Package ‘edeaR’

October 1, 2020

Type Package

Title Exploratory and Descriptive Event-Based Data Analysis

Version 0.8.6

Date 2020-10-01

Description Exploratory and descriptive analysis of event based data. Provides methods for describing and selecting process data, and for preparing event log data for process mining. Builds on the S3-class for event logs implemented in the package 'bupaR'.

License MIT + file LICENSE

Depends R(>= 3.0.0)

Imports bupaR (>= 0.4.1), dplyr, data.table, ggplot2, ggthemes, glue, tibble, shiny, miniUI, tidyr, shinyTime, lubridate, purrr, stringr, rlang, zoo, hms, forcats

LazyData true

RoxygenNote 7.1.0.9000

URL `https://www.bupar.net`, `https://github.com/bupaverse/edeaR`

Suggests knitr, eventdataR, rmarkdown

VignetteBuilder knitr

BugReports `https://github.com/bupaverse/edeaR/issues`

NeedsCompilation no

Author Gert Janssenswillen [aut, cre], Marijke Swennen [ctb]

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

Repository CRAN

Date/Publication 2020-10-01 12:20:02 UTC

R topics documented:

`activity_frequency` *Metric: Activity Frequency*

Description

Provides summary statistics about the frequency of activity types at the level of log, traces, cases, activity types.

Usage

```
activity_frequency(eventlog, level, append, append_column, ...)

## S3 method for class 'eventlog'
activity_frequency(
  eventlog,
  level = c("log", "trace", "activity", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
activity_frequency(
  eventlog,
  level = c("log", "trace", "activity", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, trace, case, activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.

`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`...`	Deprecated arguments
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

- At log level, This metric shows the summary statistics of the frequency of activities throughout the complete event log.
- On the level of the cases, this metric showsthe absolute and relative number of times the different activity types occur in each case. The absolute number shows the number of distinct activity types that occur in each of the cases. The relative number is calculated based on the total activity executions in the case.
- On trace level, this metric presents the absolute and relative number of times a specific activity type occurs in each trace.
- On the level of the activities, this metric provides the absolute and relative frequency of a specific activity in the complete event log.

Methods (by class)

- `eventlog`: Compute activity frequency for eventlog
- `grouped_eventlog`: Compute activity frequency for grouped event log

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`activity_presence` *Metric: Activity Presence*

Description

Calculates for each activity type in what percentage of cases it is present.

Usage

```
activity_presence(eventlog, append, append_column, sort, ...)

## S3 method for class 'eventlog'
activity_presence(
  eventlog,
  append = F,
  append_column = "absolute",
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
activity_presence(
  eventlog,
  append = F,
  append_column = "absolute",
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.
`...`	Deprecated arguments

Details

An indication of variance can be the presence of the activities in the different cases. This metric shows for each activity the absolute number of cases in which each activity occurs together with its relative presence.

Methods (by class)

- `eventlog`: Compute activity presence for event log
- `grouped_eventlog`: Compute activity presence for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

Examples

```
## Not run:
data <- data.frame(case = rep("A",5),
activity_id = c("A","B","C","D","E"),
activity_instance_id = 1:5,
lifecycle_id = rep("complete",5),
timestamp = 1:5,
resource = rep("resource 1", 5))
```

```
log <- bupaR::eventlog(data,case_id = "case",
activity_id = "activity_id",
activity_instance_id = "activity_instance_id",
lifecycle_id = "lifecycle_id",
timestamp = "timestamp",
resource_id = "resource")

activity_presence(log)

## End(Not run)
```

`change_day` *Adjust days in work schedule*

Description

Adjust days in work schedule

Usage

```
change_day(work_schedule, day, start_time, end_time)
```

Arguments

`work_schedule`	Work schedule created with create_work_schedule
`day`	A numeric vector containing the days to be changed. 1 = monday.
`start_time`	The new start time for selected days (hh:mm:ss)
`end_time`	The new end time for selected days (hh:mm:ss)

`create_work_schedule` *Create work schedule*

Description

Create work schedule

Usage

```
create_work_schedule(start_time = "9:00:00", end_time = "17:00:00")
```

Arguments

`start_time`	Character indicating the usual start time for workdays (hh:mm:ss)
`end_time`	Character indicating the usual end time for workdays (hh:mm:ss)

edeaR *edeaR - Exploratory and Descriptive Event-based data Analysis in R*

Description

This package provides several useful techniques for Exploratory and Descriptive analysis of event based data in R, developed by the Business Informatics Research Group of Hasselt University.

end_activities *Metric: End activities*

Description

Analyse the end activities in the process.

