Package 'proxy'

July 8, 2015

Type Package

Title Distance and Similarity Measures

Version 0.4-15

Description Provides an extensible framework for the efficient calculation of auto- and cross-proximities, along with implementations of the most popular ones.

Depends R (>= 2.4.0)

Imports stats, utils

Suggests cba

Collate registry.R database.R dist.R similarities.R dissimilarities.R util.R seal.R

License GPL-2

NeedsCompilation yes

Author David Meyer [aut, cre], Christian Buchta [aut]

Maintainer David Meyer <David.Meyer@R-project.org>

Repository CRAN

Date/Publication 2015-07-08 17:11:02

R topics documented:

dist *Matrix Distance/Similarity Computation*

Description

These functions compute and return the auto-distance/similarity matrix between either rows or columns of a matrix/data frame, or a list, as well as the cross-distance matrix between two matrices/data frames/lists.

Usage

```
dist(x, y = NULL, method = NULL, ..., diag = FALSE, upper = FALSE,
     pairwise = FALSE, by_rows = TRUE, convert_similarities = TRUE,
     auto_convert_data_frames = TRUE)
simil(x, y = NULL, method = NULL, ..., diag = FALSE, upper = FALSE,
      pairwise = FALSE, by_rows = TRUE, convert_distances = TRUE,
      auto_convert_data_frames = TRUE)

pr_dist2simil(x)
pr_simil2dist(x)

as.dist(x, FUN = NULL)
as.simil(x, FUN = NULL)

## S3 method for class 'dist'
as.matrix(x, diag = 0, ...)
## S3 method for class 'simil'
as.matrix(x, diag = NA, ...)
```

Arguments

x	For dist and simil, a numeric matrix object, a data frame, or a list. A vector will be converted into a column matrix. For as.simil and as.dist, an object of class dist and simil, respectively, or a numeric matrix. For pr_dist2simil and pr_simil2dist, any numeric vector.
y	NULL, or a similar object than x
method	a function, a registry entry, or a mnemonic string referencing the proximity measure. A list of all available measures can be obtained using pr_DB (see examples). The default for dist is "Euclidean", and for simil "correlation".
diag	logical value indicating whether the diagonal of the distance/similarity matrix should be printed by print.dist/print.simil. In the context of as.matrix the value to use on the diagonal representing self-proximities.
upper	logical value indicating whether the upper triangle of the distance/similarity matrix should be printed by print.dist/print.simil

`pairwise` logical value indicating whether distances should be computed for the pairs of `x` and `y` only.

`by_rows` logical indicating whether proximities between rows, or columns should be computed.

`convert_similarities, convert_distances`
logical indicating whether distances should be automatically converted into similarities (and the other way round) if needed.

`auto_convert_data_frames`
logical indicating whether data frames should be converted to matrices if all variables are numeric, or all are logical, or all are complex.

`FUN` optional function to be used by `as.dist` and `as.simil`. If `NULL`, it is looked up in the method registry. If there is none specified there, `FUN` defaults to `pr_simil2dist` and `pr_dist2simil`, respectively.

`...` further arguments passed to the proximity function.

Details

The interface is fashioned after `dist`, but can also compute cross-distances, and allows user extensions by means of registry of all proximity measures (see `pr_DB`).

Missing values are allowed but are excluded from all computations involving the rows within which they occur. If some columns are excluded in calculating a Euclidean, Manhattan, Canberra or Minkowski distance, the sum is scaled up proportionally to the number of columns used (compare `dist` in package **stats**).

Data frames are silently coerced to matrix if all columns are of (same) mode `numeric` or `logical`.

Distance measures can be used with `simil`, and similarity measures with `dist`. In these cases, the result is transformed accordingly using the specified coercion functions (default: $pr_simil2dist(d) = 1 - s$ and $pr_dist2simil(s) = 1/(1 + d)$). Objects of class `simil` and `dist` can be converted one in another using `as.dist` and `as.simil`, respectively.

Distance and similarity objects can conveniently be subset (see examples). Note that duplicate indexes are silently ignored.

Value

Auto distances/similarities are returned as an object of class `dist`/`simil` and cross-distances/similarities as an object of class `crossdist`/`crosssimil`.

Author(s)

David Meyer <David.Meyer@R-project.org> and Christian Buchta <Christian.Buchta@wu-wien.ac.at>

References

Anderberg, M.R. (1973), *Cluster analysis for applications*, 359 pp., Academic Press, New York, NY, USA.

Cox, M.F. and Cox, M.A.A. (2001), *Multidimensional Scaling*, Chapman and Hall.

Sokol, R.S. and Sneath P.H.A (1963), *Principles of Numerical Taxonomy*, W. H. Freeman and Co., San Francisco.

