Package 'MergeKmeans'

September 4, 2026

Type Package

Title Clustering Large Datasets by Merging K-Means Solutions

Version 0.3.0

Description Fast clustering of large datasets by hierarchically merging components of a K-means solution based on the pairwise overlap between the Gaussian mixture components implied by the K-means partition, as proposed by Melnykov and Michael (2020) <doi:10.1007/s00357-019-09314-8>. Implements the DEMP-K merging algorithm with single, Ward's, average, and complete linkages, the overlap map display for selecting the number of clusters, four K-means variants corresponding to Gaussian mixtures with spherical or elliptical, homoscedastic or heteroscedastic components, and a tool for selecting the number of K-means components.

License GPL (>= 2)

Encoding UTF-8

Imports stats, graphics, grDevices, utils, parallel

Suggests CompQuadForm, MixSim, mclust, broom, tibble, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Aqi Dong [aut, cre] (ORCID: <https://orcid.org/0009-0004-1676-3528>),
Volodymyr Melnykov [aut],
Yang-Li Liao [aut],
Peng Li [aut],
Xuejian Li [aut]

Maintainer Aqi Dong <donga2@erau.edu>

Repository CRAN

Date/Publication 2026-09-04 15:50:13 UTC

Contents

`chooseK` *Choose the number of K-means components*

Description

Computes the proportion of the total variability explained by the between-cluster variability (SSB/SSTO) of K-means solutions over a grid of `K` values, plots it, and reports a recommended `K` for `MergeKmeans()`.

Usage

```
chooseK(
  x,
  K = seq(5, 50, by = 5),
  nstart = 10,
  iter.max = 100,
  plot = TRUE,
  cores = 1,
  ...
)
```

Arguments

`x`	numeric data matrix or data frame with observations in rows.
`K`	integer vector of candidate numbers of components.
`nstart`	number of K-means restarts per candidate.
`iter.max`	maximum number of K-means iterations.
`plot`	logical; plot SSB/SSTO against `K` with the recommendation marked.
`cores`	number of CPU cores used to evaluate the grid in parallel (POSIX systems only).
`...`	further arguments passed to `stats::kmeans()`.

Details

Following Melnykov and Michael (2020), a suitable number of components is found at or somewhat *beyond* the elbow of the curve, the point at which the increase in the explained proportion becomes marginal. Over-estimating `K` is safe (the merging procedure is robust to it), but `K` must exceed the number of clusters sought: setting `K` equal to the anticipated cluster count defeats the merging step.

The elbow is located by the chord method: after rescaling both axes to the unit interval, it is the grid point farthest above the straight line joining the first and the last point of the curve. The recommendation is the next grid value, or the last grid value when the elbow lies at the end of the grid. The location depends on the range of the grid (a grid that starts beyond the true elbow cannot find it), so the grid should begin at a small value and extend well past the number of components one would consider; the default `seq(5, 50, by = 5)` is a reasonable starting point for most datasets.

Value

Invisibly, a data frame with columns `K` and `ssb.ssto`, with the recommended `K` in `attr(, "recommended")`.

References

Melnykov, V. and Michael, S. (2020) Clustering large datasets by merging K-means solutions. *Journal of Classification*, 37, 97-123. doi:10.1007/s00357019093148

See Also

`MergeKmeans()`

Examples

```
set.seed(5)
x <- rbind(matrix(rnorm(400), ncol = 2),
           matrix(rnorm(400, mean = 5), ncol = 2))
out <- chooseK(x, K = 2:12, nstart = 5)
attr(out, "recommended")
```

`MergeKmeans` *Clustering by merging K-means solutions (DEMP-K)*

Description

Fits the DEMP-K clustering procedure of Melnykov and Michael (2020). A K-means solution with `K` components (chosen larger than the anticipated number of clusters) is interpreted as a Gaussian mixture fitted by the classification EM algorithm; the pairwise overlap between all component pairs is computed in closed form; and components are merged by hierarchical clustering with `1 - overlap` as the dissimilarity. The merge tree is cut at a requested number of clusters `G`, at an overlap threshold `omega.star`, or automatically (`G = "auto"`).

