Title

stata.com

import delimited — Import delimited text data

Description
Quick start
Menu
Syntax
Options for import delimited
Options for export delimited
Remarks and examples
Also see

Description

`import delimited` reads into memory a text-delimited file from disk. Regardless of the program that created the file, `import delimited` reads text (ASCII, UTF-8, or UTF-16) files in which there is one observation per line and the values are separated by commas, tabs, or some other delimiter. The first line of the file can contain the variable names.

Stata has other commands for importing data. If you are not sure that `import delimited` will do what you are looking for, see [D] **import** and **[U] 21 Entering and importing data**.

`export delimited`, by default, writes data into a file in comma-separated (`.csv`) format. `export delimited` also allows you to specify any separation character delimiter that you prefer. The exported text file is UTF-8 encoded.

Quick start

Load comma-delimited `mydata.csv` with 2 variables to be named `v1` and `v2`

```
import delimited v1 v2 using mydata
```

As above, but with variable names on the first row

```
import delimited mydata
```

As above, but with variable names in row 5 and an ignorable header in the first 4 rows

```
import delimited mydata, varnames(5)
```

Load only columns 2 to 300 and the first 1,000 rows with variable names in row 1

```
import delimited mydata, colrange(2:300) rowrange(:1000)
```

Load tab-delimited data from `mydata.txt`

```
import delimited mydata.txt, delimiters(tab)
```

Load semicolon-delimited data from `mydata.txt`

```
import delimited mydata.txt, delimiters(";")
```

Force columns 2 to 6 to be read as string to preserve leading zeros

```
import delimited mydata, stringcols(2/6)
```

Export data in memory to `mydata.csv`

```
export delimited mydata
```

As above, but export only `v1` and `v2`

```
export delimited v1 v2 using mydata
```

As above, but output numeric values for variables with value labels

```
export delimited v1 v2 using mydata, nolabel
```

Menu

import delimited

File > Import > Text data (delimited, *.csv, ...)

export delimited

File > Export > Text data (delimited, *.csv, ...)

Syntax

Load a delimited text file

```
import delimited [using] filename [, import_delimited_options]
```

Rename specified variables from a delimited text file

```
import delimited extvarlist using filename [, import_delimited_options]
```

Save data in memory to a delimited text file

```
export delimited [using] filename [if] [in] [, export_delimited_options]
```

Save subset of variables in memory to a delimited text file

```
export delimited [varlist] using filename [if] [in] [, export_delimited_options]
```

If *filename* is specified without an extension, `.csv` is assumed for both `import delimited` and `export delimited`. If *filename* contains embedded spaces, enclose it in double quotes.

extvarlist specifies variable names of imported columns.

import_delimited_options	Description
`delimiters("`*chars*`"[, collapse|asstring])`	use *chars* as delimiters
`rowrange([`*start*`][:`*end*`])`	row range of data to load
`colrange([`*start*`][:`*end*`])`	column range of data to load
`varnames(#|nonames)`	treat row # of data as variable names or the data do not have variable names
`case(preserve|lower|upper)`	preserve the case or read variable names as lowercase (the default) or uppercase
`asdouble`	import all floating-point data as `doubles`
`asfloat`	import all floating-point data as `floats`
`clear`	replace data in memory
`bindquotes(loose|strict|nobind)`	specify how to handle double quotes in data
`stripquotes(yes|no|default)`	remove or keep double quotes in data
`numericcols(`*numlist*`|_all)`	force specified columns to be numeric
`stringcols(`*numlist*`|_all)`	force specified columns to be string
`encoding("`*encoding*`")`	specify the encoding of the text file being imported

export_delimited_options	Description
Main	
`delimiter("`*char*`" | tab)`	use *char* as delimiter
`novarnames`	do not write variable names on the first line
`nolabel`	output numeric values (not labels) of labeled variables
`datafmt`	use the variables' display format upon export
`quote`	always enclose strings in double quotes
`replace`	overwrite existing *filename*

Options for import delimited

`delimiters("`*chars*`"[ , collapse | asstring])` allows you to specify other separation characters. For instance, if values in the file are separated by a semicolon, specify `delimiters(";")`. By default, `import delimited` will check if the file is delimited by tabs or commas based on the first line of data. Specify `delimiters("\t")` to use a tab character, or specify `delimiters("whitespace")` to use whitespace as a delimiter.

