Package: vclust (via r-universe)

October 15, 2026

Title Weighted Clustering of Quantitative and Categorical Variables

Version 0.0.1

Description Methods for weighted hierarchical clustering of variables. Variables may be quantitative, categorical, or a mixture of both. Following the selection of a cutpoint, the resulting clusters may be summarized by their first principal component score for the purpose of dimensionality reduction.

URL https://brettklamer.com/work/vclust

License GPL-3

LazyData TRUE

Depends R (>= 4.3.0)

Imports Rdpack, FactoMineR

Suggests tinytest, rmarkdown, ClustOfVar

RdMacros Rdpack

Language en-US

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/pak/sysreqs cmake libfreetype6-dev make libicu-dev libpng-dev libuv1-dev zlib1g-dev

Repository https://bklamer.r-universe.dev

Date/Publication 2026-02-17 01:33:37 UTC

RemoteUrl https://bitbucket.org/bklamer/vclust

RemoteRef HEAD

RemoteSha dbc7f7ec205066f5922b8288bd5f19f8a7881a4b

Contents

build_tree 2
cor_matrix 6

cor_sq_matrix 7
cut_tree 9
famd 11
plot.build_tree 13
weighted_cancor 14
weighted_cor 16
weighted_eta_sq 18

Index **21**

`build_tree` *Weighted hierarchical clustering of variables*

Description

Performs ascendant hierarchical clustering of variables for quantitative, categorical, or mixed datasets with row weights. For each candidate cluster, a quantitative synthetic variable is computed as the first component of a weighted factorial method (`FactoMineR::FAMD()` for mixed data, `FactoMineR::PCA()` for purely quantitative data, `FactoMineR::MCA()` for purely categorical data). The homogeneity of a cluster is defined as the sum, over variables in the cluster, of weighted squared Pearson correlations (for quantitative variables) and weighted squared correlation ratios (η^2) (for categorical variables) with the synthetic variable. Clusters are agglomerated by minimizing the loss of homogeneity when two clusters are merged, reproducing the methods of `ClustOfVar::hclustvar()`.

Usage

```
build_tree(data, weights = NULL, monotone = FALSE)
```

Arguments

`data`	(data.frame) A `data.frame` with numeric and/or factor columns for clustering. Character columns will be converted to factor. Optionally, including a column of numeric weights. Must not contain missing values.
`weights`	(Scalar character or `NULL`) Column name of row weights. If `NULL` (default), equal weights are used. Weights must be nonnegative, with $\sum_i w_i > 0$. Zeros are allowed (rows with zero weight contribute nothing).
`monotone`	(Scalar logical: `FALSE`) Whether or not dendrogram heights should be monotone (to prevent inversions).

Details

If missing values are present in your data, you must either impute them or subset to complete cases before analysis. For example,

```
data_cc <- data[complete.cases(data), ]
```

If variables with zero-variance or less than two factor levels are present in your data, you must exclude them before analysis. For example,

```
w <- data$weights
valid_vars <- function(x) {
  if(is.numeric(x)) {
    vclust:::weighted_var(x, w) > .Machine$double.eps
  } else if(is.factor(x) || is.character(x) || is.logical(x)) {
    nlevels(droplevels(as.factor(x))) > 1
  } else {
    FALSE
  }
}
are_valid <- vapply(data, valid_vars, logical(1))
data_valid <- data[are_valid]
```

Let $\mathbf{x}_1, \ldots, \mathbf{x}_{p_1}$ be quantitative variables and $\mathbf{z}_1, \ldots, \mathbf{z}_{p_2}$ be categorical variables measured on n observations, with $p = p_1 + p_2$. Denote a candidate cluster of variables as C_k. A partition $P_K = (C_1, \ldots, C_K)$ groups variables into K clusters. For a given cluster C_k, a quantitative synthetic variable $\mathbf{y}_k \in \mathbb{R}^n$ is defined as the variable most related to all variables in C_k:

$$\mathbf{y}_k = \arg\max_{\mathbf{u}\in\mathbb{R}^n} \left\{ \sum_{\mathbf{x}_j \in C_k} r^2(\mathbf{u}, \mathbf{x}_j) \; + \; \sum_{\mathbf{z}_j \in C_k} \eta^2(\mathbf{u} \mid \mathbf{z}_j) \right\},$$

where r^2 is the weighted squared Pearson correlation and η^2 is the weighted squared correlation ratio (the fraction of the variance of $\mathbf{u}$ explained by the categories of $\mathbf{z}_j$).

