Package ‘DistatisR’

December 5, 2022

Type Package

Title DiSTATIS Three Way Metric Multidimensional Scaling

Version 1.1.1

Author Derek Beaton [aut, com, ctb], Herve Abdi [aut, cre]

Maintainer Herve Abdi <herve@utdallas.edu>

Description Implement DiSTATIS and CovSTATIS (three-way multidimensional scaling). DiSTATIS and CovSTATIS are used to analyze multiple distance/covariance matrices collected on the same set of observations. These methods are based on Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012) <doi:10.1002/wics.198>.

Imports prettyGraphs, car, readxl, dplyr, tidytext

Depends R (>= 3.5.0)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Repository CRAN

Date/Publication 2022-12-05 08:32:43 UTC

R topics documented:

DistatisR-package *implements three way metric multidimensional scaling: DISTATIS and COVSTATIS.*

Description

`DistatisR`: package implements three way metric multidimensional scaling: DISTATIS and COVSTATIS.

Details

Analyzes sets of distance (or covariance) matrices collected on the same set of observations and find common and specific metric spaces.

Package:	DistatisR
Type:	Package
Version:	1.1.0
Date:	2022-09-28
License:	GPL-2
Depends:	prettyGraphs (>= 2.0.0), car

The example shown here comes from Abdi *et al.* (2007), `distatis` paper on the sorting task.

Author(s)

Derek Beaton [aut, com, ctb], & Herve Abdi [aut, cre]

Maintainer: Herve Abdi <herve@utdallas.edu>

References

https://personal.utdallas.edu/~herve/

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition.* (San Diego, CA, USA). pp. 42-47.

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, **18**, 627–640.

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds): *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University & Ho Chi Minh City Publishing House. pp. 5–18.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classiffers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

Chollet, S., Valentin, D., & Abdi, H. (2014). The free sorting task. In. P.V. Tomasco & G. Ares (Eds), *Novel Techniques in Sensory Characterization and Consumer Profiling.* Boca Raton: Taylor and Francis.

Valentin, D., Chollet, S., Nestrud, M., & Abdi, H. (2018). Sorting and similarity methodologies. In. S. Kemp, S., J. Hort, & T. Hollowood (Eds.), *Descriptive Analysis in Sensory Evaluation*. London: Wiley-Blackwell.

See Also

distatis BootFactorScores BootFromCompromise DistanceFromSort distatis GraphDistatisAll GraphDistatisBoot GraphDistatisCompromise GraphDistatisPartial GraphDistatisRv mmds prettyGraphs

Examples

```
# Here we use the sorting task from Abdi et al.' (2007) paper.
# where 10 Assessors sorted 8 beers.

#-----------------------------------------------------------------------------
#  1. Get the data from the 2007 sorting example
#      this is the way they look from Table 1 of
#      Abdi et al. (2007).
#                       Assessors
#                  1 2 3 4 5 6 7 8 9 10
# Beer        Sex  f m f f m m m m f m
#            -----------------------------
#Affligen          1 4 3 4 1 1 2 2 1 3
#Budweiser         4 5 2 5 2 3 1 1 4 3
#Buckler_Blonde    3 1 2 3 2 4 3 1 1 2
#Killian           4 2 3 3 1 1 1 2 1 4
#St. Landelin      1 5 3 5 2 1 1 2 1 3
#Buckler_Highland  2 3 1 1 3 5 4 4 3 1
#Fruit Defendu     1 4 3 4 1 1 2 2 2 4
#EKU28             5 2 4 2 4 2 5 3 4 5


# 1.1. Create the
#     Name of the Beers
BeerName <- c('Affligen', 'Budweiser','Buckler Blonde',
              'Killian','St.Landelin','Buckler Highland',
              'Fruit Defendu','EKU28')
# 1.2. Create the name of the Assessors
#      (F are females, M are males)
Juges <- c('F1','M2', 'F3', 'F4', 'M5', 'M6', 'M7', 'M8', 'F9', 'M10')

# 1.3. Get the sorting data
SortData <- c(1, 4, 3, 4, 1, 1, 2, 2, 1, 3,
              4, 5, 2, 5, 2, 3, 1, 1, 4, 3,
              3, 1, 2, 3, 2, 4, 3, 1, 1, 2,
              4, 2, 3, 3, 1, 1, 1, 2, 1, 4,
              1, 5, 3, 5, 2, 1, 1, 2, 1, 3,
```

```
              2, 3, 1, 1, 3, 5, 4, 4, 3, 1,
              1, 4, 3, 4, 1, 1, 2, 2, 2, 4,
              5, 2, 4, 2, 4, 2, 5, 3, 4, 5)
# 1.4 Create a data frame
Sort <- matrix(SortData,ncol = 10, byrow= TRUE, dimnames = list(BeerName, Juges))
#     (alternatively we could have read a csv file)
# 1.5 Example of how to read a csv filw
# Sort <- read.table("BeeerSortingTask.csv", header=TRUE,
#   sep=",", na.strings="NA", dec=".", row.names=1, strip.white=TRUE)

#-------------------------------------------------------------------
# 2. Create the set of distance matrices
#  (one distance matrix per assessor)
#    (uses the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)
#-------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance as parameter
testDistatis <- distatis(DistanceCube)
# The factor scores for the beers are in
# testDistatis$res4Splus$F
# the factor scores for the assessors are in (RV matrice)
#  testDistatis$res4Cmat$G

#-------------------------------------------------------------------
# 4. Inferences on the beers obtained via bootstrap
#    here we use two different bootstraps:
#    1. Bootstrap on factors (very fast but could be too liberal
#         when the number of assessors is very large)
#    2. Complete bootstrap obtained by computing sets of compromises
#       and projecting them (could be significantly longer because a lot
#       of computations is required)
#
# 4.1 Get the bootstrap factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
#
# 4.2 Get the boostrap from full bootstrap (default niter = 1000)
 F_fullBoot <- BootFromCompromise(DistanceCube,niter=1000)


#-------------------------------------------------------------------
# 5. Create the Graphics
# 5.1 an Rv map
 rv.graph.out <- GraphDistatisRv(testDistatis$res4Cmat$G)
# 5.2 a compromise plot
 compromise.graph.out <- GraphDistatisCompromise(testDistatis$res4Splus$F)
# 5.3 a partial factor score plot
 partial.scores.graph.out <-
  GraphDistatisPartial(testDistatis$res4Splus$F,testDistatis$res4Splus$PartialF)
# 5.4 a bootstrap confidence interval plot
 #5.4.1 with ellipses
 boot.graph.out.ell <- GraphDistatisBoot(testDistatis$res4Splus$F,BootF)
 #or
 # boot.graph.out <- GraphDistatisBoot(testDistatis$res4Splus$F,F_fullBoot)
```

```
#5.4.2 with hulls
boot.graph.out.hull <- GraphDistatisBoot(testDistatis$res4Splus$F,BootF,ellipses=FALSE)
#or
# boot.graph.out <- GraphDistatisBoot(testDistatis$res4Splus$F,F_fullBoot,ellipses=FALSE)
#5.5 all the plots at once
all.plots.out <-
GraphDistatisAll(testDistatis$res4Splus$F,testDistatis$res4Splus$PartialF,
BootF,testDistatis$res4Cmat$G)
```

`amariSorting` *25 assessors twice sort and describe 12 amaris (i.e., bitter)*

Description

`sortingAmari`: 25 assessors twice sort and describe 12 amaris (i.e., bitter). The data consist in a list containing 3 objects: 1) `Sorting` a data frame with the 12 by 25*2 sorting data, 2) `cubeOfVocabulary`: an array of dimensions a 12 products * 41 words (vocabulary) * 50 assessors-repetition (i.e., 25 assessors *2 repetitions), and 3) a data frame: `information4Amaris` storing the description of the amaris.

Usage

```
data("amariSorting")
```

Format

a list containing 3 objects: 1) `Sorting` a data frame with the 12 by 25*2 sorting data, 2) `cubeOfVocabulary`: and array of dimensions a 12 products * 41 words (vocabulary) * 50 assessors-repetition (i.e., 25 assessors *2 repetitions), and 3) a data frame: `information4Amaris` storing the description of the amaris.

Details

The assessors are described by their 5 character names with the following code: the first letter m/f give the gender of the assessors (male versus female), the second and third characters go from 01 to 25 and uniquely identify the assessor, the fourth and fifth characters identify the repetition (r1 vs. r2).

Some words of the vocabulary have been shortened for convenience; here is the list of the short and long versions of the descriptors that have shortened: 'coffee_chocolate' <- 'coffee', fortified wine' <- 'wine', 'red fruit' <- 'red', 'soy sauce' <- 'soy', spirit (green)' <- 'spirit', 'vegetal_green' <- 'vegetal', and 'warm spice' <- 'spice'.

In the data sets, the amaris are identified with shortened names, the whole names can be found in the data frame `information4Amaris`.

Author(s)

Jacob Lahne, Hervé Abdi, & Hildegarde Heymann

Source

Lahne, J., Abdi, H., & Heymann, H. (2018). #' Rapid sensory profiles with DISTATIS and barycentric text projection: An example with amari, bitter herbal liqueurs. *Food Quality and Preference*, 66, 36-43. (available from https://personal.utdallas.edu/~herve/).

References

Lahne, J., Abdi, H., & Heymann, H. (2018). Rapid sensory profiles with DISTATIS and barycentric text projection: An example with amari, bitter herbal liqueurs. *Food Quality and Preference*, 66, 36-43.

`beersBlindSorting` *Novices and Experts sorted 3 types of beers from 3 different brewers without and without seeing the beers.*

Description

`beersBlindSorting`: several different groups of Novices and Beer-Experts sorted 9 beers with (Vision) or without (Blind) visual information. The 9 beers were 3 types of beers (blond, amber, and dark) obtained from 3 different brewers (Pelforth, Chti, & Leffe).

Usage

```
beersBlindSorting
```

Format

A list with 11 lists each storing a 9*9*N_k `cubeOfDistance` and one 9*9 distance table. Specifically:

$EV 9*9* 17 Experts, Vision

$EBr1 9*9* 13 Experts, Blind, rep 1

$EBr2 9*9* 13 Experts, Blind, rep 2

$EBr3 9*9* 13 Experts, Blind, rep 3

$EBr4 9*9* 13 Experts, Blind, rep 4

$NV 9*9* 21 Novices, Vision

$NBr1 9*9* 18 Novices, Blind, rep 1

$NBr2 9*9* 18 Novices, Blind, rep 2

$NBr3 9*9* 18 Novices, Blind, rep 3

$NBr4 9*9* 18 Novices, Blind, rep 4

$N2B 9*9* 37 Novices, Blind. (Group 2)

Details

Nine different commercial beers (denoted PelfBL, PelfA, PelfBR, ChtiBL, ChtiA, ChtiBR, LeffBL, LeffA, and LeffBR) were evaluated. These beers came from three different breweries: Pelforth (noted Pelf), Chti, (Chti), and Leffe (Leff), and each brewery provided three types of beer: blond (BL), amber (A), and dark (BR).

For each sorting task the data file gives the sorting distance matrix: A 9-beers by 9-beers distance matrix in which at the intersection of a row (representing one beer) and a column (representing another beer) a value of 0 indicates that these two beers were sorted in the same group and a value of 1 indicates that these two beers were sorted in different groups.

