Package 'anocva'

November 10, 2017

Type Package

Title A Non-Parametric Statistical Test to Compare Clustering Structures

Version 0.1.1

Author Maciel C. Vidal [aut, cre], Andre Fujita [aut]

Maintainer Maciel C. Vidal <calebe@ime.usp.br>

Description Provides ANOCVA (ANalysis Of Cluster VAriability), a non-parametric statistical test to compare clustering structures with applications in functional magnetic resonance imaging data (fMRI). The ANOCVA allows us to compare the clustering structure of multiple groups simultaneously and also to identify features that contribute to the differential clustering.

License GPL (>= 3)

LazyLoad Yes

Encoding UTF-8

Depends R (>= 2.10.0)

NeedsCompilation no

Imports cluster

Suggests MASS, igraph

Repository CRAN

RoxygenNote 6.0.1

Date/Publication 2017-11-10 04:27:50 UTC

R topics documented:

anocva *ANalysis Of Cluster VAriability*

Description

The ANOCVA (ANalysis Of Cluster VAriability) is a non-parametric statistical test to compare clusters with applications in functional magnetic resonance imaging data. The ANOCVA allows us to compare the clustering structure of multiple groups simultaneously and also to identify features that contribute to the differential clustering.

Usage

```
anocva(dataDist, id, replicates = 1000, r = NULL,
  clusteringFunction = NULL, p = 1, maxClust = 20,
  criterion = c("slope", "silhouette"), showElapTime = TRUE)
```

Arguments

dataDist	A matrix with multiple matrices of dissimilarites. Given that a subject with N items (e.g. ROIs) has a matrix of dissimilarities of size NxN, the dataDist matrix should contain the dissimilarity matrix of all subjects (n) of all populations, resulting in a three-dimensional (nxNxN) matrix.
id	A list in range 1,2,...,n, where id[i] identifies the population id for the i-th subject.
replicates	The number of bootstrap replicates. The default value is 1000.
r	The optimal number of clusters. If NULL, then it will be estimated by the slope criterion in the interval 2..20.
clusteringFunction	Determines the clustering function that will be used. The default function is 'spectralClustering'. The clustering function is suposed to return the clustering labels.
p	Slope adjust parameter. Only used if r is unknown.
maxClust	The maximum number of clusters to be tried if estimating optimal number of clusters. The default value is 20.
criterion	The criterion that will be used for estimating the number of clusters (if r is unknown). The options are "slope" or "silhouette". If not defined, "slope" will be used.
showElapTime	Determines whether the total processing time should be displayed. The default value is TRUE.

Details

The test statistic used is the one proposed by Caetano de Jesus (2017).

Value

ANOCVA p-values

References

Fujita A, Takahashi DY, Patriota AG, Sato JR (2014a) A non-parametric statistical test to compare clusters with applications in functional magnetic resonance imaging data. Statistics in Medicine 33: 4949–4962

Vidal MC, Sato JR, Balardin JB, Takahashi DY, Fujita A (2017) ANOCVA in R: a software to compare clusters between groups and its application to the study of autism spectrum disorder. Frontiers in Neuroscience 11:1–8

Caetano de Jesus DA. (2017) Evaluation of ANOCVA test for cluster comparison through simulations. Master Dissertation. Institute of Mathematics and Statistics, University of São Paulo.

