

Package ‘gmeans’

October 24, 2026

Title G-means Clustering

Version 0.3.0

Description Gaussian-means (G-means) clustering is a clustering algorithm that extends the k-means algorithm by automatically determining the number of clusters.

License MIT + file LICENSE

URL <https://m-muecke.github.io/gmeans/>,
<https://github.com/m-muecke/gmeans>

BugReports <https://github.com/m-muecke/gmeans/issues>

Depends R (>= 4.1.0)

Imports stats

Suggests broom, clue, data.table (>= 1.17.0), generics, ggplot2,
knitr, rmarkdown, testthat (>= 3.3.0), withr (>= 2.4.0)

VignetteBuilder knitr

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Author Maximilian Mücke [aut, cre] (ORCID:
<<https://orcid.org/0009-0000-9432-9795>>)

Maintainer Maximilian Mücke <muecke.maximilian@gmail.com>

Repository CRAN

Date/Publication 2026-10-24 15:50:07 UTC

Contents

ad.test	2
compute_wss	3
gmeans	4
gmeans_tidiers	6
predict.gmeans	7
summary.gmeans	9

Index	10
--------------	-----------

ad.test	<i>Anderson-Darling Normality Test</i>
---------	--

Description

Perform the Anderson-Darling normality test.

Usage

```
ad.test(x)
```

Arguments

x (numeric())
Vector of data values. Missing values are allowed, but the number of non-missing values must be greater than 7.

Details

The Anderson-Darling test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^n (2i-1) [\ln(z_i) + \ln(1-z_{n+1-i})]$$

where $z_i = \Phi\left(\frac{x_i - \bar{x}}{s}\right)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\bar{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the modified statistic $A_*^2 = A^2(1.0 + 0.75/n + 2.25/n^2)$ according to Table 4.9 in Stephens (1986).

Value

A list inheriting from classes "htest" containing the following components:

- **statistic**: the value of the statistic.
- **p.value**: the p-value of the test.
- **method**: the character string "Anderson-Darling normality test".
- **data.name**: a character string giving the name(s) of the data.

Source

Adapted from `nortest::ad.test()`

References

Stephens, A. M (1986). “Goodness-of-Fit-Techniques.” In D’Agostino, B. R (eds.), chapter Tests based on EDF statistics. CRC Press.

Thode, C. H (2002). *Testing for normality*, 1 edition. CRC Press. doi:10.1201/9780203910894.

See Also

`stats::shapiro.test()` for performing the Shapiro-Wilk test for normality. `nortest::cvm.test()`, `nortest::lillie.test()`, `nortest::pearson.test()`, `nortest::sf.test()` for performing further tests for normality. `stats::qqnorm()` for producing a normal quantile-quantile plot.

Examples

```
set.seed(123)
ad.test(rnorm(100, mean = 5, sd = 3))
ad.test(runif(100, min = 2, max = 4))
```

compute_wss

Compute Within-Cluster Sum of Squares

Description

Compute Within-Cluster Sum of Squares

Usage

```
compute_wss(object, newdata = NULL)
```

Arguments

object	(any) Class inheriting from "kmeans".
newdata	(matrix()) New data to predict on, a numeric matrix or a data frame. Columns are matched to the centers by name and unused columns are ignored.

Details

WSS is defined as

$$\sum_{i=1}^n \|x_i - \mu_{j(i)}\|^2,$$

where x_i is a data point and $\mu_{j(i)}$ is the centroid of the cluster to which x_i is assigned. When new data is provided, the function predicts the nearest cluster for each new observation and computes the WSS for these points based on their predicted clusters.

Value

A `numeric()` vector with one within-cluster sum of squares per cluster, in the order of the rows of `object$centers`. Clusters with no assigned points contribute 0.

Examples

```
km <- kmeans(mtcars, 5)
compute_wss(km)
# or with new data
compute_wss(km, mtcars)
```

gmeans

*G-means Clustering***Description**

Perform G-means clustering on a data matrix.