Multiple groups of novices and experts participated to the experiments. In the blind condition, the group of experts and one group of novices repeated four times the sorting taks (replication 1 to 4).

Author(s)

Maud Lelièvre, Sylvie Chollet , Hervé Abdi, and Dominique Valentin.

Source

A longer description of the data, story, first analysis, etc. can be found in: Lelièvre M., Chollet, S., Abdi, H., & Valentin, B. (2009). Beer trained and untrained assessors rely more on vision than on taste when they categorize beers. *Chemosensory Perception, 2*, 143-153. available from https://personal.utdallas.edu/~herve/abdi-lcav09-inpress.pdf

---

BeersFlashProfile *An example of an excel file storing the Flash Profile of 6 (fictitious) assessors evaluating 7 (imaginary) beers. This excel file can be read by* read.df.excel.

Description

BeersFlashProfile: An example of an excel file storing the Flash Profile of 6 (fictitious) assessors evaluating 7 (imaginary) beers. This excel file can be read by read.df.excel.

Details

In this example of Flash Profiling 6 (fictitious) assessors evaluated 7 (imaginery) beers. First, Each assessor chose a set of descriptors suited to describe these beers and then ranked (or rated in variations of the technique) the beers for each dimension. Note that the descriptors as well as the number of descriptors vary with the judges.

Note: The names of the variables starts with the Judges ID (J1- to J6-).

Note: the data are stored in the Excel Sheet called Rankings of the excel file BeersFlashProfile.xlsx.

FileName

BeersFlashProfile.xlsx

Author(s)

Hervé Abdi

Source

Abdi, H, & Valentin, D. (2007). https://personal.utdallas.edu/~herve/

References

Abdi, H., & Valentin, D. (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University & Ho Chi Minh City Publishing House. pp. 5-18.

See Also

BeersProjectiveMapping BeersProjectiveMapping_xlsx

Examples

```
# get the path and file name
path2file <- system.file("extdata",
                  "BeersFlashProfile.xlsx", package = 'DistatisR')
# read the data in excel file with read.df.excel
beerDataFlash  <- read.df.excel(path = path2file,
                           sheet = 'Rankings')$df.data
 # the Flash Profiling data are now in the data.frame beerDataFlash
```

BeersProjectiveMapping *7 (fictitious) assessors sort and verbally describe 7 Beers using* Projective Mapping.

Description

`BeersProjectiveMapping`: 7 (fictitious) assessors evaluated 7 Beers using *Projective Mapping* (with verbal description).

Usage

```
data("BeersProjectiveMapping")
```

Format

a list with 3 elements: 1) `ProjectiveMapping`: a matrix of dimensions 7 beers by 7*2 assessors-dimensions of the coordinates of the beers on the sheet of paper; 2) `Vocabulary`: a Beers (rows) by Assessors (columns) data.frame where each element of `Vocabulary` stores the words used by one assessor to describe a beer (words are separated with spaces); and 3) `CT.vocabulary` a matrix storing the I Products by N words (from the Vocabulary) contingency table, in `CT.vocabulary` the number at the intersection of a row (beer) and a column (word) is the number of assessors who used this word to describe that beer.

Details

First, Each assessor positioned the 7 beers on a sheet of paper according to the perceived similarity between the beers. For each assessor, the position of the beers was recorded from the X and Y coordinates. Second, the assessors were asked if they could describe each beer with some freely chosen descriptors. These descriptors are stored in a dataframe with 7 elements (one per assessor) where each element of the dataframe is a 7 component vector (one per beer) where each element stores the words used to describe a beer (words are separated with spaces).

Author(s)

Hervé Abdi

Source

Abdi, H, & Valentin, D. (2007). https://personal.utdallas.edu/~herve/

References

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University & Ho Chi Minh City Publishing House. pp. 5-18.

`BeersProjectiveMapping_xlsx` *An example of an excel file with Projective Mapping data and vocabulary. This excel file can be read by* `read.df.excel`.

Description

`BeersProjectiveMapping_xlsx`: an example of an excel file with Projective Mapping and vocabulary. This excel file can be read by `read.df.excel`. In this example 7 (fictitious) assessors evaluated 7 Beers.

Details

In this example of projective mapping with vocabulary, 7 (fictitious) assessors evaluated 7 Beers First, each assessor positionned the 7 beers on a sheet of paper according to the perceived similarity between the beers. For each assessor, the position of the beers is recorded from the X and Y coordinates. Second, the assessors are asked if they can describe the beers with some freely chosen descriptors. These descriptors are stored in a list with 7 elements (one per assessor) where each element of the list is a 7 component vector (one per beer) where each element stores the words used to describe a beer (words are separated with spaces). The coordinates of the beers on the sheet of paper are stored in the sheet `Maps`, the Vocabulary generated by the assessors is stored in the sheet `Vocabulary`

FileName

BeersProjectiveMapping_xlsx.xlsx

Author(s)

Hervé Abdi

Source

Abdi, H, & Valentin, D. (2007). https://personal.utdallas.edu/~herve/

References

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University & Ho Chi Minh City Publishing House. pp. 5-18.

See Also

BeersProjectiveMapping

Examples

```
# get the path and file name
path2file <- system.file("extdata",
       "BeersProjectiveMapping_xlsx.xlsx", package = 'DistatisR')
# read the data in excel file with read.df.excel
beerDataPM  <- read.df.excel(path = path2file,
                           sheet = 'Maps',
                       voc.sheet = 'Vocabulary')
```

BootFactorScores — *Computes observation factor scores Bootstrap replicates from partial factor scores.*

Description

`BootFactorScores`: Computes Bootstrap replicates of the factor scores of the observations from the partial factor scores. `BootFactorScores` is typically used to create confidence intervals and to compute Bootstrap ratios.

Usage

```
BootFactorScores(PartialFS, niter = 1000)
```

Arguments

`PartialFS`	The partial factor scores (e.g., as obtained from `distatis`).
`niter`	number of boostrap iterations (default = 1000)

Value

the output is a 3-way array of dimensions "number of observations by number of factors by number of replicates."

Technicalities

The input of `BootFactorScores` is obtained from the `distatis` function, the output is a 3-way array of dimensions number of observations by number of factors by number of replicates. The output is typically used to plot confidence intervals (i.e., ellipsoids or convex hulls) or to compute t-like statistic called *bootstrap ratios*. To compute a bootstrapped sample a set of K distance matrices is selected with replacement from the original set of K distance matrices. The partial factors scores of the selected distance matrices are then averaged to produce the bootstrapped estimate of the factor scores of the observations. This approach is also called *partial boostrap* by Lebart (2007, see also Chateau & Lebart 1996). It has the advantage of being very fast even for very large data sets. Recent work (Cadoret & Husson, 2012), however, suggests that partial boostrap could lead to optimistic bootstrap estimates when the number of distance matrices is large and that it is preferable to use instead a *total boostrap* approach (i.e., creating new compromises by resampling and then projecting them on the common solution see function `BootFromCompromise`, and Cadoret & Husson, 2012 see also Abdi *et al*., 2009 for an example).

Author(s)

Herve Abdi

References

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5-18.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classiffers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

These papers are available from https://personal.utdallas.edu/~herve/

Additional references:

Cadoret, M., Husson, F. (2012) Construction and evaluation of confidence ellipses applied at sensory data. *Food Quality and Preference*, **28**, 106–115.

Chateau, F., & Lebart, L. (1996). Assessing sample variability in the visualization techniques related to principal component analysis: Bootstrap and alternative simulation methods. In A. Prats (Ed.), *Proceedings of COMPSTAT 2006.* Heidelberg: Physica Verlag.

Lebart, L. (2007). Which bootstrap for principal axes methods? In *Selected contributions in data analysis and classification, COMPSTAT 2006*. Heidelberg: Springer Verlag.

See Also

BootFromCompromise GraphDistatisBoot

Examples

```
# 1. Load the Sort data set from the SortingBeer example
#    (available from the DistatisR package)
data(SortingBeer)
# Provide an 8 beers by 10 assessors set of
# results of a sorting task
#-----------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (ues the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)

#-----------------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance as parameter
testDistatis <- distatis(DistanceCube)
# The factor scores for the beers are in
# testDistatis$res4Splus$F
# the partial factor score for the beers for the assessors are in
#  testDistatis$res4Splus$PartialF
#
# 4. Get the bootstraped factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
```

BootFromCompromise — BootFromCompromise*: Computes Bootstrap replicates of the (observation) factor scores by creating bootstrapped compromises.*

Description

BootFromCompromise Computes observation Bootstrap replicates of the factor scores from bootstrapped compromises. BootFromCompromise is typically used to create confidence intervals and to compute Bootstrap ratios.

Usage

```
BootFromCompromise(
  LeCube2Distance,
  niter = 1000,
  Norm = "MFA",
  Distance = TRUE,
  RV = TRUE,
  nfact2keep = 3
)
```

Arguments

LeCube2Distance	The array of distance used to call distatis
niter	The number of bootstrap iterations (default = 1000)
Norm	should be the same as for the original call to distatis
Distance	should be the same as for the original call to distatis
RV	should be the same as for the original call to distatis
nfact2keep	number of factors to keep for the results

Value

the output is a 3-way array of dimensions "number of observations by number of factors by number of replicates."

Technicalities

The input of BootFromCompromise is the original cubeOfData used to compute the compromise by the function distatis. BootFromCompromise computes Bootstrap replicates of the observations by randomly selecting the observations with replacement. The output of BootFromCompromise is a 3-way array of dimensions "number of observations by number of factors by number of replicates." The output is typically used to plot confidence intervals (i.e., ellipsoids or convex hulls) or to compute t-like statistic called *bootstrap ratios*.

To compute a bootstrapped sample, a set of K distance matrices is selected with replacement from the original set of K distance matrices. A distatis compromise is then computed and projected on the factor space of the original solution to obtain the bootstrapped factor scores. This approach is also called *total boostrap* by Lebart (2007, see also Chateau and Lebart 1996, see also Abdi *et al*., 2009 for an example). Compared to the partial bootstrap (see help for `BootFactorScores`). This approach has the desadvantage of being slow especially for large data sets, but recent work (Cadoret & Husson, 2012) suggests that partial boostrap (i.e., computed from the partial factor scores) could lead to optimistic bootstrap estimates when the number of distance matrices is large and that it is preferable to use instead the *total boostrap*.

Author(s)

Herve Abdi

References

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5-18.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classiffers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

These papers are available from https://personal.utdallas.edu/~herve/

Additional references:

Cadoret, M., Husson, F. (2012) Construction and evaluation of confidence ellipses applied at sensory data. *Food Quality and Preference*, **28**, 106–115.

Chateau, F., & Lebart, L. (1996). Assessing sample variability in the visualization techniques related to principal component analysis: Bootstrap and alternative simulation methods. In A. Prats (Ed.),*Proceedings of COMPSTAT 2006.* Heidelberg: Physica Verlag.

Lebart, L. (2007). Which bootstrap for principal axes methods? In *Selected contributions in data analysis and classification, COMPSTAT 2006*. Heidelberg: Springer Verlag.

See Also

`BootFactorScores` `GraphDistatisBoot`.