In the unweighted `ClustOfVar::hclustvar()` method, $\mathbf{y}_k$ is the first component of `PCAmixdata::PCAmix()` on the variables in C_k, and the homogeneity of the cluster equals the first eigenvalue λ_1^k of `PCAmixdata::PCAmix()`. In this weighted extension, we replace `PCAmixdata::PCAmix()` using **FactoMineR**:

- FactoMineR::FAMD for mixed clusters,
- FactoMineR::PCA for purely quantitative clusters,
- FactoMineR::MCA for purely categorical clusters,

each with row weights. The homogeneity is computed as

$$H(C_k) = \sum_{\mathbf{x}_j \in C_k} r^2(\mathbf{y}_k, \mathbf{x}_j) + \sum_{\mathbf{z}_j \in C_k} \eta^2(\mathbf{y}_k \mid \mathbf{z}_j),$$

where $\mathbf{y}_k$ is the first-axis score (`res$ind$coord[,1L]`) of the corresponding weighted **FactoMineR** result.

Weighted Measures:
Let $w_i \geq 0$ denote row weights for $i = 1, \ldots, n$, normalized to sum to one: $\tilde{w}_i = w_i / \sum_j w_j$. The weighted mean, variance, and covariance are

$$\bar{x} = \sum_{i=1}^{n} \tilde{w}_i \, x_i,$$

$$\mathrm{var}_w(x) = \sum_{i=1}^{n} \tilde{w}_i \, (x_i - \bar{x})^2,$$

$$\mathrm{cov}_w(x, y) = \sum_{i=1}^{n} \tilde{w}_i \, (x_i - \bar{x})(y_i - \bar{y}),$$

and the weighted Pearson correlation is

$$r_w(x, y) = \frac{\mathrm{cov}_w(x, y)}{\sqrt{\mathrm{var}_w(x)\,\mathrm{var}_w(y)}}.$$

For a numeric variable x and a factor g with groups indexed by s, the weighted squared correlation ratio is

$$\eta^2(x \mid g) = \frac{\mathrm{SSB}}{\mathrm{SST}} = \frac{\sum_s W_s \, (\bar{x}_s - \bar{x})^2}{\sum_i \tilde{w}_i \, (x_i - \bar{x})^2},$$

where $W_s = \sum_{i \in s} \tilde{w}_i$, $\bar{x}_s = \sum_{i \in s} \tilde{w}_i x_i / W_s$, and $\bar{x} = \sum_i \tilde{w}_i x_i$.

Agglomeration Criterion:

Following `ClustOfVar::hclustvar()`, the dissimilarity between two clusters A and B is the loss of homogeneity when merging:

$$d(A, B) = H(A) + H(B) - H(A \cup B).$$

At each step, the algorithm merges the pair with the smallest $d(A, B)$. The dendrogram height recorded for the new node is $d(A, B)$.

Dendrogram heights:

A dendrogram inversion occurs when a merge step records a smaller dissimilarity than a preceding merge step. In the dendrogram, this appears as a branch that merges at a lower height than its children branches. Generally, hierarchical clustering should be defined by non-decreasing distances as the algorithm progresses and combines larger clusters. For plotting stability, a `base::cummax()` adjustment (via `monotone = TRUE`) is used to enforce monotone heights. This adjusts the recorded heights for plotting but does not change the merge order, cluster memberships, or variable ordering. Dendrogram inversions are rare in practice.

Value

A list of class `"build_tree"` with elements:

- `merge`: Integer matrix of size $(p - 1) \times 2$ describing the merge history in hclust convention. Row i of merge describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then variable -j was merged at this stage. If j is positive then the merge was with the cluster formed at the (earlier) stage j of the algorithm. Thus negative entries in merge indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
- `height`: Numeric vector of length $p - 1$ with dissimilarities $d(A, B) = H(A) + H(B) - H(A \cup B)$ recorded at each merge.

- `order`: A vector giving the permutation of the original variables suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches.
- `labels`: Character vector of variable names.
- `clusmat`: Integer matrix $p \times p$ giving cluster memberships of variables at each level (columns correspond to $k = 1, \ldots, p$ clusters, where column j gives the cluster assignment when the tree is cut into j clusters).
- `call`: The matched call.
- `data`: The data frame of variables used for clustering, including the weight column (if any). The attribute `".row.weights"` stores the weight column name.
- `method`: `"build_tree"`

The attribute `".row.weights"` stores the weight column name.

