Package ‘PTAk’

February 6, 2019

Version 1.3-34

Date 2019-02-06

Title Principal Tensor Analysis on k Modes

Author Leibovici & Sabatier (1998) <doi:10.1016/S0024-3795(97)81516-9>, Didier G. Leibovici (2010)<doi:10.18637/jss.v034.i10>

Maintainer Didier G. Leibovici <GeotRYcs@gmail.com>

Depends tensor

Imports graphics, stats, utils

Description A multiway method to decompose a tensor (array) of any order, as a generalisation of SVD also supporting non-identity metrics and penalisations. 2-way SVD with these extensions is also available. The package includes also some other multiway methods: PCAn (Tucker-n) and PARAFAC/CANDECOMP with these extensions.

License GPL (>= 2)

URL http://c3s2i.free.fr/

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-06 16:13:22 UTC

R topics documented:

APSOLU3 *Associated 3-modes Principal Tensors of a 3-modes Principal Tensor*

Description

Computes all the 2-modes solutions associated to the given Principal Tensor of the given tensor.

Usage

```
APSOLU3(X,solu,pt3=NULL,nbPT2=1,
                  smoothing=FALSE,smoo=list(NA),
                         verbose=getOption("verbose"),file=NULL, ...)
```

Arguments

`X`	a tensor (as an array) of order *3*, if non-identity metrics are used `X` is a list with `data` as the array and `met` a list of metrics
`solu`	a `PTAk` object
`pt3`	a number identifying in `solu` the Principal Tensor to use or the last (if `NULL`)
`nbPT2`	integer, if 1 all solutions will be computed otherwise at maximum nbPT2 solutions
`smoothing`	see `SVDgen`
`smoo`	see `PTA3`
`verbose`	control printing
`file`	output printed at the prompt if `NULL`, or printed in the given '`file`'
`...`	any other arguments passed to SVDGen or other functions

Details

For each component of the identified Principal Tensor given in `solu`, an SVD of the contracted product of `X` and the component is done. This gives all the associated Principal Tensors which updates `solu` supposed to contain Principal Tensors of `X`.

Value

an updated `PTAk` object

Note

Usually (i.e. as in `PTA3` and `PTAk`) the principal tensor used is the first Principal Tensor of X (and is the last updated in `solu`). If it is another Principal Tensor, the obtained associated solutions do not *stricto sensu* refer to the SVD-*k*modes decomposition (because the orthogonality is defined in the whole tensor space not necessarily on each component space) but are still meaningful.

Author(s)

Didier G. Leibovici

References

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329

See Also

`PTA3`, `APSOLUk`

`APSOLUk` *Associated k-modes Principal Tensors of a k-modes Principal Tensor*

Description

Computes all the (k-1)-modes associated solutions to the given Principal Tensor of the given tensor. Calls recursively PTAk.

Usage

```
APSOLUk(X,solu,nbPT,nbPT2=1,
                      smoothing=FALSE,smoo=list(NA),
                       minpct=0.1,ptk=NULL,
                        verbose=getOption("verbose"),file=NULL,
                         modesnam=NULL, ...)
```

Arguments

X	a tensor (as an array) of order *k*, if non-identity metrics are used X is a list with data as the array and met a list of metrics
solu	a PTAk object
nbPT	a number or a vector of dimension *(k-2)*
nbPT2	integer, if 0 no 2-modes solutions will be computed, 1 means all, >1 otherwise
smoothing	see SVDgen
smoo	see PTA3
minpct	numerical 0-100 to control of computation of future solutions at this level and below
ptk	a number identifying in solutions the Principal Tensor to use or the last (if NULL)
verbose	control printing
file	output printed at the prompt if NULL, or printed in the given 'file'
modesnam	character vector of the names of the modes, if NULL "mo 1" ..."mo k"
...	any other arguments passed to PTAk or other functions

Details

For each component of the identified Principal Tensor given in solutions, a PTA-*(k-1)*modes of the contracted product of X and the component is done. This gives all the associated Principal Tensors which updates solutions supposed to contain a Principal Tensors of X at the first place. For full description of arguments see PTAk.

Value

an updated PTAk object

Note

Usually (*i.e.* as in PTA3 and PTAk) the principal tensor used is the first Principal Tensor of X (and is the last updated in solutions). If it is another Principal Tensor, the obtained associated solutions do not *stricto sensu* refer to the SVD-*k*modes decomposition (because the orthogonality is defined in the whole tensor space not necessarily on each component space) but are still meaningful. This function is usually called by PTAk but can be used on its own to carry on a PTAk analysis if X is the projected (of the original data) on the orthogonal of all the *k*modes Principal Tensor. In other words the ptk rank-one tensor in solutions should be the first best rank-one tensor approximating X for this decomposition analysis to be called PTA-*k*modes.

Author(s)

Didier G. Leibovici

References

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

See Also

PTAk

CANDPARA *CANonical DECOMPosition analysis and PARAllel FACtor analysis*

Description

Performs the identical models known as PARAFAC or CANDECOMP model.

Usage

```
CANDPARA(X,dim=3,test=1E-8,Maxiter=1000,
                      smoothing=FALSE,smoo=list(NA),
                       verbose=getOption("verbose"),file=NULL,
                        modesnam=NULL,addedcomment="")
```

Arguments

`X`	a tensor (as an array) of order k, if non-identity metrics are used `X` is a list with `data` as the array and `met` a list of metrics.
`dim`	a number specifying the number of rank-one tensors
`test`	control of convergence
`Maxiter`	maximum number of iterations allowed for convergence
`smoothing`	see SVDgen
`smoo`	see PTA3
`verbose`	control printing
`file`	output printed at the prompt if `NULL`, or printed in the given '`file`'
`modesnam`	character vector of the names of the modes, if `NULL` `"mo 1"` ...`"mo k"`
`addedcomment`	character string printed after the title of the analysis

Details

Looking for the best rank-one tensor approximation (LS) the three methods described in the package are equivalent. If the number of tensors looked for is greater then one the methods differs: PTA-*k*modes will look for best approximation according to the *orthogonal rank* (*i.e.* the rank-one tensors are orthogonal), PCA-*k*modes will look for best approximation according to the *space ranks* (*i.e.* the ranks of all (simple) bilinear forms , that is the number of components in each space), PARAFAC/CANDECOMP will look for best approximation according to the *rank* (*i.e.* the rank-one tensors are not necessarily orthogonal). For sake of comparisons the PARAFAC/CANDECOMP method and the PCA-*n*modes are also in the package but complete functionnality of the use these methods and more complete packages may be checked at the www site quoted below.

Value

a `CANDPARA` (inherits from `PTAk`) object

Note

The use of metrics (diagonal or not) and smoothing extends flexibility of analysis. This program runs slow! A PARAFAC orthogonal can be done with PTAk looking only for k-modes Principal Tensors *i.e.* with the options `nbPT=c(rep(0,k-2),dim), nbPT2=0`. It is identical to look in any `PTAk` decomposition only for the *k*modes solution but obviously with unecessary computations.

Author(s)

Didier G. Leibovici

References

Caroll J.D and Chang J.J (1970) *Analysis of individual differences in multidimensional scaling via n-way generalization of 'Eckart-Young' decomposition*. Psychometrika 35,283-319.

Harshman R.A (1970) *Foundations of the PARAFAC procedure: models and conditions for 'an explanatory' multi-mode factor analysis*. UCLA Working Papers in Phonetics, 16,1-84.

Kroonenberg P (1983) *Three-mode Principal Component Analysis: Theory and Applications*. DSWO press. Leiden.(related references in http://three-mode.leidenuniv.nl)

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

`CauRuimet` *Robust estimation of within group varinace-covariance*

Description

Gives a robust estimate of an unknown within group covariance, aiming either to look for dense groups or to sparse groups (outliers) according to *local variance and weighting function* choice.

Usage

```
CauRuimet(Z,ker=1,m0=1,withingroup=TRUE,
          loc=substitute(apply(Z,2,mean,trim=.1)),matrixmethod=TRUE, Nrandom=3000)
```

Arguments

`Z`	matrix
`ker`	either numerical or a function: if numerical the weighting function is $e^{(-ker\ t)}$, otherwise ker=function(t){return(expression)} is a positive decreasing function.
`m0`	is a graph of neighbourhood or another proximity matrix, the hadamard product of the proximities will be operated
`withingroup`	logical,if `TRUE` the aim is to give a robust estimate for dense groups, if `FALSE` the aim is to give a robust estimate for outliers
`loc`	a vector of locations or a function using mean, median, to give an estimate of it
`matrixmethod`	if `TRUE` (only with `withingroup`) uses some matrix computation rather than double looping as suggests the formula below
`Nrandom`	if `Nrandom < dim(Z)[1]`) uses only a `Nrandom` sample from rows of `Z` and `m0` if applicable.

Details

When withingroup is `TRUE`, local(defined by the weighting) variance formula is returned, aiming at finding dense groups:

$$W_l = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m0_{ij} ker(d^2_{S^-}(Z_i, Z_j))(Z_i - Z_j)'(Z_i - Z_j)}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m0_{ij} ker(d^2_{S^-}(Z_i, Z_j))}$$

where $d^2_{S^-}(.,.)$ is the squared euclidian distance with S^- the inverse of a robust sample covariance (i.e. using `loc` instead of the mean) ; if `FALSE` robust Total weighted variance or if `m0` not 1 Global weighted variance, is returned:

$$W_o = \frac{\sum_{i=1}^{n} ker(d^2_{S^-}(Z_i, \tilde{Z}))(Z_i - \tilde{Z})'(Z_i - \tilde{Z})}{\sum_{i=1}^{n} ker(d^2_{S^-}(Z_i, \tilde{Z}))}$$

$$W_g = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m0_{ij}.ker(d^2_{S^-}(Z_i, Z_j))(Z_i - \tilde{Z})'(Z_j - \tilde{Z})}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m0_{ij} ker(d^2_{S^-}(Z_i, Z_j))}$$

where $\tilde{Z}$ is the vector `loc`.
If `m0` is a graph of neighbourhood and ker is the function returning 1 (no proximity due to distance is used) the function will return (when `withingroup=TRUE`) the *local variance-covariance* matrix as define in Lebart(1969).

