Package: cclustr (via r-universe)

May 18, 2026

Title Consensus Clustering Methods for Multiple Imputed Data

Version 0.1.1

Description Provides tools for performing consensus clustering on multiple imputed datasets. The package supports a range of clustering algorithms across imputations, including hierarchical methods (e.g., Ward, single, complete, average) and partition-based approaches such as k-means, k-medoids (PAM), fuzzy clustering, model-based clustering ('mclust'), and methods for mixed or categorical data (k-modes and k-prototypes). A co-assignment matrix is constructed to quantify agreement between partitions, and consensus solutions are derived via hierarchical clustering applied to the resulting dissimilarity matrix. Additional functions are provided for validation and visualization of clustering results, facilitating robust analysis in the presence of missing data. Consensus clustering framework is based on Monti et al. (2003) <doi:10.1023/A:1023949509487>, rank aggregation methods follow Pihur et al. (2007) <doi:10.1093/bioinformatics/btm158>, and the PAC (Proportion of Ambiguous Clustering) metric is based on Senbabaoglu et al. (2014) <doi:10.1038/srep06207>.

License MIT + file LICENSE

URL https://github.com/andrews06ml/cclustr

BugReports https://github.com/andrews06ml/cclustr/issues

Depends R (>= 4.0)

Imports cluster, e1071, fpc, graphics, stats, viridisLite, klaR, clustMixType, proxy, mclust

Suggests knitr, mice, mlbench, rmarkdown, spelling, testthat (>= 3.0.0), utils

Encoding UTF-8

Language en-US

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

NeedsCompilation no

Author Anhuar Duran Mendoza [aut], Andres Montenegro Lemus [aut, cre], Mario Pacheco Lopez [aut]

Maintainer Andres Montenegro Lemus <andresfemole@gmail.com>

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

Date/Publication 2026-05-18 20:50:08 UTC

RemoteUrl https://github.com/cran/cclustr

RemoteRef HEAD

RemoteSha 1bc93d4eb2ad2e60e9e5f92665705ea1e4a64d02

Contents

`as_mild_list` *Standardize multiple imputation outputs into a unified list*

Description

Converts multiple imputation outputs from different packages into a standardized list of completed datasets, where each element corresponds to one imputed dataset. Supported formats include `mids` objects from **mice**, long-format data frames, `amelia` objects, `imputationList` objects from **mitools**, and plain lists of data frames. After conversion, consistency and data quality checks are applied across all imputed datasets.

This function is designed as a preprocessing step for clustering multiple imputed datasets, ensuring strict comparability across imputations.

Usage

```
as_mild_list(x)
```

Arguments

x	An imputation object. Accepted classes and formats: • `mids`: output of `mice::mice()`. • `data.frame`: long-format output of `mice::complete(x, action = "long")`. Must contain a `.imp` column. • `amelia`: output of `Amelia::amelia()`. • `imputationList`: output of `mitools::imputationList()`. • `list`: a plain list where every element is a `data.frame` representing one completed dataset.

Details

The function performs the following consistency checks before returning:

- Identical number of rows across all imputed datasets.
- Identical number of columns across all imputed datasets.
- Identical column names across all imputed datasets.
- Absence of `NA`, `NaN`, and `Inf` values.
- Absence of numeric columns with all values equal, as such variables provide no information for distance-based clustering and may lead to degenerate solutions.

All datasets are coerced to base `data.frame` to avoid compatibility issues with `tibble` or other subclasses.

Value

A named list of `data.frame` objects, one per imputed dataset. All datasets are guaranteed to have identical dimensions and column names, and to contain no `NA`, `NaN`, `Inf`, or all equal values in a column.

References

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, **63**(3), 581-592. doi:10.1093/biomet/63.3.581

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. doi:10.18637/jss.v045.i03

Honaker, J., King, G., & Blackwell, M. (2011). Amelia II: A Program for Missing Data. *Journal of Statistical Software*, **45**(7), 1-47. doi:10.18637/jss.v045.i07

See Also

`cluster_imputations`, `consensus_clustering`

Examples

```
# ------------------------------------------------------------
# Example 1: Simple list of completed datasets
# ------------------------------------------------------------
set.seed(123)

imp_list <- replicate(3, {
  data.frame(
    x = rnorm(10),
    y = rnorm(10)
  )
}, simplify = FALSE)

mild <- as_mild_list(imp_list)

length(mild)      # number of imputations
str(mild[[1]])    # structure of one dataset


# ------------------------------------------------------------
# Example 2: List of completed datasets with mice
# ------------------------------------------------------------

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

  set.seed(123)

  # Example 1: Using a mids object
  imp <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)
  mild <- as_mild_list(imp)

  length(mild)        # number of imputations
  str(mild[[1]])      # structure of one completed dataset

  # Example 2: Using long-format data
  long_df <- mice::complete(imp, action = "long", include = FALSE)
  mild2 <- as_mild_list(long_df)

  length(mild2)
}
```

---

choose_best_clustering

Select the optimal number of clusters from a validation table

---

Description

Identifies the best value of k from the validation metrics produced by `validate_clustering`, using a weighted rank aggregation strategy. Each metric is ranked independently and then combined into

a single score using user-defined weights. The k with the lowest weighted rank score is selected as optimal. In case of ties, a secondary metric is used as a tiebreaker.

Usage

```
choose_best_clustering(
  validation_table,
  consensus_results,
  weights = NULL,
  prefer_stability = NULL,
  tie_breaker = c("silhouette", "pac", "dunn", "ch", "db", "ari_between",
    "ari_consensus")
)
```

Arguments

`validation_table`
: A `data.frame` as returned by `validate_clustering`, containing one row per evaluated k and columns `pac`, `silhouette_mean`, `ari_mean_between_imputations`, `ari_consensus_mean`, `calinski_harabasz_mean`, `davies_bouldin_mean`, and `dunn_index`.

`consensus_results`
: A list as returned by `consensus_clustering`, either for a single k or for multiple k values. Used to retrieve the consensus labels and co-assignment matrix for the selected k.

