**Welcome to the Institute for Digital Reseach and Education**

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
Descriptive information and statistics**

This module shows common commands for showing descriptive information and descriptive statistics about data files.

**Getting an overview of your file**

The **sysuse** command loads a specified Stata-format dataset that was shipped with Stata. Here we will use the **auto** data file.

**sysuse** auto

The **describe** command shows you basic information about a Stata data file. As you can see, it tells us the number of observations in the file, the number of variables, the names of the variables, and more.

**describe**

Contains data from auto.dta

obs: 74

vars: 12 17 Feb 1999 10:49

size: 3,108 (99.6% of memory free)

-------------------------------------------------------------------------------

1. make str17 %17s

2. price int %9.0g

3. mpg byte %9.0g

4. rep78 byte %9.0g

5. hdroom float %9.0g

6. trunk byte %9.0g

7. weight int %9.0g

8. length int %9.0g

9. turn byte %9.0g

10. displ int %9.0g

11. gratio float %9.0g

12. foreign byte %9.0g

-------------------------------------------------------------------------------

Sorted by:

The **codebook** command is a great tool for getting a quick overview of the variables in the data file. It produces a kind of electronic codebook from the data file. Have a look at what it produces below.

**codebook**

make -------------------------------------------------------------- (unlabeled)

type: string (str17)

unique values: 74 coded missing: 0 / 74

examples: "Cad. Deville"

"Dodge Magnum"

"Merc. XR-7"

"Pont. Catalina"

warning: variable has embedded blanks

price ------------------------------------------------------------- (unlabeled)

type: numeric (int)

range: [3291,15906] units: 1

unique values: 74 coded missing: 0 / 74

mean: 6165.26

std. dev: 2949.5

percentiles: 10% 25% 50% 75% 90%

3895 4195 5006.5 6342 11385

mpg --------------------------------------------------------------- (unlabeled)

type: numeric (byte)

range: [12,41] units: 1

unique values: 21 coded missing: 0 / 74

mean: 21.2973

std. dev: 5.7855

percentiles: 10% 25% 50% 75% 90%

14 18 20 25 29

rep78 ------------------------------------------------------------- (unlabeled)

type: numeric (byte)

range: [1,5] units: 1

unique values: 5 coded missing: 5 / 74

tabulation: Freq. Value

2 1

8 2

30 3

18 4

11 5

hdroom ------------------------------------------------------------ (unlabeled)

type: numeric (float)

range: [1.5,5] units: .1

unique values: 8 coded missing: 0 / 74

tabulation: Freq. Value

4 1.5

13 2

14 2.5

13 3

15 3.5

10 4

4 4.5

1 5

trunk ------------------------------------------------------------- (unlabeled)

type: numeric (byte)

range: [5,23] units: 1

unique values: 18 coded missing: 0 / 74

mean: 13.7568

std. dev: 4.2774

percentiles: 10% 25% 50% 75% 90%

8 10 14 17 20

weight ------------------------------------------------------------ (unlabeled)

type: numeric (int)

range: [1760,4840] units: 10

unique values: 64 coded missing: 0 / 74

mean: 3019.46

std. dev: 777.194

percentiles: 10% 25% 50% 75% 90%

2020 2240 3190 3600 4060

length ------------------------------------------------------------ (unlabeled)

type: numeric (int)

range: [142,233] units: 1

unique values: 47 coded missing: 0 / 74

mean: 187.932

std. dev: 22.2663

percentiles: 10% 25% 50% 75% 90%

157 170 192.5 204 218

turn -------------------------------------------------------------- (unlabeled)

type: numeric (byte)

range: [31,51] units: 1

unique values: 18 coded missing: 0 / 74

mean: 39.6486

std. dev: 4.39935

percentiles: 10% 25% 50% 75% 90%

34 36 40 43 45

displ ------------------------------------------------------------- (unlabeled)

type: numeric (int)

range: [79,425] units: 1

unique values: 31 coded missing: 0 / 74

mean: 197.297

std. dev: 91.8372

percentiles: 10% 25% 50% 75% 90%

97 119 196 250 350

gratio ------------------------------------------------------------ (unlabeled)

type: numeric (float)

range: [2.19,3.89] units: .01

unique values: 36 coded missing: 0 / 74

mean: 3.01486

std. dev: .456287

percentiles: 10% 25% 50% 75% 90%

2.43 2.73 2.955 3.37 3.72

foreign ----------------------------------------------------------- (unlabeled)

type: numeric (byte)

range: [0,1] units: 1

unique values: 2 coded missing: 0 / 74

tabulation: Freq. Value

52 0

22 1

Another useful command for getting a quick overview of a data file is the **inspect** command. Here is what the **inspect** command produces for the auto data file.

**inspect**

make: Number of Observations

------- Non-

Total Integers Integers

| Negative - - -

| Zero - - -

| Positive - - -

| ----- ----- -----

| Total - - -

| Missing 74

+---------------------- -----

-8.99e+307 74

(0 unique value)

price: Number of Observations

-------- Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 74 74 -

| # ----- ----- -----

| # Total 74 74 -

| # # . . . Missing -

+---------------------- -----

3291 15906 74

(74 unique values)

mpg: Number of Observations

------ Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 74 74 -

| # # ----- ----- -----

| # # # Total 74 74 -

| # # # # . Missing -

+---------------------- -----

12 41 74

(21 unique values)

rep78: Number of Observations

-------- Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 69 69 -

| # # ----- ----- -----

| # # # Total 69 69 -

| . # # # # Missing 5

+---------------------- -----

1 5 74

(5 unique values)

hdroom: Number of Observations

--------- Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 74 37 37

| # # # ----- ----- -----

| # # # # Total 74 37 37

| # # # # # Missing -

+---------------------- -----

1.5 5 74

(8 unique values)

trunk: Number of Observations

-------- Non-

Total Integers Integers

| # Negative - - -

| # # Zero - - -

| # # Positive 74 74 -

| # # # ----- ----- -----

| # # # # # Total 74 74 -

| # # # # # Missing -

+---------------------- -----

5 23 74

(18 unique values)

weight: Number of Observations

--------- Non-

Total Integers Integers

| # # Negative - - -

| # # Zero - - -

| # # # # Positive 74 74 -

| # # # # ----- ----- -----

| # # # # Total 74 74 -

| # # # # # Missing -

+---------------------- -----

1760 4840 74

(64 unique values)

length: Number of Observations

--------- Non-

Total Integers Integers

| # Negative - - -

| # # Zero - - -

| # # Positive 74 74 -

| # # # ----- ----- -----

| # # # # # Total 74 74 -

| # # # # # Missing -

+---------------------- -----

142 233 74

(47 unique values)

turn: Number of Observations

------- Non-

Total Integers Integers

| # Negative - - -

| # # Zero - - -

| # # # Positive 74 74 -

| # # # ----- ----- -----

| # # # # Total 74 74 -

| # # # # . Missing -

+---------------------- -----

31 51 74

(18 unique values)

displ: Number of Observations

-------- Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 74 74 -

| # # ----- ----- -----

| # # # # Total 74 74 -

| # # # # . Missing -

+---------------------- -----

79 425 74

(31 unique values)

gratio: Number of Observations

--------- Non-

Total Integers Integers

| # Negative - - -

| # Zero - - -

| # Positive 74 - 74

| # # # # ----- ----- -----

| # # # # Total 74 - 74

| # # # # # Missing -

+---------------------- -----

2.19 3.89 74

(36 unique values)

foreign: Number of Observations

---------- Non-

Total Integers Integers

| # Negative - - -

| # Zero 52 52 -

| # Positive 22 22 -

| # ----- ----- -----

| # # Total 74 74 -

| # # Missing -

+---------------------- -----

0 1 74

(2 unique values)

The **list** command is useful for viewing all or a range of observations. Here we look at **make, price, mpg, rep78 and** **foreign** for the first 10 observations.

list **make price mpg rep78 foreign in 1/10**

make price mpg rep78 foreign

1. Dodge Magnum 5886 16 2 0

2. Datsun 510 5079 24 4 1

3. Ford Mustang 4187 21 3 0

4. Linc. Versailles 13466 14 3 0

5. Plym. Sapporo 6486 26 . 0

6. Plym. Arrow 4647 28 3 0

7. Cad. Eldorado 14500 14 2 0

8. AMC Spirit 3799 22 . 0

9. Pont. Catalina 5798 18 4 0

10. Chev. Nova 3955 19 3 0

**Creating tables**

The **tabulate** command is useful for obtaining frequency tables. Below, we make a table for **rep78** and a table for **foreign**. The command can also be shortened to **tab.**

**tabulate rep78**

rep78 | 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**

foreign | Freq. Percent Cum.

------------+-----------------------------------

0 | 52 70.27 70.27

1 | 22 29.73 100.00

------------+-----------------------------------

Total | 74 100.00

The **tab1** command can be used as a shortcut to request tables for a series of variables (instead of typing the **tabulate** command over and over again for each variable of interest).

