frget — Copy variables from linked frame

Description

`frget` copies variables and their associated metadata from the data in the linked frame to the data in the current frame. Copy means copying the relevant observations from the linked frame to the appropriate observations in the current frame. If you would like to refer to a variable in another frame without copying that variable into the current frame, see [D] **fralias**.

See [D] **frames intro** if you do not know what a frame is.

Quick start

Obtain variables `v1`, `v2`, and `v3` from another frame linked to by linkage `lnk`

```
frget v1 v2 v3, from(lnk)
```

Obtain variables `v4` and `v5` via linkage `lnk`, naming them `newv4` and `newv5` in the current frame

```
frget newv4=v4 newv5=v5, from(lnk)
```

Obtain all variables via linkage `lnk`, prefixing them with `l_`

```
frget *, from(lnk) prefix(l_)
```

Obtain all variables via linkage `lnk`, excluding those matching pattern `ind*`

```
frget *, from(lnk) exclude(ind*)
```

Syntax

`frget` *varlist*`, from(`*linkname*`)` [*rename_options*] (1)

`frget` *newvar*$_1$ `=` *varname*$_1$ [*newvar*$_2$ `=` *varname*$_2$ [...]]`, from(`*linkname*`)` (2)

linkname is the name of a *linkvar* in the current frame that was created by `frlink`; see [D] **frlink**.

rename_options	Description
`prefix(`*string*`)`	prefix new variable names with *string*
`suffix(`*string*`)`	suffix new variable names with *string*
`exclude(`*varlist*`)`	exclude specified variables

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Syntax 1 copies the variable names specified by *varlist* from the frame linked by *linkname* to the current frame.

Syntax 2 copies *varname*$_1$ from the frame linked by *linkname* to *newvar*$_1$ in the current frame. Similarly, *varname*$_2$ is copied to *newvar*$_2$ and so on.

Copy means copy and clone. Display formats, variable labels, value labels, notes, and characteristics are also copied.

Options

`from(`*linkname*`)` specifies the identity of the linked frame from which variables are copied. Linkages to frames are created by the `frlink` command. Linkages are usually named for the frame to which they link. Linkage `counties` links to frame `counties`, and so you specify `from(counties)`. If linkage `c` links to frame `counties`, you specify `from(c)`. `from()` is required.

`prefix(`*string*`)` specifies a string to be prefixed to the names of the new variables created in the current frame. Say that you type

```
. frget inc*, from(counties)
```

to request that variables `income` and `income_family` be copied to the current frame. If variable `income` already exists in the current frame, the command would issue an error message to that effect and copy neither variable. To copy the two variables, you could type

```
. frget inc*, from(counties) prefix(c_)
```

Then the variables would be copied to variables named `c_income` and `c_income_family`.

`suffix(`*string*`)` works like `prefix(`*string*`)`, the difference being that the string is suffixed rather than prefixed to the variable names. Both options may be specified if you wish.

`exclude(`*varlist*`)` specifies variables that are not to be copied. An example of the option is

```
frget *, from(counties) exclude(emp*)
```

All variables except variables starting with `emp` would be copied.

More correctly, all variables except `emp*`, `_*`, and the match variables would be copied because `frget` always omits the underscore and match variables. See the explanation below.

Remarks and examples

Remarks are presented under the following headings:

Overview
Everything you need to know about frget

Overview

You have data on people and data on counties. You loaded the datasets and created a linkage named `uscounties` by typing

```
. use people
. frame create uscounties
. frame uscounties: use uscounties
. frlink m:1 countyid, frame(uscounties)
```

See example 1 in [D] **frlink** for details.

Among the variables in `uscounties.dta` is `median_income`. You could copy the variable to the person data in the current frame by typing either of the following:

```
. frget median_income, from(uscounties)
. frget medinc = median_income, from(uscounties)
```

The first command names the copy `median_income` in the current frame. The second names it `medinc`.

Everything you need to know about frget

Here is everything you need to know in outline form:

We make two assumptions in what follows:

A1. The current frame contains data on people. A frame named uscounties contains data on counties. That is, we assume

```
. use people
. frame create uscounties
. frame uscounties: use uscounties
```

A2. The frames are linked on the match variable countyid, which appears in both datasets. The linkage between the frames is named uscounties, the same name as the frame being linked. That is, we assume

```
. frlink m:1 countyid, frame(uscounties)
```

1. What it means to copy a linked variable

   When you type

   ```
   . frget median_income, from(uscounties)
   ```

   frget copies variable median_income from frame uscounties to the current frame. Well, we say it copies the variable, but the process is more complicated than that. frget copies the relevant observations of median_income from frame uscounties to the appropriate observations in the current frame. In the process, frget duplicates some observations and ignores others.

   If the person in observation 1 lives in county 401, then the median income recorded for county 401 in the uscounties frame is copied to observation 1 in the current frame.

   If the people in observations 2, 33, and 65 in the current frame reside in county 207, then the median income recorded for county 207 is duplicated in observations 2, 33, and 65 of the current frame.

   If the person in observation 3 lives in county 599 and there is no county 599 in the uscounties frame, then missing value . or "" is stored in observation 3.

   A copy of a variable from a linked frame is a copy of the relevant observations of the variable to the appropriate observations in the current frame when relevant observations exist.

2. frget can copy variables one at a time

   To copy variable median_income from frame uscounties to the current frame, type

   ```
   . frget median_income, from(uscounties)
   ```

To instead copy median_income to a new variable named medinc in the current frame, type

```
. frget medinc=median_income, from(uscounties)
```

3. frget allows variable names to be abbreviated

   frget allows abbreviations if you have not set varabbrev off. If median_income is the only variable beginning with median in the linked frame, you can type

   ```
   . frget median, from(uscounties)
   ```

   Variable median_income will be copied, and the new variable in the current frame will be named median_income.