`weights`
: A named numeric vector specifying the relative importance of each metric in the rank aggregation. Expected names are `pac`, `silhouette`, `ari_between`, `ari_consensus`, `ch`, `db`, and `dunn`. If `NULL` (default), weights are set automatically based on `prefer_stability`. When provided explicitly, `prefer_stability` is ignored.

`prefer_stability`
: Controls the default weighting strategy when `weights` is `NULL`. Accepted values are:

  - `NULL` (default): all metrics receive equal weight, providing a balanced selection that does not favor any particular aspect of clustering quality.
  - `TRUE`: stability metrics (`pac`, `ari_between`, `ari_consensus`) receive higher weights.
  - `FALSE`: internal compactness metrics (`silhouette`, `ch`, `db`, `dunn`) receive higher weights.

`tie_breaker`
: A character string specifying the secondary metric used to break ties in the weighted rank score. One of `"silhouette"` (default), `"pac"`, `"dunn"`, `"ch"`, `"db"`, `"ari_between"`, or `"ari_consensus"`.

Details

The rank aggregation proceeds as follows:

1. Each metric is ranked from best to worst (rank 1 = best), accounting for the direction of optimality: lower is better for pac and davies_bouldin_mean; higher is better for all remaining metrics.
2. For each k, the weighted average rank is computed using the provided or default weights. Rows with NA in some metrics are handled by renormalizing the weights over available metrics only.
3. The k with the lowest weighted rank score is selected. If multiple k values share the minimum score, the tiebreaker metric is used; if still tied, the smallest k is preferred.

Default weights when prefer_stability = NULL (default): pac = 1, silhouette = 1, ari_between = 1, ari_consensus = 1, ch = 1, db = 1, dunn = 1.

Default weights when prefer_stability = TRUE: pac = 2, silhouette = 1.5, ari_between = 2, ari_consensus = 2, ch = 1, db = 1, dunn = 1.

Default weights when prefer_stability = FALSE: pac = 1, silhouette = 2, ari_between = 1, ari_consensus = 1, ch = 2, db = 2, dunn = 2.

No automatic selection method replaces domain knowledge. The scores_table and plot_validation_metrics are provided so that the user can inspect individual metrics and make an informed decision.

Value

A named list with the following elements:

- best_k: integer, the selected optimal number of clusters.
- best_consensus: integer vector of consensus cluster labels for the selected k, one per observation.
- best_coassignment: numeric matrix of size $n \times n$ with co-assignment probabilities for the selected k.
- best_consensus_result: the full consensus result object for the selected k, as returned by consensus_clustering.
- scores_table: the input validation_table with an additional score column containing the weighted rank score for each k, sorted by k.
- weights: named numeric vector of weights effectively used in the aggregation.
- tie_breaker: character string indicating the tiebreaker metric used.

References

Pihur, V., Datta, S., & Datta, S. (2007). Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach. *Bioinformatics*, **23**(13), 1607-1615. doi:10.1093/bioinformatics/btm158

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**, 193-218. doi:10.1007/BF01908075

See Also

validate_clustering, consensus_clustering, plot_validation_metrics

Examples

```
# ------------------------------------------------------------
# Example 1: Basic validation with simulated partitions
# ------------------------------------------------------------

# 1. Simulated validation table for k = 2, 3, 4
val_table <- data.frame(
  k                            = 2:4,
  pac                          = c(0.15, 0.08, 0.20),
  silhouette_mean              = c(0.42, 0.61, 0.38),
  ari_mean_between_imputations = c(0.80, 0.91, 0.75),
  ari_consensus_mean           = c(0.82, 0.93, 0.77),
  calinski_harabasz_mean       = c(120,  198,  105),
  davies_bouldin_mean          = c(0.85, 0.54, 0.97),
  dunn_index                   = c(0.30, 0.48, 0.27)
)

# 2. Simulated consensus result for each k
set.seed(123)
n <- 30
make_cons <- function(k) {
  list(
    k            = k,
    consensus    = sample(seq_len(k), n, replace = TRUE),
    coassignment = matrix(runif(n * n), n, n)
  )
}
cons_results <- list(k2 = make_cons(2),
                     k3 = make_cons(3),
                     k4 = make_cons(4))

# 3. Default selection (equal weights)
best <- choose_best_clustering(val_table, cons_results)
best$best_k
head(best$scores_table)


# 4. Custom weights
best_custom <- choose_best_clustering(
  val_table, cons_results,
  weights = c(pac = 3, silhouette = 1, ari_between = 3,
              ari_consensus = 3, ch = 1, db = 1, dunn = 1)
)


# ------------------------------------------------------------
# Example 2: Full pipeline with mice
# ------------------------------------------------------------
if (requireNamespace("mice", quietly = TRUE)) {

set.seed(123)
imp   <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)
```

```
mild  <- as_mild_list(imp)
parts <- cluster_imputations(mild, method = "ward.D2", k = 2:5)
cons  <- consensus_clustering(parts)
val   <- validate_clustering(parts, cons)

# Default selection (equal weights)
best <- choose_best_clustering(val, cons)
best$best_k

# Stability-focused selection
best2 <- choose_best_clustering(val, cons,
                                prefer_stability = TRUE)

# Compactness-focused selection
best3 <- choose_best_clustering(val, cons,
                                prefer_stability = FALSE,
                                tie_breaker      = "dunn")

# Custom weights
best4 <- choose_best_clustering(val, cons,
                                weights = c(pac = 3, silhouette = 1,
                                            ari_between = 3, ari_consensus = 3,
                                            ch = 1, db = 1, dunn = 1))
}
```

`cluster_imputations` *Perform clustering on multiple imputed datasets*

Description

Applies a clustering algorithm to each completed dataset in a standardized imputation list (as produced by `as_mild_list`). Supports hierarchical clustering, k-means, PAM, fuzzy c-means, model-based clustering, k-modes (for categorical data), and k-prototypes (for mixed data). Accepts a single value of k or a range, optionally scales the data prior to clustering, and supports multiple distance metrics including Gower, Jaccard, Simple Matching Coefficient, and user-defined distances.

