Package 'OTrecod'

September 20, 2021

Title Data Fusion using Optimal Transportation Theory

Version 0.1.1

Maintainer Gregory Guernec <otrecod.pkg@gmail.com>

Description In the context of data fusion, the package provides a set of functions dedicated to the solving of 'recoding problems' using optimal transportation theory (Gares, Guernec, Savy (2019) <doi:10.1515/ijb-2018-0106> and Gares, Omer (2020) <doi:10.1080/01621459.2020.1775615>). From two databases with no overlapping part except a subset of shared variables, the functions of the package assist users until obtaining a unique synthetic database, where the missing information is fully completed.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.5)

Imports stats, dplyr, mice, missMDA, plyr, FactoMineR, StatMatch, proxy, rdist, ROI, ROI.plugin.glpk, ompr, ompr.roi, party, vcd

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Gregory Guernec [aut, cre],
Valerie Gares [aut],
Pierre Navaro [ctb],
Jeremy Omer [ctb],
Philippe Saint-Pierre [ctb],
Nicolas Savy [ctb]

Repository CRAN

Date/Publication 2021-09-20 09:50:03 UTC

R topics documented:

`api29` *Student performance in California schools: the results of the county 29*

Description

This database is a sample of the API program https://www.cde.ca.gov/re/pr/api.asp that ended in 2018. The sample is extracted from the data `api` of the package **survey**, related to the results of the county 29 (Nevada). The database contains information for the 418 schools of this county having at least 100 students. Missing information has been randomly (and voluntary) added to the `awards` and `ell` variables (4% and 7% respectively). Several variables have been voluntary categorized from their initial types.

Usage

```
api29
```

Format

A data.frame with 418 schools (rows) and 12 variables

cds the school identifier

apicl_2000 the API score in 2000 classed in 3 ordered levels: `[200-600]`,`(600-800]`,`(800-1000]`

stype the school type in a 3 ordered levels factor: `Elementary`, `Middle` or `High School`

awards the school eligible for awards program ? Two possible answers: `No` or `Yes`. This variable counts 4% of missing information.

acs.core the number of core academic courses in the school

api.stu the number of students tested in the school

acs.k3.20 the average class size years K-3 in the school. This variable is stored in a 3-levels factor: `Unknown`, `<=20`, `>20`.

grad.sch the percentage of parents with postgraduate education stored in a 3 ordered levels factor of percents: `0`, `1-10`, `>10`

ell the percentage of English language learners stored in a 4 ordered levels factor: `[0-10]`,`(10-30]`,`(30-50]`,`(50-100]`. This variable counts 7% of missing information.

mobility the percentage of students for whom this is the first year at the school, stored in 2 levels: `[0-20]` and `(20-100]`

meals the percentage of students eligible for subsidized meals stored in a 4 balanced levels factor (By quartiles): `[0-25]`, `(25-50]`, `(50-75]`, `(75-100]`

full the percentage of fully qualified teachers stored in a 2-levels factor: `1`: For strictly less than 90%, `2` otherwise

Source

This database is a sample of the data `api` from the package **survey**.

`api35` *Student performance in California schools: the results of the county 35*

Description

This database is a sample of the API program `https://www.cde.ca.gov/re/pr/api.asp` that ended in 2018. The sample is extracted from the data `api` of the package **survey**, related to the results of the county 35 (San Benito). The database contains information for the 362 schools of this county having at least 100 students. Missing information has been randomly (and voluntary) added to the `awards` and `ell` variables (4% and 7% respectively). Several variables have been voluntary categorized from their initial types.

Usage

```
api35
```

Format

A data.frame with 362 schools (rows) and 12 variables

cds the school identifier

apicl_1999 the API score in 1999 classed in 4 ordered levels: `G1`,`G2`,`G3`, `G4`

stype the school type in a 3 ordered levels factor: `Elementary`, `Middle` or `High School`

awards the school eligible for awards program ? Two possible answers: `No` or `Yes`. This variable counts 4% of missing information.

acs.core the number of core academic courses in the school

api.stu the number of students tested in the school

acs.k3.20 the average class size years K-3 in the school. This variable is stored in a 3-levels factor: `Unknown`, `<=20`, `>20`.

grad.sch the percentage of parents with postgraduate education stored in a 3 ordered levels factor of percents: `0`, `1-10`, `>10`

ell the percentage of English language learners stored in a 4 ordered levels factor: `[0-10]`,`(10-30]`,`(30-50]`,`(50-100]`. This variable counts 7% of missing information.

mobility the percentage of students for whom this is the first year at the school, stored in 2 levels: `1` and `2`

meals the percentage of students eligible for subsidized meals stored in a 4 balanced levels factor (By quartiles): `[0-25]`, `(25-50]`, `(50-75]`, `(75-100]`

full the percentage of fully qualified teachers stored in a 2-levels factor: `1`: For strictly less than 90%, `2` otherwise

Source

This database is a sample of the data `api` from the package **survey**.

`avg_dist_closest` *avg_dist_closest()*

Description

This function computes average distances between levels of two categorical variables located in two distinct databases.

Usage

```
avg_dist_closest(proxim, percent_closest = 1)
```

Arguments

`proxim`	a `proxim_dist` object
`percent_closest`	a ratio between 0 and 1 corresponding to the desired part of rows (or statistical units, or individuals) that will participate to the computation of the average distances between levels of factors or between an individual (a row) and levels of only one factor. Indeed, target variables are factors and each level of factor is characterized by a subset of rows, themselves characterized by their covariate profiles. These rows can be ordered according to their distances at their factor level. When this ratio is set to 1 (default setting), all rows participate to the computation, nevertheless when this ratio is less than 1, only rows with the smallest factor level distances will be kept for the computation (see 'Details').

Details

The function avg_dist_closest is an intermediate function for the implementation of original algorithms dedicated to the solving of recoding problems in data fusion using Optimal Transportation theory (for more details, consult the corresponding algorithms called OUTCOME, R_OUTCOME, JOINT and R_JOINT, in the reference (2)). The function avg_dist_closest is so directly implemented in the OT_outcome and OT_joint functions but can also be used separately. The function avg_dist_closest uses, in particular, the distance matrix D (that stores distances between rows of A and B) from the function proxim_dist to produce three distinct matrices saved in a list object. Therefore, the function requires in input, the specific output of the function proxim_dist which is available in the package and so must be used beforehand. In consequence, do not use this function directly on your database, and do not hesitate to consult the provided examples provided for a better understanding.

DEFINITION OF THE COST MATRIX

Assuming that A and B are two databases with a set of shared variables and that a same information (referred to a same target population) is stored as a variable Y in A and Z in B, such that Y is unknown in B and Z is unknown in A, whose encoding depends on the database (n_Y levels in A and n_Z levels in B). A distance between one given level y of Y and one given level z of Z is estimated by averaging the distances between the two subsets of individuals (units or rows) assigned to y in A and z in B, characterized by their vectors of covariates. The distance between two individuals depends on the variations between the shared covariates, and so depends on the chosen distance function using the function proxim_dist. For these computations, all the individuals concerned by these two levels can be taken into account, or only a part of them, depending on the argument percent_closest. When percent_closest < 1, the average distance between an individual i and a given level of factor z only uses the corresponding part of individuals related to z that are the closest to i. Therefore, this choice influences the estimations of average distances between levels of factors but also permits to reduce time computation when necessary.

The average distance between each individual of Y (resp. Z) and each levels of Z (resp. Y) are returned in output, in the object DindivA (DindivB respectively). The average distance between each levels of Y and each levels of Z are returned in a matrix saved in output (the object Davg). Davg returns the computation of the cost matrix D, whose dimensions ($n_Y \times n_Z$) correspond to the number of levels of Y (rows) and Z (columns). This matrix can be seen as the ability for an individual (row) to move from a given level of the target variable (Y) in A to a given level of Z in the database B (or vice versa).

Value

A list of 3 matrices is returned:

Davg — the cost matrix whose number of rows corresponds to n_Y, the number of levels of the target variable Y in the database A, and whose number of columns corresponds to n_Z: the number of levels of the target variable in B. In this case, the related cost matrix can be interpreted as the ability to move from one level of Y in A to one level of Z in B. Davg[P,Q] refers to the average distance between the modality P of Y (only known in A) and modality Q of Z (only known in B).

DindivA — a matrix whose number of rows corresponds to the number of rows of the first database A and number of columns corresponds to n_Z, the number of levels of the target variable Z in the second database B. DindivA[i,Q] refers to the

average distance between the i^{th} individual (or row) of the first database and a chosen proportion of individuals (`percent_closest` set by the user) of the second database having the modality Q of Z.

`DindivB` a matrix whose number of rows corresponds to the number of rows of the second database B and number of columns corresponds to nA, the number of levels of the target variable in the first database A. DindivB[k,P] refers to the average distance between the k^{th} individual (or row) of the second database and a chosen proportion of individuals (depending on `percent_closest`) of the first database having the modality P of Y.

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615

See Also

`proxim_dist`

Examples

```
data(simu_data)
### The covariates of the data are prepared according to the distance chosen
### using the transfo_dist function

### Example with The Manhattan distance

try1 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "M")
res1 = proxim_dist(try1,norm = "M")

# proxim_dist() fixes the chosen distance function,
# and defines neighborhoods between profiles and individuals

# The following row uses only 80 percents of individuals of each level
# of factors for the computation of the average distances:

res_new  = avg_dist_closest(res1,percent_closest = 0.80)
```

compare_lists *compare_lists()*

Description

This function compares the elements of two lists of same length.

Usage

```
compare_lists(listA, listB)
```

Arguments

listA	a first list
listB	a second list

Value

A boolean vector of same length as the two lists, which ith element is TRUE if the ith element is different between the 2 lists, or FALSE otherwise

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

Examples

```
data1 = data.frame(Gender = rep(c("m","f"),5),Age = rnorm(5,20,4))
data2 = data.frame(Gender = rep(c("m","f"),5),Age = rnorm(5,21,5))

list1 = list(A = 1:4, B = as.factor(c("A","B","C")), C = matrix(1:6,ncol = 3))
list2 = list(A = 1:4, B = as.factor(c("A","B")), C = matrix(1:6,ncol = 3))
list3 = list(A = 1:4, B = as.factor(c("A","B","C")),C = matrix(c(1:5,7),ncol = 3))
list4 = list(A = 1:4, B = as.factor(c("A","B","C")), C = matrix(1:6,ncol = 2))
list5 = list(A = 1:4, B = as.factor(c("A","B")), C = matrix(1:6,ncol = 2))
list6 = list(A = 1:4, B = as.factor(c("A","B")), C = data1)
list7 = list(A = 1:4, B = as.factor(c("A","B")), C = data2)

OTrecod::compare_lists(list1,list2)
OTrecod::compare_lists(list1,list3)
OTrecod::compare_lists(list1,list4)
OTrecod::compare_lists(list1,list5)
OTrecod::compare_lists(list6,list7)
```

error_group *error_group()*

Description

This function studies the association between two categorical distributions with different numbers of modalities.

Usage

```
error_group(REF, Z, ord = TRUE)
```

Arguments

REF	a factor with a reference number of levels.
Z	a factor with a number of levels greater than the number of levels of the reference.
ord	a boolean. If TRUE, only neighboring levels of Z will be grouped and tested together.

Details

Assuming that Y and Z are categorical variables summarizing a same information, and that one of the two related encodings is unknown by user because this latter is, for example, the result of predictions provided by a given model or algorithm, the function error_group searches for potential links between the modalities of Y to approach at best the distribution of Z.

Assuming that Y and Z have n_Y and n_Z modalities respectively so that $n_Y > n_Z$, in a first step, the function error_group combines modalities of Y to build all possible variables Y' verifying $n_{Y'} = n_Z$. In a second step, the association between Z and each new variable Y' generated is measured by studying the ratio of concordant pairs related to the confusion matrix but also using standard criterions: the Cramer's V (1), the Cohen's kappa coefficient (2) and the Spearman's rank correlation coefficient.

According to the type of Y, different combinations of modalities are tested:

- If Y and Z are ordinal (ord = TRUE), only consecutive modalities of Y will be grouped to build the variables Y'.
- If Y and Z are nominal (ord = FALSE), all combinations of modalities of Y (consecutive or not) will be grouped to build the variables Y'.

All the associations tested are listed in output as a data.frame object. The function error_group is directly integrated in the function verif_OT to evaluate the proximity of two multinomial distributions, when one of them is estimated from the predictions of an OT algorithm.

Example: Assuming that $Y = (1, 1, 2, 2, 3, 3, 4, 4)$ and $Z = (1, 1, 1, 1, 2, 2, 2, 2)$, so $n_Y = 4$ and $n_Z = 2$ and the related coefficient of correlation $cor(Y, Z)$ is 0.89. Are there groupings of modalities of Y which contribute to improving the proximity between Y and Z ? From Y,

the function error_group gives an answer to this question by successively constructing the variables: $Y_1 = (1,1,1,1,2,2,2,2)$, $Y_2 = (1,1,2,2,1,1,2,2)$, $Y_3 = (1,1,2,2,2,2,1,1)$ and tests $\text{cor}(Z,Y_1) = 1$, $\text{cor}(Z,Y_2) = 0$, $\text{cor}(Z,Y_3) = 0$. Here, the tests permit to conclude that the difference of encodings between Y and Z resulted in fact in a simple grouping of modalities.

Value

A data.frame with five columns:

combi	the first column enumerates all possible groups of modalities of Y to obtain the same number of levels as the reference.
error_rate	the second column gives the corresponding rate error from the confusion matrix (ratio of non-diagonal elements)
Kappa	this column indicates the result of the Cohen's kappa coefficient related to each combination of Y
Vcramer	this column indicates the result of the Cramer's V criterion related to each combination of Y
RankCor	this column indicates the result of the Spearman's coefficient of correlation related to each combination of Y

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. Cramér, Harald. (1946). Mathematical Methods of Statistics. Princeton: Princeton University Press.
2. McHugh, Mary L. (2012). Interrater reliability: The kappa statistic. Biochemia Medica. 22 (3): 276–282

Examples

```
# Basic examples:
Z1 = as.factor(sample(1:3,50,replace = TRUE)); length(Z1)
Z3 = as.factor(sample(1:2,50,replace = TRUE)); length(Z3)
Z2 = as.factor(sample(c("A","B","C","D"),50, replace = TRUE)); length(Z2)
Z4 = as.factor(sample(c("A","B","C","D","E"),50, replace = TRUE)); length(Z4)

# By only grouping consecutive levels of Z1:
error_group(Z1,Z4)
# By only all possible levels of Z1, consecutive or not:
error_group(Z3,Z1,FALSE)
```

```
### using a sample of the tab_test object (3 complete covariates)
### Y1 and Y2 are a same variable encoded in 2 different forms in DB 1 and 2:
### (4 levels for Y1 and 3 levels for Y2)

data(tab_test)
# Example with n1 = n2 = 70 and only X1 and X2 as covariates
tab_test2 = tab_test[c(1:70,5001:5070),1:5]

### An example of JOINT model (Manhattan distance)
# Suppose we want to impute the missing parts of Y1 in DB2 only ...
try1J = OT_joint(tab_test2, nominal = c(1,4:5), ordinal = c(2,3),
                 dist.choice = "M", which.DB = "B")

# Error rates between Y2 and the predictions of Y1 in the DB 2
# by grouping the levels of Y1:
error_group(try1J$DATA2_OT$Z,try1J$DATA2_OT$OTpred)
table(try1J$DATA2_OT$Z,try1J$DATA2_OT$OTpred)
```

---

ham *ham()*

Description

This function computes a matrix distance using the Hamming distance as proximity measure.

