Package 'phynotype'

September 1, 2026

Title Clustering and Consensus Meta-Clustering

Version 0.6.2

Author Imad El Badisy [aut, cre]

Maintainer Imad El Badisy <elbadisyimad@gmail.com>

Description Tools for clustering, consensus meta-clustering,
validation, exploratory interpretation, cluster prediction, and
plotting. The package provides a clustering workflow with consensus clustering following Strehl and Ghosh (2002)
<https://www.jmlr.org/papers/v3/strehl02a.html>.

URL https://CRAN.R-project.org/package=phynotype

BugReports https://github.com/ielbadisy/phynotype/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports cluster, clustMixType, dbscan, functionals, ggplot2, ggrepel,
mclust, parallel, rlang, stats, utils

Suggests FactoMineR, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-01 11:30:02 UTC

Contents

`phynotype-package` *phynotype: Clustering Workflows and Consensus Meta-Clustering*

Description

`phynotype` provides tools for unsupervised phenotyping workflows. The six-step pipeline is:

1. **Cluster**: fit a single clustering solution with `cluster()`.
2. **Meta-cluster**: aggregate candidate solutions into a consensus partition with `metacluster()`.
3. **Validate**: score a solution with internal and external metrics using `validate()`.
4. **Explore**: summarize cluster sizes, feature profiles, and two-dimensional embeddings with `explore()`.
5. **Predict**: assign new observations to learned clusters with `predict.cluster_fit()`.
6. **Interpret**: explain the clustering rule with `feature_importance()`, `lime_explain()`, and `ceteris_paribus()`.

All steps operate through S3 generic functions and return structured objects with `print()`, `summary()`, and `plot()` methods.

Supported clustering methods

`"kmeans"` Lloyd's k-means algorithm for numeric data.

`"pam"` Partitioning Around Medoids for numeric data.

`"hclust"` Hierarchical clustering via `stats::hclust()`.

`"agnes"` Agglomerative nesting via `cluster::agnes()`.

`"dbscan"` Density-based spatial clustering via `dbscan::dbscan()`.

`"gmm"` Gaussian mixture models via `mclust::Mclust()`.

`"kproto"` K-prototypes for mixed numeric/categorical data.

`"kmm"` K-Mixed-Modes, a native mixed-data algorithm.

Validation metrics

Internal metrics: silhouette width (Rousseeuw, 1987), Calinski-Harabasz index (Calinski and Harabasz, 1974), Davies-Bouldin index (Davies and Bouldin, 1979), total within-cluster sum of squares, and bootstrap ARI. External metrics (when reference labels are available): adjusted Rand index (Hubert and Arabie, 1985) and normalized mutual information (Strehl and Ghosh, 2002).

Mixed-type data

Use `prepare_mixed_data()` to encode mixed-type data frames into a numeric matrix before distance-based methods. Use `mixed_distance()` to build a Gower distance matrix (Gower, 1971) before hierarchical methods. The exploratory embedding layer automatically selects PCA for numeric data, FAMD for mixed numeric/categorical data, MCA for categorical data, and classical MDS for distance objects.

Author(s)

Maintainer: Imad EL BADISY <elbadisyimad@gmail.com>

References

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.

Calinski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**(1), 1–27.

Davies, D.L. and Bouldin, D.W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1**(2), 224–227.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**(1), 193–218.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, **3**, 583–617.

Gower, J.C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, **27**(4), 857–874.

See Also

`cluster()`, `metacluster()`, `validate()`, `explore()`, `predict.cluster_fit()`, `feature_importance()`, `lime_explain()`, `ceteris_paribus()`

`adjusted_rand_index` *Adjusted Rand Index*

Description

Computes the Adjusted Rand Index (ARI) between two partitions, correcting the Rand Index for chance agreement (Hubert and Arabie, 1985). ARI equals 1 for identical partitions and has expectation 0 under independent random labelings.

Usage

```
adjusted_rand_index(x, y)
```

Arguments

`x`	Integer or factor vector of cluster labels (first partition).
`y`	Integer or factor vector of cluster labels (second partition). Must have the same length as `x`.

Value

A single numeric value in $(-\infty, 1]$.

References

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**(1), 193–218.

Examples

```
adjusted_rand_index(c(1, 1, 2, 2), c(1, 1, 2, 3))
```

centers *Cluster centers*

Description

Extract the numeric cluster center matrix. Available for k-means and hierarchical methods trained on numeric data; returns NULL for PAM, k-prototypes, and GMM (use prototypes() or membership() instead).

Usage

```
centers(x, ...)
```

Arguments

x	A cluster_fit or metacluster_fit object.
...	Unused.

Value

A numeric matrix with one row per cluster and one column per feature, or NULL.