Value

a matrix

Note

As mentioned by Caussinus and Ruiz a good strategy to reveal dense groups with generalised PCA would be to reveal outliers first using the metric W_o^{-1} and remove them before using the metric W_l^{-1}. Based on theoretical considerations they recommand for the choice of `ker`, with the decreasing function $e^{(-ker\ t)}$: a lower bound of 1 if `withingroup` and something fairly small say in the interval [0.05;0.3] otherwise.

Author(s)

Didier G. Leibovici

References

Caussinus, H and Ruiz, A (1990) *Interesting Projections of Multidimensional Data by Means of Generalized Principal Components Analysis.* COMPSTAT90, Physica-Verlag, Heidelberg,121-126.

Faraj, A (1994) *Interpretation tools for Generalized Discriminant Analysis.*In: New Approches in Classification and Data Analysis, Springer-Verlag, 286-291, Heidelberg.

Lebart, L (1969) *Analyse statistique de la contiguit<e9>e.*Publication de l'Institut de Statistiques Universitaire de Paris, XVIII,81-112.

Leibovici D (2008) *Spatio-temporal Multiway Decomposition using Principal Tensor Analysis on k-modes: the R package* **PTAk** . to be submitted soon at Journal of Statisticcal Software.

See Also

`SVDgen`

Examples

```
data(iris)
 iris2 <- as.matrix(iris[,1:4])
 dimnames(iris2)[[1]] <- as.character(iris[,5])

D2 <- CauRuimet(iris2,ker=1,withingroup=TRUE)
D2 <- Powmat(D2,(-1))
iris2 <- sweep(iris2,2,apply(iris2,2,mean))
res <- SVDgen(iris2,D2=D2,D1=1)
plot(res,nb1=1,nb2=2,cex=0.5)
summary(res,testvar=0)

# the same in a demo function
# source(paste(R.home(),"/library/PTAk/demo/CauRuimet.R",sep=""))
# demo.CauRuimet(ker=4,withingroup=TRUE,openX11s=FALSE)
# demo.Cauruimet(ker=0.15,withingroup=FALSE,openX11s=FALSE)
```

CONTRACTION *Contraction of two tensors*

Description

Computes the contraction product of two tensors as a generalisation of matrix product.

Usage

```
CONTRACTION(X,z, Xwiz=NULL,zwiX=NULL,rezwiX=FALSE,usetensor=TRUE)
CONTRACTION.list(X,zlist,moins=1,zwiX=NULL,usetensor=TRUE,withapply=FALSE)
```

Arguments

`X`	a tensor(as an array) of any order
`z`	another tensor (with at least one space in common)
`zlist`	a list of lists like a `solution.PTAk` at least with `v` for every list(here `v` can be any array)
`Xwiz`	`Xwiz` is to specify the entries of `X` to contract with entries of `z` specified by `zwiX`, if `Xwiz` `NULL` `dim(z)[zwiX]` matching `dim(X)` will do without ambiguity (taking all `z` dimensions if `zwiX` is `NULL`). In `CONTRACTION.list` it is not set as one supposes the contractions in the list to operate follow the dimensions of X
`zwiX`	idem as `Xwiz`. If both `Xwiz` and `zwiX` are `NULL` `zwiX`is replaced by full possibilities (`1:length(dimz)`) then `Xwiz` is looked for. In `CONTRACTION.list` it is the vector for dimensions in the `v` to contract with X. Only 1-way dimension for each `v`.
`moins`	the elements in `zlist` to skip (see also `TENSELE`)
`rezwiX`	logical if `TRUE` (and zwiX is `NULL`) rematches the dimensions in for `zwiX`: useful only if the dimensions of z were not following the Xwiz order and are not equals.
`usetensor`	if `TRUE` uses `tensor` (add-on package)
`withapply`	if `TRUE` (only for vectors in `zlist` uses `apply`

Details

Like two matrices *contract* according to the appropriate dimensions (columns matching rows) when one performs a matrix product, this operation does pretty much the same thing for tensors(array) and specified contraction dimensions given by `Xwiz` and `zwiX` which should match. The function is actually written like: transforms both tensors as matrices with the “matching tensor product" of their contraction dimensions in columns (for higher order tensor) and rows (the other one), performs the matrix product and rebuild the result as a tensor(array). Without using `tensor`, if `Xwiz` and/or `zwiX` are not specified the functions tries to match all `z` dimensions onto the dimensions of X where X is the higher order tensor (if it is not the case in the arguments the function swaps them).

Value

A tensor of dimension `c(dim(X)[-Xwiz],dim(z)[-zwiX])` if `X` has got a bigger order than `z`.

Note

This operation generalises the *matrix* product to the *contracted* product of any two tensors(arrays), and should theoretically perform the tensor product if no matching (no contraction) but has not been implemented. I recently put the option of using `tensor` which does exactly the same thing faster as well as it is from `C`. When using `tensor` if `Xwiz` or `zwiX` are `NULL` they are replaced by the full possibilities.

Author(s)

Didier G. Leibovici

References

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

Schwartz L (1975) *Les Tenseurs*. Herman, Paris.

See Also

PTAk, APSOLUk

Examples

```
 library(tensor)
   z <-  array(1:12,c(2,3,2))
   X <- array(1:48,c(3,4,2,2))
   Xcz <- CONTRACTION(X,z,Xwiz=c(1,3,4),zwiX=c(2,3,1))
   dim(Xcz)   # 4
   Xcz1 <- CONTRACTION(X,z,Xwiz=c(3,4),zwiX=c(1,3))
   dim(Xcz1) # 3,4,3
   Xcz2 <- CONTRACTION(X,z,Xwiz=c(3,4),zwiX=c(3,1))
   Xcz1[,,1]
   Xcz2[,,1]
   #######
   sval0 <- list(list(v=c(1,2,3,4)),list(v=rep(1,3)),list(v=c(1,3)))
   tew <- array(1:24,c(4,3,2))
    CONTRACTION.list(tew,sval0,moins=1)
       #this is equivalent to the following which may be too expensive for big datasets
    CONTRACTION(tew,TENSELE(sval0,moins=1),Xwiz=c(2,3))
   ##
     CONTRACTION.list(tew,sval0,moins=c(1,2)) #must be equal to
     CONTRACTION(tew,sval0[[3]]$v,Xwiz=3)
```

COS2-CTR *Interpretation summaries*

Description

After a FCA2, a SVDgen, a FCAk or a PTAk computes the traditional guides for interpretations used in PCA and correspondence analysis: COS2 or the percentage of variability rebuilt by the component and CTR or the amount of contribution towards that component.

Usage

```
COS2(solu, mod=1, solnbs=2:4)
CTR(solu, mod=1, solnbs=1:4)
```

Arguments

solu	an object inheriting from class PTAk, representing a generalised singular value decomposition
mod	an integer representing the mode number entry, 1 is row, 2 columns, ...
solnbs	a vector of integers representing the tensor numbers in the listing summary

Details

Classical measures helping to interpret the plots in PCA, FCA and in PTAk as well. The sum of the COS2 across all the components needed to rebuild fully the tensor analysed) would make 1000 and the sum pf the CTR across the entry mode would be 1000.

Value

a matrix whose columns are the COS2 or CTR as per thousands (%■) for the mode considered

Author(s)

Didier G. Leibovici

References

Escoufier Y (1985) *L'Analyse des correspondances : ses propriétés et ses extensions*. ISI 45th session Amsterdam.

Leibovici D(1993) *Facteurs à Mesures Répétées et Analyses Factorielles : applications à un suivi Epidémiologique*. Université de Montpellier II. PhD Thesis in Mathématiques et Applications (Biostatistiques).

Leibovici DG (2010) *Spatio-temporal Multiway Decomposition using Principal Tensor Analysis on k-modes:the R package* **PTAk**. Journal of Statistical Software, 34(10), 1-34. http://www.jstatsoft.org/v34/i10/

See Also

PTAk, FCA2, FCAk, summary.FCAk, plot.PTAk

Examples

```
 data(crimerate)
  cri.FCA2 <- FCA2(crimerate)
   summary(cri.FCA2)
    plot(cri.FCA2, mod = c(1,2), nb1 = 2, nb2 = 3) # unscaled
    plot(cri.FCA2, mod = c(1,2), nb1 = 2, nb2 = 3, coefi =
     list(c(0.130787,0.130787),c(0.104359,0.104359)) ) # symmetric-map biplot
   CTR(cri.FCA2, mod = 1, solnbs = 2:4)
   CTR(cri.FCA2, mod = 2, solnbs = 2:4)
 COS2(cri.FCA2, mod = 2, solnbs = 2:4)
 ##### useful fonctions
 ##selecting and sorting out dimensions positive and negative sides
 "ctrcos2" <-function(Ta, mod=1, dim=2, NegPos=TRUE, select=c("avg",12,"none"),nbdig=2,
  cos2min=333){
dim=c(dim, dim+1)
ctr=CTR(Ta,mod=mod,solnbs=dim);cos2=COS2(Ta,mod=mod,solnbs=dim)
val=round(Ta[[mod]]$v[dim,][1,],digits=nbdig)
oo=order(ctr[,1],decreasing=TRUE)
if(NegPos)oo=order(val,decreasing=TRUE)
out=cbind(ctr[oo,1],cos2[oo,1],val[oo])
colnames(out)=c("ctr","cos2",paste0("dim",dim[1]))
if(select[1]=="none") sout=0
if(select[1]=="avg") sout= 1000/length(val)
if(is.numeric(select[1])) sout=select[1]
return(out[ out[,1]>=sout | out[,2]>=cos2min, ])
}#ctrcos2

## plot ctr cos2
"plotctrcos2"<-function(sol,mod12=c(1,2),dim=2, ratio2avg=TRUE, col=c(1,2),pch=c("-","°"),
posi=c(2,3),reposi=TRUE, cos2min=333,select="avg",...){
### ctrcos2 ini Ta,mod=1,dim=2, NegPos=TRUE, select=c("avg",12,"none"), nbdig=2, cos2min=333
pre<-function(mod=1,soldim=2, ...){
diim=length(sol[[mod]]$v[1,])
ctrov= ctrcos2(sol,dim= soldim,mod=mod,...)
x=ctrov[,1]*sign(ctrov[,3])
if(ratio2avg)x=round(x/(1000/diim),2)
y=ctrov[,2]
lab=rownames(ctrov)
len=dim(ctrov)[1]
return(list("x"=x,"y"=y,"len"=len,"lab"=lab))
}
if(length(col)<length(mod12))col=rep(col,length(mod12))
if(length(pch)<length(mod12))pch=rep(pch,length(mod12))
x=NULL;y=NULL;coul=NULL;pchl=NULL;lab=NULL;poslab=NULL
for(m in mod12){
prep=pre(mod=m,soldim=dim,...)
x=c(x,prep$x);y=c(y,prep$y);
        coul=c(coul,rep(col[m],prep$len));pchl=c(pchl,rep(pch[m],prep$len))
```

```
repos=rep(posi[m],prep$len); if(reposi)repos=sample(1:4,prep$len, replace=TRUE)
lab=c(lab,prep$lab);poslab=c(poslab,repos)
}
summsol=summary(sol)

if(match("FCA2" ,class(sol),nomatch=0)>0) xlabe=paste0( "Global pct ",
round(summsol[dim,4],2), " FCA pct",round(summsol[dim,5],2)) else
xlabe=paste0( " local pxt",round(summsol[dim,4],2), " Global pct", round(summsol[dim,5],2))
dimi=paste0("dim",dim)
if(ratio2avg)ctrlab="CTR (signed ctr /(uniform ctr))" else ctrlab="CTR (signed)"
if(!is.null(cos2min))cos2lab=paste("COS2 (> ",cos2min,")")else cos2lab= "COS2"
plot(x,y,xlab=ctrlab, main=paste(dimi, xlabe ),ylab= cos2lab,col=coul,
             pch=pchl,ylim=c(min(y),1050),xlim=c(min(x-0.5),max(x+0.5)))
abline(v=0,col=4,lty=2)
abline(v=1,col=3,lty=2)
abline(v=-1,col=3,lty=2)
text(x,y,lab,pos=poslab, col=coul)
return(cbind(x,y,lab,coul,pchl,poslab))
}#plotctrcos2

ctrcos2(cri.FCA2,mod=1)
ctrcos2(cri.FCA2,mod=2)
 plotctrcos2(cri.FCA2)
```