Examples

```
# 1. Load the Sort data set from the SortingBeer example
#    (available from the DistatisR package)
data(SortingBeer)
# Provide the "8 beers by 10 assessors" results of a sorting task
#-----------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (uses the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)
```

```
#-----------------------------------------------------------------------------
# 3. Call the distatis function with the cube of distance as parameter
testDistatis <- distatis(DistanceCube)
# The factor scores for the beers are in
# testDistatis$res4Splus$F
# the partial factor scores for the beers for the assessors are in
#  testDistatis$res4Splus$PartialF
#
# 4. Get the bootstraped factor scores (with default 1000 iterations)
#    Here we use the "total bootstrap"
 F_fullBoot <- BootFromCompromise(DistanceCube,niter=1000)
```

`Chi2Dist` *Computes the χˆ2 distance between the rows of a rectangular matrix (with positive elements).*

Description

`Chi2Dist`: Computes the $I \times I$ matrix **D** which is the χ^2 distance matrix between the rows of an $I \times J$ rectangular matrix **X** (with non-negative elements), and provides the $I \times 1$ **m** vector of mass (where the mass of a row is the sum of the entries of this row divided by the grand total of the matrix). When the distance matrix and the associated vector of masses are used as input to the function `mmds` the results will give the factor scores of the correspondence analysis of the matrix **X**. The `function` is used by the function `Chi2DistanceFromSort` that computes the χ^2 distance for the results of a sorting task.

Usage

```
Chi2Dist(X)
```

Arguments

`X`	A rectangle matrix with non-negative elements

Value

Sends back a list:

`$Distance`	the squared χ^2 distance matrix computed the rows of matrix **X**.
`masses`	the vector of masses of the rows of of matrix **X**.

Author(s)

Herve Abdi

References

The procedure and references are detailled in (Paper available from https://personal.utdallas.edu/~herve/): Abdi, H. (2007). Distance. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 304–308.

And in:
Abdi, H., & Valentin, D. (2006). *Mathematiques pour les Sciences Cognitives (Mathematics for Cognitive Sciences).* Grenoble: PUG.

See also (for the example):

Abdi, H., & Williams, L.J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, **2**, 433–459.

See Also

Chi2DistanceFromSort distatis mmds

Examples

```
# Here is a data matrix from Abdi & Williams (2012)
# page 449, Table 15. Punctuation of 6 French authors
Punctuation = matrix(c(
  7836,   13112,   6026,
 53655,   102383, 42413,
115615,   184541, 59226,
161926,   340479, 62754,
 38177,   105101, 12670,
 46371,    58367, 14299),
     ncol =3,byrow = TRUE)
colnames(Punctuation) <-c('Period','Comma','Other')
rownames(Punctuation) <-c('Rousseau','Chateaubriand',
                   'Hugo','Zola','Proust','Giroudoux')
# 1. Get the Chi2 distance matrix
#     between the rows of Punctuation
Dres <- Chi2Dist(Punctuation)
# check that the mds of the Chi2 distance matrix
# with CA-masses gives the CA factor scores for I
# 2. Use function mmds from DistatisR
#
testmds <- mmds(Dres$Distance,masses=Dres$masses)
# Print the MDS factor scores from mmds
print('Factor Scores from mds')
print(testmds$FactorScores)
print('It matches CA on X (see Abdi & Williams, 2010. Table 16, p. 449)')
# Et voila!
```

Chi2DistanceFromSort   Chi2DistanceFromSort*: Creates a 3-dimensional $\chi\hat{}2$ distance array from the results of a sorting task.*

Description

Chi2DistanceFromSort: Takes the results from a (plain) sorting task where K assessors sort I observations into (mutually exclusive) groups (i.e., one object is in one an only one group). Chi2DistanceFromSort creates an $I \times I \times K$ array of distance in which each of the k "slices" stores the (sorting) distance matrix of the kth assessor. In one of these distance matrices, the distance between rows is the χ^2 distance between rows when the results of the task are coded as 0/1 group coding (i.e., the "complete disjunctive coding" as used in multiple correspondence analysis, see Abdi & Valentin, 2007, for more)

Usage

```
Chi2DistanceFromSort(X)
```

Arguments

X	gives the results of a sorting task (see example below) as a objects (row) by assessors (columns) matrix.

Details

The ouput ot the function Chi2DistanceFromSort is used as input for the function distatis.

The input should have assessors as columns and observations as rows (see example below)

Value

Chi2DistanceFromSort returns an $I \times I \times K$ array of K distances matrices (between the I observations)

Author(s)

Herve Abdi

References

See examples in

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, **18**, 627–640.

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products.* Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5–18.

Abdi, H., & Valentin, D. (2007). Multiple correspondence analysis. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics.* Thousand Oaks (CA): Sage. pp. 651-657.

These papers are available from https://personal.utdallas.edu/~herve/

See Also

distatis

Examples

```
#  1. Get the data from the 2007 sorting example
#      this is the eay they look from Table 1 of
#      Abdi et al. (2007).
#                        Assessors
#                  1 2 3 4 5 6 7 8 9 10
# Beer        Sex  f m f f m m m m f m
#             -----------------------------
#Affligen          1 4 3 4 1 1 2 2 1 3
#Budweiser         4 5 2 5 2 3 1 1 4 3
#Buckler_Blonde    3 1 2 3 2 4 3 1 1 2
#Killian           4 2 3 3 1 1 1 2 1 4
#St. Landelin      1 5 3 5 2 1 1 2 1 3
#Buckler_Highland  2 3 1 1 3 5 4 4 3 1
#Fruit Defendu     1 4 3 4 1 1 2 2 2 4
#EKU28             5 2 4 2 4 2 5 3 4 5

#
# 1.1. Create the
#     Name of the Beers
BeerName <- c('Affligen', 'Budweiser','Buckler Blonde',
              'Killian','St.Landelin','Buckler Highland',
              'Fruit Defendu','EKU28')
# 1.2. Create the name of the Assessors
#      (F are females, M are males)
Juges <- c('F1','M2', 'F3', 'F4', 'M5', 'M6', 'M7', 'M8', 'F9', 'M10')

# 1.3. Get the sorting data
SortData <- c(1, 4, 3, 4, 1, 1, 2, 2, 1, 3,
              4, 5, 2, 5, 2, 3, 1, 1, 4, 3,
              3, 1, 2, 3, 2, 4, 3, 1, 1, 2,
              4, 2, 3, 3, 1, 1, 1, 2, 1, 4,
              1, 5, 3, 5, 2, 1, 1, 2, 1, 3,
              2, 3, 1, 1, 3, 5, 4, 4, 3, 1,
              1, 4, 3, 4, 1, 1, 2, 2, 2, 4,
              5, 2, 4, 2, 4, 2, 5, 3, 4, 5)
# 1.4 Create a data frame
Sort <- matrix(SortData,ncol = 10, byrow= TRUE, dimnames = list(BeerName, Juges))
#
#-----------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (use the function DistanceFromSort)
```

```
DistanceCube <- Chi2DistanceFromSort(Sort)
#-----------------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance
#       obtained from DistanceFromSort as a parameter for the distatis function
testDistatis <- distatis(DistanceCube)
```

`computePartial4Groups` *Computes group alphas and group factor scores for K groups of observations in* `distatis`.

Description

`computePartial4Groups`: Computes group alphas and group factor scores for K groups of observations used to compute the compromise (i.e., matrix **S**) in a `distatis` analysis.

Usage

```
computePartial4Groups(resDistatis, DESIGN)
```

Arguments

`resDistatis`	The results of a Distatis analysis (as performed by the `DistatisR::distatis`) function.
`DESIGN`	A Design vector describing the groups of observations

Details

In DISTATIS, the compromise is computed as a weighted (with the alpha-coefficients) sum of the K pseudo-covariance matrices **S**k, `computePartial4Groups` sums all the alpha coefficients of a group to compute each group partial factor scores.

Value

A list with

- "GroupFS: " The K weighted mean coordinates
- "groupAlpha: " The K alpha weights

Author(s)

Hervé Abdi

See Also

`distatis`

ComputeSplus *ComputeSplus*

Description

Compute the compromise matrix for STATIS/DISTATIS

Usage

```
ComputeSplus(CubeCP, alpha)
```

Arguments

CubeCP	A 3D array of cross-product matrices.
alpha	The vector of weights

Value

The compromise matrix computed as the alpha-weighted sum of the cross-product matrices.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
            dim = c(3, 3, 2))
ComputeSplus(D3, alpha = c(1, 0.5))
```

CP2MFAnormedCP *CP2MFAnormedCP*

Description

MFA normalizes a cube of cross-product matrices

Usage

```
CP2MFAnormedCP(CP3)
```

Arguments

CP3	A 3D array of cross-product matrices

Value

The 3D array of the normalized cross-product matrices.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
           dim = c(3, 3, 2))
CP2MFAnormedCP(D3)
```

CP2NuclearNormedCP *CP2NuclearNormedCP*

Description

Nuclear Norm normalizes a cube of cross-product matrices

Usage

```
CP2NuclearNormedCP(CP3)
```

Arguments

CP3	A 3D array of cross-product matrices

Value

The 3D array of the normalized cross-product matrices.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
           dim = c(3, 3, 2))
CP2NuclearNormedCP(D3)
```

`CP2SUMPCAnormedCP` *CP2SUMPCAnormedCP*

Description

SUMPCA normalizes a cube of cross-product matrices.

Usage

```
CP2SUMPCAnormedCP(CP3)
```

Arguments

`CP3`	A 3D array of cross-product matrices

Value

The 3D array of the normalized cross-product matrices.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
            dim = c(3, 3, 2))
CP2SUMPCAnormedCP(D3)
```

`createCubeOfCovDis` *compute a cube of covariance and a cube of distance between the items (rows) of a brick of measurements (when all blocks have the same number of variables).*

Description

`createCubeOfCovDis` compute a cube of covariance and a cube of (squared) Euclidean distance between the items (rows) of a brick of measurements. The variables describing the items can scaled to norm 1 and centered. The whole matrix can be scaled by its first eigenvalue (a la DISTATIS). All "slices" of the brick should have the same number of variables. For different number of variables per block, see `list2CubeOfCov`.

Usage

```
createCubeOfCovDis(brickOfData, scale = TRUE, center = TRUE, ev.scale = TRUE)
```

Arguments

brickOfData	a I items by J quantitative variables by K assessors.
scale	(Default: TRUE), when TRUE scale to norm 1 each column for each slice.
center	(Default: TRUE), when TRUE centers each column.
ev.scale	(Default: TRUE), when TRUE normalizes each slice (i.e., each I items by J matrix) so that its first eigenvalue is equal to 1.

Details

The input of `createCubeOfCovDis` is a I items by J quantitative variables by K assessors (as obtained, e.g., from a projective mapping task).

By default `createCubeOfCovDis` centers and normalizes each column for each slice of the brick and then normalize each covariance matrix such that the first eigenvalue of each covariance matrix is equal to 1.

A `distatis` analysis of the Distance matrices with the option `Distance = TRUE` will give the same results as the `distatis` analysis of the Covariance matrices with the option `Distance = FALSE`.

Value

a list with 1) `cubeOfCovariance` a cube of K I by I covariance matrices; and 2) codecubeOfDistance a cube of K I by I (squared) Euclidean distance matrices.