Usage

```
end_activities(eventlog, level, append, ...)

## S3 method for class 'eventlog'
end_activities(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
end_activities(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`

`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

- On log levels, this metric shows the absolute and relative number of activities that are the last activity in one or more of the cases.
- On the level of the specific cases, this metric provides an overview of the end activity of each case.
- On the activity level This metric calculates for each activity the absolute and relative number of cases that end with this activity type. Similar to the start activities metric, the relative number is calculated as a portion of the number of cases, being the number of \"opportunities\" that an activity could be the end activity. The cumulative sum is added to have an insight in the number of activities that is required to cover a certain part of the total.
- On the level of the distinct resources, an overview of which resources execute the last activity per case can be of interest for a company. Probably this person is responsible for the correct communication to the customer.
- Finally, on the resource-activity level, this metric shows for each occurring resource-activity combination the absolute and relative number of times this resource executes this activity as an end activity in a case.

Methods (by class)

- `eventlog`: Compute end activities for eventlog
- `grouped_eventlog`: Compute end activities for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`filter_activity` *Filter: Activity*

Description

Filters the log based on activities

Usage

```
filter_activity(eventlog, activities, reverse, ...)

## S3 method for class 'eventlog'
filter_activity(eventlog, activities, reverse = FALSE, ...)

## S3 method for class 'grouped_eventlog'
filter_activity(eventlog, activities, reverse = FALSE, ...)

ifilter_activity(eventlog)

## S3 method for class 'activitylog'
filter_activity(eventlog, activities, reverse = FALSE, ...)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`activities`	Character vector containing one or more activity identifiers.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

The method filter_activity can be used to filter on activity identifiers. It has an activities argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter eventlog for activity labels
- `grouped_eventlog`: Filter grouped eventlog for activity labels
- `activitylog`: Filter activity_log for activity labels

See Also

`vignette("filters","edeaR")`

filter_activity_frequency

Filter: Activity frequency

Description

Filters the log based on frequency of activities.

Usage

```
filter_activity_frequency(eventlog, interval, percentage, reverse, ...)

## S3 method for class 'eventlog'
filter_activity_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

## S3 method for class 'grouped_eventlog'
filter_activity_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

ifilter_activity_frequency(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`interval`	An activity frequency interval (numeric vector of length 2). Half open interval can be created using NA.
`percentage`	The target coverage of activity instances. A percentile of 0.9 will return the most common activity types of the eventlog, which account for at least 90% of the activity instances.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

Filtering the event log based in activity frequency can be done in two ways: using an interval of allowed frequencies, or specify a coverage percentage.

- percentage: When filtering using a percentage p%, the filter will return p frequency. The filter will retain additional activity labels as long as the number of activity instances does not exceed the percentage threshold.
- interval: When filtering using an interval, activity labels will be retained when their absolute frequency fall in this interval. The interval is specified using a numeric vector of length 2. Half open intervals can be created by using NA. E.g., ‘c(10, NA)‘ will select activity labels which occur 10 times or more.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter eventlog on activity frequency
- `grouped_eventlog`: Stratified filter for grouped eventlog

See Also

`vignette("filters","edeaR")`

---

filter_activity_instance

title Filter: Activity instance

Description

Filters the log based on activity instance identifier

Usage

```
filter_activity_instance(eventlog, activity_instances, reverse)

## S3 method for class 'eventlog'
filter_activity_instance(eventlog, activity_instances = NULL, reverse = FALSE)

## S3 method for class 'grouped_eventlog'
filter_activity_instance(eventlog, activity_instances = NULL, reverse = FALSE)

ifilter_activity_instance(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`activity_instances`	A vector of activity instance identifiers
`reverse`	Logical, indicating whether the selection should be reversed.

Details

The method filter_activity_instance can be used to filter on activity instance identifiers. It has an activity_instances argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter for eventlogs
- `grouped_eventlog`: Stratified filter for grouped eventlogs

See Also

`vignette("filters","edeaR")`

`filter_activity_presence`

Filter: Activity Presence

Description

Filters cases based on the presence (or absence) of activities

Usage

```
filter_activity_presence(eventlog, activities, method, reverse)

## S3 method for class 'eventlog'
filter_activity_presence(
  eventlog,
  activities = NULL,
  method = c("all", "one_of", "none", "exact", "only"),
  reverse = FALSE
)
```

```
## S3 method for class 'grouped_eventlog'
filter_activity_presence(
  eventlog,
  activities = NULL,
  method = c("all", "one_of", "none", "exact", "only"),
  reverse = FALSE
)

ifilter_activity_presence(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`activities`	Character vector containing one or more activity identifiers.
`method`	Filter method. If "all", each of the activities should be present. If "one_of", at least one of them should be present. If "none", none of the activities are allowed to occur in the filtered traces.
`reverse`	Logical, indicating whether the selection should be reversed.

Details

This functions allows to filter cases that contain certain activities. It requires as input a vector containing one or more activity labels and it has a method argument. The latter can have the values all, none or one_of.

- When set to ‘all‘, it means that all the specified activity labels must be present for a case to be selected
- ‘none‘ means that they are not allowed to be present.
- ‘one_of‘ means that at least one of them must be present.
- ‘only‘ means that only (a set of) these activities are allowed to be present
- ‘exact‘ means that only exactly these activities can be present (although multiple times and in random orderings)

When only one activity label is supplied, note that methods all and one_of will be identical.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log on presence of activities.
- `grouped_eventlog`: Filter grouped event log on presence of activities.

See Also

`vignette("filters","edeaR")`

`filter_case` *title Filter: Case*

Description

Filters the log based on case identifier

Usage

```
filter_case(eventlog, cases, reverse)

## S3 method for class 'eventlog'
filter_case(eventlog, cases = NULL, reverse = FALSE)

## S3 method for class 'grouped_eventlog'
filter_case(eventlog, cases = NULL, reverse = FALSE)

ifilter_case(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`cases`	A vector of cases identifiers
`reverse`	Logical, indicating whether the selection should be reversed.