References

Chavent M, Kuentz-Simonet V, Liquet B, Saracco J (2012). "ClustOfVar: An R Package for the Clustering of Variables." *Journal of Statistical Software*, **50**(13), 1–16. doi:10.18637/jss.v050.i13. https://www.jstatsoft.org/index.php/jss/article/view/v050i13.

Lê S, Josse J, Husson F (2008). "FactoMineR: A Package for Multivariate Analysis." *Journal of Statistical Software*, **25**(1), 1–18. doi:10.18637/jss.v025.i01.

See Also

`cut_tree()`

Examples

```
#-------------------------------------------------------------------------
# build_tree() examples
#-------------------------------------------------------------------------
library(vclust)

# Example 1: Quantitative variables with equal weights
set.seed(42)
n <- 60
data <- data.frame(
  col1 = rnorm(n),
  col2 = rnorm(n),
  col3 = rnorm(n),
  col4 = rnorm(n),
  weights = rep(1, n)
)

tree <- build_tree(data = data, weights = "weights")
plot(tree)
rect.hclust(tree = tree, k = 2)
head(tree$merge)
head(tree$height)
```

```
# Example 2: Mixed data with equal weights
set.seed(70)
n <- 80
X <- data.frame(
  col1 = rnorm(n),
  col2 = rnorm(n) + 0.6*rnorm(n)
)
Z <- data.frame(
  col3 = factor(sample(letters[1:3], n, TRUE)),
  col4 = factor(sample(c("L","M","H"), n, TRUE))
)

w <- rep(1, n) # equal weights
data <- cbind(X, Z, w)

tree_mix <- build_tree(data = data, weights = "w")
plot(tree_mix)
rect.hclust(tree = tree_mix, k = 2)
head(tree_mix$merge)
head(tree_mix$height)

# Example 3: Mixed data with unequal weights
set.seed(70)
w <- rexp(n) # positive, exponential weights

data <- cbind(X, Z, w)

tree_mix <- build_tree(data = data, weights = "w")
plot(tree_mix)
rect.hclust(tree = tree_mix, k = 2)
head(tree_mix$merge)
head(tree_mix$height)
```

`cor_matrix` *Weighted Pearson correlation matrix*

Description

Computes a symmetric matrix of Pearson correlations, optionally using row weights.

Usage

```
cor_matrix(data, weights = NULL)
```

Arguments

`data`	(data.frame or matrix) The dataframe from which variables are selected. Pairwise complete associa-

tions are calculated for instances of missing values. An error will return if any columns in `data` are non-numeric.

`weights` (Scalar character or `NULL`)
The name of the column in `data` containing row weights. All values must be finite, nonnegative, and with a strictly positive sum. If `NULL`, an unweighted computation is performed.

Details

Pairwise complete associations are calculated for instances of missing values.

Value

A numeric symmetric matrix with row and column names equal to `names(data)` (excluding the weight column, if any).

Examples

```
#-------------------------------------------------------------------------
# cor_matrix() examples
#-------------------------------------------------------------------------
library(vclust)

set.seed(123)
n <- 200
df <- data.frame(
  x = rnorm(n),
  y = rnorm(n),
  z = rnorm(n),
  w = rexp(n, rate = 1)
)

# Correlation matrix
S1 <- cor_matrix(df, weights = "w")
round(S1, 3)
```

`cor_sq_matrix` *Weighted squared association matrix for mixed data*

Description

Computes a symmetric matrix of squared association measures for a mix of numeric and categorical variables, optionally using row weights. For numeric–numeric pairs the entry is the weighted squared Pearson correlation r_w^2. For numeric–factor pairs the entry is the weighted eta-squared η_w^2. For factor–factor pairs the entry is the weighted squared first canonical correlation $\rho_{w,1}^2$.

Usage

```
cor_sq_matrix(data, weights = NULL)
```

Arguments

data	(data.frame) The dataframe from which variables are selected. Numeric columns are treated as continuous, while character and logical columns are coerced to factors. Pairwise complete associations are calculated for instances of missing values.
weights	(Scalar character or NULL) The name of the column in data containing row weights. All values must be finite, nonnegative, and with a strictly positive sum. If NULL, an unweighted computation is performed.

Details

Pairwise complete associations are calculated for instances of missing values. Character and logical variables in data are coerced to factors.

Value

A numeric symmetric matrix with row and column names equal to names(data) (excluding the weight column, if any).