**tab1 rep78 foreign**

-> tabulation of rep78

rep78 | 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

-> tabulation of foreign

foreign | Freq. Percent Cum.

------------+-----------------------------------

0 | 52 70.27 70.27

1 | 22 29.73 100.00

------------+-----------------------------------

Total | 74 100.00

We can use the **plot** option to make a plot to visually show the tabulated values.

**tabulate rep78, plot**

rep78 | Freq.

------------+------------+-----------------------------------------------------

1 | 2 |\*\*

2 | 8 |\*\*\*\*\*\*\*\*

3 | 30 |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 | 18 |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5 | 11 |\*\*\*\*\*\*\*\*\*\*\*

------------+------------+-----------------------------------------------------

Total | 69

We can also make crosstabs using **tabulate**. Let's look at the repair history broken down by foreign and domestic cars.

**tabulate rep78 foreign**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 2 0 | 2

2 | 8 0 | 8

3 | 27 3 | 30

4 | 9 9 | 18

5 | 2 9 | 11

-----------+----------------------+----------

Total | 48 21 | 69

With the **column** option, we can request column percentages. Notice that about 86% of the foreign cars received a rating of 4 or 5. Only about 23% of domestic cars were rated that highly.

**tabulate rep78 foreign, column**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 2 0 | 2

| 4.17 0.00 | 2.90

-----------+----------------------+----------

2 | 8 0 | 8

| 16.67 0.00 | 11.59

-----------+----------------------+----------

3 | 27 3 | 30

| 56.25 14.29 | 43.48

-----------+----------------------+----------

4 | 9 9 | 18

| 18.75 42.86 | 26.09

-----------+----------------------+----------

5 | 2 9 | 11

| 4.17 42.86 | 15.94

-----------+----------------------+----------

Total | 48 21 | 69

| 100.00 100.00 | 100.00

We can use the **nofreq** option to suppress the frequencies, and just focus on the percentages.

**tabulate rep78 foreign, column nofreq**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 4.17 0.00 | 2.90

2 | 16.67 0.00 | 11.59

3 | 56.25 14.29 | 43.48

4 | 18.75 42.86 | 26.09

5 | 4.17 42.86 | 15.94

-----------+----------------------+----------

Total | 100.00 100.00 | 100.00

Note that the order of the options does not matter. Just remember that the options must come after the comma.

**tabulate rep78 foreign, nofreq column**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 4.17 0.00 | 2.90

2 | 16.67 0.00 | 11.59

3 | 56.25 14.29 | 43.48

4 | 18.75 42.86 | 26.09

5 | 4.17 42.86 | 15.94

-----------+----------------------+----------

Total | 100.00 100.00 | 100.00

**Generating summary statistics with summarize**

For summary statistics, we can use the **summarize** command. Let's generate some summary statistics on **mpg**.

**summarize mpg**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 74 21.2973 5.785503 12 41

We can use the **detail** option of the **summarize** command to get more detailed summary statistics.

**summarize mpg, detail**

mpg

-------------------------------------------------------------

Percentiles Smallest

1% 12 12

5% 14 12

10% 14 14 Obs 74

25% 18 14 Sum of Wgt. 74

50% 20 Mean 21.2973

Largest Std. Dev. 5.785503

75% 25 34

90% 29 35 Variance 33.47205

95% 34 35 Skewness .9487176

99% 41 41 Kurtosis 3.975005

To get these values separately for foreign and domestic, we could use the **by foreign:** prefix as shown below. Note that we first had to sort the data before using **by foreign:**.

**sort foreign**

**by foreign: summarize mpg**

-> foreign= 0

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 52 19.82692 4.743297 12 34

-> foreign= 1

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 22 24.77273 6.611187 14 41

This is not the most efficient way to do this. Another way, which does not require the data to be sorted, is by using the **summarize( )** option as part of the **tabulate** command.

**tabulate foreign, summarize(mpg)**

| Summary of mpg

foreign | Mean Std. Dev. Freq.

------------+------------------------------------

0 | 19.826923 4.7432972 52

1 | 24.772727 6.6111869 22

------------+------------------------------------

Total | 21.297297 5.7855032 74

Here is another example, showing the average price of cars for each level of repair history

**tabulate rep78, summarize(price)**

| Summary of price

rep78 | Mean Std. Dev. Freq.

------------+------------------------------------

1 | 4564.5 522.55191 2

2 | 5967.625 3579.3568 8

3 | 6429.2333 3525.1398 30

4 | 6071.5 1709.6083 18

5 | 5913 2615.7628 11

------------+------------------------------------

Total | 6146.0435 2912.4403 69

**Summary**

Provide information about the current data file, including the number of variables and observations and a listing of the variables in a data file.

**describe**

Produce codebook like information for the current data file.

**codebook**

Provide a quick overview of data file.

**inspect**

List out the variables **make** and **mpg**.

**list model mpg**

Make a table of **mpg**.

**tabulate mpg**

Make a two way table of **rep78** by **foreign**.

**tabulate rep78 foreign**

Produce summary statistics of **mpg** and **price**.

**summarize mpg price**

Produce summary statistics for **mpg** separately for foreign and domestic cars.

**sort foreign**

**by foreign: summarize(mpg)**

Produce summary statistics for **mpg** by **foreign** (prior sorting not required).

**tabulate foreign, summarize(mpg)**

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

**IDRE Research Technology Group**

[**High Performance Computing**](https://idre.ucla.edu/hpc)

[**Statistical Computing**](https://idre.ucla.edu/stats)

[**GIS and Visualization**](https://idre.ucla.edu/gis-visualization)

* [High Performance Computing](https://idre.ucla.edu/hpc)
* [GIS](https://idre.ucla.edu/gis)
* [Statistical Computing](https://idre.ucla.edu/stats)
* [Hoffman2 Cluster](https://idre.ucla.edu/hoffman2)
* [Mapshare](https://idre.ucla.edu/mapshare)
* [Classes](https://idre.ucla.edu/events/classes)
* [Hoffman2 Account Application](https://idre.ucla.edu/hoffman2/getting-started)
* [Visualization](https://idre.ucla.edu/visualization)
* [Conferences](https://idre.ucla.edu/events/conferences)
* [Hoffman2 Usage Statistics](https://idre.ucla.edu/hoffman2/cluster-statistics)
* [3D Modeling](https://idre.ucla.edu/3d-modeling)
* [Reading Materials](https://idre.ucla.edu/reading-materials)
* [UC Grid Portal](http://portal.ucgrid.org/)
* [Technology Sandbox](https://idre.ucla.edu/technology-sandbox)
* [IDRE Listserv](http://lists.ucla.edu/cgi-bin/mailman/listinfo/idremailinglist)
* [UCLA Grid Portal](http://grid.ucla.edu/)
* [Tech Sandbox Access](https://idre.ucla.edu/gis-visualization/sandbox-access)
* [IDRE Resources](https://idre.ucla.edu/resources)
* [Shared Cluster & Storage](https://idre.ucla.edu/hpc/shared-cluster-program)
* [Data Centers](https://idre.ucla.edu/data-centers)
* [Social Sciences Data Archive](http://dataarchives.ss.ucla.edu/)
* [About IDRE](https://idre.ucla.edu/about)
* [About](https://idre.ucla.edu/about)
* [Contact](https://idre.ucla.edu/contact)
* [News](https://idre.ucla.edu/news)
* [Events](https://idre.ucla.edu/events)
* [Our Experts](https://idre.ucla.edu/experts)
* © 2013 UC Regents
* [Terms of Use & Privacy Policy](http://www.ucla.edu/terms-of-use/)

Citing: <http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm>

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
Getting help using Stata**

This module shows resources you can use to help you learn and use Stata.

**Stata online help**

When you know the name of the command you want to use (e.g., summarize), you can use the Stata help to get a quick summary of the command and its syntax. You can do this in two ways:  
1. type **help summarize** in the command window, or  
2. click **Help**, **Stata Command**, then type **summarize**.

Here is what **help summarize** looks like.

**help summarize**

help summarize dialog: summarize

---------------------------------------------------------------------

Title

[R] summarize -- Summary statistics

Syntax

summarize [varlist] [if] [in] [weight] [, options]

options description

---------------------------------------------------------------

Main

detail display additional statistics

meanonly suppress the display; only calculate the

mean; programmer's option

format use variable's display format

separator(#) draw separator line after every # variables;

default is separator(5)

---------------------------------------------------------------

varlist may contain time-series operators; see tsvarlist.

by may be used with summarize; see by.

aweights, fweights, and iweights are allowed. However,

iweights may not be used with the detail option; see weight.

Description

summarize calculates and displays a variety of univariate

summary statistics. If no varlist is specified, summary

statistics are calculated for all the variables in the dataset.

Also see ci for calculating the standard error and confidence

intervals of the mean.