---

datasets *data used for demo in SVDgen, PTA3*

Description

The `crimerate` dataset provides crime rates per 100,000 people in seven categories for each of the fifty states (USA) in 1977. The `timage12` dataset is an image from fMRI analysis (one brain slice), it is a *t*-statistic image over 12 subjects of the activation (verbal) parameter. The `Zone_climTUN` is an object of class Map representing montly (`12`) measurements in Tunisia of `10` climatic indicators. The grid of `2599` cells was stored previously as a shapefile and read using `read.shape`.

Usage

```
data(crimerate)
data(timage12)
data(Zone_climTUN)
```

Format

`crimerate` is a matrix of `50 x 7` for the `crimerate` data.

`timage12` is a matrix `91 x 109` for `timage12` data.

Source

`crimerate` comes from SAS. The `timage12` comes from FMRIB center, University of Oxford. The `Zone_climTUN` comes from WorldCLIM database 2000 see references along with description of the indicators in Leibovici et al.(2007).

References

Leibovici D, Quillevere G, Desconnets JC (2007). A Method to Classify Ecoclimatic Arid and Semi-Arid Zones in Circum-Saharan Africa Using Monthly Dynamics of Multiple Indicators. IEEE Transactions on Geoscience and Remote Sensing, 45(12), 4000-4007.

`FCA2` *Correspondence Analysis for 2-way tables*

Description

Performs a particular `SVDgen` data as a ratio Observed/Expected under complete independence with metrics as margins of the contingency table (in frequencies).

Usage

```
FCA2(X, nbdim =NULL, minpct = 0.01, smoothing = FALSE,
        smoo = rep(list(function(u) ksmooth(1:length(u), u, kernel = "normal",
       bandwidth = 3, x.points = (1:length(u)))$y), length(dim(X))),
     verbose = getOption("verbose"), file = NULL, modesnam = NULL,
   addedcomment = "", chi2 = FALSE, E = NULL, ...)
```

Arguments

`X`	a matrix table of positive values
`nbdim`	a number of dimension to retain, if `NULL` the default value of maximum possible number of dimensions is kept
`minpct`	numerical 0-100 to control of computation of future solutions at this level and below
`smoothing`	see `SVDgen`
`smoo`	see `SVDgen`
`verbose`	control printing
`file`	output printed at the prompt if `NULL`, or printed in the given '`file`'
`modesnam`	character vector of the names of the modes, if `NULL` `"mo 1"` ...`"mo k"`
`addedcomment`	character string printed if `printt` after the title of the analysis
`chi2`	print the chi2 information when computing margins in `FCAmet`
`E`	if not `NULL` is a matrix with the same dimensions as X with the same margins
`...`	any other arguments passed to SVDGen or other functions

Details

Gives the SVD-2modes decomposition of the $1 + \chi^2/N$ of the contingency table of full count $N = \sum X_{ij}$, i.e. complete independence + lack of independence (including marginal independences) as shown for example in Lancaster(1951)(see reference in Leibovici(1993 or 2000)). Noting $P = X/N$, a SVD of the (3)-uple is done, that is :

$$((D_I^{-1} \otimes D_J^{-1})..P, \quad D_I, \quad D_J)$$

where the metrics are diagonals of the corresponding margins. For full description of arguments see PTAk. If E is not NULL an FCAk-modes relatively to a model is done (see Escoufier(1985) and therin reference Escofier(1984) for a 2-way derivation), e.g. for a three way contingency table $k = 3$ the *4*-tuple data and metrics is:

$$((D_I^{-1} \otimes D_J^{-1} \otimes D_K^{-1})(P - E), \quad D_I, \quad D_J, \quad D_K)$$

If E was the complete independence (product of the margins) then this would give an AFCk but without looking at the marginal dependencies (i.e. for a three way table no two-ways lack of independence are looked for).

Value

a FCA2 (inherits FCAk and PTAk) object

Author(s)

Didier G. Leibovici

References

Escoufier Y (1985) *L'Analyse des correspondances : ses propriétés et ses extensions*. ISI 45th session Amsterdam.

Leibovici D(1993) *Facteurs à Mesures Répétées et Analyses Factorielles : applications à un suivi Epidémiologique*. Université de Montpellier II. PhD Thesis in Mathématiques et Applications (Biostatistiques).

Leibovici D (2000) *Multiway Multidimensional Analysis for Pharmaco-EEG Studies*.http://www.fmrib.ox.ac.uk/analysis/techrep/tr00dl2/tr00dl2.pdf

Leibovici DG (2010) *Spatio-temporal Multiway Decomposition using Principal Tensor Analysis on k-modes:the R package* **PTAk**. Journal of Statistical Software, 34(10), 1-34. http://www.jstatsoft.org/v34/i10/

See Also

PTAk, FCAmet, summary.FCAk

Examples

```
data(crimerate)
cri.FCA2 <- FCA2(crimerate)
summary(cri.FCA2)
```

```
 plot(cri.FCA2, mod = c(1,2), nb1 = 2, nb2 = 3) # unscaled
 plot(cri.FCA2, mod = c(1,2), nb1 = 2, nb2 = 3, coefi =
  list(c(0.130787,0.130787),c(0.104359,0.104359)) )# symmetric-map biplot
CTR(cri.FCA2, mod = 1, solnbs = 2:4)
CTR(cri.FCA2, mod = 2, solnbs = 2:4)
COS2(cri.FCA2, mod = 2, solnbs = 2:4)
```

FCAk *Generalisation of Correspondence Analysis for k-way tables*

Description

Performs a particular PTAk data as a ratio Observed/Expected under complete independence with metrics as margins of the multiple contingency table (in frequencies).

Usage

```
FCAk(X,nbPT=3,nbPT2=1,minpct=0.01,
                smoothing=FALSE,smoo=rep(list(
                         function(u)ksmooth(1:length(u),u,kernel="normal",
                         bandwidth=3,x.points=(1:length(u)))$y),length(dim(X))),
                       verbose=getOption("verbose"),file=NULL,
                         modesnam=NULL,addedcomment="",chi2=TRUE,E=NULL, ...)
```

Arguments

X	a multiple contingency table (array) of order *k*
nbPT	a number or a vector of dimension *(k-2)*
nbPT2	if 0 no 2-modes solutions will be computed, 1 =all, >1 otherwise
minpct	numerical 0-100 to control of computation of future solutions at this level and below
smoothing	see SVDgen
smoo	see SVDgen
verbose	control printing
file	output printed at the prompt if NULL, or printed in the given 'file'
modesnam	character vector of the names of the modes, if NULL "mo 1" ..."mo k"
addedcomment	character string printed if printt after the title of the analysis
chi2	print the chi2 information when computing margins in FCAmet
E	if not NULL is an array with the same dimensions as X
...	any other arguments passed to SVDGen or other functions

Details

Gives the SVD-*k*modes decomposition of the $1 + \chi^2/N$ of the multiple contingency table of full count $N = \sum X_{ijk...}$, i.e. complete independence + lack of independence (including marginal independences) as shown for example in Lancaster(1951)(see reference in Leibovici(2000)). Noting $P = X/N$, a `PTAk` of the $(k+1)$-uple is done, e.g. for a three way contingency table $k = 3$ the *4*-uple data and metrics is:

$$((D_I^{-1} \otimes D_J^{-1} \otimes D_K^{-1})P, \quad D_I, \quad D_J, \quad D_K)$$

where the metrics are diagonals of the corresponding margins. For full description of arguments see `PTAk`. If `E` is not `NULL` an FCAk-modes relatively to a model is done (see Escoufier(1985) and therin reference Escofier(1984) for a 2-way derivation), e.g. for a three way contingency table $k = 3$ the *4*-tuple data and metrics is:

$$((D_I^{-1} \otimes D_J^{-1} \otimes D_K^{-1})(P - E), \quad D_I, \quad D_J, \quad D_K)$$

If `E` was the complete independence (product of the margins) then this would give an `AFCk` but without looking at the marginal dependencies (i.e. for a three way table no two-ways lack of independence are looked for).

