Package 'AcrossTic'

October 12, 2022

Version 1.0-3

Date 2016-08-12

Title A Cost-Minimal Regular Spanning Subgraph with TreeClust

Author Dave Ruth, Sam Buttrey

Maintainer Sam Buttrey <buttrey@nps.edu>

Depends treeClust (>= 1.1-6), lpSolve

Description Construct minimum-cost regular spanning subgraph as part of a non-parametric two-sample test for equality of distribution.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2016-08-13 11:01:25

R topics documented:

`AcrossTic-package` *A Cost-Minimal Regular Spanning Subgraph with TreeClust*

Description

Construct minimum-cost regular spanning subgraph as part of a non-parametric two-sample test for equality of distribution.

Details

The DESCRIPTION file:

Package:	AcrossTic
Version:	1.0-3
Date:	2016-08-12
Title:	A Cost-Minimal Regular Spanning Subgraph with TreeClust
Author:	Dave Ruth, Sam Buttrey
Maintainer:	Sam Buttrey <buttrey@nps.edu>
Depends:	treeClust (>= 1.1-6), lpSolve
Description:	Construct minimum-cost regular spanning subgraph as part of a non-parametric two-sample test for equality of
License:	GPL (>= 2)

Index of help topics:

```
rRegMatch Compute r-regular matching spanning trees
print.AcrossTic print an AcrossTic object, the output from rRegMatch
plot.AcrossTic print an AcrossTic object, the output from rRegMatch
ptest perform permutation test on AcrossTic object
print.AcrossTicPtest print an AcrossTic permutation test object
```

This primarily provides rRegMatch, which for arguments X and r produces a minimum-distance r-regular subgraph of the rows of X.

Author(s)

Dave Ruth, Sam Buttrey

Maintainer: Sam Buttrey <buttrey@nps.edu>

References

David Ruth, "A new multivariate two-sample test using regular minimum-weight spanning subgraphs," J. Stat. Distributions and Applications (2014)

Examples

```
set.seed (123)
X <- matrix (rnorm (100), 50, 2) # Create data...
y <- rep (c (1, 2), each=25) # ...and class membership
## Not run: rRegMatch (X, r = 3, y = y)
```

plot.AcrossTic *Plot function for simple AcrossTic objects*

Description

Plot an object of class AcrossTic (see details). Currently intended for two-class objects built with two-dimensional Xs.

Usage

```
## S3 method for class 'AcrossTic'
 plot(x, X.values, y, grp.cols = c(2, 4), grp.pch = c(16, 17), ...)
```

Arguments

x	AcrossTic object, normally the output from rRegMatch.
X.values	Matrix of data. If not spplied the function looks in x.
y	Vector with two distinct values giving the label for each observation.
grp.cols	Colors for the two groups. Default: 2 and 4.
grp.pch	Plotting points for the two groups. Default: 16 and 17.
...	Other arguments, passed on to plot.

Details

This demonstrates a graph of the matching of the rRegMatch type. Points are plotted in 2d; then within-group matches are shown with dotted lines and between-group pairings with solid ones. If X has more than two columns, the first two are used, with a warning. If Y is supplied it will be used; if not, it will be extracted from x; if no y is found, an error is issued. Y must have exactly two distinct values.

Value

No output. Side effect: a plot is produced.

Author(s)

David Ruth and Sam Buttrey

Examples

```
set.seed (123)
X <- matrix (rnorm (100), 50, 2) # Create data...
y <- rep (c (0, 1), each=25) # ...and class membership
plot (rRegMatch (X, r = 3, y = y))
```

print.AcrossTic *Print method for AcrossTic objects*

Description

Print some attributes of an AcrossTic object to the screen.

Usage

```
## S3 method for class 'AcrossTic'
 print(x, ...)
```

Arguments

x	AcrossTic item (output from rRegMatch) to be printed
...	Other arguments, currently ignored.

Value

None.

Author(s)

Sam Buttrey

print.AcrossTicPtest *Print output of AcrossTic permutation test*

Description

Print the output of a permutation test on an AcrossTic object (see ptest)

Usage

```
## S3 method for class 'AcrossTicPtest'
 print(x, ...)
```

Arguments

x	Object of class AcrossTicPtest
...	Other arguments, currently ignored.

Details

The output from ptest has class AcrossTicPtest. This function prints such an object.

Value

None

Author(s)

Sam Buttrey

`ptest` *Permutation test for AcrossTic objects*

Description

This function permutes the "y" entries in an AcrossTic object and computes the cross-count statistic for each permutation. This generates a null distribution suitable for use in a permutation test.

