How do I use file write to write results or other information to an external text file?

Authored by

stats writer

July 1, 2024

Using file write to write results or other information to an external text file involves using a programming language or a specific software application to save data or output to a text file. This process typically involves opening the text file, writing the desired information or results to it, and then saving the file. This allows for easy storage and organization of data and allows for easy retrieval of the information at a later time. It is commonly used in various fields, such as data analysis, programming, and research, as a way to store and share information in a structured and accessible format.

How do I use file write to write results or other information to an external text file? | Stata FAQ

The file write command can be used to write results or other information generated by Stata to a text file which can be read using a variety of programs found on most, if not all, computers. In the example below, we create a file containing descriptive statistics in a compact format, which can be read either by humans or as data by Stata or another program. To do this we generate descriptive statistics for four variables (read, math, science, and socst) from a dataset that contains data on student scores on achievement tests. Then we will write the descriptive statistics to a text file. We will show two sets of similar syntax, one that creates a

tab separated text
file, and one to write the same information using a fixed format. The end result
will be a text file, the contents of which look something like the table below.

variable mean median variance s.d. skewness kurtosis N
read 52.230 50.000 105.123 10.253 0.195 2.363 200
math 52.645 52.000 87.768 9.368 0.284 2.337 200
science 51.850 53.000 98.028 9.901 -0.187 2.428 200
socst 52.405 52.000 115.257 10.736 -0.379 2.459 200

Using file write to create a tab separated text file

The first line of code below opens the dataset via the web. In order to write
to a file, we first need to open that file, so the second line below
instructs Stata to open a file (file open), and tells Stata that within
Stata we will refer to this file as myfile. The word using followed by a file path and
name tell Stata were we want the text file we are going to write to to be saved. In the options

**(i.e. after the comma) we
specify write which tells Stata that we are going to be writing (i.e.
adding information to, but not getting information from) the file.
At the end of the line, specifying the replace option tells Stata that if it encounters
another file with that file path and name, that it is okay to overwrite the old
file. In the third line of code, we set more off, which tells Stata to allow our
screen to scroll, rather than stopping at the end of each line with "more" at
the bottom of our screen.**

```
use https://stats.idre.ucla.edu/stat/stata/notes/hsb2, clear
file open myfile using "D:datatab_sep.txt", write replace
set more off
```

**Before we go any further, it may be useful to know that the command structure
of all file related commands is similar. They start with the word file, followed by the type of operation we want**

to perform
(in this case, open the file), then we see whatever name that we have
given the file within Stata (also known as a handle), followed by something
else, what this something else is will depend on what we are doing with the
file. You have to specify the name of the file (in our example it is called
myfile), because it is possible to have more than one file open at a time,
therefore the name of the file is used to indicate which file we are referring
to with any given command.

Once the file is open, we can write our header, which will label the columns in
the file. Although it appears on multiple lines, Stata will see the following syntax as
a single command, because we have ended all but the last line in ///. The /// tells Stata
to ignore the line break and any blank spaces, and to instead to look for the
first non-space

character on the following line, and begin reading there. Looking at the first line of the syntax below, we see the command file write myfile. The file tells Stata that we want to do something involving a file, write indicates what we want to do to the file, and myfile tells Stata which file we want to write to. The rest of the command falls into the category of "something else" that is mentioned above. In this case, the something else is the text we want to write, plus formatting commands. The text we want to actually see in our file is in quotation marks, the _tab commands tell Stata to insert a tab in that spot, and the _n tells Stata to go to the next line.

```
file write myfile "variable" _tab "mean" ///
_tab "median" _tab "variance" ///
_tab "s.d." _tab "skewness" ///
_tab "kurtosis" _tab "N" _n
```

The first line below starts with foreach var of varlist followed by a

list of variables and finally an open curly brace ({). This command tells
Stata we want to perform
the operations between the open curly brace ({) and the close curly brace (}
) below once for each variable in the in a list.
The var indicates that we will use the local macro var in place of
the names of our variables.
The next line gives the command sum, asking for descriptive statistics for whichever variable name
Stata is currently substituting for `var'. Stata uses a ` (it's below the ~
character at the top
of your keyboard) followed by the name of the local macro, and then a ' (a normal
apostrophe) to represent whatever
you want to fill in (e.g. `var'). We use the , detail option so that
Stata will return a greater variety of descriptive statistics. We are not really
interested in the output this produces on screen, but when we run this command
Stata also places information in the returned results,

**which we can then use to
write to our file. (See our Stata FAQ How can I access information stored after I run a command in Stata (returned results)?
for more information on returned results.)**