Examples

```
# Install packages if necessary
# install.packages('MASS')
# install.packages('cluster')

library(anocva)
library(MASS)
library(cluster)

set.seed(5000)

# Defines a k-means function that returns cluster labels directly
myKmeans = function(dist, k){
  return(kmeans(dist, k, iter.max = 50, nstart = 5)$cluster)
}

# Number of subjects in each population
nsub = 20
# Number of items in each subject
nitem = 30

# Generate simulated data
data = array(NA, c(nsub*2, nitem*2, 2))
dataDist = array(NA, c(nsub*2, nitem*2, nitem*2))
meanx = 2
delta = 0.5
# Covariance matrix
sigma = matrix(c(0.03, 0, 0, 0.03), 2)
for (i in seq(nsub*2)){
  sub = rbind(mvrnorm(nitem, mu = c(0, 0), Sigma = sigma ),
              mvrnorm(nitem, mu = c(meanx,0), Sigma = sigma))
  data[i,,] = sub
  # If it's a sample of population 2.
  if (i > nsub){
    data[i,10,1] = data[i,10,1] + delta
```

```
  }
  # Euclidian distance
  dataDist[i,,] = as.matrix(dist(data[i,,]))
}

# Population 1 subject
plot(data[5,,], asp = 1, xlab = '', ylab = '', main = 'Population 1 - subject example')

# Population 2 subject
plot(data[35,,], asp = 1, xlab = '', ylab = '', main = 'Population 2 - subject example')

# The first nsub subjects belong to population 1 while the next nsub subjects belong to population 2
id = c(rep(1, nsub), rep(2, nsub))

## Not run:
# ANOCVA call with different clustering function (myKmeans) and inside estimation of
# the number of clusters (r)
res1 = anocva(dataDist, id, replicates=500, r = NULL,
              clusteringFunction = myKmeans,
              p = 1, criterion = "slope")

## End(Not run)

# Estimate the number of clusters previously by using Spectral Clustering and Slope criterion
r = nClustMulti(dataDist, clusteringFunction = spectralClustering, criterion = 'slope')

# Calls ANOCVA statistical test
res = anocva(dataDist, id, replicates=500, r = r,
             clusteringFunction = spectralClustering,
             p = 1, criterion = "slope")

# DeltaS p-value
res$pValueDeltaS

# DeltaSq p-values
res$pValueDeltaSq

# Identifies which items have significant p-values with a significance level of 0.05.
which(res$pValueDeltaSq < 0.05)

# Identifies which items have significant FDR adjusted p-values (q-values)
# with a significance level of 0.05.
qValue = p.adjust(res$pValueDeltaSq, "fdr")
which(qValue < 0.05)
```

---

checkRange01 *Check for 0,1 Interval Normalization.*

Description

Verifies if the data is normalized in the range 0,1. If they are not, the normalization is performed and a warning issued.

Usage

```
checkRange01(data)
```

Arguments

data	A matrix of data

Value

The data matrix normalized in the range 0,1.

Examples

```
set.seed(2000)

simuData = runif(100, min = 0.5, max=7)
sprintf("The minimum value is %.2f and the maximum is %.2f.", min(simuData), max(simuData))

simuData = checkRange01(simuData)
sprintf("Now the minimum value is %.2f and the maximum is %.2f.", min(simuData), max(simuData))
```

---

nClust *Optimal Number of Clusters Estimation*

Description

Estimates the optimal number of clusters using either Slope or Silhouette criterion. The optimal number of clusters will be verified in the range 2,..., maxClust.

Usage

```
nClust(meanDist, p = 1, maxClust = 20, clusteringFunction,
  criterion = c("slope", "silhouette"))
```

Arguments

meanDist	An NxN matrix that represents the distances between the N items of the sample.
p	Slope adjust parameter.
maxClust	The maximum number of clusters to be tried. The default value is 20.
clusteringFunction	The clustering function to be used.
criterion	The criterion that will be used for estimating the number of clusters. The options are "slope" or "silhouette". If not defined, "slope" will be used.

Value

The optimal number of clusters.

References

Fujita A, Takahashi DY, Patriota AG (2014b) A non-parametric method to estimate the number of clusters. Computational Statistics & Data Analysis 73:27–39

Rousseeuw PJ (1987) Sihouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20:53–65

Examples

```
# Install packages if necessary
# install.packages('MASS')
# install.packages('cluster')

library(MASS)
library(cluster)
library(anocva)

set.seed(2000)

# Defines a k-means function that returns cluster labels directly
myKmeans = function(dist, k){
  return(kmeans(dist, k, iter.max = 50, nstart = 5)$cluster)
}

# Generate simulated data
nitem = 70
sigma = matrix(c(0.04, 0, 0, 0.04), 2)
simuData = rbind(mvrnorm(nitem, mu = c(0, 0), Sigma = sigma ),
             mvrnorm(nitem, mu = c(3,0), Sigma = sigma),
             mvrnorm(nitem, mu = c(2.5,2), Sigma = sigma))

plot(simuData, asp = 1, xlab = '', ylab = '', main = 'Data for clustering')

# Calculate distances and perform {0,1} normalization
distMatrix = as.matrix(dist(simuData))
distMatrix = checkRange01(distMatrix)

# Estimate the optimal number of clusters
r = nClust(meanDist = distMatrix, p = 1, maxClust = 10,
           clusteringFunction = myKmeans, criterion = "silhouette")
sprintf("The optimal number of clusters found was %d.", r)

# K-means Clustering
labels = myKmeans(distMatrix, r)

plot(simuData, col = labels, asp = 1, xlab = '', ylab = '', main = 'K-means clustered data')
```

