Package 'CurricularComplexity'

November 4, 2025

Type Package

Title Toolkit for Analyzing Curricular Complexity

Version 1.0

Author David Reeping [aut, cre]

Maintainer David Reeping <reepindp@ucmail.uc.edu>

Description Enables educational researchers and practitioners to calculate the curricular complexity of a plan of study, visualize its prerequisite structure at scale, and conduct customizable analyses. The original tool can be found at <https://curricularanalytics.org>. Additional functions to explore curriculum complexity from the literature are also included.

Depends R (>= 3.6.2), igraph (>= 1.2.5)

Encoding UTF-8

License MIT + file LICENSE

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-11-04 19:10:14 UTC

Contents

`admissibility_test` *Automatically check for data entry issues*

Description

This function takes in a plan of study and a course, then checks for potential data entry issues. It will detect issues in formatting with the csv (such as notes creating empty rows), if there are cycles in the network, and if pre- and corequisites are appropriately defined.

Usage

```
admissibility_test(plan_of_study)
```

Arguments

`plan_of_study`	igraph object - An igraph object created using the create_plan_of_study function

Value

List of errors to correct for cycles, prereqs, and coreqs

`average_sequencing` *Calculates the average sequencing in a program*

Description

This function calculates the average sequencing in the program using the delay factors of the courses. The second argument, expected_time_to_degree is optional. If it is not NULL, the average sequencing will be for courses extending the student's time to degree.

Usage

```
average_sequencing(plan_of_study, expected_time_to_degree = NULL)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
expected_time_to_degree	Numeric - The term where students are expected to finish (often 8)

Value

Numeric - the average sequencing in the program

---

blocking_factor *Calculates the blocking factor of a course*

Description

This function takes in a plan of study and a course, then finds that course's blocking factor. The value is the number of courses 'blocked' by failing the given course.

Usage

```
blocking_factor(plan_of_study, course, include_coreqs = TRUE)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
course	Numeric (vertex id) or String - The course to calculate the blocking factor of
include_coreqs	logical - Indicates whether corequisites should be included in the calculation

Value

Numeric - the blocking factor

core_collapse *Calculates the core collapse sequence for a plan of study*

Description

This function takes in a plan of study network and constructs the "core collapse sequence." The core collapse sequence progressively removes courses from the plan of study with increasing prereq counts and calculates the proportion of courses deleted at each step. The process stops when all of the vertices have been removed. A sequence that decreases quickly to zero typically indicates that the network is generally uniform with its prereqs. A sequence with more erratic values that does not settle to zero smoothly would imply more dense sets of prereqs.

Usage

```
core_collapse(plan_of_study)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function

Value

List of two items: (1) sequence - the core collapse sequence, (2) the associated network for each entry

create_plan_of_study *Create a plan of study igraph object*

Description

This function takes in a set of courses, their terms, prerequisites, and corequisites. Optional arguments include the number of credits, pass rates, lost credits from transferring, and the frequency of course offerings. The function creates an igraph structure of edges and nodes with the given qualities.

Usage

```
create_plan_of_study(
  Course,
  Term,
  Prereq,
  Coreq,
  Credits = NULL,
  LostCredits = NULL,
  PassRate = NULL,
  Timing = NULL,
  Institution = NULL
)
```

Arguments

Course	atomic vector - strings for each course
Term	a numeric atomic vector - the term each course is offered
Prereq	atomic vector - strings of the courses' prereqs, separated by commas
Coreq	atomic vector - strings of the courses' coreqs, separated by commas
Credits	numeric atomic vector - number of credits each course is worth (optional)
LostCredits	numeric atomic vector - (for transfer students) identifies if credit for the course is not applied toward a student's degree, 1. If it is, 0. (optional)
PassRate	numeric atomic vector - pass rates by class (optional)
Timing	numeric atomic vector - number of times the course is offered in 2 years (optional)
Institution	atomic vector - strings of course affiliations (CC or FY)

Details

It is recommended that the user imports the data from a csv file to ensure the indices for each atomic vector correspond to the attributes of one course.

Value

An igraph object of the prerequisite structure

---

cruciality *Calculates the cruciality of a course*

Description

This function takes in a plan of study and a course, then finds that course's cruciality. The value is the sum of the blocking and delay factors of the course

Usage

```
cruciality(plan_of_study, course, include_coreqs = TRUE)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
course	Numeric (vertex id) or String - The course to calculate the cruciality of
include_coreqs	logical - Indicates whether corequisites should be included in the calculation

Value

Numeric - the course's cruciality

curriculum_rigidity *Calculates the curriculum rigidity*

Description

This function takes in a plan of study and then finds the curriculum's rigidity. The rigidity is the beta index of the graph, which is the number of prerequisites divided by the number of courses

