Package 'vimpclust'

January 8, 2021

Type Package

Title Variable Importance in Clustering

Version 0.1.0

Description An implementation of methods related to sparse clustering and variable importance in clustering. The package currently allows to perform sparse k-means clustering with a group penalty, so that it automatically selects groups of numerical features. It also allows to perform sparse clustering and variable selection on mixed data (categorical and numerical features), by preprocessing each categorical feature as a group of numerical features. Several methods for visualizing and exploring the results are also provided. M. Chavent, J. Lacaille, A. Mourer and M. Olteanu (2020)<https://www.esann.org/sites/default/files/proceedings/2020/ES2020-103.pdf>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports PCAmixdata, ggplot2, Polychrome, mclust, rlang

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Alex Mourer [aut],
Marie Chavent [aut, ths],
Madalina Olteanu [aut, ths, cre]

Maintainer Madalina Olteanu <madalina.olteanu@dauphine.psl.eu>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2021-01-08 09:30:03 UTC

R topics documented:

`DataMice` *Mice Protein Expression Data Set*

Description

The data set consists of the expression levels of 68 proteins that produced detectable signal in the nuclear fraction of cortex for a sample of 72 mice. There are 38 control mice and 34 trisomic mice. Several measurements were recorded for each protein and for each mouse. The measurements containing missing observations in the original data were suppressed, so that one has between 12 and 15 measurements per protein and per mouse.

Mice may be further described based on the treatment they received (injected with memantine or saline), and on their behaviour (stimulated to learn or not).

Usage

```
DataMice
```

Format

A data frame of 72 rows (mice) and 905 columns (variables):

Fields

`Protein_X_meas_Y` Numerical. The expression level for protein X at measurement Y. X has values between 1 and 68, Y has values between 1 and 12 or 15, according to the number of measurements.

`Genotype` Categorical. Two values: "Control" and "Ts65Dn" (trisomic mouse).

`Treatment` Categorical. Two values: "Memantine" and "Saline".

`Behaviour` Categorical. Two values: "C/S" (stimulated to learn) and "S/C" (not stimulated to learn).

`Class.mouse` Categorical. This variables creates eight classes of mice, based on crossing the categories of `Genotype`, `Behaviour` and `Treatment`.

`MouseID` Factor. The key variable identifying each mouse in the sample.

Source

https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression

References

C. Higuera, K.J. Gardiner, and K.J. Cios (2015) Self-organizing feature maps identify proteins critical to learning in a mouse model of Down syndrome. PLoS ONE 10(6): e0129126.

`groupsoft` *Group soft-thresholding operator*

Description

This function implements the group soft-thresholding operator for a vector which elements are priorly split into groups. For the complete mathematical formulation, the reader may refer to the references below.

Usage

```
groupsoft(b, lambda, index = 1:length(b), sizegroup = TRUE)
```

Arguments

`b`	a numerical vector.
`lambda`	a positive scalar containing the regularization parameter.
`index`	a vector of integers of size `length(b)` containing the group membership for each element of `b`. By default, `index=1:length(b)` i.e. each element of `b` constitutes its own group.
`sizegroup`	a boolean. if TRUE, the size of the groups is taken into account in the thresholding operation.

Value

Returns the sparse vector after the group soft-thresholding operation.

References

M. Chavent, J. Lacaille, A. Mourer and M. Olteanu (2020). Sparse k-means for mixed data via group-sparse clustering, to appear in ESANN proceedings.

M. Yuan and Y. Lin (2006). Model selection and estimation in regression with grouped variables. J. R. Statist. Soc. B, Vol. 68(1), p. 49-67.

See Also

`groupsparsewkm`

Examples

```
b <- c(0.1, 0.2, 0.8, 0.1, 0.1, 0.3)
index <- c(1,1,2,2,3,3)
lambda <- 0.1
groupsoft(b=b, lambda=lambda, index=index, sizegroup=TRUE)
lambda <- 0.3
groupsoft(b=b, lambda=lambda, index=index, sizegroup=TRUE)
lambda <- 0.8
groupsoft(b=b, lambda=lambda, index=index, sizegroup=TRUE)
```

---

groupsparsewkm *Group-sparse weighted k-means*

Description

This function performs group-sparse weighted k-means on a set of observations described by numerical variables organized in groups. It generalizes the sparse clustering algorithm introduced by Witten & Tibshirani (2010) to groups. While the algorithm clusters the observations, the groups of variables are supposed priorly known. The algorithm computes a series of weights associated to the groups of variables, the weights indicating the importance of each group in the clustering process.

