

# Package: geokmeans (via r-universe)

June 22, 2026

**Type** Package

**Title** A Collection of Fast, Exact and Eco-Friendly k-Means Clustering Algorithms

**Version** 0.1.0

**Description** A collection of fast k-means clustering algorithms under a single, uniform interface. The core method is Geometric-k-means, a bound-free algorithm of Sharma et al. (2026) <[doi:10.1007/s10994-025-06891-1](https://doi.org/10.1007/s10994-025-06891-1)> that uses geometry to restrict computation to the data points able to change clusters, substantially reducing distance computations and runtime while returning the same result as standard k-means. Also included are Lloyd's algorithm, Elkan, Hamerly, Annulus, Exponion, and Ball k-means. All algorithms are implemented in 'C++' via 'Rcpp' and 'RcppEigen' and return the final centroids, optional per-point cluster assignments, and computational statistics.

**License** GPL-3

**Encoding** UTF-8

**Imports** Rcpp

**LinkingTo** Rcpp, RcppEigen

**SystemRequirements** C++17

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

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** <https://github.com/parichit/Geometric-k-means>

**BugReports** <https://github.com/parichit/Geometric-k-means/issues>

**NeedsCompilation** yes

**Author** Parichit Sharma [aut, cre, cph], Hasan Kurban [aut]

**Maintainer** Parichit Sharma <[parishar@iu.edu](mailto:parishar@iu.edu)>

**Config/roxygen2/version** 8.0.0

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-22 18:31:39 UTC

**RemoteUrl** <https://github.com/cran/geokmeans>

**RemoteRef** HEAD

**RemoteSha** fa9a822ae3f4c9483b3de0ddcb38d282d6781ae9

## Contents

kmeans_algorithms . . . . .	2
kmeans_dc . . . . .	5

<b>Index</b>	<b>7</b>
--------------	----------

---

kmeans_algorithms	<i>k-Means clustering algorithms</i>
-------------------	--------------------------------------

---

## Description

Run one of the bundled k-means variants on a numeric data matrix. All functions share the same interface and return value; they differ only in the acceleration strategy used internally. [geo\\_kmeans\(\)](#) runs the bound-free Geometric-k-means method.

## Usage

```
geo_kmeans(
  data,
  centers,
  iter_max = 100L,
  threshold = 0.001,
  init = c("random", "sequential"),
  seed = NULL,
  with_labels = TRUE,
  verbose = FALSE,
  drop_empty = TRUE
)
```

```
lloyd_kmeans(
  data,
  centers,
  iter_max = 100L,
  threshold = 0.001,
  init = c("random", "sequential"),
  seed = NULL,
  with_labels = TRUE,
  verbose = FALSE,
  drop_empty = TRUE
)
```

```
)  
  
elkan_kmeans(  
  data,  
  centers,  
  iter_max = 100L,  
  threshold = 0.001,  
  init = c("random", "sequential"),  
  seed = NULL,  
  with_labels = TRUE,  
  verbose = FALSE,  
  drop_empty = TRUE  
)  
  
hamerly_kmeans(  
  data,  
  centers,  
  iter_max = 100L,  
  threshold = 0.001,  
  init = c("random", "sequential"),  
  seed = NULL,  
  with_labels = TRUE,  
  verbose = FALSE,  
  drop_empty = TRUE  
)  
  
annulus_kmeans(  
  data,  
  centers,  
  iter_max = 100L,  
  threshold = 0.001,  
  init = c("random", "sequential"),  
  seed = NULL,  
  with_labels = TRUE,  
  verbose = FALSE,  
  drop_empty = TRUE  
)  
  
exponion_kmeans(  
  data,  
  centers,  
  iter_max = 100L,  
  threshold = 0.001,  
  init = c("random", "sequential"),  
  seed = NULL,  
  with_labels = TRUE,  
  verbose = FALSE,  
  drop_empty = TRUE
```

```

)

ball_kmeans(
  data,
  centers,
  iter_max = 100L,
  threshold = 0.001,
  init = c("random", "sequential"),
  seed = NULL,
  with_labels = TRUE,
  verbose = FALSE,
  drop_empty = TRUE
)

