
SAS code matched to Stata code
Basic Conceptual Differences Between SAS and Stata
Stata's Windows Mode
SAS Functions Matched to Stata Functions
How to Submit a Do-file in Batch Mode
Tools to transfer data between SAS and Stata
SAVASTATA
Why use SAVASTATA?
SAVAGAS
USESAS
SAVAS (c-shell) script
STATA_HRAPPER
SAS2HRAPPER
SAS macros inspired by Stata programs
Stata programs
Tools to transfer data between SAS matrix catalog files and SAS datasets
Data Security Plans
Secure Data Facility
M&E and Health Information System Strengthening Tools

SAVAS (c-shell) script

Save SAS datasets as Stata datasets/Save Stata datasets as SAS datasets. Based on the file extensions savas... makes SAS (version 6.09 or later) datafile copies of Stata (any version up to and including the version of Stata being run) datafiles. or: makes Stata (version 7 or later) datafile copies of SAS (version 6.09 or later) datafiles.

If you want to use **savas**

on UNIX/Linux machines, then feel free to download /usr/ it here:

Download savas.csh (c-shell script).

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As...". Otherwise, you will need to save the web page as a plain text file (not as htm/htm). You may want to subsequently rename the file from `savas.csh` to **savas**.

You will also need the SAS macros SAVASTATA and CHAR2FMT.

Download: savastata.sas and char2fmt.sas.

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As...". Otherwise, you will need to save the web pages as plain text files (not as htm/htm). The best way to download SAVASTATA and CHAR2FMT is to use Stata's command

ssc install to get the Stata program usesas which also uses the SAVASTATA and CHAR2FMT SAS macros. The SAS macro file `savastata.sas`

will be in the same directory as other ado-files that start with the letter "s" and the char2fmt.sas file will be in the same directory as ones that start with the letter "c":

```
sac install usesas , replace
```

You will also need the Stata program savasas. Use Stata's command

ssc install to get all the files required for savasas.

```
sac install savasas , replace
```

Here is the **savas**

man page.

Click here to download the savas man page. If you are not prompted to "Save To Disk", then right-click the link and choose "Save Target Link As...". Otherwise, you will need to save the web page as a plain text file (not as htm/htm).

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About savas

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Date: 02Dec2003
Last updated: 25Mar2008

Make a Stata datafile from a SAS datafile or a SAS datafile from a Stata datafile

```
savas [-options] DataFileNam.ext ...
```

Examples

(The dollar sign indicates a UNIX/Linux prompt. Do not type the dollar sign to use **savas**)

```
$ savas mystata.dta $ savas mysas.sas7bdat $ savas -/data mystata.dta $ savas -r mystata.dta $ savas -r ../group/mysa
```

Description

savas uses both SAS and Stata installed on the same Linux/Unix machine to make copies of one or more SAS/Stata datasets as Stata/SAS files. The output dataset will have the same name, but with the appropriate filename extension:

SAS Version 9/8: `.sas7bdat`

SAS Version 6 UNIX: `.sas01`

SAS Version 6 Linux: `.sas02`

SAS 6 Transport/sport: `.sas`, `.sport`, `.sxp`, `.sxport`, `.sxas`, `.stx`, `.vtx`, `.vtx`, `.txsas` Or `.sxpo` file extensions plus whatever file extension the file might have.

SAS Transport files created by PROC CPORT `.cport` and `.xsp` file extensions plus whatever file extension the file might have.

SPSS portable files: `.por` file extension.

Stata: `.dta`

savas can convert SPSS portable files to Stata thanks to SAS's SPSS read-only engine. **savas** recognizes these files by the "`.por`" file extension. **NOTE**: Starting in SPSS 11, SPSS will open and save SAS `sas7bdat` files.

By default the Stata/SAS file is created in the same directory as the SAS/Stata datafile, but with the appropriate filename extension and contains all observations and every variable in the SAS/Stata datafile. **savas** requires the use of both Stata and SAS on the same machine.

savas cannot process files that have filenames or are in directories that contain single or double quotes.

The procedure is as follows:

1. **savas** creates a Stata/SAS program that loads the Stata/SAS dataset into Stata/SAS and calls the **savas** Stata/SAS program.
1. **savas** uses either Stata's command fdasave to save the dataset in memory temporarily as a SAS sport datafile or has SAS write the data to an ASCII text file.
1. **savas** writes a Stata/SAS input program to load the dataset into Stata/SAS and to assign variable names, labels (and formats).
1. **savas** runs the program in Stata/SAS in batch mode to load the data.
1. Stata/SAS saves the data as whatever version Stata/SAS file type specified.

NOTE: If saving to old versions of SAS or Stata that have variable name restrictions less than the version of the dataset being processed, **savas** checks for variable names that are too long for the output dataset; and, if the "**-rename**" option is issued, **savas** renames them to the first 8 characters or up to 7 plus a number. In addition, it will display this list of renamed variables.

If the SAS/Stata dataset is sorted by one or more variables, the Stata/SAS dataset will also be sorted by those same variables. The maximum length for a string variable to be passed on to SAS is 200 characters. In such cases, the first 200 characters will be taken and passed on to SAS (this is a limitation of the SAS sport dataset used to transfer data from Stata to SAS). If saving a SAS dataset as a Stata dataset, long character variables will be truncated to the maximum length that Stata will allow. This maximum may be 80 or 244 depending on what version of Stata is being used. Stata's help page on limits will let you know which applies. **savas** will report which, if any, variables were truncated and to what length they were truncated. Stata variables labels can be up to 80 characters in length.

Options

Option	Explanation
-ci-curdir	**savas** saves the Stata/SAS dataset to the current working directory, even though the Stata/SAS dataset may be located elsewhere:
-rename	specifies that any required renaming of variable names is to be done. The **-rename** option is only necessary when saving to a older version of SAS or Stata or when variable names are not unique in SAS. When saving to an older version **rename** attempts to rename long variable names (more than 8 characters) to be unique by shortening all long variable names to the first 6 characters or up to the 7 plus a number; **savas** lists all variables that were renamed. If more than one dataset is submitted to **savas**, then this option will only work for the first dataset. Check out the **-force** option.
-ri-replace	By default, **savas** warns the user if the output dataset already exists, and asks permission to overwrite it. Option **-replace** suppresses this interactive behavior and replaces any existing output dataset without warning. If more than one dataset is submitted to **savas**, then this option will only work for the first dataset Check out the **-force** option.
-force	is equivalent to using both **-rename** and **-replace** and will maintain these options if more than one dataset is submitted to **savas**.
-check	creates two check files for the user to compare the input dataset with the output dataset to make sure **savas** created the files correctly. This is a comparison that should be done after any datafile is converted to any other type of datafile by any software. The files are created in the same directory as the output datafile and are named starting with the name of the datafile followed by either "`_ckdsmds.lst`" (SAS) or "`_STATAdsmds.log`" (Stata), e.g. "`mydata_ckdsmds.lst`" and "`mydata_STATAdsmds.log`".
-fmts/-formats	specifies to either save value labels that exist in the Stata dataset as SAS formats in a file that will have the same name as the datafile but with the "`.sas7bcat`" file extension or to use such a file if creating a Stata dataset. This formats catalog file will be created or needs to be in the same directory as the SAS datafile. By default value labels are not saved or created. **NOTE**: SAS formats have to be 8 characters or less and cannot end in a number. **savas** makes some attempt to rename invalid SAS formats, but it would be best for you to rename or drop them in Stata before using **savas**. Stata does not allow string variables to have user-defined formats numbers with decimal values.
-sas6	indicates to save the Stata file as a SAS version 6 file: SAS 9 will read/open SAS 6 files but will not save to a version 6 SAS dataset.
-sasx	indicates to save the Stata file as a SAS version 6 transport/sport file using the /sport engine.
-o/-old	indicates to save the Stata file as previous version of Stata to the current version, e.g., version 8.
-i/-intercooled	indicates to save the Stata file as intercooled. This is only necessary if Stata SE or Stata MP is being used.
-char2lab	indicates to use the SAS macro CHAR2FMT to convert long character variables to numeric with Stata value labels. This is like Stata's encode command. This option is only helpful when saving to a Stata 9 or higher dataset since Stata 9 added the feature of allowing value labels to be up to 32,000 characters long.
-q/-quotes	indicates to replace double quotes (") occurring in character variables with single quotes (') and replace compound quotes (`` or '') occurring in variable labels or formats with single quotes ('). **savas** cannot process character variables with double quotes or variable labels or formats with compound quotes when converting a dataset from SAS to Stata.
-s/-xport	**savas** converts SAS transport files into Stata datafiles. **NOTE**: Multiple transport datafiles can be processed at a time but all datafiles need to be SAS transport files. There can be no intermixing of regular SAS/Stata datafiles and transport files when using this option.
-f/-float	prevents the use of Stata's variable type "double": All variables whose SAS precision would require Stata's **double** type are created as **float**. This option may lead to a loss of precision, but saves space. A **float** is stored in 4 bytes, a **double** in 8 bytes.
-rights	sets the file permission of the new SAS file to be whatever default file permissions would be for a new file in that directory. The default permissions are the same as the Stata datafile.
-b/-beep	beeps upon completion.
-s/-silent	be silent; in this case, **savas** does not print any output to the screen, except for error messages. By default, **savas** tells what stage of the conversion process is currently being executed, and it reports number of variables, number of observations, and more.
-ascii/-sascode	specifies that only a datafile and an input program are to be created. By default, **savas** executes all four steps outlined above; The **-ascii/-sascode** option aborts this process after step 3. The user then needs to read in the data manually using Stata/SAS: **savas** writes a SAS program (`mydata_infile.sas`) to read in the SAS datafile (`mydata.sas`) or **savas** writes a Stata do-file (`mydata_infile.do`) to read in the ASCII datafile (`_mydata_.raw`).
-mi/-messy	**savas** specifies that all the intermediary files created by **savas** during its operation are not to be deleted. The **-messy** option prevents **savas** from cleaning up after it has finished. This option is mostly useful for debugging purposes in order to find out where something went wrong. All intermediary files have a name starting with an underscore (_) followed by the process ID and are located in the temp directory.
-obs=n	converts only the first n observations. By default, **savas** converts all observations of the Stata/SAS dataset.
-varfile=filename	may be used to select only a subset of variables to be included in the Stata/SAS dataset. This will speed up the conversion process and is useful in situations where the number of variables is too large for a non-Stata SE (Special Edition) file, more than 2,047 variables. The filename is the name of a file whose contents are variable names only. These variable names are case-insensitive when saving to Stata. If saving to SAS, multiple variables can be listed using any of Stata's specified varlist rules. For example, **var1** is understood as **var1**, **var2** ... or if saving to Stata, multiple variables with the same stem can be specified as ranges according to general SAS rules. For example, **var1-var20** is understood as **var1**, **var2**, ..., **var20**.
-n/-nice=n	runs SAS/Stata nicely. The default is 20. This should be used if you have a very large datafile and there are others using the UNIX/Linux box. For example: `$ savas -n=10 mydata.dta`

Features

savas attempts to transfer Stata value labels to SAS formats and vice versa. **savas** creates only one format per value label and vice versa rather than creating a new format or value label for each variable that was assigned that format or value label. So, if you have a SAS dataset with one **yes_no**, format assigned to twenty variables, the new Stata dataset will have one **yes_no** value label assigned to those twenty variables. Date formats are translated as closely as possible. Fixed SAS formats (**Pw.d**) translate into Stata's **%w.d** format. SAS date formats are translated as closely as possible. Unformatted variables get Stata's default formats for the appropriate data type (**%8.0g** for bytes and ints, **%9.0g** for floats, and **%10.0g** for doubles), except for long variables, which **savas** formats as **%12.0g**. **savas** can process multiple files at a time: Try

```
$ savas *.sas7bdat
```

or:

```
$ savas *.dta
```

savas stamps the SAS creation date and time on the Stata dataset name; so that the Stata user knows not only when the Stata dataset was created, but also the original SAS creation date and time. Not all SAS variable names are acceptable in Stata.

savasattempts to prevent conflicts by using uppercase names for reserved names. These reserved names are:

- _all
- _b
- byte
- _coef
- _cons
- double
- float
- if
- in
- int
- long
- _n
- _pred
- _rc
- _se
- _skip
- _uniform
- using
- with
- names starting with "**str**" and followed by an integer. (For example, name "**street**" does not pose any problems, but a SAS variable named "**str10**" will be translated into a Stata variable named "**STR10**");
- A SAS variable named "_n" translates into "_____N" (and a warning is issued.)

