Package 'recoder'

October 14, 2022

Type Package

Title A Simple and Flexible Recoder

Description Simple, easy to use, and flexible functionality for recoding variables. It allows for simple piecewise definition of transformations.

Version 0.1

Depends R (>= 3.1), stringr (>= 1.0.0)

Date 2015-07-08

Author Ali Sanaei

Maintainer Ali Sanaei <ali@ischool.berkeley.edu>

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-10 20:33:14

R topics documented:

`recoder-package` *A Simple and Flexible Recoder*

Description

Simple, easy to use, and quite flexible function for recoding variables. Among the features is that it can handle compound conditions and can include functional transformations on the go, which together allow for rapid piecewise definition of functions.

Details

Package:	recoder
Type:	Package
Version:	1.0
Date:	2015-07-08
License:	GPL (>= 2)

Author(s)

Ali Sanaei

`recoder` *Simple and Handy Recoder*

Description

A flexible function for recoding a variable. It can be used for anything from simple explicit recoding to piecewise-defined transformations.

Usage

```
recoder(var, recode, other = NULL, na = NA, as.what = NULL)
```

Arguments

`var`	`var` is a variable (vector) being recoded.
`recode`	`recode` contains the instructions for how the recoding is done. Each instruction is in the form of `condition: newvalue` and instrunctions must be separated by semicolons (`;`).
`other`	When `other=NULL`, elements that do not satisfy any of the coditions in the instructions given by `recode` will by defaul remain unchanger, otherwise they will be changed to `other`
`na`	When `na=NULL`, `NA` values will not be changed in the vector, otherwise, they will be changed to `na`.
`as.what`	if `as.what` is specified, the final vector is forced to be of the given type. For example, it will be made a factor if `as.what= "factor"` is specified.

Details

recode contains the instructions for how the recoding is done. Each instruction is in the form of condition: newvalue and instrunctions must be separated by semicolons (;). E.g., 1:2; 2:-1; 3:0 will change all ones to two, twos to negative one, and threes to zero. Instruction can include compound conditions and functions for newvalue expressed in terms of $. For example, "<0:0; >2:4; >=0 & <=2 : $^2" will give zero for all the values below 0, 4 for all the values above 2, and squared values for anything between 0 and 2.

The input can be other types beside numeric. For characters and factors, it is important that the form "old":"new" includes quotations around both the left-hand side and the right-hand side.

When the input is a factor, recoding works as expected, but new levels cannot be assigned. In order to assign new levels, the input vector should be made into something else (e.g., a character) and then the argument as.what could be set to "factor" to get a factor as the end result.

Value

The recoded vector is returned.

Author(s)

Ali Sanaei

See Also

cut, recode

Examples

```
u = rnorm(1000)
u.2 = recoder(u,'<-2: -2; >2 : 2' ,other = 0)
plot(u,u.2)

# use a function of 'x' for assigning new values based on old values
u.3 = recoder(u,'<-2: -2; >2 : 2; >=-2 & <=2 : $^3/4')
# NOTE: instructions in recode are executed from left to right
# So u.3 and u.4 are the same
u.4 = recoder(u,'<-2: -2; >=-2 : $^3/4;  >2 : 2')
plot(u, u.3)

# another example of a piecewise defined function
x = runif(1000,-1,10)
y = recoder(x,'>0 & <1: sqrt($); >=1 : 1+log($)' ,other=NA)
plot(y~x)
table(is.na(y))

# we can also create other types of variables
k = rpois(n = 10000, lambda = 3)
evodd = recoder(k %% 2, '0: "even"; 1: "odd" ')
table(evodd)

# the output could also be made a factor, etc., using "as.what" argument
```

```
evodd.factor = recoder(k %% 2, '0: "even"; 1: "odd" ', as.what='factor')

# characters are recoded similarly
str = c('a','b','x','z',NA)
str2 = recoder(str, '"x":"c"; "z":"d" ')
str3 = recoder(str, '"a":1; "b":2; "x":3; "z":4 ')
print(str3) #note that this is still not numeric
num3 = recoder(str, '"a":1; "b":2; "x":3; "z":4 ', as.what='numeric')
print(num3)

# when input is a factor, the levels are fixed
f1 = factor( c('a','b','x','z',NA)  )
f2 = recoder( f1, '"x":"a"; "z":"b" ')

## Not run:
f3 = recoder(f1, '"x":"c"; "z":"d" ') # "c" and "d" are not recognized

## End(Not run)
f3 = recoder( as.character(f1), '"x":"c"; "z":"d" ' , as.what= "factor")
```

Index