Package 'OTclust'

December 18, 2020

Title Mean Partition, Uncertainty Assessment, Cluster Validation and Visualization Selection for Cluster Analysis

Version 1.0.4

Author Lixiang Zhang [aut, cre],
Beomseok Seo [aut],
Lin Lin [aut],
Jia Li [aut]

Maintainer Lixiang Zhang <lzz46@psu.edu>

Description Providing mean partition for ensemble clustering by optimal transport alignment(OTA), uncertainty measures for both partition-wise and cluster-wise assessment and multiple visualization functions to show uncertainty, for instance, membership heat map and plot of covering point set. A partition refers to an overall clustering result.

Depends R (>= 3.1.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Suggests knitr, rmarkdown, tsne, umap, HDclust, dbscan, flexclust, mclust

VignetteBuilder knitr

LinkingTo Rcpp

Imports Rcpp, ggplot2, RColorBrewer, magrittr, class

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-12-17 23:10:02 UTC

R topics documented:

`align` *Optimal Transport Alignment*

Description

This function aligns an ensemble of partitions with a reference partition by optimal transport.

Usage

```
align(data)
```

Arguments

`data`	– a numeric matrix of horizontally stacked cluster labels. Each column contains cluster labels for all the data points according to one clustering result. The reference clustering result is put in the first column, and the first cluster must be labeled as 1.

Value

a list of alignment result.

`distance`	Wasserstein distances between the reference partition and the others.
`numcls`	the number of clusters for each partition.
`statistics`	average tightness ratio, average coverage ratio, 1-average jaccard distance.
`cap`	cluster alignment and points based (CAP) separability.
`id`	switched labels.
`cps`	covering point set.
`match`	topological relationship statistics between the reference partition and the others.
`Weight`	weight matrix.

Examples

```
data(sim1)
# the number of clusters.
C = 4
# calculate baseline method for comparison.
kcl = kmeans(sim1$X,C)
# align clustering results for convenience of comparison.
compar = align(cbind(sim1$z,kcl$cluster))
```

---

clustCPS *CPS Analysis for cluster validation..*

Description

Covering Point Set Analysis for validating clustering results. It conducts alignment among different results and then calculates the covering point set. The return contains several statistics which can be directly used as input for mplot or cplot. If you want to design your own workflow, you can use function CPS instead.

Usage

```
clustCPS(
  data,
  k,
  l = TRUE,
  pre = TRUE,
  noi = "after",
  cmethod = "kmeans",
  dimr = "PCA",
  vis = "tsne",
  ref = NULL,
  nPCA = 50,
  nEXP = 100
)
```

Arguments

data	– data given in a matrix format, where rows are samples, and columns are variables.
k	– number of clusters.
l	– logical. If True, log-transforamtion will be carried out on the data.
pre	– logical. If True, pre-dimension reduction will be carried out based on the variance.
noi	– adding noise before or after the dimension reduction, choosing between "before" and "after", default "after".
cmethod	– clustering method, choosing from "kmeans" and "mclust", default "kmeans".

dimr	– dimension reduction technique, choose from "none" and "PCA", default "PCA".
vis	– the visualization method to be used, such as "tsne" and "umap", default "tsne". Also, you can provide your own visualization coordinates in a numeric matrix of two columns.
ref	– optional, clustering reuslt in a vector format and the first cluster is labeled as 1. If provided it will be used as the reference, if not we will generate one.
nPCA	– number of principal conponents to use, default 50.
nEXP	– number of perturbed clustering reuslts for CPS Analysis, default 100.

Value

a list used for mplot or cplot, in which tight_all is the overall tightness, member is the matrix used for the membership plot, set is the matrix for the covering point set plot, tight is the vector of cluster-wise tightness, vis is the visualization coordinates, ref is the reference labels and topo is the topological relationship between clusters for point-wise uncertainty assessment.

