Package ‘CurricularAnalytics’

May 29, 2024

Title Exploring and Analyzing Academic Curricula

Version 1.0.0

Description Provides an implementation of ‘Curricular Analytics’, a framework for analyzing and quantifying the complexity of academic curricula. Curricula are modelled as directed acyclic graphs and analytics are provided based on path lengths and edge density. This work directly comes from Heileman et al. (2018) <doi:10.48550/arXiv.1811.09676>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

URL `https://github.com/Danyulll/CurricularAnalytics`

BugReports `https://github.com/Danyulll/CurricularAnalytics/issues`

Imports dplyr (>= 1.1.4), igraph (>= 2.0.3), stats (>= 4.4.0), tools (>= 4.4.0), utils (>= 4.4.0), visNetwork (>= 2.1.2)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Krasnov [aut, cre] (<`https://orcid.org/0009-0003-4104-3162`>), Dr. Irene Vrbik [aut, cph]

Maintainer Daniel Krasnov <`danielkrasnovdk@hotmail.com`>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-05-29 10:40:08 UTC

R topics documented:

`blocking_factor` *Calculate blocking factor*

Description

A helper function for calculating the blocking factor for each node and the total blocking factor of a curriculum graph.

Usage

```
blocking_factor(node_list, edge_list)
```

Arguments

`node_list`	Dataframe with an 'id' column for each node and a 'term' column specifying which term the course is to be taken in.
`edge_list`	Dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes. Entries must use node ids from `node_list`.

Details

Blocking quantifies when a failing a course would result in being blocked from registering for future courses. More formally the blocking factor of a node v_i is defined as

$$b_c(v_i) = \sum_{v_j \in V} I(v_i, v_j)$$

where I is the indicator function:

$$= \begin{cases} 1, & \text{if } v_i \to v_j \\ 0, & \text{if } v_i \not\to v_j \end{cases}$$

The blocking factor for an entire curriculum graph G_c is defined as

$$b(G_c) = \sum_{v_i \in V} b_c(v_i)$$

Value

A list that contains the following:

`bynode`	A dataframe containing the blocking factor of each node
`total`	The total blocking factor of the curriculum graph

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. "Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations." arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
node_list <-
data.frame(
id = 1:4,
label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
term = c(1, 1, 2, 2)
)
bf_list <- blocking_factor(node_list,edge_list)
print(bf_list)
# Output:
# $bynode
# id bf
# 2  1  2
# 3  2  0
# 4  3  1
# 5  4  0
# $total
# [1] 3
```

---

centrality_factor *Calculate centrality*

Description

A helper function for calculating the centrality for each node.

Usage

```
centrality_factor(node_list, edge_list)
```

Arguments

`node_list`	Dataframe with an 'id' column for each node and a 'term' column specifying which term the course is to be taken in.
`edge_list`	Dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes. Entries must use node ids from `node_list`.

Details

A course is considered central if it has many requisite edges flowing in and out of the node. More formally it is the number of long paths that include the node. That is, consider a curriculum graph G_c and a vertex v_i. A long path is a path that satisfies the following conditions:

- v_i, v_j, v_k are distinct
- $v_j \to v_i \to v_k$
- v_j is a source node (in-degree zero)
- v_k is a sink node (out-degree zero)

Let $P_{v_i} = \{p_1, p_2, \ldots\}$ denote the set of all paths defined as above. Then the centrality of a node v_i is given by

$$q(v_i) = \sum_{l=1}^{|P_{v_i}|} \#(p_l)$$

More plainly this is the number of paths containing v_i of at least length 3 where v_i is neither a source nor sink node.

Value

A dataframe containing the centrality of each node

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. “Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations.” arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
node_list <-
data.frame(
 id = 1:4,
 label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
 term = c(1, 1, 2, 2)
)

cf_df <- centrality_factor(node_list,edge_list)
print(cf_df)
# Output:
#   id cf
#1  1  0
#2  2  0
#3  3  3
#4  4  0
```

curriculum_graph_from_csv *Create Curriculum From CSV File*

Description

Generates a curriculum graph from a csv file.

Usage

```
curriculum_graph_from_csv(filepath)
```

Arguments

`filepath` A csv file path with a table where each row is a course and the columns are as follows:

- id: an integer id for the course
- label: a string with the name of the course
- term: an integer specifying what term the course is to be taken
- requisites: a list of all pre- and co-requisite course ids of the form 1;2;3;...

Value

A list that contains the following:

`node_list` A dataframe of course nodes containing their id, term, blocking factor (bf), delay factor (df), centrality (cf), and cruciality (sc)

`edge_list` A dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes.

`network` Igraph network object representing the curriculum graph

`sc_total` Total structural complexity of the curriculum graph

`bf_total` Total blocking factor of the curriculum graph

`df_total` Total delay factor of the curriculum graph

Examples

```
# Have filepath point to a csv of the following form
#id label term requisites
#1 MATH 100 1
#2 DATA 101 1
#3 MATH 101 2 1
#4 MATH 221 2 3
#5 STAT 230 3 3;2
filepath <-
system.file("extdata", "Example-Curriculum.csv", package = "CurricularAnalytics")
C <- curriculum_graph_from_csv(filepath)
plot_curriculum_graph(C)
```

curriculum_graph_from_list

Create Curriculum Graph Object

Description

Generates a curriculum graph from a node and edge list.

