

## Introduction to Stata

Stata is a general-purpose statistical software package created in 1985 that offers users a graphical interface to manage, analyze, and graph data. The word 'Stata' is a combination of the words 'statistics' and 'data.' Stata utilizes command line interface so users can type commands to perform specific tasks. In addition, Stata has menus and dialog boxes that give the user access to nearly all built-in commands.

User-written commands can be added to Stata using ado-files. Stata is case-sensitive; thus, it distinguishes between lower and upper case letters. Most Stata built-in commands are lower case, a convention most programmers follow.

Looking at Stata, you'll see four principle boxes: Results, Command, Variables, and Review.

Results window displays your input and output, which includes output procedures. If a command generates lengthy output that one does not want to display, the user can type quietly in front of that command.

Command window is where the user enters a command. To run a command, press Enter. Stata understands most abbreviations for commands and variable names, as long as the abbreviation is unique. For example, the user can abbreviate the command regress to reg. However, imagine two variables named perseat and percabinet. Stata would be unable to distinguish between the two variables if the abbreviation per was used. However, pers and perc would be acceptable. Further, the user could call both variables using an asterisk, per\*.

Variables window displays the variables listed in the data set. This will be blank when there is no data in Stata's memory. The user can click on variables to include them on the command line.

Review window records all previously entered commands. The user can click on any past command to include it on the command line. Or, the user can page-up or page-down to access past commands in the command box.

```
.clear
```

Type clear in the command box to clear Stata's memory.

## Log-Files

A log is a record of your Stata session. It is ideal to maintain a record of everything you do when working with data|this reduces the possibility of ever wondering what you just did, or, even worse, wondering what you did two weeks ago. The user specifies when to start and save a log-file, log using "file path", and when to end a log-file, log close. This functions well when entering commands line-by-line in the command box or running batch commands in a do-file. As a quick example, I'll show you how to start a log file, run a simple command, and close the log file.

Then, you'll go to where the log-file is saved to see the output.

```
.log using <.....>
```

In the command box, type log using followed by where you want the document saved on your computer. Next, upload the .csv data file using insheet using followed by where you saved the .csv file. Type describe and look at the output. Next, type log close to end the log file. Finally, go to where you saved the log document and open the log-file. This will be a complete replication of everything you just did in Stata after opening the log-file and before closing the log-file.

```
. log using "C://Stata/StataOverview.log", replace
```

```
-----  
log: C://Stata/StataOverview.log
```

```
log type: text
```

```
opened on: 19 May 2015, 09:47:57
```

```
sysuse auto.dta, clear
```

```
(12 vars, 74 obs)
```

```
. describe
Contains data
obs: 74
vars: 12
size: 3,626 (99.7% of memory free)
```

```
-----
storage display value
variable name type format label variable label
-----
```

```
make str17 %17s
price int %8.0g
mpg byte %8.0g
rep78 byte %8.0g
headroom float %9.0g
trunk byte %8.0g
weight int %8.0g
length int %8.0g
turn byte %8.0g
displacement int %8.0g
gear_ratio float %9.0g
foreign str8 %9s
-----
```

Sorted by:

Note: dataset has changed since last saved

```
. log close
```

```
log: /Users/jasoneichorst/documents/Program Help Files/Stata
/StataOverview/Guide.log
```

```
log type: text
```

```
closed on: 19 May 2015, 09:48:19
-----
```

## Delimit

Some commands are followed by a lot of information. Thus, using a single line can be overwhelming and difficult to read. This is particularly the case for path names and when producing graphs, which is explained below. The user can instruct Stata when to read a line break as a new command or when it is apart of the line above it. This is accomplished using # delimit to introduce line breaks in your commands without 'confusing' Stata. When using delimit, the user must use a semi-colon to indicate the 'end' of every command and comment.

This is an important feature when using do-files. I will replicate the above do-file using the delimit command. Delimit is started using # delimit ; and ended using # delimit cr.

The main difference between below and above is that the file path can be written on two lines in Stata, instead of written on one long line. Notice the user of the semi-colon for command lines and comment lines while using # delimit.

```
/* Purpose: I use commands in this do-file that will be used to develop
```

```
A Beginner's Guide to Using Stata. */
```

```
* Stata Version 10
```

```
# delimit ;
```

```
* start log file;
```

```
log using "/Users/jasoneichorst/documents/Program Help Files/Stata
/StataOverview/Guide.log", replace;
```

```
* close log file;
log close;
# delimit cr
```

In the remaining portion of this guide, I simply show what the command looks like and use line breaks for presentational purposes. The reader has the option of using # delimit in his/her do-file or not.