Value

a `FCAk` (inherits `PTAk`) object

Author(s)

Didier G. Leibovici

References

Escoufier Y (1985) *L'Analyse des correspondances : ses propri<e9>t<e9>s et ses extensions*. ISI 45th session Amsterdam.

Leibovici D(1993) *Facteurs <e0> Mesures R<e9>p<e9>t<e9>es et Analyses Factorielles : applications <e0> un suivi <e9>pid<e9>miologique*. Universit<e9> de Montpellier II. PhD Thesis in Math<e9>matiques et Applications (Biostatistiques).

Leibovici D (2000) *Multiway Multidimensional Analysis for Pharmaco-EEG Studies*.http://www.fmrib.ox.ac.uk/analysis/techrep/tr00dl2/tr00dl2.pdf

Leibovici D (2008) *Spatio-temporal Multiway Decomposition using Principal Tensor Analysis on k-modes:the R package* **PTAk** . to be submitted soon at Journal of Statisticcal Software.

See Also

`PTAk`, `FCAmet`, `summary.FCAk`

Examples

```
# try the demo
  # demo.FCAk()
```

`FCAmet` *Tool used in Generalisation of Correspondence Analysis for k-way tables*

Description

Computes the ratio Observed/Expected under complete independence with margins of the multiple contingency table (in frequencies) and gives `chi2` statistic of lack of complete independence.

Usage

```
FCAmet(X,chi2=FALSE,E=NULL,No0margins=TRUE)
```

Arguments

`X`	a multiple contingency table (array) of order *k*
`chi2`	if `TRUE` prints the chi2 statistic information
`E`	if not `NULL` represent a model which would be used for an `FCAk` relatively to a model
`No0margins`	if TRUE, prevents zero margins in replacing cells involved by the min of the non-zero margins /nb of zero cells

Value

a list with

`data`	an array `(X/count (-E))/Indepen` where `Indepen` is the array obtained from he products of the margins
`met`	a list wherein each entry is the vector of the corresponding margins i.e. `apply(X,i,sum)/count`
`count`	is the total sum `sum(X)`.

Note

The statistics and metrics do not depend on `E`. The statistic given measure only the lack of independence.

Author(s)

Didier G. Leibovici

See Also

`FCAk`

howtoPTAk — *howto for Principal Tensors Analysis of a k-modes Tensor*

Description

A mini guide to handle PTAk model decomposition

Usage

```
howtoPTAk()
```

Details

The PTAk decomposition aims at building an approximation of a given multiway data, represented as a tensor, based on a variance criterion. This approximation is given by a set of rank one tensors, orthogonal to each other, in a nested algorithm process and so controlling the level of approximation by the amount of variability extracted and represented by the sum of squares of the singular values (associated to the rank one tensors). In that respect it offers a way of generalising PCA to tensors of order greater than 2.

The reference in JSS provides details about preparing a dataset and running a general PTAk and particularities for spatio-temporal data.

The license is GPL-3, support can be provided via http://c3s2i.free.fr, donations via Paypal to c3s2i@free.fr are welcome.

Author(s)

Didier G. Leibovici

References

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k.* Linear Algebra and its Applications, 269:307-329 Leibovici D.G. (accepted in 2009) *Spatio-temporal Multiway Decompositions using Principal Tensor Analysis on k-modes: the R-package PTAk.* Journal of Statistical Software, vol x issue x, http://www.jstatsoft.org

See Also

PTA3, PTAk ,FCAk

INITIA *Initialisation used in SINGVA*

Description

Gives the first Tucker1 components of a given tensor.

Usage

```
INITIA(X,modesnam=NULL,method="svds",dim=1,...)
```

Arguments

`X`	a tensor (as an array) of order *k*
`modesnam`	a character vector of the names of the modes
`method`	uses either the inbuilt SVD `method="svd"` or a power algorithm giving only the first `method="Presvd"` or any other function given applying to the column space of a matrix and returning a list with `v` (in columns vectors as in `svd`) and `d`. The method `method="svds"` performs alike `method="svd"` but on a sum of tables instead of the Tucker1 approach.
`dim`	default 1 in each space otherwise specify the number of dimensions e.g. `c(2,3..,2)` (with `"Presvd"` dim is obviously 1)
`...`	extra arguments of the method `method`: the first argument is fixed (see details).

Details

Computes the first (or `dim`) right singular vector (or other summaries) for every representation of the tensor as a matrix with `dim(X)[i]` columns, `i=1...k`.

Value

a list (of length *k*) of lists with arguments:

`v`	the singular vectors in rows
`modesnam`	a character object naming the mode, `"m i"` otherwise
`n`	labels of mode `i` entries as given in `dimnames` of the data, can be `NULL`
`d`	the corresponding first singular values

Note

The collection these eigenvectors, is known as the Tucker1 solution or initialisation related to PCA-3modes or PCA-*n*modes models. If a function is given it may include `dim` as argument.

Author(s)

Didier G. Leibovici

References

Kroonenberg P.M (1983) *Three-mode Principal Component Analysis: Theory and Applications*. DSWO Press, Leiden.

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

See Also

SINGVA, PTAk

PCAn *Principal Component Analysis on n modes*

Description

Performs the Tucker*n* model using a space version of RPVSCC (SINGVA).

Usage

```
PCAn(X,dim=c(2,2,2,3),test=1E-12,Maxiter=400,
             smoothing=FALSE,smoo=list(NA),
               verbose=getOption("verbose"),file=NULL,
                 modesnam=NULL,addedcomment="")
```

Arguments

X	a tensor (as an array) of order *k*, if non-identity metrics are used X is a list with data as the array and met a list of metrics
dim	a vector of numbers specifying the dimensions in each space
test	control of convergence
Maxiter	maximum number of iterations allowed for convergence
smoothing	see SVDgen
smoo	see PTA3
verbose	control printing
file	output printed at the prompt if NULL, or printed in the given 'file'
modesnam	character vector of the names of the modes, if NULL "mo 1" ..."mo k"
addedcomment	character string printed after the title of the analysis

Details

Looking for the best rank-one tensor approximation (LS) the three methods described in the package are equivalent. If the number of tensors looked for is greater then one the methods differs: PTA-*k*modes will "look" for "best" approximation according to the *orthogonal rank* (*i.e.* the rank-one tensors are orthogonal), PCA-*k*modes will look for best approximation according to the *space ranks* (*i.e.* the rank of every bilinear form, that is the number of components in each space), PARAFAC/CANDECOMP will look for best approximation according to the *rank* (*i.e.* the rank-one tensors are not necessarily orthogonal). For the sake of comparisons the PARAFAC/CANDECOMP method and the PCA-*n*modes are also in the package but complete functionnality of the use these methods and more complete packages may be fetched at the www site quoted below.
Recent work from Tamara G Kolda showed on an example that *orthogonal rank* decompositions are not necesseraly nested. This makes PTA-*k*modes a model with nested decompositions not giving the exact *orthogonal rank*. So PTA-*k*modes will look for best approximation according to orthogonal tensors in a nested approximmation process.

Value

a `PCAn` (inherits `PTAk`) object

Note

The use of metrics (diagonal or not) and smoothing extend flexibility of analysis.

Author(s)

Didier G. Leibovici

References

Caroll J.D and Chang J.J (1970) *Analysis of individual differences in multidimensional scaling via n-way generalization of "Eckart-Young" decomposition*. Psychometrika 35,283-319.

Harshman R.A (1970) *Foundations of the PARAFAC procedure: models and conditions for "an explanatory" multi-mode factor analysis*. UCLA Working Papers in Phonetics, 16,1-84.

Kroonenberg P (1983) *Three-mode Principal Component Analysis: Theory and Applications*. DSWO press. Leiden.(related references in `http://three-mode.leidenuniv.nl/`)

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

Kolda T.G (2003) *A Counterexample to the Possibility of an Extension of the Eckart-Young Low-Rank Approximation Theorem for the Orthogonal Rank Tensor Decomposition*. SIAM J. Matrix Analysis, 24(2):763-767, Jan. 2003.

plot.PTAk *Plot a PTAk object*

Description

Screeplot of singular values or superposed plot of modes for one or two components (1 dimensional scatterplot with spread labels or scatterplot on two dimensions).

Usage

```
## S3 method for class 'PTAk'
plot(x, labels = TRUE, mod = 1, nb1 = 1, nb2 = NULL,
     coefi = list(NULL, NULL), xylab = TRUE, ppch = (1:length(solution)),
     lengthlabels = 2, scree = FALSE, ordered = TRUE,
    nbvs = 40, RiskJack = NULL, method = "",ZoomInOut=NULL, Zlabels=NULL, Zcol=NULL,
     poslab=c(2,1,3,3), ...)
RiskJackplot(x, nbvs = 1:20, mod = NULL, max = NULL, rescaled=TRUE, ...)
```

Arguments

x	an object inheriting from class PTAk, representing a generalised singular value decomposition
labels	logical if TRUE plots the labels given in solution[[mod]]["n"]
mod	vectors of the modes numbers to be plotted
nb1	number identifying the Principal Tensor to display on the vertical axe, can be checked using summary.PTAk
nb2	as nb1 to be displayed on the horizontal axe, if NULL the horizontal axe will be used as Index (see plot.default)
coefi	coefficients to multiply components for all modes (not just the one in mod) for rescaling or changing signs purposes; each element of the list correspond to nb1 and nb2 and are vectors of dimentions the tensor order
xylab	logical to display axes labels
ppch	a vector of length at least length(mod) used for pch=
lengthlabels	a number or a vector of numbers of characters in labels to be used for display
scree	logical to display a screeplot of squared singular values as percent of total variation
ordered	logical used when displaying the screeplot with sorted values (TRUE) or the order is given by output listing from summary.PTAk
nbvs	a maximum number of singular values to display on the screeplot or a vector of ranks
max	is the number of singular values to be considered as giving the perfect fit, NULL is the max possible in x

rescaled	boolean to rescale the y axis to 0-100
RiskJack	if not NULL is a integer, scree is TRUE and ordered is TRUE, plots on top of the scree plot a Risk plot with nbvs = 1:RiskJack. It is possible to use directly the function RiskJackplot: the default maximum dimension (argument max) is length(solution[[k]][["d"]]).
method	default is "", a value "FCA" is to be used only if solution is after an FCA with SVDgen
ZoomInOut	list used as [[1]] for xlim and [[2]] for ylim in xy-plots instead of max and min range
Zlabels	used as labels instead of x[[mod]]$n, it is a list with the same length as all modes. For example on 3 modes changing the labels of the second mode only will have to set Zlabels=list(NULL,rep("a",length(x[[2]]$n)), NULL)
Zcol	list of vectors of colours for Zlabels
poslab	integer or vector for 'pos' parameter, position of labels
...	plot arguments can be passed (except xlim, ylim, ylab,pch,xaxt for component plot, and xlab, ylab for screeplot). For example to have normed plot one can use asp=1

Details

Plot components of one or two Principal Tensors, modes are superposed if more than one is asked, or gives a screeplot. As it is using plot.default at some point some added features can be used in the ... part, especially xlab= may be useful when nb2=NULL. Plots are superposed as they correspond to the same Principal Tensor and so this gives insight to interpretation of it, but careful is recommended as only overall interpretation, once the Principal Tensor has been rebuilt mentally (*i.e.* product of signs ...) to work out oppositions or associations. The risk plot on top of a screeplot is an approximation of the Jacknife estimate of the MSE in the choice of number of dimensions (see Besse et al.(1997)).

