Package 'Anthropometry'

August 11, 2014

Type Package

Title Statistical methods for anthropometric data oriented towards the ergonomic design of products

Version 1.0

Date 2014-03-07

Author Guillermo Vinue, Irene Epifanio, Amelia Simo, M. Victoria Ibanez, Juan Domingo, Guillermo Ayala

Maintainer Guillermo Vinue <Guillermo.Vinue@uv.es>

Description

This package brings together some statistical methodologies especially developed to analyze anthropometric data. These methods are aimed at providing effective solutions to some commons problems related to Ergonomics and Anthropometry. They are based on clustering, the statistical concept of data depth, the statistical shape analysis and the archetypal analysis.

License GPL (>= 2)

URL http://www.r-project.org, http://www.uv.es/vivigui

Depends R (>= 3.0.0)

Imports shapes, rgl, archetypes, nnls, depth, FNN, ICGE, cluster

Suggests knitr, biclust, calibrate, mvtnorm, SportsAnalytics,RColorBrewer, plotrix, abind

VignetteBuilder knitr

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-03-07 09:41:02

R topics documented:

`Anthropometry-package` *Statistical methods for anthropometric data oriented towards the ergonomic design of products*

Description

This package brings together some statistical methodologies especially developed to analyze anthropometric data. These methods are aimed at providing effective solutions to some commons problems related to Ergonomics and Anthropometry. They are based on clustering, the statistical concept of data depth, the statistical shape analysis and the archetypal analysis.

Details

Package: Anthropometry
Type: Package
Version: 1.0
Date: 2014-03-07
License: GPL-2
LazyLoad: yes
LazyData: yes

accommodation: Data preprocessing before computing archetypes.
Anthropometry-internalArchetypoids: Several internal functions to compute and represent archetypes and archetypoids.
Anthropometry-internalDepth: Several internal functions to clustering based on the L1 data depth.
Anthropometry-internalHIPAM: Several internal functions used by both $HIPAM_MO$ and $HIPAM_IMO$ algorithms.
Anthropometry-internalPlotTree: Several internal functions used to build the HIPAM plot tree.
archetypesBoundary: Archetypal analysis in multivariate accommodation problem.
archetypoids: Finding archetypoids.
CCbiclustAnthropo: Cheng and Church biclustering algorithm applied to anthropometric data.
cdfDiss: CDF for the dissimilarities between women and computed medoids and standard prototypes.
checkBranchLocalIMO: Evaluation of the candidate clustering partition in $HIPAM_IMO$.
checkBranchLocalMO: Evaluation of the candidate clustering partition in $HIPAM_MO$.
cMDSwomen: Description of the dissimilarities between women's trunks.
compPerc: Computing percentiles of a certain archetypoid.
cube8: Cube of 8 landmarks.
cube34: Cube of 34 landmarks.
dataDemo: Demo database of the Spanish anthropometric survey.
dataUSAF: USAF 1967 database.
figures8landm: Figures with labelled landmarks.
getBestPamsamIMO: Generation of the candidate clustering partition in $HIPAM_IMO$.
getBestPamsamMO: Generation of the candidate clustering partition in $HIPAM_MO$.
GetDistMatrix: Dissimilarity matrix between individuals and prototypes.

HartiganShapes: Hartigan-Wong k-means for 3D shapes.
hipamAnthropom: HIPAM algorithm for anthropometric data.
hipamBigGroups: Hipam medoids of the clusters with more than 2 elements.
indivNearest: Nearest individuals to archetypes.
landmarks: Landmarks representing the woman's body.
LloydShapes: Lloyd k-means for 3D shapes.
optraProcrustes: Auxiliary optra subroutine of the Hartigan-Wong k-means for 3D shapes.
outlierHipam: Individuals of the hipam clusters with 1 or 2 elements.
overlappingRows: Overlapped biclusters by rows.
parallelepiped8: Parallelepiped of 8 landmarks.
parallelepiped34: Parallelepiped of 34 landmarks.
plotMedoids: Medoids representation.
plotTreeHipam: HIPAM dendogram.
plotTrimmOutl: Trimmed or outlier observations representation.
qtranProcrustes: Auxiliary qtran subroutine of the Hartigan-Wong k-means for 3D shapes.
screeArchetyp: Screeplot of archetypes and archetypoids.
shapes3dMod: 3D shapes plot.
skeletonsArchet: Skeleton plots of archetypal individuals.
stepArchetypesMod: Archetype algorithm to raw data.
stepArchetypoids: Run the archetypoid algorithm several times.
TDDclust: Trimmed clustering based on L1 data depth.
trimmedLloydShapes: Trimmed Lloyd k-means for 3D shapes.
trimmedoid: Trimmed k-medoids algorithm.
trimowa: Trimmed PAM with OWA operators.
WeightsMixtureUB: Calculation of the weights for the OWA operators.
xyplotPCA: PC scores for archetypes.

Author(s)

Guillermo Vinue <Guillermo.Vinue@uv.es>, Irene Epifanio, Amelia Simo, M. Victoria Ibanez, Juan Domingo, Guillermo Ayala

---

`accommodation` *Data preprocessing before computing archetypes*

Description

This function allows us to preprocess the data before computing archetypes and archetypoids. First, depending on the problem, it is possible to standardize the data or not. Second, it is possible to use the Mahalanobis distance or a depth procedure to select the accommodated subsample of data.

Usage

```
accommodation(dataRaw,stand,percAccomm,mahal=TRUE)
```

Arguments

dataRaw	Raw data. Each row corresponds to an observation and each column corresponds to an anthropometric variable. All variables are numeric.
stand	A logical value. If TRUE (FALSE) the data are (not) standardized. This option will depend on the problem.
percAccomm	Percentage of the population to accommodate (value between 0 and 1). When this percentage is equal to 1 all the individuals will be accommodated.
mahal	If percAccom is different from 1, then mahal=TRUE (mahal=FALSE) indicates that the Mahalanobis distance (a depth procedure) will be used to select the accommodated subsample of data.

Details

In somes cases, the depth procedure has the disadvantage that the desired percentage of accommodation is not under control of the analyst and it could not coincide exactly with percAccomm.

Value

A list with the following elements:

data: Database after preprocessing.

indivYes: Individuals who belong to data.

indivNo: Individuals discarded in the accommodation procedure.

Author(s)

Irene Epifanio and Guillermo Vinue

References

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

Genest, M., Masse, J.-C., and Plante, J.-F., (2012). **depth**: Depth functions tools for multivariate analysis. R package version 2.0-0.

Examples

```
## Not run:
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data pre-processing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)
preproc <- accommodation(mpulg,TRUE,0.95,FALSE)

## End(Not run)
```

`archetypesBoundary` *Archetypal analysis in multivariate accommodation problem*

Description

This function allows us to reproduce the results shown in section 2.2.2 and section 3.1 of Epifanio et al. (2013). In addition, from the results provided by this function, the other results shown in section 3.2 and section 3.3 of the same paper can be also reproduced (see section *examples* below).

Usage

```
archetypesBoundary(data,numArchet,verbose,nrep)
```

Arguments

`data`	USAF 1967 database (see `dataUSAF`). Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
`numArchet`	Number of archetypes.
`verbose`	Logical value. If TRUE, some details of the execution progress are shown (this is the same argument as that of the `stepArchetypes` function of the **archetypes** R package (Eugster (2009))).
`nrep`	For each archetype run `archetypes` nrep times (this is the same argument as that of the `stepArchetypes` function of **archetypes**).

Details

Before using this function, the more extreme (100 - percAcomm*100)% observations must be removed by means of the `accommodation` function. To this end, it is recommended that you use the Mahalanobis distance. In this case, the depth procedure has the disadvantage that the desired percentage of accommodation is not under control of the analyst and it could not coincide exactly with that indicated.

Value

A list with `numArchet` elements. Each element is a list of class attribute `stepArchetypes` with `nrep` elements.

Note

We would like to note that, some time after publishing the paper Epifanio et al. (2013), we found out that the `stepArchetypes` function standardizes the data by default (even when the data are already standardized) and this option is not always desired. In order to avoid this way to proceed, we have created the `stepArchetypesMod` function, which is used within `archetypesBoundary` instead of using `stepArchetypes`. Therefore, the results provided by `archetypesBoundary` allows us to reproduce the results of Epifanio et al. (2013) but they are now slightly different.

Author(s)

Irene Epifanio and Guillermo Vinue

References

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

Eugster, M. J., and Leisch, F., (2009). From Spider-Man to Hero - Archetypal Analysis in R, *Journal of Statistical Software* **30**, 1–23, http://www.jstatsoft.org/.

Zehner, G. F., Meindl, R. S., and Hudson, J. A., (1993). A multivariate anthropometric method for crew station design: abridged. Tech. rep. Ohio: Human Engineering Division, Armstrong Laboratory, Wright-Patterson Air Force Base.

See Also

archetypes, stepArchetypes, stepArchetypesMod, dataUSAF, indivNearest, accommodation

Examples

```
## Not run:
#The following R code allows us to reproduce the results of the paper Epifanio et al. (2013).
#First,the USAF 1967 database is read and preprocessed (Zehner et al. (1993)).
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#Procedure and results shown in section 2.2.2 and section 3.1:
res <- archetypesBoundary(preproc$data,15,FALSE,3)

#Results shown in section 3.2 (figure 3):
screeplot(res)

#3 archetypes:
a3 <- archetypes::bestModel(res[[3]])
archetypes::parameters(a3)
#7 archetypes:
a7 <- archetypes::bestModel(res[[7]])
archetypes::parameters(a7)
#Plotting the percentiles of each archetype:
#Figure 2 (b):
barplot(a3,preproc$data,percentiles=T,which="beside")
#Figure 2 (f):
barplot(a7,preproc$data,percentiles=T,which="beside")
```

```
#Results shown in section 3.3 related with PCA.
pznueva <- prcomp(preproc$data,scale=T,retx=T)
#Table 3:
summary(pznueva)
pznueva
#PCA scores for 3 archetypes:
p3 <- predict(pznueva,archetypes::parameters(a3))
#PCA scores for 7 archetypes:
p7 <- predict(pznueva,archetypes::parameters(a7))
#Representing the scores:
#Figure 4 (a):
xyplotPCA(p3[,1:2],pznueva$x[,1:2],data.col=gray(0.7),atypes.col=1,atypes.pch=15)
#Figure 4 (b):
xyplotPCA(p7[,1:2],pznueva$x[,1:2],data.col=gray(0.7),atypes.col=1,atypes.pch=15)

#Percentiles for 7 archetypes (table 5):
Fn <- ecdf(preproc$data)
round(Fn(archetypes::parameters(a7)) * 100)

#Which are the nearest individuals to archetypes?:
#Example for three archetypes:
ras <- rbind(archetypes::parameters(a3),preproc$data)
dras <- dist(ras,method="euclidean",diag=F,upper=T,p=2)
mdras <- as.matrix(dras)
diag(mdras) = 1e+11
i <- 3
nearest <- sapply(1:i,indivNearest,i,mdras)

#In addition, we can turn the standardized values to the original variables.
p <- archetypes::parameters(a7)
m <- sapply(mpulg,mean)
s <- sapply(mpulg,sd)
d <- p
for(i in 1 : 6){
 d[,i] = p[,i] * s[i] + m[i]
}
#Table 7:
t(d)

## End(Not run)
```

`archetypoids` *Finding archetypoids*

Description

Archetypoid algorithm. It is based on the PAM clustering algorithm. It is made up of two phases (a BUILD phase and a SWAP phase). In the BUILD phase, an initial set of archetypoids is determined. Unlike PAM, this collection is not derived in a stepwise format. Instead, it is suggested you choose the set made up of the nearest individuals returned by the `archetypes` function of the **archetypes**

R package (Eugster et al. (2009)). This set can be defined in two different ways, see next section *arguments*. The goal of the SWAP step is the same as that of the SWAP step of PAM, but changing the objective function. The initial vector of archetypoids is attempted to be improved. This is done by exchanging selected individuals for unselected individuals and by checking whether these replacements reduce the objective function of the archetypoid analysis problem.

More details are given in Vinue et al. (2014) (submitted).

Usage

```
archetypoids(i,data,huge=200,step,init,ArchObj,nearest,sequ,aux)
```

Arguments

`i`	Number of archetypoids.
`data`	Data matrix. Each row corresponds to an observation and each column corresponds to an anthropometric variable. All variables are numeric.
`huge`	This is a penalization added to solve the convex least squares problems regarding the minimization problem to estimate archetypoids, see Eugster et al. (2009). Default value is 200.
`step`	Logical value. If TRUE, the archetypoid algorithm is executed repeatedly within `stepArchetypoids`. Therefore, this function requires the next argument `init` (but neither the `ArchObj` nor the `nearest` arguments) that specifies the initial vector of archetypoids, which has been already computed within `stepArchetypoids`. If FALSE, the archetypoid algorithm is executed once. In this case, the `ArchObj` and `nearest` arguments are required to compute the initial vector of archetypoids.
`init`	Initial vector of archetypoids for the BUILD phase of the archetypoid algorithm. It is computed within `stepArchetypoids`. See next `nearest` argument to know how this vector is calculated.
`ArchObj`	The list returned by the `stepArchetypesMod` function. This function is a slight modification of the original `stepArchetypes` function of **archetypes** to apply the archetype algorithm to raw data. The `stepArchetypes` function standardizes the data by default and this option is not always desired. This list is needed to compute the nearest individuals to archetypes. Required when `step=FALSE`.
`nearest`	Initial vector of archetypoids for the BUILD phase of the archetypoid algorithm. Required when `step=FALSE`. This argument is a logical value: if TRUE (FALSE), the *nearest* (*which*) vector is calculated. Both vectors contain the nearest individuals to the archetypes returned by the `archetypes` function of **archetypes** (In Vinue et al. (2014), archetypes are computed after running the archetype algorithm twenty times). The *nearest* vector is calculated by computing the Euclidean distance between the archetypes and the individuals and choosing the nearest. It is used in Epifanio et al. (2013). The *which* vector is calculated by identifying consecutively the individual with the maximum value of alpha for each archetype, until getting the number of archetypes defined. It is used in Eugster (2012).

sequ	Logical value. It indicates whether a sequence of archetypoids (TRUE) or only a single number of them (FALSE) is computed. It is determined by the number of archetypes computed by means of stepArchetypesMod.
aux	If sequ=FALSE, this value is equal to i-1 since for a single number of archetypoids, the list associated with the archetype object only has one element.

Details

As mentioned, this algorithm is based on PAM. These types of algorithms aims to find good solutions in a short period of time, although not necessarily the best solution. Otherwise, the global minimum solution may always be obtained using as much time as it would be necessary, but this would be very inefficient computationally.

Value

A list with the following elements:

archet: Final vector of k archetypoids.

rss: Residual sum of squares corresponding to the final vector of k archetypoids.

archet_ini: Vector of initial archetypoids (*nearest* or *which*).

Note

It may be happen that archetypes does not find results for *k* archetypes. In this case, it is not possible to calculate the vector of nearest individuals and consequently, the vector of archetypoids. Therefore, this function will return an error message.

Author(s)

Irene Epifanio and Guillermo Vinue

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. *Submitted for publication*.

Cutler, A., and Breiman, L., (1994). Archetypal Analysis, *Technometrics* **36**, 338–347.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

Eugster, M. J., and Leisch, F., (2009). From Spider-Man to Hero - Archetypal Analysis in R, *Journal of Statistical Software* **30**, 1–23, http://www.jstatsoft.org/.

Eugster, M. J. A., (2012). Performance profiles based on archetypal athletes, *International Journal of Performance Analysis in Sport* **12**, 166–187.

