Package: vsp (via r-universe)

November 6, 2024

Type Package

Title Vintage Sparse PCA for Semi-Parametric Factor Analysis

Version 0.1.2

Description Provides fast spectral estimation of latent factors in random dot product graphs using the vsp estimator. Under mild assumptions, the vsp estimator is consistent for (degree-corrected) stochastic blockmodels, (degree-corrected) mixed-membership stochastic blockmodels, and degree-corrected overlapping stochastic blockmodels.

License MIT + file LICENSE

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

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

Depends R (>= 3.1)

Imports clue, ggplot2, glue, invertiforms, LRMF3, magrittr, Matrix, rlang, RSpectra, stats, tibble, withr

Suggests covr, dplyr, GGally, igraph, igraphdata, knitr, purrr, rmarkdown, scales, testthat (>= 3.0.0), tidygraph, tidyr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

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

Maintainer Alex Hayes <alexpghayes@gmail.com>

Repository CRAN

Date/Publication 2024-11-05 19:40:02 UTC

Contents

`bff` *Find features most associated with cluster membership*

Description

Find features most associated with cluster membership

Usage

```
bff(loadings, features, num_best)
```

Arguments

`loadings`	An `n` by `k` matrix of weights that indicates how important that ith user is to the jth cluster, i.e., the `Z` or `Y` matrix calculated by `vsp()`.
`features`	An `n` by `d` matrix of features measured for each node in the network.
`num_best`	An integer indicating how many of the top features for differentiating between loadings you want.

Details

See `vignette("bff")`.

Value

An `n` by `k` matrix whose `[i, j]` entry is the ith "most important" feature for cluster `j`.

`bind_varimax_z` *Add Z factor loadings to node table of tidygraph*

Description

Add Z factor loadings to node table of tidygraph

Usage

```
bind_varimax_z(graph, fa, ...)

bind_varimax_y(graph, fa, ...)

bind_svd_u(graph, fa, ...)

bind_svd_v(graph, fa, ...)
```

Arguments

`graph`	A tidygraph::tbl_graph object.
`fa`	Optionally, a vsp object to extract varimax loadings from. If you do not passed a vsp object, one will be created.
`...`	Arguments passed on to `vsp` `x` Either a graph adjacency matrix, igraph::igraph or tidygraph::tbl_graph. If `x` is a matrix or Matrix::Matrix then `x[i, j]` should correspond to the edge going from node `i` to node `j`. `rank` The number of factors to calculate.

Value

The same `graph` object with columns `factor1`, ..., `factor{rank}` in the table of node information.

Functions

- `bind_varimax_y()`: Add Y factor loadings to node table of tidygraph
- `bind_svd_u()`: Add left singular vectors to node table of tidygraph
- `bind_svd_v()`: Add right singular vectors to node table of tidygraph

get_svd_u *Get left singular vectors in a tibble*

Description

Get left singular vectors in a tibble

Usage

```
get_svd_u(fa, factors = 1:fa$rank)

get_svd_v(fa, factors = 1:fa$rank)

get_varimax_z(fa, factors = 1:fa$rank)

get_varimax_y(fa, factors = 1:fa$rank)
```

Arguments

fa	A vsp_fa() object.
factors	The specific columns to index into. The most reliable option here is to index with an integer vector of column indices, but you could also use a character vector if columns have been named. By default returns all factors/singular vectors.

Value

A tibble::tibble() with one row for each node, and one column containing each of the requested factor or singular vector, plus an additional id column.

Functions

- get_svd_v(): Get right singular vectors in a tibble
- get_varimax_z(): Get varimax Y factors in a tibble
- get_varimax_y(): Get varimax Z factors in a tibble

Examples

```
data(enron, package = "igraphdata")

fa <- vsp(enron, rank = 30)
fa

get_svd_u(fa)
get_svd_v(fa)

get_varimax_z(fa)
get_varimax_y(fa)
```

get_z_hubs *Get most important hubs for each Z factor*

Description

Get most important hubs for each Z factor

Usage

```
get_z_hubs(fa, hubs_per_factor = 10, factors = 1:fa$rank)

get_y_hubs(fa, hubs_per_factor = 10, factors = 1:fa$rank)
```

Arguments

fa	A vsp_fa() object.
hubs_per_factor	The number of important nodes to get per latent factor. Defaults to 10.
factors	The specific columns to index into. The most reliable option here is to index with an integer vector of column indices, but you could also use a character vector if columns have been named. By default returns all factors/singular vectors.