   When using frget's *newvar*=*varname* syntax, you can abbreviate the variable being copied that appears to the right of the equals sign:

   ```
   . frget medinc=median, from(uscounties)
   ```

4. frget can bring over groups of variables

   frget allows you to specify a *varlist*. Even though you type frget in the current frame, the *varlist* is interpreted in the linked frame. You can type

   ```
   . frget emp*, from(uscounties)
   . frget emp* median_income, from(uscounties)
   . frget emp* median, from(uscounties)
   . frget emp* m*, from(uscounties)
   . frget *, from(uscounties)
   ```

   When you specify a *varlist*, frget automatically omits the match variable or variables and any variables starting with an underscore (_). First, we will tell you why, and then, we will tell you a workaround.

   We start with a match variable. The match variable(s) in our example is match variable countyid. The variable has the same name in both frames. Pretend for a moment that frget did not exclude match variables. Then, if you tried to copy countyid, that would be an error because frget will not overwrite existing variables. That seems reasonable until you realize that it would also mean that frget would issue an error if you typed

   ```
   . frget c*, from(uscounties)
   ```

   or even if you typed

   ```
   . frget *, from(uscounties)
   ```

   frget would issue errors because c* and * would include countyid, which, being the match variable, already exists in the current frame. frget automatically omits match variables so that you can type frget c* and frget * and get all the other variables.

   frget omits _* variables because they tend to be Stata system variables that are valid only in the dataset in which they appear. You do not want them.

   What if you need to get one of these variables? Use the *newvar*=*varname* syntax. Type, for instance,

   ```
   . frget _myvar=_myvar, frame(uscounties)
   ```

   Automatic omission is not applied to this syntax.

5. frget copies all the variables specified, or none of them

   frget will not overwrite existing variables. If just one variable in the specified list already exists in the current frame, frget copies none of the variables. It issues an error.

```
    . frget emp* m*, from(uscounties)
variable mvalues already exists
r(110);
```

   If you want all the m* variables except mvalues, use the exclude() option:

```
. frget emp* m*, from(uscounties) exclude(mvalues)
```

   If you also want mvalues copied to mvals in the current frame, type

```
. frget mvals=mvalues, from(uscounties)
```

6. frget ignores repeated variables

   It is not an error to type

```
. frget employment employment, from(uscounties)
```

   We specified employment twice, but frget ignores that and copies the variable once. This is convenient because variables can be inadvertently repeated, as in

```
. frget m* employment-larea, from(uscounties)
```

   Although you cannot see it, variable mds is repeated in the example. m* contains mds, and so does employment-larea because mds is among the variables stored between them.

   When variables are repeated using the *newvar*=*varname* syntax, frget does not ignore repetition. It copies the variables you specify to each of the new variables that you specify:

```
. frget medinc=income inc=income, from(uscounties)
```

7. How to get all the variables 1: frget *

   To get all the variables, try typing

```
. frget *, from(uscounties)
```

   This sometimes works. Other times it does not because some of the variables in uscounties already exist in the current frame. When it does not work, frget lists the variable names that exist in both frames and, even better, stores them in r(dups). Thus, if you are willing to exclude those variables, you can type

```
. frget *, from(uscounties) exclude(‘r(dups)’)
```

8. How to get all the variables 2: frget *, prefix()

   Another way to get all the variables is to type

```
. frget *, from(uscounties) prefix(c_)
```

   This brings in all the variables under their original names but prefixed with c_. The variable mvalues in the linked frame, for instance, is copied to c_mvalues.

   Another advantage of this approach is how easily you can drop the copies from the data should you desire to do so. Type

```
. drop c_*
```

You can choose your own prefix. If you prefer suffixing them, type

```
. frget *, from(uscounties) suffix(_c)
```

This names the copies `mvalues_c`, etc. These names are more like the originals, at least if you use tab completion for typing them. Type the first characters of the original name and press tab. And if you wish, you can later drop the suffixed variables just as easily as prefixed ones. Type

```
. drop *_c
```

9. How to create new variables

   Assume that the `uscounties` frame contains variables `total_income` and `population`. You need `avg_income` in the current frame.

   One solution would be

   ```
   . frget total_income population, from(uscounties)
   . generate avg_income = total_income/population
   ```

   Another solution would be to use the `frval()` function to make the calculation directly:

   ```
   . generate avg_income =
   > frval(uscounties, total_income)/frval(uscounties, population)
   ```

   Here, however, is perhaps the best solution:

   ```
   . frame uscounties: generate avg_income = total_income/population
   . frget avg_income, from(uscounties)
   ```

   It is not often that one has the opportunity to save computer time and memory. The gist of this approach is to create county-level variables in the `uscounties` frame and then use `frget` to get the ones you need.

10. `frget` copies and clones variables

    When `frget` copies variables, it also copies their display formats, variable labels, value labels, notes, and characteristics.

    The new variables are not just copies. They are clones.

Stored results

`frget` stores the following in `r()`:

Scalars

`r(k)`	number of variables copied from linked frame

Macros

`r(newlist)`	new variables in the current frame
`r(srclist)`	variables copied from linked frame
`r(excluded)`	variables not copied from linked frame
`r(dups)`	variables already present in the current frame
`r(notfound)`	variables not found in the linked frame

`r(dups)` is present only if `frget` exits with an error message because a prospective new variable name already exists in the current frame.

`r(notfound)` is present only for syntax 2 when `frget` exits with an error message because a *varname* is not found in the linked frame.

Also see

[D] **frlink** — Link frames

[D] **fralias** — Alias variables from linked frames

[D] **frames intro** — Introduction to frames

[D] **merge** — Merge datasets