See Also

stepArchetypesMod, archetypes, stepArchetypoids

Examples

```
## Not run:
#SPORTIVE EXAMPLE:
#Database:
if(nzchar(system.file(package = "SportsAnalytics"))){
 data("NBAPlayerStatistics0910", package = "SportsAnalytics")
}
mat <- NBAPlayerStatistics0910[,c("TotalMinutesPlayed","FieldGoalsMade")]
rownames(mat) <- NULL

#Calculating archetypes by using the archetype algorithm:
#Data preprocessing:
preproc <- accommodation(mat,stand=TRUE,percAccomm=1)

#For reproducing results, seed for randomness:
set.seed(4321)
#Run archetype algorithm repeatedly from 1 to 15 archetypes:
lass15 <- stepArchetypesMod(data=preproc,k=1:15,verbose=FALSE,nrep=20)
screeplot(lass15)

#Calculating real archetypes:
i <- 3 #number of archetypoids.
res <- archetypoids(i,preproc,huge=200,step=FALSE,ArchObj=lass15,nearest=TRUE,sequ=TRUE)
arquets <- NBAPlayerStatistics0910[res[[1]],c("Name","TotalMinutesPlayed","FieldGoalsMade")]
res_which <- archetypoids(i,preproc,huge=200,step=FALSE,ArchObj=lass15,nearest=FALSE,sequ=TRUE)
arquets_eug <- NBAPlayerStatistics0910[res_which[[1]],
                  c("Name","TotalMinutesPlayed","FieldGoalsMade")]

col_pal <- RColorBrewer::brewer.pal(7, "Set1")
col_black <- rgb(0, 0, 0, 0.2)

plot(mat, pch = 1, col = col_black, xlim = c(0,3500), main = "NBA archetypal basketball
     players \n obtained in Eugster (2012) \n and with our proposal",
     xlab = "Total minutes played", ylab = "Field goals made")
points(mat[as.numeric(rownames(arquets)),], pch = 4, col = col_pal[1])
points(mat[as.numeric(rownames(arquets_eug)),], pch = 4, col = col_pal[1])
text(mat[as.numeric(rownames(arquets_eug)),][2,1],
     mat[as.numeric(rownames(arquets_eug)),][2,2],
     labels = arquets_eug[2,"Name"], pos = 4, col = "blue")
plotrix::textbox(c(50,800), 50, "Travis Diener")
plotrix::textbox(c(2800,3500), 780, "Kevin Durant", col = "blue")
plotrix::textbox(c(2800,3500), 270, "Jason Kidd", col = "blue")
legend("topleft",c("archetypes of Eugster","archetypes of our proposal"),
       lty= c(1,NA), pch = c(NA,22), col = c("blue","black"))


#If a specific number of archetypes is computed only:
i=3
set.seed(4321)
lass3 <- stepArchetypesMod(data=preproc,k=i,verbose=FALSE,nrep=3)
res3 <- archetypoids(i,preproc,huge=200,step=FALSE,ArchObj=lass3,nearest=TRUE,sequ=FALSE,aux=2)
arquets3 <- NBAPlayerStatistics0910[res3[[1]],c("Name","TotalMinutesPlayed","FieldGoalsMade")]
```

```
#COCKPIT DESIGN PROBLEM:
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#For reproducing results, seed for randomness:
set.seed(2010)
#Run archetype algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 20
lass <- stepArchetypesMod(data=preproc$data,k=1:numArch,verbose=FALSE,nrep=nrep)
screeplot(lass)

i <- 3 #number of archetypoids.
res <- archetypoids(i,preproc$data,huge=200,step=FALSE,ArchObj=lass,nearest=TRUE,sequ=TRUE)
res_which <- archetypoids(i,preproc$data,huge=200,step=FALSE,ArchObj=lass,nearest=FALSE,sequ=TRUE)

## End(Not run)
```

`CCbiclustAnthropo` *Cheng and Church biclustering algorithm applied to anthropometric data*

Description

This function is the implementation in R of the algorithm that uses the Cheng and Church biclustering method (from now on, CC) to find size groups (biclusters) and disaccommodated individuals.

Designing lower body garments depends not only on the waist circumference (the principal dimension in this case), but also on other secondary control dimensions (for upper body garments the bust circumference is usually required only). Biclustering identifies groups of observations with a similar pattern in a subset of attributes instead of in the whole of them. Therefore, it seems to be more interesting to use a biclustering algorithm with a set of lower body variables.

In Vinue et al. (2014) (in progress), the way of proceeding was as follows: first, all the body variables related to the lower body part included in the Spanish anthropometric survey were chosen (there were 36). Second, the data set was divided into twelve segments (classes) using waist circumference values according to the European standard. Part 3: Measurements and intervals. Finally, the CC algorithm was applied to each waist class.

Usage

```
CCbiclustAnthropo(data,waist,waistCirc,lowerVars,nsizes,nBic,diffRanges,percDisac,dir)
```

Arguments

data	Data matrix. Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
waist	Vector containing the waist values of the individuals.
waistCirc	data is segmented into twelve waist classes. This vector contains the waist values to define each one of the twelve classes.
lowerVars	Lower body dimensions.
nsizes	Number of waist sizes.
nBic	Maximum number of biclusters to be found in each waist size.
diffRanges	List with nsizes elements. Each element is a vector whose extremes indicate the acceptable boundaries for selecting variables with a similar scale. This is needed because CC may be very influenced in case of variables involved in the study are on very different scales.
percDisac	Proportion of no accommodated sample.
dir	Working directory where to save the results.

Details

Interesting results in terms of apparel design were found: an efficient partition into different biclusters was obtained. All individuals in the same bicluster can wear a garment designed for the particular body dimensions (waist and other variables) which were the most relevant for defining the group. Each group is represented by the median woman. Because the CC algorithm is nonexhaustive, i.e, some rows (and columns) do not belong to any bicluster, this property can be used to fix a proportion of no accommodated sample.

This approach was descriptive and exploratory. It is emphasized that this function cannot be used with dataDemo, because this data file does not contain variables related to the lower body part in addition to waist and hip. However, this function is included in the package in the hope that it could be helpful or useful for other researchers.

Value

A list with the following elements:

res: List with nsizes elements. Each element contains the biclustering results for each waist segment.

dims: List with nsizes elements. Each element contains the number of variables with a similar scale in each waist segment.

delta: List with nsizes elements. Each element contains the delta parameter of the CC algorithm for each waist segment.

disac: List with nsizes elements. Each element contains the number of women who not belong to any bicluster for each waist segment.

mat: List with nsizes elements. Each element contains the matrix showing which rows belong to each bicluster for each waist segment. This matrix allow us to know whether there are rows that belong to more than one bicluster, that is to say, whether there are overlapping biclusters. This is

very important in our application because each individual must be assigned to a single size. See the *Note* section.

tab_acc: List with `nsizes` elements. Each element is a list with four elements. The first component indicates how many individuals belong to a single bicluster and how many do not belong to any bicluster. The second component refers to the number of biclusters found in each segment. The third one indicates the number of women that belong to each waist segment. The fourth one coincides with the *disac* element.

ColBics: List with `nsizes` elements. Each element contains the variables that belong to each bicluster for each waist segment.

Note

In order to know whether a row belongs to more than one bicluster, we count the number of 0s in each row of the *mat* matrix returned by this function (see the *Value* section).

In case of there are res@Number - 1 0s in each row of *mat*, then each row belongs to only one bicluster. The *mat* matrix indicates with an 1 the rows that make up of the bicluster 1, with a 2 those rows that make up of the bicluster 2 and so on. In addition, it indicates with a 0 the rows that do not belong to any bicluster. Therefore, in order to check overlapping, every row must have a number of 0s equal to the total number of biclusters minus one. This one will indicate that that row belongs to a single bicluster. Otherwise, every row must have a number of 0s equal to the total number of biclusters. In this case, that row does not belong to any bicluster.

For instance, if we find two biclusters, there should be one or two 0s in each row in case of no overlapping.

Author(s)

Guillermo Vinue

References

Vinue, G., and Ibanez, M. V., (2014), *Data depth and Biclustering applied to anthropometric data. Exploring their utility in apparel design*. In progress.

Cheng, Y., and Church, G., (2000). Biclustering of expression data. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology* **8**, 93–103.

Kaiser, S., and Leisch, F., (2008). A Toolbox for Bicluster Analysis in R. Tech.rep., Department of Statistics (University of Munich).

Alemany, S., Gonzalez, J. C., Nacher, B., Soriano, C., Arnaiz, C., and Heras, H., (2010). Anthropometric survey of the Spanish female population aimed at the apparel industry. *Proceedings of the 2010 Intl. Conference on 3D Body scanning Technologies*, 307–315.

European Committee for Standardization. Size designation of clothes. Part 2: Primary and secondary dimensions. (2002).

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

`overlappingRows`

Examples

```
## Not run:
#Note: package biclust needed.
#This is an example of using this function with a certain database
#made up of body dimensions related to the lower body part.
data <- dataUser[(waist >= 58) & (waist < 115),] #dataUser is the user database.
rownames(data) <- 1:dim(data)[1]

waist <- data[,"WaistCircumference"]

waist_4 <- seq(58, 86, 4)
waist_6 <- seq(91, 115, 6)
waistCirc <- c(waist_4,waist_6)
nsizes <- length(waistCirc)

#Position of the body variables in the database:
lowerVars <- c(14, 17:25, 27, 28, 65:73, 75, 77:81, seq(100, 116, 2))

nBic <- c(2, 2, 4, rep(5, 7), 3, 3)
diffRanges <- list(c(14,20), c(24,30), c(24,30), c(33,39), c(29,35), c(29,35),
                   c(28,35), c(31,38), c(31,38), c(30,37), c(26,33), c(25,32))
percDisac <- 0.01
dir <- "/home/guillermo/"

res_bicl_antropom <- CCbiclustAnthropo(data,waist,waistCirc,lowerVars,
                                       nsizes,nBic,diffRanges,percDisac,dir)

## End(Not run)
```

`cdfDiss` *CDF for the dissimilarities between women and computed medoids and standard prototypes*

Description

This function allows us to calculate the cumulative distribution functions for the dissimilarities between all the women and the medoids obtained with the `trimowa` algorithm and for the dissimilarities between all the women and the standard prototypes defined by the European standard. Part 3: Measurements and intervals. In both cases, the dissimilarities have been computed by using the dissimilarity function obtained with `GetDistMatrix`.

These types of plots can also be used to identify the expected range of the dissimilarities, that is to say, the values between the 10 and 90th percentiles.

This function was used to obtain the Fig. 11 of Ibanez et al. (2012).

Usage

```
cdfDiss(min_med,min_med_UNE,main,xlab,ylab,leg,cexLeg,...)
```

Arguments

min_med	Vector with the dissimilarities between all the women and the medoids obtained with trimowa.
min_med_UNE	Vector with the dissimilarities between all the women and the standard prototypes.
main	A title for the plot.
xlab	A title for the x axis.
ylab	A title for the y axis.
leg	A character vector to appear in the legend.
cexLeg	Character expansion for the legend.
...	Further graphical parameters.

Value

A device with the desired plot.

Author(s)

Guillermo Vinue

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

dataDemo, WeightsMixtureUB, trimowa, GetDistMatrix

Examples

```
## Not run:
#Loading the data to apply the trimowa algorithm:
dataDef <- dataDemo
dim(dataDef)
#[1] 600   5
num.variables <- dim(dataDef)[2]
bust <- dataDef$bust
chest <- dataDef$chest

orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)

bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
```

```
K <- 3 ; alpha <- 0.01 ; niter <- 6 ; Ksteps <- 7

ahVect <- c(23,28,20,25,25)

res_trimowa <- list()
for (i in 1 : (nsizes-1)){
 data = dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
 res_trimowa[[i]] <- trimowa(data,w,K,alpha,niter,Ksteps,ahVect=ahVect)
}

#Medoids obtained with the trimowa algorithm:
medoids <- list()
for (i in 1 : (nsizes-1)){
 medoids[[i]] <- res_trimowa[[i]]$meds
}
medoids <- unlist(medoids)
length(medoids)
#[1] 36
meds <- dataDef[medoids,]

regr <- lm(chest ~ bust)

#Prototypes defined by the European standard:
hip_UNE <- c(seq(84,112,4),seq(117,132,5))          ; hip <- rep(hip_UNE,3)
waist_UNE <- c(seq(60,88,4),seq(94,112,6))          ; waist <- rep(waist_UNE,3)
bust_UNE <- c(seq(76,104,4),seq(110,128,6))         ; bust <- rep(bust_UNE,3)
chest_UNE <- predict(regr,list(bust=bust_UNE))      ; chest <- rep(chest_UNE,3)
necktoground <- c(rep(130,12),rep(134,12),rep(138,12))

medsUNE <- data.frame(chest,necktoground,waist,hip,bust)
dim(medsUNE)
#[1] 36  5

dataAll <- rbind(dataDef,meds,medsUNE)
dim(dataAll)
#[1] 672   5

bh <- (apply(as.matrix(log(dataAll)),2,range)[2,]
       - apply(as.matrix(log(dataAll)),2,range)[1,]) / ((K-1) * 8)
bl <- -3 * bh
ah <- c(28,20,30,25,23)
al <- 3 * ah
num.persons <- dim(dataAll)[1]
dataAllm <- as.matrix(dataAll)
dataAllt <- aperm(dataAllm, c(2,1))
dim(dataAllt) <- c(1,num.persons*num.variables)
rm(dataAllm)
D <- GetDistMatrix(dataAllt,num.persons,num.variables,w,bl,bh,al,ah,T)

f <- function(i, D){
 r <- min(D[i,601:636])
}
min_med <- sapply(1:600, f, D)
```

```
f1 <- function(i, D){
 r <- min(D[i,637:672])
}
min_med_UNE <- sapply(1:600, f1, D)

#CDF plot:
main <- "Comparison between sizing methods"
xlab <- "Dissimilarity"
ylab <- "Cumulative distribution function"
leg <- c("Dissimilarity between women and computed medoids",
         "Dissimilarity between women and standard prototypes")
cdfDiss(min_med,min_med_UNE,main,xlab,ylab,leg,cexLeg=0.7)

## End(Not run)
```

---

checkBranchLocalIMO *Evaluation of the candidate clustering partition in $HIPAM_IMO$*

Description

In the HIPAM algorithm, each (parent) cluster P is investigated to see if it can be divided further into new (child) clusters, or stop (in this case, P would be a terminal node).

In this version of HIPAM, called $HIPAM_IMO$, there are three different stopping criteria: First, if $|P| leq 2$, then P is a terminal node. If not, the second stopping refers to the INCA (Index Number Clusters Atypical) criterion (Irigoien et al. (2008)): if $INCA_k leq 0.2$ for all k, then P is a terminal node. Finally, the third stopping criteria uses the Mean Split Silhouette. See Vinue et al. (2013) for more details.

The foundation and performance of the HIPAM algorithm is explained in hipamAnthropom.

Usage

```
checkBranchLocalIMO(tree,x,i,maxsplit,asw.tol,local.const,orness,type,...)
```

Arguments

tree	The clustering tree being defined.
x	Data to be clustered.
i	A specific cluster of the clustering partition in a certain level of the tree.
maxsplit	The maximum number of clusters that any cluster can be divided when searching for the best clustering.
asw.tol	If this value is given, a tolerance or penalty can be introduced (asw.tol > 0 or asw.tol < 0, respectively) in the branch splitting procedure. Default value (0) is maintained. See page 154 of Wit et al. (2004) for more details.

local.const	If this value is given (meaningful values are those between -1 and 1), a proposed partition is accepted only if the associated asw is greater than this constant. Default option for this argument is maintained, that is to say, this value is ignored. See page 154 of Wit et al. (2004) for more details.
orness	Quantity to measure the degree to which the aggregation is like a min or max operation. See WeightsMixtureUB and GetDistMatrix.
type	Option 'IMO' for using $HIPAM_IMO$.
...	Other arguments that may be supplied.