See Also

prototypes(), clusters(), membership()

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
centers(fit)
```

ceteris_paribus *Ceteris paribus profiles for clustering*

Description

Compute individual conditional profiles for a fitted clustering rule. A ceteris paribus profile varies one feature over a grid while holding all other features fixed at an observed profile. It describes how the predicted cluster assignment or a cluster score changes locally as a single feature is perturbed.

Usage

```
ceteris_paribus(
  object,
  new_data,
  features = NULL,
  grid = NULL,
  grid_size = 25L,
  target = c("cluster", "score"),
  cluster = NULL,
  parallel = FALSE,
  cores = NULL,
  workers = NULL,
  progress = FALSE,
  ...
)
```

Arguments

object	A cluster_fit object with prediction support.
new_data	Row-by-feature data containing observations to explain.
features	Optional character vector or numeric column index specifying features to profile.
grid	Optional vector or named list of vectors giving profile values. If NULL, numeric features use quantile grids and categorical features use their observed levels.
grid_size	Number of grid values per feature when grid = NULL.
target	Output target. "cluster" records predicted labels; "score" records a cluster-specific prediction score.
cluster	Optional cluster whose score should be profiled when target = "score". If NULL, each observation is profiled for its baseline predicted cluster.
parallel	Logical; if TRUE, use functionals::fmap() when the suggested functionals package is installed.
cores	Optional positive integer number of cores passed to functionals::fmap() when parallel = TRUE.
workers	Optional number of workers passed to functionals::fmap(). Deprecated alias for cores.
progress	Logical; if TRUE, request progress reporting from functionals::fmap().
...	Reserved for future extensions.

Details

For observation x_i, feature j, and grid value z, define $x_{i,-j}$ as all coordinates of x_i except feature j. The ceteris paribus profile is

$$CP_{i,j}(z) = g\{f(x_{i,-j}, z)\},$$

where f is the fitted clustering rule and g extracts either the predicted cluster label or a cluster-specific score. For methods that return membership probabilities, the score is the predicted membership. For distance-based methods, phynotype converts distances into normalized radial similarity scores so that larger values indicate stronger association with a cluster prototype.

These profiles are local diagnostic curves. They should be interpreted as behavior of the fitted clustering rule under controlled perturbation, not as causal effects.

Value

A `ceteris_paribus` object with a tidy `profiles` data frame. Each row records: the observation index, the feature being varied, the grid value at that step, the predicted target (cluster label or score), the observed feature value, and the baseline prediction.

References

Biecek, P. and Burzykowski, T. (2021). *Explanatory Model Analysis*. Chapman and Hall/CRC, Boca Raton. https://ema.drwhy.ai/

Gosiewska, A. and Biecek, P. (2019). iBreakDown: Uncertainty of model explanations for non-additive predictive models. *arXiv:1903.11420*.

Apley, D.W. and Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B*, **82**(4), 1059–1086.

See Also

`lime_explain()` for local linear surrogate explanations, `feature_importance()` for global permutation importance, `predict.cluster_fit()` which is used internally.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
cp <- ceteris_paribus(fit, iris[1:3, 1:4], features = 1:2, grid_size = 10)
head(cp$profiles)
```

`cluster` *Fit a clustering solution*

Description

Fit a single clustering solution using a supported method.

Usage

```
cluster(
  x,
  method = "kmeans",
  ...,
  k = NULL,
```

```
  scale = FALSE,
  center = TRUE,
  seed = NULL
)
```

Arguments

x	Numeric matrix/data frame for numeric methods, a dist object for hierarchical methods, or a mixed-type data frame for "kproto" and other mixed-data methods.
method	Clustering method. Supported values are "kmeans", "pam", "hclust", "agnes", "dbscan", "gmm", "kproto", and "kmm".
...	Additional method-specific arguments passed to the underlying engine (e.g. nstart, iter.max, linkage, eps, minPts, lambda).
k	Number of clusters for methods that require it ("kmeans", "pam", "hclust", "agnes", "gmm", "kproto", "kmm").
scale	Logical; if TRUE, scale columns before fitting.
center	Logical; if TRUE, center columns before fitting.
seed	Optional integer random seed for reproducible results.

Details

cluster() is the single entry point for all supported methods. It dispatches to a method-specific backend registered via the internal cluster_registry, validates inputs, applies any requested preprocessing, and returns a cluster_fit object with a uniform interface regardless of the underlying engine.

Method-specific notes:

"kmeans" Minimizes total within-cluster sum of squares $\sum_j \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$ (Lloyd, 1982). Requires numeric x and k.

"pam" Partitioning Around Medoids: minimizes $\sum_i \min_g d(x_i, m_g)$ over actual observations (Kaufman and Rousseeuw, 1990). Requires the cluster package.

"hclust" / "agnes" Hierarchical agglomerative methods. Accept a precomputed dist object or raw numeric data. Specify linkage to choose the merging criterion (default "complete" for hclust, "average" for agnes). "agnes" requires the cluster package.

"dbscan" Density-Based Spatial Clustering of Applications with Noise (Ester et al., 1996). Requires eps (neighborhood radius) and minPts. Observations in sparse regions are labeled noise (cluster 0). Requires the dbscan package.

"gmm" Gaussian Mixture Model fitted via the EM algorithm (Fraley and Raftery, 2002). Returns soft membership probabilities. Requires the mclust package.

"kproto" K-prototypes for mixed numeric/categorical data (Huang, 1998). Use lambda to control the numeric-vs-categorical trade-off. Requires the clustMixType package.

"kmm" K-Mixed-Modes: a native mixed-data algorithm that minimizes a weighted prototype distance combining squared Euclidean and Hamming contributions controlled by lambda.

Value

A `cluster_fit` object with components:

`method` Character; the method used.

`clusters` Integer vector of cluster assignments.

`n_clusters` Integer; number of clusters found.

`centers` Cluster centers (numeric methods) or `NULL`.

`prototypes` Cluster prototypes (mixed methods) or `NULL`.

`membership` Soft membership matrix (GMM) or `NULL`.

