Using Non Stata Programs within Stata

Karl Keesman

SDAS

17 September 2011

Introduction

Sometimes you may wish to do something within Stata that Stata currently does not do. One solution is to run another program from within Stata.

Examples of the user written commands which use external programs:

1. **stcmd** - Stat/Transfer

Introduction

Sometimes you may wish to do something within Stata that Stata currently does not do. One solution is to run another program from within Stata.

Examples of the user written commands which use external programs:

1. **stcmd** - Stat/Transfer
2. commands that interface with WinBugs

Introduction

Sometimes you may wish to do something within Stata that Stata currently does not do. One solution is to run another program from within Stata.

Examples of the user written commands which use external programs:

1. **stcmd** - Stat/Transfer
2. commands that interface with WinBugs
3. etc.

Today we will look at interfacing 2 other programs with Stata:

1. Sending emails from Stata

Introduction

Sometimes you may wish to do something within Stata that Stata currently does not do. One solution is to run another program from within Stata.

Examples of the user written commands which use external programs:

1. **stcmd** - Stat/Transfer
2. commands that interface with WinBugs
3. etc.

Today we will look at interfacing 2 other programs with Stata:

1. Sending emails from Stata
2. Producing a PDF log output that includes the log file and graphs

Introduction

Sometimes you may wish to do something within Stata that Stata currently does not do. One solution is to run another program from within Stata.

Examples of the user written commands which use external programs:

1. **stcmd** - Stat/Transfer
2. commands that interface with WinBugs
3. etc.

Today we will look at interfacing 2 other programs with Stata:

1. Sending emails from Stata
2. Producing a PDF log output that includes the log file and graphs

Introduction

Firstly looking at sending emails from Stata. Applications could be any of the following:

1. Sending updates by email about how the analysis is going

Introduction

Firstly looking at sending emails from Stata. Applications could be any of the following:

1. Sending updates by email about how the analysis is going
2. Sending an email with the Stata log file attached when a lengthy analysis has been completed.

Introduction

Firstly looking at sending emails from Stata. Applications could be any of the following:

1. Sending updates by email about how the analysis is going
2. Sending an email with the Stata log file attached when a lengthy analysis has been completed.
3. Tailoring emails based on the information contained in your Stata dataset

Introduction

Firstly looking at sending emails from Stata. Applications could be any of the following:

1. Sending updates by email about how the analysis is going
2. Sending an email with the Stata log file attached when a lengthy analysis has been completed.
3. Tailoring emails based on the information contained in your Stata dataset
4. Randomly selecting people in your Stata data set and sending them an email.

Introduction

Firstly looking at sending emails from Stata. Applications could be any of the following:

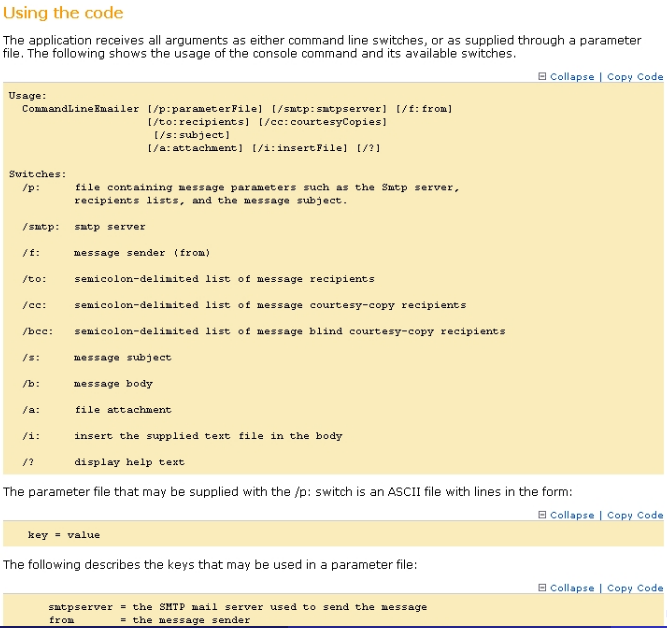
1. Sending updates by email about how the analysis is going
2. Sending an email with the Stata log file attached when a lengthy analysis has been completed.
3. Tailoring emails based on the information contained in your Stata dataset
4. Randomly selecting people in your Stata data set and sending them an email.