Value

The new resulting classification tree.

Note

This function belongs to the $HIPAM_IMO$ algorithm and it is not solely used. That is why there is no section of *examples* in this help page. See hipamAnthropom.

Author(s)

This function was originally created by E. Wit et al., and it is available freely on http://www.math.rug.nl/~ernst/book/smida.html. We have adapted it to incorporate the second stopping criterion related to INCA.

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

Pollard, K. S., and van der Laan, M. J., (2002). A method to identify significant clusters in gene expression data. *Vol. II of SCI2002 Proceedings*, 318–325.

Irigoien, I., and Arenas, C., (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units, *Statistics in Medicine* **27**, 2948–2973.

Irigoien, I., Sierra, B., and Arenas, C., (2012). ICGE: an R package for detecting relevant clusters and atypical units in gene expression, *BMC Bioinformatics* **13** 1–29.

See Also

hipamAnthropom

checkBranchLocalMO *Evaluation of the candidate clustering partition in $HIPAM_MO$*

Description

In the HIPAM algorithm, each (parent) cluster P is investigated to see if it can be divided further into new (child) clusters, or stop (in this case, P would be a terminal node).

In this version of HIPAM, called $HIPAM_MO$, there are two different stopping criteria: First, if $|P| leq 2$, then P is a terminal node. If not, the second stopping criteria uses the Mean Split Silhouette. See Vinue et al. (2013) for more details.

The foundation and performance of the HIPAM algorithm is explained in hipamAnthropom.

Usage

```
checkBranchLocalMO(tree,x,i,maxsplit,asw.tol,local.const,orness,type,...)
```

Arguments

tree	The clustering tree being defined.
x	Data to be clustered.
i	A specific cluster of the clustering partition in a certain level of the tree.
maxsplit	The maximum number of clusters that any cluster can be divided when searching for the best clustering.
asw.tol	If this value is given, a tolerance or penalty can be introduced (asw.tol > 0 or asw.tol < 0, respectively) in the branch splitting procedure. Default value (0) is maintained. See page 154 of Wit et al. (2004) for more details.
local.const	If this value is given (meaningful values are those between -1 and 1), a proposed partition is accepted only if the associated asw is greater than this constant. Default option for this argument is maintained, that is to say, this value is ignored. See page 154 of Wit et al. (2004) for more details.
orness	Quantity to measure the degree to which the aggregation is like a min or max operation. See WeightsMixtureUB and GetDistMatrix.
type	Option 'MO' for using $HIPAM_MO$.
...	Other arguments that may be supplied.

Value

The new resulting classification tree.

Note

This function belongs to the $HIPAM_MO$ algorithm and it is not solely used. That is why there is no section of *examples* in this help page. See hipamAnthropom.

Author(s)

This function was originally created by E. Wit et al., and it is available freely on http://www.math.rug.nl/~ernst/book/smida.html.

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html

Pollard, K. S., and van der Laan, M. J., (2002). A method to identify significant clusters in gene expression data. *Vol. II of SCI2002 Proceedings*, 318–325.

See Also

hipamAnthropom

---

cMDSwomen *Description of the dissimilarities between women's trunks*

Description

Unlike archetypes, archetypoids can be computed when features are unavailable. Given a dissimilarity matrix, the classical multidimensional scaling (cMDS) can be applied to obtain a description of the dissimilarities.

In Vinue et al. (2014) (submitted), the dissimilarity matrix represents the dissimilarities between women's trunks. After applying the cMDS, the database described here is obtained. Then, the archetypoid algorithm can be applied to this database, see section *examples*.

Usage

```
cMDSwomen
```

Format

A matrix with 470 rows and 4 columns.

Source

Anthropometric survey of the Spanish female population.

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. Submitted for publication.

Alemany, S., Gonzalez, J. C., Nacher, B., Soriano, C., Arnaiz, C., and Heras, H., (2010). Anthropometric survey of the Spanish female population aimed at the apparel industry. *Proceedings of the 2010 Intl. Conference on 3D Body scanning Technologies*, 307–315.

Examples

```
## Not run:
#Database:
X <- cMDSwomen
X <- as.matrix(X)

#Computation of archetypes and archetypoids:
#For reproducing results, seed for randomness:
set.seed(2010)
#Run archetype algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 20
lass <- stepArchetypesMod(data=X,k=1:numArch,verbose=FALSE,nrep=nrep)
screeplot(lass)

rss_lass <- matrix(0,nrow=numArch,ncol=nrep)
for(i in 1:numArch){
  for(j in 1:nrep){
    rss_lass[i,j] <- lass[[i]][[j]]$rss
  }
}
(rss_lass_def <- apply(rss_lass,1,min,na.rm=T))

for(i in 1:numArch){
  temp <-  stepArchetypoids(i,TRUE,X,lass)
  filename <- paste("res",i,sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename,".RData",sep=""))
}

for(i in 1:numArch){
  temp <-  stepArchetypoids(i,FALSE,X,lass)
  filename <- paste("res",i,"_which",sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename,".RData",sep=""))
}

## End(Not run)
```

---

compPerc *Computing percentiles of a certain archetypoid*

Description

This function computes the percentiles of an archetypoid for a given variable. Once these percentile values have been calculated, they can be represented by means of a barplot.

Usage

```
compPerc(column,indiv,data,digits)
```

Arguments

column	Numeric variable (column of a data frame).
indiv	A certain archetypoid.
data	Data frame that contains the columns to be analyzed.
digits	Argument of the round function (it is a integer indicating the number of decimal places to be used).

Value

Numerical vector with the percentile values of an archetypoid.

Author(s)

Guillermo Vinue

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. Submitted for publication.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

See Also

archetypoids

Examples

```
## Not run:
#Cockpit design problem:
m <- dataUSAF
sel <- c(48,40,39,33,34,36)
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#For reproducing results, seed for randomness:
set.seed(2010)
```

```
#Run archetype algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 20
lass <- stepArchetypesMod(data=preproc$data,k=1:numArch,verbose=FALSE,nrep=nrep)

#Three archetypoids:
i <- 3
res <- archetypoids(i,preproc$data,huge=200,step=FALSE,ArchObj=lass,nearest=TRUE,sequ=TRUE)
aux <- res$archet

percs <- list()
for(j in 1:length(aux)){
 percs[[j]] <- sapply(1:dim(preproc$data)[2],compPerc,aux[j],preproc$data,0)
}
m <- matrix(unlist(percs),nrow=6,ncol=length(percs),byrow=F)

barplot(m,beside=TRUE,main=paste(i," archetypoids (from nearest)",sep = ""),
        ylim=c(0,100),ylab="Percentile")

## End(Not run)
```

---

cube34 *Cube of 34 landmarks*

Description

This is a cube made up of 34 landmarks, used as controlled data in the simulation study carried out in the paper referred below.

Usage

```
cube34
```

Format

An array with one matrix of 34 rows and 3 columns.

Source

Software Rhinoceros.

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

cube8 *Cube of 8 landmarks*

Description

This is a cube made up of 8 landmarks, used as controlled data in the simulation study carried out in the paper referred below.

Usage

```
cube8
```

Format

An array with one matrix of 8 rows and 3 columns.

Source

Software Rhinoceros.

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

dataDemo *Demo database of the Spanish anthropometric survey*

Description

This a demo database for academic and training purposes. It is oriented to exemplify the use of `trimowa`, `hipamAnthropom` and `TDDclust`.

It is made up of 600 women selected randomly from the Spanish anthropometric survey and five anthropometric variables: chest circumference, neck to ground length, waist circumference, hip circumference and bust circumference. These variables have been chosen following the recommendations of experts. In addition, they are commonly used in the literature about sizing system design and they appear in the European standard to sizing system.

Usage

```
dataDemo
```

Format

A matrix with 600 rows and 5 columns. Each row corresponds to an observation, and each column corresponds to a variable.

Source

Anthropometric survey of the Spanish female population.

References

Alemany, S., Gonzalez, J. C., Nacher, B., Soriano, C., Arnaiz, C., and Heras, H., (2010). Anthropometric survey of the Spanish female population aimed at the apparel industry. *Proceedings of the 2010 Intl. Conference on 3D Body scanning Technologies*, 307–315.

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Vinue, G., and Ibanez, M. V., (2014). *Data depth and Biclustering applied to anthropometric data. Exploring their utility in apparel design*. In progress.

European Committee for Standardization. Size designation of clothes. Part 2: Primary and secondary dimensions. (2002).

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

`trimowa`, `hipamAnthropom`, `TDDclust`

`dataUSAF` *USAF 1967 database*

Description

This data set comes from the 1967 United States Air Force (USAF) survey (available from `http://www.dtic.mil/dtic/`). The 1967 USAF survey was conducted during the first three months of 1967 under the direction of the Anthropology Branch of the Aerospace Medical Research Laboratory, located in Ohio. Subjects were measured at 17 Air Force bases across the United States of America. A total of 202 variables (including body dimensions and background variables) were taken on 2420 Air Force personnel between 21 and 50 years of age.

Usage

```
dataUSAF
```

Format

A matrix with 2420 rows and 202 columns. Each row corresponds to an observation, and each column corresponds to a variable.

Source

1967 United States Air Force (USAF) survey.

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. Submitted for publication.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

`figures8landm` *Figures with labelled landmarks*

Description

This function allows us to represent the two geometric figures (a cube and a parallelepiped) of 8 landmarks, with the landmark labels. Both appear in the submitted paper Vinue et al. (2013), referred below.

Usage

```
figures8landm(figure,data)
```

Arguments

`figure`	A character, two values are admitted: if figure="cube", the cube is represented. If figure="paral", the parallelepiped is represented.
`data`	The data with the landmarks of the corresponding figure.

Value

A plot of the cube or the parallelepiped with the landmark labels.

Author(s)

Guillermo Vinue

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Examples

```
## Not run:
figures8landm("cube",cube8)
figures8landm("paral",parallelepiped8)

## End(Not run)
```

`getBestPamsamIMO` *Generation of the candidate clustering partition in $HIPAM_IMO$*

Description

The HIPAM algorithm starts with one large cluster and, at each level, a given (parent) cluster is partitioned using PAM.

In this version of HIPAM, called $HIPAM_IMO$, the number k of (child) clusters is obtained by using the INCA (Index Number Clusters Atypical) criterion (Irigoien et al. (2008)) in the following way: at each node P, if there is k such that $INCA_k > 0.2$, then the k prior to the first largest slope decrease is selected. However, this procedure does not apply either to the top node or to the generation of the new partitions from which the Mean Split Silhouette is calculated. In these cases, even when all $INCA_k < 0.2$, k = 3 is fixed as the number of groups to divide and proceed. See Vinue et al. (2013) for more details.

The foundation and performance of the HIPAM algorithm is explained in `hipamAnthropom`.

Usage

```
getBestPamsamIMO(x,maxsplit,orness=0.7,type,...)
```

Arguments

`x`	Data to be clustered.
`maxsplit`	The maximum number of clusters that any cluster can be divided when searching for the best clustering.
`orness`	Quantity to measure the degree to which the aggregation is like a min or max operation. See `WeightsMixtureUB` and `GetDistMatrix`.
`type`	Option 'IMO' for using $HIPAM_IMO$.
`...`	Other arguments that may be supplied.

Value

A list with the following elements:

medoids: The cluster medoids.

clustering: The clustering partition obtained.

asw: The asw of the clustering.

num.of.clusters: Number of clusters in the final clustering.

info: List that informs about the progress of the clustering algorithm.

profiles: List that contains the asw and sesw (stardard error of the silhouette widths) profiles at each stage of the search.

metric: Dissimilarity used (called 'McCulloch' because the dissimilarity function used is that explained in McCulloch et al. (1998)).

Note

This function belongs to the $HIPAM_IMO$ algorithm and it is not solely used. That is why there is no section of *examples* in this help page. See hipamAnthropom.

Author(s)

This function was originally created by E. Wit et al., and it is available freely on http://www.math.rug.nl/~ernst/book/smida.html. We have adapted it to incorporate the INCA criterion.

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

Pollard, K. S., and van der Laan, M. J., (2002). A method to identify significant clusters in gene expression data. *Vol. II of SCI2002 Proceedings*, 318–325.

Irigoien, I., and Arenas, C., (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units, *Statistics in Medicine* **27**, 2948–2973.

Irigoien, I., Sierra, B., and Arenas, C., (2012). ICGE: an R package for detecting relevant clusters and atypical units in gene expression, *BMC Bioinformatics* **13** 1–29.

McCulloch, C., Paal, B., and Ashdown, S., (1998). An optimization approach to apparel sizing, *Journal of the Operational Research Society* **49**, 492–499.

See Also

hipamAnthropom

`getBestPamsamMO` *Generation of the candidate clustering partition in $HIPAM_MO$*

Description

The HIPAM algorithm starts with one large cluster and, at each level, a given (parent) cluster is partitioned using PAM.

In this version of HIPAM, called $HIPAM_MO$, the number k of (child) clusters is obtained by maximizing the silhouette width (asw). See Vinue et al. (2013) for more details.

The foundation and performance of the HIPAM algorithm is explained in `hipamAnthropom`.

Usage

```
getBestPamsamMO(x,maxsplit,orness=0.7,type,...)
```

Arguments

`x`	Data to be clustered.
`maxsplit`	The maximum number of clusters that any cluster can be divided when searching for the best clustering.
`orness`	Quantity to measure the degree to which the aggregation is like a min or max operation. See `WeightsMixtureUB` and `GetDistMatrix`.
`type`	Option 'MO' for using $HIPAM_MO$.
`...`	Other arguments that may be supplied.

Value

A list with the following elements:

medoids: The cluster medoids.

clustering: The clustering partition obtained.

asw: The asw of the clustering.

num.of.clusters: Number of clusters in the final clustering.

info: List that informs about the progress of the clustering algorithm.

profiles: List that contains the asw and sesw (stardard error of the silhouette widths) profiles at each stage of the search.

metric: Dissimilarity used (called 'McCulloch' because the dissimilarity function used is that explained in McCulloch et al. (1998)).

Note

This function belongs to the $HIPAM_MO$ algorithm and it is not solely used. That is why there is no section of *examples* in this help page. See `hipamAnthropom`.

Author(s)

This function was originally created by E. Wit et al., and it is available freely on http://www.math.rug.nl/~ernst/book/smida.html.

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

Pollard, K. S., and van der Laan, M. J., (2002). A method to identify significant clusters in gene expression data. *Vol. II of SCI2002 Proceedings*, 318–325.

McCulloch, C., Paal, B., and Ashdown, S., (1998). An optimization approach to apparel sizing, *Journal of the Operational Research Society* **49**, 492–499.

See Also

hipamAnthropom

`GetDistMatrix` *Dissimilarity matrix between individuals and prototypes*

Description

In the definition of a sizing system, a distance function allows us to represent mathematically the idea of garment fit and it is a key element to quantify the misfit between an individual and the prototype.

This function computes the dissimilarity defined in McCulloch et al. (1998), which is used in trimowa and hipamAnthropom. For more details, see also Ibanez et al. (2012).

Usage

```
GetDistMatrix(x,np,nv,w,bl,bh,al,ah,progress)
```

Arguments

x	Data vector.
np	Number of observations in the database.
nv	Number of variables in the database.
w	Weights for the OWA operator computed by means of WeightsMixtureUB.
bl,bh,al,ah	Constants required to specify the distance function.
progress	Boolean variable (TRUE or FALSE) to indicate whether the progress inform must be displayed on the screen.

