Package: criticalpath (via r-universe)

September 17, 2024

Title An Implementation of the Critical Path Method

Version 0.2.1.000

Author Rubens Jose Rosa [aut, cre], Marcos dos Santos [aut], Thiago Marques [aut]

Maintainer Rubens Jose Rosa <rubens@rubensjoserosa.com>

URL https://rubensjoserosa.com/criticalpath, https://github.com/rubens2005/criticalpath

BugReports https://github.com/rubens2005/criticalpath/issues

Description An R implementation of the Critical Path Method (CPM). CPM is a method used to estimate the minimum project duration and determine the amount of scheduling flexibility on the logical network paths within the schedule model. The flexibility is in terms of early start, early finish, late start, late finish, total float and free float. Beside, it permits to quantify the complexity of network diagram through the analysis of topological indicators. Finally, it permits to change the activities duration to perform what-if scenario analysis. The package was built based on following references: To make topological sorting and other graph operation, we use Csardi, G. & Nepusz, T. (2005) <https://www.researchgate.net/publication/221995787_The_Igraph_Software_Package_for_Complex_Network_Research>; For schedule concept, the reference was Project Management Institute (2017) <https://www.pmi.org/pmbok-guide-standards/foundational/pmbok>; For standards terms, we use Project Management Institute (2017) <https://www.pmi.org/pmbok-guide-standards/lexicon>; For algorithms on Critical Path Method development, we use Vanhoucke, M. (2013) <doi:10.1007/978-3-642-40438-2> and Vanhoucke, M. (2014) <doi:10.1007/978-3-319-04331-9>; And, finally, for topological definitions, we use Vanhoucke, M. (2009) <doi:10.1007/978-1-4419-1014-1>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports dplyr, igraph, magrittr, R6, stringr, tibble

Suggests DiagrammeR, knitr, rmarkdown, testthat

Collate 'Schedule.R' 'utils-pipe.R' 'criticalpath.R'
'cpt_calculate_critical_path.R' 'cpt_schedule_status.R'
'cpt_topological_organization.R' 'cpt_utils.R' 'sch_activity.R'
'sch_relation.R' 'sch_schedule.R'
'sch_topological_indicators.R'

VignetteBuilder knitr

Repository https://rubens2005.r-universe.dev

RemoteUrl https://github.com/rubens2005/criticalpath

RemoteRef HEAD

RemoteSha e565d30e5d4cd21ebcb1d57d214385f494bd82ac

Contents

`criticalpath` *criticalpath: Critical Path Method R Implementation*

Description

`criticalpath` package is an R implementation of the Critical Path Method (CPM). CPM is a method used to estimate the minimum project duration and determine the amount of scheduling flexibility on the logical network paths within the schedule model. The flexibility is in terms of early start, early finish, late start, late finish, total float and free float. Beside, it permits to quantify the complexity of network diagram through the analysis of topological indicators. Finally, it permits to change the activities duration to perform what-if scenario analysis.

Details

With this package, you can calculate the following CPM parameters:

- Schedule duration
- Early start and finish date of each activity
- Late start and finish date of each activity
- Critical activities
- Critical path
- Total float and free float
- Gantt Matrix
- What-if scenario analysis
- Topological indicators

Author(s)

Rubens Jose Rosa (<rubens@rubensjoserosa.com>), Marcos dos Santos (<marcosdossantos@ime.eb.br>), Thiago Marques (<profestathimarques@gmail.com>)

References

Csardi, G. & Nepusz, T. (2005). The Igraph Software Package for Complex Network Research. *InterJournal*. Complex Systems. 1695.

Project Management Institute (2017) **A Guide to the Project Management Body of Knowledge (PMBOK Guide)**. Sixth Edition.

Project Management Institute (2017) **PMI Lexicon of Project Management Terms:** Version 3.2.

Vanhoucke, M. (2009) **Measuring Time**: Improving Project Performance Using Earned Value Management. Springer-Verlag US.

Vanhoucke, M. (2013) **Project Management with Dynamic Scheduling**: Baseline Scheduling, Risk Analysis and Project Control. Springer-Verlag Berlin Heidelberg.

Vanhoucke, M. (2014) **Integrated Project Management and Control**: First Comes the Theory, then the Practice. Springer International Publishing Switzerland.

See Also

On vignette package there are more information with examples about:

- How to create a schedule:
  - Create a new schedule without any information.
    - `sch_new()`
  - Add activities and relations together to an schedule.
    - `sch_add_activities()`
    - `sch_add_relations()`
  - Add activities to a schedule.
    - `sch_add_activity()`
  - Add relations to a schedule.
    - `sch_add_relation()`
- How to get schedule information:
  - Title
    - `sch_title()`
  - Reference
    - `sch_reference()`
  - Duration
    - `sch_duration()`
- How to get activities properties:
  - Activity Properties.
    - `sch_activities()`
    - `sch_get_activity()`
  - Gantt Matrix.
    - `sch_gantt_matrix()`
    - `sch_xy_gantt_matrix()`
- How to change activities duration:

  - Change Activities Duration.
    * sch_change_activities_duration()
- How to get relations properties:
  - Relation Properties
    * sch_relations()
  - Successors and Predecessors.
    * sch_all_successors()
    * sch_successors()
    * sch_all_predecessors()
    * sch_predecessors()
- How to get topological properties:
  - Topological Indicators.
    * sch_topoi_sp()
    * sch_topoi_ad()
    * sch_topoi_la()
    * sch_topoi_tf()

`Schedule` *R6 Class Representing a Schedule*

Description

This class is a representation of Precedence Diagramming Method (PDM). PDM is a technique used for constructing a schedule model in which activities are represented by nodes and are graphically linked by one or more logical relationships to show the sequence in which the activities are to be performed.

A schedule has activities and relations data-frames. With this class, it is possible to apply critical path method

Active bindings

`title` A project title for identification. It depends on user of the class. Its use are:

- `Schedule$title <- "A title"`
  - sets a title for a project.
- `Schedule$title`
  - gets the title of the project.

`reference` A reference from project origin, for example, a book, a paper, a corporation, or nothing. Its uses are:

- `Schedule$reference <- "A reference"`
  - sets a reference for a project.
- `Schedule$title`
  - gets the reference of the project.

`has_any_activity` A logical value that indicates if the schedule has any activity. A TRUE value means that the schedule has some activity; a FALSE, means that the schedule is empty.

- Usage: `Schedule$has_any_activity`

`nr_activities` Number of activities in a schedule as an integer value.

- Usage: `Schedule$nr_activities`

`activities` Return a data frame with all activities of a schedule in an activity id order. This is the main information calculated by CPM. The data frame is formed by following structure:

- **id:** Activity id.
- **name:** The name of activity.
- **duration:** A number that represents the activity's duration.
- **milestone:** A milestone is an activity with zero duration. This property indicates if an activity is a milestone or not: `TRUE` indicates it is a milestone; `FALSE` indicates it is not.
- **critical:** A critical activity is one with total float minor or equal to zero. This property indicates if an activity is critical: `TRUE` indicates it is critical; `FALSE` indicates it is not critical.
- **ES:** Early Start: is the earliest start period an activity can begin after its predecessors without violating precedence relation.
- **EF:** Early Finish: is the early start plus activity duration.
- **LS:** Late Start: is the late finish minus activity duration.
- **LF:** Late Finish: is the latest finish an activity can finish before their successors without violating precedence relation.
- **total_float:** It is the amount of period an activity can be delayed without violating the project duration. Its formula is: LS - ES or LF - EF.
- **free_float:** It is the amount of period an activity can be delayed without violating the start time of the successors activities.
- **progr_level:** Progressive level is the rank of activities counted from begin. The level of the activities that don't have predecessor is one; the level of the other activities, is one plus the maximal level of their predecessor.
- **regr_level:** Regressive level is the rank of activities counted from the end. The level of the activities that don't have successor is the maximal progressive level; the level of the other activities, is one minus the minimal level of their successor.
- **topo_float:** It is the difference between progressive level and regressive level.
- Usage: `Schedule$activities`

`has_any_relation` A logical value that indicates if the schedule has any relation. A TRUE value means that the schedule has some relation; a FALSE, means that the schedule does not have any relation.

- Usage: `Schedule$has_any_relation`

`nr_relations` Number of relations in a schedule as an integer value.

- Usage: `Schedule$nr_relations`

`relations` Return a data frame with all relations of a schedule in topological order. This is the main information calculated by CPM. The data frame is formed by following structure:

- **from:** Predecessor activity id from a relation.
- **to:** Successor activity id from a relation.

- **type:** The type of relation between activities. Its value may be: FS, FF, SS, SF.
- **lag:** The time period between activity predecessor and activity successor activity
- **critical:** A critical relation formed by two activity critical: predecessor and successor. `TRUE` indicates it is critical; `FALSE` indicates it is not critical.
- **ord:** Indicates de order that the relation was added in the schedule.
- **i_from:** It is the index of predecessor activity in the activities data frame.
- **i_to:** It is the index of successor activity in the activities data frame.
- Usage: `Schedule$relations`

`duration` An integer value that indicates the duration of a schedule.

Methods

Public methods:

- `Schedule$new()`
- `Schedule$add_activity()`
- `Schedule$add_activities()`
- `Schedule$get_activity()`
- `Schedule$add_relation()`
- `Schedule$add_relations()`
- `Schedule$add_act_rel()`
- `Schedule$print()`
- `Schedule$all_successors()`
- `Schedule$all_predecessors()`
- `Schedule$is_redundant()`
- `Schedule$change_durations()`
- `Schedule$gantt_matrix()`
- `Schedule$xy_gantt_matrix()`
- `Schedule$topoi_sp()`
- `Schedule$topoi_ad()`
- `Schedule$topoi_la()`
- `Schedule$topoi_tf()`
- `Schedule$clone()`

Method `new()`**:** Make a schedule with activities and relations between activities. The method `Schedule$new(activities, relations)` creates an schedule object from two data frames, one containing activities lists and the other the precedence relations between activities. After creation, it is applied the Critical Path Method (CPM).

