Package ‘DistatisR’

February 19, 2015

Type   Package
Title  DiSTATIS Three Way Metric Multidimensional Scaling
Version 1.0
Date 2013-07-10
Author Derek Beaton [aut, com, ctb], Cherise Chin Fatt [ctb], Herve Abdi [aut, cre]
Maintainer Derek Beaton <exposition.software@gmail.com>
Description
Implement DiSTATIS and CovSTATIS (three-way multidimensional scaling). For the analysis of multiple distance/covariance matrices collected on the same set of observations
License GPL-2
Depends prettyGraphs (>= 2.0.0), car
URL www.utdallas.edu/~herve
NeedsCompilation no
Repository CRAN
Date/Publication 2013-07-11 07:24:31

R topics documented:

DistatisR-package .................................................. 2
BootFactorScores .................................................. 5
BootFromCompromise ............................................. 7
Chi2Dist ............................................................. 9
Chi2DistanceFromSort .......................................... 11
DistAlgo ........................................................... 13
DistanceFromSort ................................................ 13
distatis ........................................................... 15
GraphDistatisAll .................................................. 18
GraphDistatisBoot ............................................... 20
GraphDistatisCompromise .................................. 22
GraphDistatisPartial ......................................... 24
GraphDistatisRv ................................................ 25
mmds .............................................................. 27
Description

DistatisR package implements three way multidimensional scaling: DISTATIS and COVSTATIS. Analyses sets of distance (or covariance) matrices collected on the same set of observations.

Details

Package: DistatisR
Type: Package
Version: 1.0
Date: 2013-07-03
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], Cherise Chin Fatt [ctb], & Herve Abdi [aut, cre]
Maintainer: Derek Beaton <exposition.software@gmail.com>

References

Note: these papers are available from www.utdallas.edu/~herve


**See Also**

distatisBootFactorScores BootFromCompromise DistanceFromSort distatis GraphDistatisAll GraphDistatisBoot GraphDistatisCompromise GraphDistatisPartial GraphDistatisRv mmds prettyGraphs

**Examples**

```r
# 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 <- c('F1', 'F2', 'F3', 'F4', 'M5', 'M6', 'M7', 'M8', 'F9', 'M10')

# Name of the Beers
BeerName <- c('Affligen', 'Budweiser', 'Buckler Blonde',
    'Killian', 'St.Landelin', 'Buckler Highland',
    'Fruit Defendu', 'EKU28')

# Create the
# Name of the Assessors
# (F are females, M are males)
Juges <- Assessors
```


# 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, 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, Judges))

# (alternatively we could have read a csv file)
# 1.5 Example of how to read a csv file
# Sort <- read.table("BeerSortingTask.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 matrix)
# 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
# and projecting them (could be significantly longer because a lot
# of computations is required)
# 2. Complete bootstrap obtained by computing sets of compromises
# 4.1 Get the bootstrap factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
# 4.2 Get the bootstrap 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 <-
**BootFactorScores**

BootFactorScores Compute observation Bootstrap replicates of the factor scores from partial factor scores

Description

BootFactorScores Compute Bootstrap replicates of the factor scores of the observations from partial factor scores. The input 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*.

Usage

BootFactorScores(PartialFS, niter = 1000)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PartialFS</td>
<td>The partial factor scores (e.g., obtained from distatis)</td>
</tr>
<tr>
<td>niter</td>
<td>number of bootstrap iterations (default = 1000)</td>
</tr>
</tbody>
</table>

Details

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 bootstrap* 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 bootstrap could lead to optimistic bootstrap estimates when the number of distance matrices is large and that it is preferable to use instead a *total bootstrap* 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).
Value

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

Author(s)

Herve Abdi

References


These papers are available from www.utdallas.edu/~herve

Additional references:


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)
# (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 partial factor score for the beers for the assessors are in
# testDistatis$res4Splus$PartialF
#
# 4. Get the bootstrapped factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)

BootFromCompromise  Compute observation Bootstrap replicates of the factor scores from bootstrapped compromises

Description

BootFactorScores computes Bootstrap replicates of the factor scores of the observations from bootstrapped compromises. The input is obtained from the same input as 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.

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

Details

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 bootstrap 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) it has the disadvantage of being slow especially for large data sets but recent work (Cadoret & Husson, 2012)
suggests that partial bootstrap (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 bootstrap.