Usage

```
cluster_imputations(
  imp_list,
  method = "ward.D2",
  k,
  scale_data = c("global", "within", "none"),
 distance = c("euclidean", "manhattan", "gower", "jaccard", "simple_matching", "custom"),
  dist_fun = NULL,
  dist_args = list(),
  ...
)
```

Arguments

imp_list	A named list of data.frame objects, as returned by as_mild_list. All datasets must have identical dimensions and column names.
method	A character string specifying the clustering algorithm. For hierarchical clustering, accepted values are "ward.D", "ward.D2", "single", "complete", "average", "centroid", "median", and "mcquitty". Additional options are: • "kmeans": k-means clustering for numeric data. • "pam": partitioning around medoids; supports mixed data when used with distance = "gower". • "fuzzy": fuzzy c-means clustering. • "mclust": model-based clustering via Gaussian mixtures. • "kmodes": clustering for purely categorical data. • "kprototypes": clustering for mixed data (numeric and categorical). Default is "ward.D2".
k	A single integer or an integer vector specifying the number of clusters. If a vector is provided, clustering is performed for each value of k.
scale_data	Character string specifying the scaling strategy for continuous numeric variables (binary variables coded as 0/1 are automatically excluded from scaling). Options are: • "global": (default) variables are scaled using pooled mean and standard deviation computed across all imputations without stacking them in memory (combinatorial variance formula). Ensures comparability of distance scales between datasets and is recommended for consensus clustering. • "within": each imputed dataset is scaled independently using its own mean and standard deviation. Useful for exploratory analyses but may introduce inconsistencies in distance scales across datasets. • "none": no scaling is applied. Scaling is automatically disabled for distance = "gower", "jaccard", or "simple_matching", and for method = "kmodes" and "kprototypes", as these handle variable scales internally.
distance	A character string specifying the distance metric used for hierarchical clustering and method = "pam". Options are: • "euclidean": (default) standard Euclidean distance. Suitable for continuous numeric data after scaling. • "manhattan": sum of absolute differences. Suitable for continuous numeric data; more robust to outliers than Euclidean. • "gower": Gower distance via cluster::daisy(). Supports mixed data types (numeric, categorical, binary). When used with hierarchical clustering, numeric variables are normalized using global min/max computed across all imputations, ensuring cross-imputation comparability. Requires package **cluster**. • "jaccard": Jaccard dissimilarity. Requires all variables to be binary (0/1 or two-level factor). Requires package **proxy**.

- "simple_matching": Simple Matching Coefficient (SMC). Requires all variables to be binary. Requires package **proxy**.
- "custom": user-defined distance function supplied via dist_fun. Must return an object of class dist.

Ignored for method = "kmeans", "fuzzy", "mclust", "kmodes", and "kprototypes", which compute distances internally.

For method = "pam", all distance options are fully supported as PAM operates on a precomputed dissimilarity matrix.

dist_fun — A function that takes a data.frame and returns a dist object. Required when distance = "custom"; ignored otherwise.

dist_args — A named list of additional arguments passed to the distance function. For distance = "gower", arguments are forwarded to cluster::daisy(); for distance = "jaccard" or "simple_matching", to proxy::dist(); for distance = "euclidean" or "manhattan", to stats::dist(); also for distance = "custom". Default is list().

... — Additional arguments passed to the underlying clustering function: stats::kmeans, e1071::cmeans, mclust::Mclust, klaR::kmodes, or clustMixType::kproto. Not used for method = "pam", which operates on a precomputed dissimilarity matrix.

Details

Scaling strategy and cross-imputation comparability:

For methods that rely on Euclidean or Manhattan distances ("kmeans", "fuzzy", "mclust", hierarchical with distance = "euclidean" or "manhattan"), scale_data = "global" is strongly recommended. It computes the pooled mean and standard deviation across all imputations using the combinatorial variance formula — without creating a stacked copy of the data in memory — and applies the same transformation to every imputation. This ensures that distances are on the same scale across datasets, which is essential for a meaningful co-assignment matrix.

Binary variables coded as 0/1 numeric are automatically excluded from scaling because standardizing them loses their binary semantics.

Gower distance and global normalization:

Gower distance normalizes each numeric variable by its range. When computed independently per imputation, different imputations may produce slightly different ranges, leading to incomparable distance scales. When distance = "gower" is used with hierarchical clustering or PAM, the function pre-normalizes numeric variables using global min/max computed across all imputations (without stacking), so that Gower distances are on a consistent scale across datasets. Categorical and factor variables are left untouched and handled normally by cluster::daisy(). Additional arguments to cluster::daisy() can be passed via dist_args.

Hierarchical clustering optimization:

When method is hierarchical and k is a vector, the distance matrix and dendrogram are computed only once per imputed dataset and then cut at each requested k, which avoids redundant computation.

Data type compatibility:

- Numeric-only: hierarchical, "kmeans", "fuzzy", "mclust".

- Categorical-only: `"kmodes"`.
- Mixed (numeric + categorical): `"pam"` with `distance = "gower"` or `"kprototypes"`.
- Binary-only: hierarchical with `distance = "jaccard"` or `"simple_matching"`.

Character variables are internally converted to factors.

Value

- If `k` is a single value: a named list of integer vectors, one per imputed dataset, containing the cluster assignment for each observation.
- If `k` is a vector: a named list of lists, where each top-level element corresponds to one value of `k` (named `"k2"`, `"k3"`, etc.), and each inner element is a named list of cluster assignment vectors, one per imputed dataset.