```

### Arguments

<code>data</code>	A numeric matrix or data frame with observations in rows and features in columns. Missing values are not allowed.
<code>centers</code>	Either a single positive integer giving the number of clusters $k$ , or a numeric matrix of initial cluster centres (one centroid per row, with <code>ncol(centers) == ncol(data)</code> ).
<code>iter_max</code>	Maximum number of iterations.
<code>threshold</code>	Convergence threshold on centroid movement.
<code>init</code>	Initialisation strategy when <code>centers</code> is a number: "random" (random observations) or "sequential" (the first $k$ observations). Ignored when <code>centers</code> is a matrix.
<code>seed</code>	Optional integer seed for the random initialisation, or NULL (the default). Initialisation uses R's random number generator: supplying a seed sets it via <code>set.seed()</code> so the result is reproducible, while NULL leaves the RNG untouched, so the ambient stream (e.g. a preceding <code>set.seed()</code> in your session) is honoured.
<code>with_labels</code>	Logical; if TRUE (default) the result includes a per-observation cluster assignment computed from the final centroids.
<code>verbose</code>	Logical; if TRUE, print the algorithm's convergence message.
<code>drop_empty</code>	Logical; if TRUE (default), clusters that end up with no assigned observations are removed from the result and the remaining cluster labels are renumbered, with a message. Requesting more clusters than the number of distinct rows in <code>data</code> is always an error.

### Value

An object of class "geokmeans": a list with components

**centroids** A  $k \times \text{ncol}(\text{data})$  matrix of final cluster centres.

**cluster** Integer vector of cluster ids (1-based), if `with_labels = TRUE`.

**iterations** Number of iterations performed.

**distance\_calculations** Total number of point-to-centroid distance computations.

**method** The algorithm used.

**k** The number of clusters.

## References

Sharma, P., Stanislaw, M., Kurban, H., Kulekci, O., and Dalkilic, M. (2026). Geometric-k-means: A Bound Free Approach to Fast and Eco-Friendly k-means. doi:10.1007/s10994025068911

## Examples

```
set.seed(1)
X <- rbind(matrix(rnorm(100, 0), ncol = 2),
            matrix(rnorm(100, 5), ncol = 2))
fit <- geo_kmeans(X, centers = 2)
fit$centroids
table(fit$cluster)

# Supplying explicit starting centroids:
geo_kmeans(X, centers = X[c(1, 51), ])
```

---

kmeans\_dc

*Run a k-means variant by name*


---

## Description

A thin dispatcher over the individual algorithm functions.

## Usage

```
kmeans_dc(
  data,
  centers,
  method = c("geokmeans", "lloyd", "elkan", "hamerly", "annulus", "exponion", "ball"),
  ...
)
```

## Arguments

data	A numeric matrix or data frame with observations in rows and features in columns. Missing values are not allowed.
centers	Either a single positive integer giving the number of clusters k, or a numeric matrix of initial cluster centres (one centroid per row, with ncol(centers) == ncol(data)).
method	The algorithm to use. One of "geokmeans", "lloyd", "elkan", "hamerly", "annulus", "exponion", "ball".
...	Further arguments passed to the chosen algorithm.

**Value**

An object of class "geokmeans"; see [geo\\_kmeans\(\)](#).

**Examples**

```
set.seed(1)
X <- rbind(matrix(rnorm(100, 0), ncol = 2),
            matrix(rnorm(100, 5), ncol = 2))
kmeans_dc(X, centers = 2, method = "elkan")
```

# Index

`annulus_kmeans` (`kmeans_algorithms`), 2  
`ball_kmeans` (`kmeans_algorithms`), 2  
`elkan_kmeans` (`kmeans_algorithms`), 2  
`exponion_kmeans` (`kmeans_algorithms`), 2  
`geo_kmeans` (`kmeans_algorithms`), 2  
`geo_kmeans()`, 2, 6  
`hamerly_kmeans` (`kmeans_algorithms`), 2  
`kmeans_algorithms`, 2  
`kmeans_dc`, 5  
`lloyd_kmeans` (`kmeans_algorithms`), 2  
`set.seed()`, 4