Author(s)

Herve Abdi

See Also

list2CubeOfCov

Examples

```
# use the data from the BeersProjectiveMapping dataset
data("BeersProjectiveMapping")
# Create the I*J_k*K brick of data
zeBrickOfData <- projMap2Cube(
                    BeersProjectiveMapping$ProjectiveMapping,
                    shape = 'flat',  nVars = 2)
# Create the cubes of Covariance and Distance
cubes <- createCubeOfCovDis(zeBrickOfData$cubeOfData)
```

`DblCenterDist` *Double Center a distance matrix*

Description

Double Center a distance matrix

Usage

```
DblCenterDist(Y)
```

Arguments

`Y`	a "distance" matrix,

Value

a cross-product matrix (if the distance is Euclidean)

Examples

```
Y <- toeplitz(c(0, 3, 3))
DblCenterDist(Y)
```

`Dist2CP` *Dist2CP*

Description

Transforms a cube of distance matrices into a cube of cross-product matrices.

Usage

```
Dist2CP(D3)
```

Arguments

`D3`	the cube of distance matrices.

Value

the cube of cross-product matrices.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
            dim = c(3, 3, 2))
Dist2CP(D3)
```

`DistAlgo` *Four computer algorithms evaluate the similarity of six faces for distatis analysis*

Description

Provide the data.frame `DistAlgo` Data set to be used to illustrated the use of the package `DistatisR`. Four algorithms evaluate the similarity (i.e., distance) between six faces (3 females and 3 males). Each algorithm provides a 6×6 distance matrix evaluating the distance between each pair of faces.

Format

an $6 \times 6 \times 4$ array. Each 6×6 matrix is a distance matrix

Author(s)

Herve Abdi

Source

Abdi et al. (2005). https://personal.utdallas.edu/~herve/

References

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition*. (San Diego, CA, USA). pp. 42–47.

DistanceFromRank — DistanceFromRank*: Creates a 3-dimensional distance array from the results of a ranking task.*

Description

DistanceFromRank: Takes the results from a (plain) ranking task where K assessors rank (with possible ties) I observations on one dimension and transform it into a brick of data to be used by distatis.

Usage

```
DistanceFromRank(X)
```

Arguments

X	gives the results of a ranking task (see example below) as an objects (rows) by assessors (columns) matrix.

Details

DistanceFromRank creates an $I \times I \times K$ array of distance in which each of the K "slices" stores the (squared Euclidean ranking) distance matrix of the kth assessor. In one of these distance matrices, the distance between two objects is computed from he Pythagorean theorem. The ouput ot the function DistanceFromRank is used as input for the function distatis.

The input should have assessors as columns and observations as rows (see example below).

Value

DistanceFromRank returns an $I \times I \times K$ array of distance.

Author(s)

Herve Abdi

See Also

distatis DistanceFromSort

Examples

```
# Use the data set WinesRankingRawData stored in an excel file.
path2file <- system.file("extdata",
           "WinesRankingRawData.xlsx", package = 'DistatisR')
ranking6Wines <- read.df.excel(path = path2file, sheet = 'Ranking')
aCubeOfDistance <- DistanceFromRank(ranking6Wines$df.data)
```

`DistanceFromSort` *Creates a 3-dimensional distance array from the results of a sorting task.*

Description

`DistanceFromSort`: Takes the results from a (plain) sorting task where K assessors sort I observations into (mutually exclusive) groups (i.e., one object is in one and only one group). `DistanceFromSort` creates an $I \times I \times K$ array of distance in which each of the k "slices" stores the (sorting) distance matrix of the kth assessor. In one of these distance matrices, a value of 0 at the intersection of a row and a column means that the object represented by the row and the object represented by the column were sorted together (i.e., they are a distance of 0), and a value of 1 means these two objects were put into different groups.

The ouput ot the function `DistanceFromSort` is used as input for the function `distatis`.

The input should have assessors as columns and observations as rows (see example below)

Usage

```
DistanceFromSort(X)
```

Arguments

`X`	gives the results of a sorting task (see example below) as a objects (row) by assessors (columns) matrix.

Value

`DistanceFromSort` returns an $I \times I \times K$ array of distance.

Author(s)

Herve Abdi

References

See examples in

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, **18**, 627–640.

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products.* Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5–18.

These papers are available from https://personal.utdallas.edu/~herve/

See Also

`distatis`

Examples

```
#  1. Get the data from the 2007 sorting example
#      this is the eay they look from Table 1 of
#      Abdi et al. (2007).
#                       Assessors
#                  1 2 3 4 5 6 7 8 9 10
# Beer        Sex  f m f f m m m m f m
#            -----------------------------
#Affligen          1 4 3 4 1 1 2 2 1 3
#Budweiser         4 5 2 5 2 3 1 1 4 3
#Buckler_Blonde    3 1 2 3 2 4 3 1 1 2
#Killian           4 2 3 3 1 1 1 2 1 4
#St. Landelin      1 5 3 5 2 1 1 2 1 3
#Buckler_Highland  2 3 1 1 3 5 4 4 3 1
#Fruit Defendu     1 4 3 4 1 1 2 2 2 4
#EKU28             5 2 4 2 4 2 5 3 4 5

#
# 1.1. Create the
#     Name of the Beers
BeerName <- c('Affligen', 'Budweiser','Buckler Blonde',
              'Killian','St.Landelin','Buckler Highland',
              'Fruit Defendu','EKU28')
# 1.2. Create the name of the Assessors
#      (F are females, M are males)
Juges <- c('F1','M2', 'F3', 'F4', 'M5', 'M6', 'M7', 'M8', 'F9', 'M10')

# 1.3. Get the sorting data
SortData <- c(1, 4, 3, 4, 1, 1, 2, 2, 1, 3,
              4, 5, 2, 5, 2, 3, 1, 1, 4, 3,
              3, 1, 2, 3, 2, 4, 3, 1, 1, 2,
              4, 2, 3, 3, 1, 1, 1, 2, 1, 4,
              1, 5, 3, 5, 2, 1, 1, 2, 1, 3,
              2, 3, 1, 1, 3, 5, 4, 4, 3, 1,
              1, 4, 3, 4, 1, 1, 2, 2, 2, 4,
              5, 2, 4, 2, 4, 2, 5, 3, 4, 5)
# 1.4 Create a data frame
Sort <- matrix(SortData,ncol = 10, byrow= TRUE, dimnames = list(BeerName, Juges))
#
#---------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (use the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)
#---------------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance
#       obtained from DistanceFromSort as a parameter for the distatis function
testDistatis <- distatis(DistanceCube)
```

distatis — *3-Way MDS based on the "STATIS" optimization procedure.*

Description

distatis: Implements the DISTATIS method which is a 3-way generalization of metric multidimensional scaling (*a.k.a.* classical MDS or principal coordinate analysis).

Usage

```
distatis(
  LeCube2Distance,
  Norm = "MFA",
  Distance = TRUE,
  double_centering = TRUE,
  RV = TRUE,
  nfact2keep = 3,
  compact = FALSE
)
```

Arguments

- `LeCube2Distance` — an "observations $\times$ observations $\times$ distance matrices" array of dimensions $I \times I \times K$. Each of the K "slices" is a $I \times I$ square distance (or covariance) matrix describing the I observations.
- `Norm` — Type of normalization used for each cross-product matrix derived from the distance (or covariance) matrices. Current options are `NONE` (do nothing), `SUMPCA` (normalize by the total inertia) or `MFA` (`default`) that normalizes each matrix so that its first eigenvalue is equal to one or `NUCLEAR` (i.e., the of the squarae root of the eigenvalues).
- `Distance` — if `TRUE` (`default`) the matrices are distance matrices, `FALSE` the matrices are treated as positive semi-definite matrices (e.g., scalar products, covariance, or correlation matrices).
- `double_centering` — if `TRUE` (`default`) the matrices are double-centered (should always be used for distances). if `FALSE` the matrices will *not* be double centered (note that these matrices should be semi positive definite matrices such as, for example, covariance matrices).
- `RV` — if `TRUE` (`default`) we use the R_V coefficient to compute the α, if `FALSE` we use the matrix scalar product.
- `nfact2keep` — (default: `3`) Number of factors to keep for the computation of the factor scores of the observations.
- `compact` — if `FALSE` (default), `distatis` provides detailed output, if `TRUE`, `distatis` sends back only the α weights (this option is used to make the bootstrap routine `BootFromCompromise` more computationally efficient).

Details

distatis takes as input a set of K distance matrices (or positive semi-definite matrices such as scalar products, covariance, or correlation matrices) describing a set of I observations. From this set of matrices distatis computes: (1) a set of factor scores that describes the similarity structure of the K distance matrices (e.g., what distance matrices describe the observations in the same way, what distance matrices differ from each other) (2) a set of factor scores (called the *compromise* factor scores) that best describes the similarity structure of the I observations and (3) I sets of partial factor scores that show how each individual distance matrix "sees" the compromise space.

distatis computes the compromise as an optimum linear combination of the cross-product matrices associated to each distance (or positive positive semi-definite) matrix.

distatis can also be applied to a set of scalar products, covariance, or correlation matrices.

DISTATIS is part of the STATIS family. It is often used to analyze the results of sorting tasks.

Value

distatis sends back the results *via* two lists: res.Cmat and res.Splus. Note that items with a * are the only ones sent back when using the compact = TRUE option.

res.Cmat Results for the between distance matrices analysis.

- res.Cmat$C The $I \times I$ **C** matrix of scalar products (or R_V between distance matrices).
- res.Cmat$vectors The eigenvectors of the **C** matrix
- res.Cmat$alpha * The α weights
- res.Cmat$value The eigenvalues of the **C** matrix
- res.Cmat$G The factor scores for the **C** matrix
- res.Cmat$ctr The contributions for res.Cmat$G,
- res.Cmat$cos2 The squared cosines for res.Cmat$G
- res.Cmat$d2 The squared Euclidean distance for res.Cmat$G.

res.Splus Results for the between observation analysis.

- res.Splus$SCP an $I \times I \times K$ array. Contains the (normalized if needed) cross product matrices corresponding to the distance matrices.
- res.Splus$Splus * The compromise (optimal linear combination of the SCP's').
- res.Splus$eigValues * The eigenvalues of the compromise).
- res.Splus$eigVectors * The eigenvectors of the compromise).
- res.Splus$tau * The percentage of explained inertia of the eigenValues).
- res.Splus$ProjectionMatrix The projection matrix used to compute factor scores and partial factor scores.
- res.Splus$F The factor scores for the observations.
- res.Splus$ctr The contributions for res.Cmat$F.
- res.Splus$cos2 The squared cosines for res.Cmat$F.
- res.Splust$d2 The squared Euclidean distance for res.Cmat$F.
- res.Splus$PartialF an $I \times$ nf2keep $\times K$ array. Contains the partial factors for the distance matrices.

Author(s)

Hervé Abdi #@seealso GraphDistatisAll GraphDistatisBoot #GraphDistatisCompromise # GraphDistatisPartial #GraphDistatisRv DistanceFromSort #BootFactorScores BootFromCompromise #as help,

References

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition*. (San Diego, CA, USA). pp. 42–47.

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, **18**, 627–640.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classifiers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

The R_V coefficient is described in

Abdi, H. (2007). RV coefficient and congruence coefficient. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 849–853.

Abdi, H. (2010). Congruence: Congruence coefficient, RV coefficient, and Mantel Coefficient. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design.* Thousand Oaks (CA): Sage. pp. 222–229.

(These papers are available from https://personal.utdallas.edu/~herve/)

Examples

