Package 'RHclust'

May 6, 2021

Type Package

Title Vectors in Partitioning

Version 1.0.0

Maintainer Joseph Handwerker <jkhndwrk@memphis.edu>

Description

Non-parametric clustering of joint pattern multi-genetic/epigenetic factors. The goal of this package is to cluster subjects based on gene features including single nucleotide polymorphisms (SNPs), DNA methylation (CPG), and gene expression (GE). The novel concept follows the general K-means (Hartigan and Wong (1979) <doi:10.2307/2346830> framework but uses weighted Euclidean distances across the gene features to cluster subjects. This approach is unique in that it attempts to capture all pairwise interactions in an effort to cluster based on their complex biological interactions.

License Artistic-2.0

Encoding UTF-8

Imports Runuran,graphics,stats,utils

NeedsCompilation no

Author Joseph Handwerker [aut, cre],
Lauren Sobral [ctb],
Meredith Ray [aut, ctb]

Repository CRAN

Date/Publication 2021-05-06 09:00:05 UTC

R topics documented:

`BinaryClass` *Binary Classification*

Description

A confusion matrix but allows for anaylsis of non-equal level data classifications.

Usage

```
BinaryClass(x)
```

Arguments

`x`	Can be a data frame dimensions at least 2 rows and 2 columns meant to represent observed and predicted values where the observed (true) values are in the first column and predicted columns in the second column. Can also be a table from 'table()'.

Details

BinaryClass() is similar to a confusion matrix with binary classification outputs. The true positive values per column are identified based on the maximum number of assignments per category.

Value

`Table`	the results of 'table()' on 'x'
`Accuracy`	overall accuracy of classification
`CI`	confidence interval of overall accuracy using Clopper-Pearson Interval
`Group Measures`	the sensitivity, specificity, positive predictive value, negative predictive value, prevelance detection rate, detection prevalence, and balanced accuracy for each class

Author(s)

jkhndwrk@memphis.edu

Examples

```
# Basic example
true = c(rep(1,5), rep(2,5), rep(3,5), rep(4,5))
pred = c(rep(1,4),4,rep(2,5),2,rep(3,4),1,rep(4,4))
df = cbind(true,pred)
dff = table(pred,true)
BinaryClass(df)
BinaryClass(dff)

true = c(rep(1,5), rep(2,5), rep(3,5), rep(4,5))
```

```
pred = c(rep(1,5),rep(2,5),rep(3,10))
df = cbind(true,pred)
dff = table(pred,true)
BinaryClass(df)
BinaryClass(dff)


sd = SimData(k = c(10,40,50))
out = VIP(sd, v = 3, optimize = 'elbow', nstart = 5)
df = out$`BC Test`
out_table = table(df[,2], df[,1])
BinaryClass(df)
BinaryClass(out_table)

## Looping through different clusters

sd = SimData(seed = 1, gene = 1)
acc = NULL
for (i in 1:5){
 out = VIP(sd, v = i, optimize = 'off', nstart = 5)
 acc[i] = BinaryClass(out$`BC Test`)$Accuracy
}

plot(acc, type = 'b', main = 'Accuracy Comparison', xlab = 'Clusters', ylab = 'Acc')
```

SimData *GE, CPG, SNP Simulated Data*

Description

Simulated data generator containing continuous variables representing gene expression (GE) data and DNA methylation data as M-values (GPG), and categorical variable representing single nucleotide polymorphisms (SNP). GE and CPG data are simulated from a normal distribution and SNP data is simulated from a multinomial distribution.

Usage

```
SimData(seed = NULL, gene = 36,
        k = c(33,33,34),
        GEbar = 5, GEsd = 0.5,
        CPGbar = 4, CPGsd = 0.5,
        SameCPG = FALSE, SameSNP = FALSE,
        ProbDist = NULL)
```

Arguments

seed	Set specified seed for reproducibility
gene	Numeric input that specifies the number genes

k	Cluster pattern/distribution across subjects formatted as a vector, i.e. c(33,33,34) representing 33 subjects in the first cluster, 33 in the second cluster, and 34 in the third cluster.
GEbar	Optional numeric input to change the mean distribution of GE data
GEsd	Optional numeric input to change the standard deviation of GE data
CPGbar	Optional numeric input to change the mean distribution of CPG data
CPGsd	Optional numeric input to change the standard deviation of CPG data
SameCPG	Logical value that if set to True sets the distribution of each CPG cluster around the same mean
SameSNP	Logical value that if set to True changes the probability distribution of SNPs to be the same per cluster
ProbDist	Optional list input that allows the change of SNP probability distributions per cluster. Default list stops at 5 cluster distributions. Default ProbDist = list(c(0.50,0.25,0.25), c(0.20,0.55,0.25), c(0.30,0.15,0.55), c(0.20,0.50,0.30), c(0.45,0.20,0.35))

Details

SimData simply creates simulated data that aims to represent real world data for gene expression (GE), DNA methylations (CPG), and single neucleotide polymorphisms (SNP). The goal of this function is to allow the user the ability to manipulate their data for testing of the main VIP() function.