Usage

```
groupsparsewkm(
  X,
  centers,
  lambda = NULL,
  nlambda = 20,
  index = 1:ncol(X),
  sizegroup = TRUE,
  nstart = 10,
  itermaxw = 20,
  itermaxkm = 10,
  scaling = TRUE,
  verbose = 1,
  epsilonw = 1e-04
)
```

Arguments

X	a numerical matrix or a dataframe of dimension n (observations) by p (variables).
centers	an integer representing the number of clusters.
lambda	a vector of numerical values (or a single value) providing a grid of values for the regularization parameter. If NULL (by default), the function computes its own lambda sequence of length nlambda (see details).

nlambda	an integer indicating the number of values for the regularization parameter. By default, nlambda=20.
index	a vector of integers of size p providing the group membership for each variable. By default, index=1:ncol(X) i.e. no groups or groups of size 1.
sizegroup	a boolean. If TRUE, the group sizes (number of variables in each group) are taken into account in the penalty term (see details). By default, sizegroup=TRUE.
nstart	an integer representing the number of random starts in the k-means algorithm. By default, nstart=10.
itermaxw	an integer indicating the maximum number of iterations for the inside loop over the weights w. By default, itermaxw=20.
itermaxkm	an integer representing the maximum number of iterations in the k-means algorithm. By default, itermaxkm=10.
scaling	a boolean. If TRUE, variables are scaled to zero mean and unit variance. By default, scaling=TRUE.
verbose	an integer value. If verbose=0, the function stays silent, if verbose=1 (default option), it prints whether the stopping criterion over the weights w is satisfied.
epsilonw	a positive numerical value. It provides the precision of the stopping criterion over w. By default, epsilonw =1e-04.

Details

Group-sparse weighted k-means performs clustering on data described by numerical variables priorly partitionned into groups, and automatically selects the most discriminant groups by setting to zero the weights of the non-discriminant ones.

The algorithm is based on the optimization of a cost function which is the weighted between-class variance penalized by a group L1-norm. The groups must be priorly defined through expert knowledge. If there is no group structure (each group contains one variable only), the algorithm reduces to the sparse weighted k-means introduced in Witten & Tibshirani (2010). The penalty term may take into account the size of the groups by setting sizegroup=TRUE (see Chavent et al. (2020) for further details on the mathematical expression of the optimized criterion). The importance of the penalty term may be adjusted through the regularization parameter lambda. If lambda=0, there is no penalty applied to the weighted between-class variance. The larger lambda, the larger the penalty term and the number of groups with null weights.

The output of the algorithm is three-folded: one gets a partitioning of the data, a vector of weights associated to each group, and a vector of weights associated to each variable. Weights equal to zero imply that the associated variables or the associated groups do not participate in the clustering process.

Since it is difficult to chose the regularization parameter lambda without prior knowledge, the function builds automatically a grid of parameters and finds the partitioning and the vectors of weights associated to each value in the grid.

Note that when the regularization parameter is equal to 0 (no penalty applied), the output is different from that of a regular k-means, since the optimized criterion is a weighted between-class variance and not the between-class variance only.

Value

lambda	a numerical vector containing the regularization parameters (a grid of values).
W	a p by length(lambda) numerical matrix. It contains the weights associated to each variable.
Wg	a L by length(lambda) numerical matrix, where L is the number of groups. It contains the weights associated to each group.
cluster	a n by length(lambda) integer matrix. It contains the cluster memberships, for each value of the regularization parameter.
sel.feat	a numerical vector of the same length as lambda, giving the number of selected variables for each value of the regularization parameter.
sel.groups	a numerical vector of the same length as lambda, giving the number of selected groups of variables for each value of the regularization parameter.
Z	a matrix of size n by p containing the scaled data if scaling=TRUE, and a copy of X otherwise.
bss.per.feature	a matrix of size p by length(lambda). It contains the between-class variance computed for each variable.

References

Witten, D. M., & Tibshirani, R. (2010). A framework for feature selection in clustering. Journal of the American Statistical Association, 105(490), p.713-726.