Examples

```
# CPS Analysis on validation of clustering result
data(YAN)
# Suppose you generate the visualization coordinates on your own
x1=matrix(seq(1,nrow(YAN),1),ncol=1)
x2=matrix(seq(1,nrow(YAN),1),ncol=1)
# Using nEXP=50 for illustration, usually use nEXP greater 100
y=clustCPS(YAN[,1:100], k=7, l=FALSE, pre=FALSE, noi="after",vis=cbind(x1,x2), nEXP = 50)
# visualization of the results
mplot(y,4)
```

---

cplot *Covering Point Set Plot*

Description

Output the Covering Point Set plot of the required cluster. The return of clustCPS, visCPS or CPS can be directly used as the input.

Usage

```
cplot(result, k)
```

Arguments

result	– the return from function clustCPS, visCPS or CPS.
k	– which cluster that you want to see the covering point set plot.

Value

covering point set plot of the required cluster.

Examples

```
# CPS analysis on selection of visualization methods
data(vis_pollen)
c=visCPS(vis_pollen$vis, vis_pollen$ref)
# visualization of the results
mplot(c,2)
cplot(c,2)
```

CPS *CPS Analysis on a collection of clustering results*

Description

Covering Point Set Analysis of given clustering results. It conducts alignment among different results and then calculates the covering point set. The return contains several statistics which can be directly used as input for mplot or cplot. By using this function you can design your own workflow instead of using clustCPS, see vignette for more destails.

Usage

```
CPS(ref, vis, pert)
```

Arguments

ref	– the reference clustering result in a vector, the first cluster is labeled as 1.
vis	– the visualization coordinates in a numeric matrix of two columns.
pert	– a collection of clustering results in a matrix format, each column represents one clustering result.

Value

a list used for mplot or cplot, in which tight_all is the overall tightness, member is the matrix used for the membership heat map, set is the matrix for the covering point set plot, tight is the vector of cluster-wise tightness, vis is the visualization coordinates, ref is the reference labels and topo is the topological relationship between clusters for point-wise uncertainty assessment.

Examples

```
# CPS analysis on selection of visualization methods
data(vis_pollen)
k1=kmeans(vis_pollen$vis,max(vis_pollen$ref))$cluster
k2=kmeans(vis_pollen$vis,max(vis_pollen$ref))$cluster
k=cbind(as.matrix(k1,ncol=1),as.matrix(k2,ncol=1))
c=CPS(vis_pollen$ref, vis_pollen$vis, pert=k)
# visualization of the results
mplot(c,2)
cplot(c,2)
```

`ensemble` *Generate an ensemble of partitions.*

Description

Generate multiple clustering results (that is, partitions) based on multiple versions of perturbed data using a specified baseline clustering method.

Usage

```
ensemble(data, nbs, clust_param, clustering = "kmeans", perturb_method = 1)
```

Arguments

`data`	– data that will be perturbed.
`nbs`	– the number of clustering partitions to be generated.
`clust_param`	– parameters for pre-defined clustering methods. If clustering is "kmeans", "Mclust", "hclust", this is an integer indicating the number of clusters. For "dbscan", a numeric indicating epsilon. For "HMM-VB", a list of parameters.
`clustering`	– baseline clustering methods. User specified functions or example methods included in package ("kmeans", "Mclust", "hclust", "dbscan", "HMM-VB") can be used. Refer to the Detail.
`perturb_method`	– adding noise is `0` and bootstrap resampling is `1`. Default is bootstrap resampling. # perturb_method=0 perturbed by adding Gaussian noise.

Value

a matrix of cluster labels of the ensemble partitions. Each column is cluster labels of an individual clustering result.

Examples

```
data(sim1)
# the number of clusters.
C = 4
ens.data = ensemble(sim1$X[1:10,], nbs=10, clust_param=C, clustering="kmeans", perturb_method=1)
```

jaccard *Jaccard similarity matrix.*

Description

This function calculates Jaccard similarity matrix between two partitions.

Usage

```
jaccard(x, y)
```

Arguments

x, y	– vectors of cluster labels

Value

a matrix of Jaccard similarity between clusters in two partitions.