Details

The method filter_case can be used to filter on case identifiers. It has an cases argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter for eventlogs
- `grouped_eventlog`: Stratified filter for grouped eventlogs

See Also

`vignette("filters","edeaR")`

`filter_case_condition` *title Filter: Case*

Description

Filters cases using a condition

Usage

```
filter_case_condition(eventlog, condition, reverse)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`condition`	A condition
`reverse`	Logical, indicating whether the selection should be reversed.

Details

Only keeps cases if the condition is valid for at least one event.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

See Also

`vignette("filters","edeaR")`

`filter_endpoints` *Filter: Start and end activities*

Description

Filters the log based on a provided set of start and end activities

Usage

```
filter_endpoints(
  eventlog,
  start_activities,
  end_activities,
  percentage,
  reverse,
  ...
)

## S3 method for class 'eventlog'
filter_endpoints(
  eventlog,
  start_activities = NULL,
  end_activities = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

## S3 method for class 'grouped_eventlog'
filter_endpoints(
  eventlog,
  start_activities = NULL,
  end_activities = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

ifilter_endpoints(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`start_activities`	A vector of activity identifiers, or NULL
`end_activities`	A vector of activity identifiers, or NULL
`percentage`	A percentage p to be used as percentile cut off. When this is used, the most common endpoint-pairs will be selected until at least the p% of the cases are selected.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

The filter_endpoints method filters cases based on the first and last activity label. It can be used in two ways: by specifying vectors with allowed start activities and/or allowed end activities, or by specifying a percentile. In the latter case, the percentile value will be used as a cut off. For example, when set to 0.9, it will select the most common endpoint pairs which together cover at least 90

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log stratified

See Also

`vignette("filters","edeaR")`

`filter_endpoints_conditions`
Filter: Start and end conditions

Description

Filters cases where the first and/or last activity adhere to the specified conditions

Usage

```
filter_endpoints_conditions(
  eventlog,
  start_condition,
  end_condition,
  reverse,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`start_condition`	A logical condition
`end_condition`	A logical condition
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

The filter_endpoints method filters cases based on the first and last activity label. It can be used in two ways: by specifying vectors with allowed start activities and/or allowed end activities, or by specifying a percentile. In the latter case, the percentile value will be used as a cut off. For example, when set to 0.9, it will select the most common endpoint pairs which together cover at least 90

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

See Also

`vignette("filters","edeaR")`

`filter_infrequent_flows` *Filter cases based on infrequent flows.*

Description

Filter cases based on infrequent flows.

Usage

```
filter_infrequent_flows(eventlog, min_n)
```

Arguments

`eventlog`	The event log to use
`min_n`	Cases containing a flow that occurs less than min_n times are discarded.

Value

Filtered event log.

`filter_lifecycle` *Filter: Life cycle*

Description

Filters the log based on the life cycle id

Usage

```
filter_lifecycle(eventlog, lifecycle, reverse, ...)

## S3 method for class 'eventlog'
filter_lifecycle(eventlog, lifecycle, reverse = FALSE, ...)

## S3 method for class 'grouped_eventlog'
filter_lifecycle(eventlog, lifecycle, reverse = FALSE, ...)

ifilter_lifecycle(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`lifecycle`	Character vector containing one or more life cycle identifiers.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

The method filter_lifecycle can be used to filter on life cycle identifiers. It has an lifecycle argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter eventlog on life cycle labels
- `grouped_eventlog`: Filter grouped eventlog on life cycle labels

See Also

`vignette("filters","edeaR")`

`filter_lifecycle_presence` *Filter: Life cycle Presence*

Description

Filters activity instances based on the presence (or absence) of life cycles

Usage

```
filter_lifecycle_presence(eventlog, lifecycle, method, reverse)

## S3 method for class 'eventlog'
filter_lifecycle_presence(
  eventlog,
  lifecycle = NULL,
  method = c("all", "one_of", "none", "exact", "only"),
  reverse = FALSE
)

## S3 method for class 'grouped_eventlog'
filter_lifecycle_presence(
  eventlog,
  lifecycle = NULL,
  method = c("all", "one_of", "none", "exact", "only"),
  reverse = FALSE
)

ifilter_lifecycle_presence(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`lifecycle`	Character vector containing one or more life cycle identifiers.
`method`	Filter method. If "all", each of the life cycle labels should be present. If "one_of", at least one of them should be present. If "none", none of the life cycle labels are allowed to occur in the filtered activity instances.
`reverse`	Logical, indicating whether the selection should be reversed.

Details

This functions allows to filter activity instances that (do not) contain certain life cycle identifiers. It requires as input a vector containing one or more life cycle labels and it has a method argument. The latter can have the values all, none or one_of.

- When set to ‘all‘, it means that all the specified life cycle labels must be present for an activity instance to be selected

- ‘none‘ means that they are not allowed to be present.
- ‘one_of‘ means that at least one of them must be present.
- ‘only‘ means that only (a set of) these life cycle labels are allowed to be present
- ‘exact‘ means that only exactly these life cycle labels can be present (although multiple times and in random orderings)

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log on presence of life cycle labels.
- `grouped_eventlog`: Filter grouped event log on presence of life cycle labels.

See Also

`vignette("filters","edeaR")`

`filter_precedence` *Filter: precedence relations*

Description

Filters cases based on the precedence relations between two sets of activities.

Usage