Value

A tibble::tibble() where each row corresponds to a single hub, and three columns:

- id: Node id of hub node
- factor: Which factor that node is a hub for. Nodes can be hubs of multiple factors.
- loading: The actual value of the hubs factor loading for that factor.

Functions

- get_y_hubs(): Get most important hubs for each Y factor

Examples

```
data(enron, package = "igraphdata")

fa <- vsp(enron, rank = 30)
fa

get_z_hubs(fa)
get_y_hubs(fa)
```

plot_ipr_pairs *Plot pairs of inverse participation ratios for singular vectors*

Description

When IPR for a given singular vector is O(1) rather than O(1 / sqrt(n)), this can indicate that the singular vector is localizing on a small subset of nodes. Oftentimes this localization indicates overfitting. If you see IPR values that are not close to zero (where "close to zero" is something you sort of have to pick up over time), then you need to some further investigation to see if you have localization and that localization corresponds to overfitting. Note, however, that not all localization is overfitting.

Usage

```
plot_ipr_pairs(fa)
```

Arguments

fa	A vsp_fa() object.

Value

A tibble::tibble() with one row for each node, and one column containing each of the requested factor or singular vector, plus an additional id column.

plot_mixing_matrix *Plot the mixing matrix B*

Description

Plot the mixing matrix B

Usage

```
plot_mixing_matrix(fa)
```

Arguments

fa	A vsp_fa() object.

Value

A tibble::tibble() with one row for each node, and one column containing each of the requested factor or singular vector, plus an additional id column.

`plot_varimax_z_pairs` *Create a pairs plot of select Y factors*

Description

To avoid overplotting, plots data for a maximum of 1000 nodes. If there are more than 1000 nodes, samples 1000 nodes randomly proportional to row norms (i.e. nodes with embeddings larger in magnitude are more likely to be sampled).

Usage

```
plot_varimax_z_pairs(fa, factors = 1:min(5, fa$rank), ...)

plot_varimax_y_pairs(fa, factors = 1:min(5, fa$rank), ...)

plot_svd_u(fa, factors = 1:min(5, fa$rank))

plot_svd_v(fa, factors = 1:min(5, fa$rank))
```

Arguments

`fa`	A `vsp_fa()` object.
`factors`	The specific columns to index into. The most reliable option here is to index with an integer vector of column indices, but you could also use a character vector if columns have been named. By default returns all factors/singular vectors.
`...`	Arguments passed on to `GGally::ggpairs` `data` data set using. Can have both numerical and categorical data. `mapping` aesthetic mapping (besides `x` and `y`). See `aes()`. If `mapping` is numeric, `columns` will be set to the `mapping` value and `mapping` will be set to `NULL`. `columns` which columns are used to make plots. Defaults to all columns. `title,xlab,ylab` title, x label, and y label for the graph `upper` see Details `lower` see Details `diag` see Details `params` deprecated. Please see `wrap_fn_with_param_arg` `axisLabels` either "show" to display axisLabels, "internal" for labels in the diagonal plots, or "none" for no axis labels `columnLabels` label names to be displayed. Defaults to names of columns being used. `labeller` labeller for facets. See `labellers`. Common values are `"label_value"` (default) and `"label_parsed"`.

`switch` switch parameter for facet_grid. See `ggplot2::facet_grid`. By default, the labels are displayed on the top and right of the plot. If `"x"`, the top labels will be displayed to the bottom. If `"y"`, the right-hand side labels will be displayed to the left. Can also be set to `"both"`

`showStrips` boolean to determine if each plot's strips should be displayed. `NULL` will default to the top and right side plots only. `TRUE` or `FALSE` will turn all strips on or off respectively.

`legend` May be the two objects described below or the default `NULL` value. The legend position can be moved by using ggplot2's theme element `pm + theme(legend.position = "bottom")`

a numeric vector of length 2 provides the location of the plot to use the legend for the plot matrix's legend. Such as `legend = c(3,5)` which will use the legend from the plot in the third row and fifth column

a single numeric value provides the location of a plot according to the display order. Such as `legend = 3` in a plot matrix with 2 rows and 5 columns displayed by column will return the plot in position `c(1,2)`

a object from `grab_legend()` a predetermined plot legend that will be displayed directly

`cardinality_threshold` maximum number of levels allowed in a character / factor column. Set this value to NULL to not check factor columns. Defaults to 15

`progress` `NULL` (default) for a progress bar in interactive sessions with more than 15 plots, `TRUE` for a progress bar, `FALSE` for no progress bar, or a function that accepts at least a plot matrix and returns a new `progress::progress_bar`. See `ggmatrix_progress`.