Usage

```
ptest(acobj, y, edge.weights, n = 1000)
```

Arguments

`acobj`	Object of class AcrossTic, output from `rRegMatch`.
`y`	Character, factor or logical indicating class membership for each observation. Normally this will be found inside `acobj`.
`edge.weights`	Vector of weights associated with each match. If omitted, the default is a vector of 1's of the proper length, unless the "acobj" object was computed with partial matching, in which case omitting edge.weights produces an error.
`n`	Integer, number of simulations. Default, 1000.

Details

This function permutes the y's `n` times and computes the cross-count-match statistic. If the observed value in the `acobj` is generally smaller than the permuted values, we conclude the distributions of the classes are different.

Value

A list with class AcrossTicPtest and three components:

`sims`	Vector of `n` cross-count values computed under permutation
`observed`	Observed cross-count statistic
`p.value`	P-value for test

Author(s)

Sam Buttrey and Dave Ruth

See Also

rRegMatch

Examples

```
set.seed (123)
X <- matrix (rnorm (100), 50, 2) # Create data...
y <- rep (c ("One", "Two"), each=25) # ...and class membership
## Not run: ptest (rRegMatch (X, r = 3, y = y)) # p = .479
X[1:25,] <- X[1:25,] + 1
## Not run: ptest (rRegMatch (X, r = 3, y = y)) # p = .037
```

---

rRegMatch *Regular matching with minimum-cost spanning subgraphs*

Description

This function matches each observation in X to r others so as to minimize the total distance across all matches. Optionally it computes the cross-count statistic – the number of matches associated with two observations from different classes.

Usage

```
rRegMatch(X, r, y = NULL, dister = "daisy", dist.args = list(), keep.X = nrow(X) < 100,
    keep.D = (dister == "treeClust.dist"), relax = (N >= 100), thresh = 1e-6)
```

Arguments

X	Matrix or data frame of data, or inter-point distances represented in an object inheriting from "dist"
r	Integer number of matches. The matching is "regular" in that every observation is matched to exactly r others (or, if relax=TRUE, every observation is matched to others with weights in [0, 1] that add up to r).
y	Vector of class membership indices. This is used to compute the cross-count statistic. Optional.
dister	Function to compute inter-point distances. This must take as its first argument a matrix of data argument name x. Default: daisy. If all the columns are numeric, this produces unweighted Euclidean distance by default.
dist.args	List of argument to the dister function.
keep.X	If TRUE, and X was supplied, keep the X matrix in the output object. Default: TRUE if X was supplied and also nrow (X) < 100.
keep.D	If TRUE, keep the distance object in the output. Default: TRUE if the treeClust.dist function is being used to compute the distances (since in that case the distances are random).

relax	If FALSE, solve the exact problem where each observation gets exactly r non-zero pairings, each with weight 1. If TRUE, solve the relaxed problem, where each observation has at least r non-zero pairings, each with its own weight between 0 and 1, the weights adding up to r. The exact problem gets very slow with large samples.
thresh	Weights smaller than this are considered to be exactly zero. Default: 1e-6.

Details

This function solves an optimization problem to extract the set of pairings which make the total weight (distance) associated with all pairings a minimum, subject to the constraint that every observation is paired to r others (or to enough others to have a total pair-weight of r).

Value

A list of class AcrossTic, with elements:

matches	A two-column matrix, each row gving the indices of one matched pair.
total.dist	total distance across all matches – the optimal value from the optimization problem.
status	Status of result – if the optimum was found, a vector of length 1 with name "TM_OPTIMAL_SOLUTION_FOUND" and value 0.
time.required	Time taken to run the optimization, as reported by `system.time()`.
call	The call made to the function, from `match.call`.
r	The value of r, as supplied at the time of the call.
dister	The value of dister, as supplied at the time of the call.
dist.args	The value of dist.args, as supplied at the time of the call.
X.supplied	Logical indicating whether X was supplied.
X	X matrix, if it was available and asked to be kept
y	y vector, as supplied
edge.weights	vector, of length `nrow(matches)`, giving the distances for each match. For the exact problem (`relax = FALSE`), each value is equal to 0 or 1. For the relaxed problem (`relax = TRUE`), each value is between 0 and 1, with values summing to `(r * nrow(X) / 2)`.
cross.sum	Sum of matcher.costs across all matches
cross.count	Number of matches between two observations of different classes, possibly weighted
nrow.X, ncol.X	dimension of X matrix

Author(s)

David Ruth and Sam Buttrey

References

David Ruth, "A new multivariate two-sample test using regular minimum-weight spanning subgraphs," J. Stat. Distributions and Applications (2014)

Examples

```
set.seed (123)
X <- matrix (rnorm (100), 50, 2) # Create data...
y <- rep (c (1, 2), each=25) # ...and class membership
rRegMatch (X, r = 3, y = y)
## Not run: plot (rRegMatch (X, r = 3, y = y)) # to see picture
```

Index