Details

At the computational level, it is asummed that all the bh values are negative, all the bl values are positive and all the al and ah slopes are positive (the sign of al is changed within the function when computing the dissimilarities).

Value

A symmetric np x np matrix of dissimilarities.

Note

This function requires a C code called cast.c. In order to use GetDistMatrix outside the package, the dynamic-link library is called by means of the sentence dyn.load("cast.so") (In Windows, it would be dyn.load("cast.dll")).

Author(s)

Juan Domingo

References

McCulloch, C., Paal, B., and Ashdown, S., (1998). An optimization approach to apparel sizing, *Journal of the Operational Research Society* **49**, 492–499.

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Leon, T., Zuccarello, P., Ayala, G., de Ves, E., and Domingo, J., (2007), Applying logistic regression to relevance feedback in image retrieval systems, *Pattern Recognition* **40**, 2621–2632.

See Also

trimowa, hipamAnthropom

Examples

```
## Not run:
#Data loading:
dataDef <- dataDemo
bust <- dataDef$bust
#First bust class:
data <- dataDef[(bust >= 74) & (bust < 78), ]
num.variables <- dim(data)[2]

#Weights calculation:
orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)
```

```
#Constants required to specify the distance function:
K <- 3
bh <- (apply(as.matrix(log(data)),2,range)[2,]
       - apply(as.matrix(log(data)),2,range)[1,]) / ((K-1) * 8)
bl <- -3 * bh
ah <- c(23,28,20,25,25)
al <- 3 * ah

#Data processing.
num.persons <- dim(data)[1]
num.variables <- dim(data)[2]
datam <- as.matrix(data)
datat <- aperm(datam, c(2,1))
dim(datat) <- c(1,num.persons*num.variables)

#Dissimilarity matrix:
D <- GetDistMatrix(datat,num.persons,num.variables,w,bl,bh,al,ah,TRUE)


## End(Not run)
```

`HartiganShapes` *Hartigan-Wong k-means for 3D shapes*

Description

The basic foundation of k-means is that the sample mean is the value that minimizes the Euclidean distance from each point, to the centroid of the cluster to which it belongs. Two fundamental concepts of the statistical shape analysis are the Procrustes mean and the Procrustes distance. Therefore, by integrating the Procrustes mean and the Procrustes distance we can use k-means in the shape analysis context.

The k-means method has been proposed by several scientists in different forms. In computer science and pattern recognition the k-means algorithm is often termed the Lloyd algorithm (see Lloyd (1982)). However, in many texts, the term k-means algorithm is used for certain similar sequential clustering algorithms. Hartigan and Wong (1979) use the term k-means for an algorithm that searches for the locally optimal k-partition by moving points from one cluster to another.

This function allows us to use the Hartigan-Wong version of k-means adapted to deal with 3D shapes.

Usage

```
HartiganShapes(dg,K,Nsteps=10,niter=10,stopCr=0.0001,simul,initLl,initials,print)
```

Arguments

`dg`	Array with the 3D landmarks of the sample objects. Each row corresponds to an observation, and each column corresponds to a dimension (x,y,z).
`K`	Number of clusters.

`Nsteps`	Number of steps per initialization. Default value is 10.
`niter`	Number of random initializations. Default value is 10.
`stopCr`	Relative stopping criteria. Default value is 0.0001.
`simul`	A logical value. If TRUE, this function is used for a simulation study.
`initLl`	Logical value. If TRUE, see next argument `initials`. If FALSE, they are new random initial values.
`initials`	If `initLl=TRUE`, they are the same random initial values used in each iteration of `LloydShapes`. If `initLl=FALSE` this argument must be passed simply as an empty vector.
`print`	Logical value. If TRUE, some messages associated with the running process are displayed.

Details

There have been several attempts to adapt the k-means algorithm in the context of the statistical shape analysis, each one adapting a different version of the k-means algorithm (Amaral et al. (2010), Georgescu (2009)). In Vinue, G. et al. (2014) (submitted), it is demonstrated that the Lloyd k-means represents a noticeable reduction in the computation involved when the sample size increases, compared with the Hartigan-Wong k-means. We state that Hartigan-Wong should be used in the shape analysis context only for very small samples.

Value

A list with the following elements:

ic1: Optimal clustering.

copt: Optimal centers.

vopt: Optimal objective function.

If a simulation study is carried out, the following elements are returned:

compTime: Computational time.

AllRate: Allocation rate.

Note

This function is based on the kmns.m file available from `http://people.sc.fsu.edu/~jburkardt/m_src/asa136/asa136.html`

Author(s)

Guillermo Vinue

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Hartigan, J. A., and Wong, M. A., (1979). A K-Means Clustering Algorithm, *Applied Statistics*, 100–108.

Lloyd, S. P., (1982). Least Squares Quantization in PCM, *IEEE Transactions on Information Theory* **28**, 129–137.

Amaral, G. J. A., Dore, L. H., Lessa, R. P., and Stosic, B., (2010). k-Means Algorithm in Statistical Shape Analysis, *Communications in Statistics - Simulation and Computation* **39(5)**, 1016–1026.

Georgescu, V., (2009). Clustering of Fuzzy Shapes by Integrating Procrustean Metrics and Full Mean Shape Estimation into K-Means Algorithm. *In IFSA-EUSFLAT Conference.*

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

See Also

LloydShapes, trimmedLloydShapes, landmarks, cube8, parallelepiped8, cube34, parallelepiped34, procGPA, optraProcrustes, qtranProcrustes

Examples

```
## Not run:
#CLUSTERING INDIVIDUALS ACCORDING TO THEIR SHAPE:
landmarks1 <- na.exclude(landmarks)
dim(landmarks1)
#[1] 574 198
(num.points <- (dim(landmarks1)[2]) / 3)
#[1] 66
landmarks2 <- landmarks1[1:50,] #In the interests of simplicity of the computation involved.
(n <- dim(landmarks2)[1])
#[1] 50
colnames(landmarks2)[seq(1,198,3)]

dg <- array(0,dim = c(num.points,3,n))
for(k in 1:n){
 for(l in 1:3){
  dg[,l,k] <-  as.matrix(as.vector(landmarks2[k,][seq(l,
                          dim(landmarks2)[2]+(l-1),by=3)]),ncol=1,byrow=T)
 }
}
shapes::plotshapes(dg[,,1])
calibrate::textxy(dg[,1,1],dg[,2,1],labs=1:num.points,cex=0.7)

K <- 3 ; Nsteps <- 5 ; niter <- 3 ; stopCr <- 0.0001
resHA <- HartiganShapes(dg,K,Nsteps,niter,stopCr,FALSE,FALSE,c(),TRUE)

#Numerical and graphical results:
asig <- resHA$ic1  #table(asig) shows the clustering results.
copt <- resHA$copt #optimal centers.

#Generalised Procrustes analysis into each cluster:
out_proc <- list()
for(h in 1 : K){
 out_proc[[h]] = shapes::procGPA(dg[, , asig == h], distances = T, pcaoutput = T)
}

data <- dataDemo[1:50,]
```

```
boxplot(data$necktoground ~ as.factor(asig), main = "Neck to ground")

shapes::plotshapes(out_proc[[1]]$rotated)
points(copt[,,1], col = 2)
legend("topleft", c("Rotated data", "Mean shape"), pch = 1, col = 1:2, text.col = 1:2)
title("Procrustes rotated data for cluster 1 \n with its mean shape superimposed", sub = "Plane xy")



#SIMULATION STUDY:
#Definition of the cluster of cubes:
Ms_cube <- cube8
#Ms_cube <- cube34 #for the case of 34 landmarks.
colMeans(Ms_cube)
dim(Ms_cube)
shapes::plotshapes(Ms_cube[,,1])

#Number of landmarks and variables:
k_cube <- dim(Ms_cube)[1]
vars_cube <- k_cube * dim(Ms_cube)[2]

#Covariance matrix (0.01, 9, 36):
sigma_cube <- 0.01
Sigma_cube <- diag(sigma_cube,vars_cube)

#Sample size of each cluster (25, 250, 450):
n_cube <- 25

#Cluster of cubes:
simu1_cube <- mvtnorm::rmvt(n_cube,Sigma_cube,df=99)[,c(1 : k_cube * dim(Ms_cube)[2]
        - 2, 1 : k_cube * dim(Ms_cube)[2] - 1, 1 : k_cube * dim(Ms_cube)[2])]
Simu1_cube <- as.vector(Ms_cube) + t(simu1_cube)
dim(Simu1_cube)

#Labels vector to identify the elements in the cluster of cubes:
etiqs_cl1 <- paste("cube_", 1:n_cube, sep = "")

#First cluster:
cl1 <- array(Simu1_cube, dim = c(k_cube, dim(Ms_cube)[2], n_cube),
             dimnames = list(NULL, NULL, etiqs_cl1))
colMeans(cl1)
dim(cl1)

#Definition of the cluster of parallelepipeds:
Ms_paral <- parallelepiped8
#Ms_paral <- parallelepiped34 #for the case of 34 landmarks.
colMeans(Ms_paral)
dim(Ms_paral)

#Number of landmarks and variables:
k_paral <- dim(Ms_paral)[1]
vars_paral <- k_paral * dim(Ms_paral)[2]
```

```
#Covariance matrix (0.01, 9, 36):
sigma_paral <- 0.01
Sigma_paral <- diag(sigma_paral,vars_paral)

#Sample size of each cluster (25, 250, 450):
n_paral <- 25

#Cluster of parallelepipeds:
simu1_paral <- mvtnorm::rmvt(n_paral, Sigma_paral, df = 99)[,c(1 : k_paral *
                   dim(Ms_paral)[2] - 2, 1 : k_paral * dim(Ms_paral)[2] -
                   1, 1 : k_paral * dim(Ms_paral)[2])]
Simu1_paral <- as.vector(Ms_paral) + t(simu1_paral)
dim(Simu1_paral)

#Labels vector to identify the elements in the cluster of parallelepipeds:
etiqs_cl2 <- paste("Parallelepiped_", 1:n_paral, sep = "")

#Second cluster:
cl2 <- array(Simu1_paral, dim = c(k_paral, dim(Ms_paral)[2], n_paral),
             dimnames = list(NULL, NULL, etiqs_cl2))
colMeans(cl2)
dim(cl2)

#Combine both clusters:
dg <- abind::abind(cl1,cl2)
str(dg)
shapes3dMod(dg[,,1], loop = 0, type = "p", color = 2, joinline = c(1:1),
            axes3 = TRUE, rglopen = TRUE, main = "First figure")

#First, the Lloyd algorithm is executed and then the Hartigan algorithm with
#the same initial values used by the Lloy algorithm is executed:
K <- 2 ; Nsteps <- 5 ; niter <- 3 ; stopCr <- 0.0001
resLLSim <- LloydShapes(dg,K,Nsteps,niter,stopCr,TRUE,TRUE)
resHASim <- HartiganShapes(dg,K,Nsteps,niter,stopCr,TRUE,TRUE,resLLSim$initials,TRUE)

## End(Not run)
```

---

hipamAnthropom *HIPAM algorithm for anthropometric data*

Description

The HIerarchical Partitioning Around Medoids clustering method (HIPAM) was originally created to gene clustering (Wit et al. (2004)). The HIPAM algorithm is a divisive hierarchical clustering method based on the PAM algorithm.

This function is a HIPAM algorithm adapted to deal with anthropometric data. To that end, a different dissimilarity function is incorporated. This function is that explained in McCulloch et al. (1998) and it is implemented in GetDistMatrix. We call it d_MO. In addition, a different method to obtain a classification tree is also incorporated.

Two HIPAM algorithms are proposed. The first one, called $HIPAM_MO$, is a HIPAM that uses d_MO. The second one, $HIPAM_IMO$, is a HIPAM algorithm that uses d_MO and the INCA (Index Number Clusters Atypical) statistic criterion (Irigoien et al. (2008)) to decide the number of child clusters and as a stopping rule.

See Vinue et al. (2013) for more details.

Usage

```
hipamAnthropom(x,asw.tol=0,maxsplit=5,local.const=NULL,orness=0.7,type,
               ahVect=c(23,28,20,25,25),...)
```

Arguments

x	Data frame. In our approach, this is each one of the subframes originated after segmenting the whole anthropometric Spanish survey in twelve bust segments, according to the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals. Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
asw.tol	If this value is given, a tolerance or penalty can be introduced (asw.tol > 0 or asw.tol < 0, respectively) in the branch splitting procedure. Default value (0) is maintained. See page 154 of Wit et al. (2004) for more details.
maxsplit	The maximum number of clusters that any cluster can be divided when searching for the best clustering.
local.const	If this value is given (meaningful values are those between -1 and 1), a proposed partition is accepted only if the associated asw is greater than this constant. Default option for this argument is maintained, that is to say, this value is ignored. See page 154 of Wit et al. (2004) for more details.
orness	Quantity to measure the degree to which the aggregation is like a min or max operation. See WeightsMixtureUB and GetDistMatrix.
type	Type of HIPAM algorithm to be used. The possible options are 'MO' (for $HIPAM_MO$) and 'IMO' (for $HIPAM_IMO$).
ahVect	Constants that define the ah slopes of the distance function in GetDistMatrix. Given the five variables considered, this vector is c(23,28,20,25,25). This vector would be other according to the variables considered.
...	Other arguments that may be supplied to the internal functions of the HIPAM algorithms.

Details

The $HIPAM_MO$ algorithm uses the getBestPamsamMO and checkBranchLocalMO functions, while the $HIPAM_IMO$ algorithm uses the getBestPamsamIMO and checkBranchLocalIMO functions.

For more details of HIPAM, see van der Laan et al. (2003), Wit et al. (2004) and the manual of the **smida** R package.

Value

A list with the following elements:

clustering: Final clustering that corresponds to the last level of the tree.

asw: The asw of the final clustering.

n.levels: Number of levels in the tree.

medoids: Medoids of all of the clusters in the tree.

active: Activity status of each cluster (FALSE for every cluster of the final partition).

development: Matrix that indicates the ancestors of the final clusters.

num.of.clusters: Number of clusters in the final clustering.

metric: Dissimilarity used (called 'McCulloch' because the dissimilarity function used is that explained in McCulloch et al. (1998)).

Note

All the functions related to the HIPAM algorithm were originally created by E. Wit et al., and they are available freely on http://www.math.rug.nl/~ernst/book/smida.html. In order to develop the $HIPAM_MO$ and $HIPAM_IMO$ algorithms, we have used and adapted them.

Author(s)

Guillermo Vinue

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

van der Laan, M. J., and Pollard, K. S., (2003). A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap, *Journal of Statistical Planning and Inference* **117**, 275–303.

Pollard, K. S., and van der Laan, M. J., (2002). A method to identify significant clusters in gene expression data. *Vol. II of SCI2002 Proceedings*, 318–325.

Irigoien, I., and Arenas, C., (2008). INCA: New statistic for estimating the number of clusters and identifying atypical units, *Statistics in Medicine* **27**, 2948–2973.

Irigoien, I., Sierra, B., and Arenas, C., (2012). ICGE: an R package for detecting relevant clusters and atypical units in gene expression, *BMC Bioinformatics* **13**, 1–29.

McCulloch, C., Paal, B., and Ashdown, S., (1998). An optimization approach to apparel sizing, *Journal of the Operational Research Society* **49**, 492–499.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

Alemany, S., Gonzalez, J. C., Nacher, B., Soriano, C., Arnaiz, C., and Heras, H., (2010). Anthropometric survey of the Spanish female population aimed at the apparel industry. *Proceedings of the 2010 Intl. Conference on 3D Body scanning Technologies*, 307–315.