### Programming

Stata provides a wide variety of built-in commands, but not in\_nite. However, Stata is programmable, which allows for new features to be added. This is accomplished using ado-files. Ado-files are indistinguishable to built-in commands, but are important for adapting Stata to perform your specific task, or use a command that performs a task that somebody else has developed. Ado-files can easily be installed; in fact, there are seven different locations, which can be categorized in three ways, where Stata looks for ado-files. For further detail, I suggest reading the [U] User's Guide or [P] Programming manuals.

### Do files

Do-Files are perfect for reproducing your work, which is important when applying the scientific method. Instead of entering and running commands line-by-line in the command box, do-files allow the user to place commands in a text file and run them in batch. If one wishes, the user can also run sections (or even single lines) of an entire do-file. This is accomplished by highlighting the part of the do-file of interest and clicking on the `do' icon or the `run' icon (for `quiet' evaluations) at the top right of the page. The user can click on the same icons without highlighting sections to run the entire file. This text file can easily be saved and run again without having to rewrite every command used in your prior analysis. To open a new do-file, go to File, in the menu bar, and select \New Do-file." You can also select the do-file icon, which looks like a notebook pad, to open a new do-file.

Stata reads commands. However, there are many times when the users wants to include \comments" about a command or further explain the motivations behind the code. For example, the user may want to provide a reason for using a particular command so he/she can recall the motivations when reading the do\_ile two years in the future. To perform this task, the user need to \comment out" lines that are used for explanation. Comment indicators include \* for single lines, and /\* and \*/ for multiple lines. In addition, the user should also use the first few lines to state your name, the project/purpose, date, and Stata version. Also, you can also create log-files in your do-files.

This is how a typical do-file will look using single line comments (name, date and version), multiple line comments (purpose), and two commands (log commands).

### Examples of do-files

```
* Demo of do file #1
clear
sysuse sp500
describe
summarize
generate volatility=high-low
generate lnvolume=ln(volume)
generate time=_n
graph twoway line close time
more
```

```

graph twoway scatter change volatility
correlate change volatility
regress change volatility lnvolume
generate changel=change[_n-1]
regress change changel volatility lnvolume

```

\* Demo of do file #2

```

clear
infile year gasexp pop gasp income pnc puc ppt pd pn ps ///
using "c:\course09\ec570\data\gasoline.txt"
label data "Greene [2008], Table F2.2: The U.S. Gasoline Market"
label variable year "Year, 1953-2004"
label variable gasexp "Total U.S. gasoline expenditure"
label variable pop "U.S. total population in thousands"
label variable gasp "Price index for gasoline"
label variable income "Per capita disposable income"
label variable pnc "Price index for new cars"
label variable puc "Price index for used cars"
label variable ppt "Price index for public transportation"
label variable pd "Aggregate price index for consumer durables"
label variable pn "Aggregate price index for consumer nondurables"
label variable ps "Aggregate price index for consumer services"
describe
summarize
* save it as a Stata dataset
save "c:\course09\ec570\data\gasoline", replace
/*
generate g=ln(gasexp/pop) // log-per-capita gas consumption
generate y=ln(income/pop) // log-per-capita income
generate pg=ln(gasp) // log price of gas
generate pnew=ln(pnc) // log price of new cars
generate pused=ln(puc) // log price of used cars
regress g y pg pnew pused
*/

```

\* Demo of do file #3

```

set more off
clear
//use "c:\course09\ec570\data\gasoline"
use http://www.econ.pdx.edu/faculty/KPL/ec570/data/gasoline
describe
summarize
generate g=ln(gasexp/pop) // log-per-capita gas consumption
generate y=ln(income/pop) // log-per-capita income
generate pg=ln(gasp) // log price of gas
generate pnew=ln(pnc) // log price of new cars
generate pused=ln(puc) // log price of used cars
regress g y pg pnew pused
// dynamic model with lagged variables