Downloading the program

To send emails we need to download another program. There are probably others, but for this example CommandLineEmailer has been used.
This can be obtained from:
http://www.codeproject.com/KB/IP/cpcommandlineemailer.aspx
eg. Download compiled utility - 6.05 Kb
(You must login to download - free and easy to do)

The program can be used with switches or a parameter file; we'll use the parameter file.

Program details



Using the code

The application receives all arguments as either command line switches, or as supplied through a parameter file. The following shows the usage of the console command and its available switches.

```
Usage:
  CommandLineEmailer [/p:parameterFile] [/smtp:smtpserver] [/f:from]
                     [/to:recipients] [/cc:courtesyCopies]
                      [/s:subject]
                     [/a:attachment] [/i:insertFile] [/?]

Switches:
  /p:      file containing message parameters such as the Smtp server,
           recipients lists, and the message subject.

  /smtp:   smtp server

  /f:      message sender (from)

  /to:     semicolon-delimited list of message recipients

  /cc:     semicolon-delimited list of message courtesy-copy recipients

  /bcc:    semicolon-delimited list of message blind courtesy-copy recipients

  /s:      message subject

  /b:      message body

  /a:      file attachment

  /i:      insert the supplied text file in the body

  /?       display help text
```

The parameter file that may be supplied with the /p: switch is an ASCII file with lines in the form:

```
   key = value
```

The following describes the keys that may be used in a parameter file:

```
      smtpserver = the SMTP mail server used to send the message
      from       = the message sender
```

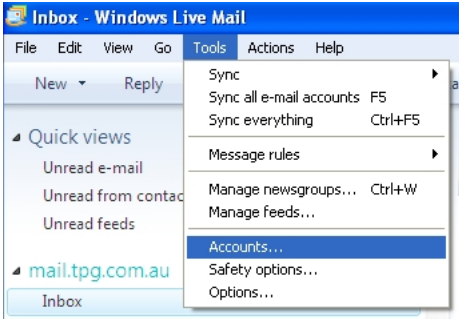
Code - Example 1

Emailing interim results.

```
capture erase kk2.txt
set more off

forvalues i=1/2000 {                                   // <–the program loop

                                                       // data to run the email program
 if mod(‘i’,100)==0 {                                  // <–1
  tempname fh
  file open ‘fh’ using kk2.txt, write                  // <–2
  file write ‘fh’ ”smtpserver = mail.whatever.com.au” _n  // <–3
  file write ‘fh’ ”from = myeamail@whatever.com.au” _n    // <–4
  file write ‘fh’ ”to = reciever@whatever.com.au” _n      // <–5
  file write ‘fh’ ”subject = Test Message” _n          // <–6
  file write ‘fh’ ”body = ‘i’ Test Message” _n         // <–7
  file close ‘fh’                                      // <–8

  !CommandLineEmailer /p:kk2.txt                       // <–9

  erase kk2.txt                                        // <–10
 }
}
exit
```

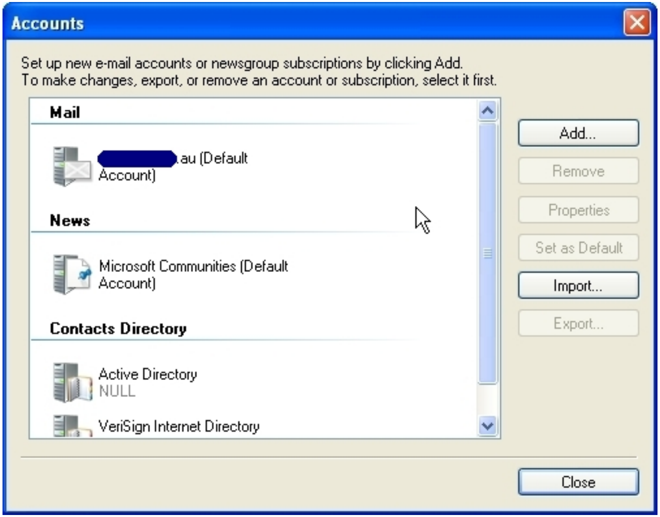
PDF log files

In Windows Mail - Tools - Accounts



PDF log files

Click onto Mail account and then properties



Accounts

Set up new e-mail accounts or newsgroup subscriptions by clicking Add.
To make changes, export, or remove an account or subscription, select it first.