```
filter_precedence(
  eventlog,
  antecedents,
  consequents,
  precedence_type,
  filter_method,
  reverse
)

## S3 method for class 'eventlog'
filter_precedence(
  eventlog,
  antecedents,
  consequents,
  precedence_type = c("directly_follows", "eventually_follows"),
  filter_method = c("all", "one_of", "none"),
  reverse = FALSE
)

## S3 method for class 'grouped_eventlog'
filter_precedence(
  eventlog,
  antecedents,
  consequents,
  precedence_type = c("directly_follows", "eventually_follows"),
  filter_method = c("all", "one_of", "none"),
  reverse = FALSE
)

ifilter_precedence(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`antecedents, consequents`	The set of antecendent and consequent activities. Both are character vectors containing at leaste one activity identifier. All pairs of antecedents and consequents are turned into seperate precedence rules.
`precedence_type`	When `directly_follows`, the consequent activity should happen immediately after the antecedent activities. When `eventually_follows`, other events are allowed to happen in between.
`filter_method`	When `all`, only cases where all the relations are valid are preserved. When `one_of`, all the cases where at least one of the conditions hold are preserved. When `none`, none of the relations are allowed.
`reverse`	Logical, indicating whether the selection should be reversed.

Details

In order to extract a subset of an event log which conforms with a set of precedence rules, one can use the filter_precedence method. There are two types of precendence relations which can be tested: activities that should directly follow each other, or activities that should eventually follow each other. The type can be set with the precedence type argument. Further, the filter requires a vector of one or more antecedents (containing activity labels), and one or more consequents. Finally, also a filter method argument can be set. This argument is relevant when there is more than one antecedent or consequent. In such a case, you can specify that all possible precedence combinations must be present (all), at least one of them (one of), or none (none).

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`filter_precedence_condition` *Filter: precedence relations*

Description

Filters cases based on the precedence relations between two sets of activities.

Usage

```
filter_precedence_condition(
  eventlog,
  antecedent_condition,
  consequent_condition,
  precedence_type,
  reverse
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`antecedent_condition, consequent_condition`	The antecendent and consequent conditions
`precedence_type`	When `directly_follows`, the consequent condition should hold immediately after the antecedent condition hold When `eventually_follows`, other events are allowed to happen in between.
`reverse`	Logical, indicating whether the selection should be reversed.

Details

In order to extract a subset of an event log which conforms with a set of precedence rules, one can use the filter_precedence method. There are two types of precendence relations which can be tested: activities that should directly follow each other, or activities that should eventually follow each other. The type can be set with the precedence type argument. Further, the filter requires a vector of one or more antecedents (containing activity labels), and one or more consequents. Finally, also a filter method argument can be set. This argument is relevant when there is more than one antecedent or consequent. In such a case, you can specify that all possible precedence combinations must be present (all), at least one of them (one of), or none (none).

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

See Also

`vignette("filters","edeaR")`

`filter_precedence_resource`
Filter: precedence relations with identical resources

Description

Filters cases based on the precedence relations between two sets of activities, where both antecendent and consequent have to be executed by the same resource.

Usage

```
filter_precedence_resource(
  eventlog,
  antecedents,
  consequents,
  precedence_type,
  filter_method,
  reverse
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`antecedents, consequents`	The set of antecendent and consequent activities. Both are character vectors containing at least one activity identifier. All pairs of antecedents and consequents are turned into seperate precedence rules.
`precedence_type`	When `directly_follows`, the consequent activity should happen immediately after the antecedent activities. When `eventually_follows`, other events are allowed to happen in between.
`filter_method`	When `all`, only cases where all the relations are valid are preserved. When `one_of`, all the cases where at least one of the conditions hold are preserved. When `none`, none of the relations are allowed.
`reverse`	Logical, indicating whether the selection should be reversed.

Details

In order to extract a subset of an event log which conforms with a set of precedence rules, one can use the filter_precedence method. There are two types of precendence relations which can be tested: activities that should directly follow each other, or activities that should eventually follow each other. The type can be set with the precedence type argument. Further, the filter requires a vector of one or more antecedents (containing activity labels), and one or more consequents. Finally, also a filter method argument can be set. This argument is relevant when there is more than one antecedent or consequent. In such a case, you can specify that all possible precedence combinations must be present (all), at least one of them (one of), or none (none). In case an activity instance exists of more than one events with different resource identifiers, only the first will be considered.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

See Also

`vignette("filters","edeaR")`

`filter_processing_time`

Filter: Processing Time

Description

Filters cases based on their processing time.

