Package 'BMisc'

January 26, 2022

Title Miscellaneous Functions for Panel Data, Quantiles, and Printing Results

Version 1.4.4

Description These are miscellaneous functions for working with panel data, quantiles, and printing results. For panel data, the package includes functions for making a panel data balanced (that is, dropping missing individuals that have missing observations in any time period), converting id numbers to row numbers, and to treat repeated cross sections as panel data under the assumption of rank invariance. For quantiles, there are functions to make distribution functions from a set of data points (this is particularly useful when a distribution function is created in several steps), to combine distribution functions based on some external weights, and to invert distribution functions. Finally, there are several other miscellaneous functions for obtaining weighted means, weighted distribution functions, and weighted quantiles; to generate summary statistics and their differences for two groups; and to add or drop covariates from formulas.

Depends R (>= 2.1.0)

Imports data.table, Rcpp

License GPL-2

Suggests testthat (>= 3.0.0), plm, tibble

Encoding UTF-8

RoxygenNote 7.1.2

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Brantly Callaway [aut, cre]

Maintainer Brantly Callaway <brantly.callaway@uga.edu>

Repository CRAN

Date/Publication 2022-01-26 22:00:05 UTC

R topics documented:

`addCovToFormla` *Add a Covariate to a Formula*

Description

`addCovFromFormla` adds some covariates to a formula; covs should be a list of variable names

Usage

```
addCovToFormla(covs, formla)
```

Arguments

`covs`	should be a list of variable names
`formla`	which formula to add covariates to

Value

formula

Examples

```
formla <- y ~ x
addCovToFormla(list("w","z"), formla)

formla <- ~x
addCovToFormla("z", formla)
```

`blockBootSample` *Block Bootstrap*

Description

make draws of all observations with the same id in a panel data context. This is useful for bootstrapping with panel data.

Usage

```
blockBootSample(data, idname)
```

Arguments

`data`	data.frame from which you want to bootstrap
`idname`	column in data which contains an individual identifier

Value

data.frame bootstrapped from the original dataset; this data.frame will contain new ids

Examples

```
data(LaborSupply, package="plm")
bbs <- blockBootSample(LaborSupply, "id")
nrow(bbs)
head(bbs$id)
```

`BMisc` *BMisc*

Description

A set of miscellaneous helper functions

checkfun *Check Function*

Description

The check function used for optimizing to get quantiles

Usage

```
checkfun(a, tau)
```

Arguments

a	vector to compute quantiles for
tau	between 0 and 1, ex. .5 implies get the median

Value

numeric value

Examples

```
x <- rnorm(100)
x[which.min(checkfun(x, 0.5))] ##should be around 0
```

combineDfs *Combine Two Distribution Functions*

Description

Combines two distribution functions with given weights by pstrat

Usage

```
combineDfs(y.seq, dflist, pstrat = NULL, ...)
```

Arguments

y.seq	sequence of possible y values
dflist	list of distribution functions to combine
pstrat	a vector of weights to put on each distribution function; if weights are not provided then equal weight is given to each distribution function
...	additional arguments that can be past to BMisc::makeDist

Value

ecdf

Examples

```
x <- rnorm(100)
y <- rnorm(100,1,1)
Fx <- ecdf(x)
Fy <- ecdf(y)
both <- combineDfs(seq(-2,3,0.1), list(Fx,Fy))
plot(Fx, col="green")
plot(Fy, col="blue", add=TRUE)
plot(both, add=TRUE)
```

---

compareBinary *Compare Variables across Groups*

Description

compareBinary takes in a variable e.g. union and runs bivariate regression of x on treatment (for summary statistics)

Usage

```
compareBinary(
  x,
  on,
  dta,
  w = rep(1, nrow(dta)),
  report = c("diff", "levels", "both")
)
```

Arguments

x	variables to run regression on
on	binary variable
dta	the data to use
w	weights
report	which type of report to make; diff is the difference between the two variables by group

Value

matrix of results

cs2panel *Cross Section to Panel*

Description

Turn repeated cross sections data into panel data by imposing rank invariance; does not require that the inputs have the same length

Usage

```
cs2panel(cs1, cs2, yname)
```

Arguments

cs1	data frame, the first cross section
cs2	data frame, the second cross section
yname	the name of the variable to calculate difference for (should be the same in each dataset)

Value

the change in outcomes over time

dropCovFromFormla *Drop a Covariate from a Formula*

Description

dropCovFromFormla adds drops some covariates from a formula; covs should be a list of variable names

Usage

```
dropCovFromFormla(covs, formla)
```

Arguments

covs	should be a list of variable names
formla	which formula to drop covariates from

Value

formula

Examples

```
formla <- y ~ x + w + z
dropCovFromFormla(list("w","z"), formla)

dropCovFromFormla("z", formla)
```