```

```
generate g1=g[_n-1]
regress g y pg_pnew pused g1
// hypothesis testing
test pnew pused
test pnew+pused=0
// prediction
predict e, residual
predict ghat, xb
twoway scatter e ghat
// structural break
```

### Must-know commands

- System
  - clear
  - exit
  - log
  - set
  - # delimiter
  - net
  - search
  - help
  
- Data Management
  - use
  - infile, infix
  - list
  - describe
  - keep, drop
  - generate, replace, rename
  - save, outfile
  
- Data Analysis
  - summarize
  - correlate
  - graph
    - twoway, scatter,...
    - hist
  
- Statistical Analysis
  - regress
  - predict
  - test
  - dwstat
  - hettest

### Web Data

First, use `webuse set [http:// ...]` to specify the URL from which the data set will be

obtained. The command `webuse query` reports the current URL. You must do this prior to loading the data set, unless the default URL is the target URL. The command to set the default URL is `webuse set`, without a subsequent URL.

```
. webuse query
(prefix now "http://www.stata-press.com/data/r10")
. webuse set http://www.stata-press.com/data/r9/
(prefix now "http://www.stata-press.com/data/r9")
Second, use webuse \_lename to load the data set over the web
. webuse auto.dta, clear
(1978 Automobile Data)
Or, you can simply enter
. use http://www.stata-press.com/data/r9/auto.dta, clear
(1978 Automobile Data)
```

The same can be done for `.csv` or `.txt` data online with `insheet` using.

```
. insheet using http://www.owlnet.rice.edu/~jaeichorst/auto.csv, clear
(13 vars, 74 obs)
. insheet using http://www.owlnet.rice.edu/~jaeichorst/auto.txt, clear
(13 vars, 74 obs)
```

#### Change the Directory

This is what you have been waiting for! Changing the directory vastly simplifies the programming process. More specifically, `cd` changes the working directory to the specified drive and directory. This makes it possible to upload data by just using the `_le` name. In addition, when you share do-`_les` with colleagues, they can just change the directory instead of changing the file path in the necessary commands. Notice the difference between the file names below and the file names above after I change the directory. Use `cd` to accomplish this task.

```
. cd "/Users/jasoneichorst/documents/Program Help Files/Stata/StataOverview/"
/Users/jasoneichorst/Documents/Program Help Files/Stata/StataOverview/
```

#### Stata Data

Stata uses `.dta` format and can be uploaded using the `use` command. After changing the directory, I only have to use the `_le` name in that directory to perform the task. Of course, the file must be in that directory!

```
. use auto.dta, clear
(1978 Automobile Data)
```

#### Analyze Data

Stata offers a variety of commands so that the user can efficiently and effectively analyze the data without looking directly at the data set. Most of Stata's commands share a common syntax, which is `[prefix command:] command [varlist ] [if ] [in] [, options]`

where items enclosed in square brackets are optional.

`describe` produces a description of the data set in memory, listing the variable names and their labels.

```
.sysuse auto.dta, clear
. describe
```

Contains data from `auto.dta`

obs: 74 1978 Automobile Data

vars: 12 18 Aug 2009 20:46

size: 3,478 (99.7% of memory free) (`_dta` has notes)

---

```
storage display value
variable name type format label variable label
```

---

```
make str18 %-18s Make and Model
price int %8.0gc Price
mpg int %8.0g Mileage (mpg)
rep78 int %8.0g Repair Record 1978
headroom float %6.1f Headroom (in.)
trunk int %8.0g Trunk space (cu. ft.)
weight int %8.0gc Weight (lbs.)
length int %8.0g Length (in.)
turn int %8.0g Turn Circle (ft.)
displacement int %8.0g Displacement (cu. in.)
gear_ratio float %6.2f Gear Ratio
foreign byte %8.0g origin Car type
```

---

```
. label variable price "too many dollars"
. describe
```

```
Contains data from auto.dta
obs: 74 1978 Automobile Data
vars: 12 20 Aug 2009 12:10
size: 3,478 (99.7% of memory free) (_dta has notes)
```

---

```
storage display value
variable name type format label variable label
```

---

```
make str18 %-18s Make and Model
price int %8.0gc too many dollars
mpg int %8.0g Mileage (mpg)
rep78 int %8.0g Repair Record 1978
headroom float %6.1f Headroom (in.)
trunk int %8.0g Trunk space (cu. ft.)
weight int %8.0gc Weight (lbs.)
length int %8.0g Length (in.)
turn int %8.0g Turn Circle (ft.)
displacement int %8.0g Displacement (cu. in.)
gear_ratio float %6.2f Gear Ratio
foreign byte %8.0g origin Car type
```

