Package ‘gdim’

July 22, 2025

Title Estimate Graph Dimension using Cross-Validated Eigenvalues

Version 0.1.0

Description Cross-validated eigenvalues are estimated by
splitting a graph into two parts, the training and the test graph.
The training graph is used to estimate eigenvectors, and
the test graph is used to evaluate the correlation between the training
eigenvectors and the eigenvectors of the test graph.
The correlations follow a simple central limit theorem that can
be used to estimate graph dimension via hypothesis testing, see
Chen et al. (2021) <doi:10.48550/arXiv.2108.03336> for details.

License GPL (>= 3)

URL https://github.com/RoheLab/gdim, https://rohelab.github.io/gdim/

BugReports https://github.com/RoheLab/gdim/issues

Depends Matrix, R (>= 3.5)

Imports dplyr, ggplot2, irlba, magrittr, methods, progress, rlang,
stats, tibble

Suggests epca, fastRG, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Fan Chen [aut] (ORCID: <https://orcid.org/0000-0003-4508-6023>),
Alex Hayes [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0002-4985-5160>),
Karl Rohe [aut]

Maintainer Alex Hayes <alexpghayes@gmail.com>

Repository CRAN

Date/Publication 2023-09-05 20:20:02 UTC

Contents

eigcv 2
plot.eigcv 4
print.eigcv 5

Index **7**

---

eigcv *Compute cross-validate eigenvalues*

Description

Estimate graph dimension via eigenvalue cross-validation (EigCV). A graph has dimension k if the first k eigenvectors of its adjacency matrix are correlated with its population eigenspace, and the others are not. Edge bootstrapping sub-samples the edges of the graph (without replacement). Edge splitting separates the edges into a training part and a testing part.

Usage

```
eigcv(
  A,
  k_max,
  ...,
  num_bootstraps = 10,
  test_portion = 0.1,
  alpha = 0.05,
 method = c("none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr"),
  laplacian = FALSE,
  regularize = TRUE
)
```

Arguments

`A`	The adjacency matrix of graph. Must be non-negative and integer valued.
`k_max`	The maximum dimension of the graph to consider. This many eigenvectors are computed. Should be a non-negative integer smallish relative the dimensions of `A`.
`...`	Ignored.
`num_bootstraps`	The number of times to bootstrap the graph. Since cross-validated eigenvalues are based on a random graph split, they are themselves random. By repeatedly computing cross-validated eigenvalues for different sample splits, the idea is to smooth away some of the randomness due to the graph splits. A small number of bootstraps (3 to 10) usually suffices. Defaults to `10`. Test statistics (i.e. z-scores for cv eigenvalues) are averaged across bootstraps and the p-values will be calculated based on the averaged statistics.

`test_portion`	The portion of the graph to put into the test graph, as opposed to the training graph. Defaults to `0.1`. Must be strictly between zero and one.
`alpha`	Significance level for hypothesis tests. Each dimension `1, ..., k_max` is tested when estimating graph dimension, and the overall graph dimension is taken to be the smallest number of dimensions such that all the tests reject.
`method`	Method to adjust p-values for multiple testing. Must be one of `"none"`, `"holm"`, `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"`, `"BY"`, or `"fdr"`. Passed to `stats::p.adjust()`. Defaults to `"none"`.
`laplacian`	Logical value indicating where to compute cross-validated eigenvalues for the degree-normalize graph Laplacian rather than the graph adjacency matrix. Experimental and should be used with caution. Defaults to `FALSE`.
`regularize`	Only applicable when `laplacian == TRUE`, in which case this parameter controls whether or not the degree-normalized graph Laplacian is regularized. Defaults to `TRUE`.

Value

A `eigcv` object, which is a list with the following named elements.

- `estimated_dimension`: inferred graph dimension.
- `summary`: summary table of the tests.
- `num_bootstraps`: number of bootstraps performed.
- `test_portion`: graph splitting probability used.
- `alpha`: significance level of each test.

Examples

```
library(fastRG)

set.seed(27)

B <- matrix(0.1, 5, 5)
diag(B) <- 0.3

model <- sbm(
  n = 1000,
  k = 5,
  B = B,
  expected_degree = 40,
  poisson_edges = FALSE,
  allow_self_loops = FALSE
)

A <- sample_sparse(model)

eigs<- eigcv(A, k_max = 10)
eigs

plot(eigs, type = "z-score")    # default
plot(eigs, type = "adjacency")
plot(eigs, type = "laplacian")
```

---

plot.eigcv *Plot cross-validated eigenvalues*

Description

Plot cross-validated eigenvalues

Usage

```
## S3 method for class 'eigcv'
plot(x, type = c("z-score", "adjacency", "laplacian"), threshold = 2, ...)
```

Arguments

`x`	An `eigcv` object created by a call to `eigcv()`.
`type`	Specifies what to plot. Must be one of the following options: • `"z-score"`, in which case the Z-statistic test scores are plotted for each value of `k` (i.e. dimension of the eigenspace). • `"adjacency"` in which case the cross-validated eigenvalues of the adjacency matrix are plotted for each value of `k`. • `"laplacian"` in which case the cross-validated eigenvalues of the graph Laplacian matrix are plotted for each value of `k`.
`threshold`	Only used when `type == "z-score"`. Adds a horizontal line at the value of `threshold`, which should be a numeric of length one. Defaults to `2`.
`...`	Ignored.

Value

A `ggplot2` object.

Examples

```
library(fastRG)

set.seed(27)

B <- matrix(0.1, 5, 5)
diag(B) <- 0.3

model <- sbm(
  n = 1000,
  k = 5,
  B = B,
  expected_degree = 40,
  poisson_edges = FALSE,
  allow_self_loops = FALSE
)

A <- sample_sparse(model)

eigs<- eigcv(A, k_max = 10)
eigs

plot(eigs, type = "z-score")    # default
plot(eigs, type = "adjacency")
plot(eigs, type = "laplacian")
```

`print.eigcv` *Print cross-validated eigenvalues*

Description

Print cross-validated eigenvalues

Usage

```
## S3 method for class 'eigcv'
print(x, ...)
```

Arguments

`x`	An `eigcv` object created by a call to `eigcv()`.
`...`	Ignored.

Value

x, but invisibly.

Examples

```
library(fastRG)

set.seed(27)

B <- matrix(0.1, 5, 5)
diag(B) <- 0.3

model <- sbm(
  n = 1000,
  k = 5,
  B = B,
  expected_degree = 40,
  poisson_edges = FALSE,
  allow_self_loops = FALSE
)

A <- sample_sparse(model)

eigs<- eigcv(A, k_max = 10)
eigs

plot(eigs, type = "z-score")    # default
plot(eigs, type = "adjacency")
plot(eigs, type = "laplacian")
```

Index

eigcv, 2
eigcv(), *4*, *5*

plot.eigcv, 4
print.eigcv, 5

stats::p.adjust(), *3*