Package: speakeasyR (via r-universe)

September 18, 2024

Title Fast and Robust Multi-Scale Graph Clustering

Version 0.1.3

Description A graph community detection algorithm that aims to be performant on large graphs and robust, returning consistent results across runs. SpeakEasy 2 (SE2), the underlying algorithm, is described in Chris Gaiteri, David R. Connell & Faraz A. Sultan et al. (2023) <doi:10.1186/s13059-023-03062-0>. The core algorithm is written in 'C', providing speed and keeping the memory requirements low. This implementation can take advantage of multiple computing cores without increasing memory usage. SE2 can detect community structure across scales, making it a good choice for biological data, which often has hierarchical structure. Graphs can be passed to the algorithm as adjacency matrices using base 'R' matrices, the 'Matrix' library, 'igraph' graphs, or any data that can be coerced into a matrix.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports Matrix, methods

Suggests igraph, scRNAseq, SummarizedExperiment, knitr, rmarkdown, testthat (>= 3.0.0)

URL https://github.com/SpeakEasy-2/speakeasyR

BugReports https://github.com/SpeakEasy-2/speakeasyR/issues

VignetteBuilder knitr

Config/testthat/edition 3

SystemRequirements arpack (optional)

Repository https://speakeasy-2.r-universe.dev

RemoteUrl https://github.com/speakeasy-2/speakeasyr

RemoteRef HEAD

RemoteSha 79faabcef450780e3e69fc30e8ff2212c5b34286

Contents

`cluster` *SpeakEasy 2 community detection*

Description

Group nodes into communities.

Usage

```
cluster(
  graph,
  discard_transient = 3,
  independent_runs = 10,
  max_threads = 0,
  seed = 0,
  target_clusters = 0,
  target_partitions = 5,
  subcluster = 1,
  min_clust = 5,
  verbose = FALSE,
  is_directed = "detect"
)
```

Arguments

`graph`	A graph or adjacency matrix in a form that can be converted to `matrix` or `Matrix::dgCMatrix` using an `as.matrix()` coercion method. Accepted types include `matrix`, `dgCMatrix`, `ngCMatrix`, and `igraph::graphs`.
`discard_transient`	The number of partitions to discard before tracking.
`independent_runs`	How many runs SpeakEasy2 should perform.
`max_threads`	The maximum number of threads to use. By default this is the same as the number of independent runs. If max_threads is greater than or equal to the number of processing cores, all cores may run. If max_threads is less than the number of cores, at most max_threads cores will run.
`seed`	Random seed to use for reproducible results. SpeakEasy2 uses a different random number generator than R, but if the seed is not explicitly set, R's random number generator is used create one. Because of this, setting R's RNG will also cause reproducible results.

target_clusters
: The number of random initial labels to use.

target_partitions
: Number of partitions to find per independent run.

subcluster
: Depth of clustering. If greater than 1, perform recursive clustering.

min_clust
: Smallest clusters to recursively cluster. If subcluster not set to a value greater than 1, this has no effect.

verbose
: Whether to provide additional information about the clustering or not.

is_directed
: Whether the graph should be treated as directed or not. By default, if the graph is symmetric it is treated as undirected.

Value

A membership vector. If subclustering, returns a matrix with number of rows equal to the number of recursive clustering. Each row is the membership at different hierarchical scales, such that the last rows are the highest resolution.

Examples

```
if (require("igraph")) {
  graph <- igraph::graph.famous("zachary")
  membership <- cluster(graph, max_threads = 2)
}
```

---

knn_graph *K-nearest neighbors graph*

Description

Create a directed sparse graph with edges to each nodes k nearest neighbors. Nearness is calculated as the inverse of the euclidean distance between two columns.

Usage

```
knn_graph(mat, k, weighted = FALSE)
```

Arguments

mat
: A matrix to be compared column-by-column.

k
: How many nearest neighbors to collect.

weighted
: By default, a binary edge is made between a node and each of it's k closest nodes. Set `weighted` to `TRUE` to weigh each edge by the similarity (inverse of euclidean distance).

Value

A directed sparse adjacency matrix with `k * ncol(mat)` nonzero edges. Each column has k edges connected to the k closest columns (not including itself).

Examples

```
# Simple random graph
mat <- matrix(runif(100) > 0.75, nrow = 5)
knn_graph(mat, 3)

## Don't run because loading data is slow.

if (requireNamespace("scRNAseq") &&
  requireNamespace("SummarizedExperiment")) {
  # Single Cell RNA data
  library(Matrix)

  expression <- scRNAseq::FletcherOlfactoryData()
  cell_types <- expression$cluster_id

  ## Filter genes with low expression. Remove any genes with less than 10
  ## cells with with any reads.
  counts <- SummarizedExperiment::assay(expression, "counts")
  indices <- rowSums(counts > 0) > 10
  counts <- counts[indices, ]

  ## Normalize by shifted logarithm
  target <- median(colSums(counts))
  size_factors <- colSums(counts) / target
  counts_norm <- log(t(t(counts) / size_factors + 1))

  ## Dimension reduction
  counts_norm <- t(prcomp(t(counts_norm), scale. = FALSE)$x)[1:50, ]

  adj <- knn_graph(counts_norm, 10)
}
```

`order_nodes` *Group nodes by community*

Description

Reorders the graph to group nodes in the same community together. Useful for viewing community structure of a graph using a `heatmap()`.

Usage

```
order_nodes(graph, membership, is_directed = "detect")
```

Arguments

`graph`	The graph or adjacency matrix the membership vector was created for.

membership	A vector or matrix listing node communities. The output from cluster() (should also work for other clustering algorithms that return membership in the same format).
is_directed	Whether the graph should be treated as directed or not. By default, if the graph is symmetric it is treated as undirected.

Details

Communities are ordered by size, so nodes in the largest community are first. Within a community, nodes are order by highest-to-lowest degree.

If membership is in matrix form (the output from cluster() with subcluster > 1) a matrix is returned with the indices for level one in row 1 and level n in row n. Each row reorders the communities of the previous row such that, at the second level, nodes are still grouped by the first level communities. This allows the hierarchical structure to be viewed.

See vignette for a multilevel example.

Value

An index vector or matrix. The number of rows are equal to the value of subcluster passed to cluster().

Examples

```
if (require("igraph")) {
  n_nodes <- 100
  n_types <- 3
  # Mixing parameter (likelihood an edge is between communities).
  mu <- 0.3
  pref <- matrix(mu, n_types, n_types)
  diag(pref) <- 1 - mu
  g <- igraph::sample_pref(n_nodes, types = n_types, pref.matrix = pref)
  # Use a dense matrix representation to easily apply index.
  adj <- as(g[], "matrix")
  memb <- speakeasyR::cluster(adj, seed = 222, max_threads = 2)
  ordering <- speakeasyR::order_nodes(adj, memb)

  heatmap(adj[ordering, ordering], scale = "none", Rowv = NA, Colv = NA)

}
```

Index