Mail

[redacted].au (Default Account)

News

Microsoft Communities (Default Account)

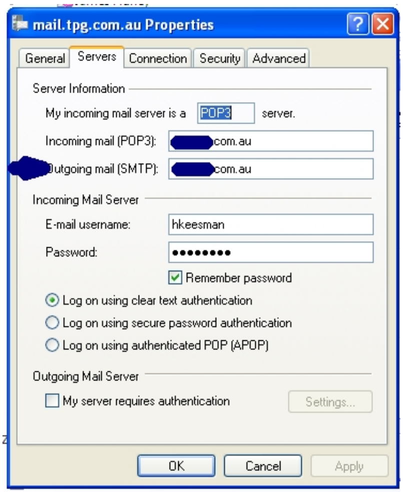
Contacts Directory

Active Directory
NULL

VeriSign Internet Directory

PDF log files

Click on Servers Tab



mail.tpg.com.au Properties

General | Servers | Connection | Security | Advanced

Server Information

My incoming mail server is a POP3 server.

Incoming mail (POP3): [redacted]com.au

[redacted]utgoing mail (SMTP): [redacted]com.au

Incoming Mail Server

E-mail username: hkeesman

Password: ●●●●●●●●●

[✓] Remember password

(●) Log on using clear text authentication

( ) Log on using secure password authentication

( ) Log on using authenticated POP (APOP)

Outgoing Mail Server

[ ] My server requires authentication

Code - Example 1

Emailing interim results.

```
capture erase kk2.txt
set more off

forvalues i=1/2000 {                                          // <–the program loop

                                                              // data to run the email program
 if mod(‘i’,100)==0 {                                         // <–1
  tempname fh
  file open ‘fh’ using kk2.txt, write                         // <–2
  file write ‘fh’ ”smtpserver = mail.whatever.com.au” _n      // <–3
  file write ‘fh’ ”from = myeamail@whatever.com.au” _n        // <–4
  file write ‘fh’ ”to = reciever@whatever.com.au” _n          // <–5
  file write ‘fh’ ”subject = Test Message” _n                 // <–6
  file write ‘fh’ ”body = ‘i’ Test Message” _n                // <–7
  file close ‘fh’                                             // <–8

  !CommandLineEmailer /p:kk2.txt                              // <–9

  erase kk2.txt                                               // <–10
 }
}
exit
```

Code - Example 2

Emailing an attachment:

```
capture erase kk2.txt
set more off
log using c:/kklog,text replace

forvalues i=1/2000 {                              //creating log file- to be emiled
display "Looping index: ‘i’"
}
log close

  tempname fh
  file open ‘fh’ using kk2.txt, write
  file write ‘fh’ "smtpserver = mail.whatever.com.au" _n
  file write ‘fh’ "from = myeamail@whatever.com.au" _n
  file write ‘fh’ "to = reciever@whatever.com.au" _n
  file write ‘fh’ "subject = Test Message" _n
  file write ‘fh’ "body = Test Message" _n
  file write ‘fh’ "attachment = c:\kklog.log" _n    // <–10
  file close ‘fh’

  !CommandLineEmailer /p:kk2.txt

  erase kk2.txt
exit
```

Code - Example 3

Today's example of an email that you wish to send:

Dear **Chev. Chevette** car owner,
Congratulations on owning a **Chev. Chevette** This car with a mileage of **29** ranked **67** out of **74** in our survey, which is a **VERY GOOD** result.

Regards

Example 3 - data

Getting the data ready for the email

```
cd c:/
capture erase kk2.txt
set more off
sysuse auto, clear
```

```
egen rank=rank(mpg) , unique
recode rank 1/20=1 21/40=2 41/60=3 61/74=4 ,gen(result)
label define rank1 1 bad 2 fair 3 good 4 ”very good”
label values result rank1
decode result, gen(r1)
```