Usage

```
ham(mat_1, mat_2)
```

Arguments

mat_1	a vector, a matrix or a data.frame of binary values that may contain missing data
mat_2	a vector, a matrix or a data.frame of binary values with the same number of columns as mat_1 that may contain missing data

Details

ham returns the pairwise distances between rows (observations) of a single matrix if mat_1 equals mat_2. Otherwise ham returns the matrix distance between rows of the two matrices mat_1 and mat_2 if this 2 matrices are different in input. Computing the Hamming distance stays possible despite the presence of missing data by applying the following formula. Assuming that A and B are 2 matrices such as ncol(A) = ncol(B). The Hamming distance between the i^{th} row of A and the k^{th} row of B equals:

$$\text{ham}(A_i, B_k) = \frac{\sum_j 1_{\{A_{ij} \neq B_{kj}\}}}{\sum_j 1} \times \left(\frac{\sum_j 1}{\sum_j 1_{\{!\text{is.na}(A_{ij}) \& !\text{is.na}(B_{kj})\}}} \right)$$

where: $i = 1, \ldots, \text{nrow}(A)$ and $k = 1, \ldots, \text{nrow}(B)$; And the expression located to the right term of the multiplication corresponds to a specific weigh applied in presence of NAs in A_i and/or B_k.

This specificity is not implemented in the `cdist` function and the Hamming distance can not be computed using the `dist` function either.

The Hamming distance can not be calculated in only two situations:

1. If a row of A or B has only missing values (ie for each of the columns of A or B respectively).
2. The union of the indexes of the missing values in row i of A with the indexes of the missing values in row j of B concerns the indexes of all considered columns.

Example: Assuming that $\text{ncol}(A) = \text{ncol}(B) = 3$, if $A_i = (1, \text{NA}, 0)$ and $B_j = (\text{NA}, 1, \text{NA})$, for each column, either the information in row i is missing in A, or the information is missing in B, which induces: $\text{ham}(A_i, B_k) = \text{NA}$.

If `mat_1` is a vector and `mat_2` is a matrix (or data.frame) or vice versa, the length of `mat_1` must be equal to the number of columns of `mat_2`.

Value

A distance matrix

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

Roth R (2006). Introduction to Coding Theory. Cambridge University Press.

Examples

```
set.seed(3010); aaa = sample(c(0,1),12,replace = TRUE)
set.seed(3007); bbb = sample(c(0,1),15,replace = TRUE)
A = matrix(aaa, ncol = 3)
B = matrix(bbb, ncol = 3)

# These 2 matrices have no missing values

# Matrix of pairwise distances with A:
ham(A,A)

# Matrix of distances between the rows of A and the rows of B:
ham(A,B)
```

```
# If mat_1 is a vector of binary values:
ham(c(0,1,0),B)

# Now by considering A_NA and B_NA two matrices built from A and B respectively,
# where missing values have been manually added:
A_NA        = A
A_NA[3,1]   = NA
A_NA[2,2:3] = rep(NA,2)

B_NA = B
B_NA[2,2] = NA

ham(A_NA,B_NA)
```

---

imput_cov *imput_cov()*

Description

This function performs imputations on incomplete covariates, whatever their types, using functions from the package **MICE** (Van Buuren's Multiple Imputation) or functions from the package **missMDA** (Simple Imputation with Multivariate data analysis).

Usage

```
imput_cov(
  dat1,
  indcol = 1:ncol(dat1),
  R_mice = 5,
  meth = rep("pmm", ncol(dat1)),
  missMDA = FALSE,
  NB_COMP = 3,
  seed_choice = sample(1:1e+06, 1)
)
```

Arguments

dat1	a data.frame containing the variables to be imputed and those involved in the imputations
indcol	a vector of integers. The corresponding column indexes (or numbers) corresponding to the variables to be imputed and those involved in the imputations.
R_mice	an integer. The number of imputed database generated with MICE method (5 by default).
meth	a vector of characters which specifies the imputation method to be used for each column in dat1. "pmm" for continuous covariates or by default option, "logreg" for binary covariates, "polr" for ordinal covariates, "polyreg" for categorical covariates (no order), (cf mice for more details).

missMDA	a boolean. If TRUE, missing values are imputed using the factoral analysis for mixed data (imputeFAMD) from the **missMDA** package (2).
NB_COMP	an integer corresponding to the number of components used in FAMD to predict the missing entries (3 by default) when the missMDA option is TRUE.
seed_choice	an integer used as argument by the set.seed() for offsetting the random number generator (Random integer by default)

Details

By default, the function impute_cov handles missing information using multivariate imputation by chained equations (MICE, see (1) for more details about the method) by integrating in its syntax the function mice. All values of this last function are taken by default, excepted the required number of multiple imputations, which can be fixed by using the argument R_mice, and the chosen imputation method for each variable (meth argument), that corresponds to the argument defaultMethod of the function mice. When multiple imputations are required (for MICE only), each missing information is imputed by a consensus value: the average of the candidate values will be retained for numerical variables, while the most frequent class will be remained for categorical variables (ordinal or not). The output MICE_IMPS stores the imputed databases to allow users to build their own consensus values by themselves and(or) to eventually assess the variabilities related to the proposed imputed values if necessary. For this method, a random number generator must be fixed or sampled using the argument seed_choice.

When the argument missMDA is equalled to TRUE, incomplete values are replaced (single imputation) using a method based on dimensionality reduction called factor analysis for mixed data (FAMD) using the the imputeFAMD function of the **missMDA** package (2). Using this approach, the function imput_cov keeps all the default values integrated in the function imputeFAMD excepted the number of dimensions used for FAMD which can be fixed by users (3 by default).

Value

A list of 3 or 4 objects (depending on the missMDA argument). The first three following objects if missMDA = TRUE, otherwise 4 objects are returned:

RAW	a data.frame corresponding to the raw database
IMPUTE	a character indicating the type of selected imputation
DATA_IMPUTE	a data.frame corresponding to the completed (consensus if multiple imputations) database
MICE_IMPS	only if missMDA = FALSE. A list object containing the R imputed databases generated by MICE

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1–67. urlhttps://www.jstatsoft.org/v45/i03/
2. Josse J, Husson F (2016). missMDA: A Package for Handling Missing Values in Multivariate Data Analysis. Journal of Statistical Software, 70(1), 1–31. doi: 10.18637/jss.v070.i01

Examples

```
# Imputation of all incomplete covariates in the table simu_data:
data(simu_data)

# Here we keep the complete variable "Gender" in the imputation model.
# Using MICE (REP = 3):
imput_mice = imput_cov(simu_data,indcol = 4:8,R_mice = 3,
                       meth = c("logreg","polyreg","polr","logreg","pmm"))
summary(imput_mice)


# Using FAMD (NB_COMP = 3):
imput_famd = imput_cov(simu_data,indcol = 4:8,
                       meth = c("logreg","polyreg","polr","logreg","pmm"),
                       missMDA = TRUE)
summary(imput_famd)
```

`indiv_grp_closest` *indiv_grp_closest()*

Description

This function sequentially assigns individual predictions using a nearest neighbors procedure to solve recoding problems of data fusion.

Usage

```
indiv_grp_closest(
  proxim,
  jointprobaA = NULL,
  jointprobaB = NULL,
  percent_closest = 1,
  which.DB = "BOTH"
)
```

Arguments

`proxim`	a `proxim_dist` object or an object of similar structure

`jointprobaA` a matrix whose number of columns corresponds to the number of modalities of the target variable Y in database A, and which number of rows corresponds to the number of modalities of Z in database B. It gives an estimation of the joint probability of (Y, Z) in A. The sum of cells of this matrix must be equal to 1

`jointprobaB` a matrix whose number of columns equals to the number of modalities of the target variable Y in database A, and which number of rows corresponds to the number of modalities of Z in database B. It gives an estimation of the joint probability of (Y, Z) in B. The sum of cells of this matrix must be equal to 1

`percent_closest` a value between 0 and 1 (by default) corresponding to the fixed `percent closest` of individuals remained in the computation of the average distances

`which.DB` a character string (with quotes) that indicates which individual predictions need to be computed: only the individual predictions of Y in B ("B"), only those of Z in A ("A") or the both ("BOTH" by default)

Details

A. THE RECODING PROBLEM IN DATA FUSION

Assuming that Y and Z are two variables which refered to the same target population in two separate databases A and B respectively (no overlapping rows), so that Y and Z are never jointly observed. Assuming also that A and B share a subset of common covariates X of any types (same encodings in A and B) completed or not. Integrating these two databases often requires to solve the recoding problem by creating an unique database where the missing information of Y and Z is fully completed.

B. DESCRIPTION OF THE FUNCTION

The function `indiv_grp_closest` is an intermediate function used in the implementation of an algorithm called OUTCOME (and its enrichment R-OUTCOME, see the reference (2) for more details) dedicated to the solving of recoding problems in data fusion using Optimal Transportation theory. The model is implemented in the function `OT_outcome` which integrates the function `indiv_grp_closest` in its syntax as a possible second step of the algorithm. The function `indiv_grp_closest` can also be used separately provided that the argument `proxim` receives an output object of the function `proxim_dist`. This latter is available in the package and is so directly usable beforehand.

The algorithms `OUTCOME` (and `R-OUTCOME`) are made of two independent parts. Assuming that the objective consists in the prediction of Z in the database A:

- The first part of the algorithm solves the optimization problem by providing a solution called γ that corresponds here to an estimation of the joint distribution (Y, Z) in A.
- From the first part, a nearest neighbor procedure is carried out as a second part to provide the individual predictions of Z in A: this procedure is implemented in the function `indiv_group_closest`. In other words, this function sequentially assigns to each individual of A the modality of Z that is closest.

Obviously, this algorithm runs in the same way for the prediction of Y in the database B. The function `indiv_grp_closest` integrates in its syntax the function `avg_dist_closest`. Therefore, the related argument `percent_closest` is identical in the two functions. Thus, when computing average distances between an individual i and a subset of individuals assigned to a same level of

Y or Z is required, user can decide if all individuals from the subset of interest can participate to the computation (percent_closest=1) or only a fixed part p (<1) corresponding to the closest neighbors of i (in this case percent_closest = p).

The arguments jointprobaA and jointprobaB correspond to the estimations of γ (sum of cells must be equal to 1) in A and/or B respectively, according to the which.DB argument. For example, assuming that n_{Y_1} individuals are assigned to the first modality of Y in A, the objective consists in the individual predictions of Z in A. Then, if jointprobaA[1,2] = 0.10, the maximum number of individuals that can be assigned to the second modality of Z in A, can not exceed $0.10 \times n_A$. If $n_{Y_1} \leq 0.10 \times n_A$ then all individuals assigned to the first modality of Y will be assigned to the second modality of Z. At the end of the process, each individual with still no affectation will receive the same modality of Z as those of his nearest neighbor in B.

Value

A list of two vectors of numeric values:

YAtrans	a vector corresponding to the individual predictions of Y (numeric form) in the database B using the Optimal Transportation algorithm
ZBtrans	a vector corresponding to the individual predictions of Z (numeric form) in the database A using the Optimal Transportation algorithm

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615

See Also

proxim_dist,avg_dist_closest, ,OT_outcome

Examples

```
data(simu_data)

### Example with the Manhattan distance

try1 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "M")
res1 = proxim_dist(try1,norm = "M")
```

```
### Y(Yb1) and Z(Yb2) are a same information encoded in 2 different forms:
### (3 levels for Y and 5 levels for Z)
### ... Stored in two distinct databases, A and B, respectively
### The marginal distribution of Y in B is unknown,
### as the marginal distribution of Z in A ...

# Empirical distribution of Y in database A:
freqY = prop.table(table(try1$Y)); freqY

# Empirical distribution of Z in database B
freqZ = prop.table(table(try1$Z)); freqZ

# By supposing that the following matrix called transport symbolizes
# an estimation of the joint distribution L(Y,Z) ...
# Note that, in reality this distribution is UNKNOWN and is
# estimated in the OT function by resolving an optimisation problem.

transport1 = matrix(c(0,0.35285714,0,0.09142857,0,0.03571429,
                      0,0,0.08285714,0,0.07857143,0.03142857,
                      0.32714286,0,0),ncol = 5,byrow = FALSE)

# ... So that the marginal distributions of this object corresponds to freqY and freqZ:
apply(transport1,1,sum)  # = freqY
apply(transport1,2,sum)  # = freqZ

# The affectation of the predicted values of Y in database B and Z in database A
# are stored in the following object:

res2      = indiv_grp_closest(res1,jointprobaA = transport1, jointprobaB = transport1,
                              percent_closest= 0.90)
summary(res2)

# For the prediction of Z in A only, add the corresponding argument:
res3 = indiv_grp_closest(res1,jointprobaA = transport1, jointprobaB = transport1,
percent_closest= 0.90,which.DB="A")
```

`indiv_grp_optimal` *indiv_grp_optimal()*

Description

This function assigns individual predictions to the incomplete information of two integrated data-sources by solving a linear optimization problem.

Usage

```
indiv_grp_optimal(
  proxim,
  jointprobaA,
```

```
  jointprobaB,
  percent_closest = 1,
  solvr = "glpk",
  which.DB = "BOTH"
)
```

Arguments

proxim	a proxim_dist object or an object of similar structure
jointprobaA	a matrix whose number of columns is equal to the number of modalities of the target variable Y in database A, and whose number of rows is equal to the number of modalities of Z in database B. It gives an estimation of the joint probability (Y, Z) in the database A. The sum of cells of this matrix must be equal to 1.
jointprobaB	a matrix whose number of columns is equal to the number of modalities of the target variable Y in database A, and whose number of rows is equal to the number of modalities of Z in database B. It gives an estimation of the joint probability (Y, Z) in the database B. The sum of cells of this matrix must be equal to 1.
percent_closest	a value between 0 and 1 (by default) corresponding to the fixed percent closest of individuals used in the computation of the average distances
solvr	a character string that specifies the type of method selected to solve the optimization algorithms. The default solver is "glpk".
which.DB	a character string that indicates which individual predictions are computed: only the individual predictions of Y in B ("B"), only those of Z in A ("A") or the both ("BOTH" by default).

Details

A. THE RECODING PROBLEM IN DATA FUSION

Assuming that Y and Z are two target variables which refered to the same target population in two separate databases A and B respectively (no overlapping rows), so that Y and Z are never jointly observed. Assuming also that A and B share a subset of common covariates X of any types (same encodings in A and B) completed or not. Merging these two databases often requires to solve a recoding problem by creating an unique database where the missing information of Y and Z is fully completed.

B. DESCRIPTION OF THE FUNCTION

The function indiv_grp_optimal is an intermediate function used in the implementation of an algorithm called OUTCOME (and its enrichment R-OUTCOME (2)) dedicated to the solving of recoding problems in data fusion using Optimal Transportation theory. The model is implemented in the function OT_outcome which integrates the function indiv_grp_optimal in its syntax as a possible second step of the algorithm. The function indiv_grp_optimal can nevertheless be used separately providing that the argument proxim receives an output object of the function proxim_dist. This latter is available in the package and is so directly usable beforehand.

The function indiv_grp_optimal constitutes an alternative method to the nearest neighbor procedure implemented in the function indiv_grp_closest. As for the function indiv_grp_closest,

assuming that the objective consists in the prediction of Z in the database A, the first step of the algorithm related to OUTCOME provides an estimate of γ, the solution of the optimization problem, which can be seen, in this case as an estimation of the joint distribution (Y, Z) in A. Rather than using a nearest neighbor approach to provide individual predictions, the function indiv_grp_optimal solves an optimization problem using the simplex algorithm which searches for the individual predictions of Z that minimize the computed total distance satisfying the joint probability distribution estimated in the first part. More details about the theory related to the solving of this optimization problem is described in the section 5.3 of (2).