```
# 1. Load the DistAlgo data set
# (available from the DistatisR package).
data(DistAlgo)
# DistAlgo is a 6*6*4 Array (face*face*Algorithm)
#-----------------------------------------------------------------
# 2. Call the DISTATIS routine with the array
#  of distance (DistAlgo) as parameter
DistatisAlgo <- distatis(DistAlgo)
```

GetCmat *GetCmat*

Description

Computes the RV coefficient matrix

Usage

```
GetCmat(CubeCP, RV = TRUE)
```

Arguments

CubeCP	A 3D array of cross-product matrices
RV	Boolean, if TRUE, GetCmat computes the matrix of the RV coefficients between all the slices of the 3D array, otherwise, GetCmat computes a scalar product.

Value

A matrix of either RV coefficients or scalar products.

Examples

```
D3 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
               0, 3, 3, 3, 0, 3, 3, 3, 0),
             dim = c(3, 3, 2))
GetCmat(D3)
```

GetRectCmat *GetRectCmat*

Description

Computes the rectangular RV coefficient matrix between two arrays of conformable matrices

Usage

```
GetRectCmat(CubeCP1, CubeCP2, RV = TRUE)
```

Arguments

CubeCP1	A 3D array of cross-product matrices
CubeCP2	A 3D array of cross-product matrices
RV	Boolean, if TRUE, GetCmat computes the matrix of the RV coefficients between all the slices of the 3D array, otherwise, GetCmat computes a scalar product.

Value

A rectangular matrix of either RV coefficients or scalar products.

Examples

```
D3.1 <- array(c(0, 1, 2, 1, 0, 1, 2, 1, 0,
              0, 3, 3, 3, 0, 3, 3, 3, 0),
            dim = c(3, 3, 2))
D3.2 <- array(c(1, 0, 0, 0, 1, 0, 0, 0, 1,
                 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 0, 0, 0, 0, 0, 0, 0, 0, 0),
            dim = c(3, 3, 3))
GetRectCmat(D3.1, D3.2)
```

`GraphDistatisAll` *This function combines the functionality of* `GraphDistatisCompromise`, `GraphDistatisPartial`, `GraphDistatisBoot`, *and* `GraphDistatisRv`.

Description

This function produces 4 plots: (1) a compromise plot, (2) a partial factor scores plot, (3) a bootstrap confidence intervals plot, and (4) an Rv map.

Usage

```
GraphDistatisAll(
  FS,
  PartialFS,
  FBoot,
  RvFS,
  axis1 = 1,
  axis2 = 2,
  constraints = NULL,
  item.colors = NULL,
  participant.colors = NULL,
  ZeTitleBase = NULL,
  nude = FALSE,
  Ctr = NULL,
  RvCtr = NULL,
  color.by.observations = TRUE,
  lines = TRUE,
  lwd = 3.5,
  ellipses = TRUE,
  fill = TRUE,
  fill.alpha = 0.27,
  percentage = 0.95
)
```

Arguments

FS
: The factor scores of the observations ($res4Splus$Ffrom distatis)

PartialFS
: The partial factor scores of the observations ($res4Splus$PartialF from distatis)

FBoot
: is the bootstrapped factor scores array (FBoot obtained from BootFactorScores or BootFromCompromise)

RvFS
: The factor scores of the distance matrices ($res4Cmat$G from distatis)

axis1
: The dimension for the horizontal axis of the plots.

axis2
: The dimension for the vertical axis of the plots.

constraints
: constraints for the axes

item.colors
: A $I \times 1$ matrix (with I = # observations) of color names for the observations. If NULL (default), prettyGraphs chooses.

participant.colors
: A $I \times 1$ matrix (with I = # participants) of color names for the observations. If NULL (default), prettyGraphs chooses.

ZeTitleBase
: General title for the plots.

nude
: When nude is TRUE the labels for the observations are not plotted (useful when editing the graphs for publication).

Ctr
: Contributions of each observation. If NULL (default), these are computed from FS

RvCtr
: Contributions of each participant. If NULL (default), these are computed from RvFS

color.by.observations
: if TRUE (default), the partial factor scores are colored by item.colors. When FALSE, participant.colors are used.

lines
: If TRUE (default) then lines are drawn between the partial factor score of an observation and the compromise factor score of the observation.

lwd
: Thickness of the line plotting the ellipse or hull.

ellipses
: a boolean. When TRUE will plot ellipses (from car package). When FALSE (default) will plot peeled hulls (from prettyGraphs package).

fill
: when TRUE, fill in the ellipse with color. Relevant for ellipses only.

fill.alpha
: transparency index when filling in the ellipses. Relevant to ellipses only.

percentage
: A value to determine the percent coverage of the bootstrap partial factor scores to provide ellipse or hull confidence intervals.

Value

constraints
: A set of plot constraints that are returned.

item.colors
: A set of colors for the observations are returned.

participant.colors
: A set of colors for the participants are returned.

Author(s)

Derek Beaton and Herve Abdi

See Also

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

Examples

```
# 1. Load the Sort data set from the SortingBeer example  (available from the DistatisR package)
data(SortingBeer)
# Provide an 8 beers by 10 assessors results of a sorting task
#-----------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (ues the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)

#-----------------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance as parameter
testDistatis <- distatis(DistanceCube)
# The factor scores for the beers are in
# testDistatis$res4Splus$F
# the partial factor score for the beers for the assessors are in
#  testDistatis$res4Splus$PartialF
#
# 4. Get the bootstraped factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
#-----------------------------------------------------------------------------
# 5. Create the Graphics with GraphDistatisAll
#
GraphDistatisAll(testDistatis$res4Splus$F,testDistatis$res4Splus$PartialF,
BootF,testDistatis$res4Cmat$G)
```

GraphDistatisBoot   GraphDistatisBoot *Plot maps of the factor scores of the observations and their bootstrapped confidence intervals (as confidence ellipsoids or peeled hulls) for a* DISTATIS *analysis.*

Description

GraphDistatisBoot plots maps of the factor scores of the observations from a distatis analysis. GraphDistatisBoot gives a map of the factors scores of the observations plus the bootstrapped confidence intervals drawn as "Confidence Ellipsoids" at the percentage level (see parameter percentage).

Usage

```
GraphDistatisBoot(
  FS,
  FBoot,
```

```
  axis1 = 1,
  axis2 = 2,
  item.colors = NULL,
  ZeTitle = "Distatis-Bootstrap",
  constraints = NULL,
  nude = FALSE,
  Ctr = NULL,
  lwd = 3.5,
  ellipses = TRUE,
  fill = TRUE,
  fill.alpha = 0.27,
  percentage = 0.95
)
```

Arguments

FS	The factor scores of the observations ($res4Splus$F from distatis)
FBoot	is the bootstrapped factor scores array (FBoot obtained from BootFactorScores or BootFromCompromise)
axis1	The dimension for the horizontal axis of the plots. (default = 1).
axis2	The dimension for the vertical axis of the plots (default = 2).
item.colors	When present, should be a column matrix (dimensions of observations and 1). Gives the color-names to be used to color the plots. Can be obtained as the output of this or the other graph routine. If NULL, prettyGraphs chooses.
ZeTitle	General title for the plots (default is 'Distatis-Bootstrap').
constraints	constraints for the axes
nude	When TRUE do not plot the names of the observations (default is FALSE).
Ctr	Contributions of each observation. If NULL (default), these are computed from FS.
lwd	Thickness of the line plotting the ellipse or hull (default = 3.5).
ellipses	a boolean. When TRUE (default) will plot ellipses (from the car package). When FALSE will plot peeled hulls (from prettyGraphs package).
fill	when TRUE, fill in the ellipse with color. Relevant for ellipses only.
fill.alpha	transparency index (a number between 0 and 1) when filling in the ellipses. Relevant for ellipses only (default = .27).
percentage	A value to determine the percent coverage of the bootstrap partial factor scores to provide ellipse or hull confidence intervals (default = .95).

Details

The ellipses are plotted using the function dataEllipse() from the package car. The peeled hulls are plotted using the function peeledHulls() from the package prettyGraphs.

Note that, in the current version, the graphs are plotted as R-plots and are *not* passed back by the function. So the graphs need to be saved "by hand" from the R graphic windows. We plan to improve this in a future version. See also package PTCA4CATA for ggplot2 based graphs.

Value

`constraints` A set of plot constraints that are returned.

`item.colors` A set of colors for the observations are returned.

Author(s)

Derek Beaton and Herve Abdi

References

The plots are similar to the graphs described in:

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classiffers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5–18.

These papers are available from https://personal.utdallas.edu/~herve/

See Also

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

Examples

```
# 1. Load the Sort data set from the SortingBeer example
# (available from the DistatisR package)
data(SortingBeer)
# Provide an 8 beers by 10 assessors results of a sorting task
#-----------------------------------------------------------------------------
# 2. Create the set of distance matrices (one distance matrix per assessor)
#    (ues the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)

#-----------------------------------------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance as parameter
testDistatis <- distatis(DistanceCube)
# The factor scores for the beers are in
# testDistatis$res4Splus$F
# the partial factor score for the beers for the assessors are in
#  testDistatis$res4Splus$PartialF
#
# 4. Get the bootstraped factor scores (with default 1000 iterations)
```

```
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
#-----------------------------------------------------------------------------
# 5. Create the Graphics with GraphDistatisBoot
#
GraphDistatisBoot(testDistatis$res4Splus$F,BootF)
```

GraphDistatisCompromise   *Plot maps of the factor scores of the observations for a* DISTATIS *analysis*

Description

Plot maps of the factor scores of the observations for a DISTATIS analysis. `GraphDistatis` gives a map of the factor scores for the observations. The labels of the observations are plotted by defaults but can be omitted (see the `nude=TRUE` option).

Usage

```
GraphDistatisCompromise(
  FS,
  axis1 = 1,
  axis2 = 2,
  constraints = NULL,
  item.colors = NULL,
  ZeTitle = "Distatis-Compromise",
  nude = FALSE,
  Ctr = NULL
)
```

Arguments

`FS`	The factor scores of the observations (`$res4Splus$F`from `distatis`).
`axis1`	The dimension for the horizontal axis of the plots.
`axis2`	The dimension for the vertical axis of the plots.
`constraints`	constraints for the axes
`item.colors`	A $I \times 1$ matrix (with I = # observations) of color names for the observations. If `NULL` (default), `prettyGraphs` chooses.
`ZeTitle`	General title for the plots.
`nude`	(default `FALSE`) When `nude` is `TRUE` the labels for the observations are not plotted (useful when editing the graphs for publication).
`Ctr`	Contributions of each observation. If NULL (default), these are computed from FS

Details

Note that, in the current version, the graphs are plotted as R-plots and are *not* passed back by the routine. So the graphs need to be saved "by hand" from the R graphic windows. We plan to improve this in a future version.

Value

`constraints`	A set of plot constraints that are returned.
`item.colors`	A set of colors for the observations are returned.

Author(s)

Derek Beaton and Herve Abdi

References

The plots are similar to the graphs from

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition*. (San Diego, CA, USA). pp. 42-47.

Paper available from: https://personal.utdallas.edu/~herve/

See Also

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

Examples