Not all Stata variable names are acceptable in SAS because Stata allows variable names to be different based on upper or lower or mixed case. So the variable gender can be in the same dataset as "**Gender**" or "**GEnder**" etc. **savas** attempts to prevent conflicts by testing for situations like the "gender" issue and when the **-rename** option is issued **savas** attempts to rename the variables to be unique by adding a number to the end of the variable name. If saving to an older version, then **-rename** will shorten all variable names that are longer than 8 characters.

Acknowledgements

This script was inspired by the sas2stata script developed at RAND.

Bugs

None known.

SAS character variables may be up to 32,767 characters in length; Stata 7 and 8 Intercooled limit string variables to 80 characters; Starting with Stata 9 Intercooled and SE limit string variables to 244 characters; **savas** will truncate such variables and write out a warning.

Stata Intercooled datasets are limited to 2,047 variables. Stata 6 datasets have a maximum width (number of bytes) of 6,192; Stata Intercooled datasets have a maximum width (number of bytes) of 24,564. Stata SE datasets can store as many variables as a SAS 8 dataset, 32,767 and have a max width of 12 times the number of variables. SAS 9 datasets may have an unlimited number of variables.

Research
CPC's Primary Research Areas
CPC's Signature Research Approaches
Projects
Data
Publications
Tools & Resources
Agent Based Modeling Tools
Biomedical Specimens and Population Research
Data Analysis
Stata Tutorial
Topics in SAS Programming
ML Sampling Weights
SAS Codebook
A SAS User's Guide to Stata
SAS code matched to Stata code
Basic Conceptual Differences Between SAS and Stata
Stata's Windows Mode
SAS Functions Matched to Stata Functions
How to Submit a Do-file in Batch Mode
Tools to transfer data between SAS and Stata
SAVAGETOS
Why use SAVAGETOS?
SAVSAS9
USESAS9
SAVAS (outwit-wmp)
STATA_WRAPPER
SASONWORK
SAS macros inspired by Stata programs
Stata programs
Tools to transfer data between SAS matrix catalog files and SAS datasets
Data Security Plans
Secure Data Facility
M&E and Health Information System Strengthening Tools

STATA_WRAPPER

SAS macro to run Stata code from within SAS

The **STATA_WRAPPER** SAS macro is installed when you download usesas.

But you can also download it here: stata_wrapper.sas.

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `stata_wrapper.sas` file as a plain text file not an htm/html file.

The best way to download **STATA_WRAPPER** is to use Stata's command ssc install to install usesas which **STATA_WRAPPER** requires to be installed anyway. The `stata_wrapper.sas` file will be saved in whatever directory Stata takes downloaded files that begin with the letter "s". The Stata command findfile will help you find the file `stata_wrapper.sas`.

The **STATA_WRAPPER** macro is a text file that can be edited in any word processing software. After you edit the file, save it as a text file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About STATA_WRAPPER

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Date: 20Mar2008
Last updated: 28Mar2011

Input: Most recently created SAS WORK dataset and some Stata code.

Output: by default it will be included in the SAS output window list file.

Dependency: You need to have Stata installed on your computer and have the Stata command usesas installed for **STATA_WRAPPER** to load your most recently created SAS WORK dataset into Stata. It would also be a good idea to have the Stata command savesas installed as well if you want to pass data back to your SAS session.

Description

STATA_WRAPPER could be thought of as "SAS-Catalate Stata" since it runs the Stata software that is installed on the same computer that you are running SAS on. By default the **STATA_WRAPPER** SAS macro loads the most recently created SAS dataset that is located in the WORK library/directory into Stata and runs the Stata code written after the line:

```
statata_wrapper(code) datalines;
```

The log of the Stata code will be included in the SAS output window list file unless an output directory name is specified. **STATA_WRAPPER** can also submit Stata code stored in any Stata do-file that is anywhere on your computer as well. The most recently created SAS dataset will be available to use in Stata as well unless the **'-nodata'** option is specified. Non-Windows users need to think of their Stata code being run in batch mode. So consider that windows that display graphs will not be saved unless you use the graph save command. When saving files, you need to specify the directory you want to save to as the current directory is the temporary SAS WORK library/directory which gets deleted when the SAS session is closed.

NOTE: If you are using SAS on Windows, an interactive Stata session will be started by **STATA_WRAPPER**. Do not close the Stata window until the do-file submitted to Stata by **STATA_WRAPPER** has finished running. **STATA_WRAPPER** when run in SAS on Linux/UNIX will run Stata in the background.

Required Input to STATA_WRAPPER:

Unless the option **-nodata** is specified, **STATA_WRAPPER** needs to have the most recently created dataset by SAS to be in the WORK library/directory and some Stata code to process. **STATA_WRAPPER** expects Stata code to be submitted to **STATA_WRAPPER** using the call to **STATA_WRAPPER**:

```
statata_wrapper(code) datalines;
```

See the examples below on how to submit Stata code to **STATA_WRAPPER**.

Options

NOTE: Options can be used in any order. You can specify as many as you want or none at all.

-code

Used when only writing Stata code to be submitted by **STATA_WRAPPER**. No other options can be used when using the **code** option. You have to have **datalines;** after the call to **STATA_WRAPPER**

```
** This invocation of stata_wrapper does not submit the code to Stata.  **;
 * That happens later. **;

statata_wrapper(code) datalines;
 /** This is code that will be run by Stata **/
 mysset myfiltvar [pweight= myhsweight(hv)], strata(dhjfstratumhv)
 svy:mean income weight height, over(agegroup race)
 ////  ** You should put at least one semi-colon after all your Stata code **;

** Your Stata code cannot contain any semi-colons.  Do not use &delimit; **;


** If you really want to use semi-colons in your Stata code, use "&delimit;" **;
 * and end your Stata code with a semi-colon starting at the very  **
 * beginning of the line (column 1 through column 4) **;

statata_wrapper(code) datalines;
 /** This is code that will be run by Stata **/
 &delimit;
 mysset myfiltvar [pweight= myhsweight(hv)], strata(dhjfstratumhv);
 svy:mean income weight height, over(agegroup race);
 ////  ** You need 4 semi-colons starting at the beginning of the line. **;
```

-nodata

Indicates not to load the most recently created SAS dataset in the WORK library/directory into Stata. This should be used when your Stata code does not require that dataset.

-noformats

Specifies NOT to create value labels (which is the default) from SAS user-defined formats that are stored in a SAS formats catalog file that has the same name as the dataset and is in the WORK library/directory. For example: `mydsetdata.sas7bcat`. If this file doesn't exist, usesas will look for the file `formats.sas7bcat` in the WORK library/directory.

-int

(Windows only option, not for Linux/UNIX) Keep interactive Stata running. You have to exit/close Stata before you can return to SAS or exit/close SAS since SAS is waiting for Stata to stop running.

-log

Includes the Stata log in your SAS log rather than your SAS output window list file.

-replace

If an output directory is specified and the output files you want to output already exist, **-replace** will then overwrite them with the files generated by **STATA_WRAPPER**.

Since **STATA_WRAPPER** uses the Stata command usesas to load the most recently created SAS dataset in the WORK library/directory the following are options to be used by usesas when called by **STATA_WRAPPER**:

-charZlab

Specifies to encode long SAS character variables like the Stata command encode. Character variables that are too long for a Stata string variable are maintained in value labels.

-float

Specifies that numeric variables that would otherwise be stored as numeric type double be stored with numeric type float. This option should only be used if you are certain you have no integer variables that have more than 7 digits (like an ID variable).

-quotes

Specifies that double quotes that exist in string variables are to be replaced with single quotes.

Debugging options

-messy

Puts the files generated by **STATA_WRAPPER** used to create the Stata log file in the current directory or the directory named in the pathname provided in the call to the **STATA_WRAPPER** macro if one was provided.

How to use the STATA_WRAPPER macro:

Using the **STATA_WRAPPER** macro requires that you understand how to use the **%include** SAS statement and that you know how to call a SAS macro:

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\ASUMacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\ASUMacros\statata_wrapper.sas";
```

But **STATA_WRAPPER** is likely to be installed in the directory where Stata installs files that begin with the letter "s":

```
%include "C:\sabigdata\statata_wrapper.sas";
```

The **%include** statement makes SAS aware of the **STATA_WRAPPER** macro which is in the file `statata_wrapper.sas`. To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program:

```
statata_wrapper([options], "[folder to save Stata output]" );
```

But you don't really have to specify anything if you want to use the defaults:

```
statata_wrapper;


 ** create a dataset that contains only the observations and
 *  variable(s) needed for the Stata command to be run:  **;
data test;
 set my_data (keep= age income
                  where= (gender = 1));
 run;


** Write some Stata code that will be submitted to
 * Stata by STATA_WRAPPER **;
statata_wrapper(code) datalines;
  [stata code]
  [more stata code]
  ////


** run the STATA_WRAPPER macro: **;
statata_wrapper;
```

Or you can submit a Stata do-file saved in some other directory:

```
statata_wrapper(code) datalines;
 do "C:\my_projects\my_stata_code.do"
 /// more stata code if you want
 ////


** run the STATA_WRAPPER macro: **;
statata_wrapper;
```

The information inside the parentheses is passed on to the **STATA_WRAPPER** macro. The first string of information is any/all **STATA_WRAPPER** options. The second string of information is the location you want to save your Stata results/log file if you do not want it printed in the SAS output window list file. This is somewhat like a **libname** statement.

Examples

```
 ** Include macro once in a SAS session and call it **;
 *  as many times as you like in that session.     **;
%include "C:\sabigdata\statata_wrapper.sas";

** This makes a copy of the SAS dataset in the in the work library. **;
data work.TeDataSta;
 set in.mydSASfile (keep= math cses one school gender
                    where= (gender = 1));
run;


** Write some Stata code to be submitted by STATA_WRAPPER: **;
statata_wrapper(code) datalines;
 mq cses: cses
 mq one: one
 qilsmm math cses. i(school) eqrone cses;  diff adapt srf(2)
 ////  ** end Stata code with at least an semi-colon **;


