append — Append datasets

Description

append appends Stata-format datasets stored on disk to the end of the dataset in memory. If any filename is specified without an extension, .dta is assumed.

Stata can also join observations from two datasets into one; see [D] merge. See [U] 22 Combining datasets for a comparison of append, merge, and joinby.

Quick start

Append mydata2.dta to mydata1.dta with no data in memory
append using mydata1 mydata2

As above, but with mydata1.dta in memory
append using mydata2

As above, and generate newv to indicate source dataset
append using mydata2, generate(newv)

As above, but do not copy value labels or notes from mydata2.dta
append using mydata2, generate(newv) nolabel nonotes

Only keep v1, v2, and v3 from mydata2.dta
append using mydata2, keep(v1 v2 v3)

Menu

Data > Combine datasets > Append datasets
Syntax

```
append using filename [filename ...] [, options]
```

You may enclose `filename` in double quotes and must do so if `filename` contains blanks or other special characters.

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>generate(newvar)</code></td>
<td><code>newvar</code> marks source of resulting observations</td>
</tr>
<tr>
<td><code>keep(varlist)</code></td>
<td>keep specified variables from appending dataset(s)</td>
</tr>
<tr>
<td><code>nolabel</code></td>
<td>do not copy value-label definitions from dataset(s) on disk</td>
</tr>
<tr>
<td><code>nonotes</code></td>
<td>do not copy notes from dataset(s) on disk</td>
</tr>
<tr>
<td><code>force</code></td>
<td>append string to numeric or numeric to string without error</td>
</tr>
</tbody>
</table>

Options

`generate(newvar)` specifies the name of a variable to be created that will mark the source of observations. Observations from the master dataset (the data in memory before the `append` command) will contain 0 for this variable. Observations from the first using dataset will contain 1 for this variable; observations from the second using dataset will contain 2 for this variable; and so on.

`keep(varlist)` specifies the variables to be kept from the using dataset. If `keep()` is not specified, all variables are kept.

The `varlist` in `keep(varlist)` differs from standard Stata varlists in two ways: variable names in `varlist` may not be abbreviated, except by the use of wildcard characters, and you may not refer to a range of variables, such as `price-weight`.

`nolabel` prevents Stata from copying the value-label definitions from the disk dataset into the dataset in memory. Even if you do not specify this option, label definitions from the disk dataset never replace definitions already in memory.

`nonotes` prevents `notes` in the using dataset from being incorporated into the result. The default is to incorporate notes from the using dataset that do not already appear in the master data.

`force` allows string variables to be appended to numeric variables and vice versa, resulting in missing values from the using dataset. If omitted, `append` issues an error message; if specified, `append` issues a warning message.

Remarks and examples

The disk dataset must be a Stata-format dataset; that is, it must have been created by `save` (see [D] save).

Example 1

We have two datasets stored on disk that we want to combine. The first dataset, called `even.dta`, contains the sixth through eighth positive even numbers. The second dataset, called `odd.dta`, contains the first five positive odd numbers. The datasets are
. use even
(6th through 8th even numbers)
. list

<table>
<thead>
<tr>
<th>number</th>
<th>even</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>6</td>
</tr>
<tr>
<td>2.</td>
<td>7</td>
</tr>
<tr>
<td>3.</td>
<td>8</td>
</tr>
</tbody>
</table>

. use odd
(First five odd numbers)
. list

<table>
<thead>
<tr>
<th>number</th>
<th>odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
</tr>
<tr>
<td>3.</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>4</td>
</tr>
<tr>
<td>5.</td>
<td>5</td>
</tr>
</tbody>
</table>

We will append the even data to the end of the odd data. Because the odd data are already in memory (we just used them above), we type `append using even`. The result is

. append using even
. list

<table>
<thead>
<tr>
<th>number</th>
<th>odd</th>
<th>even</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4.</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5.</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6.</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>7.</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>8.</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