Note

This function is used all for FCAk, and CANDPARA, PCAn objjects notheless for this last object other interesting plots known as jointplots have not been implemented.

Author(s)

Didier G. Leibovici

References

Besse, P Cardot, H and Ferraty, F (1997) *Simultaneous non-parametric regressions of unbalanced longitudinal data*. Computational Statistics and Data Analysis, 24:255-270.

Leibovici D (2000) *Multiway Multidimensional Analysis for Pharmaco-EEG Studies*.(submitted) http://c3s2i.free.fr/cv/recentpub.html

See Also

PTAk, PTA3, FCAk,SVDgen

Examples

```
#  see the demo function   source(paste(R.home(),"/ library/PTAk/demo/PTA3.R",sep=""));
# or    source(paste(R.home(),"/ library/PTAk/demo/PTAk.R",sep=""));

 # demo.PTA3()
```

---

preprocessings   *Few useful functions for preprocessing arrays*

Description

Choices of centering or detrending and scaling are important preprocessings for multiway analysis.

Usage

```
Multcent(dat,bi=c(1,2),by=3,
           centre=mean,
            centrebyBA=c(TRUE,FALSE),scalebyBA=c(TRUE,FALSE))
IterMV(n=10,dat,Mm=c(1,3),Vm=c(2,3),
                fFUN=mean,usetren=FALSE,
                   tren=function(x)smooth.spline(as.vector(x),df=5)$y,
                    rsd=TRUE)
Detren(dat,Mm=c(1,3),rsd=TRUE,
         tren=function(x)smooth.spline(as.vector(x),df=5)$y )
Susan1D(y,x=NULL,sigmak=NULL,sigmat=NULL,
          ker=list(function(u)return(exp(-0.5*u**2))))
```

Arguments

dat	array
bi	vector defining the "centering, bicentering or multi-centering" one wants to operate crossed with by
by	number or vector defining the entries used "with" in the other operations
centre	function used as FUN in applying "multi-centering"
centrebyBA	a bolean vector for "centering" with centre Before and After according to by
scalebyBA	idem as centrebyBA, for scaling operation
n	number of iterations between "centering" and scaling
Mm	margins to performs Detren or fFUN on
Vm	margins to scale
fFUN	function to use as FUN if usetren is FALSE
usetren	logical, to use Detren
tren	function to use in Detren

rsd	logical passed into Detren (only) to detrend or not
y	vector (length n)
x	vector of same length, if NULL it is 1:n
sigmak	parameter related to kernel bandwidth with y values (default is 1/2*range
sigmat	parameter related to kernel bandwidth with x values (default value is 8*n^{-1/5}, with a minimum number of neigbours set as one apart)
ker	a list of two kernels list("t"=function "k"=function) for each weightings (if only one given it is used for both)

Details

Multcent performs in order "centering" by by; "multicentering" for every bi with by; then scale (standard deviation) to one by by.

IterMV performs an iterative "detrending" and scaling according to te margins defined (see Leibovici(2000) and references in it).

Detren detrends (or smooths if rsd is FALSE) the data accoding to th margins given.

Susan1D performs a non-linear kernel smoothing of y against x (both reordered in the function according to orders of x) with an usual kernel (t) as for kernel regression and a kernel (t) for the values of y (the product of the kernels constitutes the non-linear weightings. This function is adapted from SUSAN algorithm (see references).

Author(s)

Didier G. Leibovici

References

Smith S.M. and J.M. Brady (1997) *SUSAN - a new approach to low level image processing*. International Journal of Computer Vision, 23(1):45-78, May 1997.

PROJOT *Orthogonal Tensor projection*

Description

Orthogonal-tensoriel projection of a tensor according to a rank-1 tensor, or a to bigger structure defined by kronecker product of matrices.

Usage

```
PROJOT(X,solu,numo=1,bortho=TRUE,Ortho=TRUE,metrics=NULL)
```

Arguments

`X`	a tensor(as an array) of any order
`solu`	an object like a `solutions.PTAk` object with at least `v`
`numo`	a vector of numbers or a list of vectors (length the order of the tensor) identifying for each space the structure to project onto, if NULL for a specific space then no projection is done for this space
`bortho`	list of logicals saying if the structures are othogonal or not.
`Ortho`	list of logicals telling the projectors on each space to be on the structure or on its orthogonal.
`metrics`	NULL or list of metrics (either diagonal or not) for each entry of `X`

Details

This function computes the *tensorial orthogonal projection* of `X` onto the *tensorial structure* defined by `solu` and `numo`. For each space (involved in the tensorial product where from `X` belongs), one defined the projector onto `solu[[i]]$v[numo,]` (or on its orthogonal if `Ortho[[i]]==TRUE`), then the result is the image of `X` by the tensorial product of the projectors, i.e.

$$(P_{S1} \otimes P_{S2} \otimes \ldots \otimes P_{Sk})(X)$$

.

Value

A tensor with dimensions as `X`

Note

For PTA-*k*modes the projection used is only on rank-one tensors (Principal Tensors), *i.e.* `numo` is a number. The code here can be used for any structure (on each spaces) and constitutes the projector onto a tensorial structure, and can define the PTAIV-*k*modes (PTAk on Instrumental Variables Leibovici(1993). (see other references for tensorial product of linear operators in Leibovici(2000) *e.g.* Dauxois et al.(1994))

Author(s)

Didier G. Leibovici <GeotRycs@gmail.com>

References

Leibovici D(1993) *Facteurs <e0> Mesures R<e9>p<e9>t<e9>es et Analyses Factorielles : applications <e0> un suivi <e9>pid<e9>miologique*. Universit<e9> de Montpellier II. PhD Thesis in Math<e9>matiques et Applications (Biostatistiques).

Leibovici D (2000) *Multiway Multidimensional Analysis for Pharmaco-EEG Studies*. `http://www.fmrib.ox.ac.uk/analysis/techrep/tr00dl2/tr00dl2.pdf`

See Also

`PTAk`

Examples

```
don <- array(1:360,c(5,4,6,3))
 don <- don + rnorm(360,10,2)

 ones <- list(list(v=rep(1,5)),list(v=rep(1,4)),list(v=rep(1,6)),list(v=rep(1,3)))
 donfc <- PROJOT(don,ones)

 apply(donfc,c(2,3,4),mean)
 apply(donfc,c(1),mean)

  # implementation de PTAIVk with obvious settings
 PTAIVk <- function(X,STruct,...)
       {X <- PROJOT(X$data,STruct,numo=Struct[[1]]$numo,Ortho=Struct[[1]]$Ortho,metrics=X$met)
         PTAk(X,...)
         }
```

PTA3 *Principal Tensor Analysis on 3 modes*

Description

Performs a truncated SVD-*3*modes analysis with or without specific metrics, penalised or not.

Usage

```
PTA3(X,nbPT=2,nbPT2=1,
                  smoothing=FALSE,
                  smoo=list(function(u)ksmooth(1:length(u),u,kernel="normal",
                             bandwidth=4,x.points=(1:length(u)))$y,
                             function(u)smooth.spline(u,df=3)$y,
                             NA),
                  minpct=0.1,verbose=getOption("verbose"),file=NULL,
                  modesnam=NULL,addedcomment="", ...)
```

Arguments

X	a tensor (as an array) of order *3*, if non-identity metrics are used `X` is a list with `data` as the array and `met` a list of metrics
nbPT	a number specifying the number of *3*modes Principal Tensors requested
nbPT2	if 0 no *2*-modes solutions will be computed, 1 =all, >1 otherwise
smoothing	logical to consider smoothing or not
smoo	a list of length 3 with lists of functions operating on vectors component for the appropriate dimension (see details)

minpct	numerical 0-100 to control of computation of future solutions at this level and below
verbose	control printing
file	output printed at the prompt if NULL, or printed in the given 'file'
modesnam	character vector of the names of the modes, if NULL "mo 1" ..."mo k"
addedcomment	character string printed after the title of the analysis
...	any other arguments passed to SVDGen or other functions

Details

According to the decomposition described in Leibovici(1993) and Leibovici and Sabatier(1998) the function gives a generalisation of the SVD (2 modes) to *3* modes. It is the same algorithm used for PTAk but simpler as the recursivity implied by the *k* modes analysis is reduced only to one level *i.e* for every *3*-modes Principal Tensors, 3 SVD are performed for every contracted product with one the three components of the *3*-modes Principal Tensors (see APSOLU3, PTAk).

Recent work from Tamara G Kolda showed on an example that *orthogonal rank* decompositions are not necesseraly nested. This makes PTA-*3*modes a model with nested decompositions not giving the exact *orthogonal rank*. So PTA-*3*modes will look for best approximation according to orthogonal tensors in a nested approximmation process. PTA3 decompositions is "a" generalisation of SVD but not the ...
With the smoothing option smoo contain a list of (lists) of functions to apply on vectors of component (within the algorithm, see SVDgen). For a given dimension (1,2,or 3) a list of functions is given. If this list consists only of one function (no list needed) this function will be used at any level all the time : if one want to smooth only for the first Principal Tensor, put list(function,NA). Now you start to understand this list will have a maximum length of nbPT and the corresponding function will be used for the corresponding *3*mode Principal Tensor. To smooth differently the associated solutions one have to put another level of nested lists otherwise the function given at the *3*mode level will be used for all. These rules are te same for PTAk.