** Save STATA_WRAPPER submit the above Stata code to Stata  **;
statata_wrapper;


** Just have STATA_WRAPPER submit some Stata code you have already written **;
 *  that will use the most recently created SAS dataset **;
data work.TeDataSta;
 set in.mySASfile (keep= state gender race region
                   where= (region= "NE"));
 run;

statata_wrapper(code) datalines;
 do "C:\MyStata\my_stata_code.do"
 ////


** Save STATA_WRAPPER submit the above Stata code to Stata  **;
statata_wrapper;


** Save the Stata log file "_[some_big_number]_stata_code.log"  **;
 ** in "C:\MyData\", overwriting it if it already exists.    **;
statata_wrapper;replace,"C:\MyData\";


** Run the Stata do-file "my_stata_code.do" in "C:\MyProjects" and **;
 * save the Stata log file "_[some_big_number]_stata_code.log"  **;
 * in "C:\MyData\", overwriting the log file if it already exists.    **;

statata_wrapper(code) datalines;
 do "C:\MyProjects\my_stata_code.do"
 ////

statata_wrapper;replace,"C:\MyData\";
```

This example is like a USESTATA SAS macro

```
statata_wrapper(code) datalines;
 use "C:\My Project\Data\some_data.dta"
 savesas using "some_data.sas7bdat"
 ////


** specify not to use the most recently created SAS dataset in WORK **;
 * since your Stata code does not need it. **;
statata_wrapper(-nodata);


** now you can access this dataset from your WORK library/directory **;
 * because no directory name was specified in the call to savesas above. **;
proc contents data= work.some_data;
 run;
```

The above example assumes that Stata is not set up to **cd** to some other directory when Stata is started up. If you are not in the SAS WORK library/directory at the moment when savesas is invoked you can use the Stata command **sas_work** to **cd** you to that directory. This command is only available in Stata while running **STATA_WRAPPER**. Here is an example using **sas_work**:

```
statata_wrapper(code) datalines;
 use "C:\My Project\Data\some_data.dta"
 sas_work
 return list  // this is optional but shows that
              // the local macro r(sas_work) exists now
 savesas using "some_data.sas7bdat"
 ////


statata_wrapper(-nodata);


proc contents data= work.some_data;
 run;
```

usesas usually sets your memory large enough, but this is an example of what to do if you need more because you are using a version of Stata previous to Stata 12:

```
statata_wrapper(code) datalines;
 preserve   /** temporarily store data **/
 clear      /** clear data so that you can set memory **/
 set mem 1g /** set memory to 1 gigabyte **/
 restore    /** load previously stored data **/
 qilsmm y dn wset other_self blame express, i(persno) ///
                 link(logit) family(binomial adapt sig(2)
 ////


statata_wrapper;
```

Setting up STATA_WRAPPER

These are instructions to edit the `stata_wrapper.sas file`.

NOTE: If you are setting up this macro on your computer for the first time, you can choose which version of Stata you are going to have **STATA_WRAPPER** use. If you do not choose to set one of the following switches **STATA_WRAPPER** will figure out what version of Stata you are running for you. This may add a noticeable amount of time to processing so you may want to set these switches to the correct version of Stata. You can easily figure out what version of Stata you are using by looking at the top of your results window in Stata or by typing in the Stata command about at the Stata command line. One advantage of leaving **STATA_WRAPPER** to figure out what version of Stata is being used is that when you upgrade your version of Stata you will not have to update **STATA_WRAPPER**.

NOTE: If you are running **STATA_WRAPPER** on the UNIX/Linux platforms then you need to be able to start a Stata batch job by:

```
stata -b do mydofile.do
```

If not, then change the setting of the **ustata** macro variable:

```
%let ustata=stata;
 /* This may work:
 %let ustata=/usr/local/stata/stata-mp;  */
```

If you are running **STATA_WRAPPER** on the Windows platform, you need to tell **STATA_WRAPPER** where the `stata.exe` file is located. If you do not know where your `stata.exe` is located, find your Stata short-cut icon, right click on it, choose "properties", and look in the "target" field. This will show you where the `stata.exe` is located on your hard drive.

Change what is inside the parentheses to the location of your `stata.exe` file:

```
%let wstata=quote(C:\Stata\stata.exe);  ** This may work. **;
```

In earlier versions of Stata, the Windows Stata executable file is named `wstata.exe`.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
        - SAVASTATA
        - Why use SAVASTATA?
        - SAVASAS
        - USESAS
        - SAVAS (oshell) script
        - STATA_WRAPPER
        - SASWRAPPER
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

SASWRAPPER

Stata command to run a SAS program from within Stata

If you want to use **saswrapper**, then feel free to use Stata's command

ssc install to download and install **saswrapper**:

```
ssc install saswrapper, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About saswrapper

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Date: 20Feb2009
Last updated: 04Mar2010

Run a SAS program from within Stata

```
saswrapper [varlist] [using filename ]  [if exp ] [in range ] [, pre_sas_prog(`" sss code "') post_sas_prog(`" sss
code "') check messy usesas nodata savasas(sss dataset name) noformats rename clear char2lab float quotes ]
```

Description

saswrapper runs a SAS program in batch and prints the output in the Stata results window/log file. This usually occurs by specifying a SAS program file after **using**, but **saswrapper** can also run SAS code specified by the **pre_sas_prog()** and/or **post_sas_prog()** options. By default, **saswrapper** will save the current data in memory using savasas and make it available in SAS's WORK library. If that is not desired, use the **nodata** option. The **usesas** option tells **saswrapper** to load the last SAS dataset created in the WORK library by the submitted SAS program into Stata using the **SAVASTATA** SAS macro.

NOTE: saswrapper calls SAS to run the SAS program. This requires the ability to run SAS on your computer. If **saswrapper** does not run SAS for you, your `saswx.ado` file may need to be edited to set the location of your SAS executable file (`sas.exe`) and your **SAVASTATA** SAS macro file (`savastata.sas`). See the setup instructions below. It may be that **saswrapper** will be able to run with the default settings in `saswx.ado`.

saswrapper uses the **SAVASTATA** SAS macro to create the Stata dataset from the SAS dataset when the **usesas** option is specified. **saswrapper** downloads the **SAVASTATA** SAS macro and stores it where user-written Stata ado-files are stored that begin with the letter "s". This macro can be used in SAS. Learn about SAVASTATA here: SAVASTATA

Options

using *sas program filename* specifies **saswrapper** to run this SAS program. **saswrapper** assumes the SAS program file extension "`.sas`" if no file extension/suffix is specified.

pre_sas_prog() and/or **post_sas_prog()** contain SAS code to be run by **saswrapper**. If SAS programming code is supplied by **using**, **pre_sas_prog()**, and **post_sas_prog()** then the order the code will be run is:

pre_sas_prog()

using

post_sas_prog()

It is best to enclose the SAS code within compound double quotes:

```
. saswrapper, pre_sas_prog(`"proc means;"')
```

in case the SAS code contains any double quotes. An interesting way to submit SAS code is to first put it in a local macro and use three forward slashes to continue the line:

```
        . local pre_sas_prog  data new; ///
          set work.stata_data; ///
           gender = "female"; ///
          run;

        .  saswrapper , pre_sas_prog(`" `pre_sas_prog' "')
```

You can put a lot of SAS code in a local macro if you don't use an equal sign (=) after the local macro name. For the above example to work, Stata has to be using the carriage return as an end-of-line delimiter and not semi-colons which is changed by the #delimit command. Stata will see Stata-style comments in the SAS code as Stata comments so avoid starting a line with a star/asterisk "*":

```
          * some comment   ; ///
```

because Stata will comment out the three forward slashes at the end and thus end the input to the local macro. Since the contents of the local macro will be one long line, do not attempt to do anything in your SAS code that expects carriage returns like the SAS **datalines** statement that reads in raw data.

check specifies to generate basic stats for both the SAS and Stata datasets for the user to make sure **saswrapper** created the files correctly. This is a comparison that should be done after any datafile is converted to any other type of datafile by any software. The SAS file is created in the same directory as the SAS program specified in **using** or if no program was specified in **using** then it will be created in the current directory. The file is named starting with the name of the datafile followed by "`_SASCheck.lst`" (SAS). e.g. "`mySASdata_SASCheck.lst`"

messy specifies that all the intermediary files created by **saswrapper** during its operation are not to be deleted. The **messy** option prevents **saswrapper** from cleaning up after it has finished. This option is mostly useful for debugging purposes in order to find out where something went wrong. All intermediary files have a name starting with an underscore "_" followed by the process ID and are located in Stata's temp directory.

usesas specifies to load the SAS dataset into memory that was most recently created in the SAS WORK library in the SAS program submitted to **saswrapper**. **saswrapper** figures out how much memory the SAS dataset will require to be loaded into Stata and sets Stata's memory for you if your memory setting is less than is required.

nodata specifies to override the default behavior of **saswrapper** which is to save the current dataset in memory to the SAS WORK library. Use this option when your SAS program is not going to use the dataset in memory.

savasas specifies a different dataset name than the name of the dataset in memory. If the dataset in memory does not have a name and the option **savasas** is not used, then the dataset in memory will be available in SAS's WORK library as "**stata_data**" (a.k.a. "**WORK.STATA_DATA**").

Options when saving data to SAS

varlist specifies what selection of variables in the dataset in memory are to be saved to the SAS dataset in the WORK library in the SAS program submitted to **saswrapper**. If no variables are specified then all variables will be saved the SAS dataset.

if *exp* subsets the dataset in memory before saving the dataset to the SAS dataset in the WORK library in the SAS program submitted to **saswrapper**.

in *range* subsets the dataset in memory before saving the dataset to the SAS dataset in the WORK library in the SAS program submitted to **saswrapper**.

noformats specifies that no value labels be saved as SAS formats.

rename specifies that any required renaming of file name and/or variable names is to be done when saving the dataset in memory to SAS using savasas. The **rename** option is only necessary when variable names are not unique in SAS. **savasas** displays the list of renamed variables. **rename** also renames the SAS file name when the name provided is not a valid SAS file name.

Options when using the usesas option

clear specifies to clear the data currently in memory before running **saswrapper**.

char2lab specifies to encode long SAS character variables like the Stata command encode. Character variables that are too long for a Stata string variable are maintained in value labels.

float specifies that numeric variables that would otherwise be stored as numeric type double be stored with numeric type float. This option should only be used if you are certain you have no integer variables that have more than 7 digits (like an ID variable).

quotes specifies that double quotes that exist in string variables are to be replaced with single quotes. Since the data are written out to an ASCII file and then read into Stata, there are rare instances when double quotes are not allowed inside string variables.

Examples

```
. saswrapper using "mySASprog.sas"

. saswrapper using "mySASprog.sas", nodata usesas clear

. saswrapper using "c:\data\mySASprog.sas", savasas(tester) check

. saswrapper , pre_sas_prog(`" proc print; "')

. saswrapper , pre(`" data new; set work.stata_data; if income > 0 then do; "' ///
        `" employed = 1; output; end; run; "') usesas clear

. saswrapper , post(`" proc univariate; "')
```

Setting up saswrapper

If you are setting up this program on your computer for the first time, you may need to edit the `saswx.ado` file to set the location of your SAS executable file (`sas.exe`). If you do not, **saswrapper** will look in the "usual" locations for it. **saswrapper** also may need to have the location of the SAS macro savastata.sas set. The `saswx.ado` file is an ASCII text file and should be saved as such after editing. Stata's do-file editor will do the job.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

SAS macros inspired by Stata programs

CHAR2FMT
Encode character variables

RUNQUIT
Run an error check to stop SAS

ISID
Test if variable(s) uniquely identify each observation in a dataset

MINIME
Reset variable lengths to minimize the size of a SAS dataset

WHICH
Find out what SAS macro SAS will invoke

REN_STEM
Rename variables

TYPE
Print a text file to SAS output window/.lst file or log

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

CHAR2FMT

Encode character variables

If you want to use **CHAR2FMT**, then feel free to download it here:

Download char2fmt.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `char2fmt.sas` file as a plain text file not an htm/html file.