Obviously, this algorithm runs in the same way for the prediction of Y in the database B. The function indiv_grp_optimal integrates in its syntax the function avg_dist_closest and the related argument percent_closest is identical in the two functions. Thus, when computing average distances between an individual i and a subset of individuals assigned to a same level of Y or Z is required, user can decide if all individuals from the subset of interest can participate to the computation (percent_closest = 1) or only a fixed part p (<1) corresponding to the closest neighbors of i (in this case percent_closest = p).

The arguments jointprobaA and jointprobaB can be seen as estimations of γ (sum of cells must be equal to 1) that correspond to estimations of the joint distributions of (Y, Z) in A and B respectively.

The argument solvr permits user to choose the solver of the optimization algorithm. The default solver is "glpk" that corresponds to the GNU Linear Programming Kit (see (3) for more details). The solver "clp" (see (4)) for Coin-or Linear Programming, convenient in linear and quadratic situations, is also directly integrated in the function. Moreover, the function actually uses the R optimization infrastructure of the package **ROI** which offers a wide choice of solver to users by easily loading the associated plugins of **ROI** (see (5)).

Value

A list of two vectors of numeric values:

YAtrans	a vector corresponding to the predicted values of Y in database B (numeric form) according to the which.DB argument
ZBtrans	a vector corresponding to the predicted values of Z in database A (numeric form) according to the which.DB argument

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615

3. Makhorin A (2011). GNU Linear Programming Kit Reference Manual Version 4.47.http://www.gnu.org/software/glpk/
4. Forrest J, de la Nuez D, Lougee-Heimer R (2004). Clp User Guide. https://www.coin-or.org/Clp/userguide/index.html
5. Theussl S, Schwendinger F, Hornik K (2020). ROI: An Extensible R Optimization Infrastructure.Journal of Statistical Software,94(15), 1-64. doi: 10.18637/jss.v094.i15

See Also

proxim_dist, avg_dist_closest, indiv_grp_closest

Examples

```
### Example using The Euclidean distance on a complete database
# For this example we keep only 200 rows:

data(tab_test)
tab_test2 = tab_test[c(1:80,5001:5080),]; dim(tab_test2)

# Adding NAs in Y1 and Y2
tab_test2[tab_test2$ident == 2, 2] = NA
tab_test2[tab_test2$ident == 1, 3] = NA

# Because all covariates are ordered in numeric form,
# the transfo_dist function is not required here

res3      = proxim_dist(tab_test2,norm = "M")

#' ### Y(Y1) and Z(Y2) are a same variable encoded in 2 different forms:
### 4 levels for Y1 and 3 levels for Y2
### ... Stored in two distinct databases, A and B, respectively
### The marginal distribution of Y in B is unknown,
### as the marginal distribution of Z in A ...

# Assuming that the following matrix called transport symbolizes
# an estimation of the joint distribution L(Y,Z) ...
# Note that, in reality this distribution is UNKNOWN and is
# estimated in the OT function by resolving the optimization problem.

# By supposing:

val_trans  = c(0.275,0.115,0,0,0,0.085,0.165,0,0,0,0.095,0.265)
transport2 = matrix(val_trans,ncol = 3,byrow = FALSE)

# Getting the individual predictions of Z in A (only)
# by computing average distances on 90% of the nearest neighbors of
# each modality of Z in B
res4      = indiv_grp_optimal(res3,jointprobaA = transport2,
            jointprobaB = transport2, percent_closest= 0.90,
            which.DB = "A")
```

```
### Example 2 using The Manhattan distance with incomplete covariates
data(simu_data)

try1 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "M")
res1 = proxim_dist(try1,norm = "M")


### Y and Z are a same variable encoded in 2 different forms:
### (3 levels for Y and 5 levels for Z)
### ... Stored in two distinct databases, A and B, respectively
### The marginal distribution of Y in B is unknown,
### as the marginal distribution of Z in A ...

# By supposing that the following matrix called transport symbolizes
# an estimation of the joint distribution L(Y,Z) ...
# Note that, in reality this distribution is UNKNOWN and is
# estimated in the OT function by resolving an optimisation problem.

transport1 = matrix(c(0,0.35285714,0,0.09142857,0,0.03571429,
                      0,0,0.08285714,0,0.07857143,0.03142857,
                      0.32714286,0,0),ncol = 5,byrow = FALSE)


# The predicted values of Y in database B and Z in
# database A are stored in the following object:

res2 = indiv_grp_optimal(res1,jointprobaA = transport1,
                         jointprobaB = transport1,
                         percent_closest= 0.90)
summary(res2)
```

---

merge_dbs *merge_dbs()*

Description

Harmonization and merging before data fusion of two databases with specific outcome variables and shared covariates.

Usage

```
merge_dbs(
  DB1,
  DB2,
  row_ID1 = NULL,
  row_ID2 = NULL,
```

```
  NAME_Y,
  NAME_Z,
  order_levels_Y = levels(DB1[, NAME_Y]),
  order_levels_Z = levels(DB2[, NAME_Z]),
  ordinal_DB1 = NULL,
  ordinal_DB2 = NULL,
  impute = "NO",
  R_MICE = 5,
  NCP_FAMD = 3,
  seed_func = sample(1:1e+06, 1)
)
```

Arguments

DB1	a data.frame corresponding to the 1st database to merge (top database)
DB2	a data.frame corresponding to the 2nd database to merge (bottom database)
row_ID1	the column index of the row identifier of DB1 if it exists (no identifier by default)
row_ID2	the column index of the row identifier of DB2 if it exists (no identifier by default)
NAME_Y	the name of the outcome (with quotes) in its specific scale/encoding from the 1st database (DB1)
NAME_Z	the name of the outcome (with quotes) in its specific scale/encoding from the 2nd database (DB2)
order_levels_Y	the levels of Y stored in a vector and sorted in ascending order in the case of ordered factors. This option permits to reorder the levels in the 1st database (DB1) if necessary.
order_levels_Z	the levels of Z stored in a vector and sorted in ascending order in the case of ordered factors. This option permits to reorder the levels in the 2nd database (DB2) if necessary.
ordinal_DB1	a vector of column indexes corresponding to ordinal variables in the 1st database (no ordinal variable by default)
ordinal_DB2	a vector of column indexes corresponding to ordinal variables in the 2nd database (no ordinal variable by default)
impute	a character equals to "NO" when missing data on covariates are kept (Default option), "CC" for Complete Case by keeping only covariates with no missing information , "MICE" for MICE multiple imputation approach, "FAMD" for single imputation approach using Factorial Analysis for Mixed Data
R_MICE	the chosen number of multiple imputations required for the MICE approach (5 by default)
NCP_FAMD	an integer corresponding to the number of components used to predict missing values in FAMD imputation (3 by default)
seed_func	an integer used as argument by the set.seed() for offsetting the random number generator (Random integer by default, only useful with MICE)

Details

Assuming that DB1 and DB2 are two databases (two separate data.frames with no overlapping rows) to be merged vertically before data fusion, the function `merge_dbs` performs this merging and checks the harmonization of the shared variables. Firslty, the two databases declared as input to the function (via the argument `DB1` and `DB2`) must have the same specific structure. Each database must contain a target variable (whose label must be filled in the argument `Y` for DB1 and in `Z` for DB2 respectively, so that the final synthetic database in output will contain an incomplete variable `Y` whose corresponding values will be missing in DB2 and another incomplete target `Z` whose values will be missing in DB1), a subset of shared covariates (by example, the best predictors of Y in DB1, and Z in DB2). Each database can have a row identifier whose label must be assigned in the argument `row_ID1` for DB1 and `row_ID2` for DB2. Nevertheless, by default DB1 and DB2 are supposed with no row identifiers. The merging keeps unchanged the order of rows in the two databases provided that Y and Z have no missing values. By building, the first declared database (in the argument `DB1`) will be placed automatically above the second one (declared in the argument `DB2`) in the final database.

Firstly, by default, a variable with the same name in the two databases is abusively considered as shared. This condition is obviously insufficient to be kept in the final subset of shared variables, and the function `merge_dbs` so performs checks before merging described below.

A. Discrepancies between shared variables

- Shared variables with discrepancies of types between the two databases (for example, a variable with a common name in the two databases but stored as numeric in DB1, and stored as character in DB2) will be removed from the merging and the variable name will be saved in output (`REMOVE1`).
- Shared factors with discrepancies of levels (or number of levels) will be also removed from the merging and the variable name will be saved in output (`REMOVE2`).
- covariates whose names are specific to each database will be also deleted from the merging.
- If some important predictors have been improperly excluded from the merging due to the above-mentioned checks, it is possible for user to transform these variables a posteriori, and re-run the function.

B. Rules for the two outcomes (target variables)

The types of `Y` and `Z` must be suitable:

- Categorical (ordered or not) factors are allowed.
- Numeric and discrete outcomes with a finite number of values are allowed but will be automatically converted as ordered factors using the function `transfo_target` integrated in the function `merge_dbs`.

C. The function `merge_dbs` handles incomplete information of shared variables, by respecting the following rules:

- If `Y` or `Z` have missing values in DB1 or DB2, corresponding rows are excluded from the database before merging. Moreover, in the case of incomplete outcomes, if A and B have row identifiers, the corresponding identifiers are removed and these latters are stored in the objects `DB1_ID` and `DB2_ID` of the output.

- Before overlay, the function deals with incomplete covariates according to the argument `impute`. Users can decide to work with complete case only ("CC"), to keep ("NO") or impute incomplete information ("MICE","FAMD").
- The function `imput_cov`, integrated in the syntax of `merge_dbs` deals with imputations. Two approaches are actually available: the multivariate imputation by chained equation approach (MICE, see (3) for more details about the approach or the corresponding package **mice**), and an imputation approach from the package **missMDA** that uses a dimensionality reduction method (here a factor analysis for mixed data called FAMD (4)), to provide single imputations. If multiple imputation is required (`impute = "MICE"`), the default imputation methods are applied according to the type of the variables. The average of the plausible values will be kept for a continuous variable, while the most frequent candidate will be kept as a consensus value for a categorical variable or factor (ordinal or not).

As a finally step, the function checks that all values related to Y in B are missing and inversely for Z in A.

Value

A list containing 12 elements (13 when `impute` equals "MICE"):

`DB_READY`	the database matched from the two initial databases with common covariates and imputed or not according to the impute option
`ID1_drop`	the row numbers or row identifiers excluded of the data merging because of the presence of missing values in the target variable of DB1. NULL otherwise
`ID2_drop`	the row numbers or row identifiers excluded of the data merging because of the presence of missing values in the target variable of DB2. NULL otherwise
`Y_LEVELS`	the remaining levels of the target variable Y in the DB1
`Z_LEVELS`	the remaining Levels of the target variable Z in the DB2
`REMOVE1`	the labels of the deleted covariates because of type incompatibilies of type from DB1 to DB2
`REMOVE2`	the removed factor(s) because of levels incompatibilities from DB1 to DB2
`REMAINING_VAR`	labels of the remained covariates for data fusion
`IMPUTE_TYPE`	a character with quotes that specify the method eventually chosen to handle missing data in covariates
`MICE_DETAILS`	a list containing the details of the imputed datasets using `MICE` when this option is chosen. Raw and imputed databases imputed for DB1 and DB2 according to the number of multiple imputation selected (Only if impute = "MICE")
`DB1_raw`	a data.frame corresponding to DB1 after merging
`DB2_raw`	a data.frame corresponding to DB2 after merging
`SEED`	an integer used as argument by the `set.seed` function for offsetting the random number generator (random selection by default)

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615
3. van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1–67. urlhttps://www.jstatsoft.org/v45/i03/
4. Josse J, Husson F (2016). missMDA: A Package for Handling Missing Values in Multivariate Data Analysis. Journal of Statistical Software, 70(1), 1–31. doi: 10.18637/jss.v070.i01

See Also

imput_cov, transfo_target, select_pred

Examples

```
### Assuming two distinct databases from simu_data: data_A and data_B
### Some transformations will be made beforehand on variables to generate
### heterogeneities between the two bases.
data(simu_data)
data_A = simu_data[simu_data$DB == "A",c(2,4:8)]
data_B = simu_data[simu_data$DB == "B",c(3,4:8)]

# For the example, a covariate is added (Weight) only in data_A
data_A$Weight = rnorm(300,70,5)

# Be careful: the target variables must be in factor (or ordered) in the 2 databases
# Because it is not the case for Yb2 in data_B, the function will convert it.
data_B$Yb2    = as.factor(data_B$Yb2)

# Moreover, the Dosage covariate is stored in 3 classes in data_B (instead of 4 classes in data_B)
# to make the encoding of this covariate specific to each database.
data_B$Dosage = as.character(data_B$Dosage)
data_B$Dosage = as.factor(ifelse(data_B$Dosage %in% c("Dos 1","Dos 2"),"D1",
                          ifelse(data_B$Dosage == "Dos 3","D3","D4")))

# For more diversity, this covariate iis placed at the last column of the data_B
data_B        = data_B[,c(1:3,5,6,4)]

# Ex 1: The two databases are merged and incomplete covariates are imputed using MICE
soluc1  = merge_dbs(data_A,data_B,
                    NAME_Y = "Yb1",NAME_Z = "Yb2",
                    ordinal_DB1 = c(1,4), ordinal_DB2 = c(1,6),
                    impute = "MICE",R_MICE = 2, seed_func = 3011)
summary(soluc1$DB_READY)
```

```
# Ex 2: The two databases are merged and missing values are kept
soluc2  = merge_dbs(data_A,data_B,
                    NAME_Y = "Yb1",NAME_Z = "Yb2",
                    ordinal_DB1 = c(1,4), ordinal_DB2 = c(1,6),
                    impute = "NO",seed_func = 3011)

# Ex 3: The two databases are merged by only keeping the complete cases
soluc3  = merge_dbs(data_A,data_B,
                    NAME_Y = "Yb1",NAME_Z = "Yb2",
                    ordinal_DB1 = c(1,4), ordinal_DB2 = c(1,6),
                    impute = "CC",seed_func = 3011)

# Ex 4: The two databases are merged and incomplete covariates are imputed using FAMD
soluc4  = merge_dbs(data_A,data_B,
                    NAME_Y = "Yb1",NAME_Z = "Yb2",
                    ordinal_DB1 = c(1,4), ordinal_DB2 = c(1,6),
                    impute = "FAMD",NCP_FAMD = 4,seed_func = 2096)

# Conclusion:
# The data fusion is successful in each situation.
# The Dosage and Weight covariates have been normally excluded from the fusion.
# The covariates have been imputed when required.
```

OT_joint *OT_joint()*

Description

The function OT_joint integrates two algorithms called (JOINT) and (R-JOINT) dedicated to the solving of recoding problems in data fusion using optimal transportation of the joint distribution of outcomes and covariates.