References

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley. doi:10.1002/9780470316801

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, **2**(3), 283-304. doi:10.1023/A:1009769707641

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, **27**(4), 857-871. doi:10.2307/2528823

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, **8**(1), 289-317. doi:10.32614/RJ2016021

See Also

`as_mild_list`, `consensus_clustering`

Examples

```
# ------------------------------------------------------------
# Example 1: Numeric data, hierarchical clustering
# ------------------------------------------------------------
set.seed(123)
imp_list <- replicate(3, {
  data.frame(x = rnorm(20), y = rnorm(20))
}, simplify = FALSE)

# Single k, global scaling, euclidean distance (default)
res <- cluster_imputations(imp_list, method = "ward.D2", k = 2)
str(res)

# Range of k, manhattan distance
res_multi <- cluster_imputations(imp_list, method = "ward.D2",
                                 k = 2:4, distance = "manhattan")
```

```
str(res_multi)

# ------------------------------------------------------------
# Example 2: Mixed data, PAM with Gower distance
# ------------------------------------------------------------
set.seed(123)
imp_mixed <- replicate(3, {
  data.frame(
    x    = rnorm(20),
    grup = factor(sample(letters[1:3], 20, replace = TRUE))
  )
}, simplify = FALSE)

res_pam <- cluster_imputations(imp_mixed, method = "pam",
                               k = 2, distance = "gower")
str(res_pam)

# ------------------------------------------------------------
# Example 3: Binary data, Jaccard distance
# ------------------------------------------------------------
set.seed(123)
imp_bin <- replicate(3, {
  data.frame(
    a = sample(0:1, 20, replace = TRUE),
    b = sample(0:1, 20, replace = TRUE),
    c = sample(0:1, 20, replace = TRUE)
  )
}, simplify = FALSE)


if (requireNamespace("proxy", quietly = TRUE)) {
  res_jac <- cluster_imputations(imp_bin, method = "ward.D2",
                                 k = 2, distance = "jaccard")
  str(res_jac)
}

# ------------------------------------------------------------
# Example 4: Full pipeline with mice
# ------------------------------------------------------------
if (requireNamespace("mice", quietly = TRUE)) {

  df_pre       <- mice::nhanes
  df_pre$age   <- factor(df_pre$age)
  df_pre$hyp   <- factor(df_pre$hyp)

  imp  <- mice::mice(df_pre, m = 3, printFlag = FALSE, seed = 123)
  mild <- as_mild_list(imp)

  # PAM + Gower with global range normalization
  parts <- cluster_imputations(mild, method = "pam",
                               k = 2:4, distance = "gower")
  str(parts)
}
```

`consensus_clustering` *Build a consensus partition from multiple imputation clustering results*

Description

Constructs a consensus clustering solution from a collection of partitions obtained across multiple imputed datasets (as produced by `cluster_imputations`). The function builds a co-assignment matrix reflecting how frequently each pair of observations is assigned to the same cluster, and derives the final consensus partition via hierarchical clustering on the resulting dissimilarity. Supports both a single value of k and a range of values.

Usage

```
consensus_clustering(
  partitions,
  k = NULL,
  cluster_method = "ward.D2",
  consensus_method = c("classic", "weighted_ari")
)
```

Arguments

`partitions`	A list of cluster assignment vectors (one per imputed dataset) for a single k, or a named list of such lists (one per k value, named `"k2"`, `"k3"`, etc.) as returned by `cluster_imputations` when k is a vector.
`k`	An integer specifying the number of consensus clusters. Required when `partitions` is a flat list of vectors (single-k input). Ignored when `partitions` is a list-of-lists with names following the pattern `"k2"`, `"k3"`, etc., as k is inferred automatically from the names.
`cluster_method`	A character string specifying the agglomeration method passed to `hclust` for the consensus stage. Accepted values are `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"`, `"centroid"`, `"median"`, and `"mcquitty"`. Default is `"ward.D2"`.
`consensus_method`	A character string specifying how the co-assignment matrix is built. Options are: • `"classic"`: unweighted co-assignment; all partitions contribute equally. • `"weighted_ari"`: partitions are weighted by their pairwise ARI centrality, so more consistent partitions contribute more to the consensus. Requires **mclust**. Default is `"classic"`.

Details

The co-assignment matrix C is defined such that C_{ij} represents the weighted proportion of partitions in which observations i and j are assigned to the same cluster. The final dissimilarity is computed as $1 - C$ and passed to `hclust`.

When `consensus_method = "weighted_ari"`, the weight of each partition is proportional to its mean ARI against all other partitions, reflecting its centrality within the ensemble. Negative centrality values are floored at zero. If all centralities are zero, the function falls back to equal weights with a warning.

Value

- If `partitions` is a flat list (single `k`): a named list with the following elements:
  - `consensus_method`: character string with the method used.
  - `k`: integer, number of clusters requested.
  - `consensus`: integer vector of consensus cluster labels, one per observation.
  - `coassignment`: numeric matrix of size $n \times n$ with co-assignment probabilities between 0 and 1.
  - `hclust`: the `hclust` object from the consensus stage.
  - `weights`: named numeric vector of partition weights.
- If `partitions` is a list-of-lists (multiple `k`): a named list where each element corresponds to one value of `k` and contains the structure described above.

References

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, **52**(1-2), 91-118. doi:10.1023/A:1023949509487

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**, 193-218. doi:10.1007/BF01908075

See Also

`cluster_imputations`, `validate_clustering`

Examples

```
# ------------------------------------------------------------
# Example 1: Basic consensus clustering
# ------------------------------------------------------------
set.seed(123)
# simulate 3 partitions
partitions <- list(
  imp1 = sample(1:2, 10, replace = TRUE),
  imp2 = sample(1:2, 10, replace = TRUE),
  imp3 = sample(1:2, 10, replace = TRUE)
)

# Classic consensus
```

```
cons <- consensus_clustering(partitions, k = 2)
str(cons)


# ------------------------------------------------------------
# Example 2: consensus method with mice
# ------------------------------------------------------------
if (requireNamespace("mice", quietly = TRUE)) {

  set.seed(123)

  imp    <- mice::mice(mice::nhanes, m = 3, printFlag = FALSE)
  mild   <- as_mild_list(imp)
  parts  <- cluster_imputations(mild, method = "ward.D2", k = 3)

  # ------------------------------------------------------------
  # Single k, classic consensus
  # ------------------------------------------------------------
  cons <- consensus_clustering(parts, k = 3)

  # ------------------------------------------------------------
  # Single k, ARI-weighted consensus
 # ------------------------------------------------------------
  if (requireNamespace("mclust", quietly = TRUE)) {
    cons_w <- consensus_clustering(parts, k = 3,
                                   consensus_method = "weighted_ari")
}
  # ------------------------------------------------------------
  # Multiple k values
  # ------------------------------------------------------------
  parts_multi <- cluster_imputations(mild, method = "ward.D2", k = 2:4)
  cons_multi  <- consensus_clustering(parts_multi)
}
```