The `char2fmt.sas` is a plain text file that can be edited in any word processing software. If you edit the file, save it as a text file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About CHAR2FMT

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 20Aug2005
Last updated: 09Mar2009

Convert character variables to numeric

```
%char2fmt(dset=dataset in WORK , varlist= variable list , maxlen= max length of variables to convert );
```

Description

The **CHAR2FMT** SAS macro converts character variables to numeric and creates SAS formats to store the character information. This is like the Stata command encode.

If the character variable has a character format/value label then **CHAR2FMT** uses the actual/non-formatted variable values for the new numeric variable and ignores the character format/value label.

You could write the following SAS code:

```
data work.new (drop= relation);
 set work.old;
  if relation = "Mother" then relation_n = 1;
  else if relation = "Father" then relation_n = 2;
  else if relation = "Sister" then relation_n = 3;
  else if relation = "Brother" then relation_n = 4;

  ** make the label of the new numeric variable the same as the character one **;
  relation_n = "Relationship to respondent";
run;

proc format library=library.new;
 value relation
  1 = "Mother"
  2 = "Father"
  3 = "Sister"
  4 = "Brother"
 ;;;
run;

data new;
 set new;
  rename relation_n = relation;
  ** assign the format relation. to the variable relation **;
  format relation relation.;
run;
```

Or you could use **CHAR2FMT** to do it for you:

```
%char2fmt(dset= myData, varlist= relation);
```

Options

Option	Explanation
dset= *dataset_name*	Name of dataset (no libref) in the WORK library in which variables are to be encoded. The dataset has to be in the WORK library.
varlist= *variables to encode*	List of variables to encode to numeric variables no matter what length they are. If a varlist is specified and **maxlen=** is also specified, then only variables in the list that have lengths longer than what is specified in **maxlen=** will be encoded to numeric variables.
maxlen= *some number*	The maximum length a character variable can be and _not_ be converted to numeric. Basically, **CHAR2FMT** was written with the intention to help maintain long character variables (longer than 244 characters) when converting your data to Stata.

How to use the CHAR2FMT macro:

Using the **CHAR2FMT** SAS macro requires that you understand how to use the **%include** statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\SASmacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\char2fmt.sas";
```

The **include** statement makes SAS aware of the **CHAR2FMT** SAS macro which is in the file `char2fmt.sas`. You call the macro by adding this to your code:

```
%char2fmt(dset= myData, varlist= companyname location alliancetype );
```

Examples

```
%include "C:\SASmacros\char2fmt.sas"; ** Make SAS aware of the CHAR2FMT macro.**;

data work.new;
 set work.old;
run;

** now call the CHAR2FMT SAS macro by feeding it your dataset name and the
 * maximum length a character variable can be without getting selected
 * for conversion.  A comma separates  what you feed the CHAR2FMT SAS macro.  **;

 %char2fmt(dset= new, maxlen= 120 );

 ** or all character variables (by not specifying maxlen= ): **;
data work.new;
 set work.old;
run;

%char2fmt(dset= new);

proc contents data= work.new;
run;

 ** or only the specified character variables: **;
data work.new;
 set work.old;
run;

%char2fmt(dset= new, varlist= location companyname );
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

RUNQUIT

Run an error check to stop SAS

If you want to use **RUNQUIT**, then feel free to download load it here:

Download runquit.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `runquit.sas` file as a plain text file not an htm/html file.

If you are running SAS 9.2 or higher and want to copy and paste **RUNQUIT** into the top of all your SAS programs, **RUNQUIT** can be as simple as:

```
%macro runquit;
  ; run; quit;
  %if &syserr. ne 0 %then %do;
     %abort cancel;
  %end;
%mend runquit;
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About RUNQUIT

Programmer: Dan Blanchette
Carolina Population Center
University of North Carolina at Chapel Hill
Chapel Hill, NC USA

Date: 14May2010
Last updated: 08Jun2010

Stop SAS in your interactive Windows SAS session or batch submission when an error occurs without ending your SAS session.
`%runquit;`

Description

RUNQUIT is a SAS macro that inserts "`run;`" and "`quit;`" into your code and does an error check and if an error has occurred it stops SAS from continuing to process the rest of the submitted statements. You can just type "`%runquit;`" instead of "`run;`" or "`quit;`" in your code so that SAS will stop whenever an error occurs.

For batch SAS users, **RUNQUIT** inserts "**endsas;**" into your code when an error occurs and thus stops the program from continuing.

For interactive SAS users running SAS 9 (or higher) **RUNQUIT** does not end your interactive SAS session when an error occurs. You will not loose any of your code or data sets. **RUNQUIT** invokes the SAS macro **%ABORT** with the **CANCEL** option which only cancels submitted statements and writes this in your log for SAS 9.2 or higher:

```
ERROR: Execution canceled by an %ABORT CANCEL statement.
NOTE: The SAS System stopped processing due to receiving a CANCEL request.
```

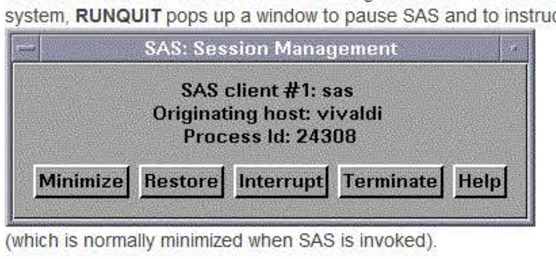
SAS 9 introduced the **%ABORT** macro but it did not have the **CANCEL** option, but "`%ABORT cancel;`" still stops SAS from processing the rest of your code in an interactive SAS session without terminating your session. It writes this error message in your log:

```
ERROR: Unrecognized option on %ABORT statement: cancel
ERROR: Execution terminated by an %ABORT statement
```

For interactive SAS Windows users running an earlier version of SAS than version 9 on the Windows operating system, **RUNQUIT** basically presses the Interrupt button (the icon with the exclamation point "!" in a circle in the SAS toolbar): for you when an error occurs. When the pop-up window comes up all you have to do is click "**OK**" two times as long as your default selected answers are:

1. Cancel Submitted Statements
and in the 2nd pop-up:
Y to cancel submitted statements

For interactive SAS Windows users running an earlier version of SAS than version 9 but not running SAS on the Windows operating system, **RUNQUIT** pops up a window to pause SAS and to instruct you to click the Interrupt button in your SAS Session Manager:



(which is normally minimized when SAS is invoked).

So, just type: type "`%runquit;`" instead of "`run;`" or "`quit;`" in your code so that SAS will stop whenever an error occurs.

How to use the RUNQUIT macro:

Using the **RUNQUIT** macro requires that you understand how to use the "`%include`" SAS statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied the `runquit.sas` file to "`C:\SASmacros\`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\runquit.sas";
```

The `include` statement makes SAS aware of the **RUNQUIT** macro which is in the file `runquit.sas`. To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program:

```
%runquit;
```

Examples

```
%include "C:\SASmacros\runquit.sas"; ** Include macro once in a SAS session and call it **;
                                      *  as many times as you like in that session.     **;

 ** This will generate an error due to an invalid libref: **;
 proc contents data=sas_please_help.shoes;
 %runquit;

 ** This proc will not be run: **;
 proc contents data=sashelp.shoes;
 %runquit;
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

ISID

Test if variable(s) uniquely identify each observation in a dataset

If you want to use **ISID**, feel free to download it here:

Download isid.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `isid.sas` file as a plain text file not an htm/html file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About ISID

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 13Feb2009
Last updated: 06Mar2009

Check a dataset to see if variable(s) uniquely identify each observation

```
%isid(dset= libref.data_set_name, varlist= variable name(s) );
```

Description

The **ISID** SAS macro is just like the Stata command isid. It tests if the variable(s) uniquely identifies each observation in the dataset. It's a good idea to run **ISID** before doing a data merge so that you know your merge variables are appropriate to use for the merge. It is most often the case that you do not want to do a merge that generates the SAS note:

```
NOTE:  MERGE Statement has more than one dataset with repeats of BY values
```

ISID will check the most recently created SAS dataset if no SAS dataset name (including the libref if not in the WORK library) is provided. **ISID** will use the variable(s) the dataset is sorted by if no variable list to test the dataset with is provided.

Options

Option	Explanation
dset= *name of dataset*	The string following "**dset=**" is the SAS dataset name (and libref if not in the WORK library). If **dset=** is not used, then **ISID** will use the most recently created SAS dataset.
varlist= *list of variable(s) to check*	The string following "**varlist=**" is a list of variables in the dataset to check to see if they uniquely identify each observation in the dataset. If **varlist=** is not used, then **ISID** will use the variables the dataset is sorted by.
missok=missok	**missok** specifies that it is okay for the variables being checked by **ISID** to have missing values. Missing values in variables that are supposed to uniquely identify an observation is generally not a good idea. When using the **missok** option **ISID** will return a note in the SAS log saying what variables that are being checked have missing values if any of the variables have at least one missing value. **ISID** will return an error message.

How to use the ISID macro:

Using the **ISID** SAS macro requires that you understand how to use the **%include** SAS statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`c:\SASmacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "c:\SASmacros\isid.sas";
```

The **%include** statement makes SAS aware of the **ISID** macro which is in the file "`isid.sas`". To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program:

```
%isid(dset= sashelp.shoes, varlist= stores);
```

The information inside the parentheses is passed on to the **ISID** macro. The string following "**dset=**" is the SAS dataset name (and libref if not in the WORK library). The string "**varlist=**" is one or more variables to test if they uniquely identify each observation in the dataset. An error message will be returned in the SAS log if the variables do not uniquely identify each observation:

```
ERROR: variables  Product Stores do not uniquely identify each observation
```

and a note will be returned if they do:

```
NOTE: variables Product Subsidiary Stores uniquely identify each observation
```

Examples

```
%include "C:\SASmacros\isid.sas"; ** Include macro once in a SAS session and call it **;
                                  *  as many times as you like in that session.     **;

%isid(dset= sashelp.shoes, varlist= region product stores sales);

%isid(dset= sashelp.shoes, varlist= region stores);

** if the most recently created dataset is sorted then no dataset name
 *  nor variable list is required **;

proc sort data= sashelp.shoes
          out= work.new;
  by region product;
run;

%isid;

%isid(dset= new);

%isid(varlist= region product stores sales);

%isid(varlist= region product stores sales, missok=missok);
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

MINIME

Reset variable lengths to minimize the size of a SAS dataset

If you want to use **MINIME**, feel free to download it here:

Download minime.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `minime.sas`

file as a plain text file not an htm/html file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About MINIME

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 22May2003
Last updated: 09Mar2009

Reset the variable lengths to minimize the size of a SAS dataset

