Package 'RHclust'

May 7, 2026

Type Package

Title Vector in Partition

Version 3.0.0

Maintainer Joseph Handwerker <jkhndwrk@memphis.edu>

Description Non-parametric clustering of joint pattern multi-genetic/epigenetic factors. This package contains functions designed to cluster subjects based on gene features including single nucleotide polymorphisms (SNPs), DNA methylation (CPG), gene expression (GE), and covariate data. 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,Rfast,dplyr,stringr,methods,utils,clue

NeedsCompilation no

Author Joseph Handwerker [aut, ctb, cre],
Allison Plaxco [aut, ctb],
Luke Pirrotta [ctb],
Lauren Sobral [ctb],
Meredith Ray [aut, ctb]

Repository CRAN

Date/Publication 2026-04-18 00:00:02 UTC

Contents

`BinaryClass` *Binary Classification*

Description

Evaluates cluster or classification accuracy when predicted and true labels do not share a common encoding. Builds a confusion matrix and uses label assignment (Hungarian or greedy) to optimally align predicted clusters to true classes before computing performance metrics.

Usage

```
BinaryClass(x, method = c("hungarian", "greedy"))
```

Arguments

`x`	A two-column data frame or matrix with true labels in the first column and predicted labels in the second, or a pre-computed `table` of true-by-predicted counts.
`method`	Assignment algorithm used to match predicted cluster labels to true class labels. `"hungarian"` (default) finds the globally optimal one-to-one assignment while `"greedy"` iteratively picks the highest-count cell.

Details

Because clustering algorithms assign arbitrary integer labels, a predicted cluster `2` may correspond to true class `1`. `BinaryClass()` resolves this by treating label alignment as an assignment problem on the confusion matrix. After alignment, per-class true positives are read from the assigned cells and used to derive sensitivity and specificity for each class.

Value

`Table`	Confusion table produced by `table(predicted, true)`.
`Accuracy`	Overall proportion of correctly assigned observations after label alignment.
`Group Measures`	Matrix of per-class `Sensitivity` and `Specificity` (columns = classes).
`Assignment`	Matrix of matched `(pred, true)` index pairs from the chosen assignment method.
`Method`	Character string indicating which assignment method was used.

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)
BinaryClass(df)


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)
BinaryClass(df)



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


## 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')
```

---

gower *VIP extension to grouped data using Gower's similarity*

Description

Clustering of subjects based on similar patterns of variables with a grouped structure, using a distance measure based on Gower's similatiry.

Usage

```
gower(study_data = NULL, index = 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

study_data	Data frame or data matrix containing grouped data to be used for clustering. Input must be in form of N x M, with N rows of subjects and M columns of data variables.
index	Data frame with information that identifies which variables belong to which constructs, and what type each varibale is (nominal, ordinal, or numeric). This dataframe should have 3 columns named "construct", "var", and "format". Each variable in the study_data dataframe should be reflected in the "var" column, with names matching exactly, in the same order that they appear in the study_data dataframe. The "construct" column must denote which group each variable belongs to, and the "format" column must denote which the type of variable it is as either "nominal", "ordinal", or "numeric".
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 Total_WSS 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

Building upon the original VIP function, this algorithm computes clusters based on weighted factors of mixed data by extending the original VIP framework to data that has a grouped nature. Clusters are assigned using a distance calculated based on Gower's similarity within each the construct. Central tendancy of numeric data can be set to either mean or median with input ct.

Data must be ordered such that rows in the data set used for clustering are in the same order as the corresponding index file columns. The current algorithm does not allow for any missing data. The aim is for individuals to be clustered into v groups, based on the value of grouped-variable data, 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.

For consistiency with the original VIP options, 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.
`Clusters`	Vector of cluster assignment.
`Construct_c_Centers`	Matrix of cluster centers for construct "c"". There will be one of these outputs for each construct, and the value of "c"" will denote the name of the construct as reflected in the index and study data.
`WSS`	Vector of within cluster sum of squares with one component per cluster.
`Total_WSS`	Sumed total of within-cluster sum of squares.
`AIC`	Value of Total_WSS with aic penalizer.
`BIC`	Value of Total_WSS with bic penalizer.
`PlotData`	Returns the Total_WSS, AIC, BIC, and v values for ploting.