Examples

```
#---------------------------------------------------------------------------
# cor_sq_matrix() examples
#---------------------------------------------------------------------------
library(vclust)

set.seed(123)
n <- 200
df <- data.frame(
  x = rnorm(n, 0, 1),
  y = 0.6 * rnorm(n) + 0.4 * rnorm(n),    # another numeric
  g = factor(sample(letters[1:3], n, replace = TRUE, prob = c(0.4, 0.4, 0.2))),
  h = factor(sample(c("yes", "no"), n, replace = TRUE)),
  z = rnorm(n),
  w = rexp(n, rate = 1)
)

# Use all variables
S1 <- cor_sq_matrix(df, weights = "w")
round(S1, 3)

# Select a subset of variables explicitly, unweighted
S2 <- cor_sq_matrix(df[c("x", "y", "g", "h")])
round(S2, 3)

# Include character and logical columns; they are coerced to factors internally
df$char_fac <- sample(c("A", "B", "C"), n, replace = TRUE)
df$logi_fac <- sample(c(TRUE, FALSE), n, replace = TRUE)
S3 <- cor_sq_matrix(df, weights = "w")
round(S3, 3)
```

```
# Missing values
df$logi_fac[1:30] <- NA
S4 <- cor_sq_matrix(df, weights = "w")
round(S4, 3)
```

`cut_tree` *Partition a tree from* `build_tree()`

Description

Cuts a tree produced by `build_tree()` into k clusters, recomputing for each cluster the weighted synthetic variable (first component score).

Usage

```
cut_tree(x, k, cor_matrix = FALSE)
```

Arguments

`x`	An object returned by `build_tree()`.
`k`	(Scalar integer) Desired number of clusters (1 <= k <= p).
`cor_matrix`	(Scalar logical: `FALSE`) If `TRUE`, returns 1) squared association matrices among variables within each cluster and 2) squared correlation matrix for the synthetic variables.

Value

A list with elements:

- `scores`: $n \times k$ data frame of synthetic variables for each cluster (first principal component).
- `scores_cor_matrix`: Pearson correlation matrix for the synthetic variables.
- `vars`: A four-column data frame:
  - `cluster`: Cluster membership
  - `variable`: Variable names
  - `squared_association`: If numeric, the weighted squared Pearson correlations r^2 with the cluster PC1. If factor, the weighted squared correlations $\eta^2(y \mid g)$ with the cluster PC1.
  - `association`: If numeric, weighted Pearson correlation r with the cluster PC1. If factor, the weighted correlations $\eta(x \mid g)$ with the cluster PC1.
- `vars_cor_matrix`: List of squared association matrices among variables within each cluster. Similarity for numeric:numeric is squared Pearson's correlation r^2; numeric:categorical is the correlation ratio $\eta^2(y \mid g)$; categorical:categorical is the squared canonical correlation between their dummy sets.

- `cluster`: Integer vector (length p) assigning each variable to a cluster.
- `H`: Homogeneity for each cluster, defined as the sum of squared associations with the cluster's first principal component.
- `Hprop`: Proportion of total homogeneity accounted for by the partition k. The gain in cohesion. Defined as $\frac{H(P_k)-H(P_1)}{H(P_p)-H(P_1)}$. Where P_k is the chosen cluster partition (K=k), P_1 is a single cluster partition (K=1), and P_p is the singleton cluster partition (K=p).
- `size`: Number of variables in each cluster.
- `k`: The chosen number of clusters.
- `call`: The matched call.
- `method`: "cut_tree"

See Also

`build_tree()`

Examples

```
#-------------------------------------------------------------------------------
# cut_tree() examples
#-------------------------------------------------------------------------------
library(vclust)

# Example 1: Quantitative variables with equal weights
set.seed(42)
n <- 60
data <- data.frame(
  col1 = rnorm(n),
  col2 = rnorm(n),
  col3 = rnorm(n),
  col4 = rnorm(n),
  weights = rep(1, n)
)

tree <- build_tree(data = data, weights = "weights")
plot(tree)
rect.hclust(tree = tree, k = 2)
cut <- cut_tree(tree, k = 2)
cut

# Example 2: Mixed data with equal weights
set.seed(70)
n <- 80
data <- data.frame(
  col1 = rnorm(n),
  col2 = rnorm(n) + 0.6*rnorm(n),
  col3 = factor(sample(letters[1:3], n, TRUE)),
  col4 = factor(sample(c("L","M","H"), n, TRUE)),
  w = rep(1, n)
)
```

```
tree2 <- build_tree(data = data, weights = "w")
plot(tree2)
rect.hclust(tree = tree2, k = 2)
cut2 <- cut_tree(tree2, k = 2)
cut2

# Example 3: Mixed data with unequal weights
data$w <- rexp(n)

tree3 <- build_tree(data = data, weights = "w")
plot(tree3)
rect.hclust(tree = tree3, k = 2)
cut3 <- cut_tree(tree3, k = 2)
cut3
```

`famd` *Factor Analysis for Mixed Data*

Description

Compute a low-dimensional representation of a data frame that may contain both numeric and categorical variables. The routine dispatches to `FactoMineR::PCA()`, `FactoMineR::MCA()`, or `FactoMineR::FAMD()` based on the variable types present.