Options

+------+

----+ Main +---------------------------------------------------

detail produces additional statistics including skewness,

kurtosis, the four smallest and four largest values, and

various percentiles.

meanonly, which is allowed only when detail is not specified,

suppresses the display of results and calculation of the

variance. Ado-file writers will find this useful for fast

calls.

format requests that the summary statistics be displayed using

the display formats associated with the variables, rather

than the default g display format; see format.

separator(#) specifies how often to insert separation lines

into the output. The default is separator(5), meaning that

a line is drawn after every 5 variables. separator(10)

would draw a line after every 10 variables. separator(0)

suppresses the separation line.

Examples

summarize

summarize mpg weight

summarize mpg weight if foreign

summarize mpg weight if foreign, detail

Also see

Manual: [R] summarize

Online: ameans, centile, cf, ci, codebook, compare, describe,

egen, inspect, lv, mean, pctile, stsum, svy: mean,

table, tabstat, tabulate summarize, xtsum

If you use the pull-down menu to get help for a command, it shows the same basic information but related commands and topics are hotlinks you can click.

When you want to search for a keyword, e.g. **memory**, you can use Stata to search for help topics that contain that keyword. You can do this in two ways:  
1. Type **search memory** in the command window, or  
2. Click **Help**, **Search**, then **memory**.

Here is what search memory looks like.

**search memory**

GS . . . . . . . . . . . . . . . . . . . . . . . . Getting Started manual

[U] Chapter 7 . . . . . . . . . . . . . . . . Setting the size of memory

(help memory)

[R] compress . . . . . . . . . . . . . . . . . . . Compress data in memory

(help compress)

[R] describe . . . . . . . Describe contents of data in memory or on disk

(help describe)

[R] discard . . . . . . . . . . . . . Drop automatically loaded programs

(help discard)

[R] drop . . . . . . . . . . . . . . . Eliminate variables or observations

(help drop)

[R] encode . . . . . . . . . . . Encode string into numeric and vice versa

(help encode)

[R] matsize . . . . . . . Set the maximum number of variables in a model

(help matsize)

[R] memory . . . . . . . . . . . . . . . . . . Memory size considerations

(help memory)

[R] query . . . . . . . . . . . . . . . . . . . Display system parameters

(help query)

[R] save . . . . . . . . . . . . . . . . . . . . . . Save and use datasets

(help save)

[R] set . . . . . . . . . . . . . . Quick reference for system parameters

(help set)

FAQ . . . . . . . . . . . . . . . Using a dataset that won't fit into RAM

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . A. Riley

1/96 How can I use a dataset that is larger than the available RAM?

http://www.stata.com/support/faqs/data/large.html

FAQ . . . . . . . . . . . . . . . . . . . . . . . . . . Memory allocation

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . C. Nguyen

6/97 I'm not able to access all available free memory from

Windows 3.1.

http://www.stata.com/support/faqs/win/memory.html#nomem

FAQ . . . . . . . . . . . . . . . . . . . . . . . . . . Memory allocation

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . C. Nguyen

6/97 How do I set the amount of memory allocated to Stata under

Windows 3.1 and 95?

http://www.stata.com/support/faqs/win/memory.html#win95prop

FAQ . . . . . . . . . . . . . . . . . . . Miscellaneous Windows questions

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . C. Nguyen

8/97 Why is Stata running very slowly?

http://www.stata.com/support/faqs/win/misc.html#slow

FAQ . . . . . . . . . . . . . . . . . . . . . . Windows memory management

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . A. Riley

2/98 Why does Windows 95 seem to be swapping even though I

haven't allocated all available memory to Stata?

http://www.stata.com/support/faqs/win/vcache.html#swap

FAQ . . . . . . . . . . . . . . . . . . . . . Macintosh memory allocation

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . C. Nguyen

6/97 I have my memory doubled/tripled by Ram Doubler but Stata

is not recognizing all of it.

http://www.stata.com/support/faqs/mac/memory.html#ramdoubler

FAQ . . . . . . . . . . . . . . . . . . . . . Macintosh memory allocation

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . C. Nguyen

6/97 How do I increase memory allocated to Stata?

http://www.stata.com/support/faqs/mac/memory.html#incmem

STB-40 ip20 . . . . . . . . . Checking for sufficient memory to add variables

(help memchk if installed) . . . . . . . . . . . . . . . . P. Sasieni

11/97 STB Reprints Vol 7, page 86

beginning-of-program check on whether there is sufficient

memory to create temporary variables

As you can see, there are lots of help topics that refer to memory. Some of the topics give you a command, and then you can get help for that command. Notice that those topics start with **GS** **[U]** or **[R]**. Those are indicating which Stata manual you could find the command (GS=Getting Started, U=Users Guide, R=Reference Guide).

The next set of topics all start with **FAQ** because these are Frequently Asked Questions from the Stata web site. You can see the title of the FAQ and the address of the FAQ. Lastly, there is a topic that starts with **STB** which stands for Stata Technical Bulletin. These refer to add-on programs that you can install into Stata. There are dozens, if not hundreds of specialized and useful programs that you can get from the Stata Technical Bulletin.

You can access this same kind of help from the pull-down menus by clicking **Help** then **Search** then type **memory**. Note how the related commands, the FAQs, and the STB all have hotlinks you can click. For example, you can click on a FAQ and it will bring up that FAQ in your web browser. Or, you could click on an STB and it would walk you through the steps of installing that STB into your copy of Stata. As you can see, there are real advantages to using the pull-down menus for getting help because it is so easy to click on the related topics.

**Stata sample data files**

Stata has some very useful data files available to you for learning and practicing Stata. For example, you can type

**sysuse auto**

to use the **auto** data file that comes with Stata. You can type

**sysuse dir**

to see the entire list of data files that ship with Stata. You can type

**help dta\_contents**

to see all of the sample data files that you can easily access from within Stata.

**Stata web pages**

The Stata web page is a wonderful resource. You can visit the main page at <http://www.stata.com> .

The **User Support** page (click User Support from main page) has a great set of resources, including  
The user support area contains:  
- FAQs  
- NetCourses  
- StataList: How to subscribe  
- StataList: Archives  
- Statalist ado-file Archives  
- Stata Bookstore  
In the bookstore, you can find books on Stata. A good intro book on Stata is **Statistics with Stata**.

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

**IDRE Research Technology Group**

[**High Performance Computing**](https://idre.ucla.edu/hpc)

[**Statistical Computing**](https://idre.ucla.edu/stats)

[**GIS and Visualization**](https://idre.ucla.edu/gis-visualization)

Citing: <http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm>, 28th/march/2013

**Welcome to the Institute for Digital Reseach and Education**

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

Stata Learning Module  
Using IF with Stata commands

This module shows the use of **if** with common Stata commands.

Let's use the **auto** data file.

**sysuse auto**

For this module, we will focus on the variables **make, rep78, foreign, mpg,** and **price.** We can use the **keep** command to keep just these five variables.

**keep make rep78 foreign mpg price**

Let's make a table of **rep78** by **foreign** to look at the repair histories of the foreign and domestic cars.

**tabulate rep78 foreign**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 2 0 | 2

2 | 8 0 | 8

3 | 27 3 | 30

4 | 9 9 | 18

5 | 2 9 | 11

-----------+----------------------+----------

Total | 48 21 | 69

Suppose we wanted to focus on just the cars with repair histories of four or better. We can use **if** suffix to do this.

**tabulate rep78 foreign if rep78 >=4**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

4 | 9 9 | 18

5 | 2 9 | 11

-----------+----------------------+----------

Total | 11 18 | 29

Let's make the above table using the **column** and **nofreq** options. The command **column** requests column percentages while the command **nofreq** suppresses cell frequencies. Note that **column** and **nofreq** come after the comma. These are options on the **tabulate** command and options need to be placed after a comma.

**tabulate rep78 foreign if rep78 >=4, column nofreq**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

4 | 81.82 50.00 | 62.07

5 | 18.18 50.00 | 37.93

-----------+----------------------+----------

Total | 100.00 100.00 | 100.00

The use of **if** is not limited to the tabulate command. Here, we use it with the **list** command.

**list if rep78 >= 4**

make price mpg rep78 foreign

3. AMC Spirit 3799 22 . 0

5. Buick Electra 7827 15 4 0

7. Buick Opel 4453 26 . 0

15. Chev. Impala 5705 16 4 0

20. Dodge Colt 3984 30 5 0

24. Ford Fiesta 4389 28 4 0

29. Merc. Bobcat 3829 22 4 0

30. Merc. Cougar 5379 14 4 0

33. Merc. XR-7 6303 14 4 0

35. Olds 98 8814 21 4 0

38. Olds Delta 88 4890 18 4 0

43. Plym. Champ 4425 34 5 0

45. Plym. Sapporo 6486 26 . 0

47. Pont. Catalina 5798 18 4 0

51. Pont. Phoenix 4424 19 . 0

53. Audi 5000 9690 17 5 1

55. BMW 320i 9735 25 4 1

56. Datsun 200 6229 23 4 1

57. Datsun 210 4589 35 5 1

58. Datsun 510 5079 24 4 1

59. Datsun 810 8129 21 4 1

61. Honda Accord 5799 25 5 1

62. Honda Civic 4499 28 4 1

63. Mazda GLC 3995 30 4 1

64. Peugeot 604 12990 14 . 1

66. Subaru 3798 35 5 1

67. Toyota Celica 5899 18 5 1

68. Toyota Corolla 3748 31 5 1

69. Toyota Corona 5719 18 5 1

70. VW Dasher 7140 23 4 1

71. VW Diesel 5397 41 5 1

72. VW Rabbit 4697 25 4 1

73. VW Scirocco 6850 25 4 1

74. Volvo 260 11995 17 5 1

Did you see that some of the observations had a value of '**.**' for **rep78**? These are missing values. For example, the value of **rep78** for the AMC Spirit is missing. Stata treats a missing value as positive infinity, the highest number possible. So, when we said **list if rep78 >= 4,** Stata included the observations where **rep78** was '**.**' as well.