`plot_consensus_dendrogram` *Plot a consensus clustering dendrogram*

Description

Displays the hierarchical clustering dendrogram derived from the consensus dissimilarity matrix $1 - C$, where C is the co-assignment matrix produced by `consensus_clustering`. Optionally draws colored rectangles around the clusters defined by the requested `k`.

Usage

```
plot_consensus_dendrogram(
  consensus_result,
```

```
  k = NULL,
  rect = TRUE,
  hang = -1,
  labels = NULL
)
```

Arguments

consensus_result	A named list as returned by consensus_clustering for a single k. Must contain at least $hclust (an hclust object) and $k (an integer specifying the number of clusters).
k	Integer. The number of clusters to display. Only required when consensus_result contains results for multiple k values. Ignored if a single-k result is passed directly.
rect	Logical. If TRUE (default), colored rectangles are drawn around the clusters via rect.hclust.
hang	Numeric. The fraction of the plot height by which labels hang below the rest of the plot. A negative value (default -1) causes all labels to hang down from zero.
labels	Logical or character vector. Controls the labels displayed at the leaves of the dendrogram. If NULL (default), observation labels are shown. If FALSE, no labels are displayed — recommended for large datasets where labels become unreadable. A character vector of the same length as the number of observations can also be supplied to use custom labels.

Details

The dendrogram is built from the hclust object stored in consensus_result$hclust, which was computed by consensus_clustering on the consensus dissimilarity matrix $1 - C$. The colored rectangles highlight the k groups obtained by cutting the dendrogram at the level that yields exactly k clusters.

This function is intended to be called on a single k. To compare dendrograms across multiple values of k, call the function iteratively and manage the graphics layout externally via par.

Value

Invisibly returns NULL. The function is called for its side effect of producing a plot.

See Also

consensus_clustering, plot_consensus_matrix, plot_validation_metrics

Examples

```
# ------------------------------------------------------------
# Example 1: consensus dendrogram with simulated results
# ------------------------------------------------------------
set.seed(123)
hc <- hclust(dist(matrix(rnorm(60), nrow = 20)), method = "ward.D2")
```

```
cons <- list(hclust = hc, k = 3)

plot_consensus_dendrogram(cons)
plot_consensus_dendrogram(cons, rect = FALSE)


# ------------------------------------------------------------
# Example 2: Dendrogram with mice
# ------------------------------------------------------------

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

set.seed(123)
imp    <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)
mild   <- as_mild_list(imp)
parts  <- cluster_imputations(mild, method = "ward.D2", k = 3)
cons   <- consensus_clustering(parts, k = 3)

# Single dendrogram
plot_consensus_dendrogram(cons)

# Without rectangles
plot_consensus_dendrogram(cons, rect = FALSE)

# Compare multiple k values
parts_multi <- cluster_imputations(mild, method = "ward.D2", k = 2:4)
cons_multi  <- consensus_clustering(parts_multi)

par(mfrow = c(1, 3), mar = c(3, 2, 2, 1))
for (k_i in 2:4) {
  plot_consensus_dendrogram(cons_multi[[paste0("k", k_i)]])
}
par(mfrow = c(1, 1))
}
```

`plot_consensus_matrix` *Plot the consensus co-assignment matrix*

Description

Displays a heatmap of the co-assignment matrix C produced by `consensus_clustering`, where each cell C_{ij} represents the weighted proportion of partitions in which observations i and j were assigned to the same cluster. Values close to 1 indicate that two observations are consistently grouped together across all imputed datasets; values close to 0 indicate the opposite. Rows and columns can be reordered by consensus cluster assignment to make the block structure visually apparent.

Usage

```
plot_consensus_matrix(
  consensus_result,
  reorder = TRUE,
  show_labels = FALSE,
  viridis_option = "D"
)
```

Arguments

consensus_result	A named list as returned by consensus_clustering for a single k. Must contain at least $coassignment (a numeric $n \times n$ matrix) and, when reorder = TRUE, $consensus (an integer vector of cluster labels).
reorder	Logical. If TRUE (default), rows and columns are reordered by the consensus cluster assignment stored in consensus_result$consensus, making the block structure of the matrix visually apparent.
show_labels	Logical. If TRUE, observation indices are displayed on both axes, ordered according to the consensus cluster assignment. Recommended only for small datasets (n < 50) as labels become unreadable for large n. If FALSE (default), tick marks are shown without labels to preserve the axis frame without visual clutter.
viridis_option	A single character specifying the color palette passed to viridis. Accepted values are "A" (magma), "B" (inferno), "C" (plasma), and "D" (viridis, default). Low co-assignment values are mapped to dark colors and high values to bright colors.

Details

The matrix is rendered using image on the consensus dissimilarity $1 - C$ reordered by cluster membership. Original graphical parameters are restored on exit via par.

A well-defined consensus solution produces a block-diagonal pattern in the heatmap, where bright blocks along the diagonal correspond to observations that are consistently assigned to the same cluster, and dark off-diagonal regions indicate observations that are rarely grouped together.

Value

Invisibly returns NULL. The function is called for its side effect of producing a plot.