Examples

```
x=c(1,2,3)
y=c(3,2,1)
jaccard(x,y)
```

mplot *Membership Heat Map*

Description

Output the membership meat map of the required cluster. The return of clustCPS, visCPS or CPS can be directly used as the input.

Usage

```
mplot(result, k)
```

Arguments

result	– the return from function clustCPS, visCPS or CPS.
k	– which cluster that you want to see the membership heat map.

Value

membership heat map of the required cluster.

Examples

```
# CPS analysis on selection of visualization methods
data(vis_pollen)
c=visCPS(vis_pollen$vis, vis_pollen$ref)
# visualization of the results
mplot(c,2)
cplot(c,2)
```

`otclust` *Mean partition by optimal transport anlignment.*

Description

This function calculates the mean partition of an ensemble of partitions by optimal transport alignment and uncertainty/stability measures.

Usage

```
otclust(ensemble, idx = NULL)
```

Arguments

`ensemble`	– a matrix of ensemble partition. Use `ensemble()` to generate an ensemble of perturbed partitions.
`idx`	– an integer indicating the index of reference partition in `ensemble`. If not specified, median partition is used as the reference partition.

Value

a list of alignment result.

`idx`	the index of reference partition.
`avedist`	average distances between each partition and all ensemble partitions.
`meanpart`	a list of mean partition.
`distance`	Wasserstein distances between mean partition and the others.
`numcls`	the number of clusters for each partition.
`statistics`	average tightness ratio, average coverage ratio, 1-average jaccard distance.
`cap`	cluster alignment and points based (CAP) separability.
`id`	switched labels.
`cps`	covering point set.
`match`	topological relationship statistics between the reference partition and the others.
`Weight`	weight matrix.

Examples

```
data(sim1)
# the number of clusters.
C = 4
ens.data = ensemble(sim1$X[1:100,], nbs=10, clust_param=C, clustering="kmeans", perturb_method=1)
# find mean partition and uncertainty statistics.
ota = otclust(ens.data)
```

`otplot` *Visulize a partition on 2 dimensional space*

Description

This function plots a partition on 2 dimensional reduced space.

Usage

```
otplot(
  data,
  labels,
  convex.hull = F,
  title = "",
  xlab = "",
  ylab = "",
  legend.title = "",
  legend.labels = NULL,
  add.text = T
)
```

Arguments

`data`	– cordinates matrix of data.
`labels`	– cluster labels in a vector, the first cluster is labeled as 1.
`convex.hull`	– logical. If it is `True`, the plot draws convex hull for each cluster.
`title`	– title
`xlab`	– xlab
`ylab`	– ylab
`legend.title`	– legend title
`legend.labels`	– legend labels
`add.text`	– default True

Value

none

Examples

```
data(sim1)
# the number of clusters.
C = 4
ens.data = ensemble(sim1$X[1:50,], nbs=50, clust_param=C, clustering="kmeans", perturb_method=1)

# find mean partition and uncertainty statistics.
ota = otclust(ens.data)
# calculate baseline method for comparison.
kcl = kmeans(sim1$X[1:50],C)

# align clustering results for convenience of comparison.
compar = align(cbind(sim1$z[1:50],kcl$cluster,ota$meanpart))
lab.match = lapply(compar$weight,function(x) apply(x,2,which.max))
kcl.algnd = match(kcl$cluster,lab.match[[1]])
ota.algnd = match(ota$meanpart,lab.match[[2]])
# plot the result on two dimensional space.
otplot(sim1$X[1:50,],ota.algnd,con=FALSE,title='Mean partition')  # mean partition by OTclust
```

---

perturb *Perturb data by adding noise, bootstrapping or mix-up*

Description

Perturb data by adding Gaussian noise, bootstrap resampling or mix-up. Gaussian noise has mean 0 and variance 0.01*average variance of all variables. The mix-up lambda is 0.9.

Usage

```
perturb(data, method = 0)
```

Arguments

`data`	– data that will be perturbed.
`method`	– adding noise is `0`, bootstrapping is `1` and mix-up is `2`. Default is adding noise.

Value

the perturbed data.