Usage

```
gmeans(x, k_init = 1L, k_max = Inf, level = 1e-04, iter.max = 100L, ...)
```

Arguments

<code>x</code>	(<code>matrix()</code>) Numeric matrix of data, or a data frame with all numeric columns. Logical input is coerced to a 0/1 matrix. Missing and infinite values are not allowed and the matrix must have at least one row and one column.
<code>k_init</code>	(<code>integer(1)</code>) Initial amount of centers. Default is 1L.
<code>k_max</code>	(<code>integer(1)</code>) Maximum amount of centers. Must be greater than or equal to <code>k_init</code> . Default is Inf.
<code>level</code>	(<code>numeric(1)</code>) Significance level for the Anderson-Darling test. Default is 0.0001. A larger level such as 0.01 can work better for clusters with fewer than about 50 points. See ad.test() for more information.
<code>iter.max</code>	(<code>integer(1)</code>) Maximum number of k-means iterations, passed to stats::kmeans() . Default is 100L, higher than the 10L of stats::kmeans() , so that k-means converges when there are many clusters.
<code>...</code>	(any) Additional arguments passed to stats::kmeans() , except centers, which is set by <code>k_init</code> . <code>nstart</code> has no effect since the initial centers are always given as a matrix.

Details

The G-means clustering algorithm is an extension of the traditional k-means algorithm that automatically determines the number of clusters by iteratively testing the Gaussianity of data within clusters. The process begins with a specified initial number of clusters (`k_init`) and iteratively increases the number of clusters until it reaches the specified maximum (`k_max`) or the data within clusters is determined to be Gaussian at the specified significance level (`level`).

The algorithm is outlined as follows:

1. Let C be the initial set of centers (usually $C \leftarrow \{\bar{x}\}$).
2. Perform k-means clustering on the dataset X using the current set of centers C , i.e., $C \leftarrow \text{kmeans}(C, X)$.
3. For each center c_j , identify the set of data points $\{x_i \mid \text{class}(x_i) = j\}$ that are assigned to c_j .
4. Use the Anderson-Darling test to check if the set of data points $\{x_i \mid \text{class}(x_i) = j\}$ follows a Gaussian distribution at the significance level α .
5. If the data points appear Gaussian, keep c_j . Otherwise, replace c_j with two new centers, found by k-means on the cluster started from $c_j \pm s\sqrt{2\lambda/\pi}$, where s is the main principal component of the cluster and λ its eigenvalue.
6. Repeat from step 2 until no more centers are added.

Value

An object of class `c("gmeans", "kmeans")` with the components of a `stats::kmeans()` object plus `k_init`, `k_max`, and `level`, the settings used to fit the model. See [gmeans_tidiers](#) for summarizing the result as data frames.

References

Hamerly, Greg, Elkan, Charles (2003). "Learning the k in k-means." In Thrun S, Saul L, Schölkopf B (eds.), *Advances in Neural Information Processing Systems*, volume 16. https://proceedings.neurips.cc/paper_files/paper/2003/file/234833147b97bb6aed53a8f4f1c7a7d8-Paper.pdf.

Examples

```
set.seed(123)
x <- rbind(
  matrix(rnorm(100, sd = 0.3), ncol = 2),
  matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2)
)
colnames(x) <- c("x", "y")
cl <- gmeans(x)
```

gmeans_tidiers

*Tidy a G-means Clustering Object***Description**

Summarize a G-means clustering model as data frames, following the conventions of the broom package. The generics come from the generics package, so load it or broom to call the methods.

Usage

```
## S3 method for class 'gmeans'
tidy(x, col.names = colnames(x$centers), ...)
```

```
## S3 method for class 'gmeans'
augment(x, data, ...)
```

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

Arguments

x	(gmeans()) An object of class "gmeans".
col.names	(character()) Column names for the centers. Defaults to the column names of the centers, or x1, x2, ... if they are unnamed.
...	(any) Additional arguments. Currently unused.
data	(matrix()) The data used to fit the model, a numeric matrix or a data frame.

Value

A data.frame():

- tidy(): one row per cluster with the centers, size, withinss, and cluster.
- augment(): data with a .cluster factor giving the cluster assignments.
- glance(): one row with k, k_init, k_max, level, totss, tot.withinss, betweenss, and iter.