If we wanted to include just the valid (non-missing) observations that are greater than or equal to 4, we can do the following to tell Stata we want only observations where **rep78 >= 4** and **rep78 is not missing**.

**list if rep78 >= 4 & !missing(rep78)**

make price mpg rep78 foreign

5. Buick Electra 7827 15 4 0

15. Chev. Impala 5705 16 4 0

20. Dodge Colt 3984 30 5 0

24. Ford Fiesta 4389 28 4 0

29. Merc. Bobcat 3829 22 4 0

30. Merc. Cougar 5379 14 4 0

33. Merc. XR-7 6303 14 4 0

35. Olds 98 8814 21 4 0

38. Olds Delta 88 4890 18 4 0

43. Plym. Champ 4425 34 5 0

47. Pont. Catalina 5798 18 4 0

53. Audi 5000 9690 17 5 1

55. BMW 320i 9735 25 4 1

56. Datsun 200 6229 23 4 1

57. Datsun 210 4589 35 5 1

58. Datsun 510 5079 24 4 1

59. Datsun 810 8129 21 4 1

61. Honda Accord 5799 25 5 1

62. Honda Civic 4499 28 4 1

63. Mazda GLC 3995 30 4 1

66. Subaru 3798 35 5 1

67. Toyota Celica 5899 18 5 1

68. Toyota Corolla 3748 31 5 1

69. Toyota Corona 5719 18 5 1

70. VW Dasher 7140 23 4 1

71. VW Diesel 5397 41 5 1

72. VW Rabbit 4697 25 4 1

73. VW Scirocco 6850 25 4 1

74. Volvo 260 11995 17 5 1

This code will also yield the same output as above.

**list if rep78 >= 4 & rep78 != .**

We can use **if** with most Stata commands. Here, we get summary statistics for **price** for cars with repair histories of 1 or 2. Note the double equal (==) represents IS EQUAL TO and the pipe ( | ) represents OR.

**summarize price if rep78 == 1 | rep78 == 2**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

price | 10 5687 3216.375 3667 14500

A simpler way to say this would be...

**summarize price if rep78 <= 2**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

price | 10 5687 3216.375 3667 14500

Likewise, we can do this for cars with repair history of 3, 4 or 5.

**summarize price if rep78 == 3 | rep78 == 4 | rep78 == 5**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

price | 59 6223.847 2880.454 3291 15906

Additionally, we can use this code to designate a range of values. Here is a summary of price for the values 3 through 5 in **rep78**.

**summarize price if inrange(rep78,3,5)**

Variable | Obs Mean Std. Dev. Min Max

----------+--------------------------------------------------------

price | 59 6223.847 2880.454 3291 15906

Let's simplify this by saying rep78 >= 3.

**summarize price if rep78 >= 3**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

price | 64 6239.984 2925.843 3291 15906

Did you see the mistake we made? We accidentally included the missing values because we forgot to exclude them. We really needed to say.

**summarize price if rep78 >= 3 & !missing(rep78)**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

price | 59 6223.847 2880.454 3291 15906

Taking a random sample

It is also possible to take a simple random sample of your datausing the **sample** command. This information can be found on our STATA FAQ page: [How can I draw a random sample of my data?](http://www.ats.ucla.edu/stat/stata/faq/sample.htm)

Summary

Most Stata commands can be followed by **if**, for example  
  
Summarize if rep78 equals 2

**summarize if rep78 == 2**

Summarize if rep78 is greater than or equal to 2

**summarize if rep78 >= 2**

Summarize if rep78 greater than 2

**summarize if rep78 > 2**

Summarize if rep78 less than or equal to 2

**summarize if rep78 <= 2**

Summarize if rep78 less than 2

**summarize if rep78 <2**

Summarize if rep78 not equal to 2

**summarize if rep78 != 2**

**If expressions can be connected with**

| for OR  
& for AND

**Missing Values**

Missing values are represented as '**.**' and are the highest value possible. Therefore, when values are missing, be careful with commands like

**summarize if rep78 > 3**

**summarize if rep78 >= 3**

**summarize if rep78 != 3**

to omit missing values, use

**summarize if rep78 > 3 & !missing(rep78)**

**summarize if rep78 >= 3 & !missing(rep78)**

**summarize if rep78 != 3 & !missing(rep78)**

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

IDRE Research Technology Gr

|  |  |
| --- | --- |
|  |  |

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
A statistical sampler in Stata**

**Version info:** Code for this page was tested in Stata 12.

This module will give a brief overview of some common statistical tests in Stata. Let's use the **auto** data file that we will use for our examples.

**sysuse auto**

**t-tests**

Let's do a t-test comparing the miles per gallon (**mpg**) of foreign and domestic cars.

**ttest mpg , by(foreign)**

Two-sample t test with equal variances

------------------------------------------------------------------------------

Group | Obs Mean Std. Err. Std. Dev. [95% Conf. Interval]

---------+--------------------------------------------------------------------

0 | 52 19.82692 .657777 4.743297 18.50638 21.14747

1 | 22 24.77273 1.40951 6.611187 21.84149 27.70396

---------+--------------------------------------------------------------------

combined | 74 21.2973 .6725511 5.785503 19.9569 22.63769

---------+--------------------------------------------------------------------

diff | -4.945804 1.362162 -7.661225 -2.230384

------------------------------------------------------------------------------

Degrees of freedom: 72

Ho: mean(0) - mean(1) = diff = 0

Ha: diff <0 Ha: diff ~="0" Ha: diff> 0

t = -3.6308 t = -3.6308 t = -3.6308

P < t = 0.0003 P > |t| = 0.0005 P > t = 0.9997

As you see in the output above, the domestic cars had significantly lower **mpg** (19.8) than the foreign cars (24.7).

**Chi-square**

Let's compare the repair rating (**rep78**) of the foreign and domestic cars. We can make a crosstab of **rep78** by **foreign**. We may want to ask whether these variables are independent. We can use the **chi2** option to request a chi-square test of independence as well as the crosstab.

**tabulate rep78 foreign, chi2**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 2 0 | 2

2 | 8 0 | 8

3 | 27 3 | 30

4 | 9 9 | 18

5 | 2 9 | 11

-----------+----------------------+----------

Total | 48 21 | 69

Pearson chi2(4) = 27.2640 Pr = 0.000

The chi-square is not really valid when you have empty cells. In such cases when you have empty cells, or cells with small frequencies, you can request Fisher's exact test with the **exact** option.

**tabulate rep78 foreign, chi2 exact**

| foreign

rep78 | 0 1 | Total

-----------+----------------------+----------

1 | 2 0 | 2

2 | 8 0 | 8

3 | 27 3 | 30

4 | 9 9 | 18

5 | 2 9 | 11

-----------+----------------------+----------

Total | 48 21 | 69

Pearson chi2(4) = 27.2640 Pr = 0.000

Fisher's exact = 0.000

**Correlation**

We can use the **correlate** command to get the correlations among variables. Let's look at the correlations among **price** **mpg** **weight** and **rep78**. (We use **rep78** in the correlation even though it is not continuous to illustrate what happens when you use correlate with variables with missing data.)

**correlate price mpg weight rep78**

(obs=69)

| price mpg weight rep78

---------+------------------------------------

price | 1.0000

mpg | -0.4559 1.0000

weight | 0.5478 -0.8055 1.0000

rep78 | 0.0066 0.4023 -0.4003 1.0000

Note that the output above said (obs=69). The **correlate** command drops data on a **listwise** basis, meaning that if any of the variables are missing, then the entire observation is omitted from the correlation analysis.

We can use **pwcorr** (pairwise correlations) if we want to obtain correlations that deletes missing data on a **pairwise** basis instead of a listwise basis. We will use the **obs** option to show the number of observations used for calculating each correlation.

**pwcorr price mpg weight rep78, obs**

| price mpg weight rep78

----------+------------------------------------

price | 1.0000

| 74

|

mpg | -0.4686 1.0000

| 74 74

|

weight | 0.5386 -0.8072 1.0000

| 74 74 74

|

rep78 | 0.0066 0.4023 -0.4003 1.0000

| 69 69 69 69

|

Note how the correlations that involve **rep78** have an N of 69 compared to the other correlations that have an N of 74. This is because **rep78** has five missing values, so it only had 69 valid observations, but the other variables had no missing data so they had 74 valid observations.