`collapse` forces `import delimited` to treat multiple consecutive delimiters as just one delimiter.

`asstring` forces `import delimited` to treat *chars* as one delimiter. By default, each character in *chars* is treated as an individual delimiter.

`rowrange([`*start*`][:`*end*`])` specifies a range of rows within the data to load. *start* and *end* are integer row numbers.

`colrange([`*start*`][:`*end*`])` specifies a range of variables within the data to load. *start* and *end* are integer column numbers.

`varnames(#|nonames)` specifies where or whether variable names are in the data. By default, `import delimited` tries to determine whether the file includes variable names. `import delimited` translates the names in the file to valid Stata variable names. The original names from the file are stored unmodified as variable labels.

`varnames(#)` specifies that the variable names are in row # of the data; any data before row # should not be imported.

`varnames(nonames)` specifies that the variable names are not in the data.

`case(preserve|lower|upper)` specifies the case of the variable names after import. The default is `case(lowercase)`.

`asdouble` imports floating-point data as type `double`. The default storage type of the imported variables is determined by `set type`.

`asfloat` imports floating-point data as type `float`. The default storage type of the imported variables is determined by `set type`.

`clear` specifies that it is okay to replace the data in memory, even though the current data have not been saved to disk.

`bindquotes(loose|strict|nobind)` specifies how `import delimited` handles double quotes in data. Specifying `loose` (the default) tells `import delimited` that it must have a matching open and closed double quote on the same line of data. `strict` tells `import delimited` that once it finds one double quote on a line of data, it should keep searching through the data for

the matching double quote even if that double quote is on another line. Specifying `nobind` tells `import delimited` to ignore double quotes for binding.

`stripquotes(yes | no | default)` tells `import delimited` how to handle double quotes. `yes` causes all double quotes to be stripped. `no` leaves double quotes in the data unchanged. `default` automatically strips quotes that can be identified as binding quotes. `default` also will identify two adjacent double quotes as a single double quote because some software encodes double quotes that way.

`numericcols(`*numlist* | `_all)` forces the data type of the column numbers in *numlist* to be numeric. Specifying `_all` will import all data as numeric.

`stringcols(`*numlist* | `_all)` forces the data type of the column numbers in *numlist* to be string. Specifying `_all` will import all data as strings.

`encoding("`*encoding*`")` specifies the encoding of the text file to be imported. The default is `encoding("latin1")`. Specify `encoding("utf-8")` for the files to be encoded in UTF-8. `import delimited` uses Java encoding. A list of available encodings can be found at http://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html.

Option `charset()` is a synonym for `encoding()`.

Options for export delimited

`delimiter("`*char*`" | tab)` allows you to specify other separation characters. For instance, if you want the values in the file to be separated by a semicolon, specify `delimiter(";")`. The default delimiter is a comma.

`delimiter(tab)` specifies that a tab character be used as the delimiter.

`novarnames` specifies that variable names not be written in the first line of the file; the file is to contain data values only.

`nolabel` specifies that the numeric values of labeled variables be written into the file rather than the label associated with each value.

`datafmt` specifies that all variables be exported using their display format. For example, the number 1000 with a display format of `%4.2f` would export as `1000.00`, not `1000`. The default is to use the raw, unformatted value when exporting.

`quote` specifies that string variables always be enclosed in double quotes. The default is to only double quote strings that contain spaces or the delimiter.

`replace` specifies that *filename* be replaced if it already exists.

Remarks and examples

stata.com

Remarks are presented under the following headings:

import delimited
export delimited
Video example

import delimited

`import delimited` reads in text data where each data point is separated by a delimiter character. The two most common types of text data to import are comma-space-value (`.csv`) text files and tab-separated text files. `import delimited` can automatically detect either a comma or a tab as the delimiter. To import your data, type

```
. import delimited filename
```

`import delimited` reads your data if

1. it can find the file; and
2. the file meets `import delimited`'s expectations as to its format.