`proportions` Value to change how much area is given for each plot. Either `NULL` (default), numeric value matching respective length, `grid::unit` object with matching respective length or `"auto"` for automatic relative proportions based on the number of levels for categorical variables.

`legends` deprecated

Value

A `ggplot2::ggplot()` plot or `GGally::ggpairs()` plot.

Functions

- `plot_varimax_y_pairs()`: Create a pairs plot of select Z factors
- `plot_svd_u()`: Create a pairs plot of select left singular vectors
- `plot_svd_v()`: Create a pairs plot of select right singular vectors

Examples

```
data(enron, package = "igraphdata")

fa <- vsp(enron, rank = 3)

plot_varimax_z_pairs(fa)
```

```
plot_varimax_y_pairs(fa)

plot_svd_u(fa)
plot_svd_v(fa)

screeplot(fa)

plot_mixing_matrix(fa)

plot_ipr_pairs(fa)
```

`screeplot.vsp_fa` *Create a screeplot from a factor analysis object*

Description

Create a screeplot from a factor analysis object

Usage

```
## S3 method for class 'vsp_fa'
screeplot(x, ...)
```

Arguments

`x`	A `vsp_fa()` object.
`...`	Ignored, included only for consistency with S3 generic.

Value

A `tibble::tibble()` with one row for each node, and one column containing each of the requested factor or singular vector, plus an additional `id` column.

`set_z_factor_names` *Give the dimensions of Z factors informative names*

Description

Give the dimensions of Z factors informative names

Usage

```
set_z_factor_names(fa, names)

set_y_factor_names(fa, names)
```

Arguments

fa	A vsp_fa() object.
names	Describe new names for Z/Y factors.

Value

A new vsp_fa() object, but the columns names of Z and the row names of B have been set to names (for set_z_factor_names), and the column names of B and the column names of Y have been set to names (for set_y_factor_names).

Functions

- set_y_factor_names(): Give the dimensions of Y factors informative names

---

vsp *Semi-Parametric Factor Analysis via Vintage Sparse PCA*

Description

This code implements TODO.

Usage

```
vsp(x, rank, ...)

## Default S3 method:
vsp(x, rank, ...)

## S3 method for class 'matrix'
vsp(
  x,
  rank,
  ...,
  center = FALSE,
  recenter = FALSE,
  degree_normalize = TRUE,
  renormalize = FALSE,
  tau_row = NULL,
  tau_col = NULL,
  kaiser_normalize_u = FALSE,
  kaiser_normalize_v = FALSE,
  rownames = NULL,
  colnames = NULL,
  match_columns = TRUE
)
```

```
## S3 method for class 'Matrix'
vsp(
  x,
  rank,
  ...,
  center = FALSE,
  recenter = FALSE,
  degree_normalize = TRUE,
  renormalize = FALSE,
  tau_row = NULL,
  tau_col = NULL,
  kaiser_normalize_u = FALSE,
  kaiser_normalize_v = FALSE,
  rownames = NULL,
  colnames = NULL,
  match_columns = TRUE
)

## S3 method for class 'dgCMatrix'
vsp(
  x,
  rank,
  ...,
  center = FALSE,
  recenter = FALSE,
  degree_normalize = TRUE,
  renormalize = FALSE,
  tau_row = NULL,
  tau_col = NULL,
  kaiser_normalize_u = FALSE,
  kaiser_normalize_v = FALSE,
  rownames = NULL,
  colnames = NULL,
  match_columns = TRUE
)

## S3 method for class 'igraph'
vsp(x, rank, ..., edge_weights = NULL)
```

Arguments

x	Either a graph adjacency matrix, igraph::igraph or tidygraph::tbl_graph. If x is a matrix or Matrix::Matrix then x[i, j] should correspond to the edge going from node i to node j.
rank	The number of factors to calculate.
...	These dots are for future extensions and must be empty.
center	Should the adjacency matrix be row *and* column centered? Defaults to FALSE.

`recenter` Should the varimax factors be re-centered around the original factor means? Only used when `center = TRUE`, defaults to `FALSE`.

`degree_normalize` Should the regularized graph laplacian be used instead of the raw adjacency matrix? Defaults to `TRUE`. If `center = TRUE`, `A` will first be centered and then normalized.

`renormalize` Should the regularized graph laplacian be used instead of the raw adjacency matrix? Defaults to `TRUE`. If `center = TRUE`, `A` will first be centered and then normalized.