It is possible to create a empty schedule, without any activity or relation with the constructor `Schedule$new()`. After that, it is possible to add activity with `add_activity` and relation with `add_relation` methods.

Usage:

```
Schedule$new(activities = NULL, relations = NULL)
```

Arguments:

`activities` Data frame with activities. If it is not informed, the schedule will be created without any activity. Its structure is:

- **id:** Activity id. It is an integer number that must be unique within a schedule.
- **name:** Activity name. It may be empty.
- **duration:** Activity duration. It is integer number without unit time. It may be zero.

`relations` Data frame with precedence relations between activities. If it is informed, the activities has to be informed too. If it is not informed, the schedule will be created without any relation. It is formed by predecessor activity e successor activity. Its structure is:

- **from:** The id of predecessor activity. Must exist an activity with from id.
- **to:** The id of successor activity. Must exist an activity with to id.
- **type:** Specifies the type of relation between activities. The default type is FS and its value may be: FS, FF, SS, SF, that means:
  - **FS:** Finish-Start relation. Activity to_id can only start after the finish of activity from_id.
  - **FF:** Finish-Finish relation. Activity to_id must finish together with activity from_id.
  - **SS:** Start-Start relation. Activity to_id must start together with activity from_id.
  - **SF:** Start-Finish relation. Activity to_id must finish when activity from_id starts.
- **lag:** The time period between activities that the successor activity must be advanced, or lated, after activity from_id. It must be an integer, less than, equal or greater than zero.

Returns: A Schedule object with CPM parameters calculated.

Method `add_activity()`**:** Add an activity to a schedule.

Usage:

```
Schedule$add_activity(id, name = "", duration = 0L)
```

Arguments:

`id` Activity id that will be used to make relation between activities. It must be unique.

`name` The name of activity. The default is an empty string.

`duration` A number that represents the activity's duration. It must be equal or greater than zero. The default value is zero.

Returns: A Schedule object with an activity added and the critical path calculated.

Method `add_activities()`**:** Add activities from a data frame to a schedule.

Usage:

```
Schedule$add_activities(activities)
```

Arguments:

`activities` A data frame with the activities to be added.

Returns: A Schedule object with activities added and CPM calculated.

Method `get_activity()`**:** Gets an activity by id. It returns a data frame with one line about activity.

Usage:

```
Schedule$get_activity(id)
```

Arguments:

`id` An activity id as defined by the user.

Returns: A data frame with one line with the activity, or an error if activity id doesn't exist.

Method `add_relation()`**:** Add a relation to a schedule.

Usage:

```
Schedule$add_relation(from, to, type = "FS", lag = 0L)
```

Arguments:

`from` The id of predecessor activity. Must exist an activity with from.

`to` The id of successor activity. Must exist an activity with to.

`type` Specifies the type of relation between activities. The default type is FS and its value may be: FS, FF, SS, SF, that means: If type is not defined, it is assumed to be FS.
FS: Finish-Start relation. Activity 'to' id can only start after the finish of activity 'from' id.
FF: Finish-Finish relation. Activity 'to' id must finish together with activity 'from' id.
SS: Start-Start relation. Activity 'to' id must start together with activity 'from' id.
SF: Start-Finish relation. Activity 'to' id must finish when activity 'from' id starts.

`lag` The time period between activities that the successor activity 'to' must be advanced after activity 'from' has been finished. The value may be negative, in such case, the activity 'to' will be anticipated 'lag' time periods. It must be an integer, less than, equal or greater than zero. If lag is not defined, it is assumed to be zero.

Returns: A Schedule object with CPM parameters calculated.

Method `add_relations()`**:** Add relations between activities from a data frame to a schedule.

Usage:

```
Schedule$add_relations(relations)
```

Arguments:

`relations` A data frame with the relations to be added.

Returns: A Schedule object with relations added and CPM calculated.

Method `add_act_rel()`**:** Add an activity and her relations to a schedule.

Usage:

```
Schedule$add_act_rel(
  id,
  name,
  duration,
  relations_id = c(),
  direction = "succ"
)
```

Arguments:

`id` Activity id. The id will be used to make relation between activities.

`name` The name of activity.

`duration` A number that represents the activity's duration. It must be equal or greater than zero.

`relations_id` A vector of ids such that will be linked with activity id. It may be relations of successor or predecessors.

`direction` Direction of relations_id: It may be "succ" or "pred". If dir="succ" the relations_id will be the successor of the activity. If dir="pred" the relations_id will be the predecessor of the activity.

Returns: A Schedule object.

Method `print()`**:** Print a description of the class

Usage:

```
Schedule$print(...)
```

Arguments:

`...` Variable parameters

Returns: A String .

Method `all_successors()`**:** List all successors from an activity: direct and indirect successors.

Usage:

```
Schedule$all_successors(id, ign_to = NULL)
```

Arguments:

`id` Activity id to be listed.

`ign_to` A relation to be ignored: id -> ign_to. Activities from this relation will be ignored.

Returns: A vector whith all activities ids.

Method `all_predecessors()`**:** List all predecessors from an activity: direct or indirect predecessors.

Usage:

```
Schedule$all_predecessors(id, ign_from = NULL)
```

Arguments:

`id` Activity id to be listed.

`ign_from` A relation to be ignored: ign_from -> id. Activities from this relation will be ignored.

Returns: A vector with all activities ids.

Method `is_redundant()`**:** Verify if a relation between two activities is redundant. A relation A->C is redundant if there are A->C, A->B, B->C relations.

Usage:

```
Schedule$is_redundant(id_from, id_to)
```

Arguments:

`id_from` From activity id.

`id_to` To activity id.

Returns: A logical `TRUE` if an arc is redundant; `FALSE` if it is not.

Method `change_durations()`**:** Change activities duration and calculate critical path. This way is faster than creating a new schedule with new durations.

Usage:

```
Schedule$change_durations(new_durations)
```

Arguments:

`new_durations` A vector with new activities' duration.

Returns: A Schedule object.

Method `gantt_matrix()`**:** Create a matrix that represents a Gantt chart, a matrix where "1" indicates that an activity is planned to be in execution.

In this matrix, the rows represent activities, whereas the columns represents the activity execution period. So, the number of columns is equal to project duration.

Usage:

```
Schedule$gantt_matrix()
```

Returns: A matrix where "1" indicates that an activity is in execution.

Method `xy_gantt_matrix()`**:** Transform a Gantt matrix in x, y coordinates and the weight one. Each point greater than zero in a Gantt matrix becomes a x, y coordinate.

Usage:

```
Schedule$xy_gantt_matrix(gantt = NULL)
```

Arguments:

`gantt` A Gantt Matrix. If it is not informed, it will use `gantt_matrix()` before this function.

Returns: A matrix x, y and weight.

Method `topoi_sp()`**: SP Serial or Parallel Topological Indicator:** It shows the closeness of a network to a serial or parallel graph. As the network becomes serial, the SP increase, until one, when the network totally serial.

Usage:

```
Schedule$topoi_sp()
```

Returns: A number between 0 and 1, inclusive.

Method `topoi_ad()`**: AD Activity Distribution Topological Indicator:** Measures the distribution of the activities over the levels. If AD is approximately equal zero, each level has same numbers of activities. Otherwise, if AD is equal one, the quantity of each level is not uniformly distributed.

Usage:

```
Schedule$topoi_ad()
```

Returns: A number between 0 and 1, inclusive.

Method `topoi_la()`**: LA Length of Arcs Topological Indicator:** Measures the presence of long arcs based on the difference between the progressive level of the end activity and the start node of each relation. If LA is approximately equal zero, the progressive level between activities is as far as possible. Otherwise, if LA is equal one, the relation distance are one.

Usage:

```
Schedule$topoi_la()
```

Returns: A number between 0 and 1, inclusive.

Method `topoi_tf()`**: TF Topological Float Indicator:** Measures the topological float of each activity. If TF = 0, there is no float between activities. If TF = 1, there is float between activities and they be shift without affecting other activities.

Usage:

```
Schedule$topoi_tf()
```

Returns: A number between 0 and 1, inclusive.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Schedule$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Rubens Jose Rosa (<rubens@rubensjoserosa.com>), Marcos dos Santos (<marcosdossantos@ime.eb.br>), Thiago Marques (<profestathimarques@gmail.com>)

References

Csardi, G. & Nepusz, T. (2005). The Igraph Software Package for Complex Network Research. *InterJournal*. Complex Systems. 1695.

Project Management Institute (2017) **A Guide to the Project Management Body of Knowledge (PMBOK Guide)**. Sixth Edition.

Project Management Institute (2017) **PMI Lexicon of Project Management Terms:** Version 3.2.

Vanhoucke, M. (2009) **Measuring Time**: Improving Project Performance Using Earned Value Management. Springer-Verlag US.

Vanhoucke, M. (2013) **Project Management with Dynamic Scheduling**: Baseline Scheduling, Risk Analysis and Project Control. Springer-Verlag Berlin Heidelberg.

Vanhoucke, M. (2014) **Integrated Project Management and Control**: First Comes the Theory, then the Practice. Springer International Publishing Switzerland.

See Also

On vignette package there is more information with examples about:

- Critical Path Method Package criticalpath.
- How to create a schedule:
  - Add activities and relations together to an schedule.
  - Add activities to a schedule.
  - Add relations to a schedule.
  - Create a schedule object from data frames.
- How to get schedule information:
  - Title, Reference and Schedule Duration.