Author(s)

aplaxco@memphis.edu maray@memphis.edu jkhndwrk@memphis.edu

Examples

```
#create simulated grouped data
study_data <- SimulatedGrouped()

#create an index file based on the simulated grouped data
index <- IndexGrouped(study_data)

#cluster into 3 groups using VIP extension to grouped data using Gower's similarity
gower(study_data, index, v = 3, optimize = "elbow", iter_max = 1000, nstart = 5,
                fit = "aic", seed = NULL, ct = "mean",
                verbose = FALSE)
```

`IndexGrouped` *Index file for simulated grouped data for use with VIP extensions to grouped data*

Description

This function creates an index file to accompany data produced using the SimulatedGrouped function.

Usage

```
IndexGrouped(study_data)
```

Arguments

study_data	Data file produced using the SimulatedGrouped function.

Details

The VIP extension to grouped data functions require a dataframe input with the applicable study data as well as an index file that names which variables are in which construct. This function produced an index file based on simulated grouped data created using the SimulatedGrouped function.

Value

This function returns an index file corresponding to the simulated grouped data created using the SimulatedGrouped function.

Author(s)

aplaxco@memphis.edu maray@memphis.edu jkhndwrk@memphis.edu

Examples

```
#create simulated grouped data
study_data <- SimulatedGrouped()

#create an index file based on the simulated grouped data
IndexGrouped(study_data)
```

SimData *GE, CPG, SNP, and Covariate Simulated Data*

Description

Simulated data generator containing continuous variables representing gene expression (GE) data and DNA methylation data as M-values (CPG), 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. Covariate data can have uniformly distributed data or normally distributed. Cluster separation can also be distinct or non-distinct.

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, SameGeneDist = TRUE,
        distinct=TRUE)
```

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 10 cluster distributions. Default problist = 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), c(0.26,0.52,0.22), c(0.23,0.21,0.56), c(0.23,0.52,0.25), c(0.51,0.25,0.24), c(0.21,0.58,0.21))
SameGeneDist	Logical that set the covariates to follow the genetic clustering scheme where the data is uniformly distributed amongst each subgroup in each cluster. Or if set to FALSE sets the covariates to be noisy and have no discernible groups.
distinct	Logical that sets the covariate data clusters to be distinct at default, or non-distinct when set to FALSE

Details

SimData creates simulated data that aims to represent real world data for gene expression (GE), DNA methylations (CPG), and single neucleotide polymorphisms (SNP). Covariate data is used to represent other potentially useful health data to further weight the genetics data. The goal of this function is to allow the user the ability to manipulate their data for testing of the VIP() or VIPcov() functions.

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.
Covariates	Simulated data for covariate data. By default contains 10 columns of cateogrical data and 10 columns of numeric data.

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)
```

`SimulatedGrouped` *Simulated grouped data for use with VIP extensions to grouped data*

Description

Simulated data generator that produces grouped data with multiple "constructs", or groups of varibales, containing a mix of both continuous variables and categorical variable. Continuous variables are simulated from a normal distribution and categorical variables are simulated from a multinomial distribution.

Usage

```
SimulatedGrouped(base = 10, cons = 3, larger = 1, data.seed = NULL)
```

Arguments

`base`	Base number of variables per construct
`cons`	Total number of constructs
`larger`	Number of larger constructs. Larger constructs will have base*3 number of variables. There must be at least 1 but less that the total number of constructs that are larger.
`data.seed`	Set specified seed for reproducibility

Details

The goal of this function is to allow the user to create example data to use with the VIP extensions to grouped data, to illustrate how the function works. Users can specify the base number of variables per construct, "base" and the number of constructs, "cons". This function is designed to simulate data with unbalanced constructs, so a user specified number of constructs have 3 times as many variables as the others. Others have the "base" number of variables. 20 percent of variables in each construct are "associated with clustering", meaning that they hold the clustering pattern. By default, the function generates data a "base" of 10 variables, 3 constructs, and one that is larger than the rest. Data is randomly generated, but users may set the data.seed option for reproducibility.

Value

The function returns a dataframe with 100 observations, and the base number of variables, constructs, and the number of larger constructs specified by the user.

