Package: mccca (via r-universe)

August 22, 2024

Type Package

Title Visualizing Class Specific Heterogeneous Tendencies in Categorical Data

Version 1.1.0.1

Author Mariko Takagishi [aut, cre]

Maintainer Mariko Takagishi <m.takagishi0728@gmail.com>

Description Performing multiple-class cluster correspondence analysis(MCCCA). The main functions are create.MCCCAdata() to create a list to be applied to MCCCA, MCCCA() to apply MCCCA, and plot.mccca() for visualizing MCCCA result. Methods used in the package refers to Mariko Takagishi and Michel van de Velden (2022)<doi:10.1080/10618600.2022.2035737>.

License GPL (>= 2)

Depends R (>= 4.1.0)

Imports magic, stringr, ggplot2, wordcloud, RColorBrewer, stats, utils, grDevices, Rcpp (>= 1.0.9)

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.2.1

LinkingTo Rcpp, RcppArmadillo

Repository CRAN

Date/Publication 2024-01-24 14:39:05 UTC

Contents

`create.MCCCAdata` *this function creates a list (class: mcccadata) to be applied to MCCCA.*

Description

Creates a list (named `mcccadata.list`) applied to MCCCA.

Usage

```
create.MCCCAdata(dat,ext.mat=ext.mat,clstr0.vec=NULL)
```

Arguments

`dat`	An (NxJ) matrix of categorical data (N:the number of observations, J:the number of variables). If `rownames(dat)` is `NULL`, `c(obj1,..,objN)` are defined as `rownames(dat)`.
`ext.mat`	An (NxH) external variable matrix (H:the number of external variable).
`clstr0.vec`	An integer vector of length N giving each observation's true cluster.

Value

Returns a list with the following elements.

`data.mat`	data matrix same as `dat`.
`data.list`	A list of C (NxJ) categorical data matrices for each class (C:the number of classes).
`clstr0.list`	A list of C vectors where each vector indicates the true cluster (given in `clstr0.vec`) to which each class of observations belongs (NULL if `clstr0.vec` is NULL).
`N.vec`	A vector of length C giving the number of observations in each class.
`Ktrue.vec`	A vector of length C giving the true number of clusters in each class (NULL if `clstr0.vec` is NULL).
`q.vec`	A vector of length J giving the number of categories in each of J categorical variables.
`class.n.vec`	An integer (from 1:C) vector of length N giving the class index of each observation. `names(class.n.vec)=rownames(dat)`.
`classname.n.vec`	A characteristic vector of length N giving the class label each observation belongs to. `names(classname.n.vec)=rownames(dat)`.
`classlabel`	A characteristic vector of length C giving the classlabel for each class.
`classlab.mat`	(Cx(H+1)) table, showing which combinations of categories of external variables each class index and class name corresponds to. The first H columns indicate the categories for each of the H external variables, and the last H+1th column indicates the corresponding class label (same as `classlabel`).

`oriindex.list` A list of length C, where each list element corresponds to a row (observation) in data.list, indicating which row of observations (in `data.mat`) each observation (in `oriindex.list`) corresponds to.

References

Takagishi & Michel van de Velden (2022): Visualizing Class Specific Heterogeneous Tendencies in Categorical Data, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2022.2035737

Examples

```
#setting
N <- 100 ; J <- 5 ; Ktrue <- 2 ; q.vec <- rep(5,J) ; noise.prop <- 0.2
extcate.vec=c(2,3)#the number of categories for each external variable

#generate categorical variable data
catedata.list <- generate.onedata(N=N,J=J,Ktrue=Ktrue,q.vec=q.vec,noise.prop = noise.prop)
data.cate=catedata.list$data.mat
clstr0.vec=catedata.list$clstr0.vec

#generate external variable data
data.ext=generate.ext(N,extcate.vec=extcate.vec)

#create mccca.list to be applied to MCCCA function
mccca.data=create.MCCCAdata(data.cate,ext.mat=data.ext,clstr0.vec =clstr0.vec)

#check which class each observation belongs to. (given by class name)
mccca.data$classname.n.vec

#A table showing that which combinations of categories of external variables
# each class index and class name corresponds to.
mccca.data$classlab.mat
```

---

create.prop *Creates a list length J of category proportion for each cluster.*

Description

Creates a list length J of category proportion for each cluster.

Usage

```
create.prop(
  J = J,
  q.vec = q.vec,
  Ktrue = Ktrue,
  strongprop = 0.8,
  which.noise = NULL
)
```

Arguments

J	The number of active variable.!!!
q.vec	A vector of length J giving the number of categories for each active variable.
Ktrue	The number of clusters in J active variables.
strongprop	A numeric value giving the strongest proportion of categories (common for all J active variables).
which.noise	A vector of length (<= J) giving the index of noise variables in J active variables. NULL indicating all variable is non-noise.