**Regression**

Let's look at doing regression analysis in Stata. For this example, let's drop the cases where **rep78** is 1 or 2 or missing.

**drop if (rep78 <= 2) | (rep78==.)**

(15 observations deleted)

Now, let's predict **mpg** from **price** and **weight**. As you see below, **weight** is a significant predictor of **mpg**, but **price** is not.

**regress mpg price weight**

Source | SS df MS Number of obs = 59

---------+------------------------------ F( 2, 56) = 47.87

Model | 1375.62097 2 687.810483 Prob > F = 0.0000

Residual | 804.616322 56 14.3681486 R-squared = 0.6310

---------+------------------------------ Adj R-squared = 0.6178

Total | 2180.23729 58 37.5902981 Root MSE = 3.7905

------------------------------------------------------------------------------

mpg | Coef. Std. Err. t P>|t| [95% Conf. Interval]

---------+--------------------------------------------------------------------

price | -.0000139 .0002108 -0.066 0.948 -.0004362 .0004084

weight | -.005828 .0007301 -7.982 0.000 -.0072906 -.0043654

\_cons | 39.08279 1.855011 21.069 0.000 35.36676 42.79882

------------------------------------------------------------------------------

What if we wanted to predict **mpg** from **rep78** as well. **rep78** is really more of a categorical variable than it is a continuous variable. To include it in the regression, we should convert **rep78** into dummy variables. Fortunately, Stata makes dummy variables easily using tabulate. The **gen(rep)** option tells Stata that we want to generate dummy variables from **rep78** and we want the stem of the dummy variables to be **rep**.

**tabulate rep78, gen(rep)**

rep78 | Freq. Percent Cum.

------------+-----------------------------------

3 | 30 50.85 50.85

4 | 18 30.51 81.36

5 | 11 18.64 100.00

------------+-----------------------------------

Total | 59 100.00

Stata has created **rep1** (1 if **rep78** is 3), **rep2** (1 if **rep78** is 4) and **rep3** (1 if **rep78** is 5). We can use the **tabulate** command to verify that the dummy variables were created properly.

**tabulate rep78 rep1**

| rep78== 3.0000

rep78 | 0 1 | Total

-----------+----------------------+----------

3 | 0 30 | 30

4 | 18 0 | 18

5 | 11 0 | 11

-----------+----------------------+----------

Total | 29 30 | 59

**tabulate rep78 rep2**

| rep78== 4.0000

rep78 | 0 1 | Total

-----------+----------------------+----------

3 | 30 0 | 30

4 | 0 18 | 18

5 | 11 0 | 11

-----------+----------------------+----------

Total | 41 18 | 59

**tabulate rep78 rep3**

| rep78== 5.0000

rep78 | 0 1 | Total

-----------+----------------------+----------

3 | 30 0 | 30

4 | 18 0 | 18

5 | 0 11 | 11

-----------+----------------------+----------

Total | 48 11 | 59

Now we can include **rep1** and **rep2** as dummy variables in the regression model.

**regress mpg price weight rep1 rep2**

Source | SS df MS Number of obs = 59

-------------+------------------------------ F( 4, 54) = 26.04

Model | 1435.91975 4 358.979938 Prob > F = 0.0000

Residual | 744.317536 54 13.7836581 R-squared = 0.6586

-------------+------------------------------ Adj R-squared = 0.6333

Total | 2180.23729 58 37.5902981 Root MSE = 3.7126

------------------------------------------------------------------------------

mpg | Coef. Std. Err. t P>|t| [95% Conf. Interval]

-------------+----------------------------------------------------------------

price | -.0001126 .0002133 -0.53 0.600 -.0005403 .0003151

weight | -.005107 .0008236 -6.20 0.000 -.0067584 -.0034557

rep1 | -2.886288 1.504639 -1.92 0.060 -5.902908 .1303314

rep2 | -2.88417 1.484817 -1.94 0.057 -5.861048 .0927086

\_cons | 39.89189 1.892188 21.08 0.000 36.09828 43.6855

------------------------------------------------------------------------------

**Analysis of variance**

If you wanted to do an analysis of variance looking at the differences in **mpg** among the three repair groups, you can use the **oneway** command to do this.

**oneway mpg rep78**

Analysis of Variance

Source SS df MS F Prob > F

------------------------------------------------------------------------

Between groups 506.325167 2 253.162583 8.47 0.0006

Within groups 1673.91212 56 29.8912879

------------------------------------------------------------------------

Total 2180.23729 58 37.5902981

Bartlett's test for equal variances: chi2(2) = 9.9384 Prob>chi2 = 0.007

If you include the tabulate option, you get mean **mpg** for the three groups, which shows that the group with the best repair rating (**rep78** of 5) also has the highest **mpg** (27.3).

**oneway mpg rep78, tabulate**

| Summary of mpg

rep78 | Mean Std. Dev. Freq.

------------+------------------------------------

3 | 19.433333 4.1413252 30

4 | 21.666667 4.9348699 18

5 | 27.363636 8.7323849 11

------------+------------------------------------

Total | 21.59322 6.1310927 59

Analysis of Variance

Source SS df MS F Prob > F

------------------------------------------------------------------------

Between groups 506.325167 2 253.162583 8.47 0.0006

Within groups 1673.91212 56 29.8912879

------------------------------------------------------------------------

Total 2180.23729 58 37.5902981

Bartlett's test for equal variances: chi2(2) = 9.9384 Prob>chi2 = 0.007

If you want to include covariates, you need to use the **anova** command. The **continuous(price weight)** option tells Stata that those variables are covariates.

**anova mpg rep78 c.price c.weight**

Number of obs = 59 R-squared = 0.6586

Root MSE = 3.71263 Adj R-squared = 0.6333

Source | Partial SS df MS F Prob > F

-----------+----------------------------------------------------

Model | 1435.91975 4 358.979938 26.04 0.0000

|

rep78 | 60.2987853 2 30.1493926 2.19 0.1221

price | 3.8421233 1 3.8421233 0.28 0.5997

weight | 529.932889 1 529.932889 38.45 0.0000

|

Residual | 744.317536 54 13.7836581

-----------+----------------------------------------------------

Total | 2180.23729 58 37.5902981

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

**Welcome to the Institute for Digital Reseach and Education**

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
An overview of Stata syntax**

This module shows the general structure of Stata commands. We will demonstrate this using **summarize** as an example, although this general structure applies to most Stata commands.

Note: This code was tested in Stata 12

Let's first use the **auto** data file.

**use auto**

As you have seen, we can type **summarize** and it will give us summary statistics for all of the variables in the data file.

**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

hdroom | 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

displ | 74 197.2973 91.83722 79 425

gratio | 74 3.014865 .4562871 2.19 3.89

foreign | 74 .2972973 .4601885 0 1

It is also possible to obtain means for specific variables. For example, below we get summary statistics just for **mpg** and **price**.

**summarize mpg price**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 74 21.2973 5.785503 12 41

price | 74 6165.257 2949.496 3291 15906

We could further tell Stata to limit the summary statistics to just foreign cars by adding an **if** clause.

**summarize mpg price if (foreign == 1)**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 22 24.77273 6.611187 14 41

price | 22 6384.682 2621.915 3748 12990

The **if** clause can contain more than one condition. Here, we ask for summary statistics for the foreign cars which get less than 30 miles per gallon.

**summarize mpg price if foreign == 1 & mpg <30**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

mpg | 17 21.94118 3.896643 14 28

price | 17 6996.235 2674.552 3895 12990

We can use the **detail** option to ask Stata to give us more detail in the summary statistics. Notice that the **detail** option goes after the comma. If the comma were omitted, Stata would give an error.

**summarize mpg price if foreign == 1 & mpg <30 , detail**

mpg

-------------------------------------------------------------

Percentiles Smallest

1% 14 14

5% 14 17

10% 17 17 Obs 17

25% 18 18 Sum of Wgt. 17

50% 23 Mean 21.94118

Largest Std. Dev. 3.896643

75% 25 25

90% 26 25 Variance 15.18382

95% 28 26 Skewness -.4901235

99% 28 28 Kurtosis 2.201759

price

-------------------------------------------------------------

Percentiles Smallest

1% 3895 3895

5% 3895 4296

10% 4296 4499 Obs 17

25% 5079 4697 Sum of Wgt. 17

50% 6229 Mean 6996.235

Largest Std. Dev. 2674.552

75% 8129 9690

90% 11995 9735 Variance 7153229

95% 12990 11995 Skewness .9818272

99% 12990 12990 Kurtosis 2.930843

Note that even though we built these parts up one at a time, they don't have to go together. Let's look at some other forms of the **summarize** command.

You can tell Stata which observation numbers you want using the **in** clause. Here we ask for summaries of observations 1 to 10. This is useful if you have a big data file and want to try out a command on a subset of observations.

