# Package 'VertexSort'

October 12, 2022

**Type** Package

**Title** Network Hierarchical Structure and Randomization

**Version** 0.1-1

**Date** 2017-07-03

**Author** Diala Abd-Rabbo

**Maintainer** Diala Abd-Rabbo <diala.abd.rabbo@gmail.com>

**Description** Permits to apply the 'Vertex Sort' algorithm (Jothi et al. (2009) <10.1038/msb.2009.52>) to a graph in order to elucidate its hierarchical structure. It also allows graphic visualization of the sorted graph by exporting the results to a cytoscape friendly format. Moreover, it offers five different algorithms of graph randomization: 1) Randomize a graph with preserving node degrees, 2) with preserving similar node degrees, 3) without preserving node degrees, 4) with preserving node in-degrees and 5) with preserving node out-degrees.

**Depends** R (>= 3.3.2), igraph, snowfall

**License** GPL (>= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-03-09 08:34:23

## R topics documented:

`VertexSort-package` *The Vertex Sort Package*

## Description

The present package is used to elucidate the hierarchical structure of a directed network using the Vertex Sort algorithm (Jothi et al., 2009). This package includes also functions for randomizing a graph in five different ways.

## Details

| | |
|---|---|
| Package: | VertexSort |
| Type: | Package |
| Version: | 0.1-1 |
| Date: | 2017-03-07 |
| License: | GPL (>= 2) |

## Author(s)

Diala Abd-Rabbo Maintainer: Diala Abd-Rabbo <diala.abd.rabbo@gmail.com>

## References

Jothi, R., Balaj, S., Wuster, A. et al. 2009 *Molecular system biology* **5**, –294-309.

`dpr, sdpr, dnpr, idpr, odpr`
*Functions for Network Randomization using different algorithms*

## Description

These functions can be run in the parallel mode to reduce the time of excution. They randomize a network according to the following five algorithms:

1. degree preserving randomization (dpr).
2. similar degree preserving randomization (sdpr).
3. degree non-preserving randomization (dnpr).

4. in-degree preserving randomization (idpr).

5. out-degree preserving randomization (odpr).

## Usage

```
dpr(vgraph, viteration_no, vparallel = FALSE, vcpus = 1)
sdpr(vgraph, viteration_no, vparallel = FALSE, vcpus = 1)
dnpr(vgraph, viteration_no, vparallel = FALSE, vcpus = 1)
idpr(vgraph, viteration_no, vparallel = FALSE, vcpus = 1)
odpr(vgraph, viteration_no, vparallel = FALSE, vcpus = 1)
```

## Arguments

| | |
|---|---|
| vgraph | the graph to be randomized. The graph should be of class igraph. |
| viteration_no | an integer scalar indicating the desired number of random graphs to be generated. |
| vparallel | a logical scalar indicating whether to use the parallel programming feature to reduce the process time. The default value is FALSE. |
| vcpus | an integer scalar determining the number of CPUs to be used when parallel is TRUE. The default value is 1. |

## Details

The dpr() function randomizes the input graph by randomly selecting two of its edges and exchanging their ends. Multiple edges having the same direction between two nodes are then removed by switching each of them with randomly selected edges. These steps are repeated 10 times the number of edges of the input graph. For more information see references (Abd-Rabbo et al. 2017).

The sdpr() function randomizes the input graph using the matching algorithm, see references (Milo et al. 2004) and examples.

The dnpr() function randomly selects two nodes with replacement to create edges.

The idpr() function randomizes the "actor_id" column of the edges data frame by randomly selecting nodes with replacement from this column. The edges data frame contains the edges making the network and contains two columns: "actor_id" and "target_id".

The odpr() function randomizes the "target_id" column of the edges data frame by randomly selecting nodes with replacement from this column. The edges data frame contains the edges making the network and contains two columns: "actor_id" and "target_id".

A log file having the name of each of these functions (e.g. dpr_log.txt, idpr_log.txt) will be created in the working directory and could be opened in order to follow the progression of long randomizations. This log file will be deleted at the end of the task.