`data_info` List of input metadata for downstream use.

References

Lloyd, S.P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, **28**(2), 129–137.

Kaufman, L. and Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York.

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the 2nd ACM SIGKDD*, pp. 226–231.

Fraley, C. and Raftery, A.E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, **97**(458), 611–631.

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.

See Also

`validate()` to score the fit, `explore()` to summarize cluster structure, `predict.cluster_fit()` to assign new observations, `metacluster()` for consensus clustering, `feature_importance()`, `lime_explain()`, `ceteris_paribus()` for interpretability.

Examples

```
# Numeric data: k-means
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
fit
clusters(fit)
centers(fit)

# PAM (requires the cluster package)
if (requireNamespace("cluster", quietly = TRUE)) {
  fit_pam <- cluster(iris[, 1:4], method = "pam", k = 3)
  fit_pam
}

# Hierarchical clustering from a distance object
d <- mixed_distance(iris[, 1:4])
fit_hc <- cluster(d, method = "hclust", k = 3)
```

```
fit_hc

# Mixed data: k-prototypes
if (requireNamespace("clustMixType", quietly = TRUE)) {
  fit_kp <- cluster(iris, method = "kproto", k = 3, seed = 1)
  fit_kp
}
```

`clusters` *Cluster assignments*

Description

Extract the integer cluster assignment vector from a fitted clustering object.

Usage

```
clusters(x, ...)
```

Arguments

`x`	A `cluster_fit` or `metacluster_fit` object.
`...`	Unused.

Value

Integer vector of length equal to the number of training observations.

See Also

`sizes()`, `n_clusters()`, `centers()`, `membership()`

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
clusters(fit)
```

`explore` *Explore clustering structure*

Description

Summarize cluster sizes, feature profiles, feature separation, and a two-dimensional embedding for a fitted clustering solution.

Usage

```
explore(
  x,
  data = NULL,
  embedding = c("auto", "pca", "famd", "mca", "mds"),
  ...
)
```

Arguments

`x`	A `cluster_fit` or `metacluster_fit` object.
`data`	Optional numeric matrix, data frame, or distance object. Defaults to the training data stored in `x`.
`embedding`	Embedding method. `"auto"` chooses PCA for numeric data, FAMD for mixed numeric/categorical data, MCA for categorical data, and classical MDS for distance objects. Explicit values `"pca"`, `"famd"`, `"mca"`, and `"mds"` are also supported.
`...`	Unused.

Details

`explore()` computes four summaries:

Size table Number of observations per cluster.

Feature summary Per-cluster mean, standard deviation, median, min, and max for each feature.

Separation table The eta-squared statistic for each feature, $\eta_j^2 = \mathrm{SS}_{B,j}/\mathrm{SS}_{T,j}$, measuring how much between-cluster variance each feature explains.

Embedding Two-dimensional projection for visualization (see `plot_clusters()`).

Value

A `cluster_explore` object with components:

`size_table` Data frame with cluster sizes.

`feature_summary` Data frame with per-cluster descriptive statistics for each feature.

`separation_table` Data frame with per-feature eta-squared values.

`embedding` Data frame with two embedding coordinates and cluster labels.

See Also

`cluster()` to fit, `plot_clusters()` to plot the embedding, `plot_feature_profiles()` to plot feature profiles.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
exp <- explore(fit)
exp$size_table
head(exp$feature_summary)
exp$separation_table
```

---

feature_importance *Permutation feature importance for clustering*

Description

Estimate global feature importance by measuring how much a fitted clustering solution changes when each feature is independently permuted. The method is model-agnostic: it uses the fitted object's `predict()` method and therefore applies to clustering engines with native prediction support.

Usage

```
feature_importance(
  object,
  data = NULL,
  features = NULL,
  metric = c("instability", "silhouette", "total_within"),
  loss = NULL,
  n_repeats = 10L,
  seed = NULL,
  parallel = FALSE,
  cores = NULL,
  workers = NULL,
  progress = FALSE,
  ...
)
```

Arguments

`object`	A `cluster_fit` object with prediction support.
`data`	Optional row-by-feature data used for evaluation. Defaults to the training data stored in `object`.
`features`	Optional character vector or numeric column index specifying features to evaluate.

`metric`	Built-in importance metric. `"instability"` measures the fraction of changed cluster assignments. `"silhouette"` measures loss of mean silhouette width. `"total_within"` measures increase in within-cluster dispersion.
`loss`	Optional custom loss function with signature `loss(data, clusters, object)`.
`n_repeats`	Number of independent permutations per feature.
`seed`	Optional integer random seed.
`parallel`	Logical; if `TRUE`, use `functionals::fmap()` when the suggested `functionals` package is installed.
`cores`	Optional positive integer number of cores passed to `functionals::fmap()` when `parallel = TRUE`.
`workers`	Optional number of workers passed to `functionals::fmap()`. Deprecated alias for `cores`.
`progress`	Logical; if `TRUE`, request progress reporting from `functionals::fmap()`.
`...`	Reserved for future extensions.

Details

Let f denote the fitted clustering rule, X the evaluation data, $\hat{c} = f(X)$ the baseline predicted partition, and $X^{(m)}_{\pi(j)}$ the data matrix where feature j has been permuted in repetition m. For the default instability metric, feature importance is

$$FI_j = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{i=1}^{n} I\{\hat{c}_i \neq f(X^{(m)}_{\pi(j)})_i\}.$$

This quantity estimates the expected fraction of assignments that change when the marginal information in feature j is broken. Larger values indicate that the fitted clustering rule relies more strongly on that feature.

