

Package ‘banditpam’

July 22, 2025

Title Almost Linear-Time k-Medoids Clustering

Version 1.0-2

Description Interface to a high-performance implementation of k-medoids clustering described in Tiwari, Zhang, Mayclin, Thrun, Piech and Shomorony (2020) ``BanditPAM: Almost Linear Time k-medoids Clustering via Multi-Armed Bandits" <<https://proceedings.neurips.cc/paper/2020/file/73b817090081cef1bca77232f4532c5d-Paper.pdf>>.

License MIT + file LICENSE

Encoding UTF-8

BugReports <https://github.com/motiwari/BanditPAM/issues>

SystemRequirements C++17

Depends R (>= 3.5.0)

RoxygenNote 7.3.2

Suggests ggplot2, knitr, MASS, rmarkdown, tinytest

LinkingTo Rcpp, RcppArmadillo

Imports R6, Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Balasubramanian Narasimhan [aut, cre],
Mo Tiwari [aut] (<https://motiwari.com>)

Maintainer Balasubramanian Narasimhan <naras@stanford.edu>

Repository CRAN

Date/Publication 2025-06-02 20:30:01 UTC

Contents

banditpam-package	2
bpam_num_threads	2
KMedoids	3

Index	6
--------------	----------

banditpam-package *banditpam is a package for fast clustering using medoids*

Description

banditpam is a high-performance package for almost linear-time k-medoids clustering. The methods are described in Tiwari, et al. 2020 (Advances in Neural Information Processing Systems 33).

Author(s)

Balasubramanian Narasimhan and Mo Tiwari

See Also

Useful links:

- Report bugs at <https://github.com/motiwari/BanditPAM/issues>

bpam_num_threads *Return the number of threads banditpam is using*

Description

Return the number of threads banditpam is using

Usage

```
bpam_num_threads()
```

Value

the number of threads banditpam is using

KMedoids

*KMedoids Class***Description**

This class wraps around the C++ KMedoids class and exposes methods and fields of the C++ object.

Active bindings

`k` (integer(1))
 The number of medoids/clusters to create

`max_iter` (integer(1))
 max_iter the maximum number of SWAP steps the algorithm runs

`build_conf` (integer(1))
 Parameter that affects the width of BUILD confidence intervals, default 1000

`swap_conf` (integer(1))
 Parameter that affects the width of SWAP confidence intervals, default 10000

`loss_fn` (character(1))
 The loss function, "lp" (for p integer > 0) or one of "manhattan", "cosine", "inf" or "euclidean"

Methods**Public methods:**

- [KMedoids\\$new\(\)](#)
- [KMedoids\\$get_algorithm\(\)](#)
- [KMedoids\\$fit\(\)](#)
- [KMedoids\\$get_medoids_final\(\)](#)
- [KMedoids\\$get_labels\(\)](#)
- [KMedoids\\$get_statistic\(\)](#)
- [KMedoids\\$print\(\)](#)
- [KMedoids\\$clone\(\)](#)

Method `new()`: Create a new KMedoids object

Usage:

```
KMedoids$new(
  k = 5L,
  algorithm = c("BanditPAM", "PAM", "FastPAM1"),
  max_iter = 1000L,
  build_conf = 1000,
  swap_conf = 10000L
)
```

Arguments:

`k` number of medoids/clusters to create, default 5

algorithm the algorithm to use, one of "BanditPAM", "PAM", "FastPAM1"
max_iter the maximum number of SWAP steps the algorithm runs, default 1000
build_conf parameter that affects the width of BUILD confidence intervals, default 1000
swap_conf parameter that affects the width of SWAP confidence intervals, default 10000
Returns: a KMedoids object which can be used to fit the banditpam algorithm to data

Method `get_algorithm()`: Return the algorithm used

Usage:

`KMedoids$get_algorithm()`

Returns: a string indicating the algorithm

Method `fit()`: Fit the KMedoids algorithm given the data and loss. It is advisable to set the seed before calling this method for reproducible results.

Usage:

`KMedoids$fit(data, loss, dist_mat = NULL)`

Arguments:

data the data matrix

loss the loss function, either "lp" (p, integer indicating L_p loss) or one of "manhattan", "cosine", "inf" or "euclidean"

dist_mat an optional distance matrix

Method `get_medoids_final()`: Return the final medoid indices after clustering

Usage:

`KMedoids$get_medoids_final()`

Returns: a vector indices of the final medoids

Method `get_labels()`: Return the cluster labels after clustering

Usage:

`KMedoids$get_labels()`

Returns: a vector of the cluster labels for the observations

Method `get_statistic()`: Get the specified statistic after clustering

Usage:

`KMedoids$get_statistic(what)`

Arguments:

what a string which should one of "dist_computations", "dist_computations_and_misc", "misc_dist", "build_dist", "swap_dist", "cache_writes", "cache_hits", or "cache_misses"

return the statistic

Method `print()`: Printer.

Usage:

`KMedoids$print(...)`

Arguments:

... (ignored).

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
KMedoids$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Generate data from a Gaussian Mixture Model with the given means:
set.seed(10)
n_per_cluster <- 40
means <- list(c(0, 0), c(-5, 5), c(5, 5))
X <- do.call(rbind, lapply(means, MASS::mvrnorm, n = n_per_cluster, Sigma = diag(2)))
obj <- KMedoids$new(k = 3)
obj$fit(data = X, loss = "l2")
meds <- obj$get_medoids_final()
plot(X[, 1], X[, 2])
points(X[meds, 1], X[meds, 2], col = "red", pch = 19)
```

Index

* **package**

banditpam-package, [2](#)

banditpam (banditpam-package), [2](#)

banditpam-package, [2](#)

bpam_num_threads, [2](#)

KMedoids, [3](#)