`getListElement` *Return Particular Element from Each Element in a List*

Description

a function to take a list and get a particular part out of each element in the list

Usage

```
getListElement(listolists, whichone = 1)
```

Arguments

`listolists`	a list
`whichone`	which item to get out of each list (can be numeric or name)

Value

list of all the elements 'whichone' from each list

Examples

```
len <- 100 # number elements in list
lis <- lapply(1:len, function(l) list(x=(-l), y=l^2) ) # create list
getListElement(lis, "x")[1] # should be equal to -1
getListElement(lis, 1)[1] # should be equal to -1
```

`getWeightedDf` *Weighted Distribution Function*

Description

Get a distribution function from a vector of values after applying some weights

Usage

```
getWeightedDf(y, y.seq = NULL, weights = NULL, norm = TRUE)
```

Arguments

y	a vector to compute the mean for
y.seq	an optional vector of values to compute the distribution function for; the default is to use all unique values of y
weights	the vector of weights, can be NULL, then will just return mean
norm	normalize the weights so that they have mean of 1, default is to normalize

Value

ecdf

---

getWeightedMean *Weighted Mean*

Description

Get the mean applying some weights

Usage

```
getWeightedMean(y, weights = NULL, norm = TRUE)
```

Arguments

y	a vector to compute the mean for
weights	the vector of weights, can be NULL, then will just return mean
norm	normalize the weights so that they have mean of 1, default is to normalize

Value

the weighted mean

---

getWeightedQuantiles *Get Weighted Quantiles*

Description

Finds multiple quantiles by repeatedly calling getWeightedQuantile

Usage

```
getWeightedQuantiles(tau, cvec, weights = NULL, norm = TRUE)
```

Arguments

tau	a vector of values between 0 and 1
cvec	a vector to compute quantiles for
weights	the weights, weighted.checkfun normalizes the weights to sum to 1.
norm	normalize the weights so that they have mean of 1, default is to normalize

Value

vector of quantiles

---

ids2rownum *Convert Vector of ids into Vector of Row Numbers*

Description

ids2rownum takes a vector of ids and converts it to the right row number in the dataset; ids should be unique in the dataset that is, don't pass the function panel data with multiple same ids

Usage

```
ids2rownum(ids, data, idname)
```

Arguments

ids	vector of ids
data	data frame
idname	unique id

Value

vector of row numbers

Examples

```
ids <- seq(1,1000,length.out=100)
ids <- ids[order(runif(100))]
df <- data.frame(id=ids)
ids2rownum(df$id, df, "id")
```

invertEcdf *Invert Ecdf*

Description

take an ecdf object and invert it to get a step-quantile function

Usage

```
invertEcdf(df)
```

Arguments

df	an ecdf object

Value

stepfun object that contains the quantiles of the df

lhs.vars *Left-hand Side Variables*

Description

Take a formula and return a vector of the variables on the left hand side, it will return NULL for a one sided formula

Usage

```
lhs.vars(formla)
```

Arguments

formla	a formula

Value

vector of variable names

Examples

```
ff <- yvar ~ x1 + x2
lhs.vars(ff)
```

makeBalancedPanel *Balance a Panel Data Set*

Description

This function drops observations from data.frame that are not part of balanced panel data set.

Usage

```
makeBalancedPanel(data, idname, tname)
```

Arguments

data	data.frame used in function
idname	unique id
tname	time period name

Value

data.frame that is a balanced panel

Examples

```
id <- rep(seq(1,100), each = 2) ## individual ids for setting up a two period panel
t <- rep(seq(1,2),100) ## time periods
y <- rnorm(200) ## outcomes
dta <- data.frame(id=id, t=t, y=y) ## make into data frame
dta <- dta[-7,] ## drop the 7th row from the dataset (which creates an unbalanced panel)
dta <- makeBalancedPanel(dta, idname="id", tname="t")
```

makeDist *Make a Distribution Function*

Description

turn vectors of a values and their distribution function values into an ecdf. Vectors should be the same length and both increasing.

Usage

```
makeDist(
  x,
  Fx,
  sorted = FALSE,
  rearrange = FALSE,
  force01 = FALSE,
  method = "constant"
)
```

Arguments

x	vector of values
Fx	vector of the distribution function values
sorted	boolean indicating whether or not x is already sorted; computation is somewhat faster if already sorted
rearrange	boolean indicating whether or not should monotize distribution function
force01	boolean indicating whether or not to force the values of the distribution function (i.e. Fx) to be between 0 and 1
method	which method to pass to approxfun to approximate the distribution function. Default is "constant"; other possible choice is "linear". "constant" returns a step function, just like an empirical cdf; "linear" linearly interpolates between neighboring points.