For score-style internal metrics, the package computes a baseline score $S(f, X)$ and a permuted score $S(f, X^{(m)}_{\pi(j)})$. For silhouette, larger is better and

$$FI_j = \frac{1}{M}\sum_{m=1}^{M}\left[S(f, X) - S(f, X^{(m)}_{\pi(j)})\right].$$

For total within-cluster sum of squares, smaller is better and

$$FI_j = \frac{1}{M}\sum_{m=1}^{M}\left[W(f, X^{(m)}_{\pi(j)}) - W(f, X)\right].$$

Users can pass a custom `loss` function for domain-specific objectives. It must accept `(data, clusters, object)` and return a numeric scalar where larger values mean worse fit.

Value

A `feature_importance` object with components:

`results` Data frame with one row per (feature, repeat) combination recording the baseline loss, the permuted loss, and the computed importance.

`summary` Data frame with one row per feature, containing the mean importance, standard error, and number of repeats.

References

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32. (Permutation importance concept introduced in the context of random forests.)

Fisher, A., Rudin, C. and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, **20**(177), 1–81.

See Also

`lime_explain()` for local explanations, `ceteris_paribus()` for individual conditional profiles, `predict.cluster_fit()` which is used internally.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
imp <- feature_importance(fit, n_repeats = 3, seed = 1)
imp$summary
```

---

lime_explain *LIME-style local explanations for clustering*

Description

Explain individual cluster assignments or cluster scores with a locally weighted surrogate model. The method builds a synthetic neighborhood around each observation, predicts the fitted clustering behavior in that neighborhood, weights perturbed samples by proximity to the explained observation, and fits a simple weighted linear model.

Usage

```
lime_explain(
  object,
  new_data,
  n_features = 5L,
  n_permutations = 500L,
  kernel_width = NULL,
  target = c("cluster", "score"),
  cluster = NULL,
```

```
  seed = NULL,
  parallel = FALSE,
  cores = NULL,
  workers = NULL,
  progress = FALSE,
  ...
)
```

Arguments

object	A cluster_fit object with prediction support.
new_data	Row-by-feature data containing observations to explain.
n_features	Maximum number of local effects returned per observation.
n_permutations	Number of perturbed neighborhood samples per observation.
kernel_width	Positive numeric kernel width. If NULL, uses 0.75 * sqrt(p), where p is the number of features.
target	Output target. "cluster" explains a one-vs-cluster local assignment indicator; "score" explains a cluster-specific prediction score.
cluster	Optional cluster to explain. If NULL, each observation is explained for its baseline predicted cluster.
seed	Optional integer random seed.
parallel	Logical; if TRUE, use functionals::fmap() when the suggested functionals package is installed.
cores	Optional positive integer number of cores passed to functionals::fmap() when parallel = TRUE.
workers	Optional number of workers passed to functionals::fmap(). Deprecated alias for cores.
progress	Logical; if TRUE, request progress reporting from functionals::fmap().
...	Reserved for future extensions.

Details

For an observation x, perturbed samples z_i, fitted clustering rule f, target extractor g, local surrogate h, and kernel $\pi_x(z_i)$, the LIME objective is

$$\hat{h} = \arg\min_{h \in H} \sum_{i=1}^{N} \pi_x(z_i) \left[g\{f(z_i)\} - h(z_i) \right]^2 + \Omega(h).$$

phynotype uses a weighted linear surrogate for H. Feature ranking is based on the absolute fitted local coefficients after standardizing features by the training-data scale. The penalty term $\Omega(h)$ is represented operationally by returning the largest n_features effects, which keeps the explanation sparse and readable without adding a heavy modeling dependency.

For target = "cluster", $g\{f(z_i)\}$ is an indicator that the predicted cluster equals the cluster being explained. For target = "score", it is the cluster-specific membership or similarity score.

Value

A `lime_explanation` object with components:

`explanations` Tidy data frame with one row per (observation, feature) pair, containing the local surrogate coefficient, absolute effect size, direction, and rank.

`neighborhoods` Data frame with all perturbed samples, their predicted responses, kernel weights, and distances to the explained observation.

References

Ribeiro, M.T., Singh, S. and Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144.

See Also

`ceteris_paribus()` for individual conditional profiles, `feature_importance()` for global importance, `predict.cluster_fit()` which is used internally.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
lx <- lime_explain(fit, iris[1:2, 1:4], n_permutations = 50, seed = 1)
lx$explanations
```

`membership` *Membership matrix*

Description

Extract the soft membership (posterior probability) matrix. Only methods that return probabilistic assignments (e.g. GMM) populate this slot; all other methods return `NULL`.

Usage

```
membership(x, ...)
```

Arguments

`x`	A `cluster_fit` or `metacluster_fit` object.
`...`	Unused.

Value

A numeric matrix with one row per observation and one column per cluster, or `NULL`.

See Also

clusters(), centers(), prototypes()

Examples

```
if (requireNamespace("mclust", quietly = TRUE)) {
  fit <- cluster(iris[, 1:4], method = "gmm", k = 3, seed = 1)
  head(membership(fit))
}
```

---

metacluster *Fit a consensus meta-clustering solution*

Description

Fit several candidate clustering solutions and summarize their agreement through a co-association consensus matrix. The consensus partition is obtained by hierarchical clustering of the dissimilarity matrix $D = 1 - C$, where C_{ij} is the proportion of candidate partitions that place observations i and j in the same cluster.