See Also

getBestPamsamMO, getBestPamsamIMO, checkBranchLocalMO, checkBranchLocalIMO, plotTreeHipam, hipamBigGroups, outlierHipam

Examples

```
## Not run:
dataDef <- dataDemo
bust <- dataDef$bust

bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7 ; type <- "IMO" #type <- "MO" for $HIPAM_{MO}$

ahVect <- c(23, 28, 20, 25, 25)

hip <- list()
for(i in 1 : (nsizes - 1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type=type,ahVect=ahVect)
}
str(hip)

ress <- list()
for(i in 1 : length(hip)){
  ress[[i]] <- table(hip[[i]]$clustering)
}
ress #clustering results in each bust size.

## End(Not run)
```

---

hipamBigGroups *Hipam medoids of the clusters with more than 2 elements*

Description

This function obtains the medoids of the clusters provided by hipamAnthropom that contains more than two individuals.

Usage

```
hipamBigGroups(i,hip)
```

Arguments

i	Each bust group considered according to the European standard to sizing system.
hip	Hipam object, see hipamAnthropom.

Value

The medoids of the clusters with more than two elements for each bust class.

Author(s)

Guillermo Vinue

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

hipamAnthropom

Examples

```
## Not run:
dataDef <- dataDemo
bust <- dataDef$bust

bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7

ahVect <- c(23,28,20,25,25)

hip <- list()
for(i in 1 : (nsizes-1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type="MO",ahVect=ahVect)
}
#Medoids of the clusters with more than two elements for the second bust class:
hipamBigGroups(2,hip)
#Medoids of the clusters with more than two elements for all the bust classes:
list.meds <- lapply(1:(nsizes-1),FUN=hipamBigGroups,hip)
```

```
## End(Not run)
```

`indivNearest` *Nearest individuals to archetypes*

Description

The nearest individual to each archetype can be obtained by simply computing the distance between the archetypes and the individuals and choosing the nearest. This is the procedure to obtain what is called the *nearest* vector, see Vinue et al. (2013). It is used within `archetypoids` and `stepArchetypoids`.

Usage

```
indivNearest(Indivs,NumArchet,mdras)
```

Arguments

`Indivs`	Vector from 1 to `NumArchet`.
`NumArchet`	Number of archetypes computed.
`mdras`	Distance matrix between the archetypes and the individuals.

Value

A vector with the nearest individuals to archetypes.

Author(s)

Irene Epifanio

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. *Submitted for publication*.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

See Also

`archetypoids`, `stepArchetypoids`, `archetypesBoundary`

Examples

```
## Not run:
#First,the USAF 1967 database is read and preprocessed (Zehner et al. (1993)).
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#Procedure and results shown in section 2.2.2 and section 3.1 of Epifanio et al. (2013):
res <- archetypesBoundary(preproc$data,15,FALSE,3)

i=3
a3 <- archetypes::bestModel(res[[i]])
ras <- rbind(archetypes::parameters(a3),preproc$data)
dras <- dist(ras,method="euclidean",diag=F,upper=T,p=2)
mdras <- as.matrix(dras)
diag(mdras) <- 1e+11
sapply(1:i,indivNearest,i,mdras)

## End(Not run)
```

---

landmarks *Landmarks representing the woman's body*

Description

The body shape of the women who belong to dataDemo is represented by a set of anatomical correspondence points, called landmarks.

This database collects the set of landmarks of each woman.

The landmarks considered were placed in three different ways:

- Automatic landmarks: automatically calculated with scanner program algorithms, based on geometrical features of the body.
- Manual landmarks: points which are not reflected on the external body geometry; they were located through palpation by expert personnel and identified by a physical marker.
- Digital landmarks: detected on the computer screen in the 3D scanned image. They are not robust on the automatic calculation but are easy to detect on the screen.

Usage

```
landmarks
```

Format

A data frame with 600 observations and 198 variables (66 landmarks times 3 dimensions).

Source

Anthropometric survey of the Spanish female population.

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Alemany, S., Gonzalez, J. C., Nacher, B., Soriano, C., Arnaiz, C., and Heras, H., (2010). Anthropometric survey of the Spanish female population aimed at the apparel industry. *Proceedings of the 2010 Intl. Conference on 3D Body scanning Technologies*, 307–315.

`LloydShapes` *Lloyd k-means for 3D shapes*

Description

The basic foundation of k-means is that the sample mean is the value that minimizes the Euclidean distance from each point, to the centroid of the cluster to which it belongs. Two fundamental concepts of the statistical shape analysis are the Procrustes mean and the Procrustes distance. Therefore, by integrating the Procrustes mean and the Procrustes distance we can use k-means in the shape analysis context.

The k-means method has been proposed by several scientists in different forms. In computer science and pattern recognition the k-means algorithm is often termed the Lloyd algorithm (see Lloyd (1982)).

This function allows us to use the Lloyd version of k-means adapted to deal with 3D shapes.

Usage

```
LloydShapes(dg,K,Nsteps=10,niter=10,stopCr=0.0001,simul,print)
```

Arguments

`dg`	Array with the 3D landmarks of the sample objects. Each row corresponds to an observation, and each column corresponds to a dimension (x,y,z).
`K`	Number of clusters.
`Nsteps`	Number of steps per initialization. Default value is 10.
`niter`	Number of random initializations. Default value is 10.
`stopCr`	Relative stopping criteria. Default value is 0.0001.
`simul`	A logical value. If TRUE, this function is used for a simulation study.
`print`	Logical value. If TRUE, some messages associated with the running process are displayed.

Details

There have been several attempts to adapt the k-means algorithm in the context of the statistical shape analysis, each one adapting a different version of the k-means algorithm (Amaral et al. (2010), Georgescu (2009)). In Vinue et al. (2014) (submitted), it is demonstrated that the Lloyd k-means represents a noticeable reduction in the computation involved when the sample size increases, compared with the Hartigan-Wong k-means. We state that Hartigan-Wong should be used in the shape analysis context only for very small samples.

Value

A list with the following elements:

asig: Optimal clustering.

copt: Optimal centers.

vopt: Optimal objective function.

initials: Random initial values used in each iteration. These values are then used by HartiganShapes.

If a simulation study is carried out, the following elements are returned:

compTime: Computational time.

AllRate: Allocation rate.

initials: Random initial values used in each iteration. These values are then used by HartiganShapes.

Author(s)

Amelia Simo

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Lloyd, S. P., (1982). Least Squares Quantization in PCM, *IEEE Transactions on Information Theory* **28**, 129–137.

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

See Also

HartiganShapes, trimmedLloydShapes, landmarks, cube8, parallelepiped8, cube34, parallelepiped34, procGPA

Examples

```
## Not run:
#CLUSTERING INDIVIDUALS ACCORDING TO THEIR SHAPE:
landmarks1 <- na.exclude(landmarks)
dim(landmarks1)
#[1] 574 198
(num.points <- (dim(landmarks1)[2]) / 3)
#[1] 66
```

```
landmarks2 <- landmarks1[1:50,] #In the interests of simplicity of the computation involved.
(n <- dim(landmarks2)[1])
#[1] 50

dg <- array(0,dim = c(num.points,3,n))
for(k in 1:n){
 for(l in 1:3){
  dg[,l,k] <-  as.matrix(as.vector(landmarks2[k,][seq(l,dim(landmarks2)[2]+
                          (l-1),by=3)]),ncol=1,byrow=T)
 }
}
shapes::plotshapes(dg[,,1])
calibrate::textxy(dg[,1,1],dg[,2,1],labs=1:num.points,cex=0.7)

K <- 3 ; Nsteps <- 5 ; niter <- 5 ; stopCr <- 0.0001
resLL <- LloydShapes(dg,K,Nsteps,niter,stopCr,FALSE,TRUE)

#Numerical and graphical results:
asig <- resLL$asig #table(asig) shows the clustering results.
copt <- resLL$copt #optimal centers.

#Generalised Procrustes analysis into each cluster:
out_proc <- list()
for(h in 1 : K){
 out_proc[[h]] = shapes::procGPA(dg[, , asig == h], distances = T, pcaoutput = T)
}

data <- dataDemo[1:50,]
boxplot(data$necktoground ~ as.factor(asig), main = "Neck to ground")

shapes::plotshapes(out_proc[[1]]$rotated)
points(copt[,,1], col = 2)
legend("topleft", c("Rotated data", "Mean shape"), pch = 1, col = 1:2, text.col = 1:2)
title("Procrustes rotated data for cluster 1 \n with its mean shape superimposed", sub = "Plane xy")



#SIMULATION STUDY:
#Definition of the cluster of cubes:
Ms_cube <- cube8
#Ms_cube <- cube34 #for the case of 34 landmarks.
colMeans(Ms_cube)
dim(Ms_cube)
shapes::plotshapes(Ms_cube[,,1])

#Number of landmarks and variables:
k_cube <- dim(Ms_cube)[1]
vars_cube <- k_cube * dim(Ms_cube)[2]

#Covariance matrix (0.01, 9, 36):
sigma_cube <- 0.01
Sigma_cube <- diag(sigma_cube,vars_cube)
```

```
#Sample size of each cluster (25, 250, 450):
n_cube <- 25

#Cluster of cubes:
simu1_cube <- mvtnorm::rmvt(n_cube,Sigma_cube,df=99)[,c(1 : k_cube * dim(Ms_cube)[2]
        - 2, 1 : k_cube * dim(Ms_cube)[2] - 1, 1 : k_cube * dim(Ms_cube)[2])]
Simu1_cube <- as.vector(Ms_cube) + t(simu1_cube)
dim(Simu1_cube)

#Labels vector to identify the elements in the cluster of cubes:
etiqs_cl1 <- paste("cube_", 1:n_cube, sep = "")

#First cluster:
cl1 <- array(Simu1_cube, dim = c(k_cube, dim(Ms_cube)[2], n_cube),
             dimnames = list(NULL, NULL, etiqs_cl1))
colMeans(cl1)
dim(cl1)

#Definition of the cluster of parallelepipeds:
Ms_paral <- parallelepiped8
#Ms_paral <- parallelepiped34 #for the case of 34 landmarks.
colMeans(Ms_paral)
dim(Ms_paral)

#Number of landmarks and variables:
k_paral <- dim(Ms_paral)[1]
vars_paral <- k_paral * dim(Ms_paral)[2]

#Covariance matrix (0.01, 9, 36):
sigma_paral <- 0.01
Sigma_paral <- diag(sigma_paral,vars_paral)

#Sample size of each cluster (25, 250, 450):
n_paral <- 25

#Cluster of parallelepipeds:
simu1_paral <- mvtnorm::rmvt(n_paral, Sigma_paral, df = 99)[,c(1 : k_paral *
           dim(Ms_paral)[2] - 2, 1 : k_paral * dim(Ms_paral)[2] - 1,
           1 : k_paral * dim(Ms_paral)[2])]
Simu1_paral <- as.vector(Ms_paral) + t(simu1_paral)
dim(Simu1_paral)

#Labels vector to identify the elements in the cluster of parallelepipeds:
etiqs_cl2 <- paste("Parallelepiped_", 1:n_paral, sep = "")

#Second cluster:
cl2 <- array(Simu1_paral, dim = c(k_paral, dim(Ms_paral)[2], n_paral),
             dimnames = list(NULL, NULL, etiqs_cl2))
colMeans(cl2)
dim(cl2)

#Combine both clusters:
dg <- abind::abind(cl1,cl2)
```

```
str(dg)
shapes3dMod(dg[,,1], loop=0, type="p", color=2, joinline=c(1:1),
            axes3=TRUE, rglopen=TRUE, main="First figure")

K <- 2 ; Nsteps <- 5 ; niter <- 5 ; stopCr <- 0.0001
resLLSim <- LloydShapes(dg,K,Nsteps,niter,stopCr,TRUE,TRUE)

## End(Not run)
```

`optraProcrustes` *Auxiliary optra subroutine of the Hartigan-Wong k-means for 3D shapes*

Description

The Hartigan-Wong version of the k-means algorithm uses two auxiliary algorithms: the optimal transfer stage (optra) and the quick transfer stage (qtran).

This function is the optra subroutine adapted to the shape analysis context. It is used within `HartiganShapes`. See Hartigan and Wong (1979) for details of the original k-means algorithm and Amaral et al. (2010) for details about its adaptation to shape analysis.

Usage

```
optraProcrustes(dg,n,c,K,ic1,ic2,nc,an1,an2,ncp,d,itran,live,indx)
```

Arguments

`dg`	Array with the 3D landmarks of the sample objects.
`n`	Number of sample objects.
`c`	Array of centroids.
`K`	Number of clusters.
`ic1`	The cluster to each object belongs.
`ic2`	This vector is used to remember the cluster which each object is most likely to be transferred to at each step.
`nc`	Number of objects in each cluster.
`an1`	$an1(l) = nc(l) / (nc(l) - 1), l=1,...,K$.
`an2`	$an2(l) = nc(l) / (nc(l) + 1), l=1,...,K$.
`ncp`	In the optimal transfer stage, ncp(l) stores the step at which cluster l is last updated, $l=1,...,K$. In the quick transfer stage, ncp(l) stores the step at which cluster l is last updated plus n, $l=1,...,K$.
`d`	Vector of distances from each object to every centroid.
`itran`	itran(l) = 1 if cluster l is updated in the quick-transfer stage (0 otherwise), $l=1,...,K$.
`live`	Vector that indicates whether a cluster is included in the live set or not.
`indx`	Number of steps since a transfer took place.

Value

A list with the following elements: *c,ic1,ic2,nc,an1,an2,ncp,d,itran,live,indx*, updated after the optimal transfer stage.

Note

This function belongs to HartiganShapes and it is not solely used. That is why there is no section of *examples* in this help page.

Note

This function is based on the optra.m file available from http://people.sc.fsu.edu/~jburkardt/m_src/asa136/asa136.html.

Author(s)

Guillermo Vinue

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Hartigan, J. A., and Wong, M. A., (1979). A K-Means Clustering Algorithm, *Applied Statistics*, 100–108.

Amaral, G. J. A., Dore, L. H., Lessa, R. P., and Stosic, B., (2010). k-Means Algorithm in Statistical Shape Analysis, *Communications in Statistics - Simulation and Computation* **39(5)**, 1016–1026.

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

See Also

HartiganShapes

---

outlierHipam *Individuals of the hipam clusters with 1 or 2 elements*

Description

This function obtains the individuals of the clusters provided by hipamAnthropom that only contains one or two observations.

Given its hierarchical nature, the HIPAM algorithm derives clusters which only contain one or two observations. In the hierarchical clustering methods, all clusters with only one element (the so-called singleton clusters) are consider as outliers. Regarding clusters with two elements, this is because three is the minimum number of elements for clustering with PAM. Taking advantage of this fact, we can also consider the clusters with two elements as outliers.

Usage

```
outlierHipam(i,hip)
```

Arguments

i	Each bust group considered according to the European standard to sizing system.
hip	Hipam object, see hipamAnthropom.

Value

The observations of the clusters with one or two elements for each bust class.

Author(s)

Guillermo Vinue

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

hipamAnthropom

Examples

```
## Not run:
dataDef <- dataDemo
bust <- dataDef$bust

bustCirc_4 <- seq(74, 102, 4)
bustCirc_6 <- seq(107, 131, 6)
bustCirc <- c(bustCirc_4, bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7

ahVect <- c(23,28,20,25,25)

hip <- list()
for(i in 1 : (nsizes - 1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
```

```
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type="MO",ahVect=ahVect)
}

#Outliers for the second bust class:
outlierHipam(2,hip)
#Outliers for all the bust classes:
list_outl1_2 <- sapply(1:(nsizes-1),outlierHipam,hip)

## End(Not run)
```

`overlappingRows` *Overlapped biclusters by rows*

Description

This function allows us to check which rows belong to more than one bicluster. It is used within the `CCbiclustAnthropo` function.