Value

Clusters	Vector of cluster assignment for each subject.
Vec	Numeric representation of values per cluster used for sensitivity measures.
GE	Simulated continuous data for GE. Means of each cluster changes by a factor of 5 with default standard deviation of 0.5.
CPG	Simulated continuous data for CPG. Means of each cluster changes by a factor of 4 with default standard deviation of 0.5.
SNP	Simulated categorical data for SNP.
GE_Index	Index names for GE.
CPG_Index	Index names for CPG.
SNP_Index	Index names for SNP.

Author(s)

jkhndwrk@memphis.edu

Examples

```
# Generating simulated data
sd = SimData()
```

```
## Specifying seed, genes, and clusters
# sd = SimData(seed = 42, gene = 18, c(10,40,50))

# Specifying probability distribution of SNP data
l = list( c(10, 35, 55),
          c(60, 20, 20),
          c(25, 50, 25))
sd = SimData(1, g = 36, ProbDist = l)

sd = SimData(1, g = 36, c(33,33,34))

# Same SNP distribution across 2 clusters
ssnp = SimData(g = 36, k = c(10,90), SameSNP = TRUE)

# Same CpG distribution across 2 cluster
scpg = SimData(g = 36, k = c(10,90), SameCPG = TRUE)
```

VIP *Vectors in Partitioning*

Description

Clustering of subjects based on similar patterns of gene expression, DNA methylation, and SNPs.

Usage

```
VIP(Simulated = NULL, SNP = NULL, CPG = NULL, GE = NULL,
                SNPname = NULL, CPGname = NULL, GEname = NULL, v,
                optimize = c('off','min','slope','elbow'),
                iter_max = 1000, nstart = 5, fit = c('aic','bic'),
                seed = NULL, type = c('Default','NoCPG','NoSNP'),
                ct = c('mean','median'), verbose = FALSE)
```

Arguments

Simulated	set to name of simulated data built from SimData(), else set to NULL for real data.
SNP	Data frame or data matrix containing categorical SNP data. Input must be in form of N x M, with N rows of subjects and M columns of SNPs. Rownames are permitted. Run SimData()$SNP for examples.
CPG	Data frame or data matrix containing numeric CPG data. Input must be in form of N x M, with N rows of subjects and M columns of CPG. Rownames are permitted. Run SimData()$CPG for examples.
GE	Data frame or data matrix containing numeric GE data. Input must be in form of N x M, with N rows of subjects and M columns of GE. Rownames are permitted. Run SimData()$GE for examples.

SNPname	Names for SNP data. Data must be a data frame of Nx2 dimensions with SNP sites as column 1, and GE indexes in column 2. Order of SNPs must match the order of the SNP columns in the argument SNP. See SimData()$SNP_Index for examples.
CPGname	Names for CPG data. Data must be a data frame of Nx2 dimensions with CPG sites as column 1, and GE indexes in column 2. Order of CPGs must match the order of the CPG columns in the argument GE. See SimData()$CPG_Index for examples.
GEname	Names for GE data. Data must be a data frame of Nx2 dimensions with GE sites as column 1, and GE indexes in column 2. Order of GEs must match the order of the GE columns in the argument GE. See SimData()$GE_Index for examples.
v	Numeric scalar or vector of number for clusters, or a range of clusters with format c(l,u) for cluster l:u
optimize	Returned the optimal number of clusters. Input 'min' returns cluster assignment with lowest WSS for clusters in v. Input 'slope' indicates whether the algorithm should pick the lowest WSS value based on the first increasing slope. Input 'elbow' fits a line between the first and last fitted WSS and finds the corresponding cluster with the maximum distance to that line. All but 'slope' return plots.
iter_max	Maximum number of iterations allowed.
nstart	If nstart > 1, repetitive computations with random initializations are computed and the result with minimum tot_dist is returned.
fit	Penalizing factor for WSS of clusters. Can be set to either 'aic' or 'bic'.
seed	Optional input to sample the same initial cluster centers.
type	Optional input for special cases for data without CPGs or SNP inputs. Options include "Default", "NoSNP", or "NoCPG"
ct	Central tendency option for cluster assignment. Options include 'mean' or 'median'.
verbose	Logical whether information about the cluster procedure should be given.

Details

Similar to k-means and k-proto clustering, this algorithm computes clusters based on weighted factors of mixed data relative to genetic/epigenetic data. Clusters are assigned using sumed euclidean distance of numerics (*GE* and *CPG*) weighted by matching categorical (*SNP*) data. Central tendancy of numeric data can be set to either mean or median with input *ct*.