Usage

```
OT_joint(
  datab,
  index_DB_Y_Z = 1:3,
  nominal = NULL,
  ordinal = NULL,
  logic = NULL,
  convert.num = NULL,
  convert.clss = NULL,
  dist.choice = "E",
  percent.knn = 1,
  maxrelax = 0,
  lambda.reg = 0,
  prox.X = 0.1,
  solvR = "glpk",
```

```
  which.DB = "BOTH"
)
```

Arguments

`datab`	a data.frame made up of two overlayed databases with at least four columns sorted in a random order. One column must be a column dedicated to the identification of the two databases ranked in ascending order (For example: 1 for the top database and 2 for the database from below, or more logically here A and B ...But not B and A!). One column (Y here but other names are allowed) must correspond to the target variable related to the information of interest to merge with its specific encoding in the database A (corresponding encoding should be missing in the database B). In the same way, one column (Z here) corresponds to the second target variable with its specific encoding in the database B (corresponding encoding should be missing in the database A). Finally, the input database must have at least one shared covariate with same encoding in A and B. Please notice that, if your data.frame has only one shared covariate (four columns) with missing values (because no imputation is desired) then a warning will appear and the algorithm will only run with complete cases.
`index_DB_Y_Z`	a vector of three indexes of variables. The first index must correspond to the index of the databases identifier column. The second index corresponds to the index of the target variable in the first database (A) while the third index corresponds to the column index related to the target variable in the second database (B).
`nominal`	a vector of column indexes of all the nominal (not ordered) variables (database identifier and target variables included if it is the case for them).
`ordinal`	a vector of column indexes of all the ordinal variables (database identifier and target variables included if it is the case for them).
`logic`	a vector of column indexes of all the boolean variables of the data.frame.
`convert.num`	indexes of the continuous (quantitative) variables. They will be automatically converted in ordered factors. By default, no continuous variables is assumed in the database.
`convert.clss`	a vector indicating for each continuous variable to convert, the corresponding desired number of levels. If the length of the argument `convert_num` exceeds 1 while the length of `convert_clss` equals 1 (only one integer), each discretization will count the same number of levels (quantiles).
`dist.choice`	a character string (with quotes) corresponding to the distance function chosen between: the euclidean distance ("E", by default), the Manhattan distance ("M"), the Gower distance ("G"), and the Hamming distance ("H") for binary covariates only.
`percent.knn`	the ratio of closest neighbors involved in the computations of the cost matrices. 1 is the default value that includes all rows in the computation.
`maxrelax`	the maximum percentage of deviation from expected probability masses. It must be equal to 0 (default value) for the `JOINT` algorithm, and equal to a strictly positive value for the R-JOINT algorithm.

lambda.reg	a coefficient measuring the importance of the regularization term. It corresponds to the R-JOINT algorithm for a value other than 0 (default value).
prox.X	a probability (betwen 0 and 1) used to calculate the distance threshold below which two covariates' profiles are supposed as neighbors. If prox.X = 1, all profiles are considered as neighbors.
solvR	a character string that specifies the type of method selected to solve the optimization algorithms. The default solver is "glpk".
which.DB	a character string indicating the database to complete ("BOTH" by default, for the prediction of Y and Z in the two databases), "A" only for the imputation of Z in A, "B" only for the imputation of Y in B.

Details

A. THE RECODING PROBLEM IN DATA FUSION

Assuming that Y and Z are two target variables which refered to the same target population in two separate databases A and B respectively (no overlapping rows), so that Y and Z are never jointly observed. Assuming also that A and B share a subset of common covariates X of any types (same encodings in A and B) completed or not. Merging these two databases often requires to solve a recoding problem by creating an unique database where the missing information of Y and Z is fully completed.

B. INFORMATIONS ABOUT THE ALGORITHM

As with the function OT_outcome, the function OT_joint provides a solution to the recoding problem by proposing an application of optimal transportation which aims is to search for a bijective mapping between the joint distributions of (Y, X) and (Z, X) in A and B (see (2) for more details). The principle of the algorithm is also based on the resolution of an optimization problem, which provides a solution γ (as called in (1) and (2)), estimate of the joint distribution of (X, Y, Z) according to the database to complete (see the argument which.DB for the choice of the database). While the algorithms OUTCOME and R_OUTCOME integrated in the function OT_outcome require post-treatment steps to provide individual predictions, the algorithm JOINT directly uses estimations of the conditional distributions $(Y|Z, X)$ in B and $(Z|Y, X)$ in A to predict the corresponding incomplete individuals informations of Y and/or Z respectively. This algorithm supposes that the conditional distribution $(Y|X)$ must be identical in A and B. Respectively, $(Z|X)$ is supposed identical in A and B. Estimations a posteriori of conditional probabilities $P[Y|X, Z]$ and $P[Z|X, Y]$ are available for each profiles of covariates in output (See the objects estimatorYB and estimatorZA). Estimations of γ are also available according to the chosen transport distributions (See the arguments gamma_A and gamma_B).

The algorithm R-JOINT gathers enrichments of the algorithm JOINT and is also available via the function OT_joint. It allows users to add a relaxation term in the algorithm to relax distributional assumptions (maxrelax>0), and (or) add also a positive regularization term (lamdba.reg>0) expressing that the transportation map does not vary to quickly with respect of covariates X. Is is suggested to users to calibrate these two parameters a posteriori by studying the stability of the individual predictions in output.

C. EXPECTED STRUCTURE FOR THE INPUT DATABASE

The input database is a data.frame that must satisfy a specific form:

- Two overlayed databases containing a common column of databases identifiers (A and B, 1 or 2, by examples, encoded in numeric or factor form)

- A column corresponding to the target variable with its specific encoding in A (For example a factor Y encoded in n_Y levels, ordered or not, with NAs in the corresponding rows of B)
- A column corresponding to another target outcome summarizing the same latent information with its specific encoding in B (By example a factor Z with n_Z levels, with NAs in rows of A)
- The order of the variables in the database have no importance but the column indexes related to the three columns previously described (ie ID, Y and Z) must be rigorously specified in the argument `index_DB_Y_Z`.
- A set of shared common categorical covariates (at least one but more is recommended) with or without missing values (provided that the number of covariates exceeds 1) is required. On the contrary to the function `OT_outcome`, please notice, that the function `OT_joint` does not accept continuous covariates therefore these latters will have to be categorized beforehand or using the provided input process (see `convert.num`).

The function `merge_dbs` is available in this package to assist user in the preparation of their databases.

Remarks about the target variables:

- A target variable can be of categorical type, but also discrete, stored in factor, ordered or not. Nevertheless, notice that, if the variable is stored in numeric it will be automatically converted in ordered factors.
- If a target variable is incomplete, the corresponding rows will be automatically dropped during the execution of the function.

The type of each variables (including ID, Y and Z) of the database must be rigorously specified, in one of the four arguments `quanti`, `nominal`, `ordinal` and `logic`.

D. TRANSFORMATIONS OF CONTINUOUS COVARIATES

Continuous shared variables (predictors) with infinite numbers of values have to be categorized before being introduced in the function. To assist users in this task, the function `OT_joint` integrates in its syntax a process dedicated to the categorization of continuous covariates. For this, it is necessary to rigorously fill in the arguments `quanti` and `convert.clss`. The first one informs about the column indexes of the continuous variables to be transformed in ordered factor while the second one specifies the corresponding number of desired balanced levels (for unbalanced levels, users must do transformations by themselves). Therefore `convert.num` and `convert.clss` must be vectors of same length, but if the length of `quanti` exceeds 1, while the length of `convert.clss` is 1, then, by default, all the covariates to convert will have the same number of classes (transformation by quantiles), that corresponds to the value specified in the argument `convert.clss`. Notice that only covariates can be transformed (not target variables) and that any incomplete information must have been taken into account beforehand (via the dedicated functions `merge_dbs` or `imput_cov` for examples). Moreover, all the indexes informed in the argument `convert.num` must also be informed in the argument `quanti`. Finally, it is recommended to declare all discrete covariates as ordinal factors using the argument `ordinal`.

E. INFORMATIONS ABOUT DISTANCE FUNCTIONS AND RELATED PARAMETERS

Each individual (or row) of a given database is here characterized by a vector of covariates, so the distance between two individuals or groups of individuals depends on similarities between covariates according to the distance function chosen by user (via the argument `dist.choice`). Actually four distance functions are implemented in `OT_joint` to take into account the most frequently encountered situation (see (3)):

- the Manhattan distance ("M")
- the Euclidean distance ("E")
- the Gower distance for mixed data (see (4): "G")
- the Hamming distance for binary data ("H")

Finally, two profiles of covariates P_1 (n_1 individuals) and P_2 (n_2 individuals) will be considered as neighbors if $dist(P_1, P_2) < prox.X \times max(dist(P_i, P_j))$ where $prox.X$ must be fixed by user ($i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$). This choice is used in the computation of the `JOINT` and `R_JOINT` algorithms. The `prox.X` argument influences a lot the running time of the algorithm. The greater, the more the value will be close to 1, the more the convergence of the algorithm will be difficult or even impossible.

Each individual i from A or B is here considered as a neighbor of only one profile of covariates P_j.

F. INFORMATIONS ABOUT THE SOLVER

The argument `solvR` permits user to choose the solver of the optimization algorithm. The default solver is "glpk" that corresponds to the GNU Linear Programming Kit (see (5) for more details). Moreover, the function actually uses the `R` optimization infrastructure of the package **ROI** which offers a wide choice of solver to users by easily loading the associated plugins of **ROI** (see (6)).

For more details about the algorithms integrated in `OT_joint`, please consult (2).

Value

A "otres" class object of 9 elements:

`time_exe`	running time of the function
`gamma_A`	estimate of γ for the completion of A. A matrix that corresponds to the joint distribution of (Y, Z, X) in A
`gamma_B`	estimate of γ for the completion of B. A matrix that corresponds to the joint distribution of (Y, Z, X) in B
`profile`	a data.frame that gives all details about the remaining P profiles of covariates. These informations can be linked to the `estimatorZA` and the `estimatorYB` objects for a better interpretation of the results.
`res_prox`	a `proxim_dist` object
`estimatorZA`	an array that corresponds to estimates of the probability distribution of Z conditional to X and Y in database A. The number of rows of each table corresponds to the total number of profiles of covariates. The first dimension of each table (rownames) correspond to the profiles of covariates sorted by order of appearance in the merged database. The second dimension of the array (columns of the tables) corresponds to the levels of Y while the third element corresponds to the levels of Z.
`estimatorYB`	an array that corresponds to estimates of the probability distribution of Y conditional to X and Z in database B. The number of rows of each table corresponds to the total number of profiles of covariates. The first dimension of each table (rownames) correspond to the profiles of covariates sorted by order of appearance in the merged database. The second dimension of the array (columns of the tables) corresponds to the levels of Z while the third element corresponds to the levels of Y.

DATA1_OT — the database A with the individual predictions of Z using an optimal transportation algorithm (JOINT) or R-JOINT

DATA2_OT — the database B with the individual predictions of Y using an optimal transportation algorithm (JOINT) or R-JOINT

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615
3. Anderberg, M.R. (1973), Cluster analysis for applications, 359 pp., Academic Press, New York, NY, USA.
4. Gower J.C. (1971). A general coefficient of similarity and some of its properties. Biometrics, 27, 623–637
5. Makhorin A (2011). GNU Linear Programming Kit Reference Manual Version 4.47.http://www.gnu.org/software/glpk/
6. Theussl S, Schwendinger F, Hornik K (2020). ROI: An Extensible R Optimization Infrastructure.Journal of Statistical Software,94(15), 1-64. doi: 10.18637/jss.v094.i15

See Also

merge_dbs, OT_outcome, proxim_dist, avg_dist_closest

Examples

```
### An example of JOINT algorithm with:
#-----
# - A sample of the database tab_test
# - Y1 and Y2 are a 2 outcomes encoded in 2 different forms in DB 1 and 2:
#   4 levels for Y1 and 3 levels for Y2
# - n1 = n2 = 40
# - 2 discrete covariates X1 and X2 defined as ordinal
# - Distances estimated using the Gower function
# Predictions are assessed for Y1 in B only
#-----

data(tab_test)
tab_test2 = tab_test[c(1:40,5001:5040),1:5]
```

```
try1J = OT_joint(tab_test2, nominal = c(1,4:5), ordinal = c(2,3),
                 dist.choice = "G", which.DB = "B")



### An example of R-JOINT algorithm using the previous database,
### and keeping the same options excepted for:
#-----
# - The distances are estimated using the Gower function
# - Inclusion of an error term in the constraints on
#   the marginals (relaxation term)
# Predictions are assessed for Y1 AND Y2 in A and B respectively
#-----

try1RJ = OT_joint(tab_test2, nominal = c(1,4:5), ordinal = c(2,3),
                  dist.choice = "G", maxrelax = 0.4,
                  which.DB = "BOTH")

### The previous example of R-JOINT algorithm with:
# - Adding a regularization term
# Predictions are assessed for Y1 and Y2 in A and B respectively
#-----

try2RJ = OT_joint(tab_test2, nominal = c(1,4:5), ordinal = c(2,3),
                  dist.choice = "G", maxrelax = 0.4, lambda.reg = 0.9,
                  which.DB = "BOTH")



### Another example of JOINT algorithm with:
#-----
# - A sample of the database simu_data
# - Y1 and Y2 are a 2 outcomes encoded in 2 different forms in DB A and B:
#   (3 levels for Y and 5 levels for Z)
# - n1 = n2 = 100
# - 3 covariates: Gender, Smoking and Age in a qualitative form
# - Complete Case study
# - The Hamming distance
# Predictions are assessed for Y1 and Y2 in A and B respectively
#-----

data(simu_data)
simu_data2 = simu_data[c(1:100,401:500),c(1:4,7:8)]
simu_data3 = simu_data2[!is.na(simu_data2$Age),]

try2J = OT_joint(simu_data3, convert.num = 6, convert.clss = 3,
                 nominal = c(1,4:5), ordinal = 2:3,
                 dist.choice = "H", which.DB = "BOTH")
```

OT_outcome *OT_outcome()*

Description

The function OT_outcome integrates two algorithms called (OUTCOME) and (R-OUTCOME) dedicated to the solving of recoding problems in data fusion using optimal transportation (OT) of the joint distribution of outcomes.

Usage

```
OT_outcome(
  datab,
  index_DB_Y_Z = 1:3,
  quanti = NULL,
  nominal = NULL,
  ordinal = NULL,
  logic = NULL,
  convert.num = NULL,
  convert.clss = NULL,
  FAMD.coord = "NO",
  FAMD.perc = 0.8,
  dist.choice = "E",
  percent.knn = 1,
  maxrelax = 0,
  indiv.method = "sequential",
  prox.dist = 0,
  solvR = "glpk",
  which.DB = "BOTH"
)
```

Arguments

datab	a data.frame made up of two overlayed databases with at least four columns sorted in a random order. One column must be a column dedicated to the identification of the two databases ranked in ascending order (For example: 1 for the top database and 2 for the database from below, or more logically here A and B ...But not B and A!). One column (Y here but other names are allowed) must correspond to the target variable related to the information of interest to merge with its specific encoding in the database A (corresponding encoding should be missing in the database B). In the same way, one column (Z here) corresponds to the second target variable with its specific encoding in the database B (corresponding encoding should be missing in the database A). Finally, the input database must have at least one shared covariate with same encoding in A and B. Please notice that, if your data.frame has only one shared covariate (four columns) with missing values (because no imputation is desired) then a warning will appear and the algorithm will only run with complete cases.

`index_DB_Y_Z` a vector of three indexes of variables. The first index must correspond to the index of the databases identifier column. The second index corresponds to the index of the target variable in the first database (A) while the third index corresponds to the column index related to the target variable in the second database (B).

`quanti` a vector of column indexes of all the quantitative variables (database identifier and target variables included if it is the case for them).

`nominal` a vector of column indexes of all the nominal (not ordered) variables (database identifier and target variables included if it is the case for them).

`ordinal` a vector of column indexes of all the ordinal variables (database identifier and target variables included if it is the case for them).

`logic` a vector of column indexes of all the boolean variables of the data.frame.

`convert.num` indexes of the continuous (quantitative) variables to convert in ordered factors if necessary. All declared indexes in this argument must have been declared in the argument `quanti` (no conversion by default).

`convert.clss` a vector indicating for each continuous variable to convert, the corresponding desired number of levels. If the length of the argument `convert_num` exceeds 1 while the length of `convert_clss` equals 1 (only one integer), each discretization will count the same number of levels (quantiles).

`FAMD.coord` a logical that must be set to TRUE when user decides to work with principal components of a factor analysis for mixed data (FAMD) instead of the set of raw covariates (FALSE is the default value).