Example 3 - email

Personalised email based on information in database:

```
if mod(‘i’,100)==0 {
 tempname fh
 file open ‘fh’ using kk2.txt, write
 file write ‘fh’ ”smtpserver = mail.whatever.com.au” _n
 file write ‘fh’ ”from = myeamail@whatever.com.au” _n
 file write ‘fh’ ”to = reciever@whatever.com.au” _n
 file write ‘fh’ ”subject = Test Message” _n
 file write ‘fh’ ”body = ‘i’ Test Message” _n
 file write ‘fh’ ”body = Congradulations on owning a ‘=make[‘i’]’ ”
 file write ‘fh’ ” This car with a milage of ‘=mpg[‘i’]’ ranked ‘=rank[‘i’]’ out of ‘=_N’
 in our survey, which is a ‘=upper(r1[‘i’])’ result” _n
 file write ‘fh’ ”body = ” _n
 file write ‘fh’ ”body = Regards ” _n
 file close ‘fh’

 !CommandLineEmailer /p:kk2.txt


 erase kk2.txt
}

exit
```

PDF log files

Converting a log file which includes graphs into a PDF file

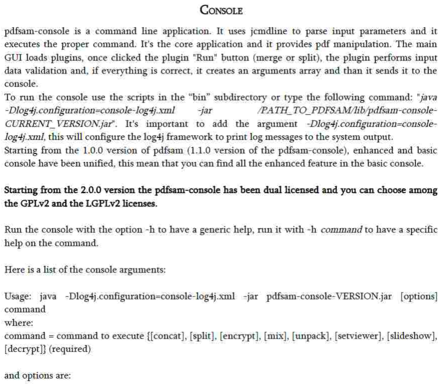
PDF log files

The second example of using another program within Stata is using Stata's 12 new Windows PDF translator for a log file and Stata's export graphic command.

For this we can use PDFsam
This program can be downloaded for free at:
http://www.pdfsam.org/

PDF log files

An extract from the tutorial:



CONSOLE

pdfsam-console is a command line application. It uses jcmdline to parse input parameters and it executes the proper command. It's the core application and it provides pdf manipulation. The main GUI loads plugins, once clicked the plugin "Run" button (merge or split), the plugin performs input data validation and, if everything is correct, it creates an arguments array and than it sends it to the console.
To run the console use the scripts in the "bin" subdirectory or type the following command: "*java -Dlog4j.configuration=console-log4j.xml -jar /PATH_TO_PDFSAM/lib/pdfsam-console-CURRENT_VERSION.jar*". It's important to add the argument *-Dlog4j.configuration=console-log4j.xml*, this will configure the log4j framework to print log messages to the system output.
Starting from the 1.0.0 version of pdfsam (1.1.0 version of the pdfsam-console), enhanced and basic console have been unified, this mean that you can find all the enhanced feature in the basic console.

Starting from the 2.0.0 version the pdfsam-console has been dual licensed and you can choose among the GPLv2 and the LGPLv2 licenses.

Run the console with the option -h to have a generic help, run it with -h *command* to have a specific help on the command.

Here is a list of the console arguments:

Usage: java -Dlog4j.configuration=console-log4j.xml -jar pdfsam-console-VERSION.jar [options] command
where:
command = command to execute {[concat], [split], [encrypt], [mix], [unpack], [setviewer], [slideshow], [decrypt]} (required)

and options are:

PDF log files

Produce a log file in the normal way and then break up the log file into sections; each section ending where a graph needs to be included. To do this look for a .gph extension. The entire log file is broken up this way.

Then the log file is translated to PDF and the graph is converted to a PDF using the graph export command.

Finally the PDF files are combined using PDFsam.

PDF log files

Example

```
cd c:/
clear all

capture erase output.pdf

//creating a do file
set more off
log using experiment, replace smcl
sysuse auto, clear
tab for
scatter mpg weight,saving(kk1, replace)
scatter mpg price, saving(kk2, replace)
tab for
log close
clear all
//end of creating a do file
```

PDF log files

```
program ltype, rclass
version 11.2
local using ‘0’

//picking up if smcl or txt
if strmatch("‘using’", "*.smcl") {
local log_ext smcl
}
else {
local log_ext="txt"
}

tempname fh
local linenum = 0
file open ‘fh’ using ‘"‘using’"’, read                    //existing log file
local a=1
tempname file‘a’

file open ‘file‘a’’ using file‘a’.txt, write replace text     //bit of log file
file read ‘fh’ line
while r(eof)==0 {
local linenum = ‘linenum’ + 1
//picking up where the graph is looking for .gph,
if ///
strmatch("‘line’","*{ txt} *.gph*") |   ///
```

