Package ‘understandBPMN’

July 22, 2025

Type Package

Title Calculator of Understandability Metrics for BPMN

Version 1.1.1

Description Calculate several understandability metrics of BPMN models. BPMN stands for business process modelling notation and is a language for expressing business processes into business process diagrams. Examples of these understandability metrics are: average connector degree, maximum connector degree, sequentiality, cyclicity, diameter, depth, token split, control flow complexity, connector mismatch, connector heterogeneity, separability, structuredness and cross connectivity. See R documentation and paper on metric implementation included in this package for more information concerning the metrics.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports XML, dplyr, purrr, tidyr, tibble, Rcpp (>= 0.12.15), devtools, usethis, R.utils

RoxygenNote 6.1.1

Depends R(>= 2.10.0)

Suggests knitr, rmarkdown, testthat

NeedsCompilation yes

Author Jonas Lieben [aut],
Gert Janssenswillen [cre]

Maintainer Gert Janssenswillen <gert.janssenswillen@uhasselt.be>

Repository CRAN

Date/Publication 2019-09-27 11:30:03 UTC

Contents

`activity_multiple_times_executed`
activity sometimes multiple times executed

Description

This functions returns true or false on whether or not an activity is sometimes multiple times executed This can be useful for measuring the understandability using behavioral profiles.

Usage

```
activity_multiple_times_executed(repetition_and_path_log, xml_internal_doc,
  activity, direct_parallel)
```

Arguments

`repetition_and_path_log`	repetition and path log list object created by the function create_repetition_and_path_log
`xml_internal_doc`	document object created using the create_internal_document function
`activity`	the activity name
`direct_parallel`	a table containing the direct and parallel relations

Value

a boolean value indicating whether it is true that an activity can be executed multiple times in the same path

Examples

```
## Not run: activity_multiple_times_executed(log, doc, "A")
```

`activity_names_repetitions` *activity names repetitions*

Description

This functions returns a list containing the repetitions with their respective activity names This can be useful for measuring the understandability using behavioral profiles.

Usage

```
activity_names_repetitions(repetition_and_path_log, xml_internal_doc)
```

Arguments

`repetition_and_path_log`	repetition and path log list object created by the function create_repetition_and_path_log
`xml_internal_doc`	document object created using the create_internal_document function

Value

a list containing the repetitions with their respective activity names

Examples

```
## Not run: activity_multiple_times_executed(log, doc, "A")
```

---

avg_connector_degree *Average connector degree*

Description

Average connector degree is defined as the average incoming and outgoing sequence flows of all gateways and activities with at least two incoming or outgoing sequence flows

Usage

```
avg_connector_degree(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the average connector degree

Examples

```
avg_connector_degree(file_path)
```

---

calculate_metrics *A calculation function for all metrics*

Description

Creation object containing all metrics, which are : the number of empty sequence flows, the number of duplicate tasks, the number of data objects, the number of pools, the number of swimlanes, the number of message flows, the density, the coefficient of network connectivity, the average connector degree, the maximum connector degree, the sequentiality, the cyclicity, the diameter, the depth, the token_split, the control flow complexity, the connector mismatch, the connector heterogeneity and the crs

Usage

```
calculate_metrics(file_path, cross_connectivity_metric = TRUE,
  signavio = FALSE, generate_new_path_log = FALSE)
```

Arguments

`file_path`	file path of the BPMN file and
`cross_connectivity_metric`	a param indicating whether cross_connectivity shall be calculated as well
`signavio`	boolean which indicates whether the file stems from signavio
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples

Value

a tibble with one row and for each metric a column

Examples

```
calculate_metrics(file_path, generate_new_path_log = TRUE)
```

`coefficient_network_connectivity` *Coefficient of network connectivity*

Description

Coefficient of network connectivity is defined as the number of sequence flows divided by the size

Usage

```
coefficient_network_connectivity(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the coefficient of network connectivity

Examples

```
coefficient_network_connectivity(file_path)
```

`cognitive_weight` *Cognitive weights*

Description

Cognitive weight is defined as a weighted sum of gateways and activities

Usage

```
cognitive_weight(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the control flow complexity

Examples

```
cognitive_weight(file_path)
```

`connectivity_level_between_pools`
The connectivity level between pools

Description

The connectivity level between pools is the number of message flows over the number of pools

Usage

```
connectivity_level_between_pools(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the connectivity level between pools

Examples

```
connectivity_level_between_pools(file_path)
```

`connector_heterogeneity` *Connector heterogeneity*

Description

Connector heterogeneity is defined as the sum of minus - p times the log of p of all gateways. p is defined as the number of a particular type of gateway divided by all gateways.