Value

Returns a list length J, each of which is a (Ktrue x qj) matrix giving the proportion for each qj category in each Ktrue cluster.

---

generate.cate.list *Generate (NxJ) categorical data matrix.*

Description

Generate an (NxJ) categorical data matrix given by prop.J.list and true cluster allocation.

Usage

```
generate.cate.list(N = N, prop.list = prop.list)
```

Arguments

N	The number of observations.
prop.list	a list length J, each of which is a vector of length qj giving the proportion for each categories.

Value

an (NxJ) categorical data matrix.

generate.catecls *Generate (NxJ) clustered categorical data matrix.*

Description

Generate an (NxJ) clustered categorical data matrix given by prop.J.list and true cluster allocation.

Usage

```
generate.catecls(
  N = N,
  J = J,
  q.vec = q.vec,
  Ktrue = Ktrue,
  prop.J.list = prop.J.list,
  clstr.vec = clstr.vec
)
```

Arguments

N	The number of observations.
J	The number of active variables.
q.vec	A vector of length J giving the number of categories for each active variable.
Ktrue	An integer indicating the number of content-based clusters used for CCRS estimation.
prop.J.list	a list of length J, where each list is a (Ktrue x qj) matrix giving the proportion for each qj category in each of the Ktrue cluster.
clstr.vec	A vector of length N giving true clusters for each observations.

Value

an (NxJ) clustered categorical data matrix.

generate.ext *generates an artificial (NxH) external variable matrix.*

Description

Generates an artificial (NxH) external variable matrix.

Usage

```
generate.ext(N,extcate.vec=extcate.vec,unbala.cate=FALSE)
```

Arguments

N	The number of observation.
extcate.vec	A vector of length H, each element indicates the number of category for each H external variables.
unbala.cate	logical value. If TRUE, the proportion of categories in the external variable is unbalanced. The default is FALSE.

Value

An (NxH) external variable matrix.

See Also

generate.catecls

Examples

```
###data setting
N <- 30 ; extcate.vec=c(2,3)
ext.mat=generate.ext(N,extcate.vec=extcate.vec)
```

---

generate.onedata *Generate (NxJ) categorical data matrix.*

Description

Generate (NxJ) categorical data matrix.

Usage

```
generate.onedata(N=100,J=5,Ktrue=3,q.vec=rep(3,5),noise.prop=0.3)
```

Arguments

N	The number of observations. Default is 100.
J	The number of active variables. Default is 5.
Ktrue	The number of true clusters. Default is 3.
q.vec	A vector of length J giving the number of categories for each active variable. Default is rep(3,5).
noise.prop	A numeric value between 0 and 1 indicating the proportion of noise variables among J variables. Default is 0.3.

Value

Returns a list with the following elements.

data.mat	A (NxJ) data frame of categorical data.
clstr0.vec	A vector of integers (from 1:Ktrue) length N giving the cluster to which each observation is allocated.

See Also

create.prop, generate.catecls

Examples

```
###data setting
N <- 30 ; J <- 10 ; Ktrue <- 2 ; q.vec <- rep(5,J) ; noise.prop <- 0.3
datagene <- generate.onedata(N=N,J=J,Ktrue=Ktrue,q.vec=q.vec,noise.prop = noise.prop)
```

MCCCA *apply MCCCA for dataset.*

Description

Applies MCCCA to mcccadata.list.

Usage

```
MCCCA(
  mccca.data,
  K.vec = K.vec,
  known.vec = NULL,
  knowncluster.list = NULL,
  nstart = 3,
  maxit = 50,
  p = 2,
  tol = 1e-08,
  verbose = TRUE,
  remove.miss = TRUE,
  kmeans.initial = TRUE
)
```

Arguments

mccca.data	A list created in create.MCCCAdata.
K.vec	An integer vector of length C (the number of classes). Each element corresponds to the number of clusters in each class specified for estimation.
known.vec	A vector of length C giving logical values indicating whether a cluster allocation in each class is known or not. The default is all FALSE.

knowncluster.list
: A vector of length C giving logical values indicating whether a cluster allocation in each class is known or not. The default is all FALSE.

nstart
: An integer indicating the number of random initial values.

maxit
: An integer indicating the maximum number of iterations.

p
: An integer indicating the dimension of quantification.The default is 2.

tol
: A numeric value indicating the absolute convergence tolerance.

verbose
: A logical value indicating. If TRUE, tracing information on the progress of the optimization is produced.

remove.miss
: A logical value indicating whether categories nobody choose are removed nor not. The default is TRUE.

kmeans.initial
: A logical value indicating whether the 1st initial value for indicator matrix is generated by kmeans or not. The default is TRUE.