PDF log files

```
local aa =word("‘line’",2)                          //name of graph
local linenum1 "‘linenum1’ ‘linenum’"
local linenum = 0
//pdf list
local list "‘list’ file‘a’.‘log_ext’"               //log
local list1 "‘list1’ -f file‘a’.pdf"                //log

//pdf graphs
local list "‘list’ ‘aa’"                            //graph
local list1 "‘list1’ -f kk‘a’.pdf"                  //graph

file close ‘file‘a’’
local ++a
tempname file‘a’
file open ‘file‘a’’ using file‘a’.txt, write replace text
file read ‘fh’ line
continue
}
else {
file read ‘fh’ line
file write ‘file‘a’’ "‘line’" _newline
}
}

local linenum1 "‘linenum1’ ‘linenum’"
local list "‘list’ file‘a’.‘log_ext’"
local list1 "‘list1’ -f file‘a’.pdf"

file close ‘fh’
file close ‘file‘a’’
```

PDF log files

```
return local ext ="‘log_ext’"
return local lines ="‘linenum1’"
return local kklist1="‘list1’" return local kklist ="‘list’"
end
```

PDF log files

```
//Calls above program and passes the name of the log file to it
ltype c:/experiment.smcl

local m=r(kklist1)
local m1=r(ext)
//converting log files and graphs to pdf
local z1=1
local z=1
tokenize ‘r(lines)’

foreach i in ‘r(kklist)’ {
//pdf graph
if mod(‘z1’,2)==0 {
graph use ‘i’
graph export kk‘z’.pdf, replace
local ++z
}
else {                                                   //log file to pdf

local zz ‘=ceil(‘1’/6)’                                  // unit scaling
if ‘zz’<4 local zz 4
```

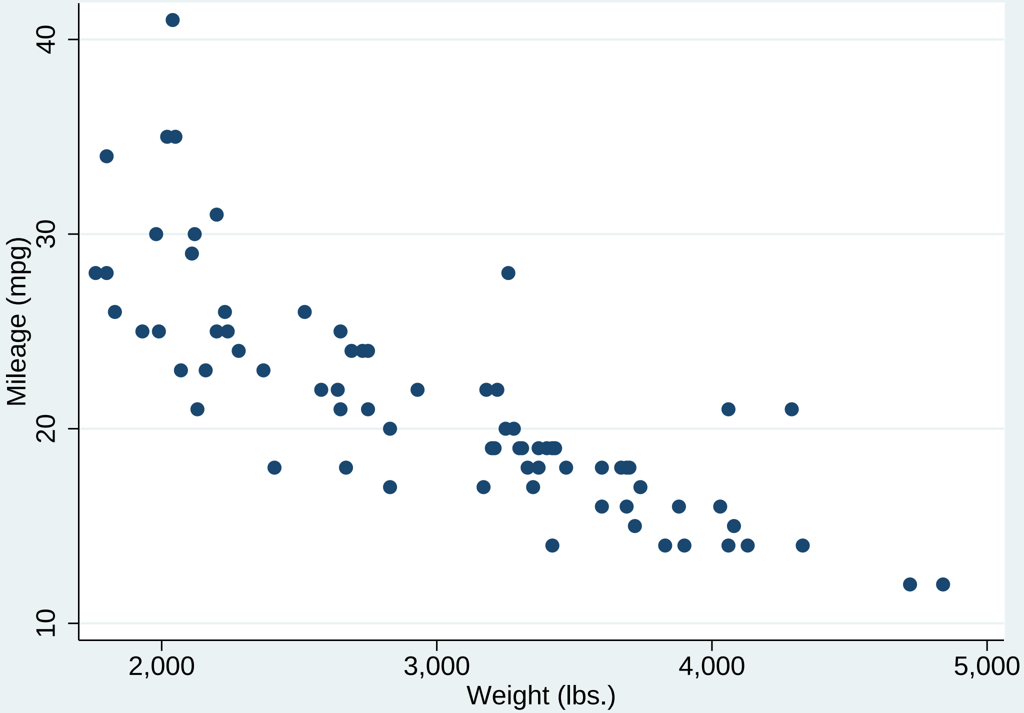
PDF log files

```
if "‘m1’"=="smcl" {
translator set smcl2pdf pagesize custom
translator set smcl2pdf pageheight ‘zz’
translator set smcl2pdf tmargin 0.5
translator set smcl2pdf bmargin 0.5

translate file‘z’.txt file‘z’.pdf , translator(smcl2pdf) logo(off) header(off) cmdnumber(off)
capture macro shift
}
if "‘m1’"=="log" {
translator set log2pdf pagesize custom
translator set log2pdf pageheight ‘zz’
translator set log2pdf tmargin 0.5
translator set log2pdf bmargin 0.5
translate file‘z’.txt file‘z’.pdf , translator(log2pdf) logo(off) header(off) cmdnumber(off)
capture macro shift
} //if log
} //if not graph
local ++z1
}                                                                                              //loop
! "C:\Program Files\Java\jre6\bin\java.exe" ///
–Dlog4j.configuration=console4j.xml ///
"C:Files–pdfsam–lib–pdfsam–textendashconsole –2.3.1e.jar" ‘m’ ///
–o c:\output.pdf concat
exit
```