Value

a PTAk object

Note

The use of metrics (diagonal or not) allows flexibility of analysis like in *2* modes *e.g.* correspondence analysis, discriminant analysis, robust analysis. Smoothing option extends the analysis towards functional data analysis, and or outliers "protection" is theoretically valid for tensors belonging to a tensor product of separable Hilbert spaces (*e.g.* Sobolev spaces) (see references in PTAk, SVDgen).

Author(s)

Didier G. Leibovici

References

Leibovici D(1993) *Facteurs <e0> Mesures R<e9>p<e9>t<e9>es et Analyses Factorielles : applications <e0> un suivi <e9>pid<e9>miologique*. Universit<e9> de Montpellier II. PhD Thesis in Math<e9>matiques et Applications (Biostatistiques).

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

Kolda T.G (2003) *A Counterexample to the Possibility of an Extension of the Eckart-Young Low-Rank Approximation Theorem for the Orthogonal Rank Tensor Decomposition*. SIAM J. Matrix Analysis, 24(2):763-767, Jan. 2003.

See Also

SVDgen, FCAk, PTAk, summary.PTAk

Examples

```
 # example using Zone_climTUN dataset
 #
# library(maptools)
# library(RColorBrewer)
# Yl=brewer.pal(11,"PuOr")
# data(Zone_climTUN)
## in fact a modified version of plot.Map was used
# plot(Zone_climTUN,ol=NA,auxvar=Zone_climTUN$att.data$PREC_OCTO)
##indicators 84 +3 to repeat
# Zone_clim<-Zone_climTUN$att.data[,c(2:13,15:26,28:39,42:53,57:80,83:95,55:56)]
# Zot <-Zone_clim[,85:87] ;temp <-colnames(Zot)
# Zot <- as.matrix(Zot)%x%t(as.matrix(rep(1,12)))
# colnames(Zot) <-c(paste(rep(temp [1],12),1:12),paste(rep(temp [2],12),1:12),
#  paste(rep(temp [3],12),1:12))
# Zone_clim <-cbind(Zone_clim[,1:84],Zot)

# Zone3w <- array(as.vector(as.matrix(Zone_clim)),c(2599,12,10))
## preprocessing
#Zone3w<-Multcent(dat=Zone3w,bi=NULL,by=3,centre=mean,
# centrebyBA=c(TRUE,FALSE),scalebyBA=c(TRUE,FALSE))
# Zone3w.PTA3<-PTA3(Zone3w,nbPT=3,nbPT2=3)
## summary and plot
# summary(Zone3w.PTA3)
#plot(Zone3w.PTA3,mod=c(2,3),nb1=1,nb2=11,lengthlabels=5,coefi=list(c(1,1,1),c(1,-1,-1)))
#plot(Zone_climTUN,ol=NA,auxvar=Zone3w.PTA3[[1]]$v[1,],nclass=30)
#plot(Zone_climTUN,ol=NA,auxvar=Zone3w.PTA3[[1]]$v[11,],nclass=30)

 ##############
 cat(" A little fun using iris3 and matching randomly 15 for each iris sample!","\n")
 cat("   then performing a PTA-3modes.  If many draws are done, plots")
 cat("   show the stability of the first and third Principal Tensors.","\n")
 cat("iris3 is centered and reduced beforehand for each original variables.","\n")
#  demo function
# source(paste(R.home(),"/library/PTAk/demo/PTA3.R",sep=""))
```

```
# demo.PTA3(bootn=10,show=5,openX11s=FALSE)
```

PTAk *Principal Tensor Analysis on k modes*

Description

Performs a truncated SVD-*k*modes analysis with or without specific metrics, penalised or not.

Usage

```
PTAk(X,nbPT=2,nbPT2=1,minpct=0.1,
               smoothing=FALSE,
                  smoo=list(NA),
                   verbose=getOption("verbose"),file=NULL,
                    modesnam=NULL,addedcomment="", ...)
```

Arguments

`X`	a tensor (as an array) of order *k*, if non-identity metrics are used `X` is a list with `data` as the array and `met` a list of metrics
`nbPT`	integer vector of length *(k-2)* specifying the maximum number of Principal Tensors requested for the *(3,...,k-1, k)* modes levels (see details), if it is not a vector every levels would have the same given nbPT value
`nbPT2`	if 0 no 2-modes solutions will be computed, 1 =all, >1 otherwise
`minpct`	numerical 0-100 to control of computation of future solutions at this level and below
`smoothing`	see `PTA3`, `SVDgen`
`smoo`	see `PTA3`
`verbose`	control printing
`file`	output printed at the prompt if `NULL`, or printed in the given '`file`'
`modesnam`	character vector of the names of the modes, if `NULL` `mo 1` ...`mo k`
`addedcomment`	character string printed if `printt` after the title of the analysis
`...`	any other arguments passed to other functions

Details

According to the decomposition described in Leibovici(1993) and Leibovici and Sabatier(1998) the function gives a generalisation of the SVD (2 modes) to *k* modes. The algorithm is recursive, calling `APSOLUk` which calls `PTAk` for *(k-1)*. `nbPT`, `nbPT2` and `minpct` control the number of Principal Tensors desired. For example `nbPT=c(2,4,3)` means a tensor of order *5* is analysed, the maximum number of 5-modes PT is set to *3*, for *each of them* one sets a maximum of *4* associated *4-modes* (for each of the five components), for *each of these later* a maximum of *2* associated *3-modes* PT is asked

(for each of the four components). Then nbPT2 complete for *2-modes* associated or not. Overall minpct controls to carry on the algorithm at any level and lower, *i.e.* stops if $100(vs^2/ssx) < minpct$ (where vs is the singular value, and *ssx* is the total sum of squares of the tensor X or the "metric transformed" X). Putting a 0 at a given level in nbPT obviously automatically puts 0 in nbPT at lower levels. Putting high values in nbPT allows control only on minpct helping to reach the full decomposition. All these controls allow to truncate the full decomposition in a level-controlled fashion. Notice the full decomposition always contains any possible choice of truncation, *i.e.* the solutions are not dependant on the truncation scheme (Generalised Eckart-Young Theorem).
Recent work from Tamara G Kolda showed on an example that *orthogonal rank* decompositions are not necesseraly nested. This makes PTA-*k*modes a model with nested decompositions not giving the exact *orthogonal rank*. So PTA-*k*modes will look for best approximation according to orthogonal tensors in a nested approximmation process.

Value

a PTAk object which consist of a list of lists. Each mode has a list in which is listed:

`$v`	matrix of components for the given mode
`$iter`	vector of iterations numbers where maximum was reach
`$test`	vector of test values at maximum
`$modesnam`	name of the mode
`$v`	matrix of components for the given mode

The last mode list has also some additional information on the analysis done:

`$d`	vector of singular values
`$pct`	percentage of sum of squares for each quared singular value
`$ssX`	vector of local sum of squares *i.e.* of the current tensor with the rescursive algorithm
`$vsnam`	vector of names given to the singular value according to a recursive data dependent scheme
`$datanam`	data reference
`$method`	call applied: could be PTAk or CANDPARA or PCAn or even SVDgen, with parameters choices
`$addedcomment`	the addedcomment (repeated) given in the call

You will notice that methods other than PTAk may not have all list elements but the essential ones such as: $v, $d, $ssX, and may also have additional ones like $coremat for PCAn (the core array).

Note

The use of metrics (diagonal or not) allows flexibility of analysis like in *2* modes *e.g.* correspondence analysis, discriminant analysis, robust analysis. Smoothing option extending the analysis towards functional data analysis is theoretically valid for Principal Tensors belonging to a tensor product of separable Hilbert spaces (*e.g.* Sobolev spaces) see Leibovici and El Maach (1997).

Author(s)

Didier G. Leibovici

References

Leibovici D(1993) *Facteurs <e0> Mesures R<e9>p<e9>t<e9>es et Analyses Factorielles : applications <e0> un suivi <e9>pid<e9>miologique*. Universit<e9> de Montpellier II. PhD Thesis in Math<e9>matiques et Applications (Biostatistiques).

Leibovici D and El Maache H (1997) *Une d<e9>composition en Valeurs Singuli<e8>res d'un <e9>l<e9>ment d'un produit Tensoriel de k espaces de Hilbert S<e9>parables*. Compte Rendus de l'Acad<e9>mie des Sciences tome 325, s<e9>rie I, Statistiques (Statistics) & Probabilit<e9>s (Probability Theory): 779-782.

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

Leibovici D (2008) *Spatio-temporal Multiway Decomposition using Principal Tensor Analysis on k-modes:the R package* **PTAk** . to be submitted soon at Journal of Statisticcal Software.

Leibovici D (2008) *A Simple Penalised algorithm for SVD and Multiway functional methods*. (to be submitted)

Kolda T.G (2003) *A Counterexample to the Possibility of an Extension of the Eckart-Young Low-Rank Approximation Theorem for the Orthogonal Rank Tensor Decomposition*. SIAM J. Matrix Analysis, 24(2):763-767, Jan. 2003.