Author(s)

aplaxco@memphis.edu maray@memphis.edu jkhndwrk@memphis.edu

Examples

```
#Generating simulated data using the function defaults
#(three total constructs, base 10 variables, one larger construct)
study_data <- SimulatedGrouped()

#Users can change the base number of variables
study_data <- SimulatedGrouped(base = 100)

#Users can change the total number of constructs
study_data <- SimulatedGrouped(cons = 5)

#Users can change the number of larger constructs
study_data <- SimulatedGrouped(larger = 2)

#Users can set the seed for data reporducibility
study_data <- SimulatedGrouped(data.seed = 1)

#Users can manipulate all of these options
study_data <- SimulatedGrouped(base = 100, cons = 5, larger = 2, data.seed = 1)
```

VIP *Vector in Partition*

Description

Clustering of subjects based on similar patterns of gene expression, DNA methylation, and SNPs. This work was funded by grant 1R03AI128227-01A1.

Usage

```
VIP(Simulated = NULL, SNP = NULL, CPG = NULL, GE = NULL,
    SNPname = NULL, CPGname = NULL, GEname = NULL, v,
    optimize = c('off','min','slope','elbow','ssi','all'),
    iter_max = 1000, nstart = 5, fit = c('wss','aic','bic','alt'),
    ALTformula = NULL, 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. Input 'ssi' to utilize the simplified silhouette for cluster assignment. The 'all' input returns the cluster assignment for all available inputs. All but 'slope' return plots.
ALTformula	This is a built in option to allow the user to specify their own penalizing formula. The required inputs are WSS and k as the penalizer is based on the within sums of squares and the cluster number
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. The SSI optimization criterion returns the clustering solution that maximizes the mean simplified silhouette score across the specified range of clusters k. Where traditional silhouette requires the average of all pairwise distances between an observation and all other observations in the two nearest clusters, this simplified version substitutes pairwise distance between observations and the two nearest cluster centers. As this distance calculation is already implemented in the assignment step of k-partition clustering, this method provides meaningful assessment of cluster separation with minimal additional cost. The overall SSI score is the mean of these individual values across all observations. In the context of this VIP algorithm, SSI values range from -1 to 1, with higher scores indicating better separation between clusters. A score approaching 0 suggests that an observation lies approximately midway between its assigned cluster and the nearest alternative cluster, while scores approaching 1 indicate clear membership in the assigned cluster. For a single cluster (k=1), or unusual cases where two clusters share an identical centroid, the SSI score defaults to 0. SSI performs best when convex, spherical, and non-nested clusters are expected. Performance declines when clusters are non-convex, irregular, or extremely noisy. 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)
```