Usage

```
curriculum_graph_from_list(node_list, edge_list)
```

Arguments

`node_list`	Dataframe with an 'id' column for each node and a 'term' column specifying which term the course is to be taken in.
`edge_list`	Dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes. Entries must use node ids from `node_list`.

Value

A list that contains the following:

`node_list`	A dataframe of course nodes containing their id, term, blocking factor (bf), delay factor (df), centrality (cf), and cruciality (sc)
`edge_list`	A dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes.
`network`	Igraph network object representing the curriculum graph
`sc_total`	Total structural complexity of the curriculum graph
`bf_total`	Total blocking factor of the curriculum graph
`df_total`	Total delay factor of the curriculum graph

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. "Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations." arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
# courses in node list must be placed sequentially in term order to be properly displayed
node_list <-
data.frame(
 id = 1:4,
 label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
 term = c(1, 1, 2, 2)
)
C <- curriculum_graph_from_list(node_list,edge_list)
plot_curriculum_graph(C)
```

`delay_factor` *Calculate delay factor*

Description

A helper function for calculating the delay factor for each node and the total delay factor of a curriculum graph.

Usage

```
delay_factor(node_list, edge_list)
```

Arguments

`node_list`	Dataframe with an 'id' column for each node and a 'term' column specifying which term the course is to be taken in.
`edge_list`	Dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes. Entries must use node ids from `node_list`.

Details

The delay factor of a course is the longest path the nodes finds itself on. More formally the delay factor of a node v_k is given by

$$d_c(v_k) = \underset{i,j,l,m}{max} \left\{ \# \left(v_i \overset{p_l}{\to} v_k \overset{p_m}{\to} v_j \right) \right\}$$

The delay factor of an entire curriculum graph G_c is defined as

$$d(G_c) = \sum_{v_k \in V} d_c(v_k)$$

Value

A list that contains the following:

`bynode`	A dataframe containing the delay factor of each node
`total`	The total delay factor of the curriculum graph

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. "Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations." arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
node_list <-
data.frame(
 id = 1:4,
 label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
 term = c(1, 1, 2, 2)
)

df_list <- delay_factor(node_list,edge_list)
print(df_list)
# Output:
# $bynode
#   id df
# 2  1  3
# 3  2  1
# 4  3  3
# 5  4  3
# $total
# [1] 10
```

---

plot_curriculum_graph *Plot a curriculum graph*

Description

Plots an interactable vizNetwork visualization of the Igraph network object representing the curriculum graph.

Usage

```
plot_curriculum_graph(curriculum_graph, width = "100%", height = 500)
```

Arguments

curriculum_graph	A curriculum_graph object created with either `curriculum_graph_from_list()` or `curriculum_graph_from_csv()`
width	A string percentage for the width of the plot, default is "100%".
height	An integer representing the number of pixels for the height, default is 500.

Value

No object is returned. Rather the graph is plotted according to the specified term order in node_list. Clicking on a node will reveal its label, structural complexity (sc), centrality (cf), blocking factor (bf), and delay factor (df)

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. “Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations.” arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
node_list <-
data.frame(
 id = 1:4,
 label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
 term = c(1, 1, 2, 2)
)
C <- curriculum_graph_from_list(node_list,edge_list)
plot_curriculum_graph(C)
```

---

structural_complexity   *Calculate structural complexity*

Description

A helper function for calculating the structural complexity for each node and the total structural complexity of a curriculum graph.

Usage

```
structural_complexity(node_list, edge_list)
```

Arguments

`node_list`	Dataframe with an 'id' column for each node and a 'term' column specifying which term the course is to be taken in.
`edge_list`	Dataframe with two columns 'from' and 'to' specifying directed edges starting at 'from' nodes directed towards 'to' nodes. Entries must use node ids from `node_list`.

Details

The structural complexity of a node v_k is defined as a linear combination of the node's delay and blocking factors. More formally

$$h(v_k) = d(v_k) + b(v_k)$$

. The structural complexity of an entire curriculum graph G_c is defined as

$$h(G_c) = d(G_c) + b(G_c) = \sum_{v_k \in V} (d_c(v_k) + b_c(v_k))$$

Value

A list that contains the following:

`bynode`	A dataframe containing the structural complexity of each node
`total`	The total structural complexity of the curriculum graph

Author(s)

Daniel Krasnov

References

Heileman, Gregory L, Chaouki T Abdallah, Ahmad Slim, and Michael Hickman. 2018. "Curricular Analytics: A Framework for Quantifying the Impact of Curricular Reforms and Pedagogical Innovations." arXiv Preprint arXiv:1811.09676.

Examples

```
edge_list <- data.frame(from = c(1, 3), to = c(3, 4))
node_list <-
data.frame(
 id = 1:4,
 label = c("MATH 100", "DATA 101", "MATH 101", "MATH 221"),
 term = c(1, 1, 2, 2)
)
sc_list <- structural_complexity(node_list,edge_list)
print(sc_list)
# Output:
# $bynode
#   id sc
# 1  1  5
# 2  2  1
# 3  3  4
# 4  4  3
# $total
# [1] 13
```

Index