Chavent, M. & Lacaille, J. & Mourer, A. & Olteanu, M. (2020). Sparse k-means for mixed data via group-sparse clustering, ESANN proceedings.

See Also

plot.spwkm, info_clust

Examples

```
data(iris)
# define two groups of variables:
# "Sepal.Length" and "Sepal.Width" in group 1
# "Petal.Length" and "Petal.Width"  in group 2
index <- c(1, 2, 1, 2)
# group-sparse k-means

out <- groupsparsewkm(X = iris[,-5], centers = 3, index = index)
# grid of regularization parameters
out$lambda
k <- 10
# weights of the variables for the k-th regularization parameter
out$W[,k]
# weights of the groups for the k-th regularization parameter
out$Wg[,k]
# partition obtained with for the k-th regularization parameter
out$cluster[,k]
```

```
# between-class variance on each variable
out$bss.per.feature[,k]
# between-class variance
sum(out$bss.per.feature[,k])/length(index)

# one variable per group (equivalent to sparse k-means)
index <- 1:4 # default option in groupsparsewkm
# sparse k-means
out <- groupsparsewkm(X = iris[,-5], centers = 3, index = index)
# or
out <- groupsparsewkm(X = iris[,-5], centers = 3)
# group weights and variable weights are identical in this case
out$Wg
out$W
```

HDdata *Statlog (Heart) Data Set*

Description

The data consists of 270 patients described by six numerical variables and eight categorical variables.

Usage

```
HDdata
```

Format

A data frame of 270 rows (patients) and 14 columns (variables):

Fields

`age` Numerical. Age in years.

`resting_blood_pressure(trestbps)` Numerical. Resting blood pressure (in mmHg) at hospital admittance.

`maximum_heart_rate_achieved(maxhr).` Numerical. Maximum heart rate achieved during exercise.

`oldpeak` Numerical. ST depression induced by exercise relative to rest.

`number_of_major_vessels_colored_by_fluoroscopy(numv)` Numerical. Number of major vessels (0-3) colored by fluoroscopy.

`sex` Categorical. Sex (1 = male; 0 = female).

`chest_pain_type(cp)` Categorical. 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic.

`fasting_blood_sugar_>_120mg/dl(fbs)` Categorical. 1 = true; 0 = false.

`resting_electrocardiographic_results(restecg)` Categorical. 0: normal, 1: ST-T wave abnormality (T wave inversions and/or ST elevation or depression >0.05mV), 2: showing probable or definite left ventricular hypertrophy by Estes' criteria.

`exercice_induced_angina(exang)` Categorical. 1 = yes; 0 = no.

`the_slope_of_the_peak_exercice_ST_segment(slope)` Categorical. 1: upsloping, 2: flat, 3: downsloping.

`thalassemia(thal)` Categorical. 3: normal blood flow, 6: fixed defect, 7: reversible defect.

`presence_or_absence_of_heart_disease(HD)` Categorical. Absence or presence of a heart disease.

Source

http://archive.ics.uci.edu/ml/datasets/statlog+(heart)

References

A. Frank and A. Asuncion. UCI machine learning repository, statlog (heart) data set, 2010.

`info_clust` *Description of a set of partitions*

Description

This function computes descriptive statistics of the clustering produced with group-sparse weighted k-means on numerical data, or with sparse weighted k-means on mixed data. It displays the average of the numerical variables per cluster, and the relative frequencies of the levels in the categorical variables per cluster.

Usage

```
info_clust(out, which.lambda, X)
```

Arguments

`out`	an object of class `spwkm`.
`which.lambda`	an integer or a vector of integers selecting the clusterings for which summaries are computed.
`X`	a matrix or a data frame. The initial data set.

Details

The values in `which.lambda` must be integers between 1 and `length(out$lambda)`. One may thus select the clusterings corresponding to specific regularization parameters, or the whole set of clusterings obtained for the whole grid of `out$lambda`.

Value

mean.by.clust	a list of numerical matrices. Each matrix contains the mean values of the numerical variables computed per cluster, for a given value of the regularization parameter.
freq.by.clust	a list of numerical matrices. Each matrix contains the relative frequencies of each level associated to categorical variables, computed per cluster and for a given value of the regularization parameter.
lambda	a scalar or a numerical vector. The selected values of the regularization parameter.