`FAMD.perc` a percent (between 0 and 1) linked to the `FAMD.coord` argument (0.8 is the default value). When this latter equals TRUE, this argument corresponds to the minimum part of variability that must be taken into account by the principal components of the FAMD method. This option fixes the remaining number of principal components for the rest of the study.

`dist.choice` a character string (with quotes) corresponding to the distance function chosen between: the euclidean distance ("E", by default), The Manhattan distance ("M"), the Gower distance ("G"), the Hamming distance ("H") for binary covariates only, and the Euclidean or Manhattan distance computed from principal components of a factor analysis of mixed data ("FAMD"). See (1) for details.

`percent.knn` the ratio of closest neighbors involved in the computations of the cost matrices. 1 is the default value that includes all rows in the computation.

`maxrelax` the maximum percentage of deviation from expected probability masses. It must be equal to 0 (default value) for the `OUTCOME` algorithm, and equal to a strictly positive value for the R-OUTCOME algorithm. See (2) for details.

`indiv.method` a character string indicating the chosen method to get individual predictions from the joint probabilities assessed, "sequential" by default, or "optimal". See the `details` section and (2) for details.

`prox.dist` a probability (between 0 and 1) used to calculate the distance threshold below which an individual (a row) is considered as a neighbor of a given profile of covariates. When shared variables are all factors or categorical, it is suggested to keep this option to 0.

`solvR` a character string that specifies the type of method selected to solve the optimization algorithms. The default solver is "glpk".

`which.DB` a character string indicating the database to complete ("BOTH" by default, for the prediction of Y and Z in the two databases), "A" only for the imputation of Z in A, "B" only for the imputation of Y in B.

Details

A. THE RECODING PROBLEM IN DATA FUSION

Assuming that Y and Z are two target variables which refered to the same target population in two separate databases A and B respectively (no overlapping rows), so that Y and Z are never jointly observed. Assuming also that A and B share a subset of common covariates X of any types (same encodings in A and B) completed or not. Merging these two databases often requires to solve a recoding problem by creating an unique database where the missing information of Y and Z is fully completed.

B. INFORMATIONS ABOUT THE ALGORITHM

The algorithm integrated in the function `OT_outcome` provides a solution to the recoding problem previously described by proposing an application of optimal transportation which aims is to search for a bijective mapping between the distributions of of Y in A and Z in B. Mathematically, the principle of the algorithm is based on the resolution of an optimization problem which provides an optimal solution γ (as called in the related articles) that transfers the distribution of Y in A to the distribution of Z in B (or conversely, according to the sense of the transport)and can be so interpreted as an estimator of the joint distribution (Y, Z) in A (or B respetively). According to this result, a second step of the algorithm provides individual predictions of Y in B (resp. of Z in A, or both, depending on the choice specified by user in the argument `which.DB`). Two possible approaches are available depending on the argument `indiv.method`:

- When `indiv.method = "sequential"`, a nearest neighbor procedure is applied. This corresponds to the use of the function `indiv_grp_closest` implemented in the function `OT_outcome`.
- When `indiv.method = "optimal"`, a linear optimization problem is solved to determine the individual predictions that minimize the sum of the individual distances in A (resp. in B) with the modalities of Z in B (resp. Y in A). This approach is applied via the function `indiv_grp_optimal` implemented in the function `OT_outcome`.

This algorithm supposes the respect of the two following assumptions:

1. Y must follow the same distribution in A and B. In the same way, Z follows the same distribution in the two databases.
2. The conditional distribution $(Y|X)$ must be identical in A and B. Respectively, $(Z|X)$ is supposed identical in A and B.

Because the first assumption can be too strong in some situations, a relaxation of the constraints of marginal distribution is possible using the argument `maxrelax`. When `indiv.method = "sequential"` and `maxrelax = 0`, the algorithm called `OUTCOME` (see (1) and (2)) is applied. In all other situations, the algorithm applied corresponds to an algorithm called `R_OUTCOME` (see (2)). A posteriori estimates of conditional probabilities $P[Y|X, Z]$ and $P[Z|X, Y]$ are available for each profile of covariates (see the output objects `estimatorYB` and `estimatorZA`). Estimates of γ are also available according to the desired direction of the transport (from A to B and/or conversely. See γ_A and γ_B).

C. EXPECTED STRUCTURE FOR THE INPUT DATABASE

The input database is a data.frame that must be saved in a specific form by users:

- Two overlayed databases containing a common column of database identifiers (A and B, 1 or 2, by examples, encoded in numeric or factor form)
- A column corresponding to the target variable with its specific encoding in A (For example a factor Y encoded in n_Y levels, ordered or not, with NAs in the corresponding rows of B)
- A column corresponding to the second target outcome with its specific endoded in B (For example a factor Z in n_Z levels, with NAs in rows of A)
- The order of the variables in the database have no importance but the column indexes related to the three columns previously described (ie ID, Y and Z) must be rigorously specified in the argument `index_DB_Y_Z`.
- A set of shared common covariates (at least one but more is recommended) of any type, complete or not (provided that the number of covariates exceeds 1) is required.

The function `merge_dbs` is available in this package to assist user in the preparation of their databases, so please, do not hesitate to use it beforehand if necessary.

Remarks about the target variables:

- A target variable can be of categorical type, but also discrete, stored in factor, ordered or not. Nevertheless, notice that, if the variable is stored in numeric it will be automatically converted in ordered factors.
- If a target outcome is incomplete, the corresponding rows will be automatically dropped during the execution of the function.

The type of each variables (including ID, Y and Z) of the database must be rigorously specified once, in one of the four arguments `quanti`,`nominal`, `ordinal` and `logic`.

D. TRANSFORMATIONS OF CONTINUOUS COVARIATES

The function `OT_outcome` integrates in its syntax a process dedicated to the categorization of continuous covariates. For this, it is necessary to rigorously fill in the arguments `convert.num` and `convert.clss`. The first one informs about the indexes in database of the continuous variables to transform in ordered factor while the second one specifies the corresponding number of desired balanced levels (for unbalanced levels, users must do transformations by themselves). Therefore `convert.num` and `convert.clss` must be vectors of same length, but if the length of `convert.num` exceeds 1, while the length of `convert.clss` is 1, then, by default, all the covariates to convert will have the same number of classes, that corresponds to the value specified in the argument `convert.clss`. Please notice that only covariates can be transformed (not outcomes) and missing informations are not taken into account for the transformations. Moreover, all the indexes informed in the argument `convert.num` must also be informed in the argument `quanti`.

E. INFORMATIONS ABOUT DISTANCE FUNCTIONS

Each individual (or row) of a given database is here characterized by their covariates, so the distance between two individuals or groups of individuals depends on similarities between covariates according to the distance function chosen by user (via the argument `dist.choice`). Actually four distance functions are implemented in `OT_outcome` to take into account the most frequently encountered situation (see (3)):

- the Manhattan distance ("M")

- the Euclidean distance ("E")
- the Gower distance for mixed data (see (4): "G")
- the Hamming distance for binary data ("H")

Moreover, it is also possible to directly apply the first three distances mentioned on coordinates extracted from a multivariate analysis (Factor Analysis for Mixed Data, see (5)) applied on raw covariates using the arguments FAMD.coord and FAMD.perc. This method is used (1).

As a decision rule, for a given profile of covariates P_j, an individual i will be considered as a neighbor of P_j if $dist(i, P_j) <$ prox.dist $\times\, max(dist(i, P_j))$ where $prox.dist$ must be fixed by user.

F. INFORMATIONS ABOUT THE SOLVER

The argument solvR permits user to choose the solver of the optimization algorithm. The default solver is "glpk" that corresponds to the GNU Linear Programming Kit (see (6) for more details). Moreover, the function actually uses the R optimization infrastructure of the package **ROI** which offers a wide choice of solver to users by easily loading the associated plugins of **ROI** (see (7)).

For more details about the algorithms integrated in OT_outcome, please consult (1) and (2).

Value

A "otres" class object of 9 elements:

time_exe	the running time of the function
gamma_A	a matrix corresponding to an estimation of the joint distribution of (Y, Z) in A
gamma_B	a matrix corresponding to an estimation of the joint distribution of (Y, Z) in B
profile	a data.frame that gives all details about the remaining P profiles of covariates. These informations can be linked to the estimatorZA and the estimatorYB objects for a better interpretation of the results.
res_prox	the outputs of the function proxim_dist
estimatorZA	an array that corresponds to estimates of the probability distribution of Z conditional to X and Y in database A. The number of rows of each table corresponds to the total number of profiles of covariates. The first dimension of each table (rownames) correspond to the profiles of covariates sorted by order of appearance in the merged database. The second dimension of the array (columns of the tables) corresponds to the levels of Y while the third element corresponds to the levels of Z.
estimatorYB	an array that corresponds to estimates of the probability distribution of Y conditional to X and Z in database B. The number of rows of each table corresponds to the total number of profiles of covariates. The first dimension of each table (rownames) correspond to the profiles of covariates sorted by order of appearance in the merged database. The second dimension of the array (columns of the tables) corresponds to the levels of Z while the third element corresponds to the levels of Y.
DATA1_OT	the database A with the individual predictions of Z using an optimal transportation algorithm (OUTCOME) or R-OUTCOME
DATA2_OT	the database B with the individual predictions of Y using an optimal transportation algorithm (OUTCOME) or R-OUTCOME

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615
3. Anderberg, M.R. (1973), Cluster analysis for applications, 359 pp., Academic Press, New York, NY, USA.
4. Gower J.C. (1971). A general coefficient of similarity and some of its properties. Biometrics, 27, 623–637.
5. Pages J. (2004). Analyse factorielle de donnees mixtes. Revue Statistique Appliquee. LII (4). pp. 93-111.
6. Makhorin A (2011). GNU Linear Programming Kit Reference Manual Version 4.47.http://www.gnu.org/software/glpk/
7. Theussl S, Schwendinger F, Hornik K (2020). ROI: An Extensible R Optimization Infrastructure.Journal of Statistical Software,94(15), 1-64. doi: 10.18637/jss.v094.i15

See Also

transfo_dist,proxim_dist, avg_dist_closest, indiv_grp_closest, indiv_grp_optimal

Examples

```
### Using a sample of simu_data dataset
### Y and Z are a same variable encoded in 2 different forms:
### (3 levels for Y and 5 levels for Z)
#--------
data(simu_data)
simu_dat  = simu_data[c(1:200,301:500),]

### An example of OUTCOME algorithm that uses:
#-----
# - A nearest neighbor procedure for the estimation of individual predictions
# - The Manhattan distance function
# - 90% of individuals from each modalities to calculate average distances
#   between individuals and modalities
# Predictions are assessed for Y in B and Z in A
#-----

try1 = OT_outcome(simu_dat, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "M", maxrelax = 0,
```

```
                  indiv.method = "sequential")
head(try1$DATA1_OT)  # Part of the completed database A
head(try1$DATA2_OT)  # Part of the completed database B

head(try1$estimatorZA[,,1])
# ... Corresponds to P[Z = 1|Y,P1] when P1 corresponds to the 1st profile of covariates (P_1)
# detailed in the 1st row of the profile object:
try1$profile[1,]   # Details of P_1

# So estimatorZA[1,1,1]= 0.2 corresponds to an estimation of:
# P[Z = 1|Y=[20-40],Gender_2=0,Treatment_2=1,Treatment_3=0,Smoking_2=1,Dosage=3,Age=65.44]
# Thus, we can conclude that all individuals with the P_1 profile of covariates have
# 20% of chance to be affected to the 1st level of Z in database A.
# ... And so on, the reasoning is the same for the estimatorYB object.




### An example of OUTCOME algorithm with same conditions as the previous example, excepted that;
# - Only the individual predictions of Y in B are required
# - The continuous covariates "age" (related index = 8) will be converted in an ordinal factors
#   of 3 balanced classes (tertiles)
# - The Gower distance is now used
###-----

try2 = OT_outcome(simu_dat, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "G", maxrelax = 0,
                  convert.num = 8, convert.clss = 3,
                  indiv.method = "sequential", which.DB = "B")


### An example of OUTCOME algorithm with same conditions as the first example, excepted that;
# - Only the individual predictions of Z in A are required
# - The continuous covariates "age" (related index = 8) will be converted in an ordinal factors
#   of 3 balanced classes (tertiles)
# - Here, the Hamming distance can be applied because, after conversion, all covariates are factors.
#  Disjunctive tables of each covariates will be automatically used to work with a set of binary
#   variables.
###-----

try3 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "H", maxrelax = 0,
                  convert.num = 8, convert.clss = 3,
                  indiv.method = "sequential",which.DB = "B")


### An example of R-OUTCOME algorithm using:
# - An optimization procedure for individual predictions on the 2 databases
# - The Manhattan distance
# - Raw covariates
###-----
try4 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "M", maxrelax = 0,
```

```
                  indiv.method = "optimal")


### An example of R-OUTCOME algorithm with:
# - An optimization procedure for individual predictions on the 2 databases
# - The use of Euclidean distance on coordinates from FAMD
# - Raw covariates
###-----

try5 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "E",
                  FAMD.coord = "YES", FAMD.perc = 0.8,
                  indiv.method = "optimal")


### An example of R-OUTCOME algorithm with relaxation on marginal distributions and:
# - An optimization procedure for individual predictions on the 2 databases
# - The use of the euclidean distance
# - An arbitrary coefficient of relaxation
# - Raw covariates
#-----

try6 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "E", maxrelax = 0.4,
                  indiv.method = "optimal")
```

---

power_set *power_set()*

Description

A function that gives the power set $P(S)$ of any non empty set S.

Usage

```
power_set(n, ordinal = FALSE)
```

Arguments

n	an integer. The cardinal of the set
ordinal	a boolean. If TRUE the power set is only composed of subsets of consecutive elements, FALSE (by default) otherwise.

Value

A list of $2^n - 1$ subsets (The empty set is excluded)

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

Devlin, Keith J (1979). Fundamentals of contemporary set theory. Universitext. Springer-Verlag

Examples

```
# Powerset of set of 4 elements
fam  =  power_set(4)

# Powerset of set of 4 elements by only keeping
# subsets of consecutive elements
fam2 =  power_set(4,ordinal = TRUE)
```

`proxim_dist` *proxim_dist()*

Description

`proxim_dist` computes the pairwise distance matrix of a database and cross-distance matrix between two databases according to various distances used in the context of data fusion.

Usage

```
proxim_dist(data_file, indx_DB_Y_Z = 1:3, norm = "E", prox = 0.3)
```

Arguments

`data_file`	a data.frame corresponding ideally to an output object of the function `transfo_dist`. Otherwise this data.frame is the result of two overlayed databases with a column of database identifier ("A" and "B", 1 and 2, for example), a target variable (called Y by example) only known in the first database, a target variable (Z) only stored in the second database, such that Y and Z summarize a same information differently encoded in the two databases and set of common covariates (at least one) of any type. The order of the variables in the data.frame have no importance. The type of the covariates must be in accordance with the chosen distance function in the `norm` option.
`indx_DB_Y_Z`	a vector of three column indexes corresponding to the database identifier, the target variable of the above database and the target variable of the below database. The indexes must be declared in this specific order.
`norm`	a character string indicating the choice of the distance function. This latest depends on the type of the common covariates: the Hamming distance for binary covariates only (`norm` = "H"), the Manhattan distance ("M", by default) and the euclidean distance ("E") for continuous covariates only, or the Gower distance for mixed covariates ("G").

prox — a ratio (betwen 0 and 1) used to calculate the distance threshold below which an individual (a row or a given statistical unit) is considered as a neighbor of a given profile of covariates. 0.3 is the default value.

