outfile — Export dataset in text format

Description

`outfile` writes data to a disk file in plain text format, which can be read by other programs. The new file is *not* in Stata format; see [D] **save** for instructions on saving data for later use in Stata.

The data saved by `outfile` can be read back by `infile`; see [D] **import**. If *filename* is specified without an extension, `.raw` is assumed unless the `dictionary` option is specified, in which case `.dct` is assumed. If your *filename* contains embedded spaces, remember to enclose it in double quotes.

Quick start

Export current dataset to space-separated `mydata.raw`

```
outfile using mydata
```

Same as above, but export only `v1`, `v2`, and `v3`

```
outfile v1 v2 v3 using mydata
```

Same as above, but export to comma-separated `mydata.csv`

```
outfile v1 v2 v3 using mydata.csv, comma
```

Export current dataset in Stata's dictionary format to `myfile.dct`

```
outfile v1 v2 v3 using mydata, dictionary
```

Do not allow observations to break across lines

```
outfile using mydata, wide
```

Menu

File > Export > Text data (fixed- or free-format)

Syntax

outfile [*varlist*] using *filename* [*if*] [*in*] [, *options*]

options	Description
Main	
dictionary	write the file in Stata's dictionary format
nolabel	output numeric values (not labels) of labeled variables; the default is to write labels in double quotes
noquote	do not enclose strings in double quotes
comma	write file in comma-separated (instead of space-separated) format
wide	force one observation per line (no matter how wide)
Advanced	
rjs	right-justify string variables; the default is to left-justify
fjs	left-justify if format width < 0; right-justify if format width > 0
runtogether	all on one line, no quotes, no space between, and ignore formats
missing	retain missing values; use only with comma
replace	overwrite the existing file

replace does not appear in the dialog box.

Options

Main

dictionary writes the file in Stata's data dictionary format. See [D] **infile (fixed format)** for a description of dictionaries. comma, missing, and wide are not allowed with dictionary.

nolabel causes Stata to write the numeric values of labeled variables. The default is to write the labels enclosed in double quotes.

noquote prevents Stata from placing double quotes around the contents of strings, meaning string variables and value labels.

comma causes Stata to write the file in comma-separated–value format. In this format, values are separated by commas rather than by blanks. Missing values are written as two consecutive commas unless missing is specified.

wide causes Stata to write the data with 1 observation per line. The default is to split observations into lines of 80 characters or fewer, but strings longer than 80 characters are never split across lines.

Advanced

rjs and fjs affect how strings are justified; you probably do not want to specify either of these options. By default, outfile outputs strings left-justified in their field.

If rjs is specified, strings are output right-justified. rjs stands for "right-justified strings".

If fjs is specified, strings are output left- or right-justified according to the variable's format: left-justified if the format width is negative and right-justified if the format width is positive. fjs stands for "format-justified strings".

runtogether is a programmer's option that is valid only when all variables of the specified *varlist* are of type string. runtogether specifies that the variables be output in the order specified, without quotes, with no spaces between, and ignoring the display format attached to each variable. Each observation ends with a new line character.

missing, valid only with comma, specifies that missing values be retained. When comma is specified without missing, missing values are changed to null strings ("").

The following option is available with outfile but is not shown in the dialog box:

replace permits outfile to overwrite an existing dataset.

Remarks and examples

outfile enables data to be sent to a disk file for processing by a non-Stata program. Each observation is written as one or more records that will not exceed 80 characters unless you specify the wide option. Each column other than the first is prefixed by two blanks.

outfile is careful to put the data in columns in case you want to read the data by using formatted input. String variables and value labels are output in left-justified fields by default. You can change this behavior by using the rjs or fjs options.

Numeric variables are output right-justified in the field width specified by their display format. A numeric variable with a display format of %9.0g will be right-justified in a nine-character field. Commas are not written in numeric variables, even if a comma format is used.

If you specify the dictionary option, the data are written in the same way, but preceding the data, outfile writes a data dictionary describing the contents of the file.

▷ Example 1: Basic usage

We have entered into Stata some data on seven employees in our firm. The data contain employee name, employee identification number, salary, and sex:

```
. list
```

	name	empno	salary	sex
1.	Carl Marks	57213	24,000	male
2.	Irene Adler	47229	27,000	female
3.	Adam Smith	57323	24,000	male
4.	David Wallis	57401	24,500	male
5.	Mary Rogers	57802	27,000	female
6.	Carolyn Frank	57805	24,000	female
7.	Robert Lawson	57824	22,500	male

The last variable in our data, sex, is really a numeric variable, but it has an associated value label.