Usage

```
filter_processing_time(eventlog, interval, percentage, reverse, units, ...)

## S3 method for class 'eventlog'
filter_processing_time(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  units = c("days", "hours", "mins", "secs", "weeks"),
  ...
)

## S3 method for class 'grouped_eventlog'
filter_processing_time(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  units = c("days", "hours", "mins", "secs", "weeks"),
  ...
)

ifilter_processing_time(eventlog)
```

Arguments

eventlog	The dataset to be used. Should be a (grouped) eventlog object.
interval	An duration interval (numeric vector of length 2) to be used for absolute. Half open interval can be created using NA.
percentage	A percentage p to be used for relative filtering.
reverse	Logical, indicating whether the selection should be reversed.
units	The time unit used for defining filter intervals.
...	Deprecated arguments.

Details

This filter can be used by using an interval or by using a percentage. The percentage will always start with the shortest cases first and stop including cases when the specified percentile is reached. On the other hand, an absolute interval can be defined instead to filter cases which have a processing time in this interval. The time units in which this interval is defined can be submitted with the units argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

filter_resource *Filter: Resource*

Description

Filters the log based on resource identifiers

Usage

```
filter_resource(eventlog, resources, reverse)

## S3 method for class 'eventlog'
filter_resource(eventlog, resources, reverse = FALSE)

## S3 method for class 'grouped_eventlog'
filter_resource(eventlog, resources, reverse = FALSE)

ifilter_resource(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`resources`	A vector of resources identifiers
`reverse`	Logical, indicating whether the selection should be reversed.

Details

#' The method filter_resource can be used to filter on resource identifiers. It has a resources argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

filter_resource_frequency *Filter: Activity frequency*

Description

Filters the log based on frequency of activities

Filtering the event log based in resource frequency can be done in two ways: using an interval of allowed frequencies, or specify a coverage percentage.

- percentage: When filtering using a percentage p%, the filter will return p frequency. The filter will retain additional resource labels as long as the number of activity instances does not exceed the percentage threshold.
- interval: When filtering using an interval, resource labels will be retained when their absolute frequency fall in this interval. The interval is specified using a numeric vector of length 2. Half open intervals can be created by using NA. E.g., ‘c(10, NA)‘ will select resource labels which occur 10 times or more.

Usage

```
filter_resource_frequency(eventlog, interval, percentage, reverse, ...)

## S3 method for class 'eventlog'
filter_resource_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

## S3 method for class 'grouped_eventlog'
filter_resource_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

ifilter_resource_frequency(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.

`interval`	An resource frequency interval (numeric vector of length 2). Half open interval can be created using NA.
`percentage`	The target coverage of activity instances. A percentile of 0.9 will return the most common resource types of the eventlog, which account for at least 90% of the activity instances.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event logs

See Also

`vignette("filters","edeaR")`

`filter_throughput_time`

Filter: Throughput Time

Description

Filters cases based on their throughput time.

Usage

```
filter_throughput_time(eventlog, interval, percentage, reverse, units, ...)

## S3 method for class 'eventlog'
filter_throughput_time(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  units = c("days", "hours", "mins", "secs", "weeks"),
  ...
)

## S3 method for class 'grouped_eventlog'
```

```
filter_throughput_time(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  units = c("days", "hours", "mins", "secs", "week"),
  ...
)

ifilter_throughput_time(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`interval`	An duration interval (numeric vector of length 2) to be used for absolute. Half open interval can be created using NA.
`percentage`	A percentage p to be used for relative filtering.
`reverse`	Logical, indicating whether the selection should be reversed.
`units`	The time unit used for defining filter intervals.
`...`	Deprecated arguments.

Details

This filter can be used by using an interval or by using a percentage. The percentage will always start with the shortest cases first and stop including cases when the specified percentile is reached. On the other hand, an absolute interval can be defined instead to filter cases which have a throughput time in this interval. The time units in which this interval is defined can be submitted with the units argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`filter_time_period` *Filter: Time Period*

Description

Function to filter eventlog using a time period.

Usage

```
filter_time_period(eventlog, interval, filter_method, force_trim, reverse, ...)

## S3 method for class 'eventlog'
filter_time_period(
  eventlog,
  interval = NULL,
  filter_method = c("contained", "intersecting", "start", "complete", "trim"),
  force_trim = FALSE,
  reverse = FALSE,
  ...
)

## S3 method for class 'grouped_eventlog'
filter_time_period(
  eventlog,
  interval = NULL,
  filter_method = c("contained", "intersecting", "start", "complete", "trim"),
  force_trim = FALSE,
  reverse = FALSE,
  ...
)

ifilter_time_period(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`interval`	A time interval. A vector of length 2 of type Date or POSIXct. Half-open intervals can be created with NA.
`filter_method`	Can be `contained`,`start`,`complete`,`intersecting` or `trim`.
`force_trim`	Logical, if true in combination with filter method trim activity instances on the edges of the interval are cut at the exact edge of the interval.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

Event data can be filtered by supplying a time window to the method filter_time_period. There are 5 different filter methods.