Usage

```
metacluster(
  x,
  methods = c("kmeans", "pam", "hclust"),
  k = 2:6,
  consensus = "coassoc",
  scale = FALSE,
  center = TRUE,
  seed = NULL,
  ...
)
```

Arguments

x	Numeric matrix or numeric data frame.
methods	Character vector of candidate clustering methods. Any method supported by cluster() may be used (e.g. c("kmeans", "pam", "hclust")).
k	Integer vector of candidate cluster counts. All combinations of methods and k are fitted and pooled.
consensus	Consensus strategy. Currently only "coassoc" (Evidence Accumulation Clustering) is implemented.
scale	Logical; if TRUE, scale columns before fitting candidates.
center	Logical; if TRUE, center columns before fitting candidates.
seed	Optional integer random seed. Each candidate is given a deterministic offset seed so results are jointly reproducible.
...	Additional arguments passed through to cluster().

Details

Co-association consensus:

For B candidate partitions, the co-association matrix is

$$C_{ij} = \frac{1}{B} \sum_{b=1}^{B} I\{i \text{ and } j \text{ are in the same cluster in partition } b\}.$$

This estimator is related to the Evidence Accumulation Clustering framework (Fred and Jain, 2002). The consensus dissimilarity $D = 1 - C$ is then hierarchically clustered (average linkage), and the optimal number of clusters is chosen by maximizing the mean silhouette width.

Stability of the ensemble is summarized by the mean pairwise partition agreement (PPA) across all candidate pairs:

$$\mathrm{PPA}(U, V) = \frac{1}{\binom{n}{2}} \sum_{i<j} I\{(u_i = u_j) = (v_i = v_j)\}.$$

Value

A `metacluster_fit` object with components:

`final_clusters` Integer vector of final consensus assignments.

`final_k` Integer; the selected number of clusters.

`coassoc_matrix` The $n \times n$ co-association matrix.

`candidate_table` Data frame listing all fitted candidate solutions.

`stability_summary` Mean, min, and max pairwise partition agreement.

`selection_summary` Per-k silhouette scores used for selection.

References

Fred, A.L.N. and Jain, A.K. (2002). Data clustering using evidence accumulation. *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, Vol. 4, pp. 276–280.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, **3**, 583–617.

See Also

`cluster()` for single solutions, `validate()` to score the consensus partition, `plot_coassoc()` to visualize the agreement matrix, `plot_consensus()` for a 2-D embedding of the final clusters.

Examples

```
mfit <- metacluster(iris[, 1:4], methods = c("kmeans", "hclust"), k = 2:4,
                    seed = 1)
mfit

# Inspect the co-association matrix
mfit$coassoc_matrix[1:4, 1:4]
```

```
# Score the consensus partition
validate(mfit)$metrics_table
```

`method_used` *Method used*

Description

Return the name of the clustering method used to fit the object.

Usage

```
method_used(x, ...)
```

Arguments

`x`	A `cluster_fit` or `metacluster_fit` object.
`...`	Unused.

Value

Character scalar (e.g. `"kmeans"`, `"pam"`, `"hclust"`).

See Also

`cluster()`

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
method_used(fit)
```

`mixed_distance` *Compute a mixed-type distance matrix*

Description

Build a Gower dissimilarity matrix for mixed-type data using `cluster::daisy()`. This is the recommended first step before hierarchical clustering (`"hclust"` or `"agnes"`) when the input contains categorical variables.

Usage

```
mixed_distance(x, metric = "gower", ...)
```

Arguments

x	Data frame containing numeric and/or categorical columns.
metric	Distance metric passed to cluster::daisy(). Defaults to "gower".
...	Additional arguments passed to cluster::daisy().

Details

For two observations x_i and x_j, the Gower coefficient (Gower, 1971) is

$$d_G(i,j) = \frac{\sum_{f=1}^{p} w_{ijf}\, \delta_{ijf}\, s_{ijf}}{\sum_{f=1}^{p} w_{ijf}\, \delta_{ijf}},$$

where δ_{ijf} indicates whether feature f is comparable (not missing), w_{ijf} is an optional weight, and s_{ijf} is the partial similarity: the scaled absolute difference for numeric features and 0 (match) or 1 (mismatch) for categorical features.

Value

A dist object.

References

Gower, J.C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, **27**(4), 857–874.

See Also

prepare_mixed_data() for converting mixed data to a numeric matrix, cluster() for fitting with the resulting distance object.

Examples

```
d <- mixed_distance(iris)
fit <- cluster(d, method = "hclust", k = 3)
fit
```

---

n_clusters *Number of clusters*

Description

Return the number of clusters in a fitted clustering object. For DBSCAN, this excludes the noise cluster (label 0).

Usage

```
n_clusters(x, ...)
```

Arguments

x	A `cluster_fit` or `metacluster_fit` object.
...	Unused.

Value

Integer scalar.

See Also

`sizes()`, `clusters()`

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
n_clusters(fit)
```

---

phynotype-scales *Discrete color/fill scales matching* `theme_phynotype`

Description

Colorblind-friendly discrete color and fill scales (Okabe-Ito palette, recycled/interpolated beyond 8 levels) for consistent group colors across **phynotype** figures.