See Also

dist for compatibility information, and pr_DB for the proximity data base.

Examples

```
### show available proximities
summary(pr_DB)

### get more information about a particular one
pr_DB$get_entry("Jaccard")

### binary data
x <- matrix(sample(c(FALSE, TRUE), 8, rep = TRUE), ncol = 2)
dist(x, method = "Jaccard")

### for real-valued data
dist(x, method = "eJaccard")

### for positive real-valued data
dist(x, method = "fJaccard")

### cross distances
dist(x, x, method = "Jaccard")

### pairwise (diagonal)
dist(x, x, method = "Jaccard",
 pairwise = TRUE)

### this is the same but less efficient
as.matrix(stats::dist(x, method = "binary"))

### numeric data
x <- matrix(rnorm(16), ncol = 4)

## test inheritance of names
rownames(x) <- LETTERS[1:4]
colnames(x) <- letters[1:4]
dist(x)
dist(x, x)

## custom distance function
f <- function(x, y) sum(x * y)
dist(x, f)

## working with lists
z <- unlist(apply(x, 1, list), recursive = FALSE)
(d <- dist(z))
dist(z, z)

## subsetting
d[[1:2]]
subset(d, c(1,3,4))
```

```
d[[c(1,2,2)]]     # duplicate index gets ignored

## transformations and self-proximities
as.matrix(as.simil(d, function(x) exp(-x)), diag = 1)

## row and column indexes
row.dist(d)
col.dist(d)
```

---

pr_DB *Registry of proximities*

Description

Registry containing similarities and distances.

Usage

```
pr_DB
pr_DB$get_field(name)
pr_DB$get_fields()
pr_DB$get_field_names()
pr_DB$set_field(name, default = NA, type = NA, is_mandatory = FALSE,
                is_modifiable = TRUE, validity_FUN = NULL)

pr_DB$entry_exists(name)
pr_DB$get_entry(name)
pr_DB$get_entries(name = NULL, pattern = NULL)
pr_DB$get_entry_names(name)
pr_DB$set_entry(...)
pr_DB$modify_entry(...)
pr_DB$delete_entry(name)

## S3 method for class 'pr_DB'
summary(object, verbosity = c("short", "long"), ...)
```

Arguments

name	character string representing the name of an entry (case-insensitive).
pattern	regular expression to be matched to all fields of class "character" in all entries.
default	optional default value for the field.
type	optional character string specifying the class to be required for this field. If type is a character vector with more than two elements, the entries will be used as fixed set of alternatives. If type is not a character string or vector, the class will be inferred from the argument given.
is_mandatory	logical specifying whether new entries are required to have a value for this field.

is_modifiable	logical specifying whether entries can be changed with respect to that field.
validity_FUN	optional function or character string with the name of a function that checks the validity of a field entry. Such a function gets the value to be investigated as argument, and should stop with an error message if the value is not correct.
object	a registry object.
verbosity	controlling the verbosity of the output of the summary method for the registry. "short" gives just a list, "long" also gives the formulas.
...	for pr_DB$set_entry and pr_DB$modify_entry: named list of fields to be modified in or added to the registry (see details). This must include the index field ("names").

Details

pr_DB represents the registry of all proximity measures available. For each measure, it comprises meta-information that can be queried and extended. Also, new measures can be added. This is done using the following accessor functions of the pr_DB object:

get_field_names() returns a character vector with all field names. get_field() returns the information for a specific field as a list with components named as described above. get_fields() returns a list with all field entries. set_field() is used to create new fields in the repository (the default value will be set in all entries).

get_entry_names() returns a character vector with (the first alias of) all entries. entry_exists() is a predicate checking if an entry with the specified alias exists in the registry. get_entry() returns the specified entry if it exists (and, by default, gives an error if it does not). get_entries() is used to query more than one entry: either those matching name exactly, or those where the regular expression in pattern matches *any* character field in an entry. By default, all values are returned. delete_entry removes an existing entry from the registry (note that only user-provided entries can be deleted). set_entry and modify_entry require a named list of arguments used as field entries. At least the names index field is required. set_entry will check for all other mandatory fields. If specified in the field meta data, each field entry and the entry as a whole is checked for validity. Note that only user-specified fields and/or entries can be modified, the data shipped with the package are read-only.

The registry fields currently available are as follows:

FUN Function to register (see below).

names Character vector with an alias(es) for the measure.

PREFUN Optional function (or function name) for preprocessing code (see below).

