

# Package ‘rata’

September 23, 2018

**Title** Manipulate Datasets Using A Stata-like Syntax

**Version** 0.0.1

**Description** This package implements tools for manipulating rectangular data sets (data sets with observations and variables) in a way that is familiar to users of a popular, but proprietary, statistical package commonly used in the social sciences.

**Depends** R (>= 3.5.1), Formula, foreign, readstata13, sandwich

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**NeedsCompilation** no

**Author** Zach Flynn [aut, cre]

**Maintainer** Zach Flynn <zlflynn@gmail.com>

## R topics documented:

collapse . . . . .	2
count . . . . .	3
describe . . . . .	3
destring . . . . .	4
do . . . . .	4
dropif . . . . .	5
dropvar . . . . .	5
forvar . . . . .	6
gen . . . . .	6
keepif . . . . .	7
keepvar . . . . .	7
listif . . . . .	8
preserve . . . . .	8
reg . . . . .	9
rename . . . . .	10
restore . . . . .	10

savedata . . . . .	11
shape . . . . .	11
tostring . . . . .	12
use . . . . .	12
varlist . . . . .	13
xtset . . . . .	13
<b>Index</b>	<b>15</b>

---

collapse	<i>collapses a data set by variables using arbitrary aggregation functions</i>
----------	--

---

**Description**

collapses a data set by variables using arbitrary aggregation functions

**Usage**

collapse(form)

**Arguments**

form                    an argument with the form  
                         ~fun1(var1)+fun2(var2)+fun3(var3)+... |byvar1+byvar2+...  
                         where fun1, fun2, and fun3 are aggregation functions like "mean", "sum", "max",  
                         etc. data will contain all unique levels of (byvar1,byvar2,...) and fun1(var1),fun2(var2)  
                         evaluated on the subset of the data set with that value of the by variables. The  
                         equivalent Stata is: collapse (fun1) var1 (fun2) var2 (fun3) var3 ..., by(byvar1  
                         byvar2)

**Examples**

```
data(Produc)
use(Produc)
listif()
collapse(~sum(emp)|year)
listif()
```

---

count	<i>Counts how many observations (optionally, satisfying a condition)</i>
-------	--

---

**Description**

Counts how many observations (optionally, satisfying a condition)

**Usage**

```
count(ifstmt = NULL)
```

**Arguments**

ifstmt	an optional argument which gives an condition that must be met for the observation to be counted
--------	--

**Examples**

```
use(cars)
count()
count("speed <= 20")
```

---

describe	<i>lists the names of the variables in the dataset</i>
----------	--

---

**Description**

lists the names of the variables in the dataset

**Usage**

```
describe(pattern = NULL)
```

**Arguments**

pattern	an optional regular expression which only returns variable names that match the expression
---------	--

**Examples**

```
use(cars)
describe()
describe("^s")
```

---

destring	<i>turn a variable with string type into a numeric value</i>
----------	--

---

**Description**

turn a variable with string type into a numeric value

**Usage**

```
destring(varlist)
```

**Arguments**

varlist            variables to convert, either in the form "var1 var2 var3" or in the form ~var1+var2+var3.

---

do	<i>Executes code on the dataset</i>
----	-------------------------------------

---

**Description**

Executes an R expression using variables from the dataset.

**Usage**

```
do(expr)
```

**Arguments**

expr            an R expression which can use any of the variable names in the current dataset

**Examples**

```
use(cars)
do({coef(lm(speed~dist))})
```

---

dropif	<i>drops rows from the dataset</i>
--------	------------------------------------

---

**Description**

drops rows from the dataset

**Usage**

```
dropif(x)
```

**Arguments**

x	a condition like (ex: "var1==2") describing the observations that should be removed from the data set.
---	--

**Examples**

```
use(cars)
listif()
dropif("speed <= 20")
listif()
```

---

dropvar	<i>drops variables in varlist format from the dataset</i>
---------	---

---

**Description**

drops variables in varlist format from the dataset

**Usage**

```
dropvar(x)
```

**Arguments**

x	a varlist either in "var1 var2 var3" format or ~var1+var2+var3 format.
---	--

**Examples**

```
use(cars)
listif()
dropvar("speed")
listif()
use(cars)
dropvar(~speed)
listif()
```

---

forvar	<i>apply a function to each of a list of variables</i>
--------	--

---

### Description

apply a function to each of a list of variables

### Usage

```
forvar(varlist, action, macro = "%var")
```

### Arguments

varlist	a list of variables in the format ~var1+var2+var3+... or as a vector of names like "var1 var2 var3".
action	a quoted expression to apply to each variable where the variable is represented in the expression by macro.
macro	an expression that will be replaced in action for each variable, by default %var.