## Value

| | |
|---|---|
| graph | list of the randomized graphs of class igraph. |

## Author(s)

Diala Abd-Rabbo <diala.abd.rabbo@gmail.com>

## References

Abd-Rabbo, D. and Michnick, S.W. 2017 *BMC Syst Biol* **11**.

Milo, R., Kashtan, N., Itzkovitz, S. 2004 *cond-mat.stat-mech* arXiv:cond-mat/0312028v2.

## See Also

igraph, snowfall.

## Examples

```
## generate a random graph of the kinase-phosphatase network by using
## each of the five algorithms

## load the VertexSort library
library(VertexSort)

## load interactions of the kinase-phosphatase network (kp-net)
data("interactions")
vs_kp_net <- vertex.sort(interactions)
kp_net    <- vs_kp_net$graph

## dpr function: randomize a network with preserving its node degrees
## notice the difference in execution time when using and
## not using the parallel programning mode
ptm <- proc.time()
rand_g <- dpr(kp_net, 4)            # without parallel mode
proc.time() - ptm

ptm <- proc.time()
rand_g <- dpr(kp_net, 4, TRUE, 4)   # with parallel mode
proc.time() - ptm

## verify that rand_g have the same in- and out-degrees as those
## of kp_net should obtain TRUE in both commands.
all(degree(kp_net, V(kp_net), "in")==degree(rand_g[[1]], V(rand_g[[1]]), "in"))
all(degree(kp_net, V(kp_net), "out")==degree(rand_g[[1]], V(rand_g[[1]]), "out"))

## sdpr function: randomize a network with preserving similar node degrees
rand_g <- sdpr(kp_net, 1)
## verify that rand_g have similar in- and out-degrees to those of
## kp_net. Should be -1, 0 or 1
sort(unique(degree(kp_net, V(kp_net), "in")-degree(rand_g[[1]], V(rand_g[[1]]), "in")))
sort(unique(degree(kp_net, V(kp_net), "out")-degree(rand_g[[1]], V(rand_g[[1]]), "out")))

## dnpr function: randomize a network without preserving its node degrees
rand_g <- dnpr(kp_net, 1)
## verify that rand_g have different in- and out-degrees of those of
```

```
## kp_net. Should get FALSE in both commands
all(degree(kp_net, V(kp_net), "in")==degree(rand_g[[1]], V(rand_g[[1]]), "in"))
all(degree(kp_net, V(kp_net), "out")==degree(rand_g[[1]], V(rand_g[[1]]), "out"))

## idpr function: randomize a network with preserving its node in-degrees
rand_g <- idpr(kp_net, 1)
## verify that rand_g have same in-degrees and different out-degrees as
## those of kp_net. Should get TRUE and FALSE respectively.
all(degree(kp_net, V(kp_net), "in")==degree(rand_g[[1]], V(rand_g[[1]]), "in"))
all(degree(kp_net, V(kp_net), "out")==degree(rand_g[[1]], V(rand_g[[1]]), "out"))

## odpr function: randomize a network with preserving its node out-degrees
rand_g <- odpr(kp_net, 1)
## verify that rand_g have same out-degrees and different in-degrees as
## those of kp_net. Should get FALSE and TRUE respectively.
all(degree(kp_net, V(kp_net), "in")==degree(rand_g[[1]], V(rand_g[[1]]), "in"))
all(degree(kp_net, V(kp_net), "out")==degree(rand_g[[1]], V(rand_g[[1]]), "out"))
```

---

`export.to.cytoscape` *Facilitates export of results of the Vertex Sort algorithm to Cytoscape*

---

## Description

The `export.to.cytoscape()` function facilitates export of a graph that was sorted by the vertex sort algorithm to the Cytoscape. It creates a list of two data frames. The first data frame contains the edges of the graph and the second data frame contains node attributes of the graph. These data frames could be written to text files and could be imported to cytoscape to allow to query certain details of the results (e.g. layer of a node) and to also permit the visualization and customization of the graphic representation of the sorted graph in order to generate figures.