POSTFUN Optional function (or function name) for postprocessing code (see below).

distance logical indicating whether this measure is a distance (TRUE) or similarity (FALSE).

convert Optional Function or function name for converting between similarities and distances when needed.

type Optional, the scale the measure applies to ("metric", "ordinal", "nominal", "binary", or "other"). If NULL, it is assumed to apply to some other unknown scale.

loop logical indicating whether FUN is just a measure, and therefore, if dist shall do the loop over all pairs of observations/variables, or if FUN does the loop on its own.

C_FUN logical indicating whether FUN is a C function.

abcd logical; if TRUE and binary data (or data to be interpreted as such) are supplied, the number of concordant and discordant pairs is precomputed for every two binary data vectors and supplied to the measure function.

formula Optional character string with the symbolic representation of the formula.

reference Optional reference (character).

description Optional description (character). Ideally, describes the context in which the measure can be applied.

A function specified as FUN parameter has mandatory arguments x and y (if abcd is FALSE), and a, b, c, d, n otherwise. Additionally, it gets all optional parameters specified by the user in the ... argument of the dist and simil functions, possibly changed and/or complemented by the corresponding (optional) PREFUN function. It must return the (diss-)similarity value computed from the arguments. x and y are two vectors from the data matrix (matrices) supplied. If abcd is FALSE, it is assumed that binary measures will be used, and the number of all n concordant and discordant pairs (x_k, y_k) precomputed and supplied instead of x and y. a, b, c, and d are the counts of all (TRUE, TRUE), (TRUE, FALSE), (FALSE, TRUE), and (FALSE, FALSE) pairs, respectively.

A function specified as PREFUN parameter has mandatory arguments x, y, p, and reg_entry, with y and p possibly being NULL depending on the task at hand. x and y are the data objects, p is a (possibly empty) list with all specified proximity parameters, and reg_entry is the registry entry (a named list containing all information specified in reg_add). The preprocessing function is allowed to change all these information, and if so, is required to return *all* arguments as a named list in the same order.

A function specified as POSTFUN parameter has two mandatory arguments: result and p. result will contain the computed raw data, i.e. a vector of length $n * (n-1)/2$ for auto-distances (see dist for details on dist objects), or a matrix for cross-distances. p contains the specified proximity parameters. Post-processing functions need to return the result object (even if unmodified).

A function specified as convert parameter should preserve the type of its argument.

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

dist

Examples

```
## create a new distance measure
mydist <- function(x,y) x * y

## create a new entry in the registry with two aliases
pr_DB$set_entry(FUN = mydist, names = c("test", "mydist"))

## look it up (index is case insensitive):
pr_DB$get_entry("TEST")
```

```
## modify the content of the description field in the new entry
pr_DB$modify_entry(names = "test", description = "foo function")

## create a new field
pr_DB$set_field("New")

## look up the test entry again (two ways)
pr_DB$get_entry("test")
pr_DB[["test"]]

## show total number of entries
length(pr_DB)

## show all entries (short list)
pr_DB$get_entries(pattern = "foo")

## show more details
summary(pr_DB, "long")

## get all entries in a list (and extract first two ones)
pr_DB$get_entries()[1:2]

## get all entries as a data frame (select first 3 fields)
as.data.frame(pr_DB)[,1:3]

## delete test entry
pr_DB$delete_entry("test")

## check if it is really gone
pr_DB$entry_exists("test")
```

`rowSums.dist` *Row Sums/Means of Sparse Symmetric Matrices*

Description

Compute the row (column) sums or means for a sparse symmetric (distance) matrix.

Usage

```
rowSums.dist(x, na.rm = FALSE)
rowMeans.dist(x, na.rm = FALSE, diag = TRUE)

colSums.dist(x, na.rm = FALSE)
colMeans.dist(x, na.rm = FALSE, diag = TRUE)
```

Arguments

x	an object of class dist.
na.rm	logical, should missing values (including NaN) be omitted from the summation?
diag	logical, should the diagonal elements be included in the computation?

Details

These functions are more efficient than expanding an object of class dist to matrix and using rowSums or rowMeans.

colSums and colMeans are provided for convenience. However, note that due to symmetry the result is always the same as for rowSums or rowMeans.

Value

A numeric vector of row sums.

Author(s)

Christian Buchta

See Also

as.matrix, as.dist, and rowSums.

Examples

```
##
x <- matrix(runif(10*2),ncol=2)
d <- dist(x)
rowSums(as.matrix(d))
rowSums.dist(d) # the same

rowMeans(as.matrix(d))
rowMeans.dist(d) # the same
rowMeans.dist(d, diag = FALSE) # not the same
## NAs
d[3] <- NA
rowSums.dist(d, na.rm = TRUE)
rowMeans.dist(d, na.rm = TRUE)
```

Index