See Also

groupsparsewkm, sparsewkm

Examples

```
data(HDdata)
out <- sparsewkm(X = HDdata[,-14], centers = 2)
info_clust(out, which.lambda=c(1,10,20), X = HDdata[,-14])
```

---

plot.spwkm *Plots from a "spwkm" object*

Description

Produces several graphics to help interpreting a spwkm object.

Usage

```
## S3 method for class 'spwkm'
plot(
  x,
  what = "weights.features",
  Which = NULL,
  xtitle = NULL,
  ytitle = NULL,
  title = NULL,
  showlegend = NULL,
  legendtitle = NULL,
  ...
)
```

Arguments

x	An object of class spwkm.
what	A character string indicating which element of x to be plotted. See section "Details" below for further information.
Which	A numerical vector indexing the groups or the variables to be displayed. See section "Details" below for further information.
xtitle	The title of the x-axis.
ytitle	The title of the y-axis.
title	The title of the graphic.
showlegend	A boolean. If showlegend=NULL (default value), the legend is displayed.
legendtitle	The title of the legend.
...	Further arguments to the plot function.

Details

The plot function allows to represent the regularization paths for a grid of values of lambda, as well as several quality criteria associated to the clustering.

For both groupsparsewkm and sparsewkm functions, the following options are available:

If what=weights.features, the regularization paths for the weights associated to the variables are displayed.

If what=sel.features, the graph represents the number of selected variables for each value of the regularization parameter lambda. In the case of sparse weighted k-means for mixed data, categorical variables are represented with dotted lines so that one easily identifies them.

If what=expl.var, the explained variance (computed as the contribution of the between-class variance to the global variance) is displayed. This criterion is computed for all variables in the data set, without taking into account the weights of the group or of the variables.

If what=w.expl.var, the explained weighted variance is computed. The difference with the criterion above is that the weights of the variables are taken into account in the computation. This leads to a criterion which, for large regularization parameters lambda, may be computed on one variable only, if its weight becomes equal to 1 and all the others are discarded.

If what=pen.crit, the graph displays the evolution of the penalized criterion, maximized by the algorithm. This criterion writes as the between-class weighted sum-of-squares, penalized by a group L1-norm. For more details on the mathematical expressions, one may refer to Chavel et al. (2020).

For the outcome of the groupsparsewkm function trained on numerical data only, two more options are available:

If what=weights.groups, the regularization paths for the weights associated to the groups of variables are displayed.

If what=sel.groups, the graph represents the number of selected groups for each value of the regularization parameter lambda.

For the outcome of the sparsewkm function trained on mixed data, two more options are also available:

If what=weights.levels, the regularization paths for the weights associated to the levels of the categorical variables are displayed.

If what=sel.levels, the graph represents the number of selected levels associated to the categorical variables plus the number of selected numerical variables, for each value of the regularization parameter lambda.

If the number of groups in groupsparsewkm or if the number of features in sparsewkm are too large to have easily interpretable graphics, one may select some groups or some variables using the argument Which. Note that when training sparsewkm on mixed data, the initial order of the variables is changed: after the processing step, numerical variables are displayed first, and categorical second. The indexing provided in Which should take this into account (see the Examples section).

Value

p an object of class ggplot.

References

M., Chavent, J. Lacaille, A. Mourer, and M. Olteanu (2020). Sparse k-means for mixed data via group-sparse clustering. To appear in ESANN proceedings.

See Also

sparsewkm, groupsparsewkm

Examples

```
# sparse weighted k-means on mixed data

data(HDdata)
out <- sparsewkm(X = HDdata[,-14], centers = 2)
plot(out, what = "weights.features")
plot(out, what = "weights.levels")
plot(out, what = "sel.features")
plot(out, what = "sel.levels")
plot(out, what = "expl.var")
plot(out, what = "w.expl.var")
plot(out, what = "pen.crit")
# plot the regularization paths for first three variables only
plot(out, what = "weights.features", Which=1:3)

# group sparse weighted k-means on numerical data
data(iris)
index <- c(1, 2, 1, 2)
out <- groupsparsewkm(X = iris[,-5], centers = 3, index = index)
plot(out, what = "weights.groups")
plot(out, what = "weights.features")
plot(out, what = "sel.groups")
plot(out, what = "sel.features")
plot(out, what = "expl.var")
plot(out, what = "w.expl.var")
plot(out, what = "pen.crit")
# plot the regularization paths for the variables in the first group only
plot(out, what = "weights.features", Which=1)
```

recodmix *Recoding mixed data*

Description

This function transforms and scales a dataset with numerical and/or categorical variables. Numerical variables are scaled to zero mean and unit variance. Categorical variables are first transformed into dummy variables according to their levels, and second centered and normalized with respect to the square roots of the relative frequencies of the levels. The complete procedure is described in Chavent et al. (2014).

