

**import** — Overview of importing data into Stata[Description](#)[Remarks and examples](#)[Reference](#)[Also see](#)

## Description

This entry provides a quick reference for determining which method to use for reading non-Stata data into memory. See [U] [21 Entering and importing data](#) for more details.

## Remarks and examples

stata.com

Remarks are presented under the following headings:

*Summary of the different methods*

*import excel*

*import delimited*

*odbc*

*infile (free format)—infile without a dictionary*

*infix (fixed format)*

*infile (fixed format)—infile with a dictionary*

*import sasxport*

*import haver (Windows only)*

*xmluse*

*Examples*

*Video example*

## Summary of the different methods

### import excel

- `import excel` reads worksheets from Microsoft Excel (.xls and .xlsx) files.
- Entire worksheets can be read, or custom cell ranges can be read.
- See [D] [import excel](#).

### import delimited

- `import delimited` reads text-delimited files.
- The data can be tab-separated or comma-separated. A custom delimiter may also be specified.
- An observation must be on only one line.
- The first line in the file can optionally contain the names of the variables.
- See [D] [import delimited](#).

### **odbc**

- ODBC, an acronym for Open DataBase Connectivity, is a standard for exchanging data between programs. Stata supports the ODBC standard for importing data via the `odbc` command and can read from any ODBC data source on your computer.
- See [D] [odbc](#).

### **infile (free format)—infile without a dictionary**

- The data can be space-separated, tab-separated, or comma-separated.
- Strings with embedded spaces or commas must be enclosed in quotes (even if tab- or comma-separated).
- An observation can be on more than one line, or there can even be multiple observations per line.
- See [D] [infile \(free format\)](#).

### **infix (fixed format)**

- The data must be in fixed-column format.
- An observation can be on more than one line.
- `infix` has simpler syntax than `infile` (fixed format).
- See [D] [infix \(fixed format\)](#).

### **infile (fixed format)—infile with a dictionary**

- The data may be in fixed-column format.
- An observation can be on more than one line.
- ASCII or EBCDIC data can be read.
- `infile` (fixed format) has the most capabilities for reading data.
- See [D] [infile \(fixed format\)](#).

### **import sasxport**

- `import sasxport` reads SAS XPORT Transport format files.
- `import sasxport` will also read value label information from a `formats.xpf` XPORT file, if available.
- See [D] [import sasxport](#).

### **import haver (Windows only)**

- `import haver` reads Haver Analytics (<http://www.haver.com/>) database files.
- See [D] [import haver](#).

## xmluse

- `xmluse` reads extensible markup language (XML) files—highly adaptable text-format files derived from the standard generalized markup language (SGML).
- `xmluse` can read either an Excel-format XML or a Stata-format XML file into Stata.
- See [D] `xmlsave`.

## Examples

### ▶ Example 1: Tab-separated data

```

-----begin example1.raw-----
 1      0      1      John Smith      m
 0      0      1      Paul Lin        m
 0      1      0      Jan Doe f
 0      0      .      Julie McDonald f
-----end example1.raw-----

```

contains tab-separated data. The `type` command with the `showtabs` option shows the tabs:

```

. type example1.raw, showtabs
1<T>0<T>1<T>John Smith<T>m
0<T>0<T>1<T>Paul Lin<T>m
0<T>1<T>0<T>Jan Doe<T>f
0<T>0<T>.<T>Julie McDonald<T>f

```

It could be read in by

```

. import delimited a b c name gender using example1

```

### ▶ Example 2: Comma-separated data

```

-----begin example2.raw-----
a,b,c,name,gender
1,0,1,John Smith,m
0,0,1,Paul Lin,m
0,1,0,Jan Doe,f
0,0,,Julie McDonald,f
-----end example2.raw-----

```

could be read in by

```

. import delimited using example2

```

### ▶ Example 3: Tab-separated data with double-quoted strings

```

-----begin example3.raw-----
 1      0      1      "John Smith"      m
 0      0      1      "Paul Lin"        m
 0      1      0      "Jan Doe"         f
 0      0      .      "Julie McDonald"      f
-----end example3.raw-----

```

## 4 import — Overview of importing data into Stata

---

contains tab-separated data with strings in double quotes.

```
. type example3.raw, showtabs
1<T>0<T>1<T>"John Smith"<T>m
0<T>0<T>1<T>"Paul Lin"<T>m
0<T>1<T>0<T>"Jan Doe"<T>f
0<T>0<T>.<T>"Julie McDonald"<T>f
```

It could be read in by

```
. infile byte (a b c) str15 name str1 gender using example3
```

or

```
. import delimited a b c name gender using example3
```

or

```
. infile using dict3
```

where the dictionary dict3.dct contains

```
-----begin dict3.dct-----
infile dictionary using example3 {
    byte    a
    byte    b
    byte    c
    str15   name
    str1    gender
}
-----end dict3.dct-----
```

◀

### ▶ Example 4: Space-separated data with double-quoted strings

```
-----begin example4.raw-----
1 0 1 "John Smith" m
0 0 1 "Paul Lin" m
0 1 0 "Jan Doe" f
0 0 . "Julie McDonald" f
-----end example4.raw-----
```

could be read in by

```
. infile byte (a b c) str15 name str1 gender using example4
```

or

```
. infile using dict4
```

where the dictionary dict4.dct contains

```
-----begin dict4.dct-----
infile dictionary using example4 {
    byte    a
    byte    b
    byte    c
    str15   name
    str1    gender
}
-----end dict4.dct-----
```