## Usage

```
export.to.cytoscape(vs_object)
```

## Arguments

| | |
|---|---|
| `vs_object` | a vertex.sort object generated by applying the `vertex.sort()` function. |

## Value

The `export.to.cytoscape()` function returns a list of two data frames:

| | |
|---|---|
| `edges` | a data frame that contains the edges of a graph that has been sorted using the Vertex Sort algorithm by the `vetex.sort()` function. |
| `node_attribute` | a data frame that contains the node attributes of a graph that has been sorted using the Vertex Sort algorithm by the `vetex.sort()` function. It includes the following columns: node identifier, node type, node layer and node level. For more details, see `vertex.sort()` function |

## Author(s)

Diala Abd-Rabbo <diala.abd.rabbo@gmail.com>

## References

Jothi, R., Balaj, S., Wuster, A. et al. 2009 *Molecular system biology* **5**, –294-309.

Abd-Rabbo, D. and Michnick, S.W. 2017 *BMC Syst Biol* **11**.

## See Also

vertex.sort.

## Examples

```
## load the VertexSort library
library(VertexSort)

## load interactions of the kinase-phosphatase network (kp-net)
data("interactions")

## apply the vertex sort algorithm
vs_kp_net <- vertex.sort(interactions)

## apply the export.to.cytoscape function
df <- export.to.cytoscape(vs_kp_net)

## view the first 6 lines of each data frame
head(df$edges)
head(df$node_attribute)
```

---

# interactions *Dataset containging interactions of the Kinase-Phosphatase Network (Abd-Rabbo et al. 2017)*

---

## Description

This dataset contains a data frame called interactions containing the binary interactions assembled in the kinase-phosphatase network. Interactions were obtained from the Kinase Interaction Database (Sharifpoor et al.) and from other data curation efforts (Abd-Rabbo et al. 2017).

## Usage

```
data("interactions")
```

## Format

This data frame contains 1087 rows and the following columns:

`kp_orf` standard name (ORF) of the kinase/phosphatase that phosphorylates/dephosphorylates the substrate

`substrate_orf` standard name (ORF) of the protein substrate phosphorylated/dephosphorylated by the kinase/phosphatase

## Details

These interactions represent binary interactions between kinases/phosphatases and their substrates. They were used in (Abd-Rabbo et al. 2017) to form the kinase-phosphatase network used with the Vertex Sort algorithm to study the hierarchical structure of the signaling regulatory network in the budding yeast.

## Source

Abd-Rabbo, D. and Michnick, S.W. 2017 *BMC Syst Biol* **11**.

## References

Sharifpoor, S., Nguyen, Ba. A.N., Young, J.Y. et al. 2011 *Genome Biol* **12**, R39.

## Examples

```
## load the VertexSort library
library(VertexSort)

## load interactions of the kinase-phosphatase network (kp-net)
data(interactions)

## view the dimentions of the interactions data frame
dim(interactions)

## view the top of the interactions data frame
head(interactions)

## The interactions data frame could be loaded into an R session
## and used as an example to apply on it the vertex sort algorithm
## or generate radom networks. See ?vertex.sort, ?dpr, ?sdpr, ?dnpr,
## ?idpr and ?odpr
```

---

`vertex.sort` *Elucidates the hierarchical structure of a directed network using the Vertex Sort algorithm*

---

## Description

The `vertex.sort()` function applies a network decomposition algorithm called the Vertex Sort to a directed network (Jothi et al. 2009) and generates an object of class vertex.sort. The Vertex Sort algorithm elucidates the hierarchical structure of a directed network by classifying its nodes into mainly four layers: the Top, Core, Bottom and Zero layers. For more information on each of these layers see details below or refer to (Jothi et al. 2009).