```
# 1. Load the DistAlgo data set (available from the DistatisR package)
data(DistAlgo)
# DistAlgo is a 6*6*4 Array (face*face*Algorithm)
#-----------------------------------------------------------------------------
# 2. Call the DISTATIS routine with the array of distance (DistAlgo) as parameter
DistatisAlgo <- distatis(DistAlgo)
# 3. Plot the compromise map with the labels for the first 2 dimensions
# DistatisAlgo$res4Splus$F are the factors scores for the 6 observations (i.e., faces)
# DistatisAlgo$res4Splus$PartialF are the partial factors scores
##(i.e., one set of factor scores per algorithm)
 GraphDistatisCompromise(DistatisAlgo$res4Splus$F)
```

`GraphDistatisPartial` *Plot maps of the factor scores and partial factor scores of the observations for a* DISTATIS *analysis.*

Description

`GraphDistatisPartial` plots maps of the factor scores of the observations from a `distatis` analysis. `GraphDistatisPartial` gives a map of the factors scores of the observations plus partial factor scores, as "seen" by each of the matrices.

Usage

```
GraphDistatisPartial(
  FS,
  PartialFS,
  axis1 = 1,
  axis2 = 2,
  constraints = NULL,
  item.colors = NULL,
  participant.colors = NULL,
  ZeTitle = "Distatis-Partial",
  Ctr = NULL,
  color.by.observations = TRUE,
  nude = FALSE,
  lines = TRUE
)
```

Arguments

`FS` The factor scores of the observations (`$res4Splus$F` from the output of `distatis`).

`PartialFS` The partial factor scores of the observations (`$res4Splus$PartialF` from `distatis`)

`axis1` The dimension for the horizontal axis of the plots.

`axis2` The dimension for the vertical axis of the plots.

`constraints` constraints for the axes

`item.colors` A $I \times 1$ matrix (with I = # observations) of color names for the observations. If `NULL` (default), `prettyGraphs` chooses.

`participant.colors` A $I \times 1$ matrix (with I = # participants) of color names for the observations. If `NULL` (default), `prettyGraphs` chooses (with function `prettyGraphs::`).

`ZeTitle` General title for the plots.

`Ctr` Contributions of each observation. If NULL (default), these are computed from FS

`color.by.observations` if `TRUE` (default), the partial factor scores are colored by `item.colors`. When `FALSE`, `participant.colors` are used.

nude — When nude is TRUE the labels for the observations are not plotted (useful when editing the graphs for publication).

lines — If TRUE (default) then lines are drawn between the partial factor score of an observation and the compromise factor score of the observation.

Details

Note that, in the current version, the graphs are plotted as R-plots and are *not* passed back by the routine. So the graphs need to be saved "by hand" from the R graphic windows. We plan to improve this in a future version.

Value

constraints — A set of plot constraints that are returned.

item.colors — A set of colors for the observations are returned.

participant.colors — A set of colors for the participants are returned.

Author(s)

Derek Beaton and Herve Abdi

References

The plots are similar to the graphs from

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition*. (San Diego, CA, USA). pp. 42-47.

Paper available from https://personal.utdallas.edu/~herve/

See Also

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

Examples

```
# 1. Load the DistAlgo data set (available from the DistatisR package)
data(DistAlgo)
# DistAlgo is a 6*6*4 Array (face*face*Algorithm)
#-----------------------------------------------------------------------------
# 2. Call the DISTATIS routine with the array of distance (DistAlgo) as parameter
DistatisAlgo <- distatis(DistAlgo)
# 3. Plot the compromise map with the labels for the first 2 dimensions
# DistatisAlgo$res4Splus$F are the factors scores for the 6 observations (i.e., faces)
# DistatisAlgo$res4Splus$PartialF are the partial factors scores
##(i.e., one set of factor scores per algorithm)
 GraphDistatisPartial(DistatisAlgo$res4Splus$F,DistatisAlgo$res4Splus$PartialF)
```

GraphDistatisRv — *Plot maps of the factor scores (from the Rv matrix) of the distance matrices for a DISTATIS analysis*

Description

Plot maps of the factor scores of the observations for a DISTATIS analysis. The factor scores are obtained from the eigen-decomposition of the between distance matrices cosine matrix (often a matrix of Rv coefficients). Note that the factor scores for the first dimension are always positive. There are used to derive the α weights for DISTATIS.

Usage

```
GraphDistatisRv(
  RvFS,
  axis1 = 1,
  axis2 = 2,
  ZeTitle = "Distatis-Rv Map",
  participant.colors = NULL,
  nude = FALSE,
  RvCtr = NULL
)
```

Arguments

`RvFS` The factor scores of the distance matrices (`$res4Cmat$G` from `distatis`).

`axis1` The dimension for the horizontal axis of the plots.

`axis2` The dimension for the vertical axis of the plots.

`ZeTitle` General title for the plots.

`participant.colors` A $I \times 1$ matrix (with I = # participants) of color names for the observations. If NULL (default), `prettyGraphs` chooses.

`nude` When `nude` is `TRUE` the labels for the observations are not plotted (useful when editing the graphs for publication).

`RvCtr` Contributions of each participant. If codeNULL (default), these are computed from `RvFS`.

Details

Note that, in the current version, the graphs are plotted as R-plots and are *not* passed back by the routine. So the graphs need to be saved "by hand" from the R graphic windows. We plan to improve this in a future version.

Value

`constraints` A set of plot constraints that are returned.

`participant.colors` A set of colors for the participants are returned.

Author(s)

Derek Beaton and Herve Abdi

References

The plots are similar to the graphs described in:

Abdi, H., Valentin, D., O'Toole, A.J., & Edelman, B. (2005). DISTATIS: The analysis of multiple distance matrices. *Proceedings of the IEEE Computer Society: International Conference on Computer Vision and Pattern Recognition*. (San Diego, CA, USA). pp. 42-47.

Abdi, H., Williams, L.J., Valentin, D., & Bennani-Dosse, M. (2012). STATIS and DISTATIS: Optimum multi-table principal component analysis and three way metric multidimensional scaling. *Wiley Interdisciplinary Reviews: Computational Statistics*, **4**, 124–167.

Abdi, H., Dunlop, J.P., & Williams, L.J. (2009). How to compute reliability estimates and display confidence and tolerance intervals for pattern classiffers using the Bootstrap and 3-way multidimensional scaling (DISTATIS). *NeuroImage*, **45**, 89–95.

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University-Ho chi Minh City Publishing House. pp. 5–18.

The R_V coefficient is described in

Abdi, H. (2007). RV coefficient and congruence coefficient. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 849–853.

Abdi, H. (2010). Congruence: Congruence coefficient, RV coefficient, and Mantel Coefficient. In N.J. Salkind, D.M., Dougherty, & B. Frey (Eds.): *Encyclopedia of Research Design.* Thousand Oaks (CA): Sage. pp. 222–229.

These papers are available from https://personal.utdallas.edu/~herve/

See Also

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

Examples

```
# 1. Load the DistAlgo data set (available from the DistatisR package)
data(DistAlgo)
# DistAlgo is a 6*6*4 Array (faces*faces*Algorithms)
#-----------------------------------------------------------------------------
# 2. Call the DISTATIS routine with the array of distance (DistAlgo) as parameter
DistatisAlgo <- distatis(DistAlgo)
```

```
# 3. Plot the compromise map with the labels for the first 2 dimensions
# DistatisAlgo$res4Cmat$G are the factors scores
#  for the 4 distance matrices (i.e., algorithms)
 GraphDistatisRv(DistatisAlgo$res4Cmat$G,ZeTitle='Rv Mat')
# Et voila!
```

`list2CubeOfCovDis` *compute a cube of covariance and a cube of distance between the items (rows) of a matrix of measurements comprising K different blocks of possibly different number of variables.*

Description

`list2CubeOfCovDis` compute a cube of covariance and a cube of (squared) Euclidean distance between the items (rows) a matrix of measurements comprising K different blocks of possibly different number of variables. The variables describing the items can scaled to norm 1 and centered. The whole matrix for a block can be scaled by its first eigenvalue (a la DISTATIS). Blocks can have different number of variables and when all blocks have same number of variables `list2CubeOfCovDis` is a more efficient alternative

Usage

```
list2CubeOfCovDis(Data, Judges, scale = TRUE, center = TRUE, ev.scale = TRUE)
```

Arguments

`Data`	a matrix of dimensions I items by J quantitative variables (structured in K blocks of J_k variables each). No Default.
`Judges`	a J components character vector identifiying the variables corresponding to each block of variables. No Default.
`scale`	(Default: `TRUE`), when `TRUE` scale to norm 1 each column for each slice.
`center`	(Default: `TRUE`), when `TRUE` centers each column.
`ev.scale`	(Default: `TRUE`), when `TRUE` normalizes each slice (i.e., each I items by J matrix) so that its first eigenvalue is equal to 1.

Details

The input of `list2CubeOfCovDis` is a I items by J quantitative variables that are organized in K blocks (i.e., submatrices) each comprising J_k variables (with sum $J_k = J$).

By default `list2CubeOfCovDis` centers and normalizes each column for each block and then normalize each covariance matrix such that the first eigenvalue of each covariance matrix (for a given block) is equal to 1.

A `distatis` analysis of the Distance matrices with the option `Distance = TRUE` will give the same results as the `distatis` analysis of the Covariance matrices with the option `Distance = FALSE`.

Value

a list with 1) `cubeOfCovariance` a cube of K I by I covariance matrices; and 2) codecubeOfDistance a cube of K I by I (squared) Euclidean distance matrices.

Author(s)

Herve Abdi

See Also

list2CubeOfCov

Examples

```
path2file <- system.file("extdata",
                        "BeersFlashProfile.xlsx",
                        package = 'DistatisR')
# read the data in the excel file with read.df.excel
beerDataFlash  <- read.df.excel(path = path2file,
                               sheet = 'Rankings')$df.data
# Extract the namers of the judges (first 2 characters)
JudgesVars <- colnames(beerDataFlash)
zeJudges <- substr(JudgesVars,1,2)
 # call list2CubeOfCovDis
test.list2 <- list2CubeOfCovDis(Data = beerDataFlash ,
                               Judges =  zeJudges)
```

`MFAnormCP` *MFAnormCP*

Description

Normalizeq a positive semi-definite matrix matrix product such that its first eigenvalue is equal to one

Usage

```
MFAnormCP(Y)
```

Arguments

`Y`	The matrix to normalize

Value

The normalized matrix

Examples

```
A <- toeplitz(c(1, 0.6))
MFAnormCP(A)
```

`mmds` *Metric (classical) Multidimensional Scaling (a.k.a Principal Coordinate Analysis) of a (squared Euclidean) Distance Matrix.*

Description

`mmds`: Perform a Metric Multidimensional Scaling (MMDS) of an (squared Euclidean) distance matrix measured between a set of objects (with or without masses).

Usage