Details

This function is the first step of a family of algorithms that solve recoding problems of data fusion using optimal transportation theory (see the details of these corresponding models OUTCOME, R_OUTCOME, JOINT and R_JOINT in (1) and (2)). The function proxim_dist is directly implemented in the functions OT_outcome and OT_joint but can also be used separately as long as the input database has as suitable structure. Nevertheless, its preparation will have to be rigorously made in two steps detailled in the following sections.

A. EXPECTED STRUCTURE FOR THE INPUT DATABASE

Firsly, the initial database required is a data.frame that must be prepared in a specific form by users. From two separate databases, the function merge_dbs available in this package can assist users in this initial merging, nevertheless notice that this preliminary transformation can also be made directly by following the imposed structure described below: two overlayed databases containing a common column of database identifiers (A and B for examples, encoded in numeric or factor form), a column corresponding to the target variable with its specific encoding in A (for example a factor Y encoded in n_Y levels, ordered or not, with NAs in the corresponding rows of B), a column corresponding to the same variable with its specific endoded in B (for example a factor Z in n_Z levels, with NAs in database A), and a set of shared covariates (at least one) between the two databases.

The order of these variables in the database have no importance but the column indexes related to database identifier, Y and Z, must be specified in the indx_DB_Y_Z option. Users can refer to the structure of the table simu_data available in the package to adapt their databases to the inital format required.

Missing values are allowed on covariates only, and are excluded from all computations involving the rows within which they occur. In the particular case where only one covariate with NAs is used, we recommend working with imputed or complete case only to avoid the presence of NA in the distance matrix that will be computed a posteriori. If the database counts many covariates and some of them have missing data, user can keep them or apply beforehand the imput_cov function on data.frame to deal with this problem.

B. DISTANCE FUNCTIONS AND TYPES OF COVARIATES

In a second step, the shared variables of the merged database will have to be encoded according to the choice of the distance function fixed by user, knowing that it is also frequent that it is the type of the variables which fixes the distance function to choose. The function transfo_dist is available in the package to assist users in this task but a user can also decide to make this preparation by themselves. Thus, with the Euclidean or Manhattan distance ((3), norm = "E" or "M"), if all types of variables are allowed, logical variables are transformed in binary variables, and categorical variables (factors ordered or not) are replaced by their related disjunctive tables (the function transfo_quali can make these specific transformations). The Hamming distance (norm = "H") only requires binary variables (all other forms are not allowed). In this context, continuous variables could have been converted in factor of k levels ($k > 2$) beforehand. The categorical covariates are then transformed in disjunctive tables (containing the $(k-1)$ corresponding binary variables) before use. With this distance, categorical variables are also transformed in disjunctive tables. Notice that, using the Hamming distance could be quite long in presence of NAs on covariates. Finally,

the Gower distance ((4), norm = "G") uses the (gower.dist) function (5) and so allows logical, categorical and numeric variables without preliminary transformations.

In conclusion, the structure of the data.frame required in input of the function proxim_dist corresponds to two overlayed databases with two target outcomes and a set of shared covariates whose encodings depend on the distance function choosen by user.

If some columns are excluded when computing an Euclidean, Manhattan, or Hamming distance between two rows, the sum is scaled up proportionally to the number of columns used in the computation as proposed by the standard (dist) function. If all pairs are excluded when computing a particular distance, instead of putting NA in the corresponding cell of the distance matrix, the process stops and an object listing the problematic rows is proposed in output. It suggests users to remove these rows before running the process again or impute NAs related to these rows (see (6) for more details).

C. PROFILES OF COVARIATES AND OUTPUT DETAILS

Whatever the type (mixed or not) and the number of covariates in the data.frame of interest, the function proxim_dist firstly detects all the possible profiles (or combinations) of covariates from the two databases, and saves them in the output profile. For example, assuming that a data.frame in input (composed of two overlayed data.frames A and B) have three shared binary covariates (identically encoded in A and B) so the sequences 011 and 101 will be considered as two distinct profiles of covariates. If each covariate is a factor of n_1, n_2 and n_3 levels respectively, so it exists at most $n_1 \times n_2 \times n_3$ possible profiles of covariates. This number is considered as a maximum here because only the profiles of covariates met in at least one of the two databases will be kept for the study.

proxim_dist classifies individuals from the two databases according to their proximities to each profile of covariates and saves the corresponding indexes of rows from A and B in two lists indXA and indXB respectively. indXA and indXB thus contain as many objects as covariates profiles and the proximity between a given profile and a given individual is defined as follows. The function also provides in output the list of all the encountered profiles of covariates. As a decision rule, for a given profile of covariates P_j, an individual i will be considered as a neighbor of P_j if $dist(i, P_j) < prox \times max(dist(i, P_j))$ where prox will be fixed by user. Set the value 0 to the prox parameter assures that each individual of A (and B respectively) is exactly the profile of one profile of covariates. Therefore, it is not recommended in presence of continuous coavariates. Conversely, assign the value 1 to prox is not recommended because it assumes that each individual is neighbor with all the encountered profiles of covariates.

Value

A list of 16 elements (the first 16 detailed below) is returned containing various distance matrices and lists useful for the algorithms that used Optimal Transportation theory. Two more objects (the last two of the following list) will be returned if distance matrices contain NAs.

FILE_NAME	a simple reminder of the name of the raw database
nA	the number of rows of the first database (A)
nB	the number of rows of the second database (B)
Xobserv	the subset of the two overlayed databases composed of the shared variables only
profile	the different encountered profiles of covariates according to the data.frame
Yobserv	the numeric values of the target variable in the first database

Zobserv	the numeric values of the target variable in the second database
D	a distance matrix corresponding to the computed distances between individuals of the two databases
Y	the n_Y levels of the target variable in numeric form, in the first database
Z	the n_Z levels of the target variable in numeric form, in the second database
indY	a list of n_Y groups of individual (or row) numbers where each group corresponds to the individuals indexes related to a given level of Y in the first database
indZ	a list of n_Z groups of individual (or row) numbers where each group corresponds to the individuals indexes related to a given level of Z in the second database
indXA	a list of individual (row) indexes from the first database, sorted by profiles of covariates according to their proximities. See the Details part for more information
indXB	a list of individual (row) indexes from the second database, sorted by profiles of covariates according to their proximities. See the Details part for more information
DA	a distance matrix corresponding to the pairwise distances between individuals of the first database
DB	a distance matrix corresponding to the pairwise distances between individuals of the second database
ROWS_TABLE	combinations of row numbers of the two databases that generate NAs in D
ROWS_TO_RM	number of times a row of the first or second database is involved in the NA process of D

Author(s)

Gregory Guernec, Valerie Gares, Jeremy Omer

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615
3. Anderberg, M.R. (1973), Cluster analysis for applications, 359 pp., Academic Press, New York, NY, USA.
4. Gower, J. C. (1971). A general coefficient of similarity and some of its properties. Biometrics, 27, 623–637.
5. D'Orazio M. (2015). Integration and imputation of survey data in R: the StatMatch package. Romanian Statistical Review, vol. 63(2)
6. Borg, I. and Groenen, P. (1997) Modern Multidimensional Scaling. Theory and Applications. Springer.

See Also

transfo_dist, imput_cov, merge_dbs, simu_data

Examples

```
data(simu_data)
### The covariates of the data are prepared according to the chosen distance
### using the transfo_dist function

### Ex 1: The Manhattan distance

try1 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "M")
res1 = proxim_dist(try1, norm = "M")  # try1 compatible with norm = "E" for Euclidean


### Ex 2: The Euclidean and Manhattan distance applied on coordinates from FAMD

try2 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "FAMD",info = 0.80)
res2_E  = proxim_dist(try2,norm = "E")

res2_M  = proxim_dist(try2, norm = "M")


### Ex 3: The Gower distance with mixed covariates

try3 = transfo_dist(simu_data[c(1:100,301:400),],quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "G")
res3 = proxim_dist(try3, norm = "G")


### Ex 4a: The Hamming distance with binary (but incomplete) covariates only

# categorization of the continuous covariates age by tertiles
try4 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),ordinal = c(2,6),
                    convert_num = 8, convert_clss = 3, prep_choice = "H")
res4 = proxim_dist(try4, norm = "H")
# Be patient ... It could take few minutes

### Ex 4b: The Hamming distance with complete cases on nominal and ordinal covariates only
simu_data_CC = simu_data[(!is.na(simu_data[,5]))&(!is.na(simu_data[,6]))
                         &(!is.na(simu_data[,7])),1:7]
try4b = transfo_dist(simu_data_CC,quanti = 3, nominal = c(1,4:5,7),ordinal = c(2,6),
                     prep_choice = "H")
res4b = proxim_dist(try4b, norm = "H")


### Ex 5: PARTICULAR CASE, If only one covariate with no NAs

try5 = try1[,c(1:3,7)]           # Only Smoking variable
```

```
try6 = try5[!is.na(try5[,4]),]   # Keep complete case
res6_M = proxim_dist(try6,norm = "M", prox = 0.10)

res7_H = proxim_dist(try6,norm = "H") # Hamming


### Ex 6: PARTICULAR CASE, many covariates but NAs in distance matrix

# We generated NAs in the try1 object so that:
# dist(A4,B102) and dist(A122,B102) returns NA whatever the norm chosen:
try7 = try1
try7[4,7:9]   = NA
try7[122,6:9] = NA
try7[300+102,4:6] = NA
res7 = proxim_dist(try7,norm = "M")
# The process stopped indicates 2 NAs and the corresponding row numbers
# The 2nd output of res7 indicates that removing first the 102th row of the database
# B is enough to solve the pb:
try8 = try7[-402,]
res8 = proxim_dist(try8,norm = "M")
```

---

select_pred *select_pred()*

Description

Selection of a subset of non collinear predictors having relevant relationships with a given target outcome using a random forest procedure.

Usage

```
select_pred(
  databa,
  Y = NULL,
  Z = NULL,
  ID = 1,
  OUT = "Y",
  quanti = NULL,
  nominal = NULL,
  ordinal = NULL,
  logic = NULL,
  convert_num = NULL,
  convert_clss = NULL,
  thresh_cat = 0.3,
  thresh_num = 0.7,
  thresh_Y = 0.2,
  RF = TRUE,
```

```
  RF_ntree = 500,
  RF_condi = FALSE,
  RF_condi_thr = 0.2,
  RF_SEED = sample(1:1e+06, 1)
)
```

Arguments

databa	a data.frame with a column of identifiers (of row or of database in the case of two concatened databases), an outcome, and a set of predictors. The number of columns can exceed the number of rows.
Y	the label of a first target variable with quotes
Z	the label of a second target variable with quotes when databa is the result of two overlayed databases.
ID	the column index of the database identifier (The first column by default) in the case of two concatened databases, a row identifier otherwise
OUT	a character that indicates the outcome to predict in the context of overlayed databases. By default, the outcome declared in the argument Y is predicted. Another possible outcome to predict can be set with the related argument Z.
quanti	a vector of integers corresponding to the column indexes of all the numeric predictors.
nominal	a vector of integers which corresponds to the column indexes of all the categorical nominal predictors.
ordinal	a vector of integers which corresponds to the column indexes of all the categorical ordinal predictors.
logic	a vector of integers indicating the indexes of logical predictors. No index remained by default
convert_num	a vector of integers indicating the indexes of quantitative variables to convert in ordered factors. No index remained by default. Each index selected has to be defined as quantitative in the argument quanti.
convert_clss	a vector of integers indicating the number of classes related to each transformation of quantitative variable in ordered factor. The length of this vector can not exceed the length of the argument convert_num. Nevertheless, if length(convert_num) > 1 and length(convert_clss) = 1, all quantitative predictors selected for discretization will have by default the same number of classes.
thresh_cat	a threshold associated to the Cramer's V coefficient (= 0.30 by default)
thresh_num	a threshold associated to the Spearman's coefficient of correlation (= 0.70 by default)
thresh_Y	a threshold linked to the RF approach, that corresponds to the minimal cumulative percent of importance measure required to be kept in the final list of predictors.
RF	a boolean sets to TRUE (default) if a random forest procedure must be applied to select the best subset of predictors according to the outcome.Otherwise, only pairwise associations between predictors are used for the selection.

RF_ntree	the number of bootsrap samples required from the row datasource during the random forest procedure
RF_condi	a boolean specifying if the conditional importance measures must be assessed from the random forest procedure (TRUE) rather than the standard variable importance measures (FALSE by default)
RF_condi_thr	a threshold linked to (1 - pvalue) of an association test between each predictor X and the other variables, given that a threshold value of zero will include all variables in the computation of the conditional importance measure of X (0.20 is the default value). Conversely, a larger threshold will only keeps the subset of variables that is strongly correlated to X for the computation of the variable importance measure of X.
RF_SEED	an integer used as argument by the set.seed() for offsetting the random number generator (random integer by default). This value is only used for RF method.

Details

The select_pred function provides several tools to identify, on the one hand, the relationships between predictors, by detecting especially potential problems of collinearity, and, on the other hand, proposes a parcimonious subset of relevant predictors (of the outcome) using appropriate random forest procedures. The function which can be used as a preliminary step of prediction in regression areas is particularly adapted to the context of data fusion by providing relevant subsets of predictors (the matching variables) to algorithms dedicated to the solving of recoding problems.

A. REQUIRED STRUCTURE FOR THE DATABASE

The expected input database is a data.frame that especially requires a specific column of row identifier and a target variable (or outcome) having a finite number of values or classes (ordinal, nominal or discrete type). Notice that if the chosen outcome is in numeric form, it will be automatically converted in ordinal type. The number of predictors is not a constraint for select_pred (even if, with less than three variables a process of variables selection has no real sense...), and can exceed the number of rows (no problem of high dimensionality here). The predictors can be continuous (quantitative), boolean, nominal or ordinal with or without missing values. In presence of numeric variables, users can decide to discretize them or a part of them by themselves beforehand. They can also choose to use the internal process directly integrated in the function. Indeed, to assist users in this task, two arguments called convert_num and convert_clss dedicated to these transformations are available in input of the function. These options make the function select_pred particularly adapted to the function OT_joint which only allows data.frame with categorical covariates. With the argument convert_num, users choose the continuous variables to convert and the related argument convert_clss specifies the corresponding number of classes chosen for each discretization. It is the reason why these two arguments must be two vectors of indexes of same length. Nevertheless, an unique exception exists when convert_clss is equalled to a scalar S. In this case, all the continuous predictors selected for conversion will be discretized with a same number of classes S. By example, if convert_clss = 4, all the continuous variables specified in the convert_num argument will be discretized by quartiles. Moreover, notice that missing values from incomplete predictors to convert are not taken into account during the conversion, and that each predictor specified in the argument convert_num must be also specified in the argument quanti. In this situation, the label of the outcome must be entered in the argument Y, and the arguments Z and OUT must keep their default values. Finally, the order of the column indexes related to the identifier and the outcome have no importance.

For a better flexibility, the input database can also be the result of two overlayed databases. In this case, the structure of the database must be similar to those observed in the datasets simu_data and tab_test available in the package with a column of database identifier, one target outcome by database (2 columns), and a subset of shared predictors. Notice that, overlaying two separate databases can also be done easily using the function merge_dbs beforehand. The labels of the two outcomes will have to be specified in the arguments Y for the top database, and in Z for the bottom one. Notice also that the function select_pred deals with only one outcome at a time that will have to be specified in the argument OUT which must be equalled to "Y" for the study of the top database or "Z" for the study of the bottom one.

Finally, whatever the structure of the database declared in input, each column index related to the database variable must be entered once (and only once) in one of the following four arguments: quanti, nominal, ordinal, logic.

B. PAIRWISE ASSOCIATIONS BETWEEN PREDICTORS

In a first step of process, select_pred calculates standard pairwise associations between predictors according to their types.