---

```
Sorted by: foreign
```

```
summary calculates and displays a variety of summary statistics. If a variable list is not
speci_ed, summary statistics are calculated for all of the variables in the data set. However, the
user can specify variables to be summarized. You can also make conditional statements
to summarize specific observations.
```

```
. sum
```

```
Variable | Obs Mean Std. Dev. Min Max
```

---

```
make | 0
price | 74 6165.257 2949.496 3291 15906
```

```
mpg | 74 21.2973 5.785503 12 41
rep78 | 69 3.405797 .9899323 1 5
headroom | 74 2.993243 .8459948 1.5 5
```

```
-----+-----
trunk | 74 13.75676 4.277404 5 23
weight | 74 3019.459 777.1936 1760 4840
length | 74 187.9324 22.26634 142 233
turn | 74 39.64865 4.399354 31 51
displacement | 74 197.2973 91.83722 79 425
```

```
-----+-----
gear_ratio | 74 3.014865 .4562871 2.19 3.89
foreign | 74 .2972973 .4601885 0 1
```

```
. sum mpg trunk weight
Variable | Obs Mean Std. Dev. Min Max
```

```
-----+-----
mpg | 74 21.2973 5.785503 12 41
trunk | 74 13.75676 4.277404 5 23
weight | 74 3019.459 777.1936 1760 4840
Let's just look at summary statistics for foreign vehicles.
```

```
. sum mpg trunk weight if foreign
Variable | Obs Mean Std. Dev. Min Max
```

```
-----+-----
mpg | 22 24.77273 6.611187 14 41
trunk | 22 11.40909 3.216906 5 16
weight | 22 2315.909 433.0035 1760 3420
10
```

list lists values of variables. If variables are not speci\_ed, then all of the variables are listed. However, the user can specify variables to reduce the size of the list. This is the easiest way to see the data without opening the data editor. In this example, I specify variables to reduce the size of the output.

```
. list make price mpg
```

```
+-----+
| make price mpg |
|-----|
1. | AMC Concord 4,099 22 |
2. | AMC Pacer 4,749 17 |
3. | AMC Spirit 3,799 22 |
4. | Buick Century 4,816 20 |
5. | Buick Electra 7,827 15 |
|-----|
6. | Buick LeSabre 5,788 18 |
7. | Buick Opel 4,453 26 |
8. | Buick Regal 5,189 20 |
9. | Buick Riviera 10,372 16 |
10. | Buick Skylark 4,082 19 |
|-----|
11. | Cad. Deville 11,385 14 |
12. | Cad. Eldorado 14,500 14 |
13. | Cad. Seville 15,906 21 |
14. | Chev. Chevette 3,299 29 |
15. | Chev. Impala 5,705 16 |
```