```
%minime( variables not to minimize , variables to minimize );
```

Description

The **MINIME** SAS macro resets variable lengths of the dataset most recently created in the WORK library to the safest minimum for all operating systems. This should reduce the size of the SAS datafile, but it could increase it, perhaps appropriately so. This is like the Stata command compress.

Options

NOTE: The parameters have to be specified in the order of this list.

Parameters	Explanation
(1st) ***variables not to minimize***	The first string inside the parentheses can be a list of one or more variables not to minimize. In general, it is not good to minimize long integer variables like ID variables in case in a future merge the dataset being merged in has longer ID variables than the ones in the current dataset being minimized. If no variables are listed not to minimize, then the default is to minimize all variables.
(2nd) ***variables to minimize***	The second string inside the parentheses after the first comma can be a list of one or more variables to minimize. If no variables are listed to minimize, then the default is to minimize all variables.

How to use the MINIME macro:

Using the **MINIME** SAS macro requires that you understand how to use the **%include** statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\SASmacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\minime.sas";
```

The **include** statement makes SAS aware of the **MINIME** SAS macro which is in the file `minime.sas`. You call the macro by adding this to your code:

```
%minime;
```

Examples

```
%include "C:\SASmacros\minime.sas"; ** Make SAS aware of the MINIME macro.**;

data work.new;
 set work.old;
run;

** now call the MINIME SAS macro to minimize all the variables in the dataset **;

%minime;

*******************************************************************
 -- To see the attrib statement MINIME used to reset the variable
     lengths, use the PRINTME SAS macro after MINIME has run:
*******************************************************************;

%printme;

proc contents data= work.new;
run;


** or all character variables **;
data work.new;
 set work.old;
run;

%minime( , _character_ );  ** notice that the first field is empty and
                            *  that a comma separates the two fields **;

proc contents data= work.new;
run;


** or only the numeric variables except for you ID variable: **;
data work.new;
 set work.old;
run;

%minime( companyid , _numeric_ );

proc contents data= work.new;
run;

** or all variables except for you ID variable: **;
data work.new;
 set work.old;
run;

%minime( companyid );

proc contents data= work.new;
run;
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

WHICH

Find out what SAS macro SAS will invoke

If you want to use **WHICH**, feel free to download it here:

Download which.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `which.sas` file as a plain text file not an htm/html file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About WHICH

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 03Nov2009
Last updated: 09Dec2010

Find a SAS macro file in the directory it is stored on your computer:

`%which(name of macro without the percent sign , options );`

Description

The **WHICH** SAS macro is like the Stata command (and UNIX/Linux command) which. It looks in your SASAUTOS path for the location of a SAS macro that SAS will automatically find when: `%macroname` is used in SAS code and then it will print the first 10 lines of the macro in your log window/file.

In UNIX/Linux you can add directories of other locations of SAS macros that SAS will automatically find when `%macroname` is used in SAS code by putting a file named: `sasv9.cfg` in your home directory and putting this line in it:

```
-sasautos ( '!SASROOT/sasautos' '$HOME/SASmacros/' )
```

`NOTE`: It is best to add your personal directories after: `'!SASROOT/sasautos'` so that SAS will work normally for you in case you put a SAS program file that's named the same as a SAS macro that is installed with SAS like: `lowcase.sas`.

`options mautolocdisplay;` will display in the log the path to the SAS program and the file name that contains the autocall macro whenever that macro is invoked.

`%which(macro_name);` will show all macros named `macro_name` that are compiled and in any of the directories specified `SASMSTORE` option libref in the order they will be invoked by SAS. SAS will only use the first macro named `macro_name` that SAS finds compiled in a concatenated library specified in the `SASMSTORE` option but **WHICH** will find and show all of them. **WHICH** will also find and show all macros files named `macro_name.sas` in the directories specifed in the `SASAUTOS` system option but SAS will only use the first one that SAS finds.

The **WHICH** SAS macro can not be run in any version of SAS prior to version 9.

Options

Option	Explanation
name of macro	Specifies the name of the SAS macro to look for. Do not type the percent sign.
full	Print all lines of the macro (not just the first 10 lines which is the default).

How to use the WHICH macro

Using the **WHICH** SAS macro requires that you understand how to use the **%include** SAS statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\SASmacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\which.sas";
```

The **%include** statement makes SAS aware of the **WHICH** macro which is in the file "`which.sas`". To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program:

```
%which(lowcase, full);
```

The information inside the parentheses is passed on to the **WHICH** macro.

Examples

```
%include"C:\SASmacros\which.sas"; ** Include macro once in a SAS session and call it **;
                                  *  as many times as you like in that session.     **;

%which(lowcase);

** print all lines of lowcase.sas: **;
%which(lowcase , full);
```

Research
- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

REN_STEM

Rename variables

If you want to use **REN_STEM**, feel free to download it here:

Download ren_stem.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `ren_stem.sas` file as a plain text file not an htm/html file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About REN_STEM

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 27Apr2004
Last updated: 25Feb2009

Rename a list of variables by adding a specified prefix or suffix to the variable names

```
%ren_stem( dataset_name , variable list , constant , options );
```

Description

The **REN_STEM** SAS macro renames the selected variables by adding the specified prefix or suffix to the variable name. For example, say you want to have all of your "wave 1" variables start with "**w1_**". You could manually type:

```
rename height = w1_height;
rename weight = w1_weight;
rename income = w1_income;
 . . .
```

Or you could use **REN_STEM** to do it for you:

```
ren_stem(myData , _all_ not(id) , w1_ , before);
```

Options

NOTE: The parameters have to be specified in the order of this list.

Parameters	Explanation
(1st) ***dataset_name***	Full name of dataset (include libref if necessary) in which variables are to be renamed. If no dataset name is provided then the last dataset used is processed; but the call to **REN_STEM** must start with a comma since the first parameter is the dataset name: `%ren_stem( , height weight income , w1_ , before);`
(2nd) ***variable_list***	Can be of the following different valid SAS variable list styles: **gender race** - the two variables **gender** and **race** **var1-var5** - **var1 var2 var3 var4 var5** **age--sampwt** - could be: **age gender weight psu sampwt** **age-numeric-sampwt** - only the numeric variables between **age** and **sampwt** **age-character-sampwt** - only the character variables between **age** and **sampwt** **_all_** - all variables in dataset **_numeric_** - all numeric variables in dataset **_character_** - all character variables in dataset **va:** - all variables in dataset that begin with the string "**va**" You can exclude certain variables from your varlist (like your ID vars) by: `_all_ not(hhid personid)` **NOTE**: There can be no space between the word "not" and the left parenthesis! YES: `not( varlist )` NO: `not ( varlist )` ALSO: The order must be `varlist` first, then `not(varlist)`.
(3rd) ***constant***	String you want to append to the variable name as a prefix or suffix to make it a new name.
(4th) **Options**	**Explanation**
before / after	Position of constant is to be before or after variable name. [If no placement is specified then the constant will go before the existing variable name.]
N / C	If you have a variable list of both numeric and character type variables but only want to rename the character variables, use **C**. Similarly if you only want to rename the numeric variables in the varlist, use **N**. [If no vartype is specified then all variable types are renamed]. It can be upper or lower case **N/n** (numeric) or **C/c** (character).
SAScode	Prints the SAS code used by the macro to create the dataset with the renamed variables. This could be useful if you want to modify the code in some way or if you want to simply see what was done.

How to use the REN_STEM macro:

Using the **REN_STEM** macro requires that you understand how to use the **%include** statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\SASmacros\`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\ren_stem.sas";
```

The include statement makes SAS aware of the **REN_STEM** macro which is in the file `ren_stem.sas`. You call the macro by adding this to your code:

```
%ren_stem(myData, myvar1 myvar2, pre_ , before);
```

Examples

```
** Make SAS aware of the REN_STEM macro.**;
%include "C:\SASmacros\ren_stem.sas";

data new;
 set old;
run;


** now call your macro by feeding whatever variables and whatever you want to tack
 * on the beginning of the new variable names.  A comma separates what you feed
 *  the ren_stem macro.  **;
*%ren_stem(dataset , list of variables , constant , options) **;
%ren_stem(  new   , x1 x2 x3 x4 x5 y1 , round4_  , before  N);  ** eg. round4_x1 **;


** or : **;
data out.new;
 set old;
run;

%ren_stem(out.new , x1 x2 x3 x4 x5 , _max , after sascode n);  ** eg. x1_max **;


** or all character variables : **;
data new;
 set old;
run;

%ren_stem(myData , _numeric_ , _constant , after ); ** all numeric type variables eg. x1_constant **;


** or : **
data new;
 set old;
run;

%ren_stem(testdata , x1--gender not(pid) , _min , after);   ** eg. x1_min gender_min **;

proc contents data= new;
run;
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
        - CHAR2FMT
        - RUNQUIT
        - ISID
        - MINIME
        - WHICH
        - REN_STEM
        - TYPE
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

TYPE

Print a text file to SAS output window/.lst file or log

If you want to use **TYPE**, feel free to download it here:

Download type.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `type.sas` file as a plain text file not an htm/html file.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About TYPE

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Developed at Research Computing at The University of North Carolina at Chapel Hill

Date: 26February2008
Last updated: 09December2010

Print an ASCII text file SAS output window or "`.lst`" file or log

```
%type("file name and location", options );
```

Description

The **TYPE** SAS macro is just like the Stata command (and DOS command) type. It just prints your text file to the SAS output window or "`.lst`" file or SAS log. It even has the **starbang** option that Stata's type command has that prints only lines that start with "`*!`". The SAS command line command **fslist** and the SAS procedure **PROC FSLIST** are also very similar to the **TYPE** SAS macro.

Options

NOTE: The parameters have to be specified in the order of this list.

Parameters	Explanation
(1st) ***file name***	Needs to be the text file name and location to print to SAS output or log.
(2nd) **Options**	**Explanation**
-starbang	It specifies to only print lines that start with "*!"
-log	It specifies to print to the SAS log. The default is to print to the SAS output window/.lst file.

How to use the TYPE macro:

Using the TYPE macro requires that you understand how to use the **%include** SAS statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`c:\SASmacros`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "c:\SASmacros\type.sas";
```

The include statement makes SAS aware of the **TYPE** SAS macro which is in the file type.sas. To use the macro you have to make a call to it. To do that you add a line like the following to your SAS program:

```
%type("folder & name of an ASCII text file", [options]);
```

The information inside the parentheses is passed on to the **TYPE** macro. The first string of information is the ASCII text file you want to submit to Stata. The second string of information after the comma is the options you wish to pass on to the **TYPE** macro.

Examples

```
%include "C:\SASmacros\type.sas"; ** Include macro once in a SAS session and call it **;
                                  *  as many times as you like in that session.     **;

** print the text file my_file.txt to SAS output: **;
*%type( file name and location, options ) **;
 %type("C:\MyData\my_file.txt",         );


** print the text file my_file.txt to SAS log: **;
*%type( file name and location, options ) **;
 %type("C:\MyData\my_file.txt", -log    );


** only print lines that start with a star and a bang "*!" **;
** in the file C:\MyData\my_file.txt  **;
%type("C:\MyData\my_file.txt", -starbang);
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Stata programs

ADOEDIT
Edit ado-file in Stata's do-file editor

ETIME
Keep track of elapsed time

IFWINS
Subset dataset by if exp before subsetting by in range

LOOKFORIT
Search variable names and labels for some string or strings (an enhancement of lookfor)

LOOKFOR_ALL
Searches for variables/patterns in Stata data files (.dta)

LSTRFUN
Modify long local macros that contain strings

MKPROFILE
Create or edit your profile.do file or just see where Stata searches for a profile.do file

NSPLIT
Split numeric variables with integer values into new smaller numeric variables

RCD
Recursively traverse directories and execute any command in or for each one of them

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
        - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

ADOEDIT

Edit ado-file in Stata's do-file editor

If you want to use **adoedit**, then feel free to use Stata's command: ssc install to download and install **adoedit**:

```
ssc install adoedit, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About adoedit

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 06Nov2003
Last updated: 08Feb2005

Edit the ado-file that runs the specified command

`adoedit` *`cmdname`*

Description

adoedit attempts to edit whichever ado-file named `cmdname.ado` is first on the current adopath. One cannot edit files larger than 32,000 bytes in the do-file editor, nor can one edit built-in commands. This routine is designed for use by those who are writing their own ado-files; users are exhorted to make a copy of any official ado-files, and to save modified official ado-files in the ado directory, rather than the Stata directory.