Usage

```
overlappingRows(i,resBicluster)
```

Arguments

`i`	Bicluster number.
`resBicluster`	An object of class Biclust.

Details

In order to know how this function works, it is necessary to understand the following commands:

res.bicl@RowxNumber[,1] indicates the rows that belong to the bicluster 1, by assigning a TRUE value to the position of those rows inside the original matrix. By using *table(res.bicl@RowxNumber[,1])*, we obtain the number of rows belonging to bicluster 1.

*1 * res.bicl@RowxNumber[,1]* makes TRUES into 1s.

*i * res.bicl@RowxNumber[,i]* makes TRUES into the corresponding value of i.

In short, this function puts a 1 in those rows belonging to bicluster 1, a 2 in those ones of bicluster 2, and so on.

The fact that certain columns of the matrix returned by this function have a value different from 0 at the same row, will indicate that that row belong to both biclusters.

Value

A matrix with as many rows as rows of the original matrix, and as many columns as obtained biclusters.

Author(s)

Guillermo Vinue

References

Vinue, G., and Ibanez, M. V., (2014), *Data depth and Biclustering applied to anthropometric data. Exploring their utility in apparel design*. In progress.

Kaiser, S., and Leisch, F., (2008). A Toolbox for Bicluster Analysis in R. Tech.rep., Department of Statistics (University of Munich).

See Also

CCbiclustAnthropo

Examples

```
## Not run:
#Note: package biclust needed.
#This is an example of using this function with a certain database
#made up of body dimensions related to the lower body part.
data <- dataUser[(waist >= 58) & (waist < 115),] #dataUser is the user database.
rownames(data) <- 1:dim(data)[1]

waist <- data[,"WaistCircumference"]

waist_4 <- seq(58, 86, 4)
waist_6 <- seq(91, 115, 6)
waistCirc <- c(waist_4,waist_6)
nsizes <- length(waistCirc)

#Position of the body variables in the database:
lowerVars <- c(14, 17:25, 27, 28, 65:73, 75, 77:81, seq(100, 116, 2))

nBic <- c(2, 2, 4, rep(5, 7), 3, 3)
diffRanges <- list(c(14,20), c(24,30), c(24,30), c(33,39), c(29,35), c(29,35),
                   c(28,35), c(31,38), c(31,38), c(30,37), c(26,33), c(25,32))
percDisac <- 0.01
dir <- "/home/guillermo/"

res_bicl_antropom <- CCbiclustAnthropo(data,waist,waistCirc,lowerVars,
                                       nsizes,nBic,diffRanges,percDisac,dir)

#For a single size:
size <- 5
res <- res_bicl_antropom[[1]][[size]]

sapply(1 : res@Number, overlappingRows, res)

## End(Not run)
```

parallelepiped34 *Parallelepiped of 34 landmarks*

Description

This is a parallelepiped made up of 34 landmarks, used as controlled data in the simulation study carried out in the paper referred below.

Usage

```
parallelepiped34
```

Format

An array with one matrix of 34 rows and 3 columns.

Source

Software Rhinoceros.

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

parallelepiped8 *Parallelepiped of 8 landmarks*

Description

This is a parallelepiped made up of 8 landmarks, used as controlled data in the simulation study carried out in the paper referred below.

Usage

```
parallelepiped8
```

Format

An array with one matrix of 8 rows and 3 columns.

Source

Software Rhinoceros.

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

`plotMedoids` *Medoids representation*

Description

This function represents the scatter plots of bust circumference against other selected variable (chest, hip, neck to ground or waist) jointly with the medoids obtained for each bust class provided by either `trimowa` or `hipamAnthropom`. In addition, the prototypes defined by the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals can be also displayed.

Usage

```
plotMedoids(x,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,EN)
```

Arguments

`x`	Data frame. It should contain the chest, neck to ground, waist, hip and bust measurements of the individuals. In order to be able to represent them, the name of the columns of the database must be 'chest', 'necktoground', 'waist', 'hip' and 'bust' respectively, see `dataDemo`. Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
`medoids`	Medoids i.e., typical persons within the sample, obtained with `trimowa` or `hipamAnthropom`.
`nsizes`	Number of subsets (classes), into the database is segmented. In our approach, the whole anthropometric Spanish survey is segmented into twelve bust segments, according to the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals.
`bustVariable`	Bust variable.
`variable`	Anthropometric variable to be plotted. It can be 'chest', 'necktoground', 'waist' and 'hip'.
`color`	A specification for the medoids color in each bust class.
`xlim`	Axis lenght of the x axis according to the range of the bust variable.
`ylim`	Axis lenght of the y axis according to the range of the selected variable among chest, hip, neck to ground and waist.
`title`	Main title of the plot.
`EN`	A logical value. If TRUE, the prototypes defined by the European standard for each variable are represented. See section *Details* for more details.

Details

Ir order to check the goodness of trimowa, the sizes defined by the medoids can be compared with those defined by the European standard to sizing system. This standard establishes 12 sizes according to the combinations of the bust, waist and hip measurements and does not fix neither chest nor height standard measurements. We can approximate the chest measurements through a linear regression analysis, taking the bust measurements detailed in the standard as independent variable. Besides, we take as neck to ground measurements for the standard sizing system, the values 132, 136 and 140 cm because those are the most repeated values and they are those which best cover our data set. See Ibanez et al. (2012) for a complete explanation.

Value

A device with the desired plot.

Note

As mentioned, this function is especially defined for the sizes established by the European standard to sizing system. Part 3: Measurements and intervals. In order to use this function with other standard, this function must be adapted.

Author(s)

Guillermo Vinue

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

dataDemo, WeightsMixtureUB, trimowa, GetDistMatrix, trimmedoid, hipamAnthropom

Examples

```
## Not run:
#TRIMOWA ALGORITHM:
#Loading the data to apply the trimowa algorithm:
dataDef <- dataDemo
num.variables <- dim(dataDef)[2]
bust <- dataDef$bust

orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)
```

```
bustCirc_4 <- seq(74,102,4)
bustCirc_6 <- seq(107,131,6)
bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
K <- 3 ; alpha <- 0.01 ; niter <- 6 ; Ksteps <- 7

ahVect <- c(23,28,20,25,25)

res_trimowa <- list()
for (i in 1 : (nsizes-1)){
  data = dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  res_trimowa[[i]] <- trimowa(data,w,K,alpha,niter,Ksteps,ahVect=ahVect)
}

medoids <- list()
for (i in 1 : (nsizes-1)){
  medoids[[i]] <- res_trimowa[[i]]$meds #or res_trimowa[[i]][[1]]
}

bustVariable <- "bust"
xlim <- c(70,150)
color <- c("black","red","green", "blue","cyan","brown","gray","deeppink3",
           "orange","springgreen4","khaki3","steelblue1")

variable <- "chest"
range(dataDef[,variable])
#[1] 76.7755 135.8580
ylim <- c(70,140)
title <- "Medoids \n bust vs chest"

plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,FALSE)
plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,TRUE)

variable <- "hip"
range(dataDef[,variable])
#[1] 83.6 152.1
ylim <- c(80,160)
title <- "Medoids \n bust vs hip"

plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,FALSE)
plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,TRUE)

variable <- "necktoground"
range(dataDef[,variable])
#[1] 117.6 154.9
ylim = c(110,160)
title <- "Medoids \n bust vs neck to ground"

plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,FALSE)
plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,TRUE)

variable <- "waist"
```

```
range(dataDef[,variable])
#[1]  58.6 133.0
ylim <- c(50,140)
title <- "Medoids \n bust vs waist"

plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,FALSE)
plotMedoids(dataDef,medoids,nsizes,bustVariable,variable,color,xlim,ylim,title,TRUE)

#AN EXAMPLE FOR HIPAM ALGORITHM:
dataDef <- dataDemo
bust <- dataDef$bust

bustCirc_4 <- seq(74,102,4)
bustCirc_6 <- seq(107,131,6)
bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7 ; alpha <- 0.01 ; type <- "MO" #type <- "IMO" for $HIPAM_{IMO}$

ahVect <- c(23, 28, 20, 25, 25)

hip <- list()
for(i in 1 : (nsizes - 1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type=type,ahVect=ahVect)
}
str(hip)

ress <- list()
for(i in 1 : length(hip)){
  ress[[i]] <- table(hip[[i]]$clustering)
}
ress

list.meds <- lapply(1:(nsizes - 1), FUN = hipamBigGroups, hip)

bustVariable <- "bust"
xlim <- c(70,150)
color <- c("black","red","green","blue","cyan","brown","gray","deeppink3",
           "orange","springgreen4","khaki3","steelblue1")

variable <- "hip"
ylim <- c(80,160)
title <- "Medoids \n bust vs hip"

plotMedoids(dataDef,list.meds,nsizes,bustVariable,variable,color,xlim,ylim,title,FALSE)

## End(Not run)
```

---

plotTreeHipam *HIPAM dendogram*

Description

This function represents a dendrogram for the clustering results provided by a HIPAM algorithm. It is a small modification of the original `plot.tree` function of the **smida** R package, available from http://www.math.rug.nl/~ernst/book/smida.html.

Usage

```
plotTreeHipam(x,title,...)
```

Arguments

`x`	The HIPAM object to be plotted.
`title`	The title of the plot.
`...`	Other arguments that may be supplied.

Value

A device with the desired plot.

Note

This function only represents the 'tree' option of the original `plot.tree` function of **smida**, because we believe that this option displays better the clustering results provided by HIPAM than the option '2d'.

Author(s)

This function was originally created by E. Wit et al., and it is available freely on http://www.math.rug.nl/~ernst/book/smida.html. We have slightly modified.

References

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Wit, E., and McClure, J., (2004). *Statistics for Microarrays: Design, Analysis and Inference*. John Wiley & Sons, Ltd.

Wit, E., and McClure, J., (2006). Statistics for Microarrays: Inference, Design and Analysis. R package version 0.1. http://www.math.rug.nl/~ernst/book/smida.html.

See Also

`hipamAnthropom`

Examples

```
## Not run:
dataDef <- dataDemo
bust <- dataDef$bust

interv <- c("74-78","78-82","82-86","86-90","90-94","94-98","98-102",
            "102-107","107-113","113-119","119-125","125-131")
bustCirc_4 <- seq(74,102,4)
bustCirc_6 <- seq(107,131,6)
bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7 ; alpha <- 0.01 ; type <- "IMO" #type <- "MO" for $HIPAM_{MO}$

ahVect <- c(23, 28, 20, 25, 25)

hip <- list()
for(i in 1 : (nsizes - 1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type=type,ahVect=ahVect)
}
str(hip)

#Dendogram for the first bust class:
plotTreeHipam(hip[[1]],title=paste("Proposed Hierarchical PAM Clustering \n",interv[1]))

## End(Not run)
```

---

plotTrimmOutl *Trimmed or outlier observations representation*

Description

This function represents the scatter plots of bust circumference against other selected variable (chest,hip,neck to ground or waist) jointly with the trimmed individuals discarded in each bust class provided by `trimowa` or with the outlier individuals provided by `hipamAnthropom`.

Usage

```
plotTrimmOutl(x,rows,nsizes,bustVariable,variable,color,xlim,ylim,title)
```

Arguments

`x`	Data frame. It should contain the chest, neck to ground, waist, hip and bust measurements of the individuals. In order to be able to represent them, the name of the columns of the database must be 'chest', 'necktoground', 'waist', 'hip' and 'bust' respectively, see `dataDemo`. Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
`rows`	Trimmed women (if `trimowa`) or outlier women (if `hipamAnthropom`).

nsizes	Number of subsets (classes), into the database is segmented. In our approach, the whole anthropometric Spanish survey is segmented into twelve bust segments, according to the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals.
bustVariable	Bust variable.
variable	Anthropometric variable to be plotted. It can be 'chest', 'necktoground', 'waist' and 'hip'.
color	A specification for the trimmed or outlier women color in each bust class.
xlim	Axis lenght of the x axis according to the range of the bust variable.
ylim	Axis lenght of the y axis according to the range of the selected variable among chest, hip, neck to ground and waist.
title	Title of the plot.

Value

A device with the desired plot.

Author(s)

Guillermo Vinue

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

See Also

dataDemo, hipamAnthropom, trimowa

Examples

```
## Not run:
#TRIMOWA ALGORITHM:
#Data loading:
dataDef <- dataDemo
num.variables <- dim(dataDef)[2]
bust <- dataDef$bust

orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)

bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
K <- 3 ; alpha <- 0.01 ; niter <- 6 ; Ksteps <- 7
```

```
ahVect <- c(23, 28, 20, 25, 25)

res_trimowa <- list()
for (i in 1 : (nsizes-1)){
  data = dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  res_trimowa[[i]] <- trimowa(data,w,K,alpha,niter,Ksteps,ahVect=ahVect)
}

trimmed <- list()
for (i in 1 : (nsizes-1)){
  trimmed[[i]] <- res_trimowa[[i]][[10]] #or res_trimowa[[i]]$trimm
}

bustVariable <- "bust"
xlim <- c(70,150)
color <- c("black","red","green","blue","cyan","brown","gray","deeppink3",
           "orange","springgreen4","khaki3","steelblue1")

variable <- "chest"
range(dataDef[,variable])
#[1] 76.7755 135.8580
ylim <- c(70,140)
title <- "Trimmed women \n bust vs chest"

plotTrimmOutl(dataDef,trimmed,nsizes,bustVariable,variable,color,xlim,ylim,title)

variable <- "hip"
range(dataDef[,variable])
#[1] 83.6 152.1
ylim <- c(80,160)
title <- "Trimmed women \n bust vs hip"

plotTrimmOutl(dataDef,trimmed,nsizes,bustVariable,variable,color,xlim,ylim,title)

variable <- "necktoground"
range(dataDef[,variable])
#[1] 117.6 154.9
ylim = c(110,160)
title <- "Trimmed women \n bust vs neck to ground"

plotTrimmOutl(dataDef,trimmed,nsizes,bustVariable,variable,color,xlim,ylim,title)

variable <- "waist"
range(dataDef[,variable])
#[1]  58.6 133.0
ylim <- c(50,140)
title <- "Trimmed women \n bust vs waist"

plotTrimmOutl(dataDef,trimmed,nsizes,bustVariable,variable,color,xlim,ylim,title)

#AN EXAMPLE FOR HIPAM ALGORITHM:
dataDef <- dataDemo
bust <- dataDef$bust
```

```
bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
maxsplit <- 5 ; orness <- 0.7 ; alpha <- 0.01 ; type <- "MO" #type <- "IMO" for $HIPAM_{IMO}$

ahVect <- c(23, 28, 20, 25, 25)

hip <- list()
for(i in 1 : (nsizes - 1)){
  data =  dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  d <- as.matrix(data)
  hip[[i]] <- hipamAnthropom(d,maxsplit=maxsplit,orness=orness,type=type,ahVect=ahVect)
}
str(hip)

ress <- list()
for(i in 1 : length(hip)){
  ress[[i]] <- table(hip[[i]]$clustering)
}
ress

bustVariable <- "bust"
xlim <- c(70,150)
color <- c("black","red","green","blue","cyan","brown","gray","deeppink3",
           "orange","springgreen4","khaki3","steelblue1")

variable <- "hip"
ylim <- c(80,160)
title <- "Medoids \n bust vs hip"

list_outl1_2 <- sapply(1:(nsizes-1),outlierHipam,hip)
title <- "Outlier women \n bust vs hip"
plotTrimmOutl(dataDef,list_outl1_2,nsizes,bustVariable,variable,color,xlim,ylim,title)

## End(Not run)
```

`qtranProcrustes` *Auxiliary qtran subroutine of the Hartigan-Wong k-means for 3D shapes*

Description

The Hartigan-Wong version of the k-means algorithm uses two auxiliary algorithms: the optimal transfer stage (optra) and the quick transfer stage (qtran).