- How to get activities properties:
  - Activity Properties.
  - Gantt Matrix.
- How to change activities duration:
  - Change Activities Duration.
- How to get relations properties:
  - Relation Properties
  - Successors and Predecessors.
- How to get topological properties:
  - Topological Indicators.

`sch_activities` *Activities*

Description

Return a tibble with all activities of a schedule in an insertion order. These are the main information calculated by CPM.

Usage

```
sch_activities(sch)
```

Arguments

`sch`	A schedule object.

Details

The tibble is formed by following structure:

- **id:** Activity id.
- **name:** The name of activity.
- **duration:** A number that represents the activity's duration.
- **milestone:** A milestone is an activity with zero duration. This property indicates if an activity is a milestone or not: `TRUE` indicates it is a milestone; `FALSE` indicates it is not.
- **critical:** A critical activity is one with total float minor or equal to zero. This property indicates if an activity is critical: `TRUE` indicates it is critical; `FALSE` indicates it is not critical.
- **early_start:** Is the earliest start period an activity can begin after its predecessors without violating precedence relation.
- **early_finish:** Is the early start plus activity duration.
- **late_start:** Is the late finish minus activity duration.

- **late_finish:** Is the latest finish an activity can finish before their successors without violating precedence relation.
- **total_float:** It is the amount of period an activity can be delayed without violating the project duration. Its formula is: late_start - early_start or late_finish - early_finish
- **free_float:** It is the amount of period an activity can be delayed without violating the start time of the successors activities.
- **progr_level:** It is the rank of activities counted from begin. The level of the activities that don't have predecessor is one; the level of the other activities, is one plus the maximal level of their predecessor.
- **regr_level:** Regressive level is the rank of activities counted from the end. The level of the activities that don't have successor is the maximal progressive level; the level of the other activities, is one minus the minimal level of their successor.
- **topo_float:** It is the difference between progressive level and regressive level.

Value

A tibble with activities.

See Also

sch_has_any_activity(), sch_change_activities_duration(), sch_add_activity(), sch_nr_activities(), sch_critical_activities(), sch_add_activities(), sch_get_activity(), sch_duration().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2,3,4) %>%
  sch_add_activity(  2L, "a2" , 4L, 5) %>%
  sch_add_activity(  3L, "a3" , 9L, 10) %>%
  sch_add_activity(  4L, "a4" , 1L, 6) %>%
  sch_add_activity(  5L, "a5" , 4L, 9) %>%
  sch_add_activity(  6L, "a6" , 5L, 7) %>%
  sch_add_activity(  7L, "a7" , 1L, 8,11) %>%
  sch_add_activity(  8L, "a8" , 7L, 12) %>%
  sch_add_activity(  9L, "a9" , 8L, 12) %>%
  sch_add_activity( 10L, "a10", 3L, 12) %>%
  sch_add_activity( 11L, "a11", 3L, 12) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_activities(sch)
```

`sch_add_activities` *Add Activities*

Description

Combine several vectors for activities and their attributes into a tibble, which can be combined with other similarly-generated tibbles, resulting in unique tibble to be added in a schedule. If the schedule already contain some activities, the new activities will be added in the end.

Usage

```
sch_add_activities(sch, id, name, duration, ...)
```

Arguments

`sch`	A schedule object.
`id`	Activity id that will be used to make relation between activities. It must be unique.
`name`	The name of activity.
`duration`	A number that represents the activity's duration. It must be equal or greater than zero.
`...`	One or more vectors for associated activity attributes.

Details

A activity tibble, or atb, has at least the following columns:

- `id` (of type `integer`): Activity id. It is an integer number that must be unique within a schedule.
- `name` (of type `character`): Activity name. It may be empty string.
- `duration` (of type `integer`): Activity duration. It is integer number without unit time. It may be zero.

An arbitrary number of additional columns containing data attributes can be part of the atb, so long as they follow the aforementioned columns.

Value

A schedule with a activity tibble (atb) added.

See Also

`sch_reference()`, `sch_add_relations()`, `sch_add_activity()`, `sch_title()`, `sch_nr_activities()`, `sch_new()`, `sch_plan()`, `sch_get_activity()`, `sch_has_any_activity()`, `sch_change_activities_duration()`.

Examples

```
# Example #1
sch <- sch_new() %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
  ) %>%
  sch_plan()
sch_duration(sch)
sch_activities(sch)

# Example #2
sch <- sch_new() %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L),
    resource  = "Rubens",
    cost      = 123.45
  ) %>%
  sch_plan()
sch_duration(sch)
atb <- sch_activities(sch)
atb$resource
atb$cost

# Example #3
sch <- sch_new() %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L),
    resource  = c(
      "Rubens", "Jose", "Rosa", "Rodrigues", "Silva",
      "Rubens", "Jose", "Rosa", "Rodrigues", "Silva",
      "Rubens", "Jose", "Rosa", "Rodrigues", "Silva",
      "Rubens", "Jose"),
    cost      = c(
      123.45, 234.56, 345.56, 456.78, 567.89,
      123.45, 234.56, 345.56, 456.78, 567.89,
      123.45, 234.56, 345.56, 456.78, 567.89,
      123.45, 234.56)
  ) %>%
  sch_plan()
sch_duration(sch)
atb <- sch_activities(sch)
atb$resource
atb$cost
```

sch_add_activities_tibble *Add Activities Tibble*

Description

Add activities tibble to a schedule.

Usage

```
sch_add_activities_tibble(sch, atb)
```

Arguments

sch	A schedule object.
atb	A tibble with activities definitions.

Value

A schedule with a activity tibble (atb) added.

Examples

```
atb <- tibble::tibble(
  id        = 1:17,
  name      = paste("a", as.character(1:17), sep=""),
  duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
)
sch <- sch_new() %>%
  sch_add_activities_tibble(atb) %>%
  sch_plan()
sch_duration(sch) #4
# sch_activities(sch)
```

sch_add_activity *Add Activity*

Description

Add an activity and her relations to a schedule. The relations are optional. They will be included if a set of activity id is informed, after activity duration.

Usage

```
sch_add_activity(sch, id, name, duration, ..., direction = "succ")
```

Arguments

sch	A schedule object.
id	Activity id that will be used to make relation between activities. It must be unique.
name	The name of activity.
duration	A number that represents the activity's duration. It must be equal or greater than zero.
...	A set of activity id relation such that will be linked with activity id. It may be relations of successor or predecessors.
direction	Direction of relations: It may be "succ" or "pred". • succ: The relations_id will be the successor of the activity. • pred: The relations_id will be the predecessor of the activity.

Value

A Schedule object with an activity added to it. If relations id is present, it will be included to the schedule.

See Also

sch_change_activities_duration(), sch_has_any_activity(), sch_new(), sch_add_activities(), sch_get_activity(), sch_plan(), sch_nr_activities(), sch_add_relation().

Examples

```
# Example #1: Only with activities
sch <- sch_new() %>%
  sch_add_activity(1L, "Task 1", 5L) %>%
  sch_add_activity(2L, "Task 2", 6L) %>%
  sch_add_activity(3L, "Task 3", 8L) %>%
  sch_add_activity(4L, "Task 4", 6L) %>%
  sch_add_activity(5L, "Task 5", 9L) %>%
  sch_add_activity(6L, "Task 6", 3L) %>%
  sch_add_activity(7L, "Task 7", 4L) %>%
  sch_plan()
sch_duration(sch)
sch_activities(sch)

# Example #2: With activities and relations.
sch <- sch_new() %>%
  sch_add_activity(1L, "Task 1", 5L, 2L, 3L) %>%
  sch_add_activity(2L, "Task 2", 6L, 4L) %>%
  sch_add_activity(3L, "Task 3", 8L, 5L) %>%
  sch_add_activity(4L, "Task 4", 6L, 6L) %>%
  sch_add_activity(5L, "Task 5", 9L, 6L) %>%
  sch_add_activity(6L, "Task 6", 3L, 7L) %>%
  sch_add_activity(7L, "Task 7", 4L) %>%
  sch_plan()
sch_duration(sch)
```

```
sch_activities(sch)
sch_relations(sch)
```

`sch_add_relation` *Add Relation*

Description

Add a relation to a schedule.

Usage

```
sch_add_relation(sch, from, to, type = "FS", lag = 0L)
```

Arguments

`sch`	A schedule object.
`from`	The id of predecessor activity. Must exist an activity with from.
`to`	The id of successor activity. Must exist an activity with to.
`type`	Specifies the type of relation between activities. The default type is FS and its value may be: FS, FF, SS, SF, that means: • **FS:** Finish-Start relation. Activity 'to' id can only start after the finish of activity 'from' id. • **FF:** Finish-Finish relation. Activity 'to' id must finish together with activity 'from' id. • **SS:** Start-Start relation. Activity 'to' id must start together with activity 'from' id. • **SF:** Start-Finish relation. Activity 'to' id must finish when activity 'from' id starts. If type is not defined, it is assumed to be FS.
`lag`	The time period between activities that the successor activity 'to' must be advanced after activity 'from' has been finished. The value may be negative, in such case, the activity 'to' will be anticipated 'lag' time periods. It must be an integer, less than, equal or greater than zero. If lag is not defined, it is assumed to be zero.

Value

A Schedule object with a relation added.

See Also

`sch_has_any_relation()`, `sch_nr_relations()`, `sch_add_relations()`, `sch_plan()`, `sch_validate()`, `sch_add_activities()`, `sch_new()`.

Examples