Usage

```
scale_color_phynotype(...)

scale_fill_phynotype(...)
```

Arguments

...	Additional arguments passed to `ggplot2::discrete_scale()`.

Value

A **ggplot2** discrete scale object.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
plot_clusters(fit) + scale_color_phynotype()
```

plot_biplot *Plot a cluster biplot*

Description

A self-contained reimplementation of factoextra's fviz_pca_biplot() and fviz_mca_biplot() geometry (individual points/arrows, variable arrows or category points, axis percentage labels, dashed origin lines), producing the same plots without depending on factoextra. Only the "pca" and "mca" embeddings are supported, matching the biplot functions factoextra itself provides; there is no FAMD biplot equivalent, and MDS has no variable space to project.

Usage

```
plot_biplot(
  x,
  data = NULL,
  embedding = c("auto", "pca", "mca"),
  variant = c("cluster", "cos2", "label"),
  top_n = NULL,
  ...
)
```

Arguments

x	A cluster_fit, metacluster_fit, or cluster_explore object.
data	Optional numeric matrix, data frame, or distance object used to compute the embedding. Defaults to the training data stored in x.
embedding	Embedding method. "auto" selects PCA for numeric data and MCA for categorical data. "pca" and "mca" may be selected explicitly.
variant	Display variant: "cluster", "cos2", or "label". See Details.
top_n	Optional integer; if supplied, only the top_n variables (or categories) contributing the most to the plotted axes are drawn.
...	Reserved for future extensions.

Details

Three display variants, matching factoextra's own documented examples (?factoextra::fviz_pca):

"cluster" Individuals colored and shaped by cluster assignment. The default.

"cos2" Individuals colored by their quality of representation (cos2) on a white-blue-orange gradient.

"label" Individuals shown as text labels (observation names or row numbers) instead of points.

For a "pca" embedding, variables are drawn as arrows from the origin, scaled so their spread matches that of the individuals (factoextra's own scaling rule (ratio * 0.7), where ratio is the ratio of the individuals' to variables' coordinate ranges). For a "mca" embedding, variable categories are drawn as unscaled points (matching factoextra::fviz_mca_biplot()'s default arrows = c(FALSE, FALSE)).

Value

A ggplot object.

See Also

plot_clusters(), explore() for the underlying embedding.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
plot_biplot(fit)
plot_biplot(fit, variant = "cos2")
plot_biplot(fit, variant = "label")
```

---

plot_clusters *Plot clustered observations in 2D*

Description

Project observations onto a two-dimensional embedding and color them by cluster assignment. For metacluster_fit objects, delegates to plot_consensus().

Usage

```
plot_clusters(
  x,
  data = NULL,
  embedding = c("auto", "pca", "famd", "mca", "mds"),
  ...
)
```

Arguments

x	A cluster_fit, metacluster_fit, or cluster_explore object.
data	Optional numeric matrix, data frame, or distance object used to compute the embedding. Defaults to the training data stored in x.
embedding	Embedding method. "auto" selects PCA for numeric data, FAMD for mixed numeric/categorical data, MCA for categorical data, and classical MDS for distance inputs. "pca", "famd", "mca", and "mds" may be selected explicitly.
...	Unused.

Value

A ggplot object.

See Also

explore() for the underlying embedding, plot_cluster_sizes(), plot_feature_profiles()

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
plot_clusters(fit)
```

`plot_cluster_sizes` *Plot cluster sizes*

Description

Display a bar chart of the number of observations per cluster.

Usage

```
plot_cluster_sizes(x, ...)
```

Arguments

`x`	A `cluster_fit`, `metacluster_fit`, or `cluster_explore` object.
`...`	Unused.

Value

A `ggplot` object.

See Also

`sizes()`, `plot_clusters()`, `plot_feature_profiles()`

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
plot_cluster_sizes(fit)
```

`plot_coassoc` *Plot the co-association matrix*

Description

Display the $n \times n$ co-association matrix as a heatmap. High values (near 1) indicate pairs of observations that were consistently co-clustered across candidate partitions.

Usage

```
plot_coassoc(x, ...)
```

Arguments

x	A metacluster_fit object.
...	Unused.

Value

A ggplot object.

See Also

metacluster(), plot_consensus()

Examples

```
mfit <- metacluster(iris[, 1:4], methods = c("kmeans", "hclust"),
                    k = 2:3, seed = 1)
plot_coassoc(mfit)
```

---

plot_consensus *Plot consensus clustering in 2D*

Description

Project the consensus cluster assignments onto a two-dimensional embedding.

Usage

```
plot_consensus(
  x,
  data = NULL,
  embedding = c("auto", "pca", "famd", "mca", "mds"),
  ...
)
```

Arguments

x	A metacluster_fit object.
data	Optional numeric matrix, data frame, or distance object used to compute the embedding. Defaults to the training data stored in x.
embedding	Embedding method. "auto" selects PCA for numeric data, FAMD for mixed numeric/categorical data, MCA for categorical data, and classical MDS for distance inputs. "pca", "famd", "mca", and "mds" may be selected explicitly.
...	Unused.

Value

A ggplot object.

See Also

metacluster(), plot_coassoc()

Examples

```
mfit <- metacluster(iris[, 1:4], methods = c("kmeans", "hclust"),
                    k = 2:3, seed = 1)
plot_consensus(mfit)
```

---

plot_dendrogram *Plot a clustering dendrogram*

Description

Draw the dendrogram from a hierarchical `cluster_fit` (fitted via `"hclust"` or `"agnes"`) or the consensus hierarchical tree from a `metacluster_fit`.