```
adoedit cmdname.ado
```

Specifying the ado file extension is tolerated.

After a successful edit, the *discard* command is issued so that Stata will load the revised ado-file. This will also clear any estimates from the last estimation.

If you wish to edit ado-files with a different text editor, please see fedit and the texteditors modules on the SSC archive.

Examples

```
. adoedit tsspell
```

Saved Results

The **adoedit** command saves in **r()**:

r(adofile)	the ado-file name and full path

Acknowledgements

This program grew out of previous work by:

Nicholas J. Cox, University of Durham, U.K.
and
Christopher F Baum, Boston College, USA

Also see

On-line: doedit, findfile

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

ETIME

Keep track of elapsed time

If you want to use **etime**, then feel free to use Stata's command ssc install to download and install **etime**:

```
ssc install etime , replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About etime

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 30Sep2003
Last updated: 10Nov2010

Time how long a program runs; or, simply, keep track of the elapsed time from point A to point B in a program

```
etime [ , start datestart(start date) timestart(start time)]
```

Description

etime is designed for programmers who want to keep track of how long a program takes to run. Using Stata's creturn macro variables:
c(current_date) and c(current_time), **etime** clocks the difference in time between calls. Like a stopwatch, **etime** needs to be started and stopped. The second and following times **etime** is submitted it displays a message that says how long it has been since the first time **etime** was submitted (in the current Stata session).

Options

start specifies to set or reset **etime**. Not specifying start is equivalent to ending or simply checking the elapsed time.

Options datestart() and timestart() need to be used when timing a command that runs etime or Stata's -macro drop _all- and -sreturn clear-.

datestart specifies to **etime** to use this start date. The date must be in the **%dD_m_cy** format like: **10 Jul 2008** which is the format of c(current_date). If you specify a start date you also have to specify time.

timestart specifies to **etime** to use this start time. The time must be in **hr:min:sec** format like: **14:20:58** which is the format of c(current_time). If you specify a start time but not the start date, the current date will be used.

Remarks

etime creates global macros **$etime_stardate_** and **$etime_startime_** .

Examples

```
. etime, start

. etime
```

Displays:

```
Elapsed time is 2 hours 27 minutes 3 seconds
```

```
. local date `c(current_date)'

. local time `c(current_time)'

. etime, d(`date') t(`time')

. etime, d(10 Jul 2008) t(03:48:14)
```

Saved Results

The **etime** command saves in **s()**:

Macros:	
s(etime)	days:hours:mins:secs elapsed since first call to **etime**
s(esecs)	total number of seconds elapsed since first call to **etime**
s(stardate)	days since Jan. 1, 1960 that **etime** started
s(startime)	number of seconds into the day that **etime** started
s(endate)	days since Jan. 1, 1960 that **etime** ended
s(endtime)	number of seconds into the day that **etime** ended plus 86,400 seconds for each day since **etime** started.
Global Macros:	
$etime_stardate_	days since Jan. 1, 1960 that **etime** started
$etime_startime_	number of seconds into the day that **etime** started

Also see

On-line: log, datefun, creturn, profiler, rmsg, timer

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

IFWINS

Subset dataset by if exp before subsetting by in range

If you want to use **ifwins**, then feel free to use Stata's command ssc install to download and install **ifwins**:

```
ssc install ifwins, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About ifwins

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 10Feb2009
Last updated: 25Feb2009

Subset dataset by if exp before subsetting by in range

```
ifwins [if exp] [in range] : stata_cmd
```

Description

ifwins is a prefix command that runs most any Stata command that does not modify the dataset in memory (e.g. generate, replace, etc.). **ifwins** will have if subset the dataset before in subsets the dataset. This is the opposite of what happens when both if and in are used in the same Stata command. For example, the following code will first subset the dataset to the first 10 observations and then subset the dataset to the specified condition:

```
. sysuse auto
. list if foreign == 1 in 1/10
```

Since the auto.dta dataset is sorted by the variable **foreign**, the above code will not list any observations because in the first 10 observations **foreign == 0** . So, if looses and in wins when "battling" over which one subsets the dataset before the other one does.

If you want to run a Stata command on a certain number of observations when a certain condition exists, you would have to:

```
. preserve
. keep if foreign == 1
. list in 1/10 make turn weight
. restore
```

or use **ifwins** as a prefix to the desired Stata command:

```
. ifwins if foreign == 1 in 1/10 :  list make turn weight
```

The above will list the first 10 observations of when the variable **foreign** is equal to 1 (one). So now if wins!...but in is still helpful.

Remarks

Since **ifwins** (when not running the list, browse, edit commands) temporarily subsets the dataset before running the submitted Stata command the observations numbers will not be the same as in the original dataset. So, the system variables **_n** and **_N** as well as the c(N) macro variable are reset to the subsetted dataset before the specified Stata command is run by **ifwins**. Also, **ifwins** will never return the error message "Obs. nos. out of range r(198)" if the range specified is at least within the number of observations in the dataset. **ifwins** corrects ranges specified by in to at least 1 observation of the specified condition since it is hard to know how many observations will be available after the if condition subsets the dataset.

The list, browse, edit commands are the only exceptions to the note above.

if and in are not allowed in the Stata command **ifwins** is specified to run.

Examples

```
. ifwins if mpg < 25 in -10/L :  list make turn weight

. ifwins if income > 1000000 in 1/200 :  summarize x1 x2 x3 x4 year89 year90
```

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
        - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

LOOKFORIT

Search variable names and labels for some string or strings (an enhancement of lookfor)

If you want to use **lookforit**, then feel free to use Stata's command ssc install to download and install **lookforit**:

```
ssc install lookforit, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About lookforit

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 19Nov2003
Last updated: 04Mar2009

Search variable names and variable labels for some string or strings

```
lookforit string [string [...]]
```

Description

lookforit is an enhancement of the base Stata command lookfor. It searches all variable names and variable labels just as lookfor does. Like lookfor, **lookforit** can search for multiple strings but can only do a simple search, i.e. no &'s, |'s, or enclosing strings in quotes. The ways that **lookforit** differs are as follows:

1. **lookforit** does a case sensitive search if the user types any upper case characters, otherwise, it performs a case insensitive search.

   ```
   . lookforit female
   ```

   finds "Female", "FEMALE", "feMale", or even "FemalePatient"

   ```
   . lookforit Female
   ```

   only finds "Female"

2. **lookforit** lists variables alphabetically

3. The variable names are clickable links to the data editor that opens Stata's data editor to that variable column.

Examples

```
. lookforit job

. lookforit employment Salary

. lookforit inc
```

Also see

Online: help for lookfor, ds

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

LOOKFOR_ALL

Searches for variables/patterns in Stata data files (.dta)

If you want to use **lookfor_all**, then feel free to use Stata's command

ssc install to download and install **lookfor_all**:

```
ssc install lookfor_all, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About lookfor_all

Programmers:
Zurab Sajaia (zsajaia@hotmail.com)
The World Bank

Michael Lokshin (mlokshin@worldbank.org)
The World Bank

Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 05Feb2007
Last updated: 06Oct2010

Searching within current directory

`lookfor_all pattern[pattern [...]] [, options]`
where *pattern* is a word that could be a name (or part of the name) of the variable and/or a word (or part of a word) in the variable label or value label name or value label if the **vlabs** option is specified or a word (or part of a word) in a note attached to the dataset or a word (or part of a word) in a note attached to a variable if the **notes** option is specified. **lookfor_all** only does case insensitive searches so searching for "test" may find "Test", "TEST", or "tESt" as well as "test".

Searching within current directory and subdirectories
`lookfor_all pattern[pattern [...]] [, subdir options]`

Main

directory(*path*) an option to specify the root directory where to search

subdir an option for searching subdirectories

describe an option to format the output

codebook another option to format the output

vlabs also search value label names and value labels

notes also search patterns in dataset notes and variable notes

dnotes also search patterns in dataset notes

dnotes also search patterns in variable notes

Advanced

maxdepth(*integer > 1*) an option to specify how far down a directory tree to search

maxvvar an option for Stata SE or Stata MP users to allow **lookfor_all** to open a dataset with up to 32,767 variables

dirfilter(*regular expression*) an option to specify a regular expression pattern that the subdirectory name has to satisfy in order to search files in that directory.

filter(*regular expression*) an option to specify a regular expression pattern that the file name has to satisfy in order to search the file.

Description

lookfor_all is an extension of the Stata program lookfor. **lookfor_all** searches through all Stata data files in a current directory and subdirectories for the variable(s) with at least one of the specified patterns. Each *pattern* could be a part/whole variable name and/or a word (part of the word) in the variable label. **lookfor_all** outputs the list of data files where the *pattern* was found and the list of variables within every data file that match the *pattern*. The files listed are clickable. By clicking on the file, Stata loads that data file into memory. **lookfor_all** can also search a directory other than the current directory by specifying the **directory(*path*)** option for a search.

Options

Main

directory(*path*) specifies the *path* where **lookfor_all** should start searching. By default **lookfor_all** searches files in the current directory.
subdir if specified **lookfor_all** searches through all Stata data files (`.dta`) in the current directory and subdirectories or the specified directory and its subdirectories. If this option is not specified **lookfor_all** searches only within the files in the current directory or the specified directory.
describe controls the way the variables that match the *pattern* are outputted. By default **lookfor_all** outputs the variables following the name of the data file they are in. Specifying **describe** makes Stata run the describe command on the variables that were found. You can control the output using the options of the Stata command describe. These options are **simple short details fullname numbers**.
codebook controls the way the variables that match the *pattern* are outputted. Specifying **codebook** makes Stata run the codebook command on the variables that were found. The output will be the number of unique observations, mean, minimum, maximum and the label of the matching variables. You can control the output using the options of the Stata command codebook.
vlabs if specified **lookfor_all** will also search through the value label names and value labels.

Advanced

maxdepth(*integer > 1*) if specified **lookfor_all** will only search through all Stata data files (`.dta`) as far down as the positive integer value that is specified. If **subdir** is not specified and **maxdepth** is, then it is assumed subdirectories should be searched.
maxvar if specified and you are using Stata SE or MP, **lookfor_all** will temporarily set maxvar to the maximum which is 32767 so that **lookfor_all** can search a dataset that contains more variables than maxvar is currently set to be able to open. Changing the setting of maxvar will stop the **page up** key from being able to access command history. So if you set maxvar to 32767 yourself before running **lookfor_all**, **lookfor_all** will not need to set it and you will be able to access your command history with the **page up** key after running **lookfor_all**.
dirfilter(*regular expression*) if specified **lookfor_all** will only search through all Stata data files (`.dta`) in the current directory or specified directory and only subdirectories that satisfy the regular expression pattern.
rfilter(*regular expression*) if specified **lookfor_all** will only search through Stata data files (`.dta`) that satisfy the regular expression pattern.

Examples of searching within the current directory

