xmlsave — Export or import dataset in XML format

Syntax

Export dataset in memory to XML format

```
xmlsave filename [if] [in] [, xmlsave_options]
```

Export subset of dataset in memory to XML format

```
xmlsave varlist using filename [if] [in] [, xmlsave_options]
```

Import XML-format dataset

```
xmluse filename [, xmluse_options]
```

xmlsave_options	Description
Main	
`doctype(dta)`	save XML file by using Stata's `.dta` format
`doctype(excel)`	save XML file by using Excel XML format
`dtd`	include Stata DTD in XML file
`legible`	format XML to be more legible
`replace`	overwrite existing *filename*

xmluse_options	Description
`doctype(dta)`	load XML file by using Stata's `.dta` format
`doctype(excel)`	load XML file by using Excel XML format
`sheet("`*sheetname*`")`	Excel worksheet to load
`cells(`*upper-left*`:`*lower-right*`)`	Excel cell range to load
`datestring`	import Excel dates as strings
`allstring`	import all Excel data as strings
`firstrow`	treat first row of Excel data as variable names
`missing`	treat inconsistent Excel types as missing
`nocompress`	do not compress Excel data
`clear`	replace data in memory

Menu

xmlsave

File > Export > XML data

xmluse

File > Import > XML data

Description

`xmlsave` and `xmluse` allow datasets to be exported or imported in XML file formats for Stata's `.dta` and Microsoft Excel's SpreadsheetML format. XML files are advantageous because they are structured text files that are highly portable between applications that understand XML.

Stata can directly import files in Microsoft Excel `.xls` or `.xlsx` format. If you have files in that format or you wish to export files to that format, see [D] **import excel**.

`xmlsave` exports the data in memory in the `dta` XML format by default. To export the data, type

```
. xmlsave filename
```

although sometimes you will want to explicitly specify which document type definition (DTD) to use by typing

```
. xmlsave filename, doctype(dta)
```

`xmluse` can read either an Excel-format XML or a Stata-format XML file into Stata. You type

```
. xmluse filename
```

Stata will read into memory the XML file *filename*`.xml`, containing the data after determining whether the file is of document type `dta` or `excel`. As with the `xmlsave` command, the document type can also be explicitly specified with the `doctype()` option.

```
. xmluse filename, doctype(dta)
```

It never hurts to specify the document type; it is actually recommended because there is no guarantee that Stata will be able to determine the document type from the content of the XML file. Whenever the `doctype()` option is omitted, a note will be displayed that identifies the document type Stata used to load the dataset.

If *filename* is specified without an extension, `.xml` is assumed.

`xmlsave` cannot save `strL`s.

Options for xmlsave

Main

`doctype(dta|excel)` specifies the DTD to use when exporting the dataset.

`doctype(dta)`, the default, specifies that an XML file will be exported using Stata's `.dta` format (see [P] **file formats .dta**). This is analogous to Stata's binary `dta` format for datasets. All data that can normally be represented in a normal `dta` file will be represented by this document type.

doctype(excel) specifies that an XML file will be exported using Microsoft's SpreadsheetML DTD. SpreadsheetML is the term given by Microsoft to the Excel XML format. Specifying this document type produces a generic spreadsheet with variable names as the first row, followed by data. It can be imported by any version of Microsoft Excel that supports Microsoft's SpreadsheetML format.

dtd when combined with doctype(dta) embeds the necessary DTD into the XML file so that a validating parser of another application can verify the dta XML format. This option is rarely used, however, because it increases file size with information that is purely optional.

legible adds indents and other optional formatting to the XML file, making it more legible for a person to read. This extra formatting, however, is unnecessary and in larger datasets can significantly increase the file size.

replace permits xmlsave to overwrite existing *filename*.xml.

Options for xmluse

doctype(dta|excel) specifies the DTD to use when loading data from *filename*.xml. Although it is optional, use of doctype() is encouraged. If this option is omitted with xmluse, the document type of *filename*.xml will be determined automatically. When this occurs, a note will display the document type used to translate *filename*.xml. This automatic determination of document type is not guaranteed, and the use of this option is encouraged to prevent ambiguity between various XML formats. Specifying the document type explicitly also improves speed, as the data are only passed over once to load, instead of twice to determine the document type. In larger datasets, this advantage can be noticeable.

doctype(dta) specifies that an XML file will be loaded using Stata's dta format. This document type follows closely Stata's binary .dta format (see [P] **file formats .dta**).

doctype(excel) specifies that an XML file will be loaded using Microsoft's SpreadsheetML DTD. SpreadsheetML is the term given by Microsoft to the Excel XML format.

sheet("*sheetname*") imports the worksheet named *sheetname*. Excel files can contain multiple worksheets within one document, so using the sheet() option specifies which of these to load. The default is to import the first worksheet to occur within *filename*.xml.