Details

Bg,Gg and Qg are scaled B,G and Q respectively, such that the average squared deviation from the origin of the row and column points is the same (See section 2.3 in the paper).

If you want to specify the cluster allocation for some or all classes, prepare the following two.

-knowncluster.list: A list of C vectors. The length of each vector in the list should be the same as the number of rows in each matrix in the data.list (ex. length(knowncluster.list[[c]])=nrow(data.list[[c]]), (c=1,..,C)). For example, suppose that data.list is a list of 4 matrices (meaning C=4), and the cluster assignment is known only for the second class, and the assignments in other classes are estimated. In this case, the second vector of knowncluster.list should be specified as the vector of cluster indexes to which the observations in each row of data.list[[2]] belong, with length nrow(data.list[[2]]), and the other vectors (1, 3, and 4) in the list can be specified as NA. For each vector in the knowncluster.list, the specified cluster index should start from 1, and there should not be any skipping numbers.

-known.vec: A vector of logical values of length C. For example, if C=4 and you want to know the cluster assignment of only the second class, it should be known.vec=c(FALSE,TRUE,FALSE,FALSE).

Value

Returns a list with the following elements.

G
: A (Kxp) quantification matrix for all clusters (K=sum(K.vec)).

Gg
: Scaled G. See details.

B
: A (Qxp) quantification matrix for all categories (Q=sum(q.vec), and q.vec is given in create.MCCCAdata).

Bg
: Scaled B.

Q
: A (Nxp) quantification matrix for all observations.

Qg
: Scaled Q.

clses.list
: A list of C vectors, giving the estimated cluster index for each observation in each class.

clses.vec	A vector of length N, where each element represents the cluster index to which the observations in the rows of data.mat (given in mccca.data) belong.
optval	A numeric value giving the optimized value of the objective function that is the smallest among all initial values.
optval.vec	A numeric vector of length nstart giving the optimized values of the objective function for each initial value.
stepconv	An integer giving the number of iterations until convergence at the initial value where the objective function was the smallest.
stepconv.vec	An integer vector of length nstart giving the number of iterations until convergence for each initial value.
catename.vec	A characteristic vector of length Q that combines the category names of each categorical variable into a single vector.
catename.vari.vec	A characteristic vector of length Q with catename.vec plus the name of categorical variable (by default, this is used as the column name of B and Bg).
cate.removed	If there is a category that no one chooses and remove.miss=TRUE, cate.removed gives which category was removed (given by the index of column in dummy matrix). Otherwise, return NULL.
cluster.vec	An integer vector of length K, where each index in the clses.list and clses.vec indicates which class it corresponds to.
q.vec	A vector of length J, same as the one given in mccca.data.
K.vec	A vector of length C, which is used as an input in this MCCCA function.
classlabel	A characteristic vector of length C, same as the one given in mccca.data.

References

Takagishi & Michel van de Velden (2022): Visualizing Class Specific Heterogeneous Tendencies in Categorical Data, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2022.2035737

See Also

create.MCCCAdata

Examples

```
#setting
N <- 100 ; J <- 5 ; Ktrue <- 2 ; q.vec <- rep(5,J) ; noise.prop <- 0.2
extcate.vec=c(2,3)#the number of categories for each external variable

#generate categorical variable data
catedata.list <- generate.onedata(N=N,J=J,Ktrue=Ktrue,q.vec=q.vec,noise.prop = noise.prop)
data.cate=catedata.list$data.mat
clstr0.vec=catedata.list$clstr0.vec

#generate external variable data
data.ext=generate.ext(N,extcate.vec=extcate.vec)
```

```
#create mccca.list to be applied to MCCCA function
mccca.data=create.MCCCAdata(data.cate,ext.mat=data.ext,clstr0.vec =clstr0.vec)

#specify the number of cluster for each of C classes
C=length(mccca.data$data.list)
K.vec=rep(2,C)

#apply MCCCA
mccca.res=MCCCA(mccca.data,K.vec=K.vec)

#plot MCCCA result
plot(mccca.res)

#if you want to specify cluster allocation in the 2nd class:
knowncluster.list=rep(list(NA),C)
#specify cluster index for the 2nd class
N2=nrow(mccca.data$data.list[[2]])
knowncluster.list[[2]]=rep(c(1,2),times=c(2,N2-2))
known.vec=c(FALSE,TRUE,FALSE,FALSE,FALSE,FALSE)
mccca.res=MCCCA(mccca.data,K.vec=K.vec,known.vec=known.vec,knowncluster.list = knowncluster.list)
```

---

plot.mccca *plot* mccca *object.*

Description

plot mccca object.