If we now wish to use a program other than Stata with these data, we must somehow get the data over to that other program. The standard Stata-format dataset created by save will not do the job—it is written in a special format that only Stata understands. Most programs, however, understand plain text datasets, such as those produced by a text editor. We can tell Stata to produce such a dataset by using outfile. Typing outfile using employee creates a dataset called employee.raw that contains all the data. We can use the Stata type command to review the resulting file:

```
. outfile using employee
. type employee.raw
"Carl Marks"            57213      24000  "male"
"Irene Adler"           47229      27000  "female"
"Adam Smith"            57323      24000  "male"
"David Wallis"          57401      24500  "male"
"Mary Rogers"           57802      27000  "female"
"Carolyn Frank"         57805      24000  "female"
"Robert Lawson"         57824      22500  "male"
```

We see that the file contains the four variables and that Stata has surrounded the string variables with double quotes.

◁

❑ Technical note

The `nolabel` option prevents Stata from substituting value-label strings for the underlying numeric values; see **[U] 12.6.3 Value labels**. The last variable in our data is really a numeric variable:

```
. outfile using employ2, nolabel
. type employ2.raw
"Carl Marks"            57213      24000          0
"Irene Adler"           47229      27000          1
"Adam Smith"            57323      24000          0
"David Wallis"          57401      24500          0
"Mary Rogers"           57802      27000          1
"Carolyn Frank"         57805      24000          1
"Robert Lawson"         57824      22500          0
```

❑

❑ Technical note

If you do not want Stata to place double quotes around the contents of string variables, you can specify the `noquote` option:

```
. outfile using employ3, noquote
. type employ3.raw
Carl Marks              57213      24000  male
Irene Adler             47229      27000  female
Adam Smith              57323      24000  male
David Wallis            57401      24500  male
Mary Rogers             57802      27000  female
Carolyn Frank           57805      24000  female
Robert Lawson           57824      22500  male
```

❑

▷ Example 2: Overwriting an existing file

Stata never writes over an existing file unless explicitly told to do so. For instance, if the file `employee.raw` already exists and we attempt to overwrite it by typing `outfile using employee`, here is what would happen:

```
. outfile using employee
file employee.raw already exists
r(602);
```

We can tell Stata that it is okay to overwrite a file by specifying the `replace` option:

```
. outfile using employee, replace
```

◁

❑ Technical note

Some programs prefer data to be separated by commas rather than by blanks. Stata produces such a dataset if you specify the `comma` option:

```
. outfile using employee, comma replace
. type employee.raw
"Carl Marks",57213,24000,"male"
"Irene Adler",47229,27000,"female"
"Adam Smith",57323,24000,"male"
"David Wallis",57401,24500,"male"
"Mary Rogers",57802,27000,"female"
"Carolyn Frank",57805,24000,"female"
"Robert Lawson",57824,22500,"male"
```

❑

▷ Example 3: Creating data dictionaries

Finally, `outfile` can create data dictionaries that `infile` can read. Dictionaries are perhaps the best way to organize your raw data. A dictionary describes your data so that you do not have to remember the order of the variables, the number of variables, the variable names, or anything else. The file in which you store your data becomes self-documenting so that you can understand the data in the future. See [D] **infile (fixed format)** for a full description of data dictionaries.

When you specify the `dictionary` option, Stata writes a `.dct` file:

```
. outfile using employee, dict replace
. type employee.dct
dictionary {
        str15  name              `"Employee name"'
        float  empno             `"Employee number"'
        float  salary            `"Annual salary"'
        float  sex     :sexlbl   `"Sex"'
}
"Carl Marks"            57213      24000  "male"
"Irene Adler"           47229      27000  "female"
"Adam Smith"            57323      24000  "male"
"David Wallis"          57401      24500  "male"
"Mary Rogers"           57802      27000  "female"
"Carolyn Frank"         57805      24000  "female"
"Robert Lawson"         57824      22500  "male"
```

◁

▷ Example 4: Working with dates

We have historical data on the S&P 500 for the month of January 2001.

```
. use https://www.stata-press.com/data/r19/outfilexmpl, clear
(S&P 500)
. describe
Contains data from https://www.stata-press.com/data/r19/outfilexmpl.dta
 Observations:            21                  S&P 500
    Variables:             6                  6 Apr 2024 16:02
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
date	int	%td		Date
open	float	%9.0g		Opening price
high	float	%9.0g		High price
low	float	%9.0g		Low price
close	float	%9.0g		Closing price
volume	int	%12.0gc		Volume (thousands)

```
Sorted by: date
```

The `date` variable has a display format of `%td` so that it is displayed as *ddmmmyyyy*.