Data must be ordered such that rows in each data set correspond to the same subject and order of the indexes match the order of the columns in the data. The current algorithm does not allow for any missing data. The aim is for *GE*, *CPG*, and *SNP* data to be clustered into *v* groups such that within sum of squares is minimized. If groups of clusters are close, the algorithm may not converge correctly and signals a warning if cluster size is reduced.

Optimization functionality was used for simulated data analysis, but is allowed for user exploratory analysis as well. *'min'* simply returns the lowest fitted WSS *fit* parameter. *'slope'* loops through clusters in *v* and returns the cluster based on the first increasing slope of fitted WSS. For example, if AIC output is c(100,80,35,50), cluster 3 would be returned since the slope increases from 3 to 4. If there is no increasing slope, the *'min'* optimizer will be returned. *'elbow'* seeks to find the

elbow of the plot based on saturation point. This worked the best for simulation studies but requires more clusters to make proper predictions, in our case it required a range of at least 5 clusters c(1,5) to search to correctly identify the 3 simulated clusters. . For ease of exploratory analysis, v=1 is allowed.

Value

`size`	Number of subjects assigned to each cluster.
`cluster`	Vector of cluster assignment.
`GECenters`	Matrix of cluster centers for GE.
`CPGCenters`	Matrix of cluster centers for CPG.
`SNPCenters`	Matrix of cluster centers for SNP.
`within`	Vector of within cluster sum of squares with one component per cluster.
`tot_within`	Sumed total of within-cluster sum of squares.
`Moved`	Number of iterations before convergence.
`AIC`	Value of tot_within with aic penalizer.
`BIC`	Value of tot_within with bic penalizer.
`outputPlot`	Returns the tot_within, aic, bic, and v values for ploting.

Author(s)

jkhndwrk@memphis.edu

References

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. Applied Statistics, 28, 100–108. 10.2307/2346830.

Examples

```
## simple output of 3 clusters assignments
sd = SimData(1, g = 36, c(33,33,34))
VIPout = VIP(sd, v = 3)

# loop through clusters 1-10 and outputs plot of WSS, AIC, and BIC
VIPout = VIP(sd, v = c(1,10))

# loop through clusters 1-10 but picks first instance of increasing slope
VIPout = VIP(sd, v = c(1,10), optimize = 'slope')


# Individual inputs
sd = SimData(1, g = 36, k = c(33,33,34))
VIPout = VIP(SNP = sd$SNP, CPG = sd$CPG, GE = sd$GE,
            SNPname = sd$SNP_Index, CPGname = sd$CPG_Index,
            GEname = sd$GE_Index,
            v = c(1,5), optimize = 'off', nstart = 5)
```

```
## Varying clusters
sd = SimData(k = c(10,40,50))
out = VIP(sd, v = c(1,6), optimize = 'elbow', nstart = 30)
```

VIPnoCPG *Vectors in Partitioning without CPG data*

Description

Clustering of subjects based on similar patterns of gene expression and SNPs.

Usage

```
VIPnoCPG(Simulated = NULL, SNP = NULL, GE = NULL,
                SNPname = NULL, GEname = NULL, v,
                optimize = c('off','min','slope','elbow'),
                iter_max = 1000, nstart = 5, fit = c('aic','bic'),
                seed = NULL, ct = c('mean','median'), verbose = FALSE)
```

Arguments

Simulated	set to name of simulated data built from SimData(), else set to NULL for real data.
SNP	Data frame or data matrix containing categorical SNP data. Input must be in form of N x M, with N rows of subjects and M columns of SNPs. Rownames are permitted. Run SimData()$SNP for examples.
GE	Data frame or data matrix containing numeric GE data. Input must be in form of N x M, with N rows of subjects and M columns of GE. Rownames are permitted. Run SimData()$GE for examples.
SNPname	Names for SNP data. Data must be a data frame of Nx2 dimensions with SNP sites as column 1, and GE indexes in column 2. Order of SNPs must match the order of the SNP columns in the argument SNP. See SimData()$SNP_Index for examples.
GEname	Names for GE data. Data must be a data frame of Nx2 dimensions with GE sites as column 1, and GE indexes in column 2. Order of GEs must match the order of the GE columns in the argument GE. See SimData()$GE_Index for examples.
v	Numeric scalar or vector of number for clusters, or a range of clusters with format c(l,u) for cluster l:u
optimize	Returned the optimal number of clusters. Input 'min' returns cluster assignment with lowest WSS for clusters in v. Input 'slope' indicates whether the algorithm should pick the lowest WSS value based on the first increasing slope. Input 'elbow' fits a line between the first and last fitted WSS and finds the corresponding cluster with the maximum distance to that line. All but 'slope' return plots.

iter_max	Maximum number of iterations allowed.
nstart	If nstart > 1, repetitive computations with random initializations are computed and the result with minimum tot_dist is returned.
fit	Penalizing factor for WSS of clusters. Can be set to either 'aic' or 'bic'.
seed	Optional input to sample the same initial cluster centers.
ct	Central tendency option for cluster assignment. Options include 'mean' or 'median'.
verbose	Logical whether information about the cluster procedure should be given.