**The next command begins on the third line and is similar to the syntax above where we wrote the header. We have used the
/// again, to allow us to put a single command on multiple lines. The third line starts with the command file write myfile and
continues with the information we actually want to write to the file. The parts
of this statement that begin with "%" (e.g. %9s or %7.3f) tell Stata how
to format whatever comes next. (Type "help format" in the command window for
more information on formatting data.) To the right of %9s is "`var'" the set of
quotation marks (" ") are used to enclose a string, in this case, the string
they enclose is `var' which refers to the local macro we are using as a**

place holder for each of our variable names. This means that at the beginning of each
line in our output file, we will see the name of one of our variables. This is
followed by _tab which inserts a tab. Next we see %7.3f which formats whatever follows it, in this case, (r(mean)) the parentheses,
like the quotation marks above,
are used to enclose the value to be printed, in this case that value to be
printed is
the mean, which is contained in r(mean).

```
foreach var of varlist read write math science socst {
sum `var', detail
file write myfile %9s "`var'" _tab %7.3f (r(mean)) ///
_tab %7.3f (r(p50)) _tab %7.3f (r(Var)) ///
_tab %7.3f (r(sd)) _tab %7.3f (r(skewness)) ///
_tab %7.3f (r(kurtosis)) _tab (r(N)) _n
}
```

Stata will cycle through the commands in the curly braces once for each
variable in our list, running descriptive statistics, then

writing them, along
with the variable name to our file, then it will move on to whatever is next.
Since we have written all the necessary information to our file we are now done
with it. The first line below tells Stata that we would like to close the file known as
myfile.
This means we will not be able to read or write to this file unless we open it again.
Finally, we turn "more" back on (set more on) returning Stata to its
default behavior.

```
file close myfile
set more on
```

If we open the text file Stata produces, we should see the text shown below, a
header row, followed by one row for each variable we wanted descriptive
statistics for.
Note that the header row and the data rows do not line up quite right, this

makes it more difficult for many humans to read, but, if a computer were to read
this file its only concern would be that each value is separated by a tab, and
our file satisfies this requirement.

variable mean median variance s.d. skewness kurtosis N
read 52.230 50.000 105.123 10.253 0.195 2.363 200
math 52.645 52.000 87.768 9.368 0.284 2.337 200
science 51.850 53.000 98.028 9.901 -0.187 2.428 200
socst 52.405 52.000 115.257 10.736 -0.379 2.459 200

Put all the code together in a single block, so it's easy for you to cut and paste
into a .do file, the code looks like this:

```
use https://stats.idre.ucla.edu/stat/stata/notes/hsb2, clear

file open myfile using "D:datatab_sep.txt", write replace
file write myfile "variable" _tab "mean" ///
_tab "median" _tab "variance" ///
_tab "s.d." _tab "skewness" ///
_tab "kurtosis" _tab "N" _n
```

```
set more off

foreach var of varlist read write math science socst {
sum `var', detail
file write myfile %9s "`var'" _tab %7.3f (r(mean)) ///
_tab %7.3f (r(p50)) _tab %7.3f (r(Var)) ///
_tab %7.3f (r(sd)) _tab %7.3f (r(skewness)) ///
_tab %7.3f (r(kurtosis)) _tab (r(N)) _n
}
file close myfile
set more on
```

Using file write to create a fixed format text file

The syntax for this example is very similar to the syntax above, the only difference is in the file write commands. In those lines of syntax, the _tab commands are replaced with _column(#) where the number sign (a.k.a.. pound sign) is the column number at which we want to start writing the piece of information that follows.

```
use https://stats.idre.ucla.edu/stat/stata/notes/hsb2, clear
```

```
file open myfile using "D:datafixed.txt", write replace
set more off

file write myfile "variable" _column(11) "mean" ///
_column(20) "median" _column(29) "variance" ///
_column(40) "s.d." _column(47) "skewness" ///
_column(57) "kurtosis" _column(68) "N" _n
foreach var of varlist read math science socst {
sum `var', detail
file write myfile %9s "`var'" _column(11) %7.3f (r(mean)) ///
_column(20) %7.3f (r(p50)) _column(29) %7.3f (r(Var)) ///
_column(38) %7.3f (r(sd)) _column(47) %7.3f (r(skewness)) ///
_column(57) %7.3f (r(kurtosis)) _column(68) (r(N)) _n

}

file close myfile
set more on
```

The text file this syntax produces is shown below. This file closely resembles the tab separated file, but has "cleaner" columns.

variable mean median variance s.d. skewness kurtosis N

read 52.230 50.000 105.123 10.253 0.195 2.363 200

math 52.645 52.000 87.768 9.368 0.284 2.337 200

science 51.850 53.000 98.028 9.901 -0.187 2.428 200

socst 52.405 52.000 115.257 10.736 -0.379 2.459 200