See Also

consensus_clustering, plot_consensus_dendrogram, plot_validation_metrics

Examples

```
# ------------------------------------------------------------
# Example 1: Consensus matrix with simulated consensus result
# ------------------------------------------------------------
set.seed(123)
```

```
n <- 20
true_labels <- rep(1:2, each = n / 2)
M <- matrix(0.1, n, n)
for (i in seq_len(n))
  for (j in seq_len(n))
    if (true_labels[i] == true_labels[j]) M[i, j] <- 0.9

cons <- list(coassignment = M,
             consensus    = true_labels,
             k            = 2)

plot_consensus_matrix(cons, viridis_option = "A")


# ------------------------------------------------------------
# Example 2: Consensus matrix with mice
# ------------------------------------------------------------

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

imp   <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)
mild  <- as_mild_list(imp)
parts <- cluster_imputations(mild, method = "ward.D2", k = 3)
cons  <- consensus_clustering(parts, k = 3)

# Default plot with viridis palette
plot_consensus_matrix(cons)

# Without reordering
plot_consensus_matrix(cons, reorder = FALSE)

# Compare color palettes
par(mfrow = c(2, 2), mar = c(4, 4, 2, 2))
for (opt in c("A", "B", "C", "D")) {
  plot_consensus_matrix(cons, viridis_option = opt)
}
par(mfrow = c(1, 1))
}
```

`plot_validation_metrics` *Plot validation metrics across candidate numbers of clusters*

Description

Produces a multi-panel line plot displaying one or more clustering validation metrics as a function of the number of clusters k, as computed by `validate_clustering` and optionally extended with a `score` column by `choose_best_clustering`. For each metric, the optimal value is highlighted in red.

Usage

```
plot_validation_metrics(
  validation_table,
  metrics = c("pac", "silhouette_mean", "ari_mean_between_imputations",
   "ari_consensus_mean", "calinski_harabasz_mean", "davies_bouldin_mean", "dunn_index",
    "score")
)
```

Arguments

validation_table	A data.frame as returned by validate_clustering or $scores_table from choose_best_clustering. Must contain a column k and at least one of the requested metric columns.
metrics	A character vector specifying which metrics to plot. Only metrics present in validation_table are plotted; others are silently ignored. Default includes all standard metrics plus score: "pac", "silhouette_mean", "ari_mean_between_imputations", "ari_consensus_mean", "calinski_harabasz_mean", "davies_bouldin_mean", "dunn_index", and "score".

Details

The number of panels is determined automatically from the number of available metrics using ceiling(sqrt(n)) rows and ceiling(n / nrow) columns, producing a layout as square as possible.

The direction of optimality is accounted for when highlighting the best value: lower is better for pac, davies_bouldin_mean, and score; higher is better for all remaining metrics. The optimal point for each metric is highlighted in red.

Original graphical parameters are restored on exit via par.

When score is present (i.e., validation_table comes from choose_best_clustering), the panel for score shows the overall weighted rank aggregation - the k with the lowest score is the recommended solution.

Value

Invisibly returns NULL. The function is called for its side effect of producing a plot.

See Also

validate_clustering, choose_best_clustering, plot_consensus_dendrogram, plot_consensus_matrix

Examples

```
# ------------------------------------------------------------
# Example 1: Validation metrics with simulated data
# ------------------------------------------------------------
val_table <- data.frame(
  k                           = 2:4,
```

```
  pac                          = c(0.15, 0.08, 0.20),
  silhouette_mean              = c(0.42, 0.61, 0.38),
  ari_mean_between_imputations = c(0.80, 0.91, 0.75),
  ari_consensus_mean           = c(0.82, 0.93, 0.77),
  calinski_harabasz_mean       = c(120,  198,  105),
  davies_bouldin_mean          = c(0.85, 0.54, 0.97),
  dunn_index                   = c(0.30, 0.48, 0.27)
)

# Plot all metrics
plot_validation_metrics(val_table)

# Plot only stability metrics
plot_validation_metrics(val_table,
                        metrics = c("pac",
                                    "ari_mean_between_imputations",
                                    "ari_consensus_mean"))

# ------------------------------------------------------------
# Example 2: metrics validation with mice
# ------------------------------------------------------------

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

set.seed(123)
imp    <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)
mild   <- as_mild_list(imp)
parts  <- cluster_imputations(mild, method = "ward.D2", k = 2:5)
cons   <- consensus_clustering(parts)
val    <- validate_clustering(parts, cons)

# Plot all available metrics (no score column yet)
plot_validation_metrics(val)

# Include score column from choose_best_clustering
best <- choose_best_clustering(val, cons)
plot_validation_metrics(best)

# Plot only stability metrics
plot_validation_metrics(val,
                        metrics = c("pac",
                                    "ari_mean_between_imputations",
                                    "ari_consensus_mean"))
}
```

---

run_mi_clustering *Run the full multiple-imputation clustering pipeline*

Description

Executes the complete multiple-imputation clustering pipeline in a single call, integrating imputation standardization, per-imputation clustering, consensus construction, validation, and optimal k selection. Designed as a convenience wrapper around as_mild_list, cluster_imputations, consensus_clustering, validate_clustering, and choose_best_clustering. When a single value of k is supplied, the selection step is skipped and the unique solution is returned directly.

Usage

```
run_mi_clustering(
  data,
  method = "ward.D2",
  k,
  scale_data = "global",
  distance = "euclidean",
  dist_fun = NULL,
  dist_args = list(),
  consensus_method = c("classic", "weighted_ari"),
  cluster_method_consensus = "ward.D2",
  pac_lower = 0.1,
  pac_upper = 0.9,
  weights = NULL,
  prefer_stability = TRUE,
  tie_breaker = c("silhouette", "pac", "dunn", "ch", "db", "ari_between",
    "ari_consensus"),
  ...
)
```

Arguments

data	An imputation object accepted by as_mild_list. Supported formats include mids objects from **mice**, long-format data frames with a .imp column, amelia objects, imputationList objects from **mitools**, and plain lists of data.frames.
method	A character string specifying the clustering algorithm applied to each imputed dataset. For hierarchical clustering, accepted values are "ward.D", "ward.D2", "single", "complete", "average", "centroid", "median", and "mcquitty". Additional options are "kmeans", "pam", "fuzzy", "mclust", "kmodes", and "kprototypes". Default is "ward.D2".
k	A single integer or an integer vector specifying the number(s) of clusters to evaluate. If a single value is supplied, the pipeline skips the choose_best_clustering step and returns that solution directly. If a vector is supplied, all values are evaluated and the optimal k is selected via weighted rank aggregation.
scale_data	A character string specifying how numeric columns are standardized prior to clustering. Accepted values are "global" (default, pooled mean and standard deviation computed across all imputations without stacking), "within" (each imputed dataset scaled independently), and "none" (no scaling applied). Scaling is automatically disabled for distance = "gower", "jaccard", or "simple_matching", and for method = "kmodes" and "kprototypes".