Usage

```
famd(data, weights = NULL, ncp = 1L)
```

Arguments

`data`	(data.frame) The data frame of interest. Must not contain missing values.
`weights`	(Scalar character or `NULL`) The column name in `data` providing row weights. If `NULL`, equal weights are used.
`ncp`	(Scalar integer) The number of principal dimensions to compute and return. Must be an integer where `ncp >= 1` and `ncp <= min(nrow(data) - 1, d)`, where `d` is the effective dimensionality of the analysis: for PCA, `d` equals the number of numeric variables; for MCA, `d` equals the total number of factor levels minus the number of factor variables; for FAMD, `d` is the sum of both.

Details

If missing values are present in your data, you must either impute them or subset to complete cases before analysis. For example,

```
data_cc <- data[complete.cases(data), ]
```

If variables with zero-variance or less than two factor levels are present in your data, you must exclude them before analysis. For example,

```
w <- data$weights
valid_vars <- function(x) {
  if(is.numeric(x)) {
    vclust:::weighted_var(x, w) > .Machine$double.eps
  } else if(is.factor(x) || is.character(x) || is.logical(x)) {
    nlevels(droplevels(as.factor(x))) > 1
  } else {
    FALSE
  }
}
are_valid <- vapply(data, valid_vars, logical(1))
data_valid <- data[are_valid]
```

Character and logical columns are coerced to factors, and factors have unused levels dropped. Observation weights are validated to be numeric, finite, non-negative, and with positive sum.

Numeric variables are centered and standardized. Factor variables are encoded as a set of indicator variables and scaled so that the total contribution of that factor variable is comparable to a single quantitative variable. FAMD can be seen as a PCA performed on a concatenation of standardized numeric variables and properly scaled indicator variables for factor variables so that each original variable contributes equally.

Value

An object returned by the corresponding `FactoMineR` function, of class `"FAMD"`, `"PCA"`, or `"MCA"` depending on the input variable types.

Examples

```
#-------------------------------------------------------------------------
# famd() examples
#-------------------------------------------------------------------------
library(vclust)

# Mixed data example with weights
set.seed(123)
n <- 100
df <- data.frame(
  x1 = rnorm(n),
  x2 = rnorm(n, mean = 2),
  grp = factor(sample(letters[1:3], n, replace = TRUE)),
  w   = rexp(n)
)

res_mix <- famd(df, weights = "w", ncp = 3)
res_mix$eig
```

```
# individual scores
head(res_mix$ind$coord)

# Numeric-only (PCA)
res_pca <- famd(iris[1:4], ncp = 2)
res_pca$eig

# Factor-only (MCA)
toy <- data.frame(
  a = factor(sample(c("yes", "no"), 50, TRUE)),
  b = factor(sample(c("L", "M", "H"), 50, TRUE))
)
res_mca <- famd(toy, ncp = 2)
res_mca$eig
```

`plot.build_tree` *Dendrogram of the hierarchical clustering of variables*

Description

Dendrogram of the hierarchical cluster of variables resulting from `build_tree()`.

Usage

```
## S3 method for class 'build_tree'
plot(x, type = "tree", sub = "", ...)
```

Arguments

`x`	An object returned by `build_tree()`.
`type`	(Scalar character) If `"tree"`, a plot of the dendrogram. If `"index"`, a plot of the aggregation levels.
`sub`	(Scalar character) Subtitle for the plot.
`...`	Additional arguments passed to `base::plot()`.

See Also

`build_tree()`

`weighted_cancor` *Weighted canonical correlation for categorical variables*

Description

Compute the first canonical correlation coefficient between two categorical variables via their indicator (dummy) sets. The coefficient equals the largest singular value of the standardized cross-covariance between the indicator sets and lies in [0, 1]. For a 2×2 table, this reduces to the absolute ϕ coefficient and Cramer's V.