Usage

```
curriculum_rigidity(plan_of_study)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function

Value

Numeric - the curriculum rigidity

deferment_factor *Calculates the deferment factor of a course*

Description

This function takes in a plan of study and a course, then finds that course's deferment factor. The value captures the number of terms the student can fail the course before extending their time to degree.

Usage

```
deferment_factor(plan_of_study, course, expected_time_to_degree)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
course	Numeric (vertex id) or String - The course to calculate the deferment factor of
expected_time_to_degree	Numeric - The term where students are expected to finish (often 8)

Value

Numeric - the deferment factor

delay_factor *Calculates the delay factor of a course*

Description

This function takes in a plan of study and a course, then finds that course's delay factor. The output is the longest path of prerequisites through the given course.

Usage

```
delay_factor(plan_of_study, course, include_coreqs = TRUE)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
course	Numeric (vertex id) or String - The course to calculate the delay factor of
include_coreqs	Logical - Calculates the delay factor using corequisites, default value is TRUE

Value

Numeric - the delay factor

explained_complexity *Calculates the explained complexity of courses extending time to degree*

Description

This function takes in the subcomplexity graph from the transfer excess courses function, then finds the transfer delay factor. The output is the proportion of complexity explained by courses extending time to degree.

Usage

```
explained_complexity(
  plan_of_study,
  expected_time_to_degree,
  term_weighted = FALSE
)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
expected_time_to_degree	Numeric - The term where students are expected to finish (often 8)
term_weighted	logical - TRUE if crucialities should be term-weighted

Value

Numeric - the explained complexity

`find_bottlenecks` *Finds the bottlenecks in the plan of study based on prerequisite relationships*

Description

This function takes in a plan of study and three parameters. In this case, we choose min_prereq,min_postreq, and min_connections. The value of min_prereq is the minimum number of prerequisites defining a bottleneck (in the user's perspective), whereas min_postreq is the minimum number of courses the given course is a prerequisite for. Finally, min_connections is the minimum total of the number of prerequisites and the number of courses the given course is a prerequisite for. A course is a bottleneck if it meets at least one of the parameters

Usage

```
find_bottlenecks(
  plan_of_study,
  min_prereq = 3,
  min_postreq = 3,
  min_connections = 5,
  include_coreqs = TRUE
)
```

Arguments

`plan_of_study`	igraph object - An igraph object created using the create_plan_of_study function
`min_prereq`	numeric - minimum number of prerequisites defining a bottleneck
`min_postreq`	numeric - minimum number of courses the given course is a prerequisite for
`min_connections`	numeric - minimum total of the number of prerequisites
`include_coreqs`	boolean - default is TRUE, treats corequisites as prerequisites and the number of courses the given course is a prerequisite for

Details

Suggested values for typical usage is find_bottleneck(x,3,3,5), which are #' provided by default. Note that min_connections >= min_prereq + min_postreq - 2. If this is violated, a warning is provided and corrected to the suggested minimum value of min_prereq + min_postreq - 2.

The output is an atomic vector of possible bottlenecks based on the user-defined parameters.

Value

atomic vector - list of courses meeting at least one condition of the three parameters

find_inbound_courses *Find all possible prerequisites to a course*

Description

This function takes in a plan of study and a course, then finds all the courses it is related to through its prerequisites

Usage

```
find_inbound_courses(plan_of_study, course)
```

Arguments

plan_of_study	An igraph object created using the create_plan_of_study function
course	The course to find all relevant prerequisites of

Value

An atomic vector of vertex ids for the course's prerequisites

find_outbound_courses *Find all possible courses that depend on a particular course*

Description

This function takes in a plan of study and a course, then finds all the courses it is related to through its prerequisites (after the course).

Usage

```
find_outbound_courses(plan_of_study, course)
```

Arguments

plan_of_study	An igraph object created using the create_plan_of_study function
course	The course to find all relevant courses that directly or indirectly have it as a prereq

Value

An atomic vector of vertex ids for the course's following courses

inflexibility_factor *Calculates inflexibility factor of a plan of study*

Description

Calculates the inflexibility factor for courses that have specific offering times extending chains beyond the expected time to degree.

Usage

```
inflexibility_factor(plan_of_study, time_to_degree)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
time_to_degree	numeric - expected time to degree, often 8

Value

list of (1) a dataframe of inflexibility factors and (2) a total inflexibility factor

plot_plan_of_study *Plots the plan of study with courses ordered by term*

Description

This function takes in a plan of study and plots it in the 'plot' window. The courses are ordered horizontally by term and vertically by the outdegree (i.e., number of prereqs) of the vertices in that column/term. The shading of the nodes corresponds to the cruciality of the course. A darker blue indicates higher cruciality while white indicates lower cruciality.

Usage

```
plot_plan_of_study(plan_of_study)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function

Details

Note that there can be some overlap where a course is covering a path for a prereq, which may make is seem like a course is a prereq for some other course when it is in fact the course in a previous semester.