`distance` A character string specifying the distance metric used for hierarchical clustering and `method = "pam"`. Options are `"euclidean"` (default), `"manhattan"`, `"gower"` (via `cluster::daisy()`, supports mixed data), `"jaccard"` (binary data, requires **proxy**), `"simple_matching"` (binary data, requires **proxy**), and `"custom"` (user-defined function via `dist_fun`). Ignored for `method = "kmeans"`, `"fuzzy"`, `"mclust"`, `"kmodes"`, and `"kprototypes"`, which compute distances internally.

`dist_fun` A function that takes a `data.frame` and returns a `dist` object. Required when `distance = "custom"`; ignored otherwise.

`dist_args` A named list of additional arguments passed to the distance function. For `distance = "gower"`, forwarded to `cluster::daisy()`; for `"jaccard"` or `"simple_matching"`, to `proxy::dist()`; for `"euclidean"` or `"manhattan"`, to `stats::dist()`; also for `"custom"`. Default is `list()`.

`consensus_method` A character string specifying the method used to build the co-assignment matrix. `"classic"` assigns equal weight to all partitions; `"weighted_ari"` weights partitions by their pairwise ARI centrality. Default is `"classic"`.

`cluster_method_consensus` A character string specifying the agglomeration method passed to `hclust` during the consensus stage. Accepted values are the same as for `method` when using hierarchical clustering. Default is `"ward.D2"`.

`pac_lower` A numeric value between 0 and 1 specifying the lower bound of the ambiguous assignment region for the PAC metric. Default is `0.1`.

`pac_upper` A numeric value between 0 and 1 specifying the upper bound of the ambiguous assignment region for the PAC metric. Default is `0.9`.

`weights` A named numeric vector specifying the relative importance of each metric in the rank aggregation used by `choose_best_clustering`. Expected names are `pac`, `silhouette`, `ari_between`, `ari_consensus`, `ch`, `db`, and `dunn`. If `NULL` (default), weights are set automatically based on `prefer_stability`. Ignored when `k` is a single value.

`prefer_stability` Logical. If `TRUE` (default), stability metrics (`pac`, `ari_between`, `ari_consensus`) receive higher weights during `k` selection. If `FALSE`, internal compactness metrics (`silhouette`, `ch`, `db`, `dunn`) receive higher weights. Ignored when `k` is a single value or when `weights` is provided explicitly.

`tie_breaker` A character string specifying the secondary metric used to break ties during `k` selection. One of `"silhouette"` (default), `"pac"`, `"dunn"`, `"ch"`, `"db"`, `"ari_between"`, or `"ari_consensus"`. Ignored when `k` is a single value.

`...` Additional arguments passed to the underlying clustering function via `cluster_imputations`: `stats::kmeans`, `e1071::cmeans`, `mclust::Mclust`, `klaR::kmodes`, or `clustMixType::kproto`. Not used for `method = "pam"`.

Details

The pipeline proceeds as follows:

1. `as_mild_list` standardizes the imputation object and validates data quality.

2. cluster_imputations applies the chosen algorithm to each imputed dataset, optionally scaling the data beforehand.
3. consensus_clustering builds a co-assignment matrix and derives the consensus partition for each requested k.
4. validate_clustering computes internal and stability metrics for each candidate k.
5. choose_best_clustering selects the optimal k via weighted rank aggregation (only when length(k) > 1).

When length(k) == 1, step 5 is bypassed and the unique solution is returned directly, with weights and tie_breaker set to NULL in the selection element.

Value

An object of class "mi_clustering_result": a named list with the following elements:

- call: the matched call.
- input_k: the value(s) of k supplied by the user.
- clustering_method: character string with the clustering method used.
- consensus_method: character string with the consensus method used.
- scale_data: A character string specifying how numeric columns are standardized prior to clustering
- imputations: standardized list of imputed datasets, as returned by as_mild_list.
- partitions: clustering assignments per imputed dataset, as returned by cluster_imputations.
- consensus_results: consensus clustering output, as returned by consensus_clustering.
- validation_table: validation metrics per k, as returned by validate_clustering.
- best_k: integer, the selected optimal number of clusters.
- best_consensus: integer vector of consensus cluster labels for the selected k, one per observation.
- best_coassignment: numeric matrix of co-assignment probabilities for the selected k.
- best_consensus_result: the full consensus result object for the selected k.
- scores_table: validation table with an additional score column when k is a vector; the plain validation table when k is a single value.
- selection: the full output of choose_best_clustering when k is a vector, or a minimal equivalent list when k is a single value.