- `contained` keeps all the events related to cases contained in the time period.
- `start` keeps all the events related to cases started in the time period.
- `complete` keeps all the events related to cases complete in the time period.
- `intersecting` keeps all the events related to cases in which at least one event started and/or ended in the time period.
- `trim` keeps all the events which started and ended in the time frame.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`filter_trace` *title Filter: Trace*

Description

Filters the log based on trace id

Usage

```
filter_trace(eventlog, trace_ids, reverse)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`trace_ids`	A vector of trace identifiers
`reverse`	Logical, indicating whether the selection should be reversed.

Details

The method filter_trace can be used to filter on trace id It has an trace_ids argument, to which a vector of identifiers can be given. The selection can be negated with the reverse argument.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

See Also

`vignette("filters","edeaR")`

---

filter_trace_frequency

Filter: Trace frequency

Description

Filters the log based the frequency of traces, using an interval or a percentile cut off.

Usage

```
filter_trace_frequency(eventlog, interval, percentage, reverse, ...)

## S3 method for class 'eventlog'
filter_trace_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

## S3 method for class 'grouped_eventlog'
filter_trace_frequency(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

ifilter_trace_frequency(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`interval`	WHen given an interval, the filter will select cases of which the trace has a frequency inside the interval.

`percentage`	When given a percentage p, the filter will select the most common traces, until at least p% of the cases is covered.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

This filter can be used to filter cases based on the frequency of the corresponding trace. A trace is a sequence of activity labels, and will be discussed in more detail in Section 6. There are again two ways to select cases based on trace frequency, by interval or by percentile cut off. The percentile cut off will start with the most frequent traces.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`filter_trace_length` *Filter: Trace length percentile*

Description

Filters cases on length, using a percentile threshold.

Usage

```
filter_trace_length(eventlog, interval, percentage, reverse, ...)

## S3 method for class 'eventlog'
filter_trace_length(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)
```

```
## S3 method for class 'grouped_eventlog'
filter_trace_length(
  eventlog,
  interval = NULL,
  percentage = NULL,
  reverse = FALSE,
  ...
)

ifilter_trace_length(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`interval`	An trace length interval (numeric vector of length 2) to be used for absolute. Half open interval can be created using NA.
`percentage`	A percentage p to be used for relative filtering.
`reverse`	Logical, indicating whether the selection should be reversed.
`...`	Deprecated arguments.

Details

This filter can be used by using an interval or by using a percentage. The percentage will always start with the shortest cases first and stop including cases when the specified percentile is reached. On the other hand, an absolute interval can be defined instead to filter cases which have a length in this interval.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

filter_trim *Filter: Trim cases*

Description

Trim cases from the first event of a set of start activities to the last event of a set of end activities.

Usage

```
filter_trim(eventlog, start_activities, end_activities, reverse)

## S3 method for class 'eventlog'
filter_trim(
  eventlog,
  start_activities = NULL,
  end_activities = NULL,
  reverse = FALSE
)

## S3 method for class 'grouped_eventlog'
filter_trim(
  eventlog,
  start_activities = NULL,
  end_activities = NULL,
  reverse = FALSE
)

ifilter_trim(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`start_activities`	A vector of activity identifiers, or NULL
`end_activities`	A vector of activity identifiers, or NULL
`reverse`	Logical, indicating whether the selection should be reversed.

Details

One can trim cases by removing one or more activity instances at the start and/or end of a case. Trimming is performed until all cases have a start and/or end point belonging to a set of allowed activity labels. This filter requires a set of allowed start activities and/or a set of allowed end activities. If one of them is not provided it will not trim the cases at this edge.The selection can be reversed, which means that only the trimmed events at the start and end of cases are retained. As such, this argument allows to cut intermediate parts out of traces.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`filter_trim_lifecycle` *Filter: Trim activity instances based on life cycle labels*

Description

Trim activity instances from the first event of a set of start life cycle labels to the last event of a set of end life cycle labels

Usage

```
filter_trim_lifecycle(eventlog, start_lifecycle, end_lifecycle, reverse)

## S3 method for class 'eventlog'
filter_trim_lifecycle(
  eventlog,
  start_lifecycle = NULL,
  end_lifecycle = NULL,
  reverse = FALSE
)

## S3 method for class 'grouped_eventlog'
filter_trim_lifecycle(
  eventlog,
  start_lifecycle = NULL,
  end_lifecycle = NULL,
  reverse = FALSE
)

ifilter_trim_lifecycle(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object.
`start_lifecycle`	A vector of life cycle identifiers, or NULL
`end_lifecycle`	A vector of life cycle identifiers, or NULL
`reverse`	Logical, indicating whether the selection should be reversed.

Details

One can trim activity instances by removing one or more events at the start and/or end of the activity instances. Trimming is performed until all activity instances have a start and/or end point belonging to a set of allowed life cycle labels. This filter requires a set of allowed start life cycle labels and/or a set of allowed life cycle labels. If one of them is not provided it will not trim the activity instances at this edge.The selection can be reversed, which means that only the trimmed events at the start and end of activity instances are retained. As such, this argument allows to cut intermediate parts out of activity instances.

Value

When given an eventlog, it will return a filtered eventlog. When given a grouped eventlog, the filter will be applied in a stratified way (i.e. each separately for each group). The returned eventlog will be grouped on the same variables as the original event log.

Methods (by class)

- `eventlog`: Filter event log
- `grouped_eventlog`: Filter grouped event log

See Also

`vignette("filters","edeaR")`

`idle_time` *Metric: Idle Time*

Description

Calculates the amount of time that no activity occurs.

Usage

```
idle_time(eventlog, level, append, append_column, units, ...)