Value

ecdf

Examples

```
y <- rnorm(100)
y <- y[order(y)]
u <- runif(100)
u <- u[order(u)]
F <- makeDist(y,u)
```

---

multiplier_bootstrap *multiplier_bootstrap*

Description

A function that takes in an influence function (an nxk matrix) and the number of bootstrap iterations and returns a Bxk matrix of bootstrap results. This function uses Rademechar weights.

Usage

```
multiplier_bootstrap(inf_func, biters)
```

Arguments

inf_func	nxk matrix of (e.g., these could be a matrix containing the influence function for different parameter estimates)
biters	the number of bootstrap iterations

Value

a Bxk matrix

panel2cs *Panel Data to Repeated Cross Sections*

Description

panel2cs takes a 2 period dataset and turns it into a cross sectional dataset. The data includes the change in time varying variables between the time periods. The default functionality is to keep all the variables from period 1 and add all the variables listed by name in timevars from period 2 to those.

Usage

```
panel2cs(data, timevars, idname, tname)
```

Arguments

data	data.frame used in function
timevars	vector of names of variables to keep
idname	unique id
tname	time period name

Value

data.frame

panel2cs2 *Panel Data to Repeated Cross Sections*

Description

panel2cs2 takes a 2 period dataset and turns it into a cross sectional dataset; i.e., long to wide. This function considers a particular case where there is some outcome whose value can change over time. It returns the dataset from the first period with the outcome in the second period and the change in outcomes over time appended to it

Usage

```
panel2cs2(data, yname, idname, tname, balance_panel = TRUE)
```

Arguments

data	data.frame used in function
yname	name of outcome variable that can change over time
idname	unique id
tname	time period name
balance_panel	whether to ensure that panel is balanced. Default is TRUE, but code runs somewhat faster if this is set to be FALSE.

Value

data from first period with .y0 (outcome in first period), .y1 (outcome in second period), and .dy (change in outcomes over time) appended to it

---

rhs *Right-hand Side of Formula*

Description

Take a formula and return the right hand side of the formula

Usage

```
rhs(formla)
```

Arguments

formla	a formula

Value

a one sided formula

Examples

```
ff <- yvar ~ x1 + x2
rhs(ff)
```

---

rhs.vars *Right-hand Side Variables*

Description

Take a formula and return a vector of the variables on the right hand side

Usage

```
rhs.vars(formla)
```

Arguments

formla	a formula

Value

vector of variable names

Examples

```
ff <- yvar ~ x1 + x2
rhs.vars(ff)

ff <- y ~ x1 + I(x1^2)
rhs.vars(ff)
```

`source_all` *source_all*

Description

Source all the files in a folder

Usage

```
source_all(fldr)
```

Arguments

`fldr`	path to a folder

`subsample` *Subsample of Observations from Panel Data*

Description

returns a subsample of a panel data set; in particular drops all observations that are not in `keepids`. If it is not set, randomly keeps `nkeep` observations.

Usage

```
subsample(dta, idname, tname, keepids = NULL, nkeep = NULL)
```

Arguments

`dta`	a data.frame which is a balanced panel
`idname`	the name of the id variable
`tname`	the name of the time variable
`keepids`	which ids to keep
`nkeep`	how many ids to keep (only used if `keepids` is not set); the default is the number of unique ids

Value

a data.frame that contains a subsample of `dta`

Examples

```
data(LaborSupply, package="plm")
nrow(LaborSupply)
unique(LaborSupply$year)
ss <- subsample(LaborSupply, "id", "year", nkeep=100)
nrow(ss)
```

`toformula` *Variable Names to Formula*

Description

take a name for a y variable and a vector of names for x variables and turn them into a formula

Usage

```
toformula(yname, xnames)
```

Arguments

`yname`	the name of the y variable
`xnames`	vector of names for x variables

Value

a formula

Examples

```
toformula("yvar", c("x1","x2"))

## should return yvar ~ 1
toformula("yvar", rhs.vars(~1))
```

TorF *TorF*

Description

A function to replace NA's with FALSE in vector of logicals

Usage

```
TorF(cond, use_isTRUE = FALSE)
```

Arguments

cond	a vector of conditions to check
use_isTRUE	whether or not to use a vectorized version of isTRUE. This is generally slower but covers more cases.

Value

logical vector

weighted.checkfun *Weighted Check Function*

Description

Weights the check function

Usage

```
weighted.checkfun(q, cvec, tau, weights)
```

Arguments

q	the value to check
cvec	vector of data to compute quantiles for
tau	between 0 and 1, ex. .5 implies get the median
weights	the weights, weighted.checkfun normalizes the weights to sum to 1.

Value

numeric

Index