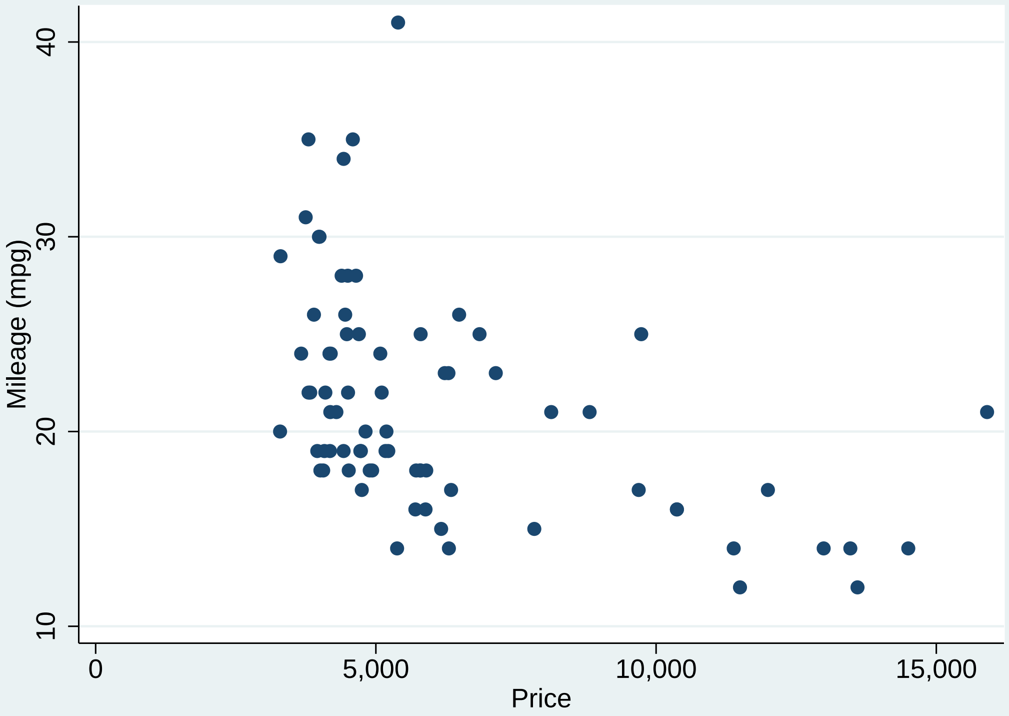
PDF log files

And the result is:

```
-------------------------------------------------------------------------------------
      name:  <unnamed>
       log:  c:\experiment.smcl
  log type:  smcl
 opened on:   7 Sep 2011, 21:08:34

. sysuse auto, clear
(1978 Automobile Data)

. tab for

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00

. scatter mpg weight,saving(kk1, replace)
(file kk1.gph saved)
```


Mileage (mpg)
40
30
20
10
2,000
3,000
4,000
5,000
Weight (lbs.)

```
. scatter mpg price, saving(kk2, replace)
(file kk2.gph saved)
```


Mileage (mpg)
10
20
30
40
0
5,000
10,000
15,000
Price

```
. tab for

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00

. log close
      name:  <unnamed>
       log:  c:\experiment.smcl
  log type:  smcl
 closed on:   7 Sep 2011, 21:08:38
-------------------------------------------------------------------------------------
```