References

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, **52**(1-2), 91-118. doi:10.1023/A:1023949509487

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

Pihur, V., Datta, S., & Datta, S. (2007). Weighted rank aggregation of cluster validation measures. *Bioinformatics*, **23**(13), 1607-1615. doi:10.1093/bioinformatics/btm158

See Also

as_mild_list, cluster_imputations, consensus_clustering, validate_clustering, choose_best_clustering

Examples

```
# ------------------------------------------------------------
# Example 1: Simulated list of imputed datasets
# ------------------------------------------------------------
set.seed(123)
make_imputation <- function() {
  data.frame(
    x1 = rnorm(30),
    x2 = rnorm(30),
    x3 = rnorm(30)
  )
}

imp_list <- lapply(1:5, function(i) make_imputation())

# Single k
res <- run_mi_clustering(imp_list, method = "ward.D2", k = 3)
res$best_k
res$best_consensus

# Range of k values (automatic selection)
res_multi <- run_mi_clustering(imp_list, method = "ward.D2", k = 2:4)
res_multi$best_k
res_multi$validation_table


# ------------------------------------------------------------
# Example 2: Full pipeline with mice
# ------------------------------------------------------------
if (requireNamespace("mice", quietly = TRUE)) {

set.seed(123)
imp <- mice::mice(mice::nhanes, m = 5, printFlag = FALSE)

# Single k
res <- run_mi_clustering(imp, method = "ward.D2", k = 3)
res$best_k
res$best_consensus

# Range of k values (automatic selection)
res_multi <- run_mi_clustering(imp, method = "ward.D2", k = 2:5)
res_multi$best_k
res_multi$validation_table

# PAM with Gower distance (mixed data)
res_gower <- run_mi_clustering(imp, method = "pam", k = 2:4,
                               distance = "gower")
```

```
# ARI-weighted consensus, compactness-focused selection
res_w <- run_mi_clustering(imp, method = "ward.D2", k = 2:5,
                           consensus_method  = "weighted_ari",
                           prefer_stability  = FALSE,
                           tie_breaker       = "dunn")
}
```

`validate_clustering` *Validate consensus clustering results across multiple imputed datasets*

Description

Computes a comprehensive set of internal and stability validation metrics for consensus clustering results obtained from multiple imputed datasets. All internal metrics are computed on the consensus dissimilarity matrix $1 - C$, where C is the co-assignment matrix, making the function robust to numeric, categorical, and mixed data types. Supports both single-k and multi-k evaluation.

Usage

```
validate_clustering(
  partitions,
  consensus_results,
  pac_lower = 0.1,
  pac_upper = 0.9
)
```

Arguments

`partitions`	A list of cluster assignment vectors (one per imputed dataset) for a single `k`, or a named list of such lists (one per `k` value, named `"k2"`, `"k3"`, etc.) as returned by `cluster_imputations`.
`consensus_results`	A list as returned by `consensus_clustering`, either for a single `k` or for multiple `k` values. Must contain at least `$consensus` and `$coassignment` for each `k`.
`pac_lower`	A numeric value between 0 and 1 specifying the lower bound of the ambiguous assignment region for the PAC metric. Default is `0.1`.
`pac_upper`	A numeric value between 0 and 1 specifying the upper bound of the ambiguous assignment region for the PAC metric. Default is `0.9`.

Details

All internal validation metrics (silhouette, Calinski-Harabasz, Davies-Bouldin, Dunn) are computed exclusively on the consensus dissimilarity matrix $1 - C$ rather than on the original feature space.

This design choice ensures compatibility with any data type and avoids recomputing distances from the raw imputed datasets.

The Davies-Bouldin index is computed using a custom medoid-based implementation derived from the consensus dissimilarity, as no standard R implementation accepts a precomputed dissimilarity matrix directly.

Helper functions prefixed with a dot (e.g., `.pac_from_coassignment`, `.mean_silhouette_from_diss`) are internal and not exported.

Value

A `data.frame` with one row per evaluated `k` and the following columns:

- `k`: number of clusters evaluated.
- `pac`: Proportion of Ambiguous Clusterings. Lower values indicate a more stable consensus. Computed from the upper triangle of the co-assignment matrix.
- `silhouette_mean`: mean silhouette width computed on the consensus dissimilarity $1 - C$. Higher values indicate better-defined clusters.
- `ari_mean_between_imputations`: mean pairwise Adjusted Rand Index (ARI) across all pairs of imputed partitions. Higher values indicate greater clustering stability.
- `ari_consensus_mean`: mean ARI between the consensus partition and each individual imputed partition. Higher values indicate that the consensus is representative of the ensemble.
- `calinski_harabasz_mean`: Calinski-Harabasz index computed via `fpc::cluster.stats()`. Higher values indicate more compact and well-separated clusters.
- `davies_bouldin_mean`: Davies-Bouldin index computed using a medoid-based approach on the consensus dissimilarity. Lower values indicate better cluster separation.
- `dunn_index`: Dunn index computed via `fpc::cluster.stats()`. Higher values indicate better cluster compactness and separation.

References

Senbabaoglu, Y., Michailidis, G., & Li, J. Z. (2014). Critical limitations of consensus clustering in class discovery. *Scientific Reports*, **4**, 6207. doi:10.1038/srep06207

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, **20**, 53-65. doi:10.1016/0377-0427(87)901257

Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**, 1-27. doi:10.1080/03610927408827101

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1**, 224-227. doi:10.1109/TPAMI.1979.4766909

Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, **4**, 95-104. doi:10.1080/01969727408546059

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**, 193-218. doi:10.1007/BF01908075

See Also

consensus_clustering, choose_best_clustering

Examples

```
# ------------------------------------------------------------
# Example 1: Basic validation with simulated partitions
# ------------------------------------------------------------
set.seed(123)

# Simulate partitions (3 imputations, k = 2)
parts <- list(
  imp1 = sample(1:2, 20, replace = TRUE),
  imp2 = sample(1:2, 20, replace = TRUE),
  imp3 = sample(1:2, 20, replace = TRUE)
)

cons <- consensus_clustering(parts, k = 2)
val  <- validate_clustering(parts, cons)

print(val)


# ------------------------------------------------------------
# Example 2: Full pipeline with mice (multiple k)
# ------------------------------------------------------------
if (requireNamespace("mice", quietly = TRUE)) {

  set.seed(123)

  imp  <- mice::mice(mice::nhanes, m = 3, printFlag = FALSE)
  mild <- as_mild_list(imp)

  parts <- cluster_imputations(mild, method = "ward.D2", k = 2:3)
  cons  <- consensus_clustering(parts)

  val <- validate_clustering(parts, cons)

  print(val)
}
```

Index