**summarize in 1/10**

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

make | 0

price | 10 5517.4 2063.518 3799 10372

mpg | 10 19.5 3.27448 15 26

rep78 | 8 3.125 .3535534 3 4

hdroom | 10 3.3 .7527727 2 4.5

trunk | 10 14.7 3.88873 10 21

weight | 10 3271 558.3796 2230 4080

length | 10 194 19.32759 168 222

turn | 10 40.2 3.259175 34 43

displ | 10 223.9 71.77503 121 350

gratio | 10 2.907 .3225264 2.41 3.58

foreign | 10 0 0 0 0

Also, recall that you can ask Stata to perform summaries for foreign and domestic cars separately using **by**, as shown below.

**sort foreign**

**by foreign: summarize**

-> foreign= 0

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

make | 0

price | 52 6072.423 3097.104 3291 15906

mpg | 52 19.82692 4.743297 12 34

rep78 | 48 3.020833 .837666 1 5

hdroom | 52 3.153846 .9157578 1.5 5

trunk | 52 14.75 4.306288 7 23

weight | 52 3317.115 695.3637 1800 4840

length | 52 196.1346 20.04605 147 233

turn | 52 41.44231 3.967582 31 51

displ | 52 233.7115 85.26299 86 425

gratio | 52 2.806538 .3359556 2.19 3.58

foreign | 52 0 0 0 0

-> foreign= 1

Variable | Obs Mean Std. Dev. Min Max

---------+-----------------------------------------------------

make | 0

price | 22 6384.682 2621.915 3748 12990

mpg | 22 24.77273 6.611187 14 41

rep78 | 21 4.285714 .7171372 3 5

hdroom | 22 2.613636 .4862837 1.5 3.5

trunk | 22 11.40909 3.216906 5 16

weight | 22 2315.909 433.0035 1760 3420

length | 22 168.5455 13.68255 142 193

turn | 22 35.40909 1.501082 32 38

displ | 22 111.2273 24.88054 79 163

gratio | 22 3.507273 .2969076 2.98 3.89

foreign | 22 1 0 1 1

Let's review all those pieces.

A command can be preceded with a by clause, as shown below, with **summarize** preceded with **by**

**by foreign: summarize**

There are many parts that can come after a command, they are each presented separately below.  
For example, **summarize** follwed by the names of variables

**summarize mpg price**

**summarize** with **in** specifying a range of records to be summarized.

**summarize in 1/10**

**summarize** with simple **if** specifying records to summarize.

**summarize if foreign == 1**

**summarize** with complex **if** specifying records to summarize.

**summarize if foreign == 1 & mpg > 30**

**summarize** followed by option(s).

**summarize , detail**

So, putting it all together, the general syntax of the summarize command can be described as:

**[by varlist:] summarize [varlist] [in range] [if exp] , [options]**

Understanding the overall syntax of Stata commands helps you remember them and use them more effectively, and it also helps you understand the help in Stata. All the extra stuff about **by**, **if** and **in** could be confusing. Let's have a look at the help file for summarize. It makes more sense knowing what the **by**, **if** and **in** parts mean.

**help summarize**

----------------------------------------------------------------------------

help for summarize (manual: [R] summarize)

-----------------------------------------------------------------------------

Summary statistics

------------------

[by varlist:] summarize [varlist] [weight] [if exp] [in range]

[, { detail | meanonly } format ]

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

**Welcome to the Institute for Digital Reseach and Education**

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

Stata Learning Module  
Using and saving files in Stata

Using and saving Stata data files

The **use** command gets a Stata data file from disk and places it in memory so you can analyze and/or modify it. A data file must be read into memory before you can analyze it. It is kind of like when you open a **Word** document; you need to read a **Word** document into **Word** before you can work with it. The **use** command below gets the Stata data file called **auto.dta** from disk and places it in memory so we can analyze and/or modify it. Since Stata data files end with **.dta** you need only say **use auto** and Stata knows to read in the file called **auto.dta**.

**sysuse auto**

The **describe** command tells you information about the data that is currently sitting in memory.

**describe**

Contains data from auto.dta

obs: 74

vars: 12 17 Feb 1999 10:49

size: 3,108 (99.6% of memory free)

-------------------------------------------------------------------------------

1. make str17 %17s

2. price int %9.0g

3. mpg byte %9.0g

4. rep78 byte %9.0g

5. hdroom float %9.0g

6. trunk byte %9.0g

7. weight int %9.0g

8. length int %9.0g

9. turn byte %9.0g

10. displ int %9.0g

11. gratio float %9.0g

12. foreign byte %9.0g

-------------------------------------------------------------------------------

Sorted by:

Now that the data is in memory, we can analyze it. For example, the **summarize** command gives summary statistics for the data currently in memory.

**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

hdroom | 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

displ | 74 197.2973 91.83722 79 425

gratio | 74 3.014865 .4562871 2.19 3.89

foreign | 74 .2972973 .4601885 0 1

Let's make a change to the data in memory. We will compute a variable called **price2** which will be double the value of price.

**generate price2 = 2\*price**

If we use the **describe** command again, we see the variable we just created is part of the data in memory. We also see a note from Stata saying **dataset has changed since last saved**. Stata knows that the data in memory has changed, and would need to be saved to avoid losing the changes. It is like when you are editing a **Word** document; if you don't save the data, any changes you make will be lost. If we shut the computer off before saving the changes, the changes we made would be lost.

**describe**

Contains data from auto.dta

obs: 74

vars: 13 17 Feb 1999 10:49

size: 3,404 (99.6% of memory free)

-------------------------------------------------------------------------------

1. make str17 %17s

2. price int %9.0g

3. mpg byte %9.0g

4. rep78 byte %9.0g

5. hdroom float %9.0g

6. trunk byte %9.0g

7. weight int %9.0g

8. length int %9.0g

9. turn byte %9.0g

10. displ int %9.0g

11. gratio float %9.0g

12. foreign byte %9.0g

13. price2 float %9.0g

-------------------------------------------------------------------------------

Sorted by:

Note: dataset has changed since last saved

The **save** command is used to save the data in memory permanently on disk. Let's save this data and call it **auto2** (Stata will save it as **auto2.dta**).

**save auto2**

file auto2.dta saved

Let's make another change to the dataset. We will compute a variable called **price3** which will be three times the value of price.

**generate price3 = 3\*price**

Let's try to save this data again to **auto2**

**save auto2**

file auto2.dta already exists

r(602);

Did you see how Stata said **file auto2.dta already exists**? Stata is worried that you will accidentally overwrite your data file. You need to use the **replace** option to tell Stata that you know that the file exists and you want to replace it.

**save auto2, replace**

file auto2.dta saved

Let's make another change to the data in memory by creating a variable called **price4** that is four times the **price**.

**generate price4 = price\*4**

Suppose we want to use the original **auto** file and we don't care if we lose the changes we just made in memory (i.e., losing the variable **price4**). We can try to **use** the **auto** file.

**sysuse auto**

no; data in memory would be lost

r(4);

See how Stata refused to **use** the file, saying **no; data in memory would be lost**? Stata did not want you to lose the changes that you made to the data sitting in memory. If you really want to discard the changes in memory, then use need to use the **clear** option on the **use** command, as shown below.

**sysuse auto, clear**

Stata tries to protect you from losing your data by doing the following:  
1. If you want to **save** a file over an existing file, you need to use the **replace** option, e.g., **save auto, replace**.  
2. If you try to **use** a file and the file in memory has unsaved changes, you need to use the **clear** option to tell Stata that you want to discard the changes, e.g., **use auto, clear**.

Before we move on to the next topic, let's clear out the data in memory.

**clear**

Using files larger than 1 megabyte

When you use a data file, Stata reads the entire file into memory. By default, Stata limits the size of data in memory to 1 megabyte (PC version 6.0 Intercooled). You can view the amount of memory that Stata has reserved for data with the **memory** command.

**memory**

Total memory 1,048,576 bytes 100.00%

overhead (pointers) 0 0.00%

data 0 0.00%

------------

data + overhead 0 0.00%

programs, saved results, etc. 1,152 0.11%

------------

Total 1,152 0.11%

Free 1,047,424 99.89%

If you try to **use** a file which exceeds the amount of memory Stata has allocated for data, it will give you an error message like this.  
**no room to add more observations**  
**r(901);**  
You can increase the amount of memory that Stata has allocated to data using the **set memory** command. For example, if you had a data file which was 1.5 megabytes, you can set the memory to, say, 2 megabytes shown below.

**set memory 2m**

(2048k)

Once you have increased the memory, you should be able to **use** the data file if you have allocated enough memory for it.

Summary

To **use** the **auto** file from disk and read it into memory

**sysuse auto**

To **save** the file **auto** from memory to disk

**save auto**

To **save** a file if the file **auto** already exists

**save auto, replace**

to **use** a file **auto** and **clear** out the current data in memory

**sysuse auto, clear**

If you want to **clear** out the data in memory, you want to lose the changes

**clear**

To allocate 2 megabytes of memory for a data file.