Usage

```
MergeKmeans(
  x,
  K = NULL,
  G = NULL,
  omega.star = NULL,
  variant = c("HoSC", "HoEC", "HeSC", "HeEC"),
  linkage = NULL,
  nstart = 10,
  iter.max = 100,
  cem.iter.max = 50,
  init = c("kmeans++", "random"),
  start = NULL,
  cores = 1,
  keep.data = TRUE,
  ...
)
```

Arguments

x	numeric data matrix or data frame with observations in rows.
K	number of K-means components; choose it comfortably *larger* than the number of clusters G you expect, so that small spherical components can fill clusters of arbitrary shape (this is the opposite of the centers argument of stats::kmeans(), which is the final cluster count). If NULL (default), K is selected automatically with chooseK() and reported in a message.
G	requested number of clusters after merging, between 1 and K, or "auto" to cut at the largest gap in the single-linkage merge sequence of the overlap matrix, measured on the scale log(overlap + 1e-4) so that negligible overlaps cannot create a gap. The rule targets well-separated clusters and is not accepted with Ward's linkage. It can only return values in 2:(K-1), cannot detect the single-cluster case (use recut(fit, G = 1) explicitly), and on overlapping or structureless clusters it has no gap to find, so its value is arbitrary - always confirm with the overlap map.
omega.star	overlap threshold in (0, 1): merging stops when no pair of merged groups attains this overlap. Supply at most one of G and omega.star. Not meaningful with Ward's linkage.
variant	K-means variant (case-insensitive; aliases in Details).
linkage	linkage for the hierarchical merging, passed to stats::hclust(): one of "ward.D", "single", "average", "complete". The default (NULL) uses "ward.D", except for univariate data where "single" is used; see Details.
nstart	number of K-means restarts.
iter.max	maximum number of K-means iterations.
cem.iter.max	maximum number of classification EM iterations for the non-spherical variants.
init	initialization of the K-means restarts: "kmeans++" (default; Arthur and Vassilvitskii, 2007) or "random" (the stats::kmeans() default seeding).

start	optional precomputed component solution to be merged instead of running K-means, given either as labels - an object of class "kmeans", a list with a cluster (or ClusterR-style clusters) element, or a bare integer vector - or as centers - a K x p matrix, or a list with a centers (or ClusterR-style centroids) element, such as the value of ClusterR::MiniBatchKmeans(). Centers are converted to labels by nearest-center assignment, which makes the partition consistent with predict.MergeKmeans() by construction; supplied labels are kept as is and are reproduced by predict() only if they are themselves nearest-center consistent. K, init, and nstart are ignored.
cores	number of CPU cores for the K-means restarts (forked via **parallel**; POSIX systems only, with a message and serial execution on Windows). Every restart is seeded from the calling RNG stream, so a fit is reproducible from set.seed() for any value of cores.
keep.data	logical; store the data in the returned object (needed only for plot(..., what = "classification") without a data argument). Set to FALSE for very large datasets.
...	further arguments passed to stats::kmeans().

Details

Four variants of K-means are supported, each corresponding to a Gaussian mixture with specific restrictions (Melnykov and Michael, 2020, Section 3 and Appendix A):

- "HoSC" (default; alias "spherical"): homoscedastic spherical components - the traditional K-means algorithm based on Euclidean distances. The fastest variant and the one recommended for large datasets.
- "HoEC" (alias "elliptical"): homoscedastic elliptical components - K-means with Mahalanobis distances and a common covariance matrix.
- "HeSC" (alias "spherical-unequal"): heteroscedastic spherical components - per-component spherical variances.
- "HeEC" (alias "elliptical-unequal"): heteroscedastic elliptical components - unrestricted per-component covariance matrices (requires the **CompQuadForm** package for the overlap computation).

Variant names are matched case-insensitively. The non-spherical variants are initialized from the K-means partition and refined by classification EM iterations.

The choice of linkage should be driven by the cluster characteristics sought: single linkage is appropriate for detecting well-separated clusters of arbitrary shape, while Ward's linkage ("ward.D", the default for p >= 2) targets compact clusters. Two situations where Ward's linkage is unreliable and single linkage should be preferred: univariate data (the components form a chain in which only adjacent pairs overlap, so the default there is "single"), and severely unequal cluster sizes (the homoscedastic overlap does not weight components by mass, and Ward merges can then cross a genuine gap). When clusters are genuinely overlapping *Gaussian* ellipsoids, classical model-based clustering (e.g. **mclust**) remains the better tool; DEMP-K is at its best for non-Gaussian or arbitrarily shaped clusters and for sample sizes where model-based clustering is impractical.

Robustness limits worth knowing (established by simulation): single linkage can chain two clusters together through even small amounts of uniform background contamination – the breakdown level

depends on the geometry, with failures observed between 1\ Ward's linkage was stable to at least 10\ heavy-tailed clusters (e.g. multivariate t with 1-2 degrees of freedom, or strong exponential tails) create bridge points that defeat overlap-based merging - "skewed" clusters are handled well, but "heavy-tailed" ones are not. The heteroscedastic variants (`"HeSC"`, `"HeEC"`) estimate their variance parameters by the closed-form maximizer of the classification likelihood under the implied mixing proportions; components whose scale cannot be estimated (fewer than two observations, or `p + 1` for `"HeEC"`) are dropped with a warning, and the fit warns when `cem.iter.max` is reached without convergence. `"HoSC"` remains the recommended default for large data.