nClustMulti *Multiple Samples Optimal Number of Clusters Estimation*

Description

Estimates the optimal number of clusters for multiple samples using either Slope or Silhouette criterion. The optimal number of clusters will be verified in the range 2,..., maxClust. Takes the mean of all samples in order to perform the estimation.

Usage

```
nClustMulti(dataDist, p = 1, maxClust = 20, clusteringFunction,
  criterion = c("slope", "silhouette"))
```

Arguments

dataDist	An matrix with n subjects. Each subject has the size of NxN and represents the distances between the elements of the sample.
p	Slope adjust parameter.
maxClust	The maximum number of clusters to be tried.
clusteringFunction	The clustering function to be used.
criterion	The criterion that will be used for estimating the number of clusters. The options are "slope" or "silhouette". If not defined, "slope" will be used.

Value

The optimal number of clusters.

References

Fujita A, Takahashi DY, Patriota AG (2014b) A non-parametric method to estimate the number of clusters. Computational Statistics & Data Analysis 73:27–39

Rousseeuw PJ (1987) Sihouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20:53–65

Examples

```
# Install packages if necessary
# install.packages('MASS')
# install.packages('cluster')

library(anocva)
library(MASS)
library(cluster)

set.seed(5000)
```

```
# A k-means function that returns cluster labels directly.
myKmeans = function(dist, k){
  return(kmeans(dist, k, iter.max = 50, nstart = 5)$cluster)
}

# Number of subjects in each population
nsub = 25
# Number of items in each subject
nitem = 60

# Generate simulated data
data = array(NA, c(nsub, nitem*2, 2))
data.dist = array(NA, c(nsub, nitem*2, nitem*2))
meanx = 2
delta = 0.5
# Covariance matrix
sigma = matrix(c(0.03, 0, 0, 0.03), 2)
for (i in seq(nsub)){
  sub = rbind(mvrnorm(nitem, mu = c(0, 0), Sigma = sigma ),
              mvrnorm(nitem, mu = c(meanx,0), Sigma = sigma))
  data[i,,] = sub
  data.dist[i,,] = as.matrix(dist(data[i,,]))
}

# Estimate the optimal number of clusters
r = nClustMulti(dataDist = data.dist, p = 1, maxClust = 20,
                clusteringFunction = myKmeans, criterion = "slope")
sprintf("The optimal number of clusters found was %d.", r)
```

`spectralClustering` *Spectral clustering*

Description

Unnormalized spectral clustering function. Uses Partitioning Around Medoids clustering instead of K-means.

Usage

```
spectralClustering(W, k)
```

Arguments

`W`	NxN similarity matrix
`k`	Number of clusters

Value

Cluster labels

References

Von Luxburg, U (2007) A tutorial on spectral clustering. Statistics and computing 17:395–416.

Ng A, Jordan M, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: Advances in Neural Information Processing Systems. Dietterich T, Becker S, Ghahramani Z (Eds.), vol. 14. MIT Press, (pp. 849–856).

Examples

```
# Install igraph if necessary
# install.packages('igraph')
# install.packages('cluster')

library(anocva)

set.seed(2000)

if (requireNamespace("igraph", quietly = TRUE)) {

  # Create a tree graph
  treeGraph = igraph::make_tree(80, children = 4, mode = "undirected")

  # Visualize the tree graph
  plot(treeGraph, vertex.size = 10, vertex.label = NA)

  # Get the adjacency matrix of the tree graph
  adj = as.matrix(igraph::get.adjacency(treeGraph))

  # Cluster the tree graph in to four clusters
  cluster = spectralClustering(adj, 4)

  # See the result clustering
  plot(treeGraph, vertex.size=10, vertex.color = cluster, vertex.label = NA)
}
```

Index