**set memory 2m**

To view the allocation of memory to data and how much is used.

**memory**

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

IDRE Research Technology Group

|  |  |
| --- | --- |
|  |  |

**Welcome to the Institute for Digital Reseach and Education**

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
Inputting your data into Stata**

This module will show how to input your data into Stata. This covers inputting data with comma delimited, tab delimited, space delimited, and fixed column data.

**Note**: all of the sample input files for this page were created by us and are not included with Stata. You can create them yourself to try out this code by copying and pasting the data into a text file.

**1. Typing data into the Stata editor**

One of the easiest methods for getting data into Stata is using the Stata data editor, which resembles an Excel spreadsheet. It is useful when your data is on paper and needs to be typed in, or if your data is already typed into an Excel spreadsheet. To learn more about the Stata data editor, see the **edit** module.

**2. Comma/tab separated file with variable names on line 1**

Two common file formats for raw data are **comma separated files** and **tab separated files**. Such files are commonly made from spreadsheet programs like **Excel**. Consider the **comma delimited** file shown below.

**type auto2.raw**

make, mpg, weight, price

AMC Concord, 22, 2930, 4099

AMC Pacer, 17, 3350, 4749

AMC Spirit, 22, 2640, 3799

Buick Century, 20, 3250, 4816

Buick Electra, 15,4080, 7827

This file has two characteristics:  
- The first line has the names of the variables separated by commas,  
- The following lines have the values for the variables, also separated by commas.

This kind of file can be read using the **insheet** command, as shown below.

**insheet using auto2.raw**

(4 vars, 5 obs)

We can check to see if the data came in right using the **list** command.

**list**

make mpg weight price

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

4. Buick Century 20 3250 4816

5. Buick Electra 15 4080 7827

Since you will likely have more observations, you can use **in** to list just a subset of observations. Below, we **list** observations 1 through 3.

**list in 1/3**

make mpg weight price

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

Now that the file has been read into Stata, you can save it with the **save** command (we will skip doing that step).

The exact same **insheet** command could be used to read a **tab delimited** file. The **insheet** command is clever because it can figure out whether you have a **comma delimited** or **tab delimited** file, and then read it. (However, **insheet** could not handle a file that uses a mixture of commas and tabs as delimiters.)

Before starting the next section, let's clear out the existing data in memory.

**clear**

**3. Comma/tab separated file (no variable names in file)**

Consider a file that is identical to the one we examined in the previous section, but it does not have the variable names on line 1

**type auto3.raw**

AMC Concord, 22, 2930, 4099

AMC Pacer, 17, 3350, 4749

AMC Spirit, 22, 2640, 3799

Buick Century, 20, 3250, 4816

Buick Electra, 15,4080, 7827

This file can be read using the **insheet** command as shown below.

**insheet using auto3.raw**

(4 vars, 5 obs)

But where did Stata get the variable names? If Stata does not have names for the variables, it names them **v1**, **v2**, **v3** etc., as you can see below.

**list**

v1 v2 v3 v4

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

4. Buick Century 20 3250 4816

5. Buick Electra 15 4080 7827

Let's clear out the data in memory, and then try reading the data again.

**clear**

Now, let's try reading the data and tell Stata the names of the variables on the **insheet** command.

**insheet make mpg weight price using auto3.raw**

(4 vars, 5 obs)

As the **list** command shows, Stata used the variable names supplied on the **insheet** command.

**list**

make mpg weight price

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

4. Buick Century 20 3250 4816

5. Buick Electra 15 4080 7827

The **insheet** command works equally well on files which use tabs as separators. Stata examines the file and determines whether commas or tabs are being used as separators and reads the file appropriately.

Now that the file has been read into Stata, you can save it with the **save** command (we will skip doing that step).

Let's clear out the data in memory before going to the next section.

**clear**

**4. Space separated file**

Consider a file where the variables are separated by spaces like the one shown below.

**type auto4.raw**

"AMC Concord" 22 2930 4099

"AMC Pacer" 17 3350 4749

"AMC Spirit" 22 2640 3799

"Buick Century" 20 3250 4816

"Buick Electra" 15 4080 7827

Note that the make of car is contained within quotation marks. This is necessary because the names contain spaces within them. Without the quotes, Stata would think AMC is the **make** and Concord is the **mpg**. If the **make** did not have spaces embedded within them, the quotation marks would not be needed.

This file can be read with the **infile** command as shown below.

**infile str13 make mpg weight price using auto4.raw**

(5 observations read)

You may be asking yourself, where did the **str13** come from? Since **make** is a character variable, we need to tell Stata that it is a character variable, and how long it can be. The **str13** tells Stata it is a **str**ing variable and that it could be up to 13 characters wide.

The **list** command confirms that the data was read correctly.

**list**

make mpg weight price

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

4. Buick Century 20 3250 4816

5. Buick Electra 15 4080 7827

Now that the file has been read into Stata, you can save it with the **save** command (we will skip doing that step).

Let's clear out the data in memory before moving on to the next section.

**clear**

**5. Fixed format file**

Consider a file using fixed column data like the one shown below.

**type auto5.raw**

AMC Concord 22 2930 4099

AMC Pacer 17 3350 4749

AMC Spirit 22 2640 3799

Buick Century 20 3250 4816

Buick Electra 15 4080 7827

Note that the variables are clearly defined by which column(s) they are located. Also, note that the **make** of car is not contained within quotation marks. The quotations are not needed because the columns define where the **make** begins and ends, and the embedded spaces no longer create confusion.

This file can be read with the **infix** command as shown below.

**infix str make 1-13 mpg 15-16 weight 18-21 price 23-26 using auto5.raw**

(5 observations read)

Here again we need to tell Stata that **make** is a **str**ing variable by preceding **make** with **str**. We did not need to indicate the length since Stata can infer that **make** can be up to 13 characters wide based on the column locations.

The **list** command confirms that the data was read correctly.

**list**

make mpg weight price

1. AMC Concord 22 2930 4099

2. AMC Pacer 17 3350 4749

3. AMC Spirit 22 2640 3799

4. Buick Century 20 3250 4816

5. Buick Electra 15 4080 7827

Now that the file has been read into Stata, you can save it with the **save** command (we will skip doing that step).

Let's clear out the data in memory before moving on to the next section.

**clear**

**6. Other methods of getting data into Stata**

This does not cover all possible methods of getting raw data into Stata, but does cover many common situations. See the Stata Users Guide for more comprehensive information on reading raw data into Stata.

Another method that should be mentioned is the use of data conversion programs. These programs can convert data from one file format into another file format. For example, they could directly create a Stata file from an Excel Spreadsheet, a Lotus Spreadsheet, an Access database, a Dbase database, a SAS data file, an SPSS system file, etc. Two such examples are Stat Transfer and DBMS Copy. Both of these products are available on SSC PCs and DBMS Copy is available on Nicco and Aristotle.

Finally, if you are using Nicco, Aristotle or the RS/6000 Cluster, there is a command specifically for converting SAS data into Stata called **sas2stata**. If you have SAS data you want to convert to Stata, this may be a useful way to get your SAS data into Stata.

**7. Summary**

Bring up the Stata data editor for typing data in.

**. edit**

Read in the comma or tab delimited file called **auto2.raw** taking the variable names from the first line of data.

**. insheet using auto2.raw, clear**

Read in the comma or tab delimited file called **auto3.raw** naming the variables mpg weight and price.

**. insheet make mpg weight price using auto3.raw, clear**

Read in the space separated file named **auto4.raw**. The variable make is surrounded by quotes because it has embedded blanks.

**. infile str13 make mpg weight price using auto4.raw, clear**

Read in the fixed format file named **auto5.raw**.

**. infix str make 1-13 mpg 15-16 weight 18-21 using auto5.raw, clear**

Other methods  
**DBMS/Copy, Stat Transfer, sas2stata, and Stata Users Guide**.

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.

[Institute for Digital Reseach and Education Home](https://idre.ucla.edu)

Help the Stat Consulting Group by [giving a gift](https://giving.ucla.edu/Standard/NetDonate.aspx?SiteNum=371)

Top of Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | [×](javascript:void(0)) | |  |  |

|  |  |  |
| --- | --- | --- |
|  | powered by |  |

Bottom of Form

[stat](http://www.ats.ucla.edu/stat) > [stata](http://www.ats.ucla.edu/stat/stata) > [modules](http://www.ats.ucla.edu/stat/stata/modules) >

You're not lost. We have a new look but the same content.

**Stata Learning Module  
Using dates in Stata**

This module will show how to use date variables, date functions, and date display formats in Stata.

**Converting dates from raw data using the "date()" function**

The trick to inputting dates in Stata is to forget they are dates, and treat them as character strings, and then later convert them into a Stata date variable. You might have the following date data in your raw data file.