Like K-means itself, the component solution depends on the scaling of the variables; consider standardizing (e.g. `scale(x)`) when variables are measured on very different scales. The merging step is fairly robust to scaling provided `K` is large enough.

For very large datasets the K-means step is the only material cost (the merging operates on the `K x K` overlap matrix). Two escape hatches are provided: `cores` forks the restarts on POSIX systems, and `start` accepts the output of any external component-stage engine - the centroids returned by `ClusterR::MiniBatchKmeans()`, a `biganalytics::bigkmeans()` fit, or a previous `stats::kmeans()` run - so that `MergeKmeans()` performs only the variant-specific parameter estimation and the merging. The data are still needed for that estimation: `start` removes the runtime ceiling of a single-threaded K-means run, not the memory requirement of holding `x`. The default Hartigan-Wong algorithm may emit "Quick-TRANSfer" warnings on some starts for very large n; results are unaffected, but `algorithm = "MacQueen"` can be passed through `...` to silence them.

If neither `G` nor `omega.star` is supplied, the fit is returned uncut and a message reports the number of clusters suggested by the largest gap in the merge heights: inspect the overlap map with `plot(fit, what = "overlap")` and then call `recut()`. `G = "auto"` accepts that suggestion directly. The threshold `omega.star` is interpretable as a minimum pairwise overlap for merging only with single, average, or complete linkage, and is therefore not accepted with Ward's linkage.

Value

An object of class `"MergeKmeans"`: a list with components

`cluster` integer vector of final cluster labels (`NULL` if the tree has not been cut).

`G` number of clusters after merging.

`components` integer vector of K-means component labels.

`component.map` integer vector mapping each component to its merged cluster.

`K` effective number of components.

`centers` `K x p` matrix of component means.

`cov` estimated covariance parameter(s); form depends on the variant.

`tau` mixing proportions implied by the variant.

`overlap` `K x K` matrix of pairwise overlaps.

`tree` the merge tree, an stats::hclust object.

`variant`, `linkage` the settings used.

`kmeans` the underlying `stats::kmeans()` fit (`NULL` when `start` was supplied).

`cem` iteration details for non-spherical variants, or `NULL`.

`data` the data matrix if `keep.data = TRUE`, else `NULL`.

References

Melnykov, V. and Michael, S. (2020) Clustering large datasets by merging K-means solutions. *Journal of Classification*, 37, 97-123. doi:10.1007/s00357019093148

Arthur, D. and Vassilvitskii, S. (2007) k-means++: the advantages of careful seeding. *Proceedings of SODA 2007*, 1027-1035.

See Also

`recut()`, `chooseK()`, `overlap_map()`, `pairwise_overlap()`, `predict.MergeKmeans()`

Examples

```
## two well-separated half-circular clusters
set.seed(7)
th <- runif(500, 0, pi)
x <- rbind(
  cbind(cos(th[1:250]), sin(th[1:250])) + matrix(rnorm(500, 0, 0.1), 250),
  cbind(1 - cos(th[251:500]), 0.4 - sin(th[251:500])) +
    matrix(rnorm(500, 0, 0.1), 250))
fit <- MergeKmeans(x, K = 10, G = 2, linkage = "single", nstart = 20)
table(fit$cluster, rep(1:2, each = 250))
plot(fit, what = "overlap")

## automatic G, and merging a partition computed elsewhere
fit2 <- MergeKmeans(x, K = 10, G = "auto", linkage = "single")
km <- kmeans(x, 10, nstart = 20)
fit3 <- MergeKmeans(x, start = km, G = 2, linkage = "single")
```

`MergeKmeans-tidiers` *Broom tidiers for MergeKmeans fits*

Description

Methods for the **broom** generics, registered when **broom** is loaded: `tidy()` returns one row per K-means component (its merged cluster, size, and center coordinates), `glance()` returns a one-row model summary, and `augment()` appends `.component` and `.cluster` columns to the data.

Usage

```
## S3 method for class 'MergeKmeans'
tidy(x, ...)

## S3 method for class 'MergeKmeans'
glance(x, ...)

## S3 method for class 'MergeKmeans'
augment(x, data = NULL, ...)
```

Arguments

x	a "MergeKmeans" object.
...	ignored.
data	the original data, for augment(); defaults to the data stored in the fit.