See Also

REBUILD, FCAk, PTA3 summary.PTAk

Examples

```
  # don <- array((1:3)%x%rnorm(6*4)%x%(1:10),c(10,4,6,3))

don <- array(1:360,c(5,4,6,3))
 don <- don + rnorm(360,1,2)

 dimnames(don) <- list(paste("s",1:5,sep=""),paste("T",1:4,sep=""),
          paste("t",1:6,sep=""),c("young","normal","old"))
   # hypothetic data on learning curve at different age and period of year

 ones <-list(list(v=rep(1,5)),list(v=rep(1,4)),list(v=rep(1,6)),list(v=rep(1,3)))

 don <- PROJOT(don,ones)
 don.sol <- PTAk(don,nbPT=1,nbPT2=2,minpct=0.01,
              verbose=TRUE,
               modesnam=c("Subjects","Trimester","Time","Age"),
                addedcomment="centered on each mode")

don.sol[[1]] # mode Subjects results and components
don.sol[[2]] # mode Trimester results and components
```

```
don.sol[[3]] # mode Time results and components
don.sol[[4]] # mode Age results and  components with additional information on the call

 summary(don.sol,testvar=2)
  plot(don.sol,mod=c(1,2,3,4),nb1=1,nb2=NULL,
     xlab="Subjects/Trimester/Time/Age",main="Best rank-one approx" )
  plot(don.sol,mod=c(1,2,3,4),nb1=4,nb2=NULL,
      xlab="Subjects/Trimester/Time/Age",main="Associated to Subject vs1111")

#  demo function
 # demo.PTAk()
```

`PTAk-internal` *Internal PTAk functions*

Description

Internal PTAk functions

Usage

```
Ginv(A)
PPMA(X,test=1E-10,pena=list(function(u)ksmooth(1:length(u),u,kernel="normal",
                       bandwidth=3,x.points=(1:length(u)))$y
                       ,NA) ,ini=mean,vsmin=1E-20,Maxiter=2000, ...)
Powmat(A,pw,eltw=FALSE)
RaoProd(A,B)
REBUILDPCAn(solu)
RESUM(solb,sola=NULL,numass=NULL,verbose=getOption("verbose"),file=NULL
                          ,summary=FALSE,testvar=0.1,with=TRUE)
svdsmooth(X,nomb=min(dim(X)),
                    smooth=list(function(u)ksmooth(1:length(u),u,kernel="normal",
                           bandwidth=3,x.points=(1:length(u)))$y),vsmin=1E-16, ...)
toplist(li)
svd.p(X,...)
```

Arguments

	These functions are not supposed to be called directly.
	a matrix
`Xest`	a zero limit number
`pena`	list of functions to be used as smoother
`ini`	initialisation method over the dual dimension
`vsmin`	zero limit for singular value

`Maxiter`	limit number of iteration
`A`	a matrix
`pw`	power value number
`eltw`	boolean to perform power elementwise or matrix power
`B`	a matrix
`solb`	an object inheriting from class `PTAk`
`sola`	an object inheriting from class `PTAk`
`solu`	an object inheriting from class `PTAk`
`numass`	position number of the associated solution, NULL is equivalent to the last in `sola`
`verbose`	boolean playing a verbose role
`file`	string pointing a destination of file output
`summary`	boolean to show the summary or not
`testvar`	threshold control for minimum percent of variability explained
`with`	boolean expression to give a supplementary selection criterion
`nomb`	integer giving the number of components to fit
`smooth`	idem as pena
`li`	any list
`...`	any other arguments passed to functions

Author(s)

Didier G. Leibovici

See Also

`PTAk`

`REBUILD` *Build an approximation of the tensor of any order*

Description

Gives the approximation of a previously analysed tensor using its given decomposition.

Usage

```
REBUILD(solutions,nTens=1:2,testvar=1 ,redundancy=FALSE)
```

Arguments

solutions
: a PTAk object

nTens
: a vector of identifying positions (numbers given in summary) for the choice of Principal Tensors to use

testvar
: control within nTens used Principal Tensor with minimum percent of variability explained

redundancy
: logical to take into account (within nTens) PT *tested* redundant during analysis (seealso RESUM) if TRUE.

Details

The function rebuilds the Principal Tensors, *i.e.* rank-one tensors of order the order of the tensor analysed, and add them up to build an approximation of the tensor analysed (according to the method used see method). This constitutes a best Least Squares (ordinary or "weighted" if metrics are used) approximation of datanam for a given *orthogonal-rank* r (number of principal tensors used), if and only if the singular values used are the r highest.

Value

A tensor with dimensions as solutions[[k]][["datanam"]].

Note

This function can be called for PARAFAC/CANDECOMP and PCAn. A specific rebuilt is implemented for this last one.

Author(s)

Didier G. Leibovici

See Also

PTAk

SINGVA *Optimisation algorithm RPVSCC*

Description

Computes the best rank-one approximation using the RPVSCC algorithm.

Usage

```
SINGVA(X,test=1E-12,PTnam="vs111",Maxiter=2000,
                  verbose=getOption("verbose"),file=NULL,
                    smoothing=FALSE,smoo=list(NA),
                     modesnam=NULL,
                      Ini="svds",sym=NULL)
```

Arguments

`X`	a tensor (as an array) of order *k*, if non-identity metrics are used `X` is a list with `data` as the array and `met` a list of metrics
`test`	numerical value to stop optimisation
`PTnam`	character giving the name of the *k*-modes Principal Tensor
`Maxiter`	if `iter > Maxiter` prompts to carry on or not, then do it every other 200 iterations
`verbose`	control printing
`file`	output printed at the prompt if `NULL`, or printed in the given '`file`'
`smoothing`	logical to use smooth functiosns or not (see `SVDgen`)
`smoo`	list of functions returning smoothed vectors (see `PTA3`)
`modesnam`	character vector of the names of the modes, if `NULL` `"mo 1"` ...`"mo k"`
`Ini`	method used for initialisation of the algorithm (see `INITIA`)
`sym`	description of the symmetry of the tensor *e.g.* c(1,1,3,4,1) means the second mode and the fifth are identical to the first

Details

The algorithm termed *RPVSCC* in Leibovici(1993) is implemented to compute the first Principal Tensor (rank-one tensor with its singular value) of the given tensor `X`. According to the decomposition described in Leibovici(1993) and Leibovici and Sabatier(1998), the function gives a generalisation to *k* modes of the *best rank-one approximation* issued from SVD whith 2 modes. It is identical to the PCA-*k*modes if only 1 dimension is asked in each space, and to PARAFAC/CANDECOMP if the rank of the approximation is fixed to 1. Then the methods differs, PTA-*k*modes will look for best approximation according to the *orthogonal rank* (*i.e.* the rank-one tensors (of the decomposition) are orthogonal), PCA-*k*modes will look for best approximation according to the *space ranks* (*i.e.* ranks of every bilinear form deducted from the original tensor, that is the number of components in each space), PARAFAC/CANDECOMP will look for best approximation according to the *rank* (*i.e.* the rank-one tensors are not necessarily orthogonal).
Recent work from Tamara G Kolda showed on an example that *orthogonal rank* decompositions are not necesseraly nested. This makes PTA-*k*modes a model with nested decompositions not giving the exact *orthogonal rank*. So PTA-*k*modes will look for best approximation according to orthogonal tensors in a nested approximmation process.

Value

a `PTAk` object (without `datanam method`)

Note

The algorithm was derived in generalising the *transition formulae* of SVD (Leibovici 1993), can also be understood as a generalisation of the *power method* (De Lathauwer et al. 2000). In this paper they also use a similar algorithm to build bases in each space, reminiscent of three-modes and *n*-modes PCA (Kroonenberg(1980)), *i.e.* defining what they called a rank-(R1,R2,...,Rn) approximation (called here *space ranks*, see `PCAn`). *RPVSCC* stands for *R*echerche de la *P*remi<e8>re *V*aleur *S*inguli<e8>re par *C*ontraction *C*ompl<ea>te.

Author(s)

Didier G. Leibovici

References

Kroonenberg P (1983) *Three-mode Principal Component Analysis: Theory and Applications*. DSWO press. Leiden.(related references in http://three-mode.leidenuniv.nl)

Leibovici D(1993) *Facteurs <e0> Mesures R<e9>p<e9>t<e9>es et Analyses Factorielles : applications <e0> un suivi <e9>pid<e9>miologique*. Universit<e9> de Montpellier II. PhD Thesis in Math<e9>matiques et Applications (Biostatistiques).

Leibovici D and Sabatier R (1998) *A Singular Value Decomposition of a k-ways array for a Principal Component Analysis of multi-way data, the PTA-k*. Linear Algebra and its Applications, 269:307-329.

De Lathauwer L, De Moor B and Vandewalle J (2000) *On the best rank-1 and rank-(R1,R2,...,Rn) approximation of higher-order tensors*. SIAM J. Matrix Anal. Appl. 21,4:1324-1342.

Kolda T.G (2003) *A Counterexample to the Possibility of an Extension of the Eckart-Young Low-Rank Approximation Theorem for the Orthogonal Rank Tensor Decomposition*. SIAM J. Matrix Analysis, 24(2):763-767, Jan. 2003.

See Also

`INITIA`, `PTAk`, `PCAn`, `CANDPARA`

`summary.PTAk` *Summary of a PTA-k modes analysis*

Description

Print a summary listing of the decomposition obtained.

Usage

```
## S3 method for class 'PTAk'
summary(object,testvar=1,dontshow="*", ...)
## S3 method for class 'FCAk'
summary(object,testvar=0.5,dontshow="*", ...)
```

Arguments

`object`	an object inheriting from class `PTAk`, representing a generalised singular value decomposition
`testvar`	control within `nTens` used Principal Tensor with minimum percent of variability explained
`dontshow`	boolean criterion to remove Principal Tensors from the summary, or default is a character "*" equivalent to the criterion: `!substr(solution[[length(solution)]][["vsnam"]],1,1)=="*"`
`...`	summary generic additional arguments not used here

Details

The function prints a listing of the decomposition with historical order (instead of traditional singular value order). It is useful before any plots or reconstruction, a screeplot (using `plot.PTAk`) will be also useful. It is useful before any plots r reconstruction, a screeplot (using `plot.PTAk`) will be also useful. `summary.FCAk` is alike `summary.PTAk` but `testvar` operates on the variability of the lack of complete independence.

Value

prints on the prompt with an invisible return of the summary table

Note

At the moment can be used for `PCAn`, `CANDPRA`, better summaries will be in the next release.