```
sch <- sch_new() %>%
  sch_title("Project 3: Old Carriage House Renovation") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
  first comes the theory, then the practice. Gent: Springer, 2014, p. 11") %>%
  sch_add_activity( 1L, "a1" , 2L) %>%
  sch_add_activity( 2L, "a2" , 2L) %>%
  sch_add_activity( 3L, "a3" , 4L) %>%
  sch_add_activity( 4L, "a4" , 3L) %>%
  sch_add_activity( 5L, "a5" , 4L) %>%
  sch_add_activity( 6L, "a6" , 1L) %>%
  sch_add_activity( 7L, "a7" , 1L) %>%
  sch_add_activity( 8L, "a8" , 1L) %>%
  sch_add_activity( 9L, "a9" , 1L) %>%
  sch_add_activity(10L, "a10", 1L) %>%
  sch_add_activity(11L, "a11", 3L) %>%
  sch_add_activity(12L, "a12", 2L) %>%
  sch_add_activity(13L, "a13", 1L) %>%
  sch_add_activity(14L, "a14", 1L) %>%
  sch_add_activity(15L, "a15", 2L) %>%
  sch_add_activity(16L, "a16", 1L) %>%
  sch_add_activity(17L, "a17", 1L) %>%
  sch_add_relation( 1L,  2L) %>%
  sch_add_relation( 2L,  3L) %>%
  sch_add_relation( 3L,  4L) %>%
  sch_add_relation( 4L,  5L) %>%
  sch_add_relation( 5L,  6L) %>%
  sch_add_relation( 6L,  7L) %>%
  sch_add_relation( 6L,  8L) %>%
  sch_add_relation( 6L,  9L) %>%
  sch_add_relation( 7L, 10L) %>%
  sch_add_relation( 8L, 10L) %>%
  sch_add_relation( 9L, 10L) %>%
  sch_add_relation(10L, 11L) %>%
  sch_add_relation(10L, 13L) %>%
  sch_add_relation(11L, 12L) %>%
  sch_add_relation(12L, 15L) %>%
  sch_add_relation(13L, 14L) %>%
  sch_add_relation(14L, 15L) %>%
  sch_add_relation(15L, 16L) %>%
  sch_add_relation(16L, 17L) %>%
  sch_plan()
sch_duration(sch)
sch_activities(sch)
sch_relations(sch)
```

`sch_add_relations` *Add Relations*

Description

Combine several vectors for relation and their attributes into a tibble and add relations between activities to a schedule.

Usage

```
sch_add_relations(sch, from, to, type = "FS", lag = 0L, ...)
```

Arguments

sch	A schedule object.
from	The id of predecessor activity.
to	The id of successor activity.
type	Specifies the relation type between activities. The default type is FS and its value may be: FS, FF, SS, SF.
lag	The time period between activities that the successor activity to must be advanced after activity from has been finished.
...	One or more vectors for associated relation attributes.

Details

An relation tibble, or rtb, has at least the following columns:

- from (of type integer): The id of predecessor activity. Must exist an activity with from id.
- to (of type integer): The id of successor activity. Must exist an activity with to id.
- type (of type character) Specifies the relation type between activities. The default type is FS and its value may be: FS, FF, SS, SF, that means:
  - **FS:** Finish-Start relation. Activity 'to' id can only start after the finish of activity 'from' id.
  - **FF:** Finish-Finish relation. Activity 'to' id must finish together with activity 'from' id.
  - **SS:** Start-Start relation. Activity 'to' id must start together with activity 'from' id.
  - **SF:** Start-Finish relation. Activity 'to' id must finish when activity 'from' id starts.
- lag (of type integer): The time period between activities that the successor activity to must be advanced after activity from has been finished. The value may be negative, in such case, the activity 'to' will be anticipated 'lag' time periods. It must be an integer, less than, equal or greater than zero. If lag is not defined, it is assumed to be zero.

An arbitrary number of additional columns containing data attributes can be part of the rtb, so long as they follow the aforementioned columns.

Value

A schedule with a relation tibble (rtb) added.

See Also

sch_title(), sch_reference(), sch_add_relation(), sch_nr_relations(), sch_has_any_relation(), sch_new(), sch_plan(), sch_add_activities(), sch_validate().

Examples

```
sch <- sch_new() %>%
  sch_title("Project 1: Cost Information System") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
    first comes the theory, then the practice.Gent: Springer, 2014, p. 6"
  ) %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
  ) %>%
  sch_plan()
sch_has_any_relation(sch) # FALSE
sch_nr_relations(sch) # 0
sch_duration(sch) # 4

sch %<>%
  sch_add_relations(
    from = c(1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L,  3L,  4L,  5L,  6L,
             7L,  8L,  9L, 10L, 11L, 11L, 12L, 12L, 13L, 13L, 14L, 14L, 15L, 15L),
    to   = c(2L, 3L, 4L, 5L, 6L, 7L, 8L, 9L, 10L, 11L, 11L, 11L,
             12L, 13L, 14L, 15L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L)
  ) %>%
  sch_plan()
sch_has_any_relation(sch) # TRUE
sch_nr_relations(sch) # 26
sch_duration(sch) # 11
```

---

sch_add_relations_tibble

Add Relations Tibble

Description

Add relations tibble to a schedule.

Usage

```
sch_add_relations_tibble(sch, rtb)
```

Arguments

sch	A schedule object.
rtb	A tibble or data frame with relations definitions.

Value

A Schedule object with a relation added.

Examples

```
atb <- tibble::tibble(
  id        = 1:17,
  name      = paste("a", as.character(1:17), sep=""),
  duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
)
rtb <- data.frame(
  from = c(1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L,  3L,  4L,  5L,  6L,
           7L,  8L,  9L, 10L, 11L, 11L, 12L, 12L, 13L, 13L, 14L, 14L, 15L, 15L),
  to   = c(2L, 3L, 4L, 5L, 6L, 7L, 8L, 9L, 10L, 11L, 11L, 11L,
           12L, 13L, 14L, 15L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L)
)
sch <- sch_new() %>%
  sch_add_activities_tibble(atb) %>%
  sch_add_relations_tibble(rtb) %>%
  sch_plan()
sch_duration(sch) # 11
```

---

sch_all_predecessors *All Predecessors*

Description

List all predecessors from an activity: direct or indirect predecessors.

Usage

```
sch_all_predecessors(sch, id, ign_from = NULL)
```

Arguments

sch	A schedule object.
id	Activity id to be listed.
ign_from	A relation to be ignored: ign_from -> id. Activities from this relation will be ignored.

Value

A vector with all activities ids.

See Also

sch_successors(), sch_relations(), sch_is_redundant(), sch_all_successors(), sch_activities(), sch_non_critical_activities(), sch_predecessors().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity( 2L, "a2" , 4L,  5L, 12L) %>%
  sch_add_activity( 3L, "a3" , 9L, 10L) %>%
  sch_add_activity( 4L, "a4" , 1L,  6L) %>%
  sch_add_activity( 5L, "a5" , 4L,  9L) %>%
  sch_add_activity( 6L, "a6" , 5L,  7L) %>%
  sch_add_activity( 7L, "a7" , 1L,  8L,11L) %>%
  sch_add_activity( 8L, "a8" , 7L, 12L) %>%
  sch_add_activity( 9L, "a9" , 8L, 12L) %>%
  sch_add_activity(10L, "a10", 3L, 12L) %>%
  sch_add_activity(11L, "a11", 3L, 12L) %>%
  sch_add_activity(12L, "a12", 0L) %>%
  sch_plan()
sch_all_predecessors(sch, 2) # nothing
sch_all_predecessors(sch, 7) # 6, 4
sch_all_predecessors(sch, 10) # 3
```

---

sch_all_successors *All Successors*

Description

List all successors from an activity: direct and indirect successors.

Usage

```
sch_all_successors(sch, id, ign_to = NULL)
```

Arguments

sch	A schedule object.
id	Activity id to be listed.
ign_to	A relation to be ignored: id -> ign_to. Activities from this relation will be ignored.

Value

A vector with all activities ids.

See Also

sch_predecessors(), sch_activities(), sch_non_critical_activities(), sch_successors(), sch_relations(), sch_is_redundant(), sch_all_predecessors().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity( 2L, "a2" , 4L,  5L, 12L) %>%
  sch_add_activity( 3L, "a3" , 9L, 10L) %>%
  sch_add_activity( 4L, "a4" , 1L,  6L) %>%
  sch_add_activity( 5L, "a5" , 4L,  9L) %>%
  sch_add_activity( 6L, "a6" , 5L,  7L) %>%
  sch_add_activity( 7L, "a7" , 1L,  8L,11L) %>%
  sch_add_activity( 8L, "a8" , 7L, 12L) %>%
  sch_add_activity( 9L, "a9" , 8L, 12L) %>%
  sch_add_activity(10L, "a10", 3L, 12L) %>%
  sch_add_activity(11L, "a11", 3L, 12L) %>%
  sch_add_activity(12L, "a12", 0L) %>%
  sch_plan()

sch_all_successors(sch, 2) # 5, 9, 12
sch_all_successors(sch, 7) # 8, 11, 12
sch_all_successors(sch, 10) # 12
```

---

sch_change_activities_duration

Change Activities Duration

Description

Change activities duration and calculates critical path. This way is faster than creating a new schedule with new durations. The order of duration is the insertion order of activities.

Usage

```
sch_change_activities_duration(sch, new_durations)
```

Arguments

sch	A schedule object.
new_durations	A vector with new activities' duration.

Value

A schedule object with new durations.

See Also

sch_activities(), sch_has_any_activity(), sch_duration(), sch_nr_activities(), sch_add_activity(), sch_add_activities(), sch_get_activity().

Examples