Usage

```
weighted_cancor(x, y, w = NULL)

weighted_cancor_sq(x, y, w = NULL)
```

Arguments

`x`	(factor) The first categorical variable (with at least two observed levels). Observations with missing values are excluded pairwise. Each distinct level defines one indicator in the dummy set used for canonical correlation. Non-factor vectors will be coerced to factor.
`y`	(factor) The second categorical variable (with at least two observed levels). Observations with missing values are excluded pairwise. Each distinct level defines one indicator in the dummy set used for canonical correlation. Non-factor vectors will be coerced to factor.
`w`	(numeric or `NULL`) Weights used to compute the weighted canonical correlation. If numeric, values must be nonnegative and the same length as `x` and `y`. Zero weights are allowed. Observations with missing, negative, or non-finite values are excluded pairwise. Set to `NULL` (default) for an unweighted computation.

Details

Let $X \in \{1, \ldots, r\}$ and $Y \in \{1, \ldots, s\}$ denote categorical variables, and let $w_k \geq 0$ be case weights (internally normalized to sum to one). Denote the weighted joint distribution by

$$P = \{p_{ij}\}_{i=1..r,\, j=1..s}$$

with

$$p_{ij} = \sum_k w_k \mathbb{1}(x_k = i, y_k = j)$$

and marginals

$$p_x = P\mathbf{1}_s$$
$$p_y = P^\top \mathbf{1}_r.$$

The covariance blocks for the indicator sets are

$$S_{xx} = \mathrm{diag}(p_x) - p_x p_x^\top$$
$$S_{yy} = \mathrm{diag}(p_y) - p_y p_y^\top$$
$$S_{xy} = P - p_x p_y^\top.$$

The first canonical correlation is then

$$\rho_1 = \sigma_{\max}(S_{xx}^{-1/2}\, S_{xy}\, S_{yy}^{-1/2}),$$

where $\sigma_{\max}(\cdot)$ is the largest singular value and the inverses are taken on the support of the positive eigenvalues. This ρ_1 equals the dominant singular value in correspondence analysis of the contingency table built from X and Y, i.e., the square root of the largest principal inertia.

The helper `weighted_cancor_sq()` returns ρ_1^2, the weighted squared first canonical correlation coefficient, which takes values in $[0, 1]$.

Value

- `weighted_cancor()`: Scalar numeric in $[0, 1]$ or `NA_real_`.
- `weighted_cancor_sq()`: Scalar numeric $[0, 1]$ or `NA_real_`.

Returns `NA_real_` if all observations are removed during filtering or the total weight is nonpositive. Returns `0` if all mass lies in a single level of either variable or if numerical degeneracy prevents computation.

Examples

```
#-------------------------------------------------------------------------
# weighted_cancor() examples
#-------------------------------------------------------------------------
library(vclust)

# 2x2 example: equals |phi| and Cramer's V
x <- factor(rep(c("A", "B"), times = c(30, 70)))
y <- factor(c(rep("U", 25), rep("V", 5),  # mostly A-U
              rep("U", 20), rep("V", 50))) # mostly B-V
weighted_cancor(x, y)

# Compare to |phi| computed from weighted proportions
tab <- prop.table(table(x, y))
px <- rowSums(tab)
py <- colSums(tab)
phi <- (tab[1,1]*tab[2,2] - tab[1,2]*tab[2,1]) / sqrt(px[1]*px[2]*py[1]*py[2])
phi <- as.numeric(phi)
c(cc = weighted_cancor(x, y), abs_phi = abs(phi))
```

```
# Rows with negative weights and missing values filtered.
set.seed(1)
xw <- factor(sample(letters[1:3], 100, replace = TRUE))
yw <- factor(sample(LETTERS[1:4], 100, replace = TRUE))
w  <- runif(100)
w[c(5, 10)] <- NA_real_
w[20] <- -1  # dropped
xw[30] <- NA  # dropped
yw[40] <- NA  # dropped
weighted_cancor(xw, yw, w = w)

# Larger table where the value equals the dominant CA singular value
x3 <- factor(sample(paste0("x", 1:5), 500, TRUE))
y3 <- factor(sample(paste0("y", 1:6), 500, TRUE))
weighted_cancor(x3, y3)

# Degenerate cases
# Single observed level in x after filtering
weighted_cancor(factor(rep("only", 10)), factor(sample(c("a","b"), 10, TRUE)))
# No usable data
weighted_cancor(factor(c(NA, NA)), factor(c(NA, NA)), w = c(1, 1))
```

`weighted_cor` *Weighted Pearson correlation*

Description

Computes the weighted Pearson correlation coefficient r between two numeric vectors.