|-----|  
16. | Chev. Malibu 4,504 22 |  
17. | Chev. Monte Carlo 5,104 22 |  
18. | Chev. Monza 3,667 24 |  
19. | Chev. Nova 3,955 19 |  
20. | Dodge Colt 3,984 30 |  
|-----|  
21. | Dodge Diplomat 4,010 18 |  
22. | Dodge Magnum 5,886 16 |  
23. | Dodge St. Regis 6,342 17 |  
24. | Ford Fiesta 4,389 28 |  
25. | Ford Mustang 4,187 21 |  
|-----|  
26. | Linc. Continental 11,497 12 |  
27. | Linc. Mark V 13,594 12 |  
28. | Linc. Versailles 13,466 14 |  
29. | Merc. Bobcat 3,829 22 |  
30. | Merc. Cougar 5,379 14 |  
11  
|-----|  
31. | Merc. Marquis 6,165 15 |  
32. | Merc. Monarch 4,516 18 |  
33. | Merc. XR-7 6,303 14 |  
34. | Merc. Zephyr 3,291 20 |  
35. | Olds 98 8,814 21 |  
|-----|  
36. | Olds Cutl Supr 5,172 19 |  
37. | Olds Cutlass 4,733 19 |  
38. | Olds Delta 88 4,890 18 |  
39. | Olds Omega 4,181 19 |  
40. | Olds Starfire 4,195 24 |  
|-----|  
41. | Olds Toronado 10,371 16 |  
42. | Plym. Arrow 4,647 28 |  
43. | Plym. Champ 4,425 34 |  
44. | Plym. Horizon 4,482 25 |  
45. | Plym. Sapporo 6,486 26 |  
|-----|  
46. | Plym. Volare 4,060 18 |  
47. | Pont. Catalina 5,798 18 |  
48. | Pont. Firebird 4,934 18 |  
49. | Pont. Grand Prix 5,222 19 |  
50. | Pont. Le Mans 4,723 19 |  
|-----|  
51. | Pont. Phoenix 4,424 19 |  
52. | Pont. Sunbird 4,172 24 |  
53. | Audi 5000 9,690 17 |  
54. | Audi Fox 6,295 23 |  
55. | BMW 320i 9,735 25 |  
|-----|  
56. | Datsun 200 6,229 23 |

```

57. | Datsun 210 4,589 35 |
58. | Datsun 510 5,079 24 |
59. | Datsun 810 8,129 21 |
60. | Fiat Strada 4,296 21 |
|-----|
61. | Honda Accord 5,799 25 |
62. | Honda Civic 4,499 28 |
63. | Mazda GLC 3,995 30 |
64. | Peugeot 604 12,990 14 |
65. | Renault Le Car 3,895 26 |
|-----|
66. | Subaru 3,798 35 |
67. | Toyota Celica 5,899 18 |
68. | Toyota Corolla 3,748 31 |
69. | Toyota Corona 5,719 18 |
12
70. | VW Dasher 7,140 23 |
|-----|
71. | VW Diesel 5,397 41 |
72. | VW Rabbit 4,697 25 |
73. | VW Scirocco 6,850 25 |
74. | Volvo 260 11,995 17 |
+-----+

```

You can list all variables starting with the letter m and only if price is less than 4000.

```
. list m* if price<4000
```

```

+-----+
| make mpg |
|-----|
3. | AMC Spirit 22 |
14. | Chev. Chevette 29 |
18. | Chev. Monza 24 |
19. | Chev. Nova 19 |
20. | Dodge Colt 30 |
|-----|
29. | Merc. Bobcat 22 |
34. | Merc. Zephyr 20 |
63. | Mazda GLC 30 |
65. | Renault Le Car 26 |
66. | Subaru 35 |
|-----|
68. | Toyota Corolla 31 |
+-----+

```

tabulate displays one and two dimensional frequency tables.

```
. tabulate foreign
```

```
Car type | Freq. Percent Cum.
```

```

-----+-----
Domestic | 52 70.27 70.27
Foreign | 22 29.73 100.00
-----+-----

```

```
Total | 74 100.00
```

```
. tabulate rep78
```

Repair |  
Record 1978 | Freq. Percent Cum.

```
-----+-----  
1 | 2 2.90 2.90  
2 | 8 11.59 14.49  
3 | 30 43.48 57.97  
4 | 18 26.09 84.06  
5 | 11 15.94 100.00  
-----+-----
```

```
Total | 69 100.00  
. tabulate foreign rep78  
| Repair Record 1978  
Car type | 1 2 3 4 5 | Total  
-----+-----+-----
```

```
Domestic | 2 8 27 9 2 | 48  
Foreign | 0 0 3 9 9 | 21  
-----+-----+-----
```

```
Total | 2 8 30 18 11 | 69
```

sort arranges the observations of the current data into ascending order based on the values of the speci\_ed variables. To shorten the length of the output, the user can specify the number of rows to be displayed|this option is not limited to list.

```
. sort price  
. list make price mpg in 1/10
```

```
+-----+  
| make price mpg |  
|-----|  
1. | Merc. Zephyr 3,291 20 |  
2. | Chev. Chevette 3,299 29 |  
3. | Chev. Monza 3,667 24 |  
4. | Toyota Corolla 3,748 31 |  
5. | Subaru 3,798 35 |  
|-----|  
6. | AMC Spirit 3,799 22 |  
7. | Merc. Bobcat 3,829 22 |  
8. | Renault Le Car 3,895 26 |  
9. | Chev. Nova 3,955 19 |  
10. | Dodge Colt 3,984 30 |  
+-----+  
|-----|
```