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

Author(s)
Herve Abdi

References
These papers are available from www.utdallas.edu/~herve

Additional references:

See Also
- `BootFromCompromise`
- `BootFactorsScores`
- `GraphDistatisBoot`

Examples

```r
# 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$res4$plus$F
# the partial factor scores for the beers for the assessors are in
# testDistatis$res4$plus$PartialF
#
# 4. Get the bootstrapped factor scores (with default 1000 iterations)
# Here we use the “total bootstrap”
F_fullBoot <- BootFromCompromise(DistanceCube,niter=1000)

---

Chi2Dist

\chi^2 distance between the rows of a rectangular matrix.

Description

Compute the $I \times I$ matrix $D$ which is the $\chi^2$ 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 $\chi^2$ 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

- $D$ the squared $\chi^2$ distance matrix computed the rows of matrix $X$.
- $m$ the vector of masses of the rows of of matrix $X$.

Author(s)

Herve Abdi
References

The procedure and references are detailed in (Paper available from www.utdallas.edu/~herve):

And in:

See also (for the example):

See Also

Chi2DistanceFromSort distatis mmds

Examples

```r
# 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','Giraudoux')

# 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 mmds')
print(testmds$FactorScores)
print('It matches CA on X (see Abdi & Williams, 2010. Table 16, p. 449)')
# Et voila!
```
Chi2DistanceFromSort

Creates a 3-dimensional $\chi^2$ distance array from the results of a sorting task.

**Description**

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). 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 $k$th assessor. In one of these distance matrices, the distance between rows is the $\chi^2$ 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).

The output of the function DistanceFromSort is used as input for the function distatis.

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

**Value**

DistanceFromSort returns a $I \times I \times K$ array of distances.

**Author(s)**

Herve Abdi

**References**

See examples in


These papers are available from www.utdallas.edu/~herve
See Also

distatis

Examples

# 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  m  f  m  f  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_Blond  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)
**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 $6 \times 6$ distance matrix evaluating the distance between each pair of faces.

**Usage**

```r
data(DistAlgo)
```

**Format**

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

**Source**

Abdi et al. (2005). [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

**References**


---

**DistanceFromSort**

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

---

**Description**

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). `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 $k$th 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 output of the function `DistanceFromSort` is used as input for the function `distatis`.

**Usage**

```r
DistanceFromSort(X)
```
**DistanceFromSort**

**Arguments**

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

**Details**

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

**Value**

DistanceFromSort returns a $I \times I \times K$ array of distances

**Author(s)**

Herve Abdi

**References**

See examples in


These papers are available from [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

**See Also**

distatis

**Examples**

```r
# 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  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_Blond  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**

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

---

**Description**

Implements the DISTATIS method which a 3-way generalization of metric multidimensional scaling (a.k.a. classical MDS or principal coordinate analysis). **distatis** takes a set of $K$ distance matrices describing a set of $I$ observations and computes (1) a set of factor scores that describes the similarity structure of the 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) for the observations that best describes the similarity structure of the observations and (3) 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 matrix. **distatis** can also be applied to a set of covariance matrices.
Usage

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

Arguments

LeCube2Distance

an "observations × observations × 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) or MFA (default) that normalizes each matrix so that its first eigenvalue is equal to one.

Distance

if TRUE (default) the matrices are distance matrices, if FALSE they are covariance matrices.

RV

TRUE (default) we use the $R_V$ coefficient to compute the $\alpha$, FALSE we use the matrix scalar product.

nfact2keep

Number of factors to keep for the computation of the factor scores of the observations.

compact

if FALSE we provide detailed output, if TRUE we send back only the $\alpha$ weights (this option is used to make the bootstrap routine BootFromCompromise more efficient).

Details

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 $\alpha$ weights
- res.Cmat$value The eigenvalues of the $C$ matrix
- res.Cmat$g The factor scores for the $C$ matrix

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.
distatis

- res.Splus$Splus The compromise (linear combination of the SCP’s’)
- res.Splus$ProjectionMatrix The projection matrix used to compute factor scores and partial factor scores.
- res.Splus$F The factor scores for the observations.
- res.Splus$PartialF an $I \times nf2keep \times K$ array. Contains the partial factors for the distance matrices.