```
mmds(DistanceMatrix, masses = NULL)
```

Arguments

`DistanceMatrix`	A squared (assumed to be Euclidean) distance matrix
`masses`	A vector of masses (i.e., a set of non-negative numbers with a sum of 1) of same dimensionality as the number of rows of `DistanceMatrix`.

Details

`mmds` gives factor scores that make it possible to draw a map of the objects such that the distances between objects on the map best approximate the original distances between objects.

Value

Sends back a list

`LeF`	factor scores for the objects.
`eigenvalues`	the eigenvalues for the factor scores (i.e., a variance).
`tau`	the percentage of explained variance by each dimension.
`Contributions`	give the proportion of explained variance by an object for a dimension.

Method

MMDS transform the squared Euclidean distance matrix into a (double centered) covariance-like matrix which is then analyzed via its eigen-decomposition. The factor scores of each dimension are scaled such that their variance (i.e., the sum of their weighted squared factor scores) is equal to the eigen-value of the corresponding dimension. Note that if the `masses` vector is absent, equal masses (i.e., 1 divided by number of objects) are used.

Technicalities

the distance matrix to be analyzed is supposed to be a *squared* Euclidean distance matrix. Note also that a non Euclidean distance matrix will have negative eigenvalues that will be ignored by mmds which, therefore, gives the best Euclidean approximation to this non-Euclidean distance matrix (note that, non-metric MDS maybe a better method in these cases).

Author(s)

Herve Abdi

References

The procedure and references are detailed in: Abdi, H. (2007). Metric multidimensional scaling. In N.J. Salkind (Ed.): *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage. pp. 598–605.

(Paper available from https://personal.utdallas.edu/~herve/).

See Also

GraphDistatisCompromise distatis

Examples

```
# An example of MDS from Abdi (2007)
# Discriminability of Brain States
# Table 1.
# 1. Get the distance matrix
D <- matrix(c(
0.00, 3.47, 1.79, 3.00, 2.67, 2.58, 2.22, 3.08,
3.47, 0.00, 3.39, 2.18, 2.86, 2.69, 2.89, 2.62,
1.79, 3.39, 0.00, 2.18, 2.34, 2.09, 2.31, 2.88,
3.00, 2.18, 2.18, 0.00, 1.73, 1.55, 1.23, 2.07,
2.67, 2.86, 2.34, 1.73, 0.00, 1.44, 1.29, 2.38,
2.58, 2.69, 2.09, 1.55, 1.44, 0.00, 1.19, 2.15,
2.22, 2.89, 2.31, 1.23, 1.29, 1.19, 0.00, 2.07,
3.08, 2.62, 2.88, 2.07, 2.38, 2.15, 2.07, 0.00),
ncol = 8, byrow=TRUE)
rownames(D) <- c('Face','House','Cat','Chair','Shoe','Scissors','Bottle','Scramble')
colnames(D) <- rownames(D)
# 2. Call mmds
BrainRes <- mmds(D)
# Note that compared to Abdi (2007)
# the factor scores of mmds are equal to F / sqrt(nrow(D))
# the eigenvalues of mmds are equal to \Lambda *{1/nrow(D)}
# (ie., the normalization differs but the results are proportional)
# 3. Now a pretty plot with the prettyPlot function from prettyGraphs
prettyGraphs::prettyPlot(BrainRes$FactorScore,
           display_names = TRUE,
           display_points = TRUE,
           contributionCircles = TRUE,
```

```
            contributions = BrainRes$Contributions)
# 4. et Voila!
```

---

multiculturalSortingSpices

62 assessors from 5 countries sort 16 spice samples

Description

multiculturalSortingSpices: 62 participants from 5 different countries (USA, France, India, Spain, and Vietnam) for 16 different spices (including 6 mixtures of spices).

The data consist in a list containing 7 objects (for all sorting data the number at the intersection of a row and a colum indicates the number of the pile in which the spice was sorted);): : A data frame containing the 1) sortAll: A data frame containing the results of the sorting task for all 62 participants, 2) sortAmerican: A data frame containing the results of the sorting task for the 9 American participants, 3) sortFrench: A data frame containing the results of the sorting task for the 21 French participants, 4) sortIndian: A data frame containing the results of the sorting task for the 15 Indian participants, 5) sortSpanish: A data frame containing the results of the sorting task for the 11 Spanish participants, 6) sortVietnamese: A data frame containing the results of the sorting task for the 6 Vietnamese participants, and 7) spicesDescription A data frame containing the description of the Spices.

Usage

```
data("multiculturalSortingSpices")
```

Format

A list containing 7 objects (for all sorting data, the number at the intersection of a row and a colum indicates the number of the pile in which the spice was sorted): : A data frame containing the 1) sortAll: A data frame containing the results of the sorting task for all 62 participants, 2) sortAmerican: A data frame containing the results of the sorting task for the 9 American participants, 3) sortFrench: A data frame containing the results of the sorting task for the 21 French participants, 4) sortIndian: A data frame containing the results of the sorting task for the 15 Indian participants, 5) sortSpanish: A data frame containing the results of the sorting task for the 11 Spanish participants, 6) sortVietnamese: A data frame containing the results of the sorting task for the 6 Vietnamese participants, and 7) spicesDescription A data frame containing the description of the Spices.

Details

In the data frames, the spice blends are identifed wiht acronyms that are expended in the data frame spicesDescription.

Author(s)

Chollet, S., Valentin, D., & Abdi, H.

References

Part of these data (i.e., the French sample) is described and analyzed in Chollet, S., Valentin, D., & Abdi, H. (2014). Free sorting task. In P.V. Tomasco & G. Ares (Eds), *Novel Techniques in Sensory Characterization and Consumer Profiling*. Boca Raton: Taylor and Francis. pp 207-227.

`NuclearNormedCP` *NuclearNormedCP*

Description

Normalizes a positive semi-definite matrix by diving it by its nuclear norm (i.e., the sum of the square root of its eigen-values).

Usage

```
NuclearNormedCP(Y)
```

Arguments

`Y`	The matrix to normalize

Value

The normalized matrix

Examples

```
A <- toeplitz(c(1, 0.6))
NuclearNormedCP(A)
```

`OrangeJuiceSortingRawData` `OrangeJuiceSortingRawData`*: an example of an excel file with Sorting data and vocabulary. This excel file can be read by* `read.df.excel`*.*

Description

`OrangeJuiceSorting`: an example of an excel file with sorting data (10 Orange Juices sorted by 44 Assessors) and an an associated vocabulary contingency table (10 Orange Juices by 23 descriptors). Can be read by `read.df.excel`.

Details

In this example of a "sorting task" with vocabulary, 44 assessors sorted 10 orange juices and freely described each group of juices with a few words. The data from the sorting task are in the sheet "Sorting" and the contingency table (10 Orange Juices by 23 descriptors) is in the sheet "Vocabulary". To fetch this dataset use `system.file()` (see example below).

FileName

OrangeJuiceSortingRawData.xlsx

Author(s)

Herve Abdi

Examples

```
path2file <- system.file("extdata",
           "OrangeJuiceSortingRawData.xlsx", package = 'DistatisR')
OrangeDataSort <- read.df.excel(path = path2file,
                            sheet = 'Sorting',
                        voc.sheet = 'Vocabulary')
```

`projectVoc` *Compute barycentric projections for count-like description of the items of a* `distatis`*-type of analysis.*

Description

`projectVoc` Compute barycentric projection for count-like description of the items of a `distatis`-type of analysis. The data need to be non-negative and typically represent the vocabulary (i.e., words) used to describe the items in a sorting/ranking/projective-mapping task.

Usage

```
projectVoc(CT.voc, Fi, namesOfFactors = NULL)
```

Arguments

`CT.voc`	a matrix or data.frame storing a I items by J descriptors contingency table where the i, j-th cell gives the number of times the j-th descriptor (in the column) was used to describe the i-th item (in the row). `CT.voc` needs to contain only non-negative numbers.
`Fi`	a matrix or data.frame storing the I items by L factor scores obtained from the compromise of a `distatis` analysis or equivalent.
`namesOfFactors`	(Default: NULL), if `NULL`, `projectVoc` uses the names of the columns of `Fi` for the names of the projected factors; if `namesOfFactors` is one word then this word is used to name the factors of the projections; if `namesOfFactors` is a character vector, it is used to name the factors of the projection.

Details

two types of projection are computed: 1) a plain barycentric (words are positioned at the barycenter– a.k.a. center of mass–of the items it describes) and 2) a correspondence analysis barycentric where the variance of the projected words is equal to the variance of the items (as for correspondence analysis when using the "symmetric" representation).

Value

a list with 1) `Fvoca.bary`: the barycentric projections of the words, and 2) `Fvoca.normed`: the CA normalized (i.e., variance of projections equals eigenvalue) barycentric projections of the words.

Author(s)

Herve Abdi

Source

Abdi, H, & Valentin, D. (2007). Papers available from https://personal.utdallas.edu/~herve/

References

Abdi, H., & Valentin, D., (2007). Some new and easy ways to describe, compare, and evaluate products and assessors. In D., Valentin, D.Z. Nguyen, L. Pelletier (Eds) *New trends in sensory evaluation of food and non-food products*. Ho Chi Minh (Vietnam): Vietnam National University & Ho Chi Minh City Publishing House. pp. 5-18.

and

Lahne, J., Abdi, H., & Heymann, H. (2018). Rapid sensory profiles with DISTATIS and barycentric text projection: An example with amari, bitter herbal liqueurs. *Food Quality and Preference, 66*, 36-43.

Examples

```
# use the data from the BeersProjectiveMapping dataset
data("BeersProjectiveMapping")
# Create the I*J*K brick of data
zeBrickOfData <- projMap2Cube(
                  BeersProjectiveMapping$ProjectiveMapping,
                  shape = 'flat',  nVars = 2)
# create the cube of covariance matrices between beers
cubeOfCov <- createCubeOfCovDis(zeBrickOfData$cubeOfData)
# Call distatis
testDistatis <- distatis(cubeOfCov$cubeOfCovariance, Distance = FALSE)
# Project the vocabulary onto the factor space
F4Voc <- projectVoc(BeersProjectiveMapping$CT.vocabulary,
                    testDistatis$res4Splus$F)
```

`projMap2Cube` *\ reshape a data matrix from projective mapping into a brick of data for a* `distatis` *analysis.*

Description

`projMap2Cube` reshapes a data matrix from projective mapping into a brick of data for a `distatis` analysis. With I products, J variables, and K blocks (assessors), the original data can be 1) "flat" (e.g., I rows as products, columns as K blocks of J Variables) or 2) "long" "flat" (e.g., K blocks of I rows as products by assessors, columns as J Variables).

Usage

```
projMap2Cube(Data, shape = "flat", nVars = 2, nBlocks = NULL)
```

Arguments

- `Data` a data matrix that can be I rows by $J * K$ columns (when `"flat"`) or $I * K$ rows by J columns when `"long"`.
- `shape` (Default: `flat` when `"flat"` the data matrix has dimensions I rows by $J * K$ columns; when `"long"` the data matrix has dimensions $I * K$ rows by J columns.
- `nVars` Number of variables (default = 2), relevant only when `shape = "flat"`.
- `nBlocks` (Default = `NULL`) number of Blocks (i.e., K) of I products. Relevant only when `shape = "long"`.

Details

the output `projMap2Cube` (i.e., the brick of data) is used as input to the function `cubeOfCov` that will create the cubeOfDistance (or covariance) that will be used as input of `distatis`. `projMap2Cube` guesses the names of the products and variables from the rownames and columns of the data, but this guess needs to be verified.

Value

An I by J by K array (i.e., a brick) to be used to create a cube of distance or covariance.

Author(s)

Herve Abdi

Examples