```
sch <- sch_new() %>%
  sch_title("Project 2: Patient Transport System") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
  first comes the theory, then the practice. Gent: Springer, 2014, p. 9") %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,1L,3L,2L, 2L,2L,2L,1L, 4L,5L,3L,3L, 4L,5L,1L,5L,2L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 3L, 3L, 4L, 5L, 6L, 7L, 8L,  8L,  8L,
             8L,  8L,  9L, 10L, 11L, 12L, 13L, 13L, 14L, 14L, 15L, 15L),
    to   = c(2L, 3L, 4L, 6L, 5L, 8L, 7L, 8L, 9L, 10L, 11L,
             12L, 13L, 14L, 14L, 14L, 14L, 14L, 15L, 16L, 17L, 16L, 17L)
  ) %>%
  sch_plan()

# Project duration
sch_duration(sch) # 25
# Activities duration
atb <- sch_activities(sch)
atb$duration

# Now, change activities duration
new_durations <- c(1L,2L,5L, 4L,3L, 2L,1L, 5L, 3L,5L,5L,3L,4L, 2L,1L, 2L,4L)
sch %<>%
  sch_change_activities_duration(new_durations)

#Project duration
sch_duration(sch) # 31
# Activities duration
atb <- sch_activities(sch)
atb$duration
```

---

sch_critical_activities

Critical Activities

Description

Return a tibble with all critical activities of a schedule in an insertion order.

Usage

```
sch_critical_activities(sch)
```

Arguments

sch	A schedule object.

Value

A tibble with critical activities.

See Also

sch_get_activity(), sch_add_activities(), sch_activities(), sch_add_activity(), sch_nr_activities(), sch_non_critical_activities(), sch_has_any_activity().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2,3,4) %>%
  sch_add_activity(  2L, "a2" , 4L, 5) %>%
  sch_add_activity(  3L, "a3" , 9L, 10) %>%
  sch_add_activity(  4L, "a4" , 1L, 6) %>%
  sch_add_activity(  5L, "a5" , 4L, 9) %>%
  sch_add_activity(  6L, "a6" , 5L, 7) %>%
  sch_add_activity(  7L, "a7" , 1L, 8,11) %>%
  sch_add_activity(  8L, "a8" , 7L, 12) %>%
  sch_add_activity(  9L, "a9" , 8L, 12) %>%
  sch_add_activity( 10L, "a10", 3L, 12) %>%
  sch_add_activity( 11L, "a11", 3L, 12) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_critical_activities(sch)
```

---

sch_critical_relations

Critical Relations

Description

Return a tibble with critical relations of a schedule in topological order.

Usage

```
sch_critical_relations(sch, order = "topological")
```

Arguments

sch	A schedule object.
order	Indicates the order of relations: • "topological": The relations tibble is in topological order. • "insert": The relations tibble is in insert order.

Value

A tibble with critical relations.

See Also

sch_relations(), sch_add_activities(), sch_has_any_relation(), sch_topoi_tf(), sch_gantt_matrix(), sch_activities(), sch_topoi_la(), sch_add_relations(), sch_topoi_sp(), sch_non_critical_activities(), sch_topoi_ad(), sch_nr_relations(), sch_non_critical_relations().

Examples

```
sch <- sch_new() %>%
  sch_title("Project 3: Old Carriage House Renovation") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
  first comes the theory, then the practice. Gent: Springer, 2014, p. 11") %>%
  sch_add_activity( 1L, "a1" , 2L) %>%
  sch_add_activity( 2L, "a2" , 2L) %>%
  sch_add_activity( 3L, "a3" , 4L) %>%
  sch_add_activity( 4L, "a4" , 3L) %>%
  sch_add_activity( 5L, "a5" , 4L) %>%
  sch_add_activity( 6L, "a6" , 1L) %>%
  sch_add_activity( 7L, "a7" , 1L) %>%
  sch_add_activity( 8L, "a8" , 1L) %>%
  sch_add_activity( 9L, "a9" , 1L) %>%
  sch_add_activity(10L, "a10", 1L) %>%
  sch_add_activity(11L, "a11", 3L) %>%
  sch_add_activity(12L, "a12", 2L) %>%
  sch_add_activity(13L, "a13", 1L) %>%
  sch_add_activity(14L, "a14", 1L) %>%
  sch_add_activity(15L, "a15", 2L) %>%
  sch_add_activity(16L, "a16", 1L) %>%
  sch_add_activity(17L, "a17", 1L) %>%
  sch_add_relation(14L, 15L) %>%
  sch_add_relation( 9L, 10L) %>%
  sch_add_relation( 2L,  3L) %>%
  sch_add_relation( 8L, 10L) %>%
  sch_add_relation(10L, 13L) %>%
  sch_add_relation( 5L,  6L) %>%
  sch_add_relation(11L, 12L) %>%
  sch_add_relation(15L, 16L) %>%
  sch_add_relation( 6L,  8L) %>%
  sch_add_relation( 3L,  4L) %>%
  sch_add_relation(16L, 17L) %>%
```

```
  sch_add_relation( 6L,  7L) %>%
  sch_add_relation(10L, 11L) %>%
  sch_add_relation(13L, 14L) %>%
  sch_add_relation( 4L,  5L) %>%
  sch_add_relation( 7L, 10L) %>%
  sch_add_relation(12L, 15L) %>%
  sch_add_relation( 6L,  9L) %>%
  sch_add_relation( 1L,  2L) %>%
  sch_plan()
# In "topological" order.
sch_critical_relations(sch)
# In "insert" order.
sch_critical_relations(sch, order = "insert")
```

`sch_duration` *Duration*

Description

An integer value that indicates the duration of a schedule. **Atention:** the schedule must be planned with the function `sch_plan()`.

Usage

```
sch_duration(sch)
```

Arguments

`sch`	A schedule object.

Value

The duration of the schedule.

See Also

`sch_change_activities_duration()`, `sch_validate()`, `sch_add_activities()`, `sch_reference()`, `sch_add_relations()`, `sch_title()`, `sch_gantt_matrix()`, `sch_plan()`, `sch_new()`.

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id = c(1L, 2L, 3L, 4L),
    name = c("A", "B", "C", "D"),
    duration = c(3L, 4L, 9L, 1L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 2L),
```

```
    to   = c(2L, 3L, 4L)
  ) %>%
  sch_plan()
sch_duration(sch) # 16
```

`sch_evaluate_redundancy` *Evaluate Redundancy*

Description

Evaluates redundancy of each relation and creates another column in relation tibble. If the schedule does not have any relation, this function do nothing.

Usage

```
sch_evaluate_redundancy(sch)
```

Arguments

`sch`	Object Schedule

Value

Object Schedule redundancy column added. Or the Schedule without any modification, is trere is no relation in it.

Examples

```
atb <- tibble::tibble(
  id        = 1:17,
  name      = paste("a", as.character(1:17), sep=""),
  duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
)
rtb <- data.frame(
  from = c(1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L,  3L,  4L,  5L,  6L,
           7L,  8L,  9L, 10L, 11L, 11L, 12L, 12L, 13L, 13L, 14L, 14L, 15L, 15L),
  to   = c(2L, 3L, 4L, 5L, 6L, 7L, 8L, 9L, 10L, 11L, 11L, 11L,
           12L, 13L, 14L, 15L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L, 16L, 17L)
)
sch <- sch_new() %>%
  sch_title("Project 1: Cost Information System") %>%
  sch_reference("VANHOUCKE, Mario.
      Integrated project management and control:
      first comes the theory, then the practice.
      Gent: Springer, 2014, p. 6") %>%
  sch_add_activities_tibble(atb) %>%
  sch_add_relations_tibble(rtb) %>%
```

```
  sch_plan() %>%
  sch_evaluate_redundancy()

sch_duration(sch)  # 11L

rtb <- sch_relations(sch)
sum(rtb$redundant) # 0

sch1 <- sch %>%
  sch_add_relation(1L, 6L) %>%
  sch_add_relation(3L, 16L) %>%
  sch_add_relation(4L, 17L) %>%
  sch_plan() %>%
  sch_evaluate_redundancy()

sch_duration(sch)  # 11L

rtb <- sch_relations(sch1)
sum(rtb$redundant) # 3L
```

`sch_gantt_matrix` *Gantt Matrix*

Description

Create a matrix that represents a Gantt chart, a matrix where "1" indicates that an activity is planned to be in execution. **Atention:** the schedule must be planned with the function `sch_plan()`.

Usage

```
sch_gantt_matrix(sch)
```

Arguments

`sch`	A schedule object.

Details

In this matrix, the rows represent activities, whereas the columns represents the activity execution period. So, the number of columns is equal to project duration. The cells is an integer value that indicates the activity is in execution or not.

Value

A matrix where `1` indicates that an activity is in execution and `0`, the activity is not executing.

See Also

sch_add_activities(), sch_activities(), sch_add_relations(), sch_add_relation(), sch_relations(), sch_plan(), sch_xy_gantt_matrix().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id        = c( 1L,  2L,  3L,  4L),
    name      = c("A", "B", "C", "D"),
    duration  = c( 2L,  3L,  1L, 2L )
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 4L, 4L),
    to   = c(3L, 3L, 1L, 2L)
  ) %>%
  sch_plan()

sch_duration(sch)
gantt <- sch_gantt_matrix(sch)
gantt

# What is the effort by time period?
colSums(gantt) # 1 1 2 2 1 1

# What is the duration by activities?
rowSums(gantt) # 2 3 1 2

# what is the S curve
cumsum(colSums(gantt))
plot(cumsum(colSums(gantt)), type="l", lwd=3)
```

---

sch_get_activity *Get Activity*

Description

Gets an activity by id.

Usage

```
sch_get_activity(sch, aid)
```

Arguments

sch	A schedule object.
aid	An activity id as defined by the user.