Usage

```
recodmix(X, renamelevel = FALSE)
```

Arguments

X	a matrix or a dataframe with numerical and/or categorical variables. Categorical variables must be given as factors.
renamelevel	a boolean. If TRUE (default value), the levels of the categorical variables are renamed as 'variable_name=level_name'.

Value

X	a data frame or a matrix. The input data X with reordered columns (numerical first, categorical second).
Z	a data frame. The transformed data matrix with scaled numerical variables and scaled dummy variables coding for the levels.
index	a vector of integers. Contains an implicit partitioning of the transformed variables: each scaled numerical variable represents a group, all scaled dummy variables summarizing the levels of a categorical variable represent a group. index allows to preserve the information on the initial structure of the data, particularly for categorical variables.

References

M. Chavent, V. Kuentz-Simonet, A. Labenne and J. Saracco (2014). Multivariate analysis of mixed data: the PCAmixdata R package, arXiv:1411.4911.

Examples

```
head(HDdata)
out <- recodmix(HDdata[,-14], renamelevel=TRUE)
# reordered data (numerical/categorical)
colnames(out$X)
# transformed and scaled data
colnames(out$Z)
```

```
# transformed variables partitioning and group membership
out$index
```

`sparsewkm` *Sparse weighted k-means*

Description

This function performs sparse weighted k-means on a set of observations described by numerical and/or categorical variables. It generalizes the sparse clustering algorithm introduced in Witten & Tibshirani (2010) to any type of data (numerical, categorical or a mixture of both). The weights of the variables indicate their importance in the clustering process and discriminant variables are thus selected by means of weights set to 0.

Usage

```
sparsewkm(
  X,
  centers,
  lambda = NULL,
  nlambda = 20,
  nstart = 10,
  itermaxw = 20,
  itermaxkm = 10,
  renamelevel = TRUE,
  verbose = 1,
  epsilonw = 1e-04
)
```

Arguments

`X`	a dataframe of dimension `n` (observations) by `p` (variables) with numerical, categorical or mixed data.
`centers`	an integer representing the number of clusters.
`lambda`	a vector of numerical values (or a single value) providing a grid of values for the regularization parameter. If NULL (by default), the function computes its own lambda sequence of length `nlambda` (see details).
`nlambda`	an integer indicating the number of values for the regularization parameter. By default, `nlambda=20`.
`nstart`	an integer representing the number of random starts in the k-means algorithm. By default, `nstart=10`.
`itermaxw`	an integer indicating the maximum number of iterations for the inside loop over the weights `w`. By default, `itermaxw=20`.
`itermaxkm`	an integer representing the maximum number of iterations in the k-means algorithm. By default, `itermaxkm=10`.

renamelevel	a boolean. If TRUE (default option), each level of a categorical variable is renamed as 'variable_name=level_name'.
verbose	an integer value. If verbose=0, the function stays silent, if verbose=1 (default option), it prints whether the stopping criterion over the weights w is satisfied.
epsilonw	a positive numerical value. It provides the precision of the stopping criterion over w. By default, epsilonw =1e-04.

Details

Sparse weighted k-means performs clustering on mixed data (numerical and/or categorical), and automatically selects the most discriminant variables by setting to zero the weights of the non-discriminant ones.

The mixted data is first preprocessed: numerical variables are scaled to zero mean and unit variance; categorical variables are transformed into dummy variables, and scaled – in mean and variance – with respect to the relative frequency of each level.

The algorithm is based on the optimization of a cost function which is the weighted between-class variance penalized by a group L1-norm. The groups are implicitely defined: each numerical variable constitutes its own group, the levels associated to one categorical variable constitute a group. The importance of the penalty term may be adjusted through the regularization parameter lambda.