Examples

```
data(vis_pollen)
perturb(as.matrix(vis_pollen$vis),method=0)
```

pplot *Point-wise Uncertainty Assessment*

Description

Output both the numerical and graphical point-wise uncertainty assessment for each individual points. The return of clustCPS, visCPS or CPS can be directly used as the input.

Usage

```
pplot(result, method = 0)
```

Arguments

result	– the return from function clustCPS, visCPS or CPS.
method	– method for calculating point-wise uncertainty. Using posterior probability matrix is 0 and using topological information between clusters is 1. Default is using posterior probability matrix.

Value

a list, in which P is the posterior probability matrix that each sample belows to the reference clusters, point_stab is the point-wise stability for each sample and v is the visualization of the point-wise stability.

Examples

```
# CPS analysis on selection of visualization methods
data(vis_pollen)
k1=kmeans(vis_pollen$vis,max(vis_pollen$ref))$cluster
k2=kmeans(vis_pollen$vis,max(vis_pollen$ref))$cluster
k=cbind(as.matrix(k1,ncol=1),as.matrix(k2,ncol=1))
c=CPS(vis_pollen$ref, vis_pollen$vis, pert=k)
# Point-wise Uncertainty Assessment
pplot(c)
```

preprocess *Data preprocessing*

Description

Preprocessing for dimension reduction based on variance, it will delete the variable whose variance is smaller than 0.5*mean variance of all variables.

Usage

```
preprocess(data, l = TRUE, pre = TRUE)
```

Arguments

data	– data that needs to be processed
l	– logical. If True, log-transforamtion will be carried out on the data.
pre	– logical. If True, pre-dimension reduction will be carried out based on the variance.

Value

the processed data.

Examples

```
data(YAN)
preprocess(YAN,l=FALSE,pre=TRUE)
```

---

sim1 *Simulated toy data*

Description

A dataset containing 5000 samples and 2 features.

Usage

```
sim1
```

Format

A matrix with 5000 rows and 2 variables

---

visCPS *CPS Analysis on selecting visualization method.*

Description

Covering Point Set Analysis on the visualization results. Use K-Nearest Neighbor to generate a collection of results for CPS Analysis. The return contains several statistics which can be directly used as input for mplot or cplot.

Usage

```
visCPS(vlab, ref, nEXP = 100)
```

Arguments

vlab	– the coordinates generated by one visualizatoin method in a numeric matrix of two columns.
ref	– the true labels in a vector format, the first cluster is labeled as 1.
nEXP	– number of perturbed reuslts for CPS Analysis.

Value

a list used for mplot or cplot, in which tight_all is the overall tightness, member is the matrix used for the membership heat map, set is the matrix for the covering point set plot, tight is the vector of cluster-wise tightness, vis is the visualization coordinates, ref is the reference labels and topo is the topological relationship between clusters for point-wise uncertainty assessment.

Examples

```
# CPS analysis on selection of visualization methods
data(vis_pollen)
c=visCPS(vis_pollen$vis, vis_pollen$ref)
# visualization of the results
mplot(c,2)
cplot(c,2)
```

---

vis_pollen *Single cell gene visualization data from Pollen's paper*

Description

A dataset containing the visualization coordinates and the true cluster labels of 301 cells.

Usage

```
vis_pollen
```

Format

A list containing two components

vis visualization coordinates of cells

ref true labels of cells

Source

https://www.nature.com/articles/nbt.2967

wassDist *Wasserstein distance between two partitions.*

Description

This function calculates Wasserstein distance between two partitions.

Usage

```
wassDist(x, y)
```

Arguments

x, y	– vectors of cluster labels

Value

a distance between 0 and 1.

Examples

```
x=c(1,2,3)
y=c(3,2,1)
wassDist(x,y)
```

YAN *Single cell gene data from Yan's paper*

Description

A dataset containing 124 cells with their 3840 genes.

Usage

```
YAN
```

Format

A matrix with 124 rows and 3840 variables

Source

https://www.nature.com/articles/nsmb.2660

Index