```
# Use the data from the BeersProjectiveMapping dataset
data("BeersProjectiveMapping")
# Create the I*J_k*K brick of data
dataBrick <- projMap2Cube(BeersProjectiveMapping$ProjectiveMapping,
                        shape = 'flat',  nVars = 2)
```

read.df.excel — *read.df.excel reads distatis formated ranking or sorting data from an excel file.*

Description

read.df.excel reads distatis formated ranking or sorting data from an excel file.

Usage

```
read.df.excel(path, sheet, col_names = TRUE, voc.sheet = NULL)
```

Arguments

path	the name of the .xlsx file (including the path to the directory if needed, and the .xlsx extension). No default.
sheet	the name of the sheet where the (e.g., Sorting or Ranking) data are stored. No default.
col_names	(default TRUE) parameter col.names from readxl::read_excel: "TRUE to use the first row as column names, FALSE to get default names, or a character vector giving a name for each column."
voc.sheet	If not NULL (default) gives the name of the sheet where an optional contingency table (products by names) could be stored. Needs to have the same row names as the sorting/ranking data frame (df.data) to be useful (but he program does not check). @return a list with one data frame ‘$df.data‘ (contains the data) when ‘voc.sheet = NULL‘ or if not: two data frames ‘$df.data‘ (contains the data) and ‘$df.voc‘ (contains the vocabulary).

Details

@details The data are read from an excel file in which the rows are the Products to evaluate and the columns are the Assessors (e.g., Judges, Participants, Subjects, Evaluators). Depending upon the type of data, the numbers represent a partition, a rank, or a score. These data are used as input of DistanceFromSort or DistanceFromRank. A contingency table for the vocabulary can also be read in a different sheet. read.df.excel is a (small) shell on top of readxl::read_excel, note however that whereas readxl::read_excel returns a tibble, read.df.excel returns a list with one or two (depending upon the options) *dataframe(s)*.

Author(s)

Herve Abdi

`rv` *Function to compute the RV coefficient between to conformable matrices*

Description

Function to compute the RV coefficient between to conformable matrices

Usage

```
rv(A, B)
```

Arguments

`A`	a matrix,
`B`	a second matrix.

Value

the RV coefficient between A and B.

Examples

```
A <- toeplitz(2:1)
B <- diag(2)
rv(A, B)
```

`scale1` *A variation over the base `R` scale function that avoids the "divide by 0 = NA" problem.*

Description

`scale1`: A variation over the base `R` `scale` function. The function `scale1`: centers (if needed) and scales a vector to norm 1; if the vector contains values all equal to a constant, `scale1` sets all values to 0 (in lieu of NA as `scale` does). Usefull when pre-processing tables for PCA-like type of analyses.

Usage

```
scale1(x, scale = TRUE, center = TRUE)
```

Arguments

`x`	a vector to be scaled
`scale`	(default = `TRUE`), when `TRUE` scale the vector to norm 1, otherwise do nothing.
`center`	(default = `TRUE`), when `TRUE` center the vectors (i.e., substract the mean from all numbers), otherwise do nothing.

Value

a centered (if required) and norm-1 (if required) normalized vector.

Author(s)

Hervé Abdi

Examples

```
toto   <- runif(10)     # 10 random numbers between 0 and 1
tutu   <- scale1(toto)  # toto centered and normalized
toto0  <- rep(1,10)     # 10 numbers all equal to 1
tutu0  <- scale1(toto0) # scaled to 0 # Compare with
tutuNA <- scale(toto0)  # all numbers set to NA
```

SortingBeer *Ten Assessors sorted eight beers for* `distatis` *analysis*

Description

Provide the data.frame `Sort`: Data set to be used to illustrated the use of the package `DistatisR`. Ten assessors sorted eight beers. These data come from the Abdi et al.' (2007) paper in *Food Quality and Preference*. Each column represents the results of the sorting task for one assessor. Beers with the same number were sorted together.

Format

a data frame file containing 10 columns, 8 rows plus the names of the rows and the columns.

Source

Abdi et al. (2007), see `https://personal.utdallas.edu/~herve/`.

References

Abdi, H., Valentin, D., Chollet, S., & Chrea, C. (2007). Analyzing assessors and products in sorting tasks: DISTATIS, theory and applications. *Food Quality and Preference*, **18**, 627–640.

SortingSpice *21 French assessors sorted 16 blends of Spice for* distatis *analysis*

Description

Provide the data.frame SortSpice: Data set to illustrate the use of the package DistatisR. Assessors are sorting spices. Each column represents the results of the sorting task for one assessor. Spices with the same number were sorted together.

Format

a data frame file containing 21 columns, 16 rows plus the names of the rows and the columns.

Source

Chollet et al. (2014). Paper available from https://personal.utdallas.edu/~herve/

References

Chollet, S., Valentin, D., & Abdi, H. (2014). The free sorting task. In. P.V. Tomasco & G. Ares (Eds), *Novel Techniques in Sensory Characterization and Consumer Profiling*. Boca Raton: Taylor and Francis.

sortingWines *Novices and wines experts sort red, rosé, and white wines*

Description

sortingWines: 26 novices participants and 19 wine experts sort (by smell alone, without visual information) 18 wines (6 red, 6 rosé, and 6 whites) into three categories. The experts also performed a free sorting task on the wines (i.e. with as many groups as the wished).

The data consist in a list containing 4 objects: 1) freeSortExperts: a data frame with the 18 wines by 19 experts free sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); 2) ternarySortExperts: a data frame with the 18 wines by 19 experts ternary (i.e., in three piles) sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); 3) $ternarySortNovices: a data frame with the 18 wines by 19 novices ternary (i.e., in three piles) sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); and 4) vinesDescription a data frame storing the description of the 18 wines.

Usage

```
data("sortingWines")
```

Format

a list containing 4 objects: 1) `freeSortExperts`: a data frame with the 18 wines by 19 experts free sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); 2) `ternarySortExperts`: a data frame with the 18 wines by 19 experts ternary (i.e., in three piles) sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); 3) `$ternarySortNovices`: a data frame with the 18 wines by 19 nivices ternary (i.e., in three piles) sorting data (the number at the intersection of a row and a colum indicates the number of the pile in which the wine was sorted); and 4) `vinesDescription` a data frame storing the description of the 18 wines.

Details

The wines were served in dark glasses and the sorting task was performed with red light (this way all wines look black). In the experiment, the wines were labeled with three-digit codes, for more details see Ballester *et al.* (2009). Only the experts performed the free sorting task.

In the data sets, the wines are identified with shortened names, the whole names can be found in the data frame. All the wines were from the 2005 vintage (see Ballester *et al.*, 2009 for details)

Compared to the original data, some missing data were added to the set after imputation of the missing data (a total of 4 entries). The current data include only 19 experts out the original 27 experts. `vinesDescription`.

Author(s)

Ballester, J., Abdi, H., Langlois, J., Peyron, D., & Valentin, D.

References

For more details see:

Ballester, J., Abdi, H., Langlois, J., Peyron, D., & Valentin, D. (2009). The odor of colors: Can wine experts and novices distinguish the odors of white, red, and rosé wines? *Chemosensory Perception, 2*, 203-213.

`SUMPCAnormCP` *SUMPCAnormCP*

Description

Normalizes a positive semi-definite matrix (i.e., total intertia=1)

Usage

```
SUMPCAnormCP(Y)
```

Arguments

`Y`	Matrix to normalize

Value

Normalized matrix

Examples

```
A <- toeplitz(c(1, 0.6))
SUMPCAnormCP(A)
```

`supplementalProjection4distatis`
Supplementary element(s) projection in DISTATIS

Description

`supplementalProjection4distatis`: Computes for `distatis` the projection as supplementary element(s) (a.k.a. "out of sample") of a set of squared matrices. The matrices to be projected need to be of the same type (e.g., distance, correlation) as the matrices used inn the original call to `distatis`.

Usage

```
supplementalProjection4distatis(res.distatis, elsupp)
```

Arguments

`res.distatis`	the results of the function `distatis`
`elsupp`	the supplementary elements (i.e., a 3D array).

Value

the coordinates of the supplementary and active elements in the RV space. *** HA comment: Maybe we also want to send the square cosines to give the quality of representation.

Author(s)

Vincent Guillemot

`vocabulary2CT` *Transforms a data.frame of products by vocabulary of assessors into a products by words (from vocabulary) contingency table.*

Description

`vocabulary2CT` Transforms a data.frame of products by vocabulary of assessors into 1) a cube of 0/1 contingency tables (one per assessor); and 2) a products by words (from vocabulary) contingency table. In this contingency table, the number at the intersection of a row (product) and a column (word) is the number of assessors who used this word to describe that product.

@details the cube of 0/1 contingency tables (i.e., `cubeOfVocabulary` can also be analyzed with the package `PTCA4CATA` as a pseudo *Check All That Apply* (CATA) data set.

Usage

```
vocabulary2CT(df.voc)
```

Arguments

`df.voc`	a data frame with the vocabulary. In this data.frame each element stores the words used by one assessor to describe a product (words are separated with spaces);

Value

a list with 1) `cubeOfVocabulary`: a 0/1 array of dimension products by words (from the vocabulary) by assessors where each "products by vocabulary" slice gives the vocabulary chosen by the assessor to describe the products; and 2) `CT.vocabulary` a matrix storing the products by words contingency table.

Author(s)

Herve Abdi

See Also

`unnest_tokens` `count` `BeersProjectiveMapping`

Examples

```
# Get the BeersProjectiveMapping example
data("BeersProjectiveMapping")
aContingenyTable <- vocabulary2CT(BeersProjectiveMapping$Vocabulary)
```

`WinesRankingRawData` `WinesRankingRawData`*: an example of an excel file with (simulated) ranking data. Can be read with the function* `read.df.excel()`.

Description

`WinesRankingRawData`: an example of an excel file with (simulated) ranking data (6 wines ranked by 80 Assessors). Can be read by `read.df.excel`.

Details

In this example of a "ranking task," 80 (simulated or fictitious) assessors ranked 6 red wines from Burgundy (France). The assessor first chooses the most relevant dimension for these wines and then positions the wines on a scale from 1 to 9 for this dimension. The names of the assessors is composed of 4 characters of the general composition w/ma/f01 : 80. The assessors were 40 men and 40 women (first character w/n) and 40 American or 40 French (second character a/f). The red wines that were tasted are Irancy, Saint-Brie, Beaune, Nuits (Cote de Nuits), Beaujolais, and Beaujolais-Nouveau. Irancy & Saint-Brie are near the cities of Auxerre and Chablis, Beaune & Cote de Nuits are from central Burgundy, and Beaujolais and Beaujolais Nouveau are from the south of Burgundy; Beaujolais Nouveau is a young wine (a primeur) released in November of its year, after only a few weeks of fermentation.

Availability

To fetch this dataset use `system.file()` (see example below).

FileName

WinesRankingRawData.xlsx

Author(s)

Herve Abdi

Examples

```
path2file <- system.file("extdata",
          "WinesRankingRawData.xlsx", package = 'DistatisR')
ranking6Wines <- read.df.excel(path = path2file, sheet = 'Ranking')
```

Index