generate creates a new variable. There is another command called egen, which includes extensions for generate. For now, we will focus on the simpler command, generate

```
. gen pricePERmileage = price/mpg  
. list make pricePERmileage price mpg in 1/10
```

```
+-----+  
| make priceP~e price mpg |  
|-----|  
1. | Merc. Zephyr 164.55 3,291 20 |  
2. | Chev. Chevette 113.7586 3,299 29 |  
3. | Chev. Monza 152.7917 3,667 24 |  
4. | Toyota Corolla 120.9032 3,748 31 |
```

```

5. | Subaru 108.5143 3,798 35 |
|-----|
6. | AMC Spirit 172.6818 3,799 22 |
7. | Merc. Bobcat 174.0455 3,829 22 |
8. | Renault Le Car 149.8077 3,895 26 |
9. | Chev. Nova 208.1579 3,955 19 |
10. | Dodge Colt 132.8 3,984 30 |
+-----+

```

drop eliminates variables or observations from the data in memory.

I use preserve and restore so that I can return to using the original data set after showing some examples using drop. preserve and restore must be used together; they are a good tool for editing data while maintaining the ability to return to the original data set. Also, notice that = and == serve different purposes. A single = is used for assignment, whereas double == is used for equality.

```

. preserve
. drop headroom trunk length turn displacement gear_ratio rep78
. list in 1/10

```

```

+-----+
| make price mpg weight foreign priceP~e |
|-----|
1. | Merc. Zephyr 3,291 20 2,830 Domestic 164.55 |
2. | Chev. Chevette 3,299 29 2,110 Domestic 113.7586 |
3. | Chev. Monza 3,667 24 2,750 Domestic 152.7917 |
4. | Toyota Corolla 3,748 31 2,200 Foreign 120.9032 |
5. | Subaru 3,798 35 2,050 Foreign 108.5143 |
|-----|
6. | AMC Spirit 3,799 22 2,640 Domestic 172.6818 |
7. | Merc. Bobcat 3,829 22 2,580 Domestic 174.0455 |
8. | Renault Le Car 3,895 26 1,830 Foreign 149.8077 |
9. | Chev. Nova 3,955 19 3,430 Domestic 208.1579 |
15
10. | Dodge Colt 3,984 30 2,120 Domestic 132.8 |
+-----+

```

```

. drop if price >=5000
(37 observations deleted)
. list in 1/10

```

```

+-----+
| make price mpg weight foreign priceP~e |
|-----|
1. | Merc. Zephyr 3,291 20 2,830 Domestic 164.55 |
2. | Chev. Chevette 3,299 29 2,110 Domestic 113.7586 |
3. | Chev. Monza 3,667 24 2,750 Domestic 152.7917 |
4. | Toyota Corolla 3,748 31 2,200 Foreign 120.9032 |
5. | Subaru 3,798 35 2,050 Foreign 108.5143 |
|-----|
6. | AMC Spirit 3,799 22 2,640 Domestic 172.6818 |
7. | Merc. Bobcat 3,829 22 2,580 Domestic 174.0455 |
8. | Renault Le Car 3,895 26 1,830 Foreign 149.8077 |
9. | Chev. Nova 3,955 19 3,430 Domestic 208.1579 |
10. | Dodge Colt 3,984 30 2,120 Domestic 132.8 |
+-----+

```

```

. drop if foreign==0
(29 observations deleted)
. list
+-----+
| make price mpg weight foreign priceP~e |
|-----|
1. | Toyota Corolla 3,748 31 2,200 Foreign 120.9032 |
2. | Subaru 3,798 35 2,050 Foreign 108.5143 |
3. | Renault Le Car 3,895 26 1,830 Foreign 149.8077 |
4. | Mazda GLC 3,995 30 1,980 Foreign 133.1667 |
5. | Fiat Strada 4,296 21 2,130 Foreign 204.5714 |
|-----|
6. | Honda Civic 4,499 28 1,760 Foreign 160.6786 |
7. | Datsun 210 4,589 35 2,020 Foreign 131.1143 |
8. | VW Rabbit 4,697 25 1,930 Foreign 187.88 |
+-----+
. drop _all /* note: drop _all removes all variables from the data set without affecting value
labels, macros, and programs. Clear has the same result as drop _all, but also clears value
labels, matrices, scalars, constraints, and equalities; closes all open files and postfiles; clears
saved results; and clears Mata. */
. list
. restore