Usage

```
connector_heterogeneity(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the connector heterogeneity

Examples

```
connector_heterogeneity(file_path)
```

`connector_mismatch` *Connector mismatch*

Description

Connector mismatch is the absolute value of the difference between split gateways and join gateways for each type of gateway, ie parallel, exclusive, inclusive, complex and event based gateways

Usage

```
connector_mismatch(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the connector mismatch

Examples

```
connector_mismatch(file_path)
```

`control_flow_complexity` *Control flow complexity*

Description

Control flow complexity is defined as the sum of the outgoing of exclusive gateways, the number of parallel gateways and two to the power of all outgoing sequence flows of the inclusive gateways

Usage

```
control_flow_complexity(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the control flow complexity

Examples

```
control_flow_complexity(file_path)
```

`coupling_metric` *Coupling metric*

Description

Coupling metric is defined as the sum of the number of activities, AND-splits and a weighterd number of OR and XOR splits

Usage

```
coupling_metric(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the control flow complexity

Examples

```
coupling_metric(file_path)
```

`create_internal_doc` *A function for creating internal documents*

Description

Is used for creating xml documents which nearly every function of this package needs as an input

Usage

```
create_internal_doc(bpmn_file, signavio = FALSE)
```

Arguments

`bpmn_file`	file path of the BPMN file
`signavio`	boolean which indicates whether the file stems from signavio

Value

an object containing the xml document

Examples

```
create_internal_doc(file_path)
```

`create_path_and_repetition_log` *Path and repetition log*

Description

This function returns a list with four or three nested list objects: - One for the paths: Assumption: if a path contains a loop, the path contains one repetition (so two times) of the execution of this loop Assumption: there is no difference made between the type of gateways. So the path log is not a path log according to the definition found in the literature, but more a kind of a path log Assumption: for each split and join in the log, an extra element is added with the name "split" or "join" - One list object for the loops (repetitions) which start with a join and end with a join - One list object for the loops (repetitions) which start with a split and end with a split (- One list for the paths in which all gateways have a certain type)

Usage

```
create_path_and_repetition_log(file_path,
  add_path_log_for_structuredness = TRUE, signavio = FALSE)
```

Arguments

`file_path`	internal document containing an xml
`add_path_log_for_structuredness`	a boolean value indicating whether the structured path log should be added. Is standard TRUE
`signavio`	boolean which indicates whether the file stems from signavio

Value

a list containing the path log, a list of repetitions starting with join, a list of repetitions starting with split, (optional: structured path log)

Examples

```
create_path_and_repetition_log(file_path)
```

`cross_connectivity` *Cross Connectivity*

Description

The cross-connectivity metric that measures the strength of the links between process model elements. The definition of this new metric builds on the hypothesis that process models are easier understood and contain less errors if they have a high cross-connectivity. The metric is calculated based on the creation of a data frame containing the values of all connections

Usage

```
cross_connectivity(file_path, signavio = FALSE,
  path_log_already_created = FALSE, generate_new_path_log = FALSE,
  time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples

`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the cross connectivity of a model

Examples

```
cross_connectivity(file_path, generate_new_path_log = TRUE)
```

cyclicity *Cyclicity*

Description

Cyclicity is defined as the number of nodes on a cycle divided by the total number of nodes

Usage

```
cyclicity(file_path, signavio = FALSE,
  path_log_already_created = FALSE, generate_new_path_log = FALSE,
  time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the cyclicity

Examples

```
cyclicity(file_path, generate_new_path_log = TRUE)
```

`cyclomatic_metric` *Cyclomatic metric of McCabe*

Description

Cyclomatic metric takes into account the behavioral complexity of a process model. It is calculated by taking the number of activities minus the number of events, gateways and connector activities plus the number of strongly connected components. The number of strongly connected components is calculated by taking the number of exclusive gateways at depth level zero, when the depth is calculated only including exclusive gateways

Usage

```
cyclomatic_metric(file_path, signavio = FALSE,
  path_log_already_created = FALSE, generate_new_path_log = FALSE,
  time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the coefficient of network connectivity

Examples

```
cyclomatic_metric(file_path, generate_new_path_log = TRUE)
```

`density_process_model` *Density*

Description

Density is defined as the number of sequence flows divided by the size times the size minus one

Usage

```
density_process_model(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the density

Examples

```
density_process_model(file_path)
```

`depth` *Depth*

Description

Depth is defined as the the nesting of the process model. If there is a split gateway, the depth is increased with one. If there is a join gateway, the depth is decreased with one. The cumulative sum is taken and the maximum of the cumulative sum is calculated for each path. The nesting depth is the maximum of each path value

Usage

```
depth(file_path, signavio = FALSE, path_log_already_created = FALSE,
  generate_new_path_log = FALSE, time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the depth

Examples