## S3 method for class 'eventlog'
idle_time(
  eventlog,
  level = c("log", "case", "trace", "resource"),
  append = FALSE,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
idle_time(
  eventlog,
  level = c("log", "case", "trace", "resource"),
  append = FALSE,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, trace, or resource. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`units`	Time units to be used
`...`	Deprecated arguments
`sort`	Sort by decreasing idle time. Defaults to true. Only relevant voor trace and resource level.

Details

- On the level of the complete event log, the idle time metric provides an overview of summary statistics of the idle time per case, aggregated over the complete event log.
- The metric applied on the level of the specific cases in the event log provides an overview of the total idle time per case

- On the level of the different traces that occur in the event log, the idle time metric provides an overview of the summary statistics of the idle time for each trace in the event log.
- The metric can also be of interest on the level of the resources, to get an insight in the amount of time each resource \"wastes\" during the process.

Methods (by class)

- `eventlog`: Compute the idle time for eventlog
- `grouped_eventlog`: Compute idle time for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`number_of_repetitions` *Metric: Number of repetitions*

Description

Provides information statistics on the number of repetitions

Usage

```
number_of_repetitions(eventlog, type, level, append, ...)

## S3 method for class 'eventlog'
number_of_repetitions(
  eventlog,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
number_of_repetitions(
  eventlog,
  type = c("repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`type`	The type of repetitions, either repeat or redo.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

A repetition is an execution of an activity within a case while that activity has already been executed before, but one or more other activities are executed in between.Similar to the self-loop metric, a distinction should be made between repeat and redo repetitions. Repeat repetitions are activity executions of the same activity type that are executed not immediately following each other, but by the same resource. Redo repetitions are activity executions of the same activity type that are executed not immediately following each other and by a different resource than the first activity occurrence of this activity type.

- The number of repetitions can be calculated on the level of the complete event log. This metric shows the summary statistics of the number of repetitions within a case, which can provide insights in the amount of waste in an event log. Each combination of two or more occurrences of the same activity, executed not immediately following each other, by the same resource is counted as one repeat repetition of this activity.
- On case level, this metric provides the absolute and relative number of repetitions in each case.
- On the level of specific activities, this metric shows which activities occur the most in a repetition. The absolute and relative number of both repeat and redo repetitions is provided by this metric, giving an overview per activity.
- When looking at the different resources executing activities in the event log, it can be interesting to have an overview of which resources need more than one time to execute an activity in a case or which resources need to have an activity redone later on in the case by another resource. This metric provides the absolute and relative number of times each resource appears in a repetition.
- Finally, the same metric can be looked at on the level of specific resource-activity combinations, providing the company with specific information about which activities and which resources are involved in the repetitions. For this metric the absolute and relative number of repeat and redo repetitions is provided. Again two dieferent relative numbers are provided, one relative to the total number of executions of the activity in the complete event log, and one relative to the total number of executions performed by the resource throughout the complete event log.

Methods (by class)

- `eventlog`: Apply metric on event log
- `grouped_eventlog`: Apply metric on grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`number_of_selfloops` *Metric: Number of selfloops in trace*

Description

Provides information statistics on the number of selfloops.

Usage

```
number_of_selfloops(eventlog, type, level, append, ...)

## S3 method for class 'eventlog'
number_of_selfloops(
  eventlog,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
number_of_selfloops(
  eventlog,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`type`	The type of repetitions, either all, repeat or redo.

`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

Activity instances of the same activity type that are executed more than once immediately after each other by the same resource are in a self-loop (length-1-loop). If an activity instance of the same activity type is executed 3 times after each other by the same resource, this is defined as a size 2 self-loop.

Two types of self-loops are defined, which are repeat self-loops and redo self-loops. Repeat self-loops are activity executions of the same activity type that are executed immediately following each other by the same resource. Redo self-loops are activity executions of the same activity type that are executed immediately following each other by a different resource. Repeat and redo repetitions are explained further on.

These metrics are presented on five different levels of analysis, which are the complete event log, cases, activities, resources and resource-activity combinations.

- On the level of the complete event log, the summary statistics of the number of self-loops within a trace can give a first insight in the amount of waste in an event log. As stated earlier, each combination of two occurrences of the same activity executed by the same resource will be counted as one repeat self-loop of this activity.
- This metric on the level of cases provides an overview of the absolute and relative number of repeat and redo self-loops in each case. To calculate the relative number, each (repeat or redo) self-loop is counted as 1 occurrence, and the other activity instances are also counted as 1.
- On the level of the distinct activities in the event log, the absolute and relative number of self-loops per activity can be an indication for the company which activities are causing the most waste in the process.
- Similar to the metric on the level of the activities, the number of self-loops on the level of the resources executing the activities can give a company insights in which employee needs to repeat his or her work most often within a case, or for which employee the work he or she did should be redone by another employee within the same case. This metric shows the absolute and relative number of both repeat and redo self-loops for each resource in the event log.
- Finally, the metric can be applied to the level of the specifc resource-activity combinations, in order to get an insight in which activities are the most crucial for which resources. This metric shows the absolute and relative number of both repeat and redo self-loops for each of the resource-activity combinations that occur in the event log. Two different relative numbers are provided here, one from the resource perspective and one from the activity perspective. At the resource perspective, the denominator is the total number of executions by the resource under consideration. At the activity perspective, the denominator is the total number of occurrences of the activity under consideration.