The output of the algorithm is two-folded: one gets a partitioning of the data set and a vector of weights associated to each variable. Some of the weights are equal to 0, meaning that the associated variables do not participate in the clustering process. If lambda is equal to zero, there is no penalty applied to the weighted between-class variance in the optimization procedure. The larger the value of lambda, the larger the penalty term and the number of variables with null weights. Furthemore, the weights associated to each level of a categorical variable are also computed.

Since it is difficult to choose the regularization parameter lambda without prior knowledge, the function builds automatically a grid of parameters and finds a partition and vector of weights for each value of the grid.

Note also that the columns of the data frame X must be of class factor for categorical variables.

Value

lambda	a numerical vector containing the regularization parameters (a grid of values).
W	a p by length(lambda) matrix. It contains the weights associated to each variable.
Wm	a q by length(lambda) matrix, where q is the number of numerical variables plus the number of levels of the categorical variables. It contains the weights associated to the numerical variables and to the levels of the categorical variables.
cluster	a n by length(lambda) integer matrix. It contains the cluster memberships, for each value of the regularization parameter.
sel.init.feat	a numerical vector of the same length as lambda, giving the number of selected variables for each value of the regularization parameter.
sel.trans.feat	a numerical vector of the same length as lambda, giving the number of selected numerical variables and levels of categorical variables.

`X.transformed` a matrix of size `n` by `q`, containing the transformed data: numerical variables scaled to zero mean and unit variables, categorical variables transformed into dummy variables, scaled (in means and variance) with respect to the relative frequency of the levels.

`index` a numerical vector indexing the variables and allowing to group together the levels of a categorical variable.

`bss.per.feature` a matrix of size `q` by `length(lambda)`. It contains the between-class variance computed on the `q` transformed variables (numerical variables and levels of categorical variables).

References

Witten, D. M., & Tibshirani, R. (2010). A framework for feature selection in clustering. Journal of the American Statistical Association, 105(490), 713-726.

Chavent, M. & Lacaille, J. & Mourer, A. & Olteanu, M. (2020). Sparse k-means for mixed data via group-sparse clustering, ESANN proceedings.

See Also

`plot.spwkm`, `info_clust`, `groupsparsewkm`, `recodmix`

Examples

```
data(HDdata)

out <- sparsewkm(X = HDdata[,-14], centers = 2)
# grid of automatically selected regularization parameters
out$lambda
k <- 10
# weights of the variables for the k-th regularization parameter
out$W[,k]
# weights of the numerical variables and of the levels
out$Wm[,k]
# partitioning obtained for the k-th regularization parameter
out$cluster[,k]
# number of selected variables
out$sel.init.feat
# between-class variance on each variable
out$bss.per.feature[,k]
# between-class variance
sum(out$bss.per.feature[,k])
```

`weightedss` *Weighted sum-of-squares criteria*

Description

This function computes various weighted sum-of-squares criteria for a given partition of a dataset described by numerical features.

Usage

```
weightedss(X, cl, w = NULL)
```

Arguments

`X`	a matrice or a dataframe of size `n` (observations) by `p` (variables) with numerical features only.
`cl`	a vector of integers of length `n`. It contains the cluster membership of the data.
`w`	a numerical vector of length `p`. It contains the weights to be applied to the features. By default, `w=NULL`, which amounts to setting each weight equal to 1.

Value

`bss.per.feature`	a numerical vector of length `p` containing the weighted between sum-of-squares per feature.
`wss.per.feature`	a numerical vector of length `p` containing the weighted within sum-of-squares per feature.
`bss.per.cluster`	a numerical vector of length `K` (`K` is the number of clusters) containing the weighted between sum-of-squares per cluster.
`wss.per.cluster`	a numerical vector of length `K` containing the weighted within sum-of-squares per cluster.
`bss`	a scalar representing the weighted between sum-of-squares of the partition. It may be computed as the sum over `bss.per.feature` or `bss.per.cluster`.
`wss`	a scalar representing the weighted within sum-of-squares of the partition. It may be computed as the sum over `wss.per.feature` or `wss.per.cluster`.

Examples

```
data(iris)
out <- weightedss(X = iris[,1:4], cl = as.numeric(iris$Species))
out$bss.per.feature
out$bss.per.cluster
out$bss
```

```
w <- c(0.3,0.3,0.2,0.2)
out <- weightedss(X = iris[,1:4], cl = as.numeric(iris$Species), w=w)
out$bss.per.feature
out$bss.per.cluster
out$bss
```

Index