Usage

```
plot_dendrogram(x, ...)
```

Arguments

`x`	A hierarchical `cluster_fit` or a `metacluster_fit` object.
`...`	Additional arguments passed to the base `plot()` method for `hclust` objects.

Value

The underlying `hclust` object, invisibly.

See Also

cluster(), metacluster(), plot_consensus()

Examples

```
d <- mixed_distance(iris[, 1:4])
fit <- cluster(d, method = "hclust", k = 3)
plot_dendrogram(fit)

mfit <- metacluster(iris[, 1:4], methods = c("kmeans", "hclust"),
                    k = 2:3, seed = 1)
plot_dendrogram(mfit)
```

plot_feature_profiles *Plot feature profiles by cluster*

Description

Display per-cluster mean values for each feature as a grouped bar chart.

Usage

```
plot_feature_profiles(x, features = NULL, ...)
```

Arguments

x	A cluster_explore object (produced by explore()).
features	Optional character vector of feature names to include. Defaults to all features.
...	Unused.

Value

A ggplot object.

See Also

explore(), plot_clusters(), plot_cluster_sizes()

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
exp <- explore(fit)
plot_feature_profiles(exp)
plot_feature_profiles(exp, features = c("Sepal.Length", "Petal.Length"))
```

plot_silhouette *Plot silhouette widths*

Description

Display observation-level silhouette widths colored by cluster. Observations with negative widths lie closer to a neighboring cluster than to their own and may be misclassified.

Usage

```
plot_silhouette(x, ...)
```

Arguments

x	A cluster_fit object.
...	Unused.

Value

A ggplot object.

See Also

validate()

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
  plot_silhouette(fit)
}
```

---

predict.cluster_fit *Predict cluster assignments for new observations*

Description

Assign new observations to the clusters of a fitted cluster_fit object. The prediction strategy is method-dependent: k-means, PAM, and hierarchical methods use nearest-centroid or nearest-prototype rules; GMM uses the posterior mode; DBSCAN uses a density-based rule falling back to the nearest core point.

Usage

```
## S3 method for class 'cluster_fit'
predict(object, new_data, ...)
```

Arguments

object	A cluster_fit object. Must have been fitted on row-by-feature data (not a dist-only input).
new_data	Row-by-feature data compatible with the training data. Numeric methods expect a numeric matrix or data frame; "kproto" and "kmm" accept mixed-type data frames with the same column names used during training.
...	Additional arguments passed to method-specific predictors.

Value

A `cluster_prediction` object with components:

`clusters` Integer vector of predicted cluster assignments.

`distances` Distance matrix to each cluster center/prototype, or `NULL` for methods that do not produce distances.

`membership` Soft membership matrix (GMM only), or `NULL`.

See Also

`cluster()` to fit, `feature_importance()`, `lime_explain()`, and `ceteris_paribus()` which all use `predict()` internally.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)

# Predict on the training data
pred <- predict(fit, iris[1:10, 1:4])
pred$clusters

# Predict on new data
new_obs <- data.frame(
  Sepal.Length = c(5.0, 7.0),
  Sepal.Width  = c(3.5, 3.0),
  Petal.Length = c(1.5, 5.0),
  Petal.Width  = c(0.3, 1.8)
)
predict(fit, new_obs)$clusters
```

---

prepare_mixed_data *Prepare mixed-type data for numeric clustering methods*

Description

Convert a data frame containing numeric, logical, character, or factor columns into a numeric design matrix suitable for methods such as `"kmeans"`, `"pam"`, `"dbscan"`, and `"gmm"`.

Usage

```
prepare_mixed_data(
  x,
  center = TRUE,
  scale = TRUE,
  drop_first = FALSE,
  na_action = c("keep", "fail")
)
```

Arguments

x	Data frame, matrix, or vector-like object coercible to a data frame.
center	Logical; if TRUE, center numeric columns to zero mean after encoding.
scale	Logical; if TRUE, scale numeric columns to unit variance after encoding.
drop_first	Logical; if TRUE, drop the first dummy column for each categorical feature to avoid the dummy-variable trap. The default (FALSE) keeps the full one-hot representation, which is usually preferable for distance-based clustering.
na_action	Missing-value strategy. "keep" (default) adds an explicit "missing" level for categoricals and median-imputes numeric columns. "fail" stops with an error on any missing value.

Details

Categorical variables are one-hot encoded (full-rank by default), numeric variables are optionally centered and scaled, and missingness can be preserved as an explicit level or trigger an error.

This is intentionally a manual first step. Clustering algorithms that use Euclidean distances or Gaussian models should not silently coerce mixed data. For hierarchical methods on mixed data, use mixed_distance() instead.

Value

A numeric matrix with preprocessing metadata stored in the attribute "phynotype_preprocess".

See Also

mixed_distance() for Gower-distance computation before hierarchical clustering.

Examples

```
# Mixed data frame
df <- data.frame(
  x1 = c(1.2, 0.5, 3.1),
  x2 = factor(c("A", "B", "A"))
)
mat <- prepare_mixed_data(df)
mat
```

---

prototypes *Cluster prototypes*

Description

Extract the cluster prototype table. Prototypes are actual observations (PAM medoids) or mixed-type representatives (k-prototypes, KMM); returns NULL for methods that use centroids rather than prototypes.

Usage

```
prototypes(x, ...)
```

Arguments

x	A cluster_fit or metacluster_fit object.
...	Unused.

Value

A matrix or data frame with one row per cluster, or NULL.

See Also

centers(), clusters()

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  fit <- cluster(iris[, 1:4], method = "pam", k = 3)
  prototypes(fit)
}
```

---

sizes *Cluster sizes*

Description

Return the number of observations assigned to each cluster.