Because the `number` variable is in both datasets, the variable was extended with the new data from the file `even.dta`. Because there is no variable called `odd` in the new data, the additional observations on `odd` were forward-filled with `missing (.`). Because there is no variable called `even` in the original data, the first observations on `even` were back-filled with `missing`. 
Example 2

The order of variables in the two datasets is irrelevant. Stata always appends variables by name:

```
. use http://www.stata-press.com/data/r15/odd1
(First five odd numbers)
. describe
Contains data from http://www.stata-press.com/data/r15/odd1.dta
    obs:      5   First five odd numbers
    vars:     2   9 Jan 2016 08:41
    size:    40
  variablename   type   format     label      variable label
                storage display        value
   odd       float   %9.0g            Odd numbers
  number   float    %9.0g
Sorted by: number
. describe using http://www.stata-press.com/data/r15/even
Contains data 6th through 8th even numbers
    obs:     3   9 Jan 2016 08:43
    vars:     2
    size:    27
  variablename   type   format     label      variable label
                storage display        value
   number   byte    %9.0g
   even    float    %9.0g            Even numbers
Sorted by: number
. append using http://www.stata-press.com/data/r15/even
. list
  odd   number  even
1.  1     1     .
2.  3     2     .
3.  5     3     .
4.  7     4     .
5.  9     5     .
6.     .     6    12
7.     .     7    14
8.     .     8    16
```

The results are the same as those in the first example.

When Stata appends two datasets, the definitions of the dataset in memory, called the master dataset, override the definitions of the dataset on disk, called the using dataset. This extends to value labels, variable labels, characteristics, and date–time stamps. If there are conflicts in numeric storage types, the more precise storage type will be used regardless of whether this storage type was in the master dataset or the using dataset. If a variable is stored as a string in one dataset that is longer than in the other, the longer str# storage type will prevail. If a variable is stored as a strL in one dataset and a str# in another dataset, the strL storage type will prevail.
Technical note

If a variable is a string in one dataset and numeric in the other, Stata issues an error message unless the force option is specified. If force is specified, Stata issues a warning message before appending the data. If the using dataset contains the string variable, the combined dataset will have numeric missing values for the appended data on this variable; the contents of the string variable in the using dataset are ignored. If the using dataset contains the numeric variable, the combined dataset will have empty strings for the appended data on this variable; the contents of the numeric variable in the using dataset are ignored.

Example 3

Because Stata has five numeric variable types—byte, int, long, float, and double—you may attempt to append datasets containing variables with the same name but of different numeric types; see [U] 12.2.2 Numeric storage types.

Let’s describe the datasets in the example above:

```
. describe using http://www.stata-press.com/data/r15/odd
Contains data First five odd numbers
    obs: 5  9 Jan 2016 08:50
    vars: 2
    size: 60

storage  display  value
variable name  type  format  label  variable label
number        float  %9.0g
odd           float  %9.0g  Odd numbers
```

Sorted by:

```
. describe using http://www.stata-press.com/data/r15/even
Contains data 6th through 8th even numbers
    obs: 3  9 Jan 2016 08:43
    vars: 2
    size: 27

storage  display  value
variable name  type  format  label  variable label
number      byte  %9.0g
even        float  %9.0g  Even numbers
```