Value

A an activity information in a tibble with one line, or an error if activity id doesn't exist.

See Also

sch_activities(), sch_duration(), sch_nr_activities(), sch_add_activities(), sch_critical_activities(), sch_has_any_activity(), sch_change_activities_duration(), sch_add_activity().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id        = 1:17,
    name      = paste("a", as.character(1:17), sep=""),
    duration  = c(1L,2L,2L,4L,3L,3L,3L,2L,1L,1L,2L,1L,1L,1L,1L,2L,1L)
  ) %>%
  sch_plan()
sch_get_activity(sch, 7)
```

---

sch_has_any_activity *Has Any Activity*

Description

A logical value that indicates if the schedule has any activity. A TRUE value means that the schedule has any activity; a FALSE, means that the schedule do not have any activity.

Usage

```
sch_has_any_activity(sch)
```

Arguments

sch	A schedule object.

Value

A logical value:

- TRUE: The schedule has any activity;
- FALSE: The schedule do not have any activity.

See Also

sch_nr_activities(), sch_critical_activities(), sch_add_activities(), sch_change_activities_duration(), sch_activities(), sch_nr_relations(), sch_has_any_relation(), sch_add_activity().

Examples

```
sch <- sch_new()
sch_has_any_activity(sch) # FALSE

sch <- sch_new() %>%
  sch_add_activity(1L, "Only one", 0L) %>%
  sch_plan()
sch_has_any_activity(sch) # TRUE
```

`sch_has_any_relation` *Has Any Relation*

Description

A logical value that indicates if the schedule has any relation. A TRUE value means that the schedule has some relation; a FALSE, means that the schedule do not have any relation.

Usage

```
sch_has_any_relation(sch)
```

Arguments

`sch`	A schedule object.

Value

A logical value:

- TRUE: The schedule has any relation;
- FALSE: The schedule do not have any relation.

See Also

`sch_topoi_la()`, `sch_relations()`, `sch_add_relations()`, `sch_topoi_sp()`, `sch_has_any_activity()`, `sch_all_predecessors()`, `sch_topoi_ad()`, `sch_nr_relations()`, `sch_all_successors()`, `sch_nr_activities()`, `sch_topoi_tf()`.

Examples

```
sch <- sch_new()
sch_has_any_relation(sch) # FALSE

sch <- sch_new() %>%
  sch_add_activity(1L, "A", 2L) %>%
  sch_add_activity(2L, "B", 5L, 1L, direction = "pred")
sch_has_any_activity(sch) # TRUE
```

`sch_is_redundant` *Is Redundant*

Description

Verify if a relation between two activities is redundant. A relation A->C is redundant if there are A->C, A->B, B->C relations.

Usage

```
sch_is_redundant(sch, id_from, id_to)
```

Arguments

`sch`	A schedule object.
`id_from`	From activity id.
`id_to`	To activity id.

Value

A logical `TRUE` if an arc is redundant; `FALSE` if it is not.

See Also

`sch_all_predecessors()`, `sch_all_successors()`, `sch_gantt_matrix()`, `sch_relations()`, `sch_predecessors()`, `sch_successors()`, `sch_activities()`.

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity( 2L, "a2" , 4L,  5L, 12L) %>%
  sch_add_activity( 3L, "a3" , 9L, 10L) %>%
  sch_add_activity( 4L, "a4" , 1L,  6L) %>%
  sch_add_activity( 5L, "a5" , 4L,  9L) %>%
  sch_add_activity( 6L, "a6" , 5L,  7L) %>%
  sch_add_activity( 7L, "a7" , 1L,  8L,11L) %>%
  sch_add_activity( 8L, "a8" , 7L, 12L) %>%
  sch_add_activity( 9L, "a9" , 8L, 12L) %>%
  sch_add_activity(10L, "a10", 3L, 12L) %>%
  sch_add_activity(11L, "a11", 3L, 12L) %>%
  sch_add_activity(12L, "a12", 0L) %>%
  sch_plan()
sch_is_redundant(sch, 2, 5) # FALSE
sch_is_redundant(sch, 2, 12) # TRUE
```

sch_new *New Schedule*

Description

Create a new schedule without any information. The new schedule contains the structure to include activities and relations.

Usage

```
sch_new()
```

Value

A list with schedule definition.

See Also

sch_reference(), sch_add_activities(), sch_duration(), sch_xy_gantt_matrix(), sch_plan(), sch_add_relations(), sch_validate(), sch_non_critical_activities(), sch_title().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id = c(1L, 2L, 3L, 4L),
    name = c("A", "B", "C", "D"),
    duration = c(3L, 4L, 9L, 1L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 2L),
    to   = c(2L, 3L, 4L)
  ) %>%
  sch_plan()
sch_duration(sch) # 16
```

sch_non_critical_activities *Non Critical Activities*

Description

Return a tibble with all non critical activities of a schedule in an insertion order.

Usage

```
sch_non_critical_activities(sch)
```

Arguments

sch	A schedule object.

Value

A tibble with non critical activities.

See Also

sch_get_activity(), sch_add_activity(), sch_activities(), sch_critical_activities(), sch_has_any_activity(), sch_nr_activities(), sch_add_activities().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2,3,4) %>%
  sch_add_activity(  2L, "a2" , 4L, 5) %>%
  sch_add_activity(  3L, "a3" , 9L, 10) %>%
  sch_add_activity(  4L, "a4" , 1L, 6) %>%
  sch_add_activity(  5L, "a5" , 4L, 9) %>%
  sch_add_activity(  6L, "a6" , 5L, 7) %>%
  sch_add_activity(  7L, "a7" , 1L, 8,11) %>%
  sch_add_activity(  8L, "a8" , 7L, 12) %>%
  sch_add_activity(  9L, "a9" , 8L, 12) %>%
  sch_add_activity( 10L, "a10", 3L, 12) %>%
  sch_add_activity( 11L, "a11", 3L, 12) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_non_critical_activities(sch)
```

---

sch_non_critical_relations

Non Critical Relations

Description

Return a tibble with non critical relations of a schedule in topological order.

Usage

```
sch_non_critical_relations(sch, order = "topological")
```

Arguments

sch	A schedule object.
order	Indicates the order of relations: • "topological": The relations tibble is in topological order. • "insert": The relations tibble is in insert order.

Value

A tibble with non critical relations.

See Also

sch_relations(), sch_add_activities(), sch_has_any_relation(), sch_topoi_tf(), sch_gantt_matrix(), sch_activities(), sch_topoi_la(), sch_add_relations(), sch_topoi_sp(), sch_non_critical_activities(), sch_topoi_ad(), sch_nr_relations(), sch_critical_relations().

Examples

```
sch <- sch_new() %>%
  sch_title("Project 3: Old Carriage House Renovation") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
  first comes the theory, then the practice. Gent: Springer, 2014, p. 11") %>%
  sch_add_activity( 1L, "a1" , 2L) %>%
  sch_add_activity( 2L, "a2" , 2L) %>%
  sch_add_activity( 3L, "a3" , 4L) %>%
  sch_add_activity( 4L, "a4" , 3L) %>%
  sch_add_activity( 5L, "a5" , 4L) %>%
  sch_add_activity( 6L, "a6" , 1L) %>%
  sch_add_activity( 7L, "a7" , 1L) %>%
  sch_add_activity( 8L, "a8" , 1L) %>%
  sch_add_activity( 9L, "a9" , 1L) %>%
  sch_add_activity(10L, "a10", 1L) %>%
  sch_add_activity(11L, "a11", 3L) %>%
  sch_add_activity(12L, "a12", 2L) %>%
  sch_add_activity(13L, "a13", 1L) %>%
  sch_add_activity(14L, "a14", 1L) %>%
  sch_add_activity(15L, "a15", 2L) %>%
  sch_add_activity(16L, "a16", 1L) %>%
  sch_add_activity(17L, "a17", 1L) %>%
  sch_add_relation(14L, 15L) %>%
  sch_add_relation( 9L, 10L) %>%
  sch_add_relation( 2L,  3L) %>%
  sch_add_relation( 8L, 10L) %>%
  sch_add_relation(10L, 13L) %>%
  sch_add_relation( 5L,  6L) %>%
  sch_add_relation(11L, 12L) %>%
  sch_add_relation(15L, 16L) %>%
  sch_add_relation( 6L,  8L) %>%
  sch_add_relation( 3L,  4L) %>%
  sch_add_relation(16L, 17L) %>%
```

```
  sch_add_relation( 6L,  7L) %>%
  sch_add_relation(10L, 11L) %>%
  sch_add_relation(13L, 14L) %>%
  sch_add_relation( 4L,  5L) %>%
  sch_add_relation( 7L, 10L) %>%
  sch_add_relation(12L, 15L) %>%
  sch_add_relation( 6L,  9L) %>%
  sch_add_relation( 1L,  2L) %>%
  sch_plan()
# In "topological" order.
sch_non_critical_relations(sch)
# In "insert" order.
sch_non_critical_relations(sch, order = "insert")
```

`sch_nr_activities` *Nr. of Activities*

Description

Number of activities in a schedule as an integer value.

Usage

```
sch_nr_activities(sch)
```

Arguments

`sch`	A schedule object.

Value

A integer value indicating the number of activities.

See Also

`sch_add_activity()`, `sch_nr_relations()`, `sch_add_activities()`, `sch_activities()`, `sch_change_activities_du`
`sch_critical_activities()`, `sch_get_activity()`, `sch_has_any_relation()`.

Examples

```
sch <- sch_new()
sch_nr_activities(sch) # 0

sch <- sch_new() %>%
  sch_add_activity(1L, "Only one", 0L) %>%
  sch_plan()
sch_nr_activities(sch) # 1
```

sch_nr_relations *Nr. of Relations*

Description

Number of relations in a schedule as an integer value.