1. Between categorical predictors (ordinal, nominal and logical): Cramer's V (and Bias-corrected Cramer's V, see (1) for more details) are calculated between categorical predictors and the argument thres_cat fixed the associated threshold beyond which two predictors can be considered as redundant. A similar process is done between the target variable and the subset of categorical variables which provides in output a first table ranking the top scoring predictors. This table summarizes the ability of each variable to predict the target outcome.
2. Between continuous predictors: If the ordinal and logic arguments differ from NULL, all the corresponding predictors are beforehand converted in rank values. For numeric (quantitative), logical and ordinal predictors, pairwise correlations between ranks (Spearman) are calculated and the argument thresh_num fixed the related threshold beyond which two predictors can be considered as redundant. A similar process is done between the outcome and the subset of discrete variables which provides in output, a table ranking the top scoring predictor variates which summarizes their abilities to predict the target. In addition, the result of a Farrar and Glauber test is provided. This test is based on the determinant of the correlation matrix of covariates and the related null hypothesis of the test corresponds to an absence of collinearity between them (see (2) for more details about the method). In presence of a large number of numeric covariates and/or ordered factors, the approximate Farrar-Glauber test, based on the normal approximation of the null distribution is more adapted and its result is also provided in output. These two tests are highly sensitive and, by consequence, it suggested to consider these results as simple indicators of collinearity between predictors rather than an essential condition of acceptability.

If the initial number of predictors is not too important, these informations can be sufficient to the user for the visualization of potential problems of collinearity and for the selection of a subset of predictors (RF = FALSE). It is nevertheless often necessary to complete this visualization by an automatical process of selection like the Random Forest approach (see Breiman 2001, for a better understanding of the method) linked to the function select_pred (RF = TRUE).

C. RANDOM FOREST PROCEDURE

As a final step of the process, a random forest approach (RF(3)) is here prefered (to regression models) for two main reasons: RF methods allow notably the number of variables to exceed the number of rows and remain applicable whatever the types of covariates considered. The function

select_pred integrates in its algorithm the functions cforest and varimp of the package **party** (Hothorn, 2006) and so gives access to their main arguments.

A RF approach generally provides two types of measures for estimating the mean variable importance of each covariate in the prediction of an outcome: the Gini importance and the permutation importance. These measurements must be used with caution, by taking into account the following constraints:

1. The Gini importance criterion can produce bias in favor of continuous variables and variables with many categories. To avoid this problem, only the permutation criterion is available in the function.
2. The permutation importance criterion can overestimate the importance of highly correlated predictors.

The function select_pred proposes three different scenarios according to the types of predictors:

1. The first one consists in boiling down to a set of categorical variables (ordered or not) by discretizing all the continuous predictors beforehand, using the internal convert_num argument or another one, and then works with the conditional importance measures (RF_condi = TRUE) which give unbiased estimations. In the spirit of a partial correlation, the conditional importance measure related to a variable X for the prediction of an outcome Y, only uses the subset of variables the most correlated to X for its computation. The argument RF_condi_thr that corresponds exactly to the argument threshold of the function varimp, fixes a ratio below which a variable Z is considered sufficiently correlated to X to be used as an adjustment variable in the computation of the importance measure of X (In other words, Z is included in the conditioning for the computation, see (4) and (5) for more details). A threshold value of zero will include all variables in the computation of conditional importance measure of each predictor X, while a threshold < 1, will only include a subset of variables. Two remarks related to this method: firstly, notice that taking into account only subsets of predictors in the computation of the variable importance measures could lead to a relevant saving of execution time. Secondly, because this approach does not take into account incomplete information, the method will only be applied to complete data (incomplete rows will be temporarily removed for the study).
2. The second possibility, always in presence of mixed types predictors, consists in the execution of two successive RF procedures. The first one will be used to select an unique candidate in each susbset of correlated predictors (detecting in the 1st section), while the second one will extract the permutation measures from the remaining subset of uncorrelated predictors (RF_condi = FALSE, by default). This second possibility has the advantage to work in presence of incomplete predictors.
3. The third scenario consists in running a first time the function without RF process (RF = FALSE), and according to the presence of highly correlated predictors or not, users can choose to extract redundant predictors manually and re-runs the function with the subset of remaining non-collinear predictors to avoid potential biases introduced by the standard permutations measures.

The three scenarios finally lead to a list of uncorrelated predictors of the outcome sorted in importance order. The argument thresh_Y corresponds to the minimal percent of importance required (and fixed by user) for a variable to be considered as a reliable predictor of the outcome. Finally, because all random forest results are subjects to random variation, users can check whether the same importance ranking is achieved by varying the random seed parameter (RF_SEED) or by increasing the number of trees (RF_ntree).

Value

A list of 14 (if RF = TRUE) or 11 objects (Only the first ten objects if RF = FALSE) is returned:

seed	the random number generator related to the study
outc	the identifier of the outcome to predict
thresh	a summarize of the different thresholds fixed for the study
convert_num	the labels of the continuous predictors transformed in categorical form
DB_USED	the final database used after potential transformations of predictors
vcrm_OUTC_cat	a table of pairwise associations between the outcome and the categorical predictors (Cramer's V)
cor_OUTC_num	a table of pairwise associations between the outcome and the continuous predictors (Rank correlation)
vcrm_X_cat	a table of pairwise associations between the categorical predictors (Cramer's V)
cor_X_num	a table of pairwise associations between the continuous predictors (Cramer's V)
FG_test	the results of the Farrar and Glauber tests, with and without approximation form
collinear_PB	a table of predictors with problem of collinearity according to the fixed thresholds
drop_var	the labels of predictors to drop after RF process (optional output: only if RF=TRUE)
RF_PRED	the table of variable importance measurements, conditional or not, according to the argument condi_RF (optional output: Only if RF=TRUE)
RF_best	the labels of the best predictors selected (optional output: Only if RF=TRUE) according to the value of the argument thresh_Y

Author(s)

Gregory Guernec

<gregory.guernec@inserm.fr>

References

1. Bergsma W. (2013). A bias-correction for Cramer's V and Tschuprow's T. Journal of the Korean Statistical Society, 42, 323–328.
2. Farrar D, and Glauber R. (1968). Multicolinearity in regression analysis. Review of Economics and Statistics, 49, 92–107.
3. Breiman L. (2001). Random Forests. Machine Learning, 45(1), 5–32.
4. Hothorn T, Buehlmann P, Dudoit S, Molinaro A, Van Der Laan M (2006). "Survival Ensembles." Biostatistics, 7(3), 355–373.
5. Strobl C, Boulesteix A-L, Kneib T, Augustin T, Zeileis A (2008). Conditional Variable Importance for Random Forests. BMC Bioinformatics, 9, 307. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-307

See Also

simu_data, tab_test, OT_joint

Examples

```
### Example 1
#-----
# - From two overlayed databases: using the table simu_data
# - Searching for the best predictors of "Yb1"
# - Using the row database
# - The RF approaches are not required
#-----

data(simu_data)
test_DB1 = select_pred(simu_data,Y = "Yb1", Z = "Yb2", ID = 1, OUT = "Y",
                       quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                       thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                       RF = FALSE)

### Example 2
#-----
# - With same conditions as example 1
# - Searching for the best predictors of "Yb2"
#-----

test_DB2 = select_pred(simu_data,Y = "Yb1", Z = "Yb2", ID = 1, OUT = "Z",
                       quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                       thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                       RF = FALSE)


### Example 3
#-----
# - With same conditions as example 1
# - Using a RF approach to estimate the standard variable importance measures
#   and determine the best subset of predictors
# - Here a seed is required
#-----

test_DB3 = select_pred(simu_data,Y = "Yb1", Z = "Yb2", ID = 1, OUT = "Y",
                       quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                       thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                       RF = TRUE, RF_condi = FALSE, RF_SEED = 3023)

### Example 4
#-----
# - With same conditions as example 1
# - Using a RF approach to estimate the conditional variable importance measures
#   and determine the best subset of predictors
# - This approach requires to convert the numeric variables: Only "Age" here
#   discretized in 3 levels
#-----

test_DB4 = select_pred(simu_data,Y = "Yb1", Z = "Yb2", ID = 1, OUT = "Z",
                       quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
```

```
                        convert_num = 8, convert_clss = 3,
                        thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                        RF = TRUE, RF_condi = TRUE, RF_condi_thr = 0.60, RF_SEED = 3023)

### Example 5
#-----
# - Starting with a unique database
# - Same conditions as example 1
#-----
simu_A = simu_data[simu_data$DB == "A",-3]    # Base A

test_DB5 = select_pred(simu_A, Y = "Yb1",
                        quanti = 7, nominal = c(1,3:4,6), ordinal = c(2,5),
                        thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                        RF = FALSE)

### Example 6
#-----
# - Starting with an unique database
# - Using a RF approach to estimate the conditional variable importance measures
#   and determine the best subset of predictors
# - This approach requires to convert the numeric variables: Only "Age" here
#   discretized in 3 levels
#-----

simu_B = simu_data[simu_data$DB == "B",-2]    # Base B

test_DB6 = select_pred(simu_B, Y = "Yb2",
                        quanti = 7, nominal = c(1,3:4,6), ordinal = c(2,5),
                        convert_num = 7, convert_clss = 3,
                        thresh_cat = 0.30, thresh_num = 0.70, thresh_Y = 0.20,
                        RF = TRUE, RF_condi = TRUE, RF_condi_thr = 0.60, RF_SEED = 3023)
```

`simu_data` *A simulated dataset to test the functions of the OTrecod package*

Description

The first 300 rows belong to the database A, while the next 400 rows belong to the database B. Five covariates: `Gender`, `Treatment`, `Dosage`, `Smoking` and `Age` are common to both databases (same encodings). `Gender` is the only complete covariate. The variables `Yb1` and `Yb2` are the target variables of A and B respectively, summarizing a same information encoded in two different scales. that summarize a same information saved in two distinct encodings, that is why, `Yb1` is missing in the database B and `Yb2` is missing in the database A.

Usage

```
simu_data
```

Format

A data.frame made of 2 overlayed databases (A and B) with 700 observations on the following 8 variables.

DB the database identifier, a character with 2 possible classes: `A` or `B`

Yb1 the target variable of the database A, stored as factor and encoded in 3 ordered levels: `[20-40]`, `[40-60[`,`[60-80]` (the values related to the database B are missing)

Yb2 the target variable of the database B, stored as integer (an unknown scale from 1 to 5) in the database B (the values related to A are missing)

Gender a factor with 2 levels (`Female` or `Male`) and no missing values

Treatment a covariate of 3 classes stored as a character with 2% of missing values: `Placebo`, `Trt A`, `Trt B`

Dosage a factor with 4 levels and 5% of missing values: from `Dos 1` to `dos 4`

Smoking a covariate of 2 classes stored as a character and 10% of missing values: `NO` for non smoker, `YES` otherwise

Age a numeric corresponding to the age of participants in years. This variable counts 5% of missing values

Details

The purpose of the functions contained in this package is to predict the missing information on `Yb1` and `Yb2` in database A and database B using the Optimal Transportation Theory.

Missing information has been simulated to some covariates following a simple MCAR process.

Source

randomly generated

`tab_test` *A simulated dataset to test the library*

Description

A dataset of 10000 rows containing 3 covariables and 2 outcomes.

Usage

```
tab_test
```

Format

A data frame with 5000 rows and 6 variables:

ident identifier, 1 or 2

Y1 outcome 1 with 2 levels, observed for ident=1 and unobserved for ident=2

Y2 outcome 2 with 4 levels, observed for ident=2 and unobserved for ident=1

X1 covariate 1, integer

X2 covariate 2, integer

X3 covariate 3, integer

Source

randomly generated

`transfo_dist` *transfo_dist()*

Description

This function prepares an overlayed database for data fusion according to the distance function chosen to evaluate the proximities between units.

Usage

```
transfo_dist(
  DB,
  index_DB_Y_Z = 1:3,
  quanti = NULL,
  nominal = NULL,
  ordinal = NULL,
  logic = NULL,
  convert_num = NULL,
  convert_clss = NULL,
  prep_choice = "E",
  info = 0.8
)
```

Arguments

`DB`	a data.frame composed of exactly two overlayed databases with a column of database identifier, two columns corresponding to a same information differently encoded in the two databases and covariates. The order of the variables have no importance.

index_DB_Y_Z	a vector of exactly three integers. The first integer must correspond to the column index of the database identifier. The second integer corresponds to the index of the target variable in the first database while the third integer corresponds to the index of column related to the target variable in the second database.
quanti	the column indexes of all the quantitative variables (database identificatier and target variables included) stored in a vector.
nominal	the column indexes of all the nominal (not ordered) variables (DB identification and target variables included) stored in a vector.
ordinal	the column indexes of all the ordinal variables (DB identification and target variables included) stored in a vector.
logic	the column indexes of all the boolean variables stored in a vector.
convert_num	the column indexes of the continuous (quantitative) variables to convert in ordered factors. All indexes declared in this argument must have been declared in the argument quanti (no conversion by default).
convert_clss	according to the argument convert_num, a vector indicating for each continuous variable to convert the corresponding desired number of levels. If the length of the argument convert_num exceeds 1 while the length of convert_clss is equal to 1 (only one integer), each discretization will count the same number of levels.
prep_choice	a character string corresponding to the distance function chosen between: the euclidean distance ("E", by default), the Manhattan distance ("M"), the Gower distance ("G"), the Hamming (also called binary) distance ("H"), and a distance computed from principal components of a factor analysis of mixed data ("FAMD").
info	a ratio (between 0 and 1, 0.8 is the default value) that corresponds to the minimal part of variability that must be taken into account by the remaining principal components of the FAMD when this approach is required. This ratio will fix the number of components that will be kept with this approach. When the argument is set to 1, all the variability is considered.

Details

A. EXPECTED STRUCTURE FOR THE INPUT DATABASE

In input of this function, the expected database is the result of an overlay between two databases A and B. This structure can be guaranteed using the specific outputs of the functions merge_dbs or select_pred. Nevertheless, it is also possible to apply directly the function transfo_dist on a raw database provided that a specific structure is respected in input. The overlayed database (A placed on top of B) must count at least four columns (in an a unspecified order of appearance in the database):

- A column indicating the database identifier (two classes or levels if factor: A and B, 1 and 2, ...)
- A column dedicated to the outcome (or target variable) of the first database and denoted Y for example. This variable can be of categorical (nominal or ordinal factor) or continuous type. Nevertheless, in this last case, a warning will appear and the variable will be automatically converted in ordered factors as a prerequisite format of the database before using data fusion algorithms.

- A column dedicated to the outcome (or target variable) of the second database and denoted Z for example. As before, this variable can be of categorical (nominal or ordinal factor) or continuous type, and the variable will be automatically converted in ordered factors as a prerequisite format of the database before using data fusion algorithms.
- At least one shared variable (same encoding in the two databases). Incomplete information is possible on shared covariates only with more than one shared covariate in the final database.

In this context, the two databases are overlayed and the information related to Y in the second database must be missing as well as the information related to Z in the first one. The column indexes related to the database identifier, Y and Z must be specified in this order in the argument `index_DB_Y_Z`. Moreover, all column indexes (including those related to identifier and target variables Y and Z) of the overlayed database (DB) must be declared once (and only once), among the arguments `quanti`, `nominal`, `ordinal`, and `logic`.

B. TRANSFORMATIONS OF CONTINUOUS COVARIATES

Because some algorithms dedicated to solving recoding problems like `JOINT` and `R-JOINT` (see (1) and/or the documentation of `OT_joint`) requires the use of no continuous covariates, the function `transfo_dist` integrates in is syntax a process dedicated to the categorization of continuous variables. For this, it is necessary to rigorously fill in the arguments `convert_num` and `convert_clss`. The first one specifies the indexes of continuous variables to transform in ordered factors while the second one assigns the corresponding desired number of levels. Only covariates should be transformed (not outcomes) and missing informations are not taken into account for the transformations. Notice that all the indexes informed in the argument `convert_num` must also be informed in the argument `quanti`.