Usage

```
sizes(x, ...)
```

Arguments

x	A cluster_fit or metacluster_fit object.
...	Unused.

Value

Named integer vector, where names are the cluster labels.

See Also

clusters(), n_clusters()

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
sizes(fit)
```

---

theme_phynotype *phynotype ggplot2 theme*

Description

The single ggplot2 theme used consistently across every phynotype plot function (`plot_clusters()`, `plot_cluster_sizes()`, `plot_feature_profiles()`, `plot_consensus()`, `plot_coassoc()`, `plot_biplot()`, `plot()` methods for `feature_importance`, `ceteris_paribus`, and `lime_explanation` objects, and `plot_validation()`). Built on `ggplot2::theme_classic()` with an Okabe-Ito colorblind-safe palette, matching the style used by the `funcml` package's `theme_funcml()`.

Usage

```
theme_phynotype(base_size = 11)
```

Arguments

`base_size`	Base font size.

Value

A ggplot2 theme object.

See Also

`scale_color_phynotype()`, `scale_fill_phynotype()`

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)
plot_clusters(fit) + theme_phynotype()
```

`validate` *Validate clustering results*

Description

Compute internal and external validation metrics for a fitted clustering object or over a grid of candidate cluster counts.

Usage

```
validate(
  x,
  ...,
  method = "kmeans",
  k = NULL,
  truth = NULL,
  metrics = NULL,
  n_boot = 10
)
```

Arguments

`x`	A `cluster_fit`, `metacluster_fit`, or numeric matrix/data frame. When a raw data matrix is supplied, `validate()` fits and scores solutions for each value in `k` using `method`.
`...`	Additional arguments passed to methods.
`method`	Clustering method used when `x` is raw data.
`k`	Candidate values of `k` for direct grid validation.
`truth`	Optional integer or factor vector of reference labels. When supplied, the adjusted Rand index and normalized mutual information are appended to the metric table.
`metrics`	Optional character vector selecting a subset of metrics to return (e.g. `c("silhouette", "calinski_harabasz")`).
`n_boot`	Number of bootstrap resamples for bootstrap ARI stability (only for `"kmeans"`, `"pam"`, and `"gmm"` fits with a fixed `k`).

Details

Internal metrics:

Silhouette width (Rousseeuw, 1987):

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i),\, b(i)\}},$$

where $a(i)$ is the mean intra-cluster distance and $b(i)$ is the minimum mean distance to any other cluster. Values near 1 indicate dense, well-separated clusters.

Calinski-Harabasz index (Calinski and Harabasz, 1974):

$$\mathrm{CH} = \frac{\mathrm{BSS}/(k-1)}{\mathrm{WSS}/(n-k)},$$

where BSS is the between-cluster sum of squares and WSS is the within-cluster sum of squares. Higher values indicate better separation.

Davies-Bouldin index (Davies and Bouldin, 1979):

$$\mathrm{DB} = \frac{1}{k}\sum_{j=1}^{k} \max_{l \neq j} \frac{s_j + s_l}{d(\mu_j, \mu_l)},$$

where s_j is the mean intra-cluster scatter. Lower values are better.

Total within-cluster sum of squares:

$$\mathrm{WSS} = \sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2.$$

Bootstrap ARI (Fang and Wang, 2012): mean adjusted Rand index between the reference partition and partitions fitted on bootstrap resamples.

External metrics:

Adjusted Rand index (Hubert and Arabie, 1985):

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - E}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - E},$$

where E is the expected index under random partitions. Values near 1 indicate agreement close to the reference.

Normalized mutual information (Strehl and Ghosh, 2002):

$$\mathrm{NMI}(U, V) = \frac{I(U;V)}{\sqrt{H(U)\,H(V)}}.$$

Value

A `cluster_validation` object with components:

`metrics_table` Data frame with columns `metric`, `value`, `scale`, `direction`.

`per_cluster_table` Per-cluster mean silhouette widths, or `NULL`.

References

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.

Calinski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**(1), 1–27.

Davies, D.L. and Bouldin, D.W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1**(2), 224–227.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**(1), 193–218.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, **3**, 583–617.

Fang, Y. and Wang, J. (2012). Selection of the number of clusters via the bootstrap method. *Computational Statistics and Data Analysis*, **56**(3), 468–477.

See Also

`cluster()` to fit a solution, `plot_silhouette()` to visualize per-observation silhouette widths, `explore()` for structural summaries.

Examples

```
fit <- cluster(iris[, 1:4], method = "kmeans", k = 3, seed = 1)

# Internal metrics
val <- validate(fit)
val$metrics_table

# External metrics with known labels
val_ext <- validate(fit, truth = iris$Species)
val_ext$metrics_table

# Grid search over k values
grid_val <- validate(iris[, 1:4], method = "kmeans", k = 2:5)
grid_val$metrics_table
```

Index