cells(*upper-left*:*lower-right*) specifies a cell range within an Excel worksheet to load. The default range is the entire range of the worksheet, even if portions are empty. Often the use of cells() is necessary because data are offset within a spreadsheet, or only some of the data need to be loaded. Cell-range notation follows the letter-for-column and number-for-row convention that is popular within all spreadsheet applications. The following are valid examples:

```
. xmluse filename, doctype(excel) cells(A1:D100)
. xmluse filename, doctype(excel) cells(C23:AA100)
```

datestring forces all Excel SpreadsheetML date formats to be imported as strings to retain time information that would otherwise be lost if automatically converted to Stata's date format. With this option, time information can be parsed from the string after loading it.

allstring forces Stata to import all Excel SpreadsheetML data as string data. Although data type information is dictated by SpreadsheetML, there are no constraints to keep types consistent within columns. When such inconsistent use of data types occurs in SpreadsheetML, the only way to resolve inconsistencies is to import data as string data.

firstrow specifies that the first row of data in an Excel worksheet consist of variable names. The default behavior is to generate generic names. If any name is not a valid Stata variable name, a generic name will be substituted in its place.

missing forces any inconsistent data types within SpreadsheetML columns to be imported as missing data. This can be necessary for various reasons but often will occur when a formula for a particular cell results in an error, thus inserting a cell of type ERROR into a column that was predominantly of a NUMERIC type.

nocompress specifies that data not be compressed after loading from an Excel SpreadsheetML file. Because data type information in SpreadsheetML can be ambiguous, Stata initially imports with broad data types and, after all data are loaded, performs a compress (see [D] **compress**) to reduce data types to a more appropriate size. The following table shows the data type conversion used before compression and the data types that would result from using nocompress:

SpreadsheetML type	Initial Stata type
String	str2045
Number	double
Boolean	double
DateTime	double
Error	str2045

clear clears data in memory before loading from *filename*.xml.

Remarks and examples

stata.com

XML stands for Extensible Markup Language and is a highly adaptable text format derived from SGML. The World Wide Web Consortium is responsible for maintaining the XML language standards. See http://www.w3.org/XML/ for information regarding the XML language, as well as a thorough definition of its syntax.

The document type dta, used by both xmlsave and xmluse, represents Stata's own DTD for representing Stata .dta files in XML. Stata reserves the right to modify the specification for this DTD at any time, although this is unlikely to be a frequent event.

The document type excel, used by both xmlsave and xmluse, corresponds to the DTD developed by Microsoft for use in modern versions of Microsoft Excel spreadsheets. This product may incorporate intellectual property owned by Microsoft Corporation. The terms and conditions under which Microsoft is licensing such intellectual property may be found at

http://www.microsoft.com/openspecifications/en/us/programs/osp/default.aspx

For more information about Microsoft Office and XML, see http://msdn.microsoft.com/en-us/library/aa140066%28office.10%29.aspx.

❑ Technical note

When you import data from Excel to Stata, a common hurdle is handling Excel's use of inconsistent data types within columns. Numbers, strings, and other types can be mixed freely within a column of Excel data. Stata, however, requires that all data in a variable be of one consistent type. This can cause problems when a column of data from Excel is imported into Stata and the data types vary across rows.

By default, `xmluse` attempts to import Excel data by using the data type information stored in the XML file. If an error due to data type inconsistencies is encountered, you can use the options `firstrow`, `missing`, and `cells()` to isolate the problem while retaining as much of the data-type information as possible.

However, identifying the problem and determining which option to apply can sometimes be difficult. Often you may not care in what format the data are imported into Stata, as long as you can import them. The quick solution for these situations is to use the `allstring` option to guarantee that all the data are imported as strings, assuming that the XML file itself was valid. Often converting the data back into numeric form after they are imported into Stata is easier, given Stata's vast data management commands.

❑

▷ Example 1: Saving XML files

To export the current Stata dataset to a file, `auto.xml`, type

```
. xmlsave auto
```

To overwrite an existing XML dataset with a new file containing the variables `make`, `mpg`, and `weight`, type

```
. xmlsave make mpg weight using auto, replace
```

To export the dataset to an XML file for use with Microsoft Excel, type

```
. xmlsave auto, doctype(excel) replace
```

◁

▷ Example 2: Using XML files

Assuming that we have a file named `auto.xml` exported using the `doctype(dta)` option of `xmlsave`, we can read in this dataset with the command

```
. xmluse auto, doctype(dta) clear
```

If the file was exported from Microsoft Excel to a file called `auto.xml` that contained the worksheet `Rollover Data`, with the first row representing column headers (or variable names), we could import the worksheet by typing

```
. xmluse auto, doctype(excel) sheet("Rollover Data") firstrow clear
```

Continuing with the previous example: if we wanted just the first column of data in that worksheet, and we knew that there were only 75 rows, including one for the variable name, we could have typed

```
. xmluse auto, doc(excel) sheet("Rollover Data") cells(A1:A75) first clear
```

◁

Also see

[D] **compress** — Compress data in memory

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

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

[P] **file formats .dta** — Description of .dta file format