```
depth(file_path, generate_new_path_log = TRUE)
```

`diameter` *Diameter*

Description

Length of longest path, in practice the length of longest path. The assumption is made that one repetition for each loop is allowed and these repetitions count as well for the diameter

Usage

```
diameter(file_path, signavio = FALSE, path_log_already_created = FALSE,
  generate_new_path_log = FALSE, time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the diameter

Examples

```
diameter(file_path, generate_new_path_log = TRUE)
```

`direct_parallel_relations` *Direct and parallel relations*

Description

This functions returns a table containing all direct and parallel relations between activities. The table contains five columns: - the two first represent activity ids - the third represents the type of relations, which is parallel or direct - the last two columns are the corresponding activity names for the first two columns

Usage

```
direct_parallel_relations(repetition_and_path_log, xml_internal_doc)
```

Arguments

`repetition_and_path_log`	repetition and path log list object created by the function create_repetition_and_path_log
`xml_internal_doc`	document object created using the create_internal_document function

Value

a table as described in the description

Examples

```
## Not run: direct_parallel_relations(repetition_and_path_log, xml_internal_doc)
```

filtered_path_log_parallel

Filter path log with only traces containing the parallel gateway together with the relevant activity

Description

This functions returns a path log with no traces with a parallel gateway of which the given activity is part but not included

Usage

```
filtered_path_log_parallel(structured_path_log, xml_internal_doc,
  activity_name)
```

Arguments

structured_path_log	repetition and path log list object created by the function create_repetition_and_path_log
xml_internal_doc	document object created using the create_internal_document function
activity_name	name of the activity for the relevant filtering

Value

the filtered path log

Examples

```
## Not run: direct_parallel_relations(repetition_and_path_log, xml_internal_doc)
```

max_connector_degree

Maximum connector degree

Description

Maximum connector degree is defined as the gateway or activity with the most incoming and outgoing sequence flows

Usage

```
max_connector_degree(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the maximum connector degree

Examples

```
max_connector_degree(file_path)
```

`n_data_objects` *Data Objects*

Description

The number of data objects includes all data objects and data stores of a BPMN diagram

Usage

```
n_data_objects(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of data objects

Examples

```
n_data_objects(file_path)
```

`n_duplicate_tasks` *Duplicate tasks*

Description

Duplicate tasks are tasks which share the same name with other tasks

Usage

```
n_duplicate_tasks(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of duplicate tasks

Examples

```
n_duplicate_tasks(file_path)
```

`n_empty_sequence_flows` *Empty sequence flows*

Description

Empty sequence flow is defined as a flow which connects a split parallel gateway with a join parallel gateway without any tasks in between

Usage

```
n_empty_sequence_flows(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of empty sequence flows

Examples

```
n_empty_sequence_flows(file_path)
```

`n_message_flows` *Number of message flows*

Description

Number of message flows. Message flows are used for communication between processes and link message events

Usage

```
n_message_flows(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of message flows

Examples

```
n_message_flows(file_path)
```

`n_pools` *Number of pools*

Description

Number of pools in the process models. A pool represents an organisation or an entity

Usage

```
n_pools(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of pools

Examples

```
n_pools(file_path)
```

---

n_swimlanes *Number of swimlanes*

Description

Number of swimlanes in the pools. A swimlane represents a person, role or team

Usage

```
n_swimlanes(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the number of swimlanes

Examples

```
n_swimlanes(file_path)
```

---

separability *Separability*

Description

A cut vertex is a node which if removed, splits the diagram into two pieces The consequence is that elements which are part of each path can be defined as a cut vertex Separability is defined as the number of cut vertices divided by (the size of the model - 2)

Usage

```
separability(file_path, signavio = FALSE,
  path_log_already_created = FALSE, generate_new_path_log = FALSE,
  time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the separability

Examples

```
separability(file_path, generate_new_path_log = TRUE)
```

`sequentiality` *Sequentiality*

Description

Sequentiality is defined as the number of sequence flows connecting two tasks divided by the total number of sequence flows

Usage

```
sequentiality(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the sequentiality

Examples

```
sequentiality(file_path)
```

size_process_model *Size*

Description

The size of a model is the number of tasks, gateways and events

Usage

```
size_process_model(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the size

Examples

```
size_process_model(file_path)
```

some_traces_without_activity *activity sometimes not in traces*

Description

This functions returns true or false on whether or not an activity is sometimes not part of a trace This can be useful for measuring the understandability using behavioral profiles.