C. TRANSFORMATIONS ON THE DATABASE ACCORDING TO THE CHOSEN DISTANCE FUNCTION

These necessary transformations are related to the type of each covariate. It depends on the distance function chosen by user in the `prep_choice` argument.

1. For the Euclidean ("E") and Manhattan ("M") distances (see (2) and (3)): all the remaining continuous variables are standardized. The related recoding to a boolean variable is 1 for `TRUE` and 0 for `FALSE`. The recoding of a nominal variable of k classes corresponds to its related disjunctive table (of (k-1) binary variables)). The ordinal variables are all converted to numeric variables (please take care that the order of the classes of each of these variables is well specified at the beginning).

2. For the Hamming ("H") distance (see (2) and (3)): all the continuous variables must be transformed beforehand in categorical forms using the internal process described in section B or via another external approach. The boolean variables are all converted in ordinal forms and then turned into binaries. The recoding for nominal or ordinal variable of k classes corresponds to its related disjunctive table (i.e (k-1) binary variables)).

3. For the Gower ("G") distance (see (4)): all covariates remain unchanged

4. Using the principal components from a factor analysis for mixed data (FAMD (5)): a factor analysis for mixed data is applied on the covariates of the database and a specific number of the related principal components is remained (depending on the minimal part of variability explained by the covariates that the user wishes to keep by varying the `info` option). The function integrates in its syntax the function `FAMD` of the package **FactoMiner** (6) using default parameters. After this step, the covariates are replaced by the remaining principal components of the FAMD, and each value corresponds to coordinates linked to each component. Please notice that this method supposed complete covariates in input, nevertheless in presence of incomplete covariates, each corresponding

rows will be dropped from the study, a warning will appear and the number of remaining rows will be indicated.

Value

A data.frame whose covariates have been transformed according to the distance function or approach (for FAMD) chosen. The columns of the data.frame could have been reordered so that the database identifier, Y and Z correspond to the first three columns respectively. Moreover the order of rows remains unchanged during the process.

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615
2. Anderberg, M.R. (1973). Cluster analysis for applications, 359 pp., Academic Press, New York, NY, USA.
3. Borg, I. and Groenen, P. (1997). Modern Multidimensional Scaling. Theory and Applications. Springer.
4. Gower, J. C. (1971). A general coefficient of similarity and some of its properties. Biometrics, 27, 623–637.
5. Pages J. (2004). Analyse factorielle de donnees mixtes. Revue Statistique Appliquee. LII (4). pp. 93-111.
6. Lê S, Josse J, Husson, F. (2008). FactoMineR: An R Package for Multivariate Analysis. Journal of Statistical Software. 25(1). pp. 1-18.

See Also

transfo_quali,merge_dbs

Examples

```
### Using the table simu_data:

data(simu_data)

# 1. the Euclidean distance (same output with Manhattan distance),
try1 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "E")
# Here Yb2 was stored in numeric: It has been automatically converted in factor

# You can also convert beforehand Yb2 in ordered factor by example:
sim_data     = simu_data
sim_data$Yb2 = as.ordered(sim_data$Yb2)
```

```
try1 = transfo_dist(sim_data,quanti = 8, nominal = c(1,4:5,7),
                    ordinal = c(2,3,6), logic = NULL, prep_choice = "E")

# 2. The Euclidean distance generated on principal components
#    by a factor analysis for mixed data (FAMD):
try2 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "FAMD")

# Please notice that this method works only with rows that have complete
# information on covariates.

# 3. The Gower distance for mixed data:
try3 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),
                    ordinal = c(2,6), logic = NULL, prep_choice = "G")

# 4. The Hamming distance:
# Here the quanti option could only contain indexes related to targets.
# Column indexes related to potential binary covariates or covariates with
# finite number of values must be include in the ordinal option.
# So in simu_data, the discretization of the variable age is required (index=8),
# using the convert_num and convert_clss arguments (for tertiles = 3):

try4 = transfo_dist(simu_data,quanti = c(3,8), nominal = c(1,4:5,7),ordinal = c(2,6),
convert_num = 8, convert_clss = 3, prep_choice = "H")


### This function works whatever the order of your columns in your database:
# Suppose that we re-order columns in simu_data:
simu_data2 = simu_data[,c(2,4:7,3,8,1)]

# By changing the corresponding indexes in the index_DB_Y_Z argument,
# we observe the desired output:
try5 = transfo_dist(simu_data2,index_DB_Y_Z = c(8,1,6),quanti = 6:7, nominal = c(2:3,5,8),
                     ordinal = c(1,4), logic = NULL, prep_choice = "E")
```

`transfo_quali` *transfo_quali()*

Description

A function that transforms a factor of n(>1) levels in (n-1) binary variables.

Usage

```
transfo_quali(x, labx = NULL)
```

Arguments

x	a factor
labx	a new label for the generated binary variables (By default the name of the factor is conserved)

Value

A matrix of (n-1) binary variables

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

Examples

```
treat     = as.factor(c(rep("A",10),rep("B",15),rep("C",12)))
treat_bin = transfo_quali(treat,"trt")
```

---

transfo_target *transfo_target()*

Description

This function prepares the encoding of the target variable before running an algorithm using optimal transportation theory.

Usage

```
transfo_target(z, levels_order = NULL)
```

Arguments

z	a factor variable (ordered or not). A variable of another type will be, by default, convert to a factor.
levels_order	a vector corresponding to the values of the levels of z. When the target is ordinal, the levels can be sorted by ascending order. By default, the initial order is remained.

Details

The function `transfo_target` is an intermediate function direcly implemented in the functions `OT_outcome` and `OT_joint`, two functions dedicated to data fusion (see (1) and (2) for details). Nevertheless, this function can also be used separately to assist user in the conversion of a target variable (outcome) according to the following rules:

- A character variable is converted in factor if the argument `levels_order` is set to NULL. In this case, the levels of the factor are assigned by order of appearance in the database.
- A character variable is converted in ordered factor if the argument `levels_order` differs from NULL. In this case, the levels of the factor correspond to those assigned in the argument.
- A factor stays unchanged if the argument `levels_order` is set to NULL. Otherwise the factor is converted in ordered factor and the levels are ordered according to the argument `levels_order`.
- A numeric variable, discrete or continuous is converted in factor if the argument `levels_order` is set to NULL, and the related levels are the values assigned in ascending order.
- A numeric variable, discrete or continuous is converted in ordered factor if the argument `levels_order` differed from NULL, and the related levels correspond to those assigned in the argument.

Value

The list returned is:

`NEW`	an object of class factor of the same length as z
`LEVELS_NEW`	the levels (ordered or not) retained for z

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
2. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615

See Also

`compare_lists`

Examples

```
y      = rnorm(100,30,10)
aa     = transfo_target(y)

newlev  = unique(as.integer(y))
bb      = transfo_target(y,levels_order = newlev)
newlev2 = newlev[-1]
cc      = transfo_target(y,levels_order = newlev2)

outco   = c(rep("A",25),rep("B",50),rep("C",25))
dd      = transfo_target(outco,levels_order = c("B","C","A"))
ee      = transfo_target(outco,levels_order = c("E","C","A","F"))
ff      = transfo_target(outco)

outco2  = c(rep("A",25),NA,rep("B",50),rep("C",25),NA,NA)
gg      = transfo_target(outco2)
hh      = transfo_target(outco2,levels_order = c("B","C","A"))
```

`verif_OT` *verif_OT()*

Description

This function proposes post-process verifications after data fusion by optimal transportation algorithms.

Usage

```
verif_OT(
  ot_out,
  group.clss = FALSE,
  ordinal = TRUE,
  stab.prob = FALSE,
  min.neigb = 1
)
```

Arguments

`ot_out`	an otres object from `OT_outcome` or `OT_joint`
`group.clss`	a boolean indicating if the results related to the proximity between outcomes by grouping levels are requested in output (`FALSE` by default).
`ordinal`	a boolean that indicates if Y and Z are ordinal (`TRUE` by default) or not. This argument is only useful in the context of groups of levels (`group.clss`=TRUE).
`stab.prob`	a boolean indicating if the results related to the stability of the algorithm are requested in output (`FALSE` by default).
`min.neigb`	a value indicating the minimal required number of neighbors to consider in the estimation of stability (1 by default).

Details

In a context of data fusion, where information from a same target population is summarized via two specific variables Y and Z (two ordinal or nominal factors with different number of levels n_Y and n_Z), never jointly observed and respectively stored in two distinct databases A and B, Optimal Transportation (OT) algorithms (see the models `OUTCOME`, `R_OUTCOME`, `JOINT`, and `R_JOINT` of the reference (2) for more details) propose methods for the recoding of Y in B and/or Z in A. Outputs from the functions `OT_outcome` and `OT_joint` so provides the related predictions to Y in B and/or Z in A, and from these results, the function `verif_OT` provides a set of tools (optional or not, depending on the choices done by user in input) to estimate:

1. the association between Y and Z after recoding
2. the similarities between observed and predicted distributions
3. the stability of the predictions proposed by the algorithm

A. PAIRWISE ASSOCIATION BETWEEN Y AND Z

The first step uses standard criterions (Cramer's V, and Spearman's rank correlation coefficient) to evaluate associations between two ordinal variables in both databases or in only one database. When the argument `group.clss = TRUE`, these informations can be completed by those provided by the function `error_group`, which is directly integrate in the function `verif_OT`. Assuming that $n_Y > n_Z$, and that one of the two scales of Y or Z is unknown, this function gives additional informations about the potential link between the levels of the unknown scale. The function proceeds to this result in two steps. Firsty, `error_group` groups combinations of modalities of Y to build all possible variables Y' verifying $n_{Y'} = n_Z$. Secondly, the function studies the fluctuations in the association of Z with each new variable Y' by using adapted comparisons criterions (see the documentation of `error_group` for more details). If grouping successive classes of Y leads to an improvement in the initial association between Y and Z then it is possible to conclude in favor of an ordinal coding for Y (rather than nominal) but also to emphasize the consistency in the predictions proposed by the algorithm of fusion.

B. SIMILARITIES BETWEEN OBSERVED AND PREDICTED DISTRIBUTIONS

When the predictions of Y in B and/or Z in A are available in the `datab` argument, the similarities between the observed and predicted probabilistic distributions of Y and/or Z are quantified from the Hellinger distance (see (1)). This measure varies between 0 and 1: a value of 0 corresponds to a perfect similarity while a value close to 1 (the maximum) indicates a great dissimilarity. Using this distance, two distributions will be considered as close as soon as the observed measure will be less than 0.05.

C. STABILITY OF THE PREDICTIONS

These results are based on the decision rule which defines the stability of an algorithm in A (or B) as its average ability to assign a same prediction of Z (or Y) to individuals that have a same given profile of covariates X and a same given level of Y (or Z respectively).

Assuming that the missing information of Z in base A was predicted from an OT algorithm (the reasoning will be identical with the prediction of Y in B, see (2) and (3) for more details), the function `verif_OT` uses the conditional probabilities stored in the object `estimatorZA` (see outputs of the functions `OT_outcome` and `OT_joint`) which contains the estimates of all the conditional probabilities of Z in A, given a profile of covariates x and given a level of $Y = y$. Indeed, each individual (or row) from A, is associated with a conditional probability $P(Z = z|Y = y, X = x)$ and averaging all the corresponding estimates can provide an indicator of the predictions stability.

The function OT_joint provides the individual predictions for subject i: $\widehat{z}_i$, $i = 1, \ldots, n_A$ according to the the maximum a posteriori rule:

$$\widehat{z}_i = \text{argmax}_{z \in \mathcal{Z}} P(Z = z | Y = y_i, X = x_i)$$

The function OT_outcome directly deduces the individual predictions from the probablities $P(Z = z|Y = y, X = x)$ computed in the second part of the algorithm (see (3)).

It is nevertheless common that conditional probabilities are estimated from too rare covariates profiles to be considered as a reliable estimate of the reality. In this context, the use of trimmed means and standard deviances is suggested by removing the corresponding probabilities from the final computation. In this way, the function provides in output a table (eff.neig object) that provides the frequency of these critical probabilities that must help the user to choose. According to this table, a minimal number of profiles can be imposed for a conditional probability to be part of the final computation by filling in the min.neigb argument.

Notice that these results are optional and available only if the argument stab.prob = TRUE. When the predictions of Z in A and Y in B are available, the function verif_OT provides in output, global results and results by database. The res.stab table can produce NA with OT_outcome output in presence of incomplete shared variables: this problem appears when the prox.dist argument is set to 0 and can be simply solved by increasing this value.

Value

A list of 7 objects is returned:

nb.profil	the number of profiles of covariates
conf.mat	the global confusion matrix between Y and Z
res.prox	a summary table related to the association measures between Y and Z
res.grp	a summary table related to the study of the proximity of Y and Z using group of levels. Only if the group.clss argument is set to TRUE.
hell	Hellinger distances between observed and predicted distributions
eff.neig	a table which corresponds to a count of conditional probabilities according to the number of neighbors used in their computation (only the first ten values)
res.stab	a summary table related to the stability of the algorithm

Author(s)

Gregory Guernec

<otrecod.pkg@gmail.com>

References

1. Liese F, Miescke K-J. (2008). Statistical Decision Theory: Estimation, Testing, and Selection. Springer
2. Gares V, Dimeglio C, Guernec G, Fantin F, Lepage B, Korosok MR, savy N (2019). On the use of optimal transportation theory to recode variables and application to database merging. The International Journal of Biostatistics. Volume 16, Issue 1, 20180106, eISSN 1557-4679. doi:10.1515/ijb-2018-0106
3. Gares V, Omer J (2020) Regularized optimal transport of covariates and outcomes in data recoding. Journal of the American Statistical Association. doi: 10.1080/01621459.2020.1775615

See Also

OT_outcome, OT_joint, proxim_dist, error_group

Examples

```
### Example 1
#-----
# - Using the data simu_data
# - Studying the proximity between Y and Z using standard criterions
# - When Y and Z are predicted in B and A respectively
# - Using an outcome model (individual assignment with knn)
#-----
data(simu_data)
try1 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "G",percent.knn = 0.90, maxrelax = 0,
                  convert.num = 8, convert.clss = 3,
                  indiv.method = "sequential",which.DB = "BOTH",prox.dist = 0.30)

ver1 = verif_OT(try1); ver1



### Example 2
#-----
# - Using the data simu_data
# - Studying the proximity between Y and Z using standard criterions and studying
#   associations by grouping levels of Z
# - When only Y is predicted in B
# - Tolerated distance between a subject and a profile: 0.30 * distance max
# - Using an outcome model (individual assignment with knn)
#-----

data(simu_data)
try2 = OT_outcome(simu_data, quanti = c(3,8), nominal = c(1,4:5,7), ordinal = c(2,6),
                  dist.choice = "G",percent.knn = 0.90, maxrelax = 0, prox.dist = 0.3,
                  convert.num = 8, convert.clss = 3,
                  indiv.method = "sequential",which.DB = "B")

ver2 = verif_OT(try2, group.clss = TRUE, ordinal = TRUE); ver2


### Example 3
#-----
# - Using the data simu_data
# - Studying the proximity between Y and Z using standard criterions and studying
#   associations by grouping levels of Z
# - Studying the stability of the conditional probabilities
# - When Y and Z are predicted in B and A respectively
# - Using an outcome model (individual assignment with knn)
#-----
```

```
ver3 = verif_OT(try2, group.clss = TRUE, ordinal = TRUE, stab.prob = TRUE, min.neigb = 5); ver3
```

Index