# Package: kkmeans (via r-universe)

October 18, 2026

**Title** Fast Implementations of Kernel K-Means

**Version** 0.1.3

**Description** Implementations several algorithms for kernel k-means. The default 'OTQT' algorithm is a fast alternative to standard implementations of kernel k-means, particularly in cases with many clusters. For a small number of clusters, the implemented 'MacQueen' method typically performs the fastest. For more details and performance evaluations, see Berlinski and Maitra (2025) <doi:10.1002/sam.70032>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**LinkingTo** Rcpp

**Imports** Rcpp

**Repository** https://jdberlinski.r-universe.dev

**Date/Publication** 2026-05-20 23:22:13 UTC

**RemoteUrl** https://github.com/jdberlinski/kkmeans

**RemoteRef** HEAD

**RemoteSha** 81d7508d3b9c8718f74ed3ee66606780da2cf8d1

## Contents


cluster_new . . . . . 2
get_kernel_matrix . . . . . 2
get_mknn_dist . . . . . 3
jump_stat . . . . . 3
kkmeans . . . . . 4
matr . . . . . 5

**Index** **7**

## cluster_new *Classify new data based on kkmeans result*

### Description

Given training data, test data, and kkmeans-results, get which partitions a new set of data belong to

### Usage

```
cluster_new(test_data, train_data, train_result)
```

### Arguments

| | |
|---|---|
| test_data | the new data to be classified |
| train_data | the data to make classifications from |
| train_result | kkmeans result containing the result from classifying train_data |

### Value

A vector of class labels for test_data corresponding to the clusters present in train_result.

## get_kernel_matrix *Get the kernel matrix for a dataset*

### Description

Given a dataset, kernel function, and tuning parameter, will return the n x n kernel matrix

### Usage

```
get_kernel_matrix(data, kern = "g", param1 = 1, param2 = 1)
```

### Arguments

| | |
|---|---|
| data | data vector |
| kern | the kernel to use, one of ('gaussian', 'poly', 'sigmoid', 'laplacian'), can use first letter |
| param1 | first parameter to pass to kernel function. |
| param2 | second parameter to pass to kernel function. |

### Value

An n x n matrix for data given by the specified kernel. The value in position (i, j) corresponds to the kernel function evaluated at data[i, ] and data[j, ].

## get_mknn_dist *Get the average distance to each points k-nearest neighbor*

### Description

Given a dataset and a value k, will return the value of the average distance from each point to it's k-nearest neighbor

### Usage

```
get_mknn_dist(data, k = FALSE)
```

### Arguments

| | |
|---|---|
| data | the data vector |
| k | which neighbor to average over |

### Value

The average distance from each point to it's k-th nearest neighbor.

## jump_stat *Function to get jump statistic for varying values of k*

### Description

Obtains the jump statistic for a particular kernel for the specified number of clusters

### Usage

```
jump_stat(data, kern = "g", param = 1, k_max, eta, iter_max = 1000L)
```

### Arguments

| | |
|---|---|
| data | Numeric data to cluster. This will be converted to a matrix using as.matrix. |
| kern | The kernel to use. |
| param | The parameter value to pass to the kernel |
| k_max | The maximum number of clusters to consider |
| eta | Power for the jump statistic |
| iter_max | Maximum number of iterations to use in kkmeans call |

### Value

Sum of squares and value of jump statistic for 1, ..., K chosen clusters

kkmeans *An Efficient Kernel K-Means Algorithm*

## Description

Performs kernel k-means with the specified kernel using an optimal-transfer quick-transfer algorithm.

## Usage

```
kkmeans(
  data,
  k,
  kern = "g",
  param = 1,
  param2 = 1,
  nstart = 10,
  iter_max = 1000L,
  estimate = FALSE,
  nn = 0,
  init_centers = sample(1:k, size = nrow(data), replace = TRUE),
  method = c("otqt", "macqueen", "lloyd", "ot"),
  trueest = FALSE,
  kmat = NULL,
  random_centers = TRUE
)
```

## Arguments

| | |
|---|---|
| data | Numeric data to cluster. This will be converted to a matrix using as.matrix. |
| k | Number of clusters. |
| kern | Kernel to use, one of ('gaussian', 'poly', 'sigmoid', 'laplacian'). |
| param | value of parameter to pass to kernel function.(eg sigma in gaussian kernel). The Gaussian kernel is K(x, y) = exp(- ‖x - y‖^2 / (2*param))), and the polynomial kernel is K(x, y) = (x'y + a) ^ param |
| param2 | value of second parameter parameter to pass to the kernel function, which correspond to the offset for the sigmoid and polynomial kernels. |
| nstart | Number of times to run the algorithm. The run with the lowest total within cluster SSE (in feature space) will be returned |
| iter_max | The maximum number of iterations to allow. |
| estimate | If using the Gaussian kernel, specifying estimate = "mknn" will use an nn-nearest neighbor method for estimating param. |
| nn | How many neighbors to consider for mknn estimation. |
| init_centers | The initial values for cluster membership. If nstart is greater than 1, any start beyond the first iteration will use randomized centers. |

| | |
|---|---|
| method | Which method to use for kernel k-means iteration. One of ("otqt", "macqueen", "lloyd"). "otqt" is a method using optimal-transfer and quick-transfer heuristics similar to the Hartigan and Wong algorithm for k-means clustering. |
| trueest | Whether or not the within-cluster sum of squares should be recomputed in R after clustering is finished |
| kmat | kernel matrix, if using a custom kernel |
| random_centers | if TRUE, then assign k observations as initial clusters, assigning the remaining observations to the closest cluster. Otherwise, assign all observations to clusters at random. |

## Value

A list containing the following useful information

**cluster** The final cluster membership.

**centers** A k x p matrix, the rows of which contain the centers of the clusters in R^n (not to be confused with the clusters in feature space)

**wss** The within-cluster sum of squares for each cluster in feature space.

**param** The parameter value used.

## Examples

```
data <- as.matrix(iris[, 1:4])

# cluster using linear kernel (normal k-means)
result <- kkmeans(data, k = 3, kern = "poly", param = 1)

# cluster using gaussian kernel
# estimating the parameter with 3-nearest neighbors
result <- kkmeans(data, k = 3, kern = "g", estimate = "mknn", nn = 3)
```

---

# matr *Estimate the bandwidth parameter for a gaussian kernel using MATr*

---

## Description

given data and number of clusters K, choose a bandwidth from a grid according to the MATR algorithm

## Usage

```
matr(dat, k, grid, tol = 0.01)
```

## Arguments

| | |
|---|---|
| dat | data vector |
| k | number of clusters |
| grid | parameter grid to search on |
| tol | tolerence for choosing the "best" sigma values. Reducing the tolerance will give larger values of the bandwidth parameter |

## Value

A list with two elements. The first, l, contains the trace value for each of the values in grid. The second, best, contains the best value.

# Index

cluster_new, 2

get_kernel_matrix, 2
get_mknn_dist, 3

jump_stat, 3

kkmeans, 4

matr, 5