This function is the qtran subroutine adapted to the shape analysis context. It is used within `HartiganShapes`. See Hartigan and Wong (1979) for details of the original k-means algorithm and Amaral et al. (2010) for details about its adaptation to shape analysis.

Usage

```
qtranProcrustes(dg,n,c,K,ic1,ic2,nc,an1,an2,ncp,d,itran,indx)
```

Arguments

dg	Array with the 3D landmarks of the sample objects.
n	Number of sample objects.
c	Array of centroids.
K	Number of clusters.
ic1	The cluster to each object belongs.
ic2	This vector is used to remember the cluster which each object is most likely to be transferred to at each step.
nc	Number of objects in each cluster.
an1	$an1(l) = nc(l) / (nc(l) - 1), l=1,...,k$.
an2	$an2(l) = nc(l) / (nc(l) + 1), l=1,...,k$.
ncp	In the optimal transfer stage, ncp(l) stores the step at which cluster l is last updated, $l=1,...,k$. In the quick transfer stage, ncp(l) stores the step at which cluster l is last updated plus n, $l=1,...,k$.
d	Vector of distances from each object to every centroid.
itran	itran(l) = 1 if cluster l is updated in the quick-transfer stage (0 otherwise), $l=1,...,k$.
indx	Number of steps since a transfer took place.

Value

A list with the following elements:: *c,ic1,ic2,nc,an1,an2,ncp,d,itran,indx,icoun*, updated after the optimal transfer stage. Note that *icoun* counts the steps where a re-allocation took place.

Note

This function belongs to HartiganShapes and it is not solely used. That is why there is no section of *examples* in this help page.

Note

This function is based on the optra.m file available from http://people.sc.fsu.edu/~jburkardt/m_src/asa136/asa136.html.

Author(s)

Guillermo Vinue

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Hartigan, J. A., and Wong, M. A., (1979). A K-Means Clustering Algorithm, *Applied Statistics*, 100–108.

Amaral, G. J. A., Dore, L. H., Lessa, R. P., and Stosic, B., (2010). k-Means Algorithm in Statistical Shape Analysis, *Communications in Statistics - Simulation and Computation* **39(5)**, 1016–1026.

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

See Also

HartiganShapes

`screeArchetyp` *Screeplot of archetypes and archetypoids*

Description

This function allows us to represent in the same plot the screeplot of the archetypes and the both *nearest* and *which* archetypoids.

Usage

```
screeArchetyp(k,rss_lass_def,rss_step,rss_step_which,ylim,main,
              xlab,ylab,col=c("red","blue"),axis2,seq,leg)
```

Arguments

`k`	Number of archetypes and archetypoids.
`rss_lass_def`	Vector of the residual sum of squares (rss) associated with each archetype from 1 to `k`.
`rss_step`	Vector of the residual sum of squares (rss) associated with each *nearest* archetypoid from 1 to `k`.
`rss_step_which`	Vector of the residual sum of squares (rss) associated with each *which* archetypoid from 1 to `k`.
`ylim`	The y limits of the plot.
`main`	Title of the plot.
`xlab`	A title for the x axis.
`ylab`	A title for the y axis.
`col`	Color vector for the screeplots of the archetypoids. Default is c("red","blue").
`axis2`	A logical value. If TRUE, the y axis can be customized to have spaced tick-marks by means of the following argument `seq`.
`seq`	Vector sequence with the values of the tick-marks to be drawn in the y axis.
`leg`	If TRUE, a legend is shown.

Value

A device with the desired plot.

Author(s)

Guillermo Vinue

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. *Submitted for publication*.

Cutler, A., and Breiman, L., (1994). Archetypal Analysis, *Technometrics* **36**, 338–347.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

Eugster, M. J., and Leisch, F., (2009). From Spider-Man to Hero - Archetypal Analysis in R, *Journal of Statistical Software* **30**, 1–23, http://www.jstatsoft.org/.

Eugster, M. J. A., (2012). Performance profiles based on archetypal athletes, *International Journal of Performance Analysis in Sport* **12**, 166–187.

See Also

archetypoids, stepArchetypoids

Examples

```
## Not run:
#COCKPIT DESIGN PROBLEM:
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#Computation of archetypes and archetypoids:
#For reproducing results, seed for randomness:
set.seed(2010)
#Run archetypes algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 20
lass <- stepArchetypesMod(data=preproc$data,k=1:numArch,verbose=FALSE,nrep=nrep)

rss_lass <- matrix(0,nrow=numArch,ncol=nrep)
for(i in 1:numArch){
  for(j in 1:nrep){
    rss_lass[i,j] <- lass[[i]][[j]]$rss
  }
```

```
}
(rss_lass_def <- apply(rss_lass,1,min,na.rm=T))

#Run archetypoids algorithm repeatedly from 1 to numArch archetypes:
for(i in 1:numArch){
  temp <-  stepArchetypoids(i,TRUE,preproc$data,lass)
  filename <- paste("res", i, sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename, ".RData", sep=""))
}

for(i in 1:numArch){
  temp <-  stepArchetypoids(i,FALSE,preproc$data,lass)
  filename <- paste("res", i, "_which",sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename, ".RData", sep=""))
}

#Numerical and graphical results:
for(i in 1:numArch){
  load(paste("res", i, ".RData", sep = ""))
}
rss_step <- c()
for (i in 1:numArch){
  rss_step[i] <- get(paste("res", i, sep = ""))[[2]]
}
(rss_step <- as.numeric(rss_step))


for(i in 1:numArch){
  load(paste("res", i, "_which.RData", sep = ""))
}
rss_step_which <- c()
for (i in 1:numArch){
  rss_step_which[i] <- get(paste("res", i, "_which", sep = ""))[[2]]
}
(rss_step_which <- as.numeric(rss_step_which))

main <- "Aircraft pilots archetypes and archetypoids"
xlab <- "Archetypes/Archetypoids"
ylab <- "RSS"
screeArchetyp(numArch,rss_lass_def,rss_step,rss_step_which,c(0.005,0.040),
              main,xlab,ylab,col=c("red","blue"),TRUE,c(0.005,0.040),TRUE)

## End(Not run)
```

---

shapes3dMod *3D shapes plot*

Description

This function is a slight modification of the original shapes3d function of the **shapes** R package so that the resulting plot has customized title and axes. Specifically, the changing lines regarding the original function are those related to its argument *axes3* when it is fixed to TRUE.

Usage

```
shapes3dMod(x,loop=0,type="p",color=2,joinline=c(1:1),axes3=FALSE,rglopen=TRUE,main=main)
```

Arguments

x	See shapes3d.
loop	See shapes3d.
type	See shapes3d.
color	See shapes3d.
joinline	See shapes3d.
axes3	See shapes3d.
rglopen	See shapes3d.
main	Allows us to give the plot a title if axes3=TRUE.

Value

A device with the desired plot.

References

Dryden, I. L., (2012). **shapes** package. R Foundation for Statistical Computing, Vienna, Austria. Contributed package.

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

See Also

shapes3d

Examples

```
## Not run:
#CLUSTERING WOMEN ACCORDING TO THEIR SHAPE:
landmarks1 <- na.exclude(landmarks)
dim(landmarks1)
#[1] 574 198
(num.points <- (dim(landmarks1)[2]) / 3)
#[1] 66
landmarks2 <- landmarks1[1:50,] #In the interests of simplicity of the computation involved.
(n <- dim(landmarks2)[1])
#[1] 100

dg <- array(0,dim = c(num.points,3,n))
```

```
for(k in 1:n){
 for(l in 1:3){
  dg[,l,k] <-  as.matrix(as.vector(landmarks2[k,][seq(l,
                    dim(landmarks2)[2]+(l-1),by=3)]),ncol=1,byrow=T)
 }
}

Nclusters <- 3 ; Nsteps <- 5 ; niter <- 5 ; stopCr <- 0.0001
resLL <- LloydShapes(dg,Nclusters,Nsteps,niter,stopCr,FALSE,TRUE)
copt <- resLL$copt
shapes3dMod(copt[,,1], loop = 0, type = "p", color = 2, joinline = c(1:1),
            axes3 = TRUE, rglopen = TRUE, main = "Mean shape cluster 1")

## End(Not run)
```

---

skeletonsArchet *Skeleton plots of archetypal individuals*

Description

This function represents the skeleton plots of the archetypal observations (archetypes and archetypoids) of dataUSAF.

Usage

```
skeletonsArchet(vect,main)
```

Arguments

vect	Vector with the measurements of each archetype.
main	The title of the plot.

Value

A device with the desired plot.

Note

This function allows us to reproduce the archetypes of Figure 5 of Epifanio et al. (2013), see archetypesBoundary.

Author(s)

Guillermo Vinue

References

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

See Also

archetypesBoundary, dataUSAF

Examples

```
## Not run:
#List with the measurements of each archetype (Table 7 of Epifanio et al (2013)):
lista_arch <- list()
lista_arch[[1]] <- c(34.18, 25.85, 18.65, 39.66, 35.05, 26.73)
lista_arch[[2]] <- c(28.51, 21.23, 15.39, 33.57, 29.24, 21.26)
lista_arch[[3]] <- c(35.34, 24.94, 18.79, 36.7, 32.28, 23.41)
lista_arch[[4]] <- c(31.34, 22.27, 16.89, 38, 33.08, 25.8)
lista_arch[[5]] <- c(32.33, 25.09, 17.84, 34.46, 29.58, 22.82)
lista_arch[[6]] <- c(29.69, 24.18, 18.22, 38.07, 33.04, 24.56)
lista_arch[[7]] <- c(29.24, 22.97, 14.99, 36.88, 32.28, 24.22)

for(i in 1:length(lista_arch)){
 main <- paste("Archetype", i, sep = " ")
 skeletonsArchet(lista_arch[[i]],main)
}

#AN EXAMPLE FOR THE ARCHETYPOIDS:
#COCKPIT DESIGN PROBLEM:
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#For reproducing results, seed for randomness:
set.seed(2010)
#Run archetypes algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10
nrep <- 20
lass <- stepArchetypesMod(data=preproc$data,k=1:10,verbose=FALSE,nrep=20)

i <- 3 #number of archetypoids.
res <- archetypoids(i,preproc$data,huge=200,step=FALSE,ArchObj=lass,nearest=TRUE,sequ=TRUE)

#Looking for the individuals in the non standardized database:
aux <- mpulg[setdiff(1:dim(mpulg)[1],preproc$indivNo),]
rownames(aux) <- 1:dim(preproc$data)[1]

skeletonsArchet(aux[res[[1]][1],],"Archetypoid 1")

## End(Not run)
```

`stepArchetypesMod` *Archetype algorithm to raw data*

Description

This is a slight modification of the original `stepArchetypes` function of the **archetypes** R package to apply the archetype algorithm to raw data. The `stepArchetypes` function standardizes the data by default and this option is not always desired.

Usage

```
stepArchetypesMod(data,k,nrep=3,verbose=TRUE)
```

Arguments

`data`	Data to obtain archetypes.
`k`	Number of archetypes to compute, from 1 to `k`.
`nrep`	For each `k`, run `archetypes` `nrep` times.
`verbose`	If TRUE, the progress during exection is shown.

Value

A list with `k` elements. Each element is a list of class attribute `stepArchetypes` with `nrep` elements.

Author(s)

Guillermo Vinue based on the the original `stepArchetypes` function of **archetypes**.

References

Eugster, M. J., and Leisch, F., (2009). From Spider-Man to Hero - Archetypal Analysis in R, *Journal of Statistical Software* **30**, 1–23, http://www.jstatsoft.org/.

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. *Submitted for publication*.

See Also

`stepArchetypes`

Examples

```
## Not run:
#Cockpit design problem:
#First, the database USAF 1967 is read and preprocessed (Zehner et al (1993).).
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
```

```
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#For reproducing results, seed for randomness:
set.seed(2010)
#Run archetypes algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 3
lass <- stepArchetypesMod(data=preproc$data,k=1:numArch,verbose=FALSE,nrep=nrep)

## End(Not run)
```

`stepArchetypoids` *Run the archetypoid algorithm several times*

Description

Execute the archetypoid algorithm repeatedly. It is inspired by the `stepArchetypes` function of the **archetypes** R package.

Usage

```
stepArchetypoids(i,nearest,data,ArchObj)
```

Arguments

`i`	Number of archetypoids.
`nearest`	Initial vector of archetypoids for the BUILD phase of the archetypoid algorithm. This argument is a logical value: if TRUE (FALSE), the *nearest* (*which*) vector is calculated. Both vectors contain the nearest individuals to the archetypes returned by the `archetypes` function of **archetypes** (In Vinue et al. (2014), archetypes are computed after running the archetype algorithm twenty times). The *nearest* vector is calculated by computing the Euclidean distance between the archetypes and the individuals and choosing the nearest. It is used in Epifanio et al. (2013). The *which* vector is calculated by identifying consecutively the individual with the maximum value of alpha for each archetype, until getting the number of archetypes defined. It is used in Eugster (2012).
`data`	Data matrix. Each row corresponds to an observation and each column corresponds to an anthropometric variable. All variables are numeric.
`ArchObj`	The list returned by the `stepArchetypesMod` function. This function is a slight modification of the original `stepArchetypes` function of **archetypes** to apply the archetype algorithm to raw data. The `stepArchetypes` function standardizes the data by default and this option is not always desired. This list is needed to compute the nearest individuals to archetypes.

Value

A list with the following elements:

archet: Final vector of k archetypoids.

rss: Residual sum of squares corresponding to the final vector of k archetypoids.

inits: Vector of initial archetypoids (*nearest* or *which*).

Note

It may be happen that archetypes does not find results for *k* archetypes. In this case, it is not possible to calculate the vector of nearest individuals and consequently, the vector of archetypoids. Therefore, this function will return an error message.

Author(s)

Irene Epifanio and Guillermo Vinue

References

Vinue, G., Epifanio, I., and Alemany, S., (2014). Archetypoids: a new approach to define representative archetypal data. *Submitted for publication*.

Cutler, A., and Breiman, L., (1994). Archetypal Analysis, *Technometrics* **36**, 338–347.

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

Eugster, M. J., and Leisch, F., (2009). From Spider-Man to Hero - Archetypal Analysis in R, *Journal of Statistical Software* **30**, 1–23, http://www.jstatsoft.org/.

Eugster, M. J. A., (2012). Performance profiles based on archetypal athletes, *International Journal of Performance Analysis in Sport* **12**, 166–187.

See Also

archetypoids, archetypes, stepArchetypes

Examples

```
## Not run:
#Cockpit design problem:
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#Computation of archetypes and archetypoids:
#For reproducing results, seed for randomness:
```

```
set.seed(2010)
#Run archetypes algorithm repeatedly from 1 to numArch archetypes:
numArch <- 10 ; nrep <- 20
lass <- stepArchetypesMod(data=preproc$data,k=1:numArch,verbose=FALSE,nrep=nrep)

#Run archetypoids algorithm repeatedly from 1 to numArch archetypes:
for(i in 1:numArch){
  temp <-  stepArchetypoids(i,TRUE,preproc$data,lass)
  filename <- paste("res", i, sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename, ".RData", sep=""))
}

for(i in 1:numArch){
  temp <-  stepArchetypoids(i,FALSE,preproc$data,lass)
  filename <- paste("res", i, "_which",sep="")
  assign(filename,temp)
  save(list=c(filename),file=paste(filename, ".RData", sep=""))
}

## End(Not run)
```

`TDDclust` *Trimmed clustering based on L1 data depth*

Description

This is the trimmed version of the clustering algorithm based on the L1 depth proposed by Rebecka Jornsten (2004). She segments all the observations in clusters, and assigns to each point z in the data space, the L1 depth value regarding its cluster. A trimmed procedure is incorporated to remove the more extreme individuals of each cluster (those one with the lowest depth values), in line with `trimowa`.