See Also

[gmeans\(\)](#)

Examples

```
library(generics)
x <- as.matrix(iris[, -5])
cl <- gmeans(x)

tidy(cl)
glance(cl)
head(augment(cl, x))
```

predict.gmeans	<i>Predict Method for G-means Clustering</i>
----------------	--

Description

Predicted values based on the G-means clustering model.

Usage

```
## S3 method for class 'gmeans'
predict(
  object,
  newdata,
  method = c("euclidean", "manhattan", "minkowski"),
  p = 2,
  ...
)
```

Arguments

object	(gmeans()) An object of class "gmeans".
newdata	(matrix()) New data to predict on, a numeric matrix or a data frame. Columns are matched to the centers by name and unused columns are ignored.
method	(character(1)) Distance metric to use. Either "euclidean", "manhattan", or "minkowski". Default is "euclidean".
p	(numeric(1)) Power of the Minkowski distance. Must be positive. Default is 2.
...	(any) Additional arguments.

Details

The predict method for G-means clustering assigns new data points to the nearest cluster center identified by the G-means algorithm. The method uses the specified distance metric to calculate the distance between each new data point and all cluster centers, and then assigns each point to the cluster with the closest center.

The method argument specifies the distance metric to use. The following options:

- "euclidean": The Euclidean distance is the default metric used in the k-means and is defined as

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- "manhattan": The Manhattan distance is defined as

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- "minkowski": The Minkowski distance is defined as

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p},$$

where p is a parameter that defines the distance type (e.g., $p = 2$ for Euclidean, $p = 1$ for Manhattan).

Value

An integer() vector with one cluster index per row of newdata.

Source

Adapted from **clue**

See Also

`clue::cl_predict()` to predict on a plain `stats::kmeans()` object.

Examples

```
x <- as.matrix(iris[, -5])
cl <- gmeans(x)

newdata <- x[1:10, ]
predict(cl, newdata)
```

summary.gmeans

*Summarize a G-means Clustering Object***Description**

Summarize the result of a G-means clustering: the number of clusters found, the settings used to fit the model, a per-cluster table of sizes, within-cluster sums of squares and centers, and the overall sums of squares.

Usage

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

## S3 method for class 'summary.gmeans'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object	(gmeans()) An object of class "gmeans".
...	(any) Additional arguments. Currently unused.
x	(summary.gmeans()) An object of class "summary.gmeans".
digits	(integer(1)) Number of significant digits to print.

Value

summary() returns an object of class "summary.gmeans", a list with components:

- k, k_init, k_max, level: the number of clusters found and the settings used to fit the model.
- clusters: a data.frame() with one row per cluster and the columns cluster, size, withinss, followed by the cluster centers.
- totss, tot.withinss, betweenss, iter: as in [stats::kmeans\(\)](#).

print() returns x invisibly.

See Also

[gmeans\(\)](#)

Examples

```
x <- as.matrix(iris[, -5])
cl <- gmeans(x)
summary(cl)
```

Index

`ad.test`, [2](#)
`ad.test()`, [4](#)
`augment.gmeans(gmeans_tidiers)`, [6](#)

`clue::cl_predict()`, [8](#)
`compute_wss`, [3](#)

`glance.gmeans(gmeans_tidiers)`, [6](#)
`gmeans`, [4](#)
`gmeans()`, [6](#), [9](#)
`gmeans_tidiers`, [5](#), [6](#)

`nortest::ad.test()`, [3](#)
`nortest::cvm.test()`, [3](#)
`nortest::lillie.test()`, [3](#)
`nortest::pearson.test()`, [3](#)
`nortest::sf.test()`, [3](#)

`predict.gmeans`, [7](#)
`print.summary.gmeans(summary.gmeans)`, [9](#)

`stats::kmeans()`, [4](#), [5](#), [8](#), [9](#)
`stats::qqnorm()`, [3](#)
`stats::shapiro.test()`, [3](#)
`summary.gmeans`, [9](#)

`tidy.gmeans(gmeans_tidiers)`, [6](#)