Author(s)

Didier G. Leibovici <GeotRYcs@gmail.com>

References

Leibovici D (2000) *Multiway Multidimensional Analysis for Pharmaco-EEG Studies*.(submitted) http://c3s2i.free.fr/cv/recentpub.html

See Also

`plot.PTAk`

Examples

```
data(crimerate)
crimerate.mat <- sweep(crimerate,2,apply(crimerate,2,mean))
crimerate.mat <- sweep(crimerate.mat,2,sqrt(apply(crimerate,2,var)),FUN="/")
cri.svd <- SVDgen(crimerate.mat)
summary(cri.svd,testvar=0)
 plot(cri.svd,scree=TRUE)
 par(new=TRUE)
 RiskJackplot(cri.svd,nbvs=1:7,mod=NULL,max=NULL,rescaled=TRUE,
       axes=FALSE,ann=FALSE)
 par(new=FALSE)

 # or equivalently

 plot(cri.svd,scree=TRUE,type="b",lty=3,RiskJack=1) #set mod=NULL or c(1,2)
 ###
  data(crimerate)
  criafc <- FCAmet(crimerate,chi2=TRUE)
  cri.afc <- SVDgen(criafc$data,criafc$met[[2]],criafc$met[[1]])
   summary(cri.afc)
  plot(cri.afc,scree=TRUE)
  plot(cri.afc,scree=TRUE,type="b",lty=3,RiskJack=1,method="FCA")
```

SVDgen *SVD with metrics and smoothing approximation*

Description

Computes the generalised Singular Value Decomposition, *i.e.* with non-identity metrics. A smooth approximation can be asked to constraint the components (u and v) to be smooth.

Usage

```
SVDgen(Y, D2 = 1, D1 = 1, smoothing = FALSE, nomb = NULL,
            smoo = list(function(u)ksmooth( 1:length(u), u, kernel = "normal",
                        bandwidth = 3, x.points = (1:length(u)))$y))
```

Arguments

Y	a matrix $n \times p$
D2	metric in R^p either a vector $(p \times 1)$ or a matrix $(p \times p)$
D1	metric in R^n either a vector $(n \times 1)$ or a matrix $(n \times n)$
smoothing	logical if TRUE the smoothing methods in smoo are used
nomb	numeric number of components to extract (typically when smoothing is used less components are used as the screeplot becomes flatter faster)
smoo	list of lists of smoothing functions on a vector of the approriate dimension; if on one dimension it is NA no smoothing will be done for this one; if the length of a list is one the function is used for all components. If only one list in the list it will be used for both dimensions.

Details

The function computes the decomposition $X = UL^{1/2}V'$ where $U'D_1U = Id_p$ and $V'D_2V = Id_p$ and the diagonal matrix L containing no zeros squared singular values. If smoothing a *constraint* on Least Squares solution is used, then the above decomposition becomes an approximation (unless X belongs to the space defined by the constraints). A *Power Method* algorithm to compute each principal tensor is used wherein Alternated Least Squares are always followed by a *smoothed version* of the updated vectors. If a Spline smoothing was used the algorithm would be equivalent to use the traditional *penalised least squares* at each iteration and could be called *Penalised Power Method* or Splined Alternated Least Squares Algorithm (SALSA is already an acronym used by Besse and Ferraty (1995) in where a similar idea is developped: but smoothing operates only on variables, and is *model based* as the Alternating operates on the whole approximation *i.e.* given the choice of the dimension reduction).

Value

a `PTAk` object

Note

`SVDgen` makes use of a non-identity version `svd` (inbuilt) or `svdksmooth` which outputs like the inbuilt `svd`. The smoothing option is also implemented in PTA-kmodes, FCA-kmodes, PCAn and CANDECOMP/PARAFAC. The use of metrics (diagonal or not) allows flexibility of analysis like *e.g.* correspondence analysis, discriminant analysis, robust analysis. Smoothing option extends the analysis towards functional data analysis, and or outliers protection.

This smoothing penalising approach is theoretically valid for Principal Tensors (here order 2) belonging to a tensor product of separable Hilbert spaces (*e.g.* Sobolev spaces) see Leibovici and El Maach (1997), and in fact only valid for projection onto this space : this includes polynomial fitting, spline basis fitting ... As you are penalysing the alternating optimisation criterion you also need the to get a *robust fit* at each iteration to be able to reach stationarity and declare optimisation done. If the smoother is not linear one looses orthogonality of the corresponding components but they are usually not too much correlated and preserving one mode to be unsmoothed insured orthogonality of the whole decomposition. Alternatively `keepOrtho` insures (as a third step optimisation for each iteration) orthogonality with the previous component (but then the solution is approximatively in the space of constraints).

The flexibility of this function `smoothing` constraint should be carefully used. The function offers also the choice to change of smoothing (method or parameters) as the number of components grows as in Ramsay and Silverman (1997).

Author(s)

Didier G. Leibovici <GeotRYcs@gmail.com>

References

Leibovici D and El Maache H (1997) *Une décomposition en Valeurs Singulières d'un élément d'un produit Tensoriel de k espaces de Hilbert Séparables*. Compte Rendus de l'Académie des Sciences tome 325, série I, Statistiques (Statistics) & Probabilités (Probability Theory): 779-782.

Besse P and Ferraty F (1995) *Curvilinear fixed effect model*. Computational Statistics, 10:339-351.

Leibovici D (2008) *A Simple Penalised algorithm for SVD and Multiway functional methods*. (to be submitted)

Ramsay J.O. and Silverman B.W. (1997) *Functional Data Analysis*. Springer Series in Statistics.

See Also

`PTAk`,`PCAn`, `CANDPARA`

Examples

```
#library(stats)
 #library(tensor)

 # on smoothing
```

```
 data(longley)
 long <- as.matrix(longley[,1:7])

 long.svd <- SVDgen(long,smoothing=FALSE)
  summary.PTAk(long.svd,testvar=0)
   # X11(width=4,height=4)
  plot.PTAk(long.svd,scree=TRUE,RiskJack=0,type="b",lty=3)

 long.svdo <- SVDgen(long,smoothing=TRUE,
  smoo=list(function(u)ksmooth(1:length(u),
      u,kernel="normal",bandwidth=3,x.points=(1:length(u)))$y,NA))

  summary.PTAk(long.svdo,testvar=0)
  #  X11(width=4,height=4)
  plot.PTAk(long.svdo,scree=TRUE,RiskJack=0,type="b",lty=3)
 ###using polynomial fitting
   polyfit <- function(u,deg=length(u)/5)
       {n <- length(u);time <- rep(1,n);
        for(e in 1:deg)time<-cbind(time,(1:n)^e);return(lm.fit(time,u)$fitted.values)}
bsfit<-function(u,deg=42)
       {n <- length(u);time <- rep(1,n);
        return(lm.fit(bs(time,df=deg),u)$fitted.values)}

###
 long.svdo2 <- SVDgen(long,nomb=4,smoothing=TRUE,smoo=list(polyfit,NA))
  long.svdo2[[1]]$v[1:3,]
long.svdo[[1]]$v[1:3,]
# orthogonality may be lost with non-projective smoother

     ####
comtoplot <- function(com=1,solua=long.svd,solub=long.svdo,openX11s=FALSE,...)
         {
  if(openX11s)X11(width=4,height=4)
 yla <- c(round((100*(solua[[2]]$d[com])^2)/
     solua[[2]]$ssX[1],4),
     round((100*(solub[[2]]$d[com])^2)/solua[[2]]$ssX[1],4))

limi <- range(c(solua[[1]]$v[com,],solub[[1]]$v[com,]))
  plot(solua,nb1=com, mod=1,type="b",lty=3,lengthlabels=4,cex=0.4,
   ylimit=limi,ylab="",...)
mtext(paste("vs",com,":",yla[1],"%"),2,col=2,line=2)
 par(new=TRUE)

  plot.PTAk(solub,nb1=com,mod=1,labels=FALSE,type="b",lty=1,
  lengthlabels=4,cex=0.6,ylimit=limi,ylab="",main=paste("smooth vs",com,":",yla[2],"%"),...)
  par(new=FALSE)
}   ####
 comtoplot(com=1)



#  on using non-diagonal metrics
```

```
data(crimerate)
 crimerate.mat <- sweep(crimerate,2,apply(crimerate,2,mean))
 crimerate.mat <- sweep(crimerate.mat,2,sqrt(apply(crimerate.mat,2,var)),FUN="/")
  metW <- Powmat(CauRuimet(crimerate.mat),(-1))
  # inverse of the within "group" (to play a bit more you could set m0 relating
  # the neighbourhood of states (see CauRuimet)

 cri.svd <- SVDgen(crimerate.mat,D2=1,D1=1)
 summary(cri.svd,testvar=0)
  plot(cri.svd,scree=TRUE,RiskJack=4,type="b",lty=3)
 cri.svdo <- SVDgen(crimerate.mat,D2=metW,D1=1)
  summary(cri.svdo,testvar=0)
  plot(cri.svdo,scree=TRUE,RiskJack=5,type="b",lty=3)
 # X11(width=8,height=4)
 par(mfrow=c(1,2))
  plot(cri.svd,nb1=1,nb2=2,mod=1,lengthlabels=3)
 plot(cri.svd,nb1=1,nb2=2,mod=2,lengthlabels=4,main="canonical")
 # X11(width=8,height=4)
 par(mfrow=c(1,2))
plot(cri.svdo,nb1=1,nb2=2,mod=1,lengthlabels=3)
plot(cri.svdo,nb1=1,nb2=2,mod=2,lengthlabels=4,
      main=expression(paste("metric ",Wg^{-1})))

###########
#  demo function
 # when ima is NULL it uses the dataset timage12 but you can put any array
 # demo.SVDgen(ima=NULL,snr=3,openX11s=TRUE)
```

TENSELE *Elementary Tensor product*

Description

Computes the Tensor Product of a list of vectors (or matrices) according to a given order.

Usage

```
TENSELE(T,moins=NULL, asarray=TRUE,order=NULL,id=NULL)
```

Arguments

T	a list like a PTAk object and minimally just contains v
moins	if not NULL, vector of indexes (in the list T) to skip
asarray	logical to specify the output form TRUE gives an array, FALSE gives a vector
order	if not NULL vector of length length(T), NULL is equivalent to length(T):1 as the function makes indexes in order run slowest to fastest

id
: when T is a list of matrices, can be either a vector of length(T) giving indexes of the vectors for each space (following order) or a list of vectors of indexes.

Details

The tensor product of the vectors (or matrices) in the list T is computed, skipping or not the indexes in moins, the output tensor is either in tensor form or in vector form. The way the tensor product is done follows order.

Value

According to asarray the value is either an array, or a vector representing the tensor product of the vectors (not in moins), the dimension in order[1] running the slowest.

Author(s)

Didier G. Leibovici

See Also

REBUILD

Index