### Examples

```
use(cars)
forvar("speed dist", "gen('%var2', '%var^2')")
listif()
```

---

gen	<i>generates a new variable that is a transformation of existing variables in the dataset or replaces one</i>
-----	---

---

### Description

generates a new variable that is a transformation of existing variables in the dataset or replaces one

### Usage

```
gen(var, value, byvar = NULL, subset = NULL, replace = FALSE)
```

### Arguments

var	the name of the variable to be generated
value	the transformation of the dataset to replace the "newvar" in option form with. For example, value="sum(wage*female)" to get a variable which has total female wages. In Stata, the same command would be: "egen femalewage = total(wage*female)".

byvar	apply the value for each level of the by variables, specified either as a formula, like ~byvar1+byvar2+... or as a varlist "byvar1 byvar2 byvar3...".
subset	only generate values if the condition provided in subset is true. Make sure to enclose the expression in quotes, like so: subset="female==1 & highschool==1" to generate the values only for women who graduated from highschool. This option is used like the "if" in Stata.
replace	either TRUE or FALSE. If FALSE (default), the code refuses to alter the variable if the variable already exists. Otherwise, if replace=TRUE, then the values will be replaced.

---

keepif	<i>keeps some rows in the dataset and drops the rest</i>
--------	--

---

### Description

keeps some rows in the dataset and drops the rest

### Usage

```
keepif(x)
```

### Arguments

x	a condition like: "var1==2" in which case observations that satisfy the condition are kept and all others are removed.
---	--

### Examples

```
use(cars)
keepif("speed <= 20")
listif()
```

---

keepvar	<i>keeps some variables in the dataset and drops the others</i>
---------	---

---

### Description

keeps some variables in the dataset and drops the others

### Usage

```
keepvar(x)
```

### Arguments

x	a varlist either of the form "var1 var2 var3" or in the form ~var1+var2+var3.
---	---

**Examples**

```
use(cars)
keepvar("speed")
listif()
use(cars)
keepvar(~speed)
listif()
```

---

listif	<i>prints the part of the dataset that satisfies certain conditions</i>
--------	---

---

**Description**

prints the part of the dataset that satisfies certain conditions

**Usage**

```
listif(cond = NULL, ...)
```

**Value**

the part of the dataset that satisfies the condition and contains the specified columns

---

preserve	<i>preserve a data set before modification</i>
----------	--

---

**Description**

preserve a data set before modification

**Usage**

```
preserve(data = NULL)
```

**Arguments**

data	a data set to preserve
------	------------------------

**Value**

a value that can be passed to `restore` to restore the data set later



**Examples**

```
require(stats)
use(cars)
p <- preserve()
collapse(~mean(dist)|speed)
list()
restore(p)
list()
```

---

reg	<i>regress y on x with robust standard errors, clustered standard errors, HAC standard errors, panel fixed effects, etc</i>
-----	---

---

**Description**

regress y on x with robust standard errors, clustered standard errors, HAC standard errors, panel fixed effects, etc

**Usage**

```
reg(y, x, subset = NULL, effect = NULL, robust = TRUE, hac = NULL,
    cluster = NULL)
```

**Arguments**

y	name of the dependent variable
x	names of the independent variables in either "x1 x2 x3" format or ~x1+x2+x3 format.
subset	conditions to subset the data
effect	either "twoway", "obs", or "time" for fixed effects,
robust	whether to use robust standard errors
hac	which kernel to use for heteroskedastic and auto correlation standard errors
cluster	the name of the variable to use for clustered standard errors

**Value**

b coefficient vector  
 V covariance matrix of coefficients

---

rename	<i>renames variables in the dataset</i>
--------	---

---

**Description**

renames variables in the dataset

**Usage**

```
rename(var, newvar)
```

**Arguments**

var	the name of the variable to rename
newvar	the new name of the variable

**Examples**

```
use(cars)
listif()
rename("speed", "velocity")
listif()
```

---

restore	<i>restore a dataset from a previous preserve to be currently used</i>
---------	--