Value

A data frame (a tibble when available): per-component rows for tidy(), a single row for glance(), and data plus .component and .cluster columns for augment().

Examples

```
set.seed(6)
x <- rbind(matrix(rnorm(200), ncol = 2),
           matrix(rnorm(200, mean = 6), ncol = 2))
fit <- MergeKmeans(x, K = 6, G = 2, linkage = "single", nstart = 5)
if (requireNamespace("broom", quietly = TRUE)) {
  broom::tidy(fit)
  broom::glance(fit)
  head(broom::augment(fit, x))
}
```

---

overlap_map *Overlap map display*

Description

Draws the overlap map of Melnykov and Michael (2020, Section 3.3): a lower-triangular heat map of all pairwise component overlaps, with components arranged so that strongly overlapping components are adjacent. Cells range from pale yellow (low overlap) to dark red (high overlap). The detached bottom row summarizes each column by its darkest hue: groups of adjacent dark cells indicate components that model a common cluster, while pale cells mark gaps between well-separated clusters, making the map an intuitive device for choosing the number of clusters.

Usage

```
overlap_map(
  omega,
  order = TRUE,
  zlim = NULL,
  col = NULL,
  legend = TRUE,
  main = "Overlap map",
  cex.axis = 0.8
)
```

Arguments

omega	a symmetric matrix of pairwise overlaps, typically the overlap entry of a MergeKmeans() fit.
order	TRUE (default) to order components by the algorithm described above, FALSE to keep the original order, or an integer permutation of 1:K.
zlim	numeric range mapped onto the color scale; defaults to the range of the off-diagonal overlaps.
col	vector of colors for the scale; defaults to a yellow-to-red ramp.
legend	logical; draw the color legend.
main	plot title.
cex.axis	expansion factor for the component labels.

Details

Components are ordered as in the paper: the first two are the pair with the highest overlap, and each subsequent component is the one with the highest overlap with any component already displayed.

Value

Invisibly, the integer ordering of the components used in the display.

References

Melnykov, V. and Michael, S. (2020) Clustering large datasets by merging K-means solutions. *Journal of Classification*, 37, 97-123. doi:10.1007/s00357019093148

See Also

MergeKmeans(), plot.MergeKmeans()

Examples

```
set.seed(2)
x <- rbind(matrix(rnorm(300, 0, 1), ncol = 2),
           matrix(rnorm(300, 7, 1), ncol = 2))
fit <- MergeKmeans(x, K = 8, linkage = "single")
overlap_map(fit$overlap)
```

`pairwise_overlap` *Pairwise overlap between Gaussian components*

Description

Computes the matrix of pairwise overlaps between Gaussian mixture components, where the overlap between components k and k' is defined as the sum of the two misclassification probabilities $\omega_{kk'} = \omega_{k'|k} + \omega_{k|k'}$ (Maitra and Melnykov, 2010). The covariance structure is inferred from the form of `cov`:

Usage

```
pairwise_overlap(centers, cov, tau = NULL, lim = 50000, acc = 1e-06)
```

Arguments

`centers`	numeric matrix of component means with one row per component (`K x p`).
`cov`	covariance specification; see Details.
`tau`	optional vector of `K` mixing proportions; see Details for the defaults.
`lim`, `acc`	accuracy settings passed to `CompQuadForm::davies()` for the heteroscedastic elliptical case.

Details

- a single number: homoscedastic spherical components $\Sigma_k = \sigma^2 I$ ("HoSC"),
- a `p x p` matrix: homoscedastic elliptical components $\Sigma_k = \Sigma$ ("HoEC"),
- a vector of length `K`: heteroscedastic spherical components $\Sigma_k = \sigma_k^2 I$ ("HeSC"),
- a `p x p x K` array: heteroscedastic elliptical components ("HeEC").

Mixing proportions `tau` may be supplied in every case, so that the overlap of an arbitrary fitted Gaussian mixture can be computed. By default the homoscedastic cases use equal proportions and the heteroscedastic cases the proportions implied by the corresponding K-means variant, $\tau_k \propto \sigma_k^p$ for "HeSC" and $\tau_k \propto |\Sigma_k|^{1/2}$ for "HeEC", under which the log-ratio term in the misclassification probability vanishes (Melnykov and Michael, 2020, Appendix A). The "HeEC" case requires the **CompQuadForm** package, which evaluates the distribution of linear combinations of non-central chi-squared variables (Davies, 1980).

Value

A symmetric `K x K` matrix of pairwise overlaps with zero diagonal.