Sorted by: number

```
. describe using http://www.stata-press.com/data/r15/oddeven
Contains data First five odd numbers
    obs: 8  9 Jan 2016 08:53
    vars: 3
    size: 128

storage  display  value
variable name  type  format  label  variable label
number      float  %9.0g
odd         float  %9.0g  Odd numbers
even    float  %9.0g  Even numbers
```
The `number` variable was stored as a `float` in `odd.dta` but as a `byte` in `even.dta`. Because `float` is the more precise storage type, the resulting dataset, `oddeven.dta`, had `number` stored as a `float`. Had we instead appended `odd.dta` to `even.dta`, `number` would still have been stored as a `float`:

```stata
. use http://www.stata-press.com/data/r15/even, clear
   (6th through 8th even numbers)
. append using http://www.stata-press.com/data/r15/odd
   (note: variable number was byte, now float to accommodate using data's values)
. describe
Contains data from http://www.stata-press.com/data/r15/even.dta
    obs: 8                            stores 6th through 8th even numbers
    vars: 3                 9 Jan 2016 08:43
    size: 96

storage  display value
variable name type format label         variable label
number    float %9.0g                      
even      float %9.0g Even numbers
odd       float %9.0g Odd numbers

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

### Example 4

Suppose that we have a dataset in memory containing the variable `educ`, and we have previously given a label variable `educ "Education Level"` command so that the variable label associated with `educ` is “Education Level”. We now append a dataset called `newdata.dta`, which also contains a variable named `educ`, except that its variable label is “Ed. Lev”. After appending the two datasets, the `educ` variable is still labeled “Education Level”. See [U] 12.6.2 Variable labels.

### Example 5

Assume that the values of the `educ` variable are labeled with a value label named `educlbl`. Further assume that in `newdata.dta`, the values of `educ` are also labeled by a value label named `educlbl`. Thus there is one definition of `educlbl` in memory and another (although perhaps equivalent) definition in `newdata.dta`. When you append the new data, you will see the following:

```stata
. append using newdata
   label educlbl already defined
```

If one label in memory and another on disk have the same name, `append` warns you of the problem and sticks with the definition currently in memory, ignoring the definition in the disk file.
Technical note

When you append two datasets that both contain definitions of the same value label, the codings may not be equivalent. That is why Stata warns you with a message like “label educlbl already defined”. If you do not know that the two value labels are equivalent, you should convert the value-labeled variables into string variables, append the data, and then construct a new coding. `decode` and `encode` make this easy:

```
use newdata, clear
decode educ, gen(edstr)
drop educ
save newdata, replace
use basedata
decode educ, gen(edstr)
drop educ
append using newdata
encode edstr, gen(educ)
drop edstr
```

See [D] `encode`.

You can specify the `nolabel` option to force `append` to ignore all the value-label definitions in the incoming file, whether or not there is a conflict. In practice, you will probably never want to do this.

Example 6

Suppose that we have several datasets containing the populations of counties in various states. We can use `append` to combine these datasets all at once and use the `generate()` option to create a variable identifying from which dataset each observation originally came.

```
use http://www.stata-press.com/data/r15/capop
list
  +------------------+
<table>
<thead>
<tr>
<th>county</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Los</td>
<td>9878554</td>
</tr>
<tr>
<td>2.</td>
<td>Orange</td>
</tr>
<tr>
<td>3.</td>
<td>Ventura</td>
</tr>
</tbody>
</table>
  +------------------+
append using http://www.stata-press.com/data/r15/ilpop>
      http://www.stata-press.com/data/r15/txpop, generate(state)
label define statelab 0 "CA" 1 "IL" 2 "TX"
label values state statelab
```
. list

<table>
<thead>
<tr>
<th>county</th>
<th>pop</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>9878554</td>
<td>CA</td>
</tr>
<tr>
<td>Orange</td>
<td>2997033</td>
<td>CA</td>
</tr>
<tr>
<td>Ventura</td>
<td>798364</td>
<td>CA</td>
</tr>
<tr>
<td>Cook</td>
<td>5285107</td>
<td>IL</td>
</tr>
<tr>
<td>DeKalb</td>
<td>103729</td>
<td>IL</td>
</tr>
<tr>
<td>Will</td>
<td>673586</td>
<td>IL</td>
</tr>
<tr>
<td>Brazos</td>
<td>152415</td>
<td>TX</td>
</tr>
<tr>
<td>Johnson</td>
<td>149797</td>
<td>TX</td>
</tr>
<tr>
<td>Harris</td>
<td>4011475</td>
<td>TX</td>
</tr>
</tbody>
</table>

Also see

[D] cross — Form every pairwise combination of two datasets
[D] joinby — Form all pairwise combinations within groups
[D] merge — Merge datasets
[D] save — Save Stata dataset
[D] use — Load Stata dataset
[U] 22 Combining datasets