Usage

```
weighted_cor(x, y, w = NULL)

weighted_cor_sq(x, y, w = NULL)
```

Arguments

`x`	(numeric) The first variable whose weighted correlation with `y` is computed. Observations with missing or non-finite values are excluded pairwise.
`y`	(numeric) The second variable whose weighted correlation with `x` is computed. Observations with missing or non-finite values are excluded pairwise.
`w`	(numeric or `NULL`) Weights used to compute the weighted correlation. If numeric, values must be nonnegative and the same length as `x` and `y`. Zero weights are allowed. Observations with missing, negative, or non-finite values are excluded pairwise. Set to `NULL` (default) for an unweighted computation.

Details

This implementation uses the population definition of correlation (no Bessel $(n-1)$ correction in variance).

Observations with any non-finite value in `x`, `y`, or `w`, or with negative weight, are excluded. Zero weights are allowed and contribute nothing.

Let $w_i \geq 0$ denote row weights for $i = 1, \ldots, n$ and $\tilde{w}$ be the sum-to-1 normalized weights. For the pairwise complete subset, $\mathcal{I} \subseteq \{1, \ldots, n\}$:

$$\bar{x} = \frac{\sum_{i \in \mathcal{I}} \tilde{w}_i x_i}{\sum_{i \in \mathcal{I}} \tilde{w}_i}$$

$$\mathrm{var}_w(x) = \sum_{i \in \mathcal{I}} \tilde{w}_i \, (x_i - \bar{x})^2,$$

$$\mathrm{cov}_w(x, y) = \sum_{i \in \mathcal{I}} \tilde{w}_i \, (x_i - \bar{x})(y_i - \bar{y})$$

$$r_w(x, y) = \frac{\mathrm{cov}_w(x, y)}{\sqrt{\mathrm{var}_w(x) \, \mathrm{var}_w(y)}}.$$

The helper `weighted_cor_sq()` returns r_w^2, the weighted squared Pearson correlation.

Value

- `weighted_cor()`: Scalar numeric in $[-1, 1]$ or `NA_real_`.
- `weighted_cor_sq()`: Scalar numeric $[0, 1]$ or `NA_real_`.

Returns `NA_real_` if all observations are removed during filtering or the total weight is nonpositive. Returns `0` if either weighted variance is nonpositive.

Examples

```
#------------------------------------------------------------------------------
# weighted_cor() examples
#------------------------------------------------------------------------------
library(vclust)

# Basic usage with equal weights
x <- c(1, 2, 3, 4, 5)
y <- c(2, 1, 4, 3, 5)
w <- rep(1, length(x))
weighted_cor(x, y, w)
weighted_cor(x, y)

# Heavier weight on the last two observations
w2 <- c(1, 1, 1, 5, 5)
weighted_cor(x, y, w2)
weighted_cor_sq(x, y, w2)
```

```
# Pairwise handling of missing values
x_na <- c(1, NA, 3, 4, 5)
y_na <- c(2, 1, 4, NA, 5)
w_na <- c(1, 1, 1, NA, 1)
weighted_cor(x_na, y_na, w_na)

# Zero weights exclude observations without removing length alignment
w_zero <- c(1, 0, 1, 0, 1)
weighted_cor(x, y, w_zero)

# Degenerate case: zero variance in x
x_const <- c(3, 3, 3, 3)
y_any   <- c(1, 2, 3, 4)
weighted_cor(x_const, y_any)
```

`weighted_eta_sq` *Weighted η association*

Description

Computes the weighted η and η^2, a measure for the association between a numeric variable x and a single categorical variable y.

Usage

```
weighted_eta_sq(x, y, w = NULL)

weighted_eta(x, y, w = NULL)
```

Arguments

`x`	(numeric) The numeric variable whose weighted association with factor `y` is computed. Observations with missing or non-finite values are excluded pairwise.
`y`	(factor) The factor variable whose weighted association with numeric `x` is computed. Observations with missing or non-finite values are excluded pairwise. Non-factor inputs are coerced via `as.factor(y)`.
`w`	(numeric or `NULL`) Weights used to compute the weighted association. If numeric, values must be nonnegative and the same length as `x` and `y`. Zero weights are allowed. Observations with missing, negative, or non-finite values are excluded pairwise. Set to `NULL` (default) for an unweighted computation.