**type dates1.raw**

John 1 Jan 1960

Mary 11 Jul 1955

Kate 12 Nov 1962

Mark 8 Jun 1959

You can read these data by typing:

**infix str name 1-4 str bday 6-17 using dates1.raw**

(4 observations read)

Using the **list** command, you can see that the date information has been read correctly into **bday**.

**list**

name bday

1. John 1 Jan 1960

2. Mary 11 Jul 1955

3. Kate 12 Nov 1962

4. Mark 8 Jun 1959

Since **bday** is a string variable, you cannot do any kind of date computations with it until you make a date variable from it. You can generate a date version of **bday** using the **date()** function. The example below creates a date variable called **birthday** from the character variable **bday**. The syntax is slightly different depending on which version of Stata you are using. The difference is in how the pattern is specified. In Stata 9 it should be lower case (e.g., "dmy") and in Stata 10, it should be upper case for day, month, and year (e.g., "DMY") but lower case if you want to specify hours, minutes or seconds (e.g., "DMYhms"). Our data are in the order day, month, year, so we use "DMY" (or "dmy" if you are using Stata 9) within the **date()** command. (Unless otherwise noted, all other Stata commands on this page are the same for versions 9 and 10.)

In Stata **version 9**:

**generate birthday=date(bday,"dmy")**

In Stata **version 10**:

**generate birthday=date(bday,"DMY")**

Let's have a look at both **bday** and **birthday**.

**list**

name bday birthday

1. John 1 Jan 1960 0

2. Mary 11 Jul 1955 -1635

3. Kate 12 Nov 1962 1046

4. Mark 8 Jun 1959 -207

The values for birthday may seem confusing. The value of **birthday** for John is 0 and the value of **birthday** for Mark is -207. Dates are actually stored as **the number of days from Jan 1, 1960** which is convenient for the computer storing and performing date computations, but is difficult for you and I to read.

We can tell Stata that **birthday** should be displayed using the %d format to make it easier for humans to read.

**format birthday %d**

**list**

name bday birthday

1. John 1 Jan 1960 01jan1960

2. Mary 11 Jul 1955 11jul1955

3. Kate 12 Nov 1962 12nov1962

4. Mark 8 Jun 1959 08jun1959

The **date()** function is very flexible and can handle dates written in almost any manner. For example, consider the file **dates2.raw**.

**type dates2.raw**

John Jan 1 1960

Mary 07/11/1955

Kate 11.12.1962

Mark Jun/8 1959

These dates are messy, but they are consistent. Even though the formats look different, it is always a month day year separated by a delimiter (e.g., space slash dot or dash). We can try using the syntax from above to read in our new dates. Note that, as discussed above, for Stata version 10 the order of the date is declared in upper case letters (i.e., "MDY") while for version 9 it is declared in all lower case (i.e., "mdy").

**clear**

**infix str name 1-4 str bday 6-17 using dates2.raw**

(4 observations read)

**generate birthday=date(bday,"MDY")**

**format birthday %d**

**list**

name bday birthday

1. John Jan 1 1960 01jan1960

2. Mary 07/11/1955 11jul1955

3. Kate 11.12.1962 12nov1962

4. Mark Jun/8 1959 08jun1959

Stata was able to read those dates without a problem. Let's try an even tougher set of dates. For example, consider the dates in **dates3.raw**.

**type dates3.raw**

4-12-1990

4.12.1990

Apr 12, 1990

Apr12,1990

April 12, 1990

4/12.1990

Apr121990

Let's try reading these dates and see how Stata handles them. Again, remember that for Stata version 10 dates are declared "MDY" while for version 9 they are declared "mdy".

**clear**

**infix str bday 1-20 using dates3.raw**

(7 observations read)

**generate birthday=date(bday,"MDY")**

(1 missing value generated)

**format birthday %d**

**list**

bday birthday

1. 4-12-1990 12apr1990

2. 4.12.1990 12apr1990

3. Apr 12, 1990 12apr1990

4. Apr12,1990 12apr1990

5. April 12, 1990 12apr1990

6. 4/12.1990 12apr1990

7. Apr121990 .

As you can see, Stata was able to handle almost all of those crazy date formats. It was able to handle Apr12,1990 even though there was not a delimiter between the month and day (Stata was able to figure it out since the month was character and the day was a number). The only date that did not work was Apr121990 and that is because there was no delimiter between the day and year. As you can see, the **date()** function can handle just about any date as long as there are delimiters separating the month day and year. In certain cases Stata can read all numeric dates entered without delimiters, see **help dates** for more information.

**Converting dates from raw data using the mdy() function**

In some cases, you may have the month, day, and year stored as numeric variables in a dataset. For example, you may have the following data for birth dates from **dates4.raw**.

**type dates4.raw**

7 11 1948

1 1 1960

10 15 1970

12 10 1971

You can read in this data using the following syntax to create a separate variable for month, day and year.

**clear**

**infix month 1-2 day 4-5 year 7-10 using dates4.raw**

(4 observations read)

**list**

month day year

1. 7 11 1948

2. 1 1 1960

3. 10 15 1970

4. 12 10 1971

A Stata date variable can be created using the mdy() function as shown below.

**generate birthday=mdy(month,day,year)**

Let's format birthday using the **%d** format so it displays better.

**format birthday %d**

**list**

month day year birthday

1. 7 11 1948 11jul1948

2. 1 1 1960 01jan1960

3. 10 15 1970 15oct1970

4. 12 10 1971 10dec1971

Consider the data in **dates5.raw**, which is the same as dates4.raw except that only two digits are used to signify the year.

**type dates5.raw**

7 11 48

1 1 60

10 15 70

12 10 71

Let's try reading these dates just like we read **dates4.raw**.

**clear**

**infix month 1-2 day 4-5 year 7-10 using dates5.raw**

(4 observations read)

**generate birthday=mdy(month,day,year)**

(4 missing values generated)

**format birthday %d**

**list**

month day year birthday

1. 7 11 48 .

2. 1 1 60 .

3. 10 15 70 .

4. 12 10 71 .

As you can see, the values for **birthday** are all missing. This is because Stata assumes that the years were literally 48, 60, 70 and 71 (it does not assume they are 1948, 1960, 1970 and 1971). You can force Stata to assume the century portion is 1900 by adding 1900 to the year as shown below (note that we use **replace** instead of **generate** since the variable **birthday** already exists).

**replace birthday=mdy(month,day,year+1900)**

(4 real changes made)

**format birthday %d**

**list**

month day year birthday

1. 7 11 48 11jul1948

2. 1 1 60 01jan1960

3. 10 15 70 15oct1970

4. 12 10 71 10dec1971

**Computations with elapsed dates**

Date variables make computations involving dates very convenient. For example, to calculate everyone's age on January 1, 2000 simply use the following conversion.

**generate age2000=( mdy(1,1,2000) - birthday ) / 365.25**

**list**

month day year birthday age2000

1. 7 11 48 11jul1948 51.47433

2. 1 1 60 01jan1960 40

3. 10 15 70 15oct1970 29.21287

4. 12 10 71 10dec1971 28.06023

Please note that this formula for age does not work well over very short time spans. For example, the age for a child on their his birthday will be less than one due to using 365.25. There are formulas that are more exact but also much more complex. Here is an example courtesy of Dan Blanchette.

**generate altage = floor(([ym(2000, 1) - ym(year(birthday), month(birthday))] - [1 < day(birthday)]) / 12)**

**Other date functions**

Given a date variable, one can have the month, day and year returned separately if desired, using the **month()**, **day()** and **year()** functions, respectively.

**generate m=month(birthday)**

**generate d=day(birthday)**

**generate y=year(birthday)**

**list m d y birthday**

m d y birthday

1. 7 11 1948 11jul1948

2. 1 1 1960 01jan1960

3. 10 15 1970 15oct1970

4. 12 10 1971 10dec1971

If you'd like to return the **day of the week** for a date variable, use the **dow()** function (where 0=Sunday, 1=Monday etc.).

**gen week\_d=dow(birthday)**

**list birthday week\_d**

birthday week\_d

1. 11jul1948 0

2. 01jan1960 5

3. 15oct1970 4

4. 10dec1971 5

**Summary**

The **date()** function converts strings containing dates to date variables. The syntax varies slightly by version.

In Stata **version 9**:

**gen date2 = date(date, "dmy")**

In Stata **version 10**:

**gen date2 = date(date, "DMY")**

The **mdy()** function takes three numeric arguments (month, day, year) and converts them to a date variable.

**generate birthday=mdy(month,day,year)**

You can display elapsed times as actual dates with display formats such as the **%d** format.

**format birthday %d**

Other date functions include the **month()**, **day()**, **year()**, and **dow()** functions. For online help with dates, type **help dates** at the command line. For more detailed explanations about how Stata handles dates and date functions, please refer to the Stata Users Guide.

[How to cite this page](http://www.ats.ucla.edu/stat/mult_pkg/faq/general/citingats.htm)

[Report an error on this page or leave a comment](http://www.ats.ucla.edu/stat/apps/err_report.htm)

The content of this web site should not be construed as an endorsement of any particular web site, book, or software product by the University of California.