```

After using restore, you'll see that everything is back to the original.

```
. summarize
```

```
Variable | Obs Mean Std. Dev. Min Max
```

```
-----+
```

```
make | 0
price | 74 6165.257 2949.496 3291 15906
mpg | 74 21.2973 5.785503 12 41
rep78 | 69 3.405797 .9899323 1 5
headroom | 74 2.993243 .8459948 1.5 5
```

```
-----+
```

```
trunk | 74 13.75676 4.277404 5 23
weight | 74 3019.459 777.1936 1760 4840
length | 74 187.9324 22.26634 142 233
turn | 74 39.64865 4.399354 31 51
displacement | 74 197.2973 91.83722 79 425
```

```
-----+
```

```
gear_ratio | 74 3.014865 .4562871 2.19 3.89
foreign | 74 .2972973 .4601885 0 1
pricePERmi~e | 74 330.9413 237.2313 108.5143 1132.833
```

keep works the same way as drop, except that you specify the variables or observations to be kept rather than the variables or observations to be deleted.

```
. preserve
. keep make mpg price foreign
. list in 1/10
```

```
+-----+
```

```
| make price mpg foreign |
|-----|
```

```
1. | Merc. Zephyr 3,291 20 Domestic |
2. | Chev. Chevette 3,299 29 Domestic |
3. | Chev. Monza 3,667 24 Domestic |
4. | Toyota Corolla 3,748 31 Foreign |
5. | Subaru 3,798 35 Foreign |
17
```

```
|-----|
6. | AMC Spirit 3,799 22 Domestic |
7. | Merc. Bobcat 3,829 22 Domestic |
8. | Renault Le Car 3,895 26 Foreign |
9. | Chev. Nova 3,955 19 Domestic |
10. | Dodge Colt 3,984 30 Domestic |
```

```
+-----+
```

```
. keep if foreign==1
(52 observations deleted)
. list
```

```
+-----+
```

```
| make price mpg foreign |
|-----|
1. | Toyota Corolla 3,748 31 Foreign |
2. | Subaru 3,798 35 Foreign |
3. | Renault Le Car 3,895 26 Foreign |
4. | Mazda GLC 3,995 30 Foreign |
5. | Fiat Strada 4,296 21 Foreign |
```

```
|-----|
6. | Honda Civic 4,499 28 Foreign |
7. | Datsun 210 4,589 35 Foreign |
8. | VW Rabbit 4,697 25 Foreign |
9. | Datsun 510 5,079 24 Foreign |
10. | VW Diesel 5,397 41 Foreign |
```

```
|-----|
11. | Toyota Corona 5,719 18 Foreign |
12. | Honda Accord 5,799 25 Foreign |
13. | Toyota Celica 5,899 18 Foreign |
14. | Datsun 200 6,229 23 Foreign |
15. | Audi Fox 6,295 23 Foreign |
```

```
|-----|
16. | VW Scirocco 6,850 25 Foreign |
17. | VW Dasher 7,140 23 Foreign |
18. | Datsun 810 8,129 21 Foreign |
19. | Audi 5000 9,690 17 Foreign |
20. | BMW 320i 9,735 25 Foreign |
```

```
|-----|
21. | Volvo 260 11,995 17 Foreign |
22. | Peugeot 604 12,990 14 Foreign |
```

```
+-----+
```

```
. restore
. summarize
```

```
Variable | Obs Mean Std. Dev. Min Max
```

```
-----+
```

```
make | 0
price | 74 6165.257 2949.496 3291 15906
mpg | 74 21.2973 5.785503 12 41
rep78 | 69 3.405797 .9899323 1 5
headroom | 74 2.993243 .8459948 1.5 5
-----+-----
trunk | 74 13.75676 4.277404 5 23
weight | 74 3019.459 777.1936 1760 4840
length | 74 187.9324 22.26634 142 233
turn | 74 39.64865 4.399354 31 51
displacement | 74 197.2973 91.83722 79 425
-----+-----
gear_ratio | 74 3.014865 .4562871 2.19 3.89
foreign | 74 .2972973 .4601885 0 1
pricePERmi~e | 74 330.9413 237.2313 108.5143 1132.833
```