Author(s)

Herve Abdi

References


The $R_V$ coefficient is described in


(These papers are available from www.utdallas.edu/~herve)

See Also

GraphDistatisAll GraphDistatisBoot GraphDistatisCompromise GraphDistatisPartial GraphDistatisRv
DistanceFromSort BootFactorScores BootFromCompromise

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)
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) a Rv map.

Usage

```r
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 ($res4Splus$F from 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 $I$ matrix (with $I = # observations$) of color names for the observations. If NULL (default), prettyGraphs chooses.
- **participant.colors**: A $I$ 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.
GraphDistatisAll

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 will plot peeled hulls (from prettyGraphs package).

fill
when TRUE, fill in the ellipse with color. Related to ellipses only.

fill.alpha
transparency index when filling in the ellipses. Related 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)
# (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 partial factor score for the beers for the assessors are in
# testDistatis$res4Splus$PartialF
#
# 4. Get the bootstrapped factor scores (with default 1000 iterations)
BootF <- BootFactorScores(testDistatis$res4Splus$PartialF)
#---------------------------------------------------------------
# 5. Create the Graphics with GraphDistatisAll
#
GraphDistatisBoot

Description

GraphDistatisBoot plots maps of the factor scores of the observations from a \texttt{distatis} analysis. GraphDistatisBoot gives a map of the factors scores of the observations plus the boostrapped confidence intervals drawn as "Confidence Ellipsoids" at 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 ($res4Splush$ from \texttt{distatis})
- FBoot: is the bootstrapped factor scores array (FBoot obtained from \texttt{BootFactorsScores} or \texttt{BootFromCompromise})
- axis1: The dimension for the horizontal axis of the plots.
- axis2: The dimension for the vertical axis of the plots.
- 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.
- constraints: constraints for the axes
- nude: When TRUE do not plot the names of the observations
- Ctr: Contributions of each observation. If NULL (default), these are computed from FS
- lwd: Thickness of the line plotting the ellipse or hull.
- ellipses: a boolean. When TRUE will plot ellipses (from car package). When FALSE will plot peeled hulls (from prettyGraphs package).
- fill: when TRUE, fill in the ellipse with color. Related to ellipses only.
- fill.alpha: transparency index when filling in the ellipses. Related to ellipses only.
- percentage: A value to determine the percent coverage of the bootstrap partial factor scores to provide ellipse or hull confidence intervals.
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.

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:


These papers are available from [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

See Also

- GraphDistatisAll
- GraphDistatisCompromise
- GraphDistatisPartial
- GraphDistatisBoot
- GraphDistatisRv
- distatis

Examples

```r
# 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)
# (uses the function DistanceFromSort)
DistanceCube <- DistanceFromSort(Sort)
#-----------------------------------------------
# 3. Call the DISTATIS routine with the cube of distance as parameter
```
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 default 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 ($res4Plus$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 $I$ matrix (with $I$ = # observations) of color names for the observations. If NULL (default), prettyGraphs chooses.

zetitle
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.
GraphDistatisCompromise

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


see [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

See Also

- `GraphDistatisAll`
- `GraphDistatisCompromise`
- `GraphDistatisPartial`
- `GraphDistatisBoot`
- `GraphDistatisRv`
- `distatis`

Examples

```r
# 1. Load the DistAlgo data set (available from the DistatisR package)
data(DistAlgo)
# DistAlgo is a 6x6x4 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$res4Plus$F are the factors scores for the 6 observations (i.e., faces)
# DistatisAlgo$res4Plus$PartialF are the partial factors scores
##(i.e., one set of factor scores per algorithm)
GraphDistatisCompromise(DistatisAlgo$res4Plus$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**

```r
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 `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 $I$ matrix (with $I = \#$ observations) of color names for the observations. If NULL (default), prettyGraphs chooses.

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

- **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


see [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

**See Also**

- `GraphDistatisAll`
- `GraphDistatisCompromise`
- `GraphDistatisPartial`
- `GraphDistatisBoot`
- `GraphDistatisRv`
- `distatis`

**Examples**

```r
# 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 $\alpha$ 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 ($res4Cmat$ 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 $I$ 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 NULL (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:


The $R_V$ coefficient is described in


These papers are available from www.utdallas.edu/~herve

**See Also**

GraphDistatisAll GraphDistatisCompromise GraphDistatisPartial GraphDistatisBoot GraphDistatisRv distatis

**Examples**

```R
# 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!
```

---

**mmds**

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

**Description**

Perform an MMDS of a (Euclidean) distance matrix measured between a set of weighted objects.