Value

Plots the plan of study in the 'plot' window

reachability_factor *Calculates the reachability factor of a course*

Description

This function takes in a plan of study and a course, then finds that course's reachability factor. The value is the number of courses needed to be passed before enrolling in the given course.

Usage

```
reachability_factor(plan_of_study, course, include_coreqs = TRUE)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
course	Numeric (vertex id) or String - The course to calculate the blocking factor of
include_coreqs	logical - Indicates whether corequisites should be included in the calculation

Value

Numeric - the reachability factor

simplify_requisites *Convert requisites to original notation*

Description

This function takes in either the pre or corequisites of a plan of study as a vector, then removes any additional information like OR relationships and minimum grades such that the network can be analyzed using the traditional functions.

Usage

```
simplify_requisites(requisites)
```

Arguments

requisites	vector object - A vector describing the pre and corequisites (as strings)

Value

vector object - A simplified vector describing the pre and corequisites (as strings)

structural_complexity *Calculates structural complexity of a plan of study*

Description

This function takes in a plan of study, then finds the plan of study's structural complexity.

Usage

```
structural_complexity(
  plan_of_study,
  term_weighted = FALSE,
  include_coreqs = TRUE,
  quarters = FALSE
)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
term_weighted	logical - TRUE if crucialities should be term-weighted
include_coreqs	logical - TRUE if coreqs should be included when calculating blocking and delay factor
quarters	logical - TRUE if the plan of study uses quarters instead of semesters

Value

list of (1) a dataframe of course crucialities, delay factors, and blocking factors; (2) a numeric value of structural complexity

student_mobility_turbulence *Calculates the student mobility turbulence for a program*

Description

This metric captures the volatility in student progression by analyzing the withdraws and major changes, which can indicate structural barriers or inefficiencies in the curriculum. This is most useful to apply to a combination of programs or a unit, like a department or college. There are two coefficients, withdrawn and changed major that can be used to prioritize either of the two causes for turbulence. They are set to 1 and 0.5 by default, respectively.

Usage

```
student_mobility_turbulence(
  number_withrawn,
  number_changed_major,
  total_number_of_students,
  withdrawn_coefficient = 1,
  changed_major_coefficient = 0.5
)
```

Arguments

`number_withrawn`
: numeric - the total number of students who dropped out of a program in a given unit

`number_changed_major`
: numeric - the total number of students who changed majors in a given unit

`total_number_of_students`
: numeric - the total number of students in a given unit starting at a specific time

`withdrawn_coefficient`
: numeric - a coefficient weighting the number of students who dropped out

`changed_major_coefficient`
: numeric - a coefficient weighting the number of students who changed majors out

Value

numeric - The student mobility turbulence

---

subcomplexity_graph *Creates a subcomplexity graph for a course*

Description

This function takes in a plan of study and course, then constructs the subcomplexity graph for the course.

Usage

```
subcomplexity_graph(plan_of_study, course)
```

Arguments

`plan_of_study`	igraph object - An igraph object created using the create_plan_of_study function
`course`	Numeric (vertex id) or String - The course to find the subcomplexity graph of

Value

igraph object representing the course's subcomplexity graph.

transfer_delay_factor *Calculates the transfer delay factor of a course*

Description

This function takes in the subcomplexity graph from the transfer excess courses function, then finds the transfer delay factor. The output is the sum of the longest paths of prerequisites through courses related to those beyond the expected time to degree.

Usage

```
transfer_delay_factor(plan_of_study, expected_time_to_degree)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
expected_time_to_degree	Numeric - The term where students are expected to finish (often 8)

Value

Numeric - the transfer delay factor

transfer_excess_courses *Finds the subcomplexity graph of courses beyond expected time to degree*

Description

This function takes in a plan of study and the expected time to degree, then outputs a subcomplexity graph that contains all of the courses beyond the time to degree and their prerequisites.

Usage

```
transfer_excess_courses(
  plan_of_study,
  expected_time_to_degree,
  include_coreqs = TRUE
)
```

Arguments

plan_of_study	igraph object - An igraph object created using the create_plan_of_study function
expected_time_to_degree	Numeric - The term where students are expected to finish (often 8)
include_coreqs	Logical - Calculates the delay factor using corequisites, default value is TRUE

Value

igraph object - the subcomplexity graph

Index