```
. lookfor_all edu . lookfor_all edu, describe simple . lookfor_all edu, de short
```

Examples of searching the current directory and sub-directories

```
. lookfor_all edu, sub . lookfor_all edu, maxdepth(3) de fullnames . lookfor_all edu, sub codebook vlabs . lookfor_a
```

Examples of searching the specified directory

```
. lookfor_all edu, sub de dir("c:\data\") . lookfor_all edu, sub de dir("c:\data\") filter(^wave1)
```

Useful suggestions and feedback by Phil Bardsley and Michael Norman Mitchell are gratefully acknowledged.

This Michael Norman Mitchell's Stata Tidbit of the week on lookfor_all.

Also see

Online: lookfor, describe, and codebook

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

LSTRFUN

Modify long local macros that contain strings

If you want to use **lstrfun**, then feel free to use Stata's command ssc install to download and install **lstrfun**:

```
ssc install lstrfun, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About lstrfun

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 09Aug2010
Last updated: 09Aug2010

Modify a long local macro that contains a string

```
lstrfun local_macro_name [, options]
```

where *local_macro_name* is a valid name of either an existing local macro that will be modified or a valid name of a local macro that will be generated by **lstrfun**. The strings being modified can be as long as the maximum number of characters in a local macro minus about 50 (which is about how many characters are used to submit **lstrfun**).

Only one local macro name can be submitted and only one option can be specified.

Options

In these descriptions *string* is referring to the contents of the local macro.

lower(*macname***)** make string all lowercase.

proper(*macname***)** make string proper case.

upper(*macname***)** make string all uppercase.

ltrim(*macname***)** remove leading blanks from string.

itrim(*macname***)** replace multiple internal blanks with a single blank in string.

rtrim(*macname***)** remove trailing blanks from string.

substr(*macname*, *pos*, *length* **)** cut string from starting point for the specified length.

subinstr(*macname*, *"from"* , *"to"*, *n***)** replace "from string" with "to string" in the string for the specified number of occurrences.

subinword(*macname*, *"from"* , *"to"*, *n***)** replace "from word" with "to word" in the string for the specified number of occurrences.

strdup(*macname*, *n***)** duplicate the string the specified number of times.

reverse(*macname***)** reverse the string.

_substr(*macname*, *"tosub"*, *pos***)** substitutes "tosub" into the string at the specified position.

Submit a new local macro name to be generated when using these options since these options return numbers/codes.

strlen(*macname***)** returns the length of the string.

strpos(*macname*, *"needle"***)** returns the position of the specified string.

strmatch(*macname*, *pattern***)** returns 1 if the string matches the specified pattern.

soundex(*macname***)** returns the soundex code for the string.

soundex_nara(*macname***)** returns the U.S. Census soundex code for the string.

indexnot(*"characters"*, *macname***)** returns the position of the first character in the set of characters specified that is not found in the string, or it returns 0 (zero) if all characters in the set of characters specified are found in the string.

Description

lstrfun allows you to modify local macros that contain strings using Mata's string functions which are not restricted to the maximum length of strings in Stata like the normal string functions in Stata are limited. If you are using Stata SE or Stata MP, then your setting of maxvar affects the maximum number of characters in a macro. The maximum number of characters in a macro is only 1,081,511 when maxvar is set to 32767. When maxvar is set to 5000 (the default value) then the maximum number of characters in a macro is 165,200. The local macro submitted to **lstrfun** will be modified if it already exists or it will be created if it does not exist. **lstrfun** can be very helpful when trying to modify value labels and notes in Stata since value labels and notes are allowed to have more characters than normal string functions are allowed to handle. The option strlen() has an equivalent function in Stata's extended macro functions, but the options subinstr() and subinword() have similar functions in Stata's extended macro functions which do not allow you to specify a specific number of replacements to make.

Local macro strings that are submitted in any of the options have to be enclosed in at least double quotes. It is a good idea to enclose local macro strings in compound double quotes in case the macro contains quotes.

The maximum number of characters submitted to an option of **lstrfun** in total cannot exceed the maximum length of strings in Stata minus about 50 (which is about how many characters are used to submit **lstrfun**). The command file read can create local macros with more characters than Stata allows. If you get the following error messages when submitting **lstrfun**, then the submitted macro has too many characters:

```
unmatched quote
r(198);
too many macros
r(920);
command too long
r(1004);
```

If you are using Stata SE or Stata MP, then increase your setting of maxvar since it affects the maximum number of characters in a macro.

Examples of modifying local macros

It is a good idea to use **macval()** when submitting a macro so that if the submitted macro contains a macro: *`something'* then it is left as is.

```
. lstrfun mvar, lower(`"`mvar'"')

. lstrfun mvar, itrim(`"`macval(mvar)'"')

. lstrfun mvar, substr(`"`mvar'"', 332, 3235)

. lstrfun mvar, _substr(`"`macval(mvar)'"', "XX", 20345)
```

Examples of creating new local macros

```
. lstrfun new_mvar, strpos(`"`mvar'"', "needle in the haystack")

. lstrfun new_mvar, strmatch(`"`mvar'"', "mat*me")

. lstrfun new_mvar, strmatch(`"`macval(mvar)'"', `"`macval(mvar2)'"')

. lstrfun new_mvar, indexnot("abcdef", `"`mvar'"')
```

Useful suggestions and feedback by Nick Cox are gratefully acknowledged.

Also see

Online: string functions, Mata string manipulation functions, and extended macro functions

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

MKPROFILE

Create or edit your profile.do file or just see where Stata searches for a profile.do file

If you want to use **mkprofile**, then feel free to use Stata's command ssc install to download and install **mkprofile**:

```
ssc install mkprofile , replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About mkprofile

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 06Feb2005
Last updated: 22Apr2010

Create or edit your `profile.do` file or just see where Stata searches for `profile.do`file

```
mkprofile [, memory( integer value for number of megabytes of memory ) vlabel( integer between 8 and 32 ) list all
nthdir edit cwd ]
```

mkprofile with no options specified is equivalent to:

```
. mkprofile , memory(40) vlabel(12)
```

Description

mkprofile creates or edits your `profile.do` file. Your `profile.do` file contains the settings you would like Stata to have every time you start a Stata session. By default, **mkprofile** creates your `profile.do` file to set your variable label position to 12 and set your memory to 40m.

Options

memory(*integer* **)** sets Stata's memory to that number of megabytes. Be careful not to set it too low or too high as you will get an error message when your `profile.do` file is run.

vlabel(*integer* **)** sets the variable label position in the variable label window. If you tend not to have variable names longer than 8 or so characters, set it to 10 to allow you to easily see the variable label associated to the variable.

list specifies **mkprofile** only to list the directories that your `profile.do` file will be attempted to be created in rather than attempt to create or edit your `profile.do` file. The order of the directories listed will be the reverse order of directories that Stata will search for your `profile.do` file so that the top listed directories are more likely the directories you have permission to create your `profile.do` file. Also specifying the **all** option will make the order be the order of all the directories that Stata searches for your `profile.do` file.

all specifies **mkprofile** to list all the directories that Stata will look in for your `profile.do` file and attempt to create your `profile.do` file in the first of these directories in which it can create it. **NOTE**: These directories may require administrator directory permissions. By default **mkprofile** only tries to attempt to create your `profile.do` file in a directory you most likely have permission to create the file.

nthdir(*integer* **)** species in which of the directories listed by **mkprofile** you prefer **mkprofile** to create your `profile.do` file. If not specified, **mkprofile** will create your `profile.do` file in the first directory it can create it in. **NOTE**: The **all** option affects the order of the directories so the nth directory will be different if **all** is specified or not specified when using the **nthdir()** option.

edit will allow you to open your current `profile.do` file in the do-file editor.

cwd tells Stata to change to the directory you are currently in when you invoke **mkprofile** at the beginning of each Stata session. Note: "cwd" means "current working directory." To use this option, first cd to the directory you normally use, then use the **cwd** option in **mkprofile**. The next time you start Stata, you will be in this directory. To start Stata in a different directory in the future, edit your `profile.do` file.

NOTE: An example of how to set-up **F** keys (eg. **F4**, **F5**) on your keyboard to be used to change to set directories is implemented in the `profile.do` file using the current directory in the example. Use macro list to see what **F** keys are already being used.

Remarks

Since Stata only runs the first `profile.do` file that it finds in its search path, there is no guarantee that the `profile.do` file created by **mkprofile** is always going to be found and run by Stata. Read more about profile. **mkprofile** can be set to have a default location in which to create the `profile.do` file. For Windows "`c:\ado\`" is suggested. For UNIX/Linux and Macintosh "`~/ado/`" is suggested. A preferred directory can be set to a specific directory along Stata's search path for the `profile.do` file by editing the `mkprofile.ado` file. If a preferred directory is set, then this directory will be the only directory **mkprofile** will see as an option to try to create the `profile.do` file even if the **all** option is specified.

Examples

```
. mkprofile

. mkprofile, vlabel(10)

. mkprofile, list

. mkprofile, mem(100) vl(10) cwd nthdir(3)

. mkprofile, edit memory(100) vlab(10) cwd
```

Also see

profile, memory, varlabelpos

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

NSPLIT

Split numeric variables with integer values into new smaller numeric variables

If you want to use **nsplit**, then feel free to use Stata's command ssc install:

```
ssc install nsplit, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About nsplit

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 30Nov2004
Last updated: 02Dec2004

Split numeric variables with integer values into new smaller numeric variables.

```
nsplit [varname] [if exp] [in range] , digits(digit pattern in existing variable) [generate(newvarlist or stub)]
```

Description

nsplit creates however many new numeric variables it takes to split a numeric variable into digit pattern. If your existing numeric variable contains decimal values, then **nsplit** will only split the integer portion of the number.

Options

digits(*digit pattern in existing variable***)** describes the pattern in the existing numeric variable. If it is a repeated pattern then you only need to supply the number of digits the new variables should contain. For example:

```
. nsplit id, digits(2)
```

for an ID variable with 6 digits is equivalent to:

```
. nsplit id, digits(2 2 2)
```

generate(*newvarlist or stub***)** states what you want the new variable names to be. If only one name is listed than that will be the first characters of the new variable names: **stub1 stub2** etc. If no new variable name is en the original variable name will be used as the stub.

Examples

```
. nsplit id, digits(2 3)

. list id*

. nsplit date, digits(2) generate(month year)

. list date month year
 date month year
 1294 12 94
 588 5 88
```

Acknowledgements

Many suggestions were implemented that were made by:
Nicholas J. Cox, University of Durham, U.K.