◀

## ▷ Example 5: Fixed-column format

```

-----begin example5.raw-----
101mJohn Smith
001mPaul Lin
010fJan Doe
00 fJulie McDonald
-----end example5.raw-----

```

could be read in by

```
. infix a 1 b 2 c 3 str gender 4 str name 5-19 using example5
```

or

```
. infix using dict5a
```

where dict5a.dct contains

```

-----begin dict5a.dct-----
infix dictionary using example5 {
      a      1
      b      2
      c      3
str   gender 4
str   name   5-19
}
-----end dict5a.dct-----

```

or

```
. infile using dict5b
```

where dict5b.dct contains

```

-----begin dict5b.dct-----
infile dictionary using example5 {
byte   a      %1f
byte   b      %1f
byte   c      %1f
str1   gender %1s
str15  name   %15s
}
-----end dict5b.dct-----

```

◀

## ▷ Example 6: Fixed-column format with headings

```

-----begin example6.raw-----
line 1 : a heading
There are a total of 4 lines of heading.
The next line contains a useful heading:
-----1-----2-----3-----4-----
1      0      1      m      John Smith
0      0      1      m      Paul Lin
0      1      0      f      Jan Doe
0      0      0      f      Julie McDonald
-----end example6.raw-----

```

could be read in by

```
. infile using dict6a
```

where dict6a.dct contains

```
-----begin dict6a.dct-----
infile dictionary using example6 {
  _firstline(5)
      byte    a
      byte    b
  _column(17) byte    c          %1f
      str1    gender
  _column(33) str15   name       %15s
}
-----end dict6a.dct-----
```

or could be read in by

```
. infix 5 first a 1 b 9 c 17 str gender 25 str name 33-46 using example6
```

or could be read in by

```
. infix using dict6b
```

where dict6b.dct contains

```
-----begin dict6b.dct-----
infix dictionary using example6 {
5 first
      a      1
      b      9
      c      17
  str  gender 25
  str  name   33-46
}
-----end dict6b.dct-----
```

◀

### ▶ Example 7: Fixed-column format with observations spanning multiple lines

```
-----begin example7.raw-----
a b c gender name
1 0 1
m
John Smith
0 0 1
m
Paul Lin
0 1 0
f
Jan Doe
0 0
f
Julie McDonald
-----end example7.raw-----
```

could be read in by

```
. infile using dict7a
```

where dict7a.dct contains

```

-----begin dict7a.dct-----
infile dictionary using example7 {
  _firstline(2)
      byte  a
      byte  b
      byte  c
  _line(2)
      str1  gender
  _line(3)
      str15 name  %15s
}
-----end dict7a.dct-----

```

or, if we wanted to include variable labels,

```
. infile using dict7b
```

where dict7b.dct contains

```

-----begin dict7b.dct-----
infile dictionary using example7 {
  _firstline(2)
      byte  a          "Question 1"
      byte  b          "Question 2"
      byte  c          "Question 3"
  _line(2)
      str1  gender     "Gender of subject"
  _line(3)
      str15 name      %15s
}
-----end dict7b.dct-----

```

infix could also read these data,

```
. infix 2 first 3 lines a 1 b 3 c 5 str gender 2:1 str name 3:1-15 using example7
```

or the data could be read in by

```
. infix using dict7c
```

where dict7c.dct contains

```

-----begin dict7c.dct-----
infix dictionary using example7 {
2 first
      a          1
      b          3
      c          5
      str  gender 2:1
      str  name   3:1-15
}
-----end dict7c.dct-----

```

or the data could be read in by

```
. infix using dict7d
```

where `dict7d.dct` contains

```
----- begin dict7d.dct -----  
infix dictionary using example7 {  
2 first  
      a      1  
      b      3  
      c      5  
  
/  
      str    gender 1  
/  
      str    name   1-15  
}  
----- end dict7d.dct -----
```



### Video example

[Copy/paste data from Excel into Stata](#)

### Reference

Dicle, M. F., and J. Leventis. 2011. [Importing financial data](#). *Stata Journal* 11: 620–626.

### Also see

- [D] [edit](#) — Browse or edit data with Data Editor
- [D] [import delimited](#) — Import delimited text data
- [D] [import excel](#) — Import and export Excel files
- [D] [import haver](#) — Import data from Haver Analytics databases
- [D] [import sasxport](#) — Import and export datasets in SAS XPORT format
- [D] [infile \(fixed format\)](#) — Read text data in fixed format with a dictionary
- [D] [infile \(free format\)](#) — Read unformatted text data
- [D] [infix \(fixed format\)](#) — Read text data in fixed format
- [D] [input](#) — Enter data from keyboard
- [D] [odbc](#) — Load, write, or view data from ODBC sources
- [D] [xmlsave](#) — Export or import dataset in XML format
- [D] [export](#) — Overview of exporting data from Stata
- [U] [21 Entering and importing data](#)