If you type `import delimited myfile`, `myfile.csv` is read into Stata. If your file is called `myfile.txt`, type `import delimited using myfile.txt`. If typing `import delimited` *filename* does not produce the desired result, you may need to specify an option or try one of Stata's other import commands; see [D] **import**.

▷ Example 1

We have a `.csv` data file on automobiles called `auto.csv`.

```
. copy http://www.stata.com/examples/auto.csv auto.csv
. type auto.csv
make,price,mpg,rep78,foreign
"AMC Concord",4099,22,3,"Domestic"
"AMC Pacer",4749,17,3,"Domestic"
"AMC Spirit",3799,22,,"Domestic"
"Buick Century",4816,20,3,"Domestic"
"Buick Electra",7827,15,4,"Domestic"
"Buick LeSabre",5788,18,3,"Domestic"
"Buick Opel",4453,26,,"Domestic"
"Buick Regal",5189,20,3,"Domestic"
"Buick Riviera",10372,16,3,"Domestic"
"Buick Skylark",4082,19,3,"Domestic"
```

This file was saved by a spreadsheet and can be read by typing

```
. import delimited auto
```

To look at what we just loaded, type

```
. list
```

	make	price	mpg	rep78	foreign
1.	AMC Concord	4099	22	3	Domestic
2.	AMC Pacer	4749	17	3	Domestic
3.	AMC Spirit	3799	22	.	Domestic
4.	Buick Century	4816	20	3	Domestic
5.	Buick Electra	7827	15	4	Domestic
6.	Buick LeSabre	5788	18	3	Domestic
7.	Buick Opel	4453	26	.	Domestic
8.	Buick Regal	5189	20	3	Domestic
9.	Buick Riviera	10372	16	3	Domestic
10.	Buick Skylark	4082	19	3	Domestic

These data contain a combination of string and numeric variables. `import delimited` will determine the correct data type for each variable. You can also force the data type of a variable by using the `numericcols()` or `stringcols()` option.

◁

▷ Example 2

`import delimited` allows you to read in a subset of the text data by using the `rowrange()` and `colrange()` options. To read rows 2 through 5 of `auto.csv`, you need to specify `rowrange(3:6)` because the first row of data contains the variable names.

```
. clear
. import delimited auto, rowrange(3:6)
(5 vars, 4 obs)
. list
```

	make	price	mpg	rep78	foreign
1.	AMC Pacer	4749	17	3	Domestic
2.	AMC Spirit	3799	22	.	Domestic
3.	Buick Century	4816	20	3	Domestic
4.	Buick Electra	7827	15	4	Domestic

We used `rowrange(3:6)` instead of `rowrange(2:5)` because row 1 of the data contained the variable names.

To import the first three columns and last four rows of `auto.csv`, type

```
. clear
. import delimited auto, colrange(:3) rowrange(8)
(3 vars, 4 obs)
. list
```

	make	price	mpg
1.	Buick Opel	4453	26
2.	Buick Regal	5189	20
3.	Buick Riviera	10372	16
4.	Buick Skylark	4082	19

◁

▷ Example 3

`import delimited` can handle delimiters other than commas and tabs. Suppose that you had the `auto.txt` file.

```
. type auto.txt, showtabs
"AMC Concord"   4099   22  3   "Domestic"
"AMC Pacer"     4749   17  3   "Domestic"
"AMC Spirit"    3799   22  NA  "Domestic"
"Buick Century" 4816   20  3   "Domestic"
"Buick Electra" 7827   15  4   "Domestic"
"Buick LeSabre" 5788   18  3   "Domestic"
"Buick Opel"    4453   26  NA  "Domestic"
"Buick Regal"   5189   20  3   "Domestic"
"Buick Riviera" 10372  16  3   "Domestic"
"Buick Skylark" 4082   19  3   "Domestic"
```

We specified type's showtabs option so that no tabs are shown. These data are not tab-delimited or comma-delimited. If you use import delimited without any options, you will not get the results you expect.

```
. clear
. import delimited auto.txt
(1 var, 10 obs)
```