## Usage

```
  vertex.sort(edges)
  ## S3 method for class 'vertex.sort'
print(x, ...)
  ## S3 method for class 'vertex.sort'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `edges` | a dataframe that contains a symbolic edge list of the graph to decompose. It should contain two columns listing Actor and Target nodes, see details below. |
| `x, object` | the vertex.sort object of wich details or summary will be printed. |
| `...` | further arguments passed to or from other methods. |

## Details

The Vertex Sort algorithm is a network decomposition method that elucidates the network hierarchy (Jothi et al. 2009). It classifies network nodes into various levels. First, it transforms the directed network into an acyclic directed graph. Second, it applies the leaf removal (LR) algorithm on the network and on its transpose. Third, it merges the results of the two LR applications into a global result, in which the level of each node is determined by the possibility of a node to span many levels. Finally, it groups nodes into four non-overlapping layers: the Top, Core, Bottom and Zero layers. The Core layer is made of the nodes composing the biggest strongly connected component of the network and Top and Bottom layers contain nodes that regulate and are regulated by the Core layer respectively. Transforming a network from a cyclic to an acyclic graph is achieved by collapsing each strongly connected component into a super-node.

The `print()` function prints the details of a vertex.sort object (i.e. all the elements of the vertex.sort object).

The `summary()` function prints a summary of a vertex.sort object. It describes: the sorted graph (i.e. number of nodes, edges, layers and levels), the graph layers (i.e. number of nodes of type Actors in Top, Core and Bottom layers and the number of nodes of type Targets), and the number of excluded and disconnected nodes (i.e. number of excluded Actor nodes and the number of disconnected Actor and Target nodes).

## Value

The `vertex.sort()` function returns an object of class vertex.sort. The functions `print()` and `summary()` are used to obtain the details and a summary of the contents of the vertex.sort objet.

An object of class vertex.sort is a list containing the following elements:

| | |
|---|---|
| graph | a graph of class igraph containing the edges of the input network to be sorted. |
| edges | a dataframe containing the network edges |
| traits | a dataframe listing the network nodes and determining whether each of them is of type Actor (node with a non-zero out-degree) or of type Target (a node with a zero out-degree or a node that does not regulate an Actor node). |
| actors | a vector containing all Actor nodes. |
| targets | a vector containing all Target nodes. |
| top.actors | a vector containing Actor nodes that were classified in the Top layer. |
| core.actors | a vector containing Actor nodes that were classified in the Core layer. |
| bottom.actors | a vector containing Actor nodes that were classified in the Bottom layer. |
| excluded.actors | a vector containing Actor nodes that were excluded from the classification, because the algorithm was unable to classify them propertly. |
| disconnected.actors | a vector containing Actor nodes that were found to be disconnected from the input graph and hence were not subject to the Vertex Sort algorithm. |
| disconnected.targets | a vector containing Target nodes that were found to be disconnected from the input graph and hence were not subject to the decomposition algorithm. |
| levels.no | an integer indicating the number of levels of the sorted network. |
| nodes.in.levels | a list enumerating the nodes belonging to each level. |

## Note

In the article (Jothi et al. 2009), the authors talked about only three layers (the Top, Core and Bottom layers). However, in the present package we talk about four layers, the three layers mentioned by Jothi et al. in addition to the Zero layer corresponding to the layer containg the target nodes that do not regulate other Actor nodes.

## Author(s)

Diala Abd-Rabbo <diala.abd.rabbo@gmail.com>

## References

Jothi, R., Balaj, S., Wuster, A. et al. 2009 *Molecular system biology* **5**, –294-309.

## Examples

```
## load the VertexSort library
library(VertexSort)

## load interactions of the kinase-phosphatase network (kp-net)
data("interactions")

## apply the Vertex Sort algorithm
```

```
vs_kp_net <- vertex.sort(interactions)

## print the results (the contents of the vertex.sort object)
vs_kp_net

## print a summary of the results
summary(vs_kp_net)

## print levels.no, an element of the vertex.sort object
vs_kp_net$levels.no
```

# Index