Usage

```
sch_nr_relations(sch)
```

Arguments

sch	A schedule object.

Value

A integer value indicating the number of relations.

See Also

sch_relations(), sch_topoi_la(), sch_topoi_tf(), sch_all_successors(), sch_topoi_ad(), sch_nr_activities(), sch_topoi_sp(), sch_has_any_relation(), sch_all_predecessors(), sch_add_relations(), sch_has_any_activity().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2,3,4) %>%
  sch_add_activity(  2L, "a2" , 4L, 5) %>%
  sch_add_activity(  3L, "a3" , 9L, 10) %>%
  sch_add_activity(  4L, "a4" , 1L, 6) %>%
  sch_add_activity(  5L, "a5" , 4L, 9) %>%
  sch_add_activity(  6L, "a6" , 5L, 7) %>%
  sch_add_activity(  7L, "a7" , 1L, 8,11) %>%
  sch_add_activity(  8L, "a8" , 7L, 12) %>%
  sch_add_activity(  9L, "a9" , 8L, 12) %>%
  sch_add_activity( 10L, "a10", 3L, 12) %>%
  sch_add_activity( 11L, "a11", 3L, 12) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_nr_relations(sch) # 14
```

sch_plan *Plan Schedule*

Description

Perform schedule plan: execute topological sort and critical path calculation. All information about critical path are calculated.

Usage

```
sch_plan(sch)
```

Arguments

sch	A schedule object.

Value

A schedule with critical path calculated.

See Also

sch_gantt_matrix(), sch_duration(), sch_reference(), sch_add_activities(), sch_has_any_activity(), sch_title(), sch_new(), sch_add_relations().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id = c(1L, 2L, 3L, 4L),
    name = c("A", "B", "C", "D"),
    duration = c(3L, 4L, 9L, 1L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 2L),
    to   = c(2L, 3L, 4L)
  ) %>%
  sch_plan()
sch_duration(sch) # 16
```

sch_predecessors *Predecessors*

Description

List the direct predecessors of an activity.

Usage

```
sch_predecessors(sch, id)
```

Arguments

sch	A schedule object.
id	Activity id to be listed.

Value

A vector with all activities ids.

See Also

sch_gantt_matrix(), sch_all_successors(), sch_is_redundant(), sch_all_predecessors(), sch_successors(), sch_activities(), sch_relations().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity( 2L, "a2" , 4L,  5L, 12L) %>%
  sch_add_activity( 3L, "a3" , 9L, 10L) %>%
  sch_add_activity( 4L, "a4" , 1L,  6L) %>%
  sch_add_activity( 5L, "a5" , 4L,  9L) %>%
  sch_add_activity( 6L, "a6" , 5L,  7L) %>%
  sch_add_activity( 7L, "a7" , 1L,  8L,11L) %>%
  sch_add_activity( 8L, "a8" , 7L, 12L) %>%
  sch_add_activity( 9L, "a9" , 8L, 12L) %>%
  sch_add_activity(10L, "a10", 3L, 12L) %>%
  sch_add_activity(11L, "a11", 3L, 12L) %>%
  sch_add_activity(12L, "a12", 0L) %>%
  sch_plan()
sch_predecessors(sch, 2) # nothing
sch_predecessors(sch, 7) # 6
sch_predecessors(sch, 10) # 3
```

sch_reference *Reference*

Description

A reference from project origin, for example, a book, a paper, a corporation, or nothing.

Usage

```
sch_reference(sch, new_value = NULL)
```

Arguments

sch	A schedule object.
new_value	A new reference.

Value

- A schedule object with new reference.
- A reference.

See Also

sch_new(), sch_activities(), sch_relations(), sch_title(), sch_plan(), sch_duration(), sch_validate().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id = c(1L, 2L, 3L, 4L),
    name = c("A", "B", "C", "D"),
    duration = c(3L, 4L, 9L, 1L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 2L),
    to   = c(2L, 3L, 4L)
  ) %>%
  sch_plan()

sch_reference(sch) # empty
sch %<>% sch_reference("This schedule is from...")
sch_reference(sch)
```

`sch_relations` *Relations*

Description

Return a tibble with all relations of a schedule in topological order. These are the main information calculated by CPM.

Usage

```
sch_relations(sch, order = "topological")
```

Arguments

`sch`	A schedule object.
`order`	Indicates the order of relations: • `"topological"`**:** The relations tibble is in topological order. • `"insert"`**:** The relations tibble is in insert order.

Details

The tibble is formed by following structure:

- **from:** Predecessor activity id from a relation.
- **to:** Successor activity id to a relation.
- **type:** The type of relation between activities. Its value may be: FS, FF, SS, SF.
- **lag:** The time period between activity predecessor and activity successor activity
- **critical:** A critical relation is formed by two activity critical: predecessor and successor. `TRUE` indicates it is critical; `FALSE` indicates it is not critical.
- **ord:** Indicates de order that the relation was added in the schedule.
- **i_from:** It is the index of predecessor activity in the activities tibble.
- **i_to:** It is the index of successor activity in the activities tibble.

Value

A tibble with relations.

See Also

`sch_add_activities()`, `sch_has_any_relation()`, `sch_topoi_tf()`, `sch_gantt_matrix()`, `sch_activities()`, `sch_add_relations()`, `sch_topoi_sp()`, `sch_topoi_la()`, `sch_non_critical_activities()`, `sch_topoi_ad()`, `sch_nr_relations()`.

Examples

```
sch <- sch_new() %>%
  sch_title("Project 3: Old Carriage House Renovation") %>%
  sch_reference(
    "VANHOUCKE, Mario. Integrated project management and control:
  first comes the theory, then the practice. Gent: Springer, 2014, p. 11") %>%
  sch_add_activity( 1L, "a1" , 2L) %>%
  sch_add_activity( 2L, "a2" , 2L) %>%
  sch_add_activity( 3L, "a3" , 4L) %>%
  sch_add_activity( 4L, "a4" , 3L) %>%
  sch_add_activity( 5L, "a5" , 4L) %>%
  sch_add_activity( 6L, "a6" , 1L) %>%
  sch_add_activity( 7L, "a7" , 1L) %>%
  sch_add_activity( 8L, "a8" , 1L) %>%
  sch_add_activity( 9L, "a9" , 1L) %>%
  sch_add_activity(10L, "a10", 1L) %>%
  sch_add_activity(11L, "a11", 3L) %>%
  sch_add_activity(12L, "a12", 2L) %>%
  sch_add_activity(13L, "a13", 1L) %>%
  sch_add_activity(14L, "a14", 1L) %>%
  sch_add_activity(15L, "a15", 2L) %>%
  sch_add_activity(16L, "a16", 1L) %>%
  sch_add_activity(17L, "a17", 1L) %>%
  sch_add_relation(14L, 15L) %>%
  sch_add_relation( 9L, 10L) %>%
  sch_add_relation( 2L,  3L) %>%
  sch_add_relation( 8L, 10L) %>%
  sch_add_relation(10L, 13L) %>%
  sch_add_relation( 5L,  6L) %>%
  sch_add_relation(11L, 12L) %>%
  sch_add_relation(15L, 16L) %>%
  sch_add_relation( 6L,  8L) %>%
  sch_add_relation( 3L,  4L) %>%
  sch_add_relation(16L, 17L) %>%
  sch_add_relation( 6L,  7L) %>%
  sch_add_relation(10L, 11L) %>%
  sch_add_relation(13L, 14L) %>%
  sch_add_relation( 4L,  5L) %>%
  sch_add_relation( 7L, 10L) %>%
  sch_add_relation(12L, 15L) %>%
  sch_add_relation( 6L,  9L) %>%
  sch_add_relation( 1L,  2L) %>%
  sch_plan()
# In "topological" order.
sch_relations(sch)
# In "insert" order.
sch_relations(sch, order = "insert")
```

`sch_successors` *Successors*

Description

List the direct successors from an activity.

Usage

```
sch_successors(sch, id)
```

Arguments

sch	A schedule object.
id	Activity id to be listed.

Value

A vector with all activities ids.

See Also

sch_relations(), sch_all_predecessors(), sch_activities(), sch_gantt_matrix(), sch_predecessors(), sch_is_redundant(), sch_all_successors().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity( 2L, "a2" , 4L,  5L, 12L) %>%
  sch_add_activity( 3L, "a3" , 9L, 10L) %>%
  sch_add_activity( 4L, "a4" , 1L,  6L) %>%
  sch_add_activity( 5L, "a5" , 4L,  9L) %>%
  sch_add_activity( 6L, "a6" , 5L,  7L) %>%
  sch_add_activity( 7L, "a7" , 1L,  8L,11L) %>%
  sch_add_activity( 8L, "a8" , 7L, 12L) %>%
  sch_add_activity( 9L, "a9" , 8L, 12L) %>%
  sch_add_activity(10L, "a10", 3L, 12L) %>%
  sch_add_activity(11L, "a11", 3L, 12L) %>%
  sch_add_activity(12L, "a12", 0L) %>%
  sch_plan()

sch_successors(sch, 2) # 5, 12
sch_successors(sch, 7) # 8, 11
sch_successors(sch, 10) # 12
```

sch_title *Title*

Description

A title for project identification. It depends on user of the class. It is used to set or get project's title.

Usage

```
sch_title(sch, new_value)
```

Arguments

sch	A schedule object.
new_value	A new title.

Value

- A schedule object with new title.
- A title.

See Also

sch_relations(), sch_plan(), sch_new(), sch_validate(), sch_activities(), sch_reference(), sch_duration().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id = c(1L, 2L, 3L, 4L),
    name = c("A", "B", "C", "D"),
    duration = c(3L, 4L, 9L, 1L)
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 2L),
    to   = c(2L, 3L, 4L)
  ) %>%
  sch_plan()

sch_title(sch) # empty
sch %<>% sch_title("New title")
sch_title(sch)
```

sch_topoi_ad *AD Activity Distribution Topological Indicator*

Description

Measures the distribution of the activities over the levels. If AD is approximately equal zero, each level has same numbers of activities. Otherwise, if AD is equal one, the quantity of each level is not uniformly distributed.