References

Melnykov, V. and Michael, S. (2020) Clustering large datasets by merging K-means solutions. *Journal of Classification*, 37, 97-123. doi:10.1007/s00357019093148

Maitra, R. and Melnykov, V. (2010) Simulating data to study performance of finite mixture modeling and clustering algorithms. *Journal of Computational and Graphical Statistics*, 19(2), 354-376.

Davies, R. B. (1980) The distribution of a linear combination of chi-squared random variables. *Applied Statistics*, 29, 323-333.

See Also

`MergeKmeans()`, `overlap_map()`

Examples

```
centers <- rbind(c(0, 0), c(3, 0), c(10, 10))
pairwise_overlap(centers, cov = 1)              # spherical, sigma^2 = 1
pairwise_overlap(centers, cov = c(1, 4, 1))     # heteroscedastic spherical
```

---

plot.MergeKmeans *Plot method for MergeKmeans fits*

Description

Plot method for MergeKmeans fits

Usage

```
## S3 method for class 'MergeKmeans'
plot(x, what = c("overlap", "tree", "classification"), data = NULL, ...)
```

Arguments

`x`	a `"MergeKmeans"` object.
`what`	type of display: `"overlap"` for the overlap map (see `overlap_map()`), `"tree"` for the dendrogram of the hierarchical merging, or `"classification"` for a scatter plot of the data colored by the merged clusters (the first two principal components are shown when the data have more than two columns).
`data`	data matrix for `what = "classification"`; defaults to the data stored in the fit.
`...`	further arguments passed to `overlap_map()`, to plot.hclust(), or to plot().

Value

For `what = "overlap"`, invisibly the component ordering; otherwise `NULL`, invisibly.

See Also

MergeKmeans(), overlap_map()

Examples

```
set.seed(3)
x <- rbind(matrix(rnorm(300), ncol = 2),
           matrix(rnorm(300, mean = 6), ncol = 2))
fit <- MergeKmeans(x, K = 8, G = 2, linkage = "single")
plot(fit, what = "overlap")
plot(fit, what = "tree")
plot(fit, what = "classification")
```

---

predict.MergeKmeans *Predict cluster membership for new observations*

Description

Assigns new observations to K-means components by the decision rule of the fitted variant (nearest center in the appropriate metric) and then maps components to merged clusters.

Usage

```
## S3 method for class 'MergeKmeans'
predict(object, newdata, ...)
```

Arguments

object	a "MergeKmeans" object.
newdata	numeric matrix or data frame of new observations with the same variables as the training data.
...	ignored.

Details

On the training data, the predicted components reproduce the fitted components exactly for K-means-based fits and for converged classification EM fits. The round-trip is not guaranteed when a user-supplied start partition was not nearest-center consistent or when the CEM iterations did not converge.

Value

A list with entries component (assigned K-means components) and cluster (merged cluster labels, or NULL if the merge tree has not been cut).

See Also

MergeKmeans()

Examples

```
set.seed(4)
x <- rbind(matrix(rnorm(300), ncol = 2),
           matrix(rnorm(300, mean = 6), ncol = 2))
fit <- MergeKmeans(x, K = 8, G = 2, linkage = "single")
predict(fit, rbind(c(0, 0), c(6, 6)))
```

---

recut *Re-cut the merge tree of a MergeKmeans fit*

Description

Produces a new partition from a fitted MergeKmeans() object by cutting its merge tree at a different number of clusters or overlap threshold, without re-running K-means or the overlap computation.

Usage

```
recut(object, G = NULL, omega.star = NULL)
```

Arguments

object	a "MergeKmeans" object.
G	requested number of clusters, or "auto" to cut at the largest gap in the single-linkage merge sequence of the overlap matrix, as described in MergeKmeans(); not available with Ward's linkage.
omega.star	overlap threshold in (0, 1); meaningful with single, average, or complete linkage, where the merge heights are 1 - overlap (not accepted with Ward's linkage). Supply exactly one of G and omega.star.

Value

The input object with updated cluster, G, and component.map entries.

See Also

MergeKmeans()

Examples

```
set.seed(1)
x <- rbind(matrix(rnorm(200), ncol = 2),
           matrix(rnorm(200, mean = 6), ncol = 2))
fit <- MergeKmeans(x, K = 8, G = 2, linkage = "single")
recut(fit, G = "auto")
```

`summary.MergeKmeans` *Summary method for MergeKmeans fits*

Description

Summary method for MergeKmeans fits

Usage

```
## S3 method for class 'MergeKmeans'
summary(object, ...)
```

Arguments

`object`	a `"MergeKmeans"` object.
`...`	ignored.

Value

`object`, invisibly.

Index