**Method:** Transform the distance matrix into a (double centered) covariance matrix which is then analyze via its eigen-decomposition. The factor score 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.
mmds

Usage

mmds(DistanceMatrix, masses=NULL)

Arguments

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

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 of each dimension.
Contributions give the proportion of explained variance by an object for a dimension.

Author(s)

Herve Abdi

References


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),
  ...)}
print.Cmat

Description

Print C matrix results.

Usage

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

Arguments

x  
a list that contains items to make into the Cmat class.

...  
inherited/passed arguments for S3 print method(s).

Author(s)

Derek Beaton
print.DistatisR  \hspace{1cm} Print DistatisR results

**Description**

Print DistatisR results.

**Usage**

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

**Arguments**

- `x` a list that contains items to make into the DistatisR class.
- `...` inherited/passed arguments for S3 print method(s).

**Author(s)**

Derek Beaton

---

print.Splus  \hspace{1cm} Print S+ matrix results

**Description**

Print S+ matrix results.

**Usage**

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

**Arguments**

- `x` a list that contains items to make into the Splus class.
- `...` inherited/passed arguments for S3 print method(s).

**Author(s)**

Derek Beaton
**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.

**Usage**

```r
data(SortingBeer)
```

**Format**

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

**Source**

Abdi et al. (2007). [www.utdallas.edu/~herve](http://www.utdallas.edu/~herve)

**References**


---

**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. 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.

**Usage**

```r
data(SortingSpice)
```

**Format**

a data frame file containing 21 columns, 16 rows plus the names of the rows and the columns.
Source
Chollet et al. (2013). www.utdallas.edu/~herve

References
Index

*Topic **DistatisR**
  - Chi2Dist, 9
  - Chi2DistanceFromSort, 11
  - DistAlgo, 13
  - DistanceFromSort, 13
  - GraphDistatisBoot, 20
  - GraphDistatisCompromise, 22
  - GraphDistatisPartial, 24
  - GraphDistatisRv, 25
  - mmds, 27
  - SortingBeer, 31
  - SortingSpice, 31
*Topic **bootstrap**
  - BootFactorScores, 5
  - BootFromCompromise, 7
*Topic **datasets**
  - DistAlgo, 13
  - SortingBeer, 31
  - SortingSpice, 31
*Topic **distatis**
  - distatis, 15
  - GraphDistatisAll, 18
*Topic **mds**
  - distatis, 15
  - GraphDistatisAll, 18
  - GraphDistatisCompromise, 22
  - GraphDistatisPartial, 24
*Topic **package**
  - DistatisR-package, 2
*Topic **print**
  - print.Cmat, 29
  - print.DistatisR, 30
  - print.Splus, 30
*Topic **sample**
  - BootFactorScores, 5
  - BootFromCompromise, 7

BootFactorScores, 3, 5, 8, 17, 18, 20
BootFromCompromise, 3, 6, 7, 16–18, 20

Chi2Dist, 9
Chi2DistanceFromSort, 10, 11
CovSTATIS(distatis), 15
covstatis(distatis), 15
DistAlgo, 13
DistanceFromSort, 3, 13, 17
DiSTATIS(distatis), 15
distatis, 3, 10–14, 15, 19–21, 23–25, 27, 28
DiSTATISR(DistatisR-package), 2
DistatisR(DistatisR-package), 2
DistatisR-package, 2
GraphDistatisAll, 3, 17, 18, 19, 21, 23, 25, 27
GraphDistatisBoot, 3, 6, 8, 17–19, 20, 21, 23, 25, 27
GraphDistatisCompromise, 3, 17–19, 21, 22, 23, 25, 27, 28
GraphDistatisPartial, 3, 17–19, 21, 23, 24, 25, 27
GraphDistatisRv, 3, 17–19, 21, 23, 25, 27
mmds, 3, 9, 10, 27
prettyGraphs, 3
print.Cmat, 29
print.DistatisR, 30
print.Splus, 30
Sort(SortingBeer), 31
SortingBeer, 31
SortingSpice, 31
SortSpice(SortingSpice), 31