When import delimited tries to read data that have no tabs or commas, it is fooled into thinking that the data contain just one variable. You can use the delimiter() option to import the data correctly. delimiter(" ") tells import delimited to use spaces (" ") as the delimiter, and delimiter(, collapse) will treat multiple consecutive space delimiters as one delimiter.

```
. clear
. import delimited auto.txt, delimiter(" ", collapse)
(5 vars, 10 obs)
. describe
Contains data
  obs:            10
 vars:             5
 size:           260
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
v1              str13   %13s
v2              int     %8.0g
v3              byte    %8.0g
v4              str2    %9s
v5              str8    %9s
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
. list
```

	v1	v2	v3	v4	v5
1.	AMC Concord	4099	22	3	Domestic
2.	AMC Pacer	4749	17	3	Domestic
3.	AMC Spirit	3799	22	NA	Domestic
4.	Buick Century	4816	20	3	Domestic
5.	Buick Electra	7827	15	4	Domestic
6.	Buick LeSabre	5788	18	3	Domestic
7.	Buick Opel	4453	26	NA	Domestic
8.	Buick Regal	5189	20	3	Domestic
9.	Buick Riviera	10372	16	3	Domestic
10.	Buick Skylark	4082	19	3	Domestic

The data that were loaded now contain the correct number of variables and observations. However, the variable rep78 should be a numeric variable, but it was imported as a string because the value NA was used for missing values. To force rep78 to have a numeric storage type, use the option numericcols().

```
. clear
. import delimited auto.txt, delim(" ", collapse) numericcols(4)
(5 vars, 10 obs)
. describe
Contains data
  obs:            10
 vars:             5
 size:           250
```

variable name	storage type	display format	value label	variable label
v1	str13	%13s		
v2	int	%8.0g		
v3	byte	%8.0g		
v4	byte	%8.0g		
v5	str8	%9s		

```
Sorted by:
     Note: Dataset has changed since last saved.
. list
```

	v1	v2	v3	v4	v5
1.	AMC Concord	4099	22	3	Domestic
2.	AMC Pacer	4749	17	3	Domestic
3.	AMC Spirit	3799	22	.	Domestic
4.	Buick Century	4816	20	3	Domestic
5.	Buick Electra	7827	15	4	Domestic
6.	Buick LeSabre	5788	18	3	Domestic
7.	Buick Opel	4453	26	.	Domestic
8.	Buick Regal	5189	20	3	Domestic
9.	Buick Riviera	10372	16	3	Domestic
10.	Buick Skylark	4082	19	3	Domestic

◁

export delimited

`export delimited` creates a comma-separated text file from the Stata dataset in memory. If your goal is to send data to another Stata user, you could use `export delimited`, but it is better to send a `.dta` dataset. This will work even if you use Stata for Windows and your colleague uses Stata for Mac or Unix. All versions of Stata can read each other's `.dta` files.

To view other methods for moving your data into other applications, see [D] **export**.

▷ Example 4

To save the data currently in memory into a specified `.csv` file, type

```
. use http://www.stata-press.com/data/r14/auto, clear
(1978 Automobile Data)
. export delimited myauto
file myauto.csv saved
```

◁

▷ Example 5

You can also export a subset of the data in memory by typing

```
. use http://www.stata-press.com/data/r14/auto
(1978 Automobile Data)
. export delimited make mpg rep78 foreign in 1/10 using myauto
file myauto.csv already exists
r(602);
```

If the file already exists, you can use `replace` to write over it:

```
. export delimited make mpg rep78 foreign in 1/10 using myauto, replace
. type myauto.csv
make,mpg,rep78,foreign
AMC Concord,22,3,Domestic
AMC Pacer,17,3,Domestic
AMC Spirit,22,,Domestic
Buick Century,20,3,Domestic
Buick Electra,15,4,Domestic
Buick LeSabre,18,3,Domestic
Buick Opel,26,,Domestic
Buick Regal,20,3,Domestic
Buick Riviera,16,3,Domestic
Buick Skylark,19,3,Domestic
```

◁

Video example

Importing delimited data

Also see

[D] **export** — Overview of exporting data from Stata

[D] **import** — Overview of importing data into Stata