Details

Let $x_i \in \mathbb{R}$ denote the numeric response for observation $i = 1, \ldots, n$, $y_i \in \{1, \ldots, G\}$ denote the group membership, and $w_i \geq 0$ be the weight for observation i. Internally, non-finite x_i or w_i, missing y_i, and negative w_i observations are removed pairwise prior to computation.

The weights are normalized to sum to one, i.e., $\tilde{w}_i = w_i / \sum_{j=1}^{n} w_j$ with $\sum_i \tilde{w}_i = 1$. Define group weights $W_g = \sum_{i:y_i=g} \tilde{w}_i$ and group means $\bar{x}_g = \left(\sum_{i:y_i=g} \tilde{w}_i x_i\right) / W_g$ for groups with $W_g > 0$. The weighted grand mean is $\bar{x} = \sum_{i=1}^{n} \tilde{w}_i x_i$.

The between-group sum of squares is

$$\mathrm{SSB} = \sum_{g=1}^{G} W_g (\bar{x}_g - \bar{x})^2,$$

and the total sum of squares is

$$\mathrm{SST} = \sum_{i=1}^{n} \tilde{w}_i (x_i - \bar{x})^2.$$

The weighted eta-squared is then

$$\eta^2(x \mid y) = \frac{\mathrm{SSB}}{\mathrm{SST}}.$$

This quantity lies in $[0, 1]$ and is truncated to that interval for numerical robustness.

Groups with zero total weight $W_g = 0$ are dropped from the SSB computation. If there are fewer than two nonempty groups, or if SST is non-finite, the function returns `0`. This implementation corresponds to the standard (fixed-effects, one-way) eta-squared effect size.

The helper `weighted_eta()` returns $\eta(x \mid y)$, the weighted association, which takes values in $[0, 1]$.

Value

- `weighted_eta_sq()`: Scalar numeric $[0, 1]$ or `NA_real_`.
- `weighted_eta()`: Scalar numeric in $[0, 1]$ or `NA_real_`.

Returns `NA_real_` if all observations are removed during filtering or the total weight is nonpositive. Returns `0` if fewer than two nonempty groups remain or if the weighted total sum of squares is non-finite or effectively zero.

Examples

```
#------------------------------------------------------------------------------
# weighted_eta_sq() examples
#------------------------------------------------------------------------------
library(vclust)

set.seed(123)
n <- 30
x <- rnorm(n, mean = rep(c(0, 0.5, 1), each = n/3), sd = 1)
```

```
y <- factor(rep(LETTERS[1:3], each = n/3))

# Unweighted eta-squared (all weights equal)
weighted_eta_sq(x, y)
weighted_eta_sq(x, y, rep_len(1, n))

# With nonnegative weights that upweight group C
w <- rep_len(1, n)
w[y == "C"] <- 2
weighted_eta_sq(x, y, w)

# Rows with NA/Inf in x or w are dropped automatically
x_bad <- x
x_bad[c(2, 10)] <- NA
w_bad <- w
w_bad[5] <- Inf
weighted_eta_sq(x_bad, y, w_bad)

# Degenerate cases
## Single nonempty group after zeroing weights returns 0
w_single <- as.numeric(y == "A")
weighted_eta_sq(x, y, w_single)

## All rows removed (e.g., all weights negative) returns NA_real_
w_all_bad <- rep(-1, n)
weighted_eta_sq(x, y, w_all_bad)
```

Index

base::cummax(), *4*
base::plot(), *13*
build_tree, 2
build_tree(), *9, 10*, *13*

ClustOfVar::hclustvar(), *2–4*
cor_matrix, 6
cor_sq_matrix, 7
cut_tree, 9
cut_tree(), *5*

FactoMineR::FAMD, *3*
FactoMineR::FAMD(), *2*, *11*
FactoMineR::MCA, *3*
FactoMineR::MCA(), *2*, *11*
FactoMineR::PCA, *3*
FactoMineR::PCA(), *2*, *11*
famd, 11

PCAmixdata::PCAmix(), *3*
plot.build_tree, 13

weighted_cancor, 14
weighted_cancor(), *15*
weighted_cancor_sq *(*weighted_cancor*)*, 14
weighted_cancor_sq(), *15*
weighted_cor, 16
weighted_cor(), *17*
weighted_cor_sq *(*weighted_cor*)*, 16
weighted_cor_sq(), *17*
weighted_eta *(*weighted_eta_sq*)*, 18
weighted_eta(), *19*
weighted_eta_sq, 18
weighted_eta_sq(), *19*