Also see

On-line: split, mod(), int(), dexfcns

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
        - ADOEDIT
        - ETIME
        - IFWINS
        - LOOKFORIT
        - LOOKFOR_ALL
        - LSTRFUN
        - MKPROFILE
        - NSPLIT
        - RCD
    - Tools to transfer data between SAS matrix catalog files and SAS datasets
  - Data Security Plans
  - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

RCD

Recursively traverse directories and execute any command in or for each one of them

If you want to use **rcd**, then feel free to use Stata's command ssc install to download and install **rcd**:

```
ssc install rcd, replace
```

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About rcd

Programmers:
Nikos Askitas (nikos@iza.org)
IDSC of IZA
IZA, Bonn Germany

Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA

Date: 03Apr2009
Last updated: 03Apr2009

Recursively traverse directories and execute any command in or for each one of them

```
rcd [directory_name], [verbose] [depth(integer >= 1)] [dirfilter(regular expression)] [: command]
```

Description

rcd expects a directory name otherwise the current working directory is used. **rcd** proceeds to recursively visit all subdirectories starting there. **rcd** will display what directories it goes to in verbose mode, obey a maximum depth if one is specified and it will optionally run a command in each directory if desired. Which directories the submitted command are run in can be selected by using the **dirfilter()** option.

rcd may be thought of as a recursor. A typical example of what you can do using it is:

```
. rcd . : dir "data"
```

which would result in visiting every subdirectory starting where you are wanting to see the Stata datafiles in each one of them.

Another way to achieve the same thing would be to use the returned macros that **rcd** generates:

```
. rcd .
. forvalues i = 1(1)`r(tdirs)'{
.   dir "`r(ndir`i')'`c(dirsep)'*dta"
. }
```

Options

verbose specifies to print the directories that **rcd** visits. By default, if **rcd** is invoked with no directory specified or options specified, then **rcd** runs in verbose mode.

depth specifies an integer value greater than or equal to 1 (one) to limit how far down the directory tree **rcd** will go. **depth(1)** specifies not to look in subdirectories.

dirfilter specifies regular expression to limit what directories the specified command will be run in. If you want to see what directories your command will be run in, use **rcd** and specify to run pwd like so:

```
. rcd , dirfilter(data) : pwd
```

Remarks

Since **rcd** runs whatever Stata command and changes directories in the process, if there is a lot printed to the results window and you get tired of clicking more to continue and quit, then your current directory may not be where you started. To quickly get back to the directory you were in when you invoked **rcd** just use the returned local macro **r(sdir)** like so:

```
. cd "`r(sdir)'"
```

Examples

Only look in directories whose names contain the word "data":

```
. rcd . , dirfilter(data)
```

An example from the realm of managing your files and your data can be constructed by using the grep command:

```
. rcd .
. return list
. forvalues i = 1(1)`r(tdirs)'{
.   grep Family Children, path("`r(ndir`i')'`c(dirsep)'data")
. }
```

The code above locates and lists Stata datafiles to either use or describe all or part of each dataset found in the current directory or any of it's subdirectories whose variable names or labels contain either of the words "Family" or "Children".

Another way to do the same thing by listing as you go would be:

```
. rcd . : grep Family Children
```

Saved results

Macros:	
r(sdir)	directory **rcd** was invoked in
rcd	saves the following in **r()**:
r(tdirs)	total number of directories searched
r(ndir1)	name of the first directory searched
r(ndir2)	name of the second directory searched

Also see

Online: help for cd, dir

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
        - MAT2DAT
        - DAT2MAT
    - Data Security Plans
    - Secure Data Facility
  - M&E and Health Information System Strengthening Tools

Tools to transfer data between SAS matrix catalog files and SAS datasets

MAT2DAT
Create SAS datasets from a SAS matrix catalog file

DAT2MAT
Create a SAS matrix catalog file from SAS datasets

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
        - MAT2DAT
        - DAT2MAT
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

MAT2DAT

Create SAS datasets from a SAS matrix catalog file

If you want to use MAT2DAT, then feel free to download load it here:

Download mat2dat.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `mat2dat.sas` file as a plain text file not an htm/html file.

The DAT2MAT SAS macro is a complementary SAS macro to **MAT2DAT**.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About MAT2DAT

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Developed at Research Computing at The University of North Carolina at Chapel Hill
Date: 10Jul2008
Last updated: 12Mar2009

Create a SAS dataset(s) from a SAS matrix catalog file

```
%mat2dat( input libref , matrix catalog name , output libref );
```

Description

MAT2DAT creates a SAS dataset out of each matrix in a SAS matrix catalog file. This is especially useful if you have to transfer the matrix data from one operating system to another since SAS catalog files can only be read into SAS on the operating system the file was created on where SAS datasets can be read on any operating system. To convert the SAS datasets back to a SAS catalog file again, use the DAT2MAT SAS macro.

Options

NOTE: The parameters have to be specified in the order of this list.

Parameters	Explanation
(1st) ***input_libref***	The libref where the input SAS matrix catalog file exists.
(2nd) ***catalog_name***	The name of the input SAS matrix catalog file (without the file extension: ".sas7bcat").
(3rd) ***output_libref***	The libref where the output SAS dataset will be saved to.

How to invoke MAT2DAT:

Using the **MAT2DAT** macro requires that you understand how to use the **%include** statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`C:\SASmacros\`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\mat2dat.sas";
```

The include statement makes SAS aware of the **MAT2DAT** macro which is in the file `mat2dat.sas`. You call the macro by adding this to your code:

```
%mat2dat( LIB_IN , CATALOG_FILE , LIB_OUT );
```

Examples

```
libname LIB_IN "C:\wherever your matrix_data.sas7bcat file is\";

libname LIB_OUT "C:\wherever you want your datasets stored\";

%include "C:\wherever you put mat2dat.sas\mat2dat.sas";

%mat2dat(LIB_IN , matrix_catalog , LIB_OUT);
```

The above invocation of **MAT2DAT** expects to find the SAS matrix catalog file:

```
matrix_catalog.sas7bcat
```

in the **LIB_IN** library. If the matrix catalog file contains the matrices:

```
mat1 mat2 mat3
```

then it will create the SAS datasets:

```
matrix_catalog_mat1.sas7bdat
matrix_catalog_mat2.sas7bdat
matrix_catalog_mat3.sas7bdat
```

in the **LIB_OUT** library.

Research

- CPC's Primary Research Areas
- CPC's Signature Research Approaches
- Projects
- Data
- Publications
- Tools & Resources
  - Agent Based Modeling Tools
  - Biomedical Specimens and Population Research
  - Data Analysis
    - Stata Tutorial
    - Topics in SAS Programming
    - ML Sampling Weights
    - SAS Codebook
    - A SAS User's Guide to Stata
      - SAS code matched to Stata code
      - Basic Conceptual Differences Between SAS and Stata
      - Stata's Windows Mode
      - SAS Functions Matched to Stata Functions
      - How to Submit a Do-file in Batch Mode
      - Tools to transfer data between SAS and Stata
      - SAS macros inspired by Stata programs
      - Stata programs
      - Tools to transfer data between SAS matrix catalog files and SAS datasets
        - MAT2DAT
        - DAT2MAT
- Data Security Plans
- Secure Data Facility
- M&E and Health Information System Strengthening Tools

DAT2MAT

Create a SAS matrix catalog file from SAS datasets

If you want to use DAT2MAT then download it here:

Download dat2mat.sas

If you are not prompted to "Save To Disk", then right-click the link and choose "Save Link Target As..." Otherwise, you will need to save the web page to your computer. Make sure you save the `dat2mat.sas` file as a plain text file not an htm/html file.

The MAT2DAT SAS macro is a complementary SAS macro to **DAT2MAT**.

Disclaimer: There is no warranty on this software either expressed or implied. This program is released under the terms and conditions of GNU General Public License.

About DAT2MAT

Programmer: Dan Blanchette
The Carolina Population Center
The University of North Carolina
Chapel Hill, NC USA
Developed at Research Computing at The University of North Carolina at Chapel Hill

Date: 10Jul2008
Last updated: 12Mar2009

Create a SAS matrix catalog from SAS dataset(s)

```
%dat2mat( input libref , matrix catalog name , output libref , names of matrices to create );
```

Description

DAT2MAT creates one SAS matrix catalog file from the specified datasets and names the matrices with the specified matrix names. This is especially useful if you have to transfer the matrix data from one operating system to another since SAS catalog files can only be read into SAS on the operating system the file was created on where SAS datasets can be read on any operating system. To convert the SAS catalog file back to SAS datasets again, use the MAT2DAT SAS macro. **DAT2MAT** is designed to process SAS datasets that were created by the MAT2DAT SAS macro or datasets that have a naming convention of:
"`matrix_catalog_name_matrix_name.sas7bdat`"

Like for example the SAS dataset named:
"`w4n808_adj_id.sas7bdat`"

would create the matrix catalog file:
"`w4n808.sas7bcat`"

with a matrix named:
"**adj_id**".

Options

NOTE: The parameters have to be specified in the order of this list.

Parameters	Explanation
(1st) ***input_libref***	The libref where the input SAS dataset(s) exists.
(2nd) ***catalog_name***	The name of the SAS matrix catalog file (without the file extension: "`.sas7bcat`") to create. This string has to be the first string of the dataset names followed by an underscore (_).
(3rd) ***output_libref***	The libref where the output SAS matrix catalog file will be saved to.
(4th) ***matrix_names***	Names of the SAS matrices to create. These names have to be the end part of the SAS dataset names in the input library that start with the specified matrix catalog filename followed by and underscore (_). If no matrix names are specified then all SAS datasets in the input library that have names that start with the specified catalog name followed by an underscore (_) will be processed. So, only specify matrix names if only a subset of datasets are desired.

How to invoke DAT2MAT:

Using the **DAT2MAT** macro requires that you understand how to use the **%include** statement and that you know how to call a SAS macro.

```
%include "LOCATION AND NAME OF A FILE THAT CONTAINS SAS CODE";
```

For example, if you have copied this file to "`c:\SASmacros\`", then you tell SAS about this macro by adding the following line to your SAS program:

```
%include "C:\SASmacros\dat2mat.sas";
```

The include statement makes SAS aware of the **DAT2MAT** macro which is in the file `dat2mat.sas`. You call the macro by adding this to your code:

```
%dat2mat( LIB_IN , CATALOG_FILE , LIB_OUT , matrix names );
```

Examples

```
libname LIB_IN "C:\wherever your matrix_data.sas7bdat file is\";

libname LIB_OUT "C:\wherever you want your matrix catalog file to be saved\";

%include"c:\wherever you put dat2mat.sas\dat2mat.sas";

%dat2mat(LIB_IN , matrix_catalog , LIB_OUT , mat1 mat2 mat3);
```

The above invocation of **DAT2MAT** expects to find the SAS datasets:

```
matrix_catalog_mat1.sas7bdat
matrix_catalog_mat2.sas7bdat
matrix_catalog_mat3.sas7bdat
```

in the **LIB_IN** library and will create the matrix catalog file:

```
matrix_catalog.sas7bcat
```

in the **LIB_OUT** library and it will contain the matrices:

```
mat1 mat2 mat3
```

If the SAS datasets:

```
matrix_catalog_mat4.sas7bdat
matrix_catalogmat2.sas7bdat
```

also exist in the input library, these datasets will not be processed by **DAT2MAT** because:

```
mat4
```

was not spcecified in the matrix name list and:

```
matrix_catalogmat2.sas7bdat
```

does not have an underscore (_) following the specified matrix catalog file name.