VIPcov *Vector in Partition with covariates*

Description

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

Usage

```
VIPcov(Simulated = NULL, SNP = NULL, CPG = NULL, GE = NULL,
       SNPname = NULL, CPGname = NULL, GEname = NULL,COV = NULL, v,
       COVname = NULL, COVdist = c('euclid','gower'),
     lambda = NULL, gamma = 1, optimize = c('off','min','slope','elbow','ssi','all'),
       iter_max = 1000, nstart = 5, fit = c('wss','aic','bic','alt'),
       ALTformula = NULL, 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.
COV	Data frame or matrix containing numeric and or categorical variables used for weighting the distance matrix of the genetics data.
COVname	Data frame or data matrix containing the variable types of the covariates. Variable types can be categorical, numeric, or ordinal.
COVdist	Optional parameter for calculating the distance of the covariates. Options include euclidean distance or the gower similarity.
v	Numeric scalar or vector of number for clusters, or a range of clusters with format c(l,u) for cluster l:u
lambda	Weighting factor for the covariate data. The default value is calculated by taking the average variance of the numeric variables divided by the average concentraion of the factors for the categorical variables. When non-mixed type data are used, the value is set to 1 unless specified otherwise.
gamma	Weighting factor for the genetics distance measure. Default is set to 1
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. Input 'ssi' to utilize the simplified silhouette for cluster assignment. The 'all' input returns the cluster assignment for all available inputs. All but 'slope' return plots.

ALTformula	This is a built in option to allow the user to specify their own penalizing formula. The required inputs are WSS and k as the penalizer is based on the within sums of squares and the cluster number
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

This method is an extension of the VIP() function and allows for the clustering method to be further weighted by the covarite distance. The covariate distance measure is similar to that seen in *kproto()* where the euclidian measure of the numerics data is weighted by matching categorical variables and a weighting factor. Clsuters are assigned based on these two distance measures where the 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 after being weighted by covariates 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.

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.
Adjusted	Logical if the genetics distance measure is weighted by covariates.
Lambda	Lambda value used to weight the covariate distance measure.
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.

Z. Huang. (1998) Extensions to the k-means algorithm for clustering large data sets with categorical variables. Data Mining and Knowledge Discovery, 2:283–304, doi: 10.1023/A:1009769707641.

Szepannek G (2018). "clustMixType: User-Friendly Clustering of Mixed-Type Data in R." The R Journal, 200-208. doi:10.32614/RJ-2018-048, https://doi.org/10.32614/RJ-2018-048.

Examples

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

# loop through clusters 1-5 and outputs plot of WSS, AIC, and BIC
VIPout = VIPcov(sd, v = c(1,5), COV = sd$Covariates)

# loop through clusters 1-5 but picks first instance of increasing slope
VIPout = VIPcov(sd, v = c(1,5), optimize = 'slope', COV = sd$Covariates)


# Individual inputs
sd = SimData(1, g = 36, k = c(33,33,34))
VIPout = VIPcov(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, COV = sd$Covariates)


## Varying clusters
sd = SimData(k = c(10,40,50))
out = VIPcov(sd, v = c(1,5), optimize = 'elbow', nstart = 10, COV = sd$Covariates)
```

VIPlong *Vector in Partition Longitudinal*

Description

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

Usage

```
VIPlong(Simulated = NULL,Simulated1 = NULL, SNP = NULL, CPG = NULL, GE = NULL,
        SNPname = NULL, CPGname = NULL, GEname = NULL, v,
        compositedist = c("sum", "logsum"),
        optimize = c('off','min','slope','elbow','ssi','all'),
        iter_max = 1000, nstart = 5, fit = c('wss','aic','bic','alt'),
        ALTformula = NULL, change=c('observed','relative','log'),
        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. This set acts as time point 1 for the longitudinal aspect of the distance measures.
Simulated1	set to name of simulated data built from SimData(), else set to NULL for real data. This set acts as time point 2 for the longitudinal aspect of the distance measures.
change	Specifies how the difference between time point 1 and time point 2 are calculated. If 'observed' is chosen the change is calculated as T2 - T1. If 'relative' is chosen the change is calculated as T2-T1/T1.
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
compositedist	Specifies how the numerator aggregate combines. Sum simply takes the sum of each numerator component while logsum takes the log of each component before total summation.

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. Input 'ssi' to utilize the simplified silhouette for cluster assignment. The 'all' input returns the cluster assignment for all available inputs. All but 'slope' return plots.
ALTformula	This is a built in option to allow the user to specify their own penalizing formula. The required inputs are WSS and k as the penalizer is based on the within sums of squares and the cluster number
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

This method is an extension of the VIP() function and allows for the clustering method to take into account longitudinal aspects of the data. This method currently accounts for two time points and the distance can be the composite distances for each component or cluster based on the observed/relative differences. The composite distance takes the sum of the euclidean distances for each time point and their respective cluster and takes the aggregate before weighting by the SNP component which only accounts for one timepoint. The observed/relative clustering takes the difference between and calculates the clusters using the original VIP() method but with the inputs being the difference between each timepoint.

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
t1 = SimData(1, g = 36, c(33,33,34))
t2 <- t1

# Add some slight variation
t2$GE <- t1$GE + rnorm(length(t1$GE), mean = 0, sd = 0.1)
t2$CPG <- t1$CPG + rnorm(length(t1$CPG), mean = 0, sd = 0.1)

Sample <- c(1:10)

t2$GE[Sample,] <- t2$GE[Sample,]*4
t2$CPG[Sample,] <- t2$CPG[Sample,]*4

# Sum Composite
out <- VIPlong(Simulated = t1,Simulated1=t2, v = 2, nstart = 10, compositedist = "sum")
out$`BC Test`[Sample,1] <- out$`BC Test`[Sample,1]*10

# Log sum composite
out <- VIPlong(Simulated = t1,Simulated1=t2, v = 2, nstart = 10, compositedist = "logsum")
out$`BC Test`[Sample,1] <- out$`BC Test`[Sample,1]*10

# Observed difference
out <- VIPlong(Simulated = t1,Simulated1=t2, v = 2, nstart = 10, change="observed")
out$`BC Test`[Sample,1] <- out$`BC Test`[Sample,1]*10

# Relative difference
out <- VIPlong(Simulated = t1,Simulated1=t2, v = 2, nstart = 10, change="relative")
out$`BC Test`[Sample,1] <- out$`BC Test`[Sample,1]*10
```