Usage

```
TDDclust(x,K,lambda,Th,A,T0,alpha,lplot,Trimm,data1)
```

Arguments

`x`	Data frame. Each row corresponds to an observation, and each column corresponds to a variable. All variables must be numeric.
`K`	Number of clusters.
`lambda`	Tuning parameter that controls the influence the data depth has over the clustering, see Jornsten (2004).
`Th`	Threshold for observations to be relocated, usually set to 0.
`A`	Number of iterations.
`T0`	Simulated annealing parameter. It is the current temperature in the simulated annealing procedure.

alpha	Simulated annealing parameter. It is the decay rate, default 0.9.
lplot	Tracking convergence, default 0.
Trimm	Proportion of no accommodated sample.
data1	The same data frame as *x*, used to incorporate the trimmed observations to the rest of them for the next iteration.

Value

A list with the following elements:

NN: Cluster assignment, NN[1,] is the final partition.

Y: The multivariate median cluster representatives

DD: Depth values of the observations.

Cost: Final value of the optimal partition.

indivTrimmed: Trimmed observations.

klBest: Iteration in which the optimal partition was found.

Author(s)

This function has been defined from the original functions developed by Rebecka Jornsten, which were available freely on http://www.stat.rutgers.edu/home/rebecka/DDcl/. However, the link to this page doesn't currently exist as a result of a website redesign.

References

Jornsten R., (2004). Clustering and classification based on the L1 data depth, *Journal of Multivariate Analysis* **90**, 67–89

Vinue, G., and Ibanez, M. V., (2014). *Data depth and Biclustering applied to anthropometric data. Exploring their utility in apparel design*. In progress.

Examples

```
## Not run:
dataDef <- dataDemo[1:50,c(2,3,5)] #In the interests of simplicity of the computation involved.
data1 <- dataDemo[1:50,c(2,3,5)]

K=3     ; lambda=0.5    ; Th=0;
A=5     ; T0=0          ; alpha=.9
lplot=0 ; percTrimm=0.1

Dout <- TDDclust(x=dataDef,K=K,lambda=lambda,Th=Th,A=A,T0=T0,alpha=alpha,
                 lplot=lplot,Trimm=percTrimm,data1=data1)

table(Dout$NN[1,])

Dout$Cost
Dout$klBest

Dout$indivTrimmed
```

```
dataDef[Dout$indivTrimmed,]

## End(Not run)
```

`trimmedLloydShapes` *Trimmed Lloyd k-means for 3D shapes*

Description

The basic foundation of k-means is that the sample mean is the value that minimizes the Euclidean distance from each point, to the centroid of the cluster to which it belongs. Two fundamental concepts of the statistical shape analysis are the Procrustes mean and the Procrustes distance. Therefore, by integrating the Procrustes mean and the Procrustes distance we can use k-means in the shape analysis context.

The k-means method has been proposed by several scientists in different forms. In computer science and pattern recognition the k-means algorithm is often termed the Lloyd algorithm (see Lloyd (1982)).

This function is proposed to incorporate a modification to `LloydShapes` in order to make the k-means algorithm robust. Robustness is a property very desirable in a lot of applications. As it is well known, the results of the k-means algorithm can be influenced by outliers and extreme data, or bridging points between clusters. Garcia-Escudero et al. (1999) propose a way of making k-means more robust, which combines the k-means idea with an impartial trimming procedure: a proportion alpha (between 0 and 1) of observations are trimmed (the trimmed observations are self-determined by the data). See also `trimmedoid`.

Usage

```
trimmedLloydShapes(dg,n,alpha,K,Nsteps=10,niter=10,stopCr=0.0001,print)
```

Arguments

`dg`	Array with the 3D landmarks of the sample objects. Each row corresponds to an observation, and each column corresponds to a dimension (x,y,z).
`n`	Number of individuals.
`alpha`	Proportion of trimmed sample.
`K`	Number of clusters.
`Nsteps`	Number of steps per initialization. Default value is 10.
`niter`	Number of random initializations. Default value is 10.
`stopCr`	Relative stopping criteria. Default value is 0.0001.
`print`	Logical value. If TRUE, some messages associated with the running process are displayed.

Value

A list with the following elements:

asig: Optimal clustering.

copt: Optimal centers.

vopt: Optimal objective function.

trimmWomen: List to save the trimmed individual of each iteration.

trimmsIter: Vector with the number of iterations where the optimum was reached. The last number different from NA refers to the last iteration where the final optimum was reached.

betterNstep: Nstep of the iteration where the optimum has reached.

initials: Random initial values used in each iteration. These values can be used by `HartiganShapes`.

Note

We note that adding a trimmed procedure to the Lloyd algorithm is very direct and easy, while for the Hartigan-Wong algorithm, more modifications of the algorithm are needed, which makes the implementation of its trimmed version difficult.

Author(s)

Amelia Simo

References

Vinue, G., Simo, A., and Alemany, S., (2014). The k-means algorithm for 3D shapes with an application to apparel design. Submitted for publication.

Lloyd, S. P., (1982). Least Squares Quantization in PCM, *IEEE Transactions on Information Theory* **28**, 129–137.

Dryden, I. L., and Mardia, K. V., (1998). *Statistical Shape Analysis*, Wiley, Chichester.

Garcia-Escudero, L. A., Gordaliza, A., and Matran, C., (2003). Trimming tools in exploratory data analysis, *Journal of Computational and Graphical Statistics* **12(2)**, 434–449.

Garcia-Escudero, L. A., and Gordaliza, A., (1999). Robustness properties of k-means and trimmed k-means, *Journal of the American Statistical Association* **94(447)**, 956–969.

See Also

`LloydShapes`, `trimmedoid`

Examples

```
## Not run:
landmarks1 <- na.exclude(landmarks)
dim(landmarks1)
#[1] 574 198
(num.points <- (dim(landmarks1)[2]) / 3)
#[1] 66
landmarks2 <- landmarks1[1:50,] #In the interests of simplicity of the computation involved.
```

```
(n <- dim(landmarks2)[1])
#[1] 50

dg <- array(0,dim = c(num.points,3,n))
for(k in 1:n){
 for(l in 1:3){
  dg[,l,k] <-  as.matrix(as.vector(landmarks2[k,][seq(l,
                   dim(landmarks2)[2]+(l-1),by=3)]),ncol=1,byrow=T)
 }
}

K <- 3 ; alpha <- 0.01 ; Nsteps <- 5 ; niter <- 5 ; stopCr <- 0.0001
res <- trimmedLloydShapes(dg,n,alpha,K,Nsteps,niter,stopCr,TRUE)

#To identify the trimmed women of the optimal iteration:
iter_opt <- res$trimmsIter[length(res$trimmsIter)]
trimm_women <- res$trimmWomen[[iter_opt]][[res$betterNstep]]

#Optimal partition:
table(res$asig)

## End(Not run)
```

`trimmedoid` *Trimmed k-medoids algorithm*

Description

This is the trimmed k-medoids algorithm. It is used within `trimowa`. It is analogous to k-medoids but a proportion alpha of observations is discarded by the own procedure (the trimmed observations are self-determined by the data). Furthermore, the trimmed k-medoids is analogous to trimmed k-means. An algorithm for computing trimmed k-means can be found in Garcia-Escudero et al. (2003). See Ibanez et al. (2012) for more details.

Usage

```
trimmedoid(D,K,alpha,niter,Ksteps)
```

Arguments

`D`	Dissimilarity matrix.
`K`	Number of clusters.
`alpha`	Proportion of trimmed sample.
`niter`	Number of random initializations.
`Ksteps`	Steps per initialization.

Value

A list with the following elements:

vopt: The objective value.

copt: The trimmed medoids.

asig: The assignation of each observation (asig=0 indicates trimmed individuals).

ch: The goodness index.

Dmod: Modified data with the non-trimmed women.

qq: Vector with the non-trimmed points.

Author(s)

Irene Epifanio

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Garcia-Escudero, L. A., Gordaliza, A., and Matran, C., (2003). Trimming tools in exploratory data analysis, *Journal of Computational and Graphical Statistics* **12(2)**, 434–449.

Garcia-Escudero, L. A., and Gordaliza, A., (1999). Robustness properties of k-means and trimmed k-means, *Journal of the American Statistical Association* **94(447)**, 956–969.

See Also

dataDemo, WeightsMixtureUB, GetDistMatrix, trimowa, trimmedLloydShapes

Examples

```
## Not run:
#Loading the data:
#Example for the first bust class:
data = dataDemo[(dataDemo$bust >= 74) & (dataDemo$bust < 78), ]
num.variables <- dim(data)[2]

#Weights calculation:
orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)

#Constants required to specify the distance function:
K <- 3
bh <- (apply(as.matrix(log(data)), 2, range)[2,]
       - apply(as.matrix(log(data)), 2, range)[1,]) / ((K-1) * 8)
bl <- -3 * bh
ah <- c(28,20,30,25,23)
al <- 3 * ah

#Data processing.
```

```
num.persons <- dim(data)[1]
num.variables <- dim(data)[2]
datam <- as.matrix(data)
datat <- aperm(datam, c(2,1))
dim(datat) <- c(1,num.persons*num.variables)
rm(datam)

#Dissimilarity matrix:
D <- GetDistMatrix(datat,num.persons,num.variables,w,bl,bh,al,ah,T)
rm(datat)

trimmedoid(D,K,0.01,6,7)

## End(Not run)
```

`trimowa` *Trimmed PAM with OWA operators*

Description

This is the methodology developed in Ibanez et al. (2012) to define an efficient apparel sizing system based on clustering techniques jointly with OWA operators. In our approach, we apply the trimmed k-medoids algorithm (`trimmedoid`) to the first twelve bust classes according to the sizes defined in the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals.

Usage

```
trimowa(x,w,K,alpha,niter,Ksteps,ahVect=c(23,28,20,25,25))
```

Arguments

`x`	Data frame. In our approach, this is each one of the subframes originated after segmenting the whole anthropometric Spanish survey in twelve bust segments, according to the European standard to sizing system. Size designation of clothes. Part 3: Measurements and intervals. Each row corresponds to an observation, and each column corresponds to a variable. All variables are numeric.
`w`	The aggregation weights of the OWA operators. They are computed with the `WeightsMixtureUB`.
`K`	Number of clusters.
`alpha`	Proportion of trimmed sample.
`niter`	Number of random initializations.
`Ksteps`	Steps per initialization.
`ahVect`	Constants that define the ah slopes of the distance function in `GetDistMatrix`. Given the five variables considered, this vector is c(23,28,20,25,25). This vector would be other according to the variables considered.

Value

A list with the following elements:

meds: Centroids of the clusters. They are the medoids obtained for each bust class.

numTrim: Number of trimmed individuals in each bust class.

numClass: Number of individuals in each bust class.

noTrim: Number of of non-trimmed individuals.

C1,C2,C3,C4: Required constant values to define the distance `GetDistMatrix` (*C1* is bh, *C2* is bl, *C3* is ah and *C4* is al).

asig: Vector of the clusters to which each individual belongs.

trimm: Trimmed individuals.

Author(s)

Guillermo Vinue

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

European Committee for Standardization. Size designation of clothes. Part 3: Measurements and intervals. (2005).

See Also

`dataDemo`, `WeightsMixtureUB`, `GetDistMatrix`, `trimmedoid`

Examples

```
## Not run:
#Loading the data to apply the trimowa algorithm:
dataDef <- dataDemo
num.variables <- dim(dataDef)[2]
bust <- dataDef$bust

orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)

bustCirc_4 <- seq(74,102,4)  ; bustCirc_6 <- seq(107,131,6)  ; bustCirc <- c(bustCirc_4,bustCirc_6)
nsizes <- length(bustCirc)
K <- 3 ; alpha <- 0.01 ; niter <- 10 ; Ksteps <- 7

ahVect <- c(23,28,20,25,25)

res_trimowa <- list()
for (i in 1 : (nsizes-1)){
  data = dataDef[(bust >= bustCirc[i]) & (bust < bustCirc[i + 1]), ]
  res_trimowa[[i]] <- trimowa(data,w,K,alpha,niter,Ksteps,ahVect=ahVect)
```

```
}

## End(Not run)
```

`WeightsMixtureUB` *Calculation of the weights for the OWA operators*

Description

This function calculates the weights of the OWA operators. They can be used to adjust the compromise between the style of garments and the general comfort sensation of wearers. This function is used both in `trimowa` and `hipamAnthropom`.

Usage

```
WeightsMixtureUB(orness,dimension)
```

Arguments

`orness`	Quantity to measure the degree to which the aggregation is like a min or max operation.
`dimension`	Number of variables of the database.

Value

Vector with the weights.

Author(s)

Guillermo Ayala

References

Ibanez, M. V., Vinue, G., Alemany, S., Simo, A., Epifanio, I., Domingo, J., and Ayala, G., (2012). Apparel sizing using trimmed PAM and OWA operators, *Expert Systems with Applications* **39**, 10512–10520.

Vinue, G., Leon, T., Alemany, S., and Ayala, G., (2013). Looking for representative fit models for apparel sizing, *Decision Support Systems* **57**, 22–33.

Leon, T., Zuccarello, P., Ayala, G., de Ves, E., and Domingo, J., (2007), Applying logistic regression to relevance feedback in image retrieval systems, *Pattern Recognition* **40**, 2621–2632.

See Also

`dbinom`, `GetDistMatrix`, `trimowa`, `hipamAnthropom`

Examples

```
## Not run:
num.variables <- dim(dataDemo)[2]
orness <- 0.7
w <- WeightsMixtureUB(orness,num.variables)

## End(Not run)
```

`xyplotPCA` *PC scores for archetypes*

Description

This function is a small modification of the generic `xyplot` function of the **archetypes** R package. It shows the scores for the principal components of all individuals jointly with the scores for the computed archetypes. This function is used to obtain the Figure 4 of the subsection 3.3 of Epifanio et al. (2013).

Value

A device with the desired plot.

Note

There are no usage and arguments sections in this help file because they are the same than those of the page 25 of the reference manual of **archetypes**.

Author(s)

Irene Epifanio

References

Epifanio, I., Vinue, G., and Alemany, S., (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem, *Computers & Industrial Engineering* **64**, 757–765.

See Also

`archetypesBoundary`, `dataUSAF`

Examples

```
## Not run:
#First,the USAF 1967 database is read and preprocessed (Zehner et al. (1993)).
m <- dataUSAF
#Variable selection:
sel <- c(48,40,39,33,34,36)
#Changing to inches:
mpulg <- m[,sel] / (10 * 2.54)

#Data preprocessing:
preproc <- accommodation(mpulg,TRUE,0.95,TRUE)

#Procedure and results shown in section 2.2.2 and section 3.1:
res <- archetypesBoundary(preproc$data,15,FALSE,3)

a3 <- archetypes::bestModel(res[[3]])
a7 <- archetypes::bestModel(res[[7]])

pznueva <- prcomp(preproc$data,scale=T,retx=T)
#PCA scores for 3 archetypes:
p3 <- predict(pznueva, archetypes::parameters(a3))
#PCA scores for 7 archetypes:
p7 <- predict(pznueva,archetypes::parameters(a7))
#Figure 4 (a):
xyplotPCA(p3[,1:2],pznueva$x[,1:2],data.col=gray(0.7),atypes.col=1,atypes.pch=15)
#Figure 4 (b):
xyplotPCA(p7[,1:2],pznueva$x[,1:2],data.col=gray(0.7),atypes.col=1,atypes.pch=15)

## End(Not run)
```

Index