---

**Description**

restore a dataset from a previous preserve to be currently used

**Usage**

```
restore(envir)
```

**Arguments**

envir	a previous preserve value.
-------	----------------------------

**Value**

the preserved data set

**Examples**

```
require(stats)
use(cars)
p <- preserve()
collapse(~mean(dist)|speed)
list()
restore(p)
list()
```

---

savedata	<i>saves data to a CSV file</i>
----------	---------------------------------

---

**Description**

saves data to a CSV file

**Usage**

```
savedata(file)
```

**Arguments**

file	a file name to save the current data to
------	---

**Examples**

```
use(Produc)
savedata("Produc.csv")
```

---

shape	<i>reshapes a data set from wide to long or from long to wide formats</i>
-------	---

---

**Description**

reshapes a data set from wide to long or from long to wide formats

**Usage**

```
shape(form, direction = "long")
```

**Arguments**

form	<p>if direction="long", then the argument should have the form:  id1+id2+...~newvar stub</p> <p>where there are variables in the data set named "stubXXXX" and "newvar" is the name of the new variable that will be added to the data set which will contain the various values of "stubXXXX" on exit. The variable "stub" on exit will contain the value of "XXXX". Variables (id1,id2,...) will also be included in the dataset on exit. The command behaves like "reshape long stub, i(id1 id2 ...) j(newvar)" in Stata.</p> <p>If direction="wide", then the argument should have the form,  id1+id2+...~values1+values2+... byvar1+byvar2+...</p> <p>The variables (id1,id2,...,byvar1,byvar2,...) should uniquely identify observations in the data. On exit the dataset will contain (id1,id2,...) in addition to values1byvar1.byvar2, values2byvar1.byvar2, ... for each unique value of (byvar1,byvar2,...). The command behaves like "reshape wide values1 values2 ..., i(id1 id2 ...) j(byvar1...)"</p>
direction	either "long" or "wide" to indicate the direction to reorient the data set

---

tostring	<i>turn a variable of another type into a string variable</i>
----------	---

---

**Description**

turn a variable of another type into a string variable

**Usage**

tostring(varlist)

**Arguments**

varlist	variables to convert, either in the form "var1 var2 var3" or in the form ~var1+var2+var3.
---------	---

---

use	<i>uses a dataset, marking it as the active dataset</i>
-----	---

---

**Description**

uses a dataset, marking it as the active dataset

**Usage**

use(x, ...)

**Arguments**

`x` usually either a `data.frame` or a `csv/dta` filename to be imported. An R function which returns a `data.frame` can also be specified.

**Examples**

```
Produc
use(Produc)
savedata("Produc.csv")
listif()
dropvar(".*")
```

---

<code>varlist</code>	<i>creates a formula object from a varlist, mostly for internal use.</i>
----------------------	--

---

**Description**

creates a formula object from a varlist, mostly for internal use.

**Usage**

```
varlist(x)
```

**Arguments**

`x` the varlist to be converted in "var1 var2 var3" format. Will eventually work with globbing so that "var\*" can be used to refer to all variables that begin with var

**Value**

a formula object which can be passed to `model.frame`

---

<code>xtset</code>	<i>prepares a panel or time series dataset for lag operations</i>
--------------------	---

---

**Description**

prepares a panel or time series dataset for lag operations

**Usage**

```
xtset(timevar = NULL, obsvar = NULL)
```

**Arguments**

`timevar` the name of the variable to for the time dimension  
`obsvar` the name of the variable to use for the observation dimension

**Examples**

```
use(Produc)
xtset("year", "state")
reg("emp", "unemp", effect="twoway")
reg("emp", "unemp", effect="obs")
reg("emp", "unemp", effect="time")
```

# Index

`collapse`, [2](#)  
`count`, [3](#)

`describe`, [3](#)  
`destring`, [4](#)  
`do`, [4](#)  
`dropif`, [5](#)  
`dropvar`, [5](#)

`forvar`, [6](#)

`gen`, [6](#)

`keepif`, [7](#)  
`keepvar`, [7](#)

`listif`, [8](#)

`preserve`, [8](#)

`reg`, [9](#)  
`rename`, [10](#)  
`restore`, [10](#)

`savedata`, [11](#)  
`shape`, [11](#)

`tostring`, [12](#)

`use`, [12](#)

`varlist`, [13](#)

`xtset`, [13](#)