Usage

```
## S3 method for class 'mccca'
plot(
  x,
  main = "MCCCA result",
  catelabel = NULL,
  classlabel = NULL,
  classlabel.legend = NULL,
  xlim = NULL,
  ylim = NULL,
  sort.clssize = TRUE,
  break.size = NULL,
  output.coord = FALSE,
  connect.cord = TRUE,
  include.variname = TRUE,
  scale.gamma = TRUE,
  scatter.level = 2,
 plot.setting = list(alp.point = 0.3, alp.seg = 0.8, txtsize = 3, txtsize.legend = 10),
  ...
)
```

Arguments

x	An object of class mccca, a list of MCCCA outputs.
main	A character giving the title of biplot.
catelabel	A characteristic vector of length Q giving labels for all categories to be displayed on the biplot (Q=sum(q.vec)). If NULL, rownames(B) are used.
classlabel	A characteristic vector of length C (C:the number of class) giving labels for all classes to be displayed on the biplot. If NULL, labels specified in create.MCCCAdata are used.
classlabel.legend	A characteristic vector of length C giving labels for all classes to be used on the legend (this can be longer). If NULL, classlabel is used.
xlim	A numeric vector of length 2 giving the range of plot on the x (horizontal) axis. If NULL, the range is automatically determined.
ylim	A numeric vector of length 2 for the y (vertical) axis (same role as xlim).
sort.clssize	If TRUE, the class-specific cluster numbers are sorted in the order of cluster size. The default is TRUE.
break.size	An integer vector that adjusts the size of bubble displayed on the legend.
output.coord	If TRUE, the output will be Cocls.mat and Cocate.mat. See value.
connect.cord	If TRUE, lines are drawn between original (estimated by MCCCA) coordinates and coordinates moved to avoid overlap.
include.variname	If TRUE, variable name is included in category labels in the biplot (ex.a point of category "male" in "v1"(the name of 1st variable) is displayed as "v1:male" on the biplot).
scale.gamma	If TRUE, quantifications are scaled such that the average squared deviation from the origin of the row and column points is the same (See section 2.3 in the paper).
scatter.level	A numeric value that adjusts the scatter of points in the biplot. The higher the value, the more scattered the points are. The default is 2.
plot.setting	A list of biplot settings. See details.
...	Additional arguments passed to print.

Details

Parameters in plot.setting are as follows:

-alp.point:A numeric value from 0 to 1 which adjusts the transparency of the bubble point. The default is 0.3.

-alp.seg:A numeric value from 0 to 1 which adjusts the transparency of the segments between texts and points. The default is 0.8.

-txtsize:A numeric value which adjusts the textsize on the biplot. The default is 3.

-txtsize.legend:A numeric value which adjusts the textsize of the legend on the biplot. The default is 10.

Value

If `output.coord` is `TRUE`, returns a list with the following elements.

`Cocls.mat`	A (Kx4) coordinate matrix of clusters, where the last two columns are the coordinates estimated by MCCCA, and the first two columns are the coordinates moved from the estimated coordinates to prevent overlap.
`Cocate.mat`	A (Kx4) coordinate matrix of categories (each column plays the same role as `Cocls.mat`)

References

Takagishi & Michel van de Velden (2022): Visualizing Class Specific Heterogeneous Tendencies in Categorical Data, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2022.2035737

See Also

MCCCA

Examples

```
#setting
N <- 100 ; J <- 5 ; Ktrue <- 2 ; q.vec <- rep(5,J) ; noise.prop <- 0.2
extcate.vec=c(2,3)#the number of categories for each external variable

#generate categorical variable data
catedata.list <- generate.onedata(N=N,J=J,Ktrue=Ktrue,q.vec=q.vec,noise.prop = noise.prop)
data.cate=catedata.list$data.mat
clstr0.vec=catedata.list$clstr0.vec
#generate external variable data
data.ext=generate.ext(N,extcate.vec=extcate.vec)

#create mccca.list to be applied to MCCCA function
mccca.data=create.MCCCAdata(data.cate,ext.mat=data.ext,clstr0.vec =clstr0.vec)

#specify the number of cluster for each of C classes
C=length(mccca.data$data.list)
K.vec=rep(2,C)
#apply MCCCA
mccca.res=MCCCA(mccca.data,K.vec=K.vec)

#plot MCCCA result
plot(mccca.res)
```

Index