`tau_row` Row regularization term. Default is `NULL`, in which case we use the row degree. Ignored when `degree_normalize = FALSE`.

`tau_col` Column regularization term. Default is `NULL`, in which case we use the column degree. Ignored when `degree_normalize = FALSE`.

`kaiser_normalize_u` Whether or not to use Kaiser normalization when rotating the left singular vectors `U`. Defaults to `FALSE`.

`kaiser_normalize_v` Whether or not to use Kaiser normalization when rotating the right singular vectors `V`. Defaults to `FALSE`.

`rownames` Character vector of row names of `x`. These row names are propagated into the row names of the `U` and `Z`. Defaults to `NULL`.

`colnames` Character vector of column names of `x`. These column names are propagated into the row names of the `V` and `Y`. Defaults to `NULL`.

`match_columns` Should the columns of `Y` be re-ordered such that `Y[, i]` corresponds to `Z[, i]` to the extent possible? Defaults to `TRUE`. Typically helps with interpretation, and often makes `B` more diagonally dominant.

`edge_weights` When `x` is an igraph::igraph, an edge attribute to use to form a weighted adjacency matrix.

Details

Sparse SVDs use `RSpectra` for performance.

Value

An object of class `vsp`. TODO: Details

Examples

```
library(LRMF3)

vsp(ml100k, rank = 2)
```

vsp.svd_like *Perform varimax rotation on a low rank matrix factorization*

Description

Perform varimax rotation on a low rank matrix factorization

Usage

```
## S3 method for class 'svd_like'
vsp(
  x,
  rank,
  ...,
  centerer = NULL,
  scaler = NULL,
  recenter = FALSE,
  renormalize = FALSE,
  kaiser_normalize_u = FALSE,
  kaiser_normalize_v = FALSE,
  rownames = NULL,
  colnames = NULL,
  match_columns = TRUE
)
```

Arguments

x	Either a graph adjacency matrix, igraph::igraph or tidygraph::tbl_graph. If x is a matrix or Matrix::Matrix then x[i, j] should correspond to the edge going from node i to node j.
rank	The number of factors to calculate.
...	These dots are for future extensions and must be empty.
centerer	TODO
scaler	TODO
recenter	Should the varimax factors be re-centered around the original factor means? Only used when center = TRUE, defaults to FALSE.
renormalize	Should the regularized graph laplacian be used instead of the raw adjacency matrix? Defaults to TRUE. If center = TRUE, A will first be centered and then normalized.
kaiser_normalize_u	Whether or not to use Kaiser normalization when rotating the left singular vectors U. Defaults to FALSE.
kaiser_normalize_v	Whether or not to use Kaiser normalization when rotating the right singular vectors V. Defaults to FALSE.

rownames	Character vector of row names of x. These row names are propagated into the row names of the U and Z. Defaults to NULL.
colnames	Character vector of column names of x. These column names are propagated into the row names of the V and Y. Defaults to NULL.
match_columns	Should the columns of Y be re-ordered such that Y[, i] corresponds to Z[, i] to the extent possible? Defaults to TRUE. Typically helps with interpretation, and often makes B more diagonally dominant.

Examples

```
library(LRMF3)
library(RSpectra)

s <- svds(ml100k, k = 2)
mf <- as_svd_like(s)
fa <- vsp(mf, rank = 2)
```

---

vsp_fa *Create a vintage sparse factor analysis object*

Description

vsp_fa objects are a subclass of LRMF3::fa_like(), with additional fields u, d, v, transformers, R_U, and R_V

Usage

```
vsp_fa(
  u,
  d,
  v,
  Z,
  B,
  Y,
  transformers,
  R_U,
  R_V,
  rownames = NULL,
  colnames = NULL
)
```

Arguments

u	A matrix() of "left singular-ish" vectors.
d	A numeric() vector of "singular-ish" values.
v	A matrix() of "right singular-ish" vectors.

Z	A *matrix* of embeddings for each observation.
B	A mixing *matrix* describing how observation embeddings and topics interact. Does not have to be diagonal!
Y	A *matrix* describing the compositions of various topics or factors.
transformers	A list of transformations from the invertiforms package.
R_U	Varimax rotation matrix use to transform u into Z.
R_V	Varimax rotation matrix use to transform v into Y.
rownames	Identifying names for each row of the original data. Defaults to NULL, in which cases each row is given a row number left-padded with zeros as a name.
colnames	Identifying names for each column of the original data. Defaults to NULL, in which cases each column is given a row column left-padded with zeros as a name.

Value

A svd_fa object.

Index