Usage

```
some_traces_without_activity(repetition_and_path_log, xml_internal_doc,
  activity)
```

Arguments

`repetition_and_path_log`	repetition and path log list object created by the function create_repetition_and_path_log
`xml_internal_doc`	document object created using the create_internal_document function
`activity`	the activity name

Value

a boolean value indicating whether it is true on whether or not an activity is sometimes not part of a trace

Examples

```
## Not run: some_traces_without_activity(log, doc, "A")
```

`structuredness` *Structuredness*

Description

Structuredness measures to which extent the process model can be divided into block structured structures (matching gateways) Calculation: 1 - size of reduced process model / size of normal process model To get the reduced process model, the following rules are applied: -removal of trivial constructs (one incoming and one outgoing sequence flow) -removal of matching gateways (for loops, this means first a join then a split, for all other gateways, it's the other way around) -loops with other than XOR-gateways and non-matching gateways are kept -gateways which are the consequence of multiple start or end events are removed

Usage

```
structuredness(file_path, signavio = FALSE,
  path_log_already_created = FALSE, generate_new_path_log = FALSE,
  time_to_generate_path_log = 1500)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio
`path_log_already_created`	boolean which indicates whether the path log has already been created before or not. When you are not sure, it is best to use the standard which is false
`generate_new_path_log`	used when it is not possible to save the path log such as with the Rapid miner or in unit tests and examples
`time_to_generate_path_log`	time which is the maximum time to generate a new path log in seconds. The standard setting is 1500 seconds.

Value

an integer indicating the structuredness

Examples

```
structuredness(file_path, generate_new_path_log = TRUE)
```

task_names *Task names*

Description

A function which returns the task names together with the task ids

Usage

```
task_names(xml_internal_doc, filter_non_connector_activities = FALSE,
  signavio = FALSE)
```

Arguments

`xml_internal_doc`	document object created using the create_internal_document function
`filter_non_connector_activities`	attribute indicating whether non connector activities should be filtered. The default value is FALSE.
`signavio`	boolean which indicates whether the file stems from signavio

Value

an object containing a table with the IDs and tasknames

Examples

```
task_names(create_internal_doc(file_path))
```

token_split *Token Split*

Description

Token split is defined as the sum of the outgoing flows of parallel, inclusive and complex gateways minus one, because otherwise the token_split value is always one, while it should be zero if there are

Usage

```
token_split(file_path, signavio = FALSE)
```

Arguments

`file_path`	document object created using the create_internal_document function
`signavio`	boolean which indicates whether the file stems from signavio

Value

an integer indicating the token_split

Examples

```
token_split(file_path)
```

`traces_contain_relation` *Relation in traces*

Description

This functions returns true or false on whether there exists always or sometimes an (indirect) relation between two activities in a process model. This can be useful for measuring the understandability using behavioral profiles. Always means that wheneve activity 1 is part of the trace, activity 2 will some time follow activity 1. Sometimes means that there should be at least one case where there is an indirect relation and at least one case where there is not. The indirect relations between two activities due to a parallel construct are left out of scope for this function.

Usage

```
traces_contain_relation(repetition_and_path_log, xml_internal_doc,
  activity_1, activity_2, always = TRUE, filter_indirect = TRUE,
  precede = FALSE, alternate_response = FALSE,
  alternate_precedence = FALSE, chain_response = FALSE,
  chain_precedence = FALSE, negation_alternate_precedence = FALSE,
  negation_alternate_response = FALSE)
```

Arguments

`repetition_and_path_log`	repetition and path log list object created by the function create_repetition_and_path_log
`xml_internal_doc`	document object created using the create_internal_document function
`activity_1`	the activity name of the first activity
`activity_2`	the activity name of the second activity in the relation
`always`	a boolean value indicating whether there should be always a direct relation. If it is false, it is assumed to be tested for the sometimes case.
`filter_indirect`	a boolean value indicating whether indirect relations are targeted. If not, all relations are used
`precede`	a boolean value indicating whether precede or follows relation is tested
`alternate_response`	a boolean indicating whether an alternate response relation is tested

`alternate_precedence`
: a boolean indicating whether an alternate precedence relation is tested

`chain_response`
: a boolean indicating whether a chain response relation is tested

`chain_precedence`
: a boolean indicating whether a chain precedence relation is tested

`negation_alternate_precedence`
: a boolean indicating whether a negation alternate precedence relation is tested

`negation_alternate_response`
: a boolean indicating whether a negation alternate response relation is tested

Value

a boolean value indicating whether it is true that there is always or sometimes an indirect relation between activity_1 and activity_2

Examples

```
## Not run: traces_contain_relation(log, doc, "A", "F", TRUE, TRUE)
```

`understandBPMN` *understandBPMN - understandability metrics for BPMN models*

Description

This package provides the implementation of several comprehensibility and complexity metrics for BPMN models

Index