Usage

```
sch_topoi_ad(sch)
```

Arguments

sch	A schedule object.

Value

A number between 0 and 1, inclusive.

See Also

sch_topoi_sp(), sch_topoi_la(), sch_topoi_tf(), sch_xy_gantt_matrix(), sch_add_relations(), sch_add_activities(), sch_relations(), sch_activities().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2L,3L,4L) %>%
  sch_add_activity(  2L, "a2" , 4L, 5L) %>%
  sch_add_activity(  3L, "a3" , 9L, 10L) %>%
  sch_add_activity(  4L, "a4" , 1L, 6L) %>%
  sch_add_activity(  5L, "a5" , 4L, 9L) %>%
  sch_add_activity(  6L, "a6" , 5L, 7L) %>%
  sch_add_activity(  7L, "a7" , 1L, 8L,11L) %>%
  sch_add_activity(  8L, "a8" , 7L, 12L) %>%
  sch_add_activity(  9L, "a9" , 8L, 12L) %>%
  sch_add_activity( 10L, "a10", 3L, 12L) %>%
  sch_add_activity( 11L, "a11", 3L, 12L) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_topoi_ad(sch) # 0.4
```

sch_topoi_la *LA Length of Arcs Topological Indicator*

Description

Measures the presence of long arcs based on the difference between the progressive level of the end activity and the start node of each relation. If LA is approximately equal zero, the progressive level between activities is as far as possible. Otherwise, if LA is equal one, the relation distance are one.

Usage

```
sch_topoi_la(sch)
```

Arguments

sch	A schedule object.

Value

A number between 0 and 1, inclusive.

See Also

sch_topoi_sp(), sch_add_relations(), sch_topoi_ad(), sch_relations(), sch_xy_gantt_matrix(), sch_activities(), sch_topoi_tf(), sch_add_activities().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2L,3L,4L) %>%
  sch_add_activity(  2L, "a2" , 4L, 5L) %>%
  sch_add_activity(  3L, "a3" , 9L, 10L) %>%
  sch_add_activity(  4L, "a4" , 1L, 6L) %>%
  sch_add_activity(  5L, "a5" , 4L, 9L) %>%
  sch_add_activity(  6L, "a6" , 5L, 7L) %>%
  sch_add_activity(  7L, "a7" , 1L, 8L,11L) %>%
  sch_add_activity(  8L, "a8" , 7L, 12L) %>%
  sch_add_activity(  9L, "a9" , 8L, 12L) %>%
  sch_add_activity( 10L, "a10", 3L, 12L) %>%
  sch_add_activity( 11L, "a11", 3L, 12L) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_topoi_la(sch) # 0.07692308
```

sch_topoi_sp *SP Serial or Parallel Topological Indicator*

Description

Shows the closeness of a network to a serial or parallel graph. As the network becomes serial, the SP increase, until one; As the network becomes parallel, the SP decrease until zero.

Usage

```
sch_topoi_sp(sch)
```

Arguments

sch	A schedule object.

Value

A number between 0 and 1, inclusive.

See Also

sch_topoi_tf(), sch_activities(), sch_topoi_ad(), sch_xy_gantt_matrix(), sch_relations(), sch_topoi_la(), sch_add_activities(), sch_add_relations().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2L,3L,4L) %>%
  sch_add_activity(  2L, "a2" , 4L, 5L) %>%
  sch_add_activity(  3L, "a3" , 9L, 10L) %>%
  sch_add_activity(  4L, "a4" , 1L, 6L) %>%
  sch_add_activity(  5L, "a5" , 4L, 9L) %>%
  sch_add_activity(  6L, "a6" , 5L, 7L) %>%
  sch_add_activity(  7L, "a7" , 1L, 8L,11L) %>%
  sch_add_activity(  8L, "a8" , 7L, 12L) %>%
  sch_add_activity(  9L, "a9" , 8L, 12L) %>%
  sch_add_activity( 10L, "a10", 3L, 12L) %>%
  sch_add_activity( 11L, "a11", 3L, 12L) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_topoi_sp(sch) # 0.4545455
```

sch_topoi_tf *TF Topological Float Indicator*

Description

Measures the topological float of each activity. If TF = 0, there is no float between activities. If TF = 1, there is float between activities and they be shift without affecting other activities.

Usage

```
sch_topoi_tf(sch)
```

Arguments

sch	A schedule object.

Value

A number between 0 and 1, inclusive.

See Also

sch_topoi_ad(), sch_add_activities(), sch_add_relations(), sch_xy_gantt_matrix(), sch_topoi_la(), sch_activities(), sch_relations(), sch_topoi_sp().

Examples

```
sch <- sch_new() %>%
  sch_title("Fictitious Project Example") %>%
  sch_reference("VANHOUCKE, Mario. Measuring time:
  improving project performance using earned value management.
  Gent: Springer, 2009, p. 18") %>%
  sch_add_activity(  1L, "a1" , 0L, 2L,3L,4L) %>%
  sch_add_activity(  2L, "a2" , 4L, 5L) %>%
  sch_add_activity(  3L, "a3" , 9L, 10L) %>%
  sch_add_activity(  4L, "a4" , 1L, 6L) %>%
  sch_add_activity(  5L, "a5" , 4L, 9L) %>%
  sch_add_activity(  6L, "a6" , 5L, 7L) %>%
  sch_add_activity(  7L, "a7" , 1L, 8L,11L) %>%
  sch_add_activity(  8L, "a8" , 7L, 12L) %>%
  sch_add_activity(  9L, "a9" , 8L, 12L) %>%
  sch_add_activity( 10L, "a10", 3L, 12L) %>%
  sch_add_activity( 11L, "a11", 3L, 12L) %>%
  sch_add_activity( 12L, "a12", 0L) %>%
  sch_plan()
sch_topoi_tf(sch) # 0.2333333
```

sch_validate *Validate Schedule*

Description

Validate your schedule in terms of structure: cannot have duplicated activity id, all 'from' and 'to' relation id must exist in activities tibble and cannot have duplicated relation. This function is called by `sch_plan(plan)`. If there is an error, the schedule cannot be calculated.

Usage

```
sch_validate(sch)
```

Arguments

`sch`	A schedule object.

Details

There are two forms to use this function:

- The first is automatic, when you call `sch_plan(plan)`, the validation is called for you.
- The second, you can call `sch_plan(plan)` with your schedule, before plan, to see all error in your schedule.

In both way, the calculation schedule is stopped, because there is some error. To see the errors, you call the `sch_plan(plan)` function to find how to correct the errors.

The result of `sch_plan(plan)` is a lista with a lot of information about the error. The structure is:

- `is_valid`: A logical value that indicates if the schedule structure is valid.
  - `TRUE`: The schedule structure is NOT valid.
  - `FALSE`: The schedule structure is valid.
- `is_error_with_activities`: A logical value that indicates if there is any error with activities.
  - `TRUE`: There is any error with activities.
  - `FALSE`: There is NOT any error with activities.
- `is_error_with_relations`: A logical value that indicates if there is any error with relations.
  - `TRUE`: There is any error with relations.
  - `FALSE`: There is NOT any error with relations.
- `is_error_with_dag`: A logical value that indicates if there is any error with `igraph` object tha support the schedule.
  - `TRUE`: There is any error with igraph.
  - `FALSE`: There is NOT any error with igraph.
- `activities_errors`: A tibble that list the activities errors:

  - id: activity's id of the error
    * error: the error.
    * to_fix: suggestion of how to fix the error.
- relations_errors:
  - from: Predecessor activity id 'from' of the error.
    * to: Successor activity id from a relation.
    * error: the error.
    * to_fix: suggestion of how to fix the error.
- dag_errors: Error identified by igraph object.
- dag_igraph: The igraph object that is totally validated.

Attention: You must identify and correct all errors before call sch_plan(plan)!

Value

A list object with a description of all error.

See Also

sch_add_relation(), sch_relations(), sch_add_relations(), sch_add_activities(), sch_add_activity(), sch_activities(), sch_plan().

---

sch_xy_gantt_matrix *XY Gantt Matrix*

Description

Transform a Gantt matrix into x, y coordinates and the weight one. Each point greater than zero in a Gantt matrix becomes a x, y coordinate. **Atention:** the schedule must be planned with the function sch_plan().

Usage

```
sch_xy_gantt_matrix(sch, gantt = NULL)
```

Arguments

sch	A schedule object.
gantt	A Gantt Matrix. If it is not informed, it will use gantt_matrix() before this function.

Value

A matrix with three columns: x, y and weight.

See Also

sch_relations(), sch_activities(), sch_add_activities(), sch_add_relations(), sch_add_relation(), sch_plan(), sch_gantt_matrix().

Examples

```
sch <- sch_new() %>%
  sch_add_activities(
    id        = c( 1L,  2L,  3L,  4L),
    name      = c("A", "B", "C", "D"),
    duration  = c( 2L,  3L,  1L, 2L )
  ) %>%
  sch_add_relations(
    from = c(1L, 2L, 4L, 4L),
    to   = c(3L, 3L, 1L, 2L)
  ) %>%
  sch_plan()

sch_duration(sch)

xyw <- sch_xy_gantt_matrix(sch)
xyw
plot(xyw[, 1:2])
```

Index