Details

The details are outlined in the main VIP() function. The only difference in this function is the absence of CPG data.

Value

size	Number of subjects assigned to each cluster.
cluster	Vector of cluster assignment.
GECenters	Matrix of cluster centers for GE.
SNPCenters	Matrix of cluster centers for SNP.
within	Vector of within cluster sum of squares with one component per cluster.
tot_within	Sumed total of within-cluster sum of squares.
Moved	Number of iterations before convergence.
AIC	Value of tot_within with aic penalizer.
BIC	Value of tot_within with bic penalizer.
outputPlot	Returns the tot_within, aic, bic, and v values for ploting.

Author(s)

jkhndwrk@memphis.edu

References

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. Applied Statistics, 28, 100–108. 10.2307/2346830.

Examples

```
# No CPG data
sd = SimData()
noCPGout = VIP(sd, v = c(1,5), optimize = 'off', nstart = 30, type = 'NoCPG')

noCPGout = VIPnoCPG(sd, v = c(1,5), optimize = 'off', nstart = 30)
```

VIPnoSNP *Vectors in Partitioning without SNP data*

Description

Clustering of subjects based on similar patterns of gene expression and DNA methylation.

Usage

```
VIPnoSNP(Simulated = NULL, CPG = NULL, GE = NULL,
                CPGname = NULL, GEname = NULL, v,
                optimize = c('off','min','slope','elbow'),
                iter_max = 1000, nstart = 5, fit = c('aic','bic'),
                seed = NULL, ct = c('mean','median'), verbose = FALSE)
```

Arguments

Simulated	set to name of simulated data built from SimData(), else set to NULL for real data.
CPG	Data frame or data matrix containing numeric CPG data. Input must be in form of N x M, with N rows of subjects and M columns of CPG. Rownames are permitted. Run SimData()$CPG for examples.
GE	Data frame or data matrix containing numeric GE data. Input must be in form of N x M, with N rows of subjects and M columns of GE. Rownames are permitted. Run SimData()$GE for examples.
CPGname	Names for CPG data. Data must be a data frame of Nx2 dimensions with CPG sites as column 1, and GE indexes in column 2. Order of CPGs must match the order of the CPG columns in the argument GE. See SimData()$CPG_Index for examples.
GEname	Names for GE data. Data must be a data frame of Nx2 dimensions with GE sites as column 1, and GE indexes in column 2. Order of GEs must match the order of the GE columns in the argument GE. See SimData()$GE_Index for examples.
v	Numeric scalar or vector of number for clusters, or a range of clusters with format c(l,u) for cluster l:u
optimize	Returned the optimal number of clusters. Input 'min' returns cluster assignment with lowest WSS for clusters in v. Input 'slope' indicates whether the algorithm should pick the lowest WSS value based on the first increasing slope. Input 'elbow' fits a line between the first and last fitted WSS and finds the corresponding cluster with the maximum distance to that line. All but 'slope' return plots.
iter_max	Maximum number of iterations allowed.
nstart	If nstart > 1, repetitive computations with random initializations are computed and the result with minimum tot_dist is returned.
fit	Penalizing factor for WSS of clusters. Can be set to either 'aic' or 'bic'.
seed	Optional input to sample the same initial cluster centers.

ct	Central tendency option for cluster assignment. Options include 'mean' or 'median'.
verbose	Logical whether information about the cluster procedure should be given.

Details

The details are outlined in the main VIP() function. The only difference in this function is the absence of SNP data.

Value

size	Number of subjects assigned to each cluster.
cluster	Vector of cluster assignment.
GECenters	Matrix of cluster centers for GE.
CPGCenters	Matrix of cluster centers for CPG.
within	Vector of within cluster sum of squares with one component per cluster.
tot_within	Sumed total of within-cluster sum of squares.
Moved	Number of iterations before convergence.
AIC	Value of tot_within with aic penalizer.
BIC	Value of tot_within with bic penalizer.
outputPlot	Returns the tot_within, aic, bic, and v values for ploting.

Author(s)

jkhndwrk@memphis.edu

References

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. Applied Statistics, 28, 100–108. 10.2307/2346830.

Examples

```
# No SNP data
sd = SimData()
noSNPout = VIP(sd, v = c(1,5), optimize = 'off', nstart = 30, type = 'NoSNP')

noSNPout = VIPnoSNP(sd, v = c(1,5), optimize = 'off', nstart = 30)
```

Index