VIPnoCPG *Vector in Partition 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','ssi','all'),
        iter_max = 1000, nstart = 5, fit = c('wss','aic','bic','alt'),
       ALTformula = NULL, 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. Input 'ssi' to utilize the simplified silhouette for cluster assignment. The 'all' input returns the cluster assignment for all available inputs. All but 'slope' return plots.
ALTformula	This is a built in option to allow the user to specify their own penalizing formula. The required inputs are WSS and k as the penalizer is based on the within sums of squares and the cluster number.
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 *Vector in Partition 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','ssi','all'),
        iter_max = 1000, nstart = 5, fit = c('wss','aic','bic','alt'),
       ALTformula = NULL, 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. Input 'ssi' to utilize the simplified silhouette for cluster assignment. The 'all' input returns the cluster assignment for all available inputs. All but 'slope' return plots.
ALTformula	This is a built in option to allow the user to specify their own penalizing formula. The required inputs are WSS and k as the penalizer is based on the within sums of squares and the cluster number.
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)
```

---

vipx *VIP extension to grouped data*

Description

Clustering of subjects based on similar patterns of variables with a grouped structure, using a distance measure based on the distance employed by the original VIP function.

Usage

```
vipx(study_data = NULL, index = 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

study_data	Data frame or data matrix containing grouped data to be used for clustering. Input must be in form of N x M, with N rows of subjects and M columns of data variables.
index	Data frame with information that identifies which variables belong to which constructs, and what type each varibale is (nominal, ordinal, or numeric). This dataframe should have 3 columns named "construct", "var", and "format". Each variable in the study_data dataframe should be reflected in the "var" column, with names matching exactly, in the same order that they appear in the study_data dataframe. The "construct" column must denote which group each variable belongs to, and the "format" column must denote which the type of variable it is as either "nominal", "ordinal", or "numeric".
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 Total_WSS 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

Building upon the original VIP function, this algorithm computes clusters based on weighted factors of mixed data by extending the original VIP framework to data that has a grouped nature. Clusters are assigned using summed euclidean distance of numeric variables within each construct weighted by the percent distance with respect to categorical data in the construct. Central tendancy of numeric data can be set to either mean or median with input ct.

Data must be ordered such that rows in the data set used for clustering are in the same order as the corresponding index file columns. The current algorithm does not allow for any missing data. The aim is for individuals to be clustered into v groups, based on the value of grouped-variable data,

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.

For consistiency with the original VIP options, 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.
`Clusters`	Vector of cluster assignment.
`Construct_c_Centers`	Matrix of cluster centers for construct "c"". There will be one of these outputs for each construct, and the value of "c"" will denote the name of the construct as reflected in the index and study data.
`WSS`	Vector of within cluster sum of squares with one component per cluster.
`Total_WSS`	Sumed total of within-cluster sum of squares.
`AIC`	Value of Total_WSS with aic penalizer.
`BIC`	Value of Total_WSS with bic penalizer.
`PlotData`	Returns the Total_WSS, AIC, BIC, and v values for ploting.

Author(s)

aplaxco@memphis.edu maray@memphis.edu jkhndwrk@memphis.edu

Examples

```
#create simulated grouped data
study_data <- SimulatedGrouped()

#create an index file based on the simulated grouped data
index <- IndexGrouped(study_data)

#cluster into 3 groups using VIP extension to grouped data using Gower's similarity
vipx(study_data, index, v = 3, optimize = "elbow", iter_max = 1000, nstart = 5,
                fit = "aic", seed = NULL, ct = "mean",
                verbose = FALSE)
```

Index