```
. list
```

	date	open	high	low	close	volume
1.	02jan2001	1320.28	1320.28	1276.05	1283.27	11,294
2.	03jan2001	1283.27	1347.76	1274.62	1347.56	18,807
3.	04jan2001	1347.56	1350.24	1329.14	1333.34	21,310
4.	05jan2001	1333.34	1334.77	1294.95	1298.35	14,308
5.	08jan2001	1298.35	1298.35	1276.29	1295.86	11,155
6.	09jan2001	1295.86	1311.72	1295.14	1300.8	11,913
7.	10jan2001	1300.8	1313.76	1287.28	1313.27	12,965
8.	11jan2001	1313.27	1332.19	1309.72	1326.82	14,112
9.	12jan2001	1326.82	1333.21	1311.59	1318.55	12,760
10.	16jan2001	1318.32	1327.81	1313.33	1326.65	12,057
11.	17jan2001	1326.65	1346.92	1325.41	1329.47	13,491
12.	18jan2001	1329.89	1352.71	1327.41	1347.97	14,450
13.	19jan2001	1347.97	1354.55	1336.74	1342.54	14,078
14.	22jan2001	1342.54	1353.62	1333.84	1342.9	11,640
15.	23jan2001	1342.9	1362.9	1339.63	1360.4	12,326
16.	24jan2001	1360.4	1369.75	1357.28	1364.3	13,090
17.	25jan2001	1364.3	1367.35	1354.63	1357.51	12,580
18.	26jan2001	1357.51	1357.51	1342.75	1354.95	10,980
19.	29jan2001	1354.92	1365.54	1350.36	1364.17	10,531
20.	30jan2001	1364.17	1375.68	1356.2	1373.73	11,498
21.	31jan2001	1373.73	1383.37	1364.66	1366.01	12,953

We outfile our data and use the `type` command to view the result.

```
. outfile using sp

. type sp.raw
"02jan2001"    1320.28    1320.28    1276.05    1283.27         11294
"03jan2001"    1283.27    1347.76    1274.62    1347.56         18807
"04jan2001"    1347.56    1350.24    1329.14    1333.34         21310
"05jan2001"    1333.34    1334.77    1294.95    1298.35         14308
"08jan2001"    1298.35    1298.35    1276.29    1295.86         11155
"09jan2001"    1295.86    1311.72    1295.14     1300.8         11913
"10jan2001"     1300.8    1313.76    1287.28    1313.27         12965
"11jan2001"    1313.27    1332.19    1309.72    1326.82         14112
"12jan2001"    1326.82    1333.21    1311.59    1318.55         12760
"16jan2001"    1318.32    1327.81    1313.33    1326.65         12057
"17jan2001"    1326.65    1346.92    1325.41    1329.47         13491
"18jan2001"    1329.89    1352.71    1327.41    1347.97         14450
"19jan2001"    1347.97    1354.55    1336.74    1342.54         14078
"22jan2001"    1342.54    1353.62    1333.84     1342.9         11640
"23jan2001"     1342.9     1362.9    1339.63     1360.4         12326
"24jan2001"     1360.4    1369.75    1357.28     1364.3         13090
"25jan2001"     1364.3    1367.35    1354.63    1357.51         12580
"26jan2001"    1357.51    1357.51    1342.75    1354.95         10980
"29jan2001"    1354.92    1365.54    1350.36    1364.17         10531
"30jan2001"    1364.17    1375.68     1356.2    1373.73         11498
"31jan2001"    1373.73    1383.37    1364.66    1366.01         12953
```

The `date` variable, originally stored as an `int`, was outfiled as a string variable. Whenever Stata outfiles a variable with a date format, Stata outfiles the variable as a `string`.

◁

Also see

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

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

[U] 22 Entering and importing data