Methods (by class)

- `eventlog`: Compute number of selfloops for eventlog
- `grouped_eventlog`: Compute number of selfloops for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`number_of_traces` *Metric: Number of traces*

Description

Computes how many traces there are.

Usage

```
number_of_traces(eventlog)

## S3 method for class 'eventlog'
number_of_traces(eventlog)

## S3 method for class 'grouped_eventlog'
number_of_traces(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.

Details

This metric provides two values, the absolute and relative number of traces that occur in the event log. The relative number shows expected number of traces needed to cover 100 cases.

Methods (by class)

- `eventlog`: Number of traces in eventlog
- `grouped_eventlog`: Number of traces in each group of eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

plot *Plot Methods*

Description

Visualize metric

Usage

```
## S3 method for class 'activity_frequency'
plot(x, ...)

## S3 method for class 'activity_presence'
plot(x, ...)

## S3 method for class 'end_activities'
plot(x, ...)

## S3 method for class 'idle_time'
plot(x, ...)

## S3 method for class 'processing_time'
plot(x, ...)

## S3 method for class 'referral_matrix'
plot(x, ...)

## S3 method for class 'resource_frequency'
plot(x, ...)

## S3 method for class 'resource_involvement'
plot(x, ...)

## S3 method for class 'resource_specialisation'
plot(x, ...)

## S3 method for class 'start_activities'
plot(x, ...)

## S3 method for class 'throughput_time'
plot(x, ...)

## S3 method for class 'trace_coverage'
plot(x, ...)

## S3 method for class 'trace_length'
plot(x, ...)
```

```
## S3 method for class 'number_of_selfloops'
plot(x, ...)

## S3 method for class 'number_of_repetitions'
plot(x, ...)
```

Arguments

`x`	Data to plot
`...`	Additional variables

Value

A ggplot object, which can be customized further, if deemed necessary.

`print.activity_frequency` *Activity Frequency Print*

Description

Print Actvitity Frequency Information

Usage

```
## S3 method for class 'activity_frequency'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.idle_time` *Idle Time Print*

Description

Print idle time Information

Usage

```
## S3 method for class 'idle_time'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.number_of_repetitions` *Repetitions Print*

Description

Print Repetitions Information

Usage

```
## S3 method for class 'number_of_repetitions'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.number_of_selfloops` *Selfloops Print*

Description

Print Selfloops Information

Usage

```
## S3 method for class 'number_of_selfloops'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.processing_time` *Processing Time Print*

Description

Print Processing Time Information

Usage

```
## S3 method for class 'processing_time'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.throughput_time` *Throughput Time Print*

Description

Print Throughput Time Information

Usage

```
## S3 method for class 'throughput_time'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.trace_coverage` *Trace coverage print*

Description

Print Trace coverage Information

Usage

```
## S3 method for class 'trace_coverage'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

`print.trace_length` *Trace length Print*

Description

Print Trace length Information

Usage

```
## S3 method for class 'trace_length'
print(x, ...)
```

Arguments

`x`	Data to print
`...`	Additional arguments

processing_time *Metric: Processing time*

Description

Provides summary statistics about the processing time of the process.

Usage

```
processing_time(
  eventlog,
  level,
  append,
  append_column,
  units,
  sort,
  work_schedule,
  ...
)

## S3 method for class 'eventlog'
processing_time(
  eventlog,
 level = c("log", "trace", "case", "activity", "resource", "resource-activity"),
  append = F,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  work_schedule = NULL,
  ...
)

## S3 method for class 'grouped_eventlog'
processing_time(
  eventlog,
 level = c("log", "trace", "case", "activity", "resource", "resource-activity"),
  append = F,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  work_schedule = NULL,
  ...
)
```

Arguments

eventlog	The dataset to be used. Should be a (grouped) eventlog object. eventlog.

`level`	Level of granularity for the analysis: log, case, trace, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`units`	Time units to be used
`sort`	Sort on decreasing processing time. For case level only.
`work_schedule`	A schedule of working hours. If provided, only working hours are counted as processing time.
`...`	Deprecated arguments

Details

In contrast to the throughput time of the cases in an event log, the metrics concerning the active time or the actual processing time provide summary statistics on the processing time of events on the level of the complete event log, the specific cases, traces, the activities, and the resource-activity combinations.

- On log level, this metric calculates the summary statistics of the actual processing time per case, summarised over the complete event log.
- On case level, a list of cases with their processing time are provided.
- On trace level, the summary statistics of processing time can be calculated for each possible sequence of activities that appears in the event log.
- Duration can also be calculated on the level of each activity. For each activity, an overview of the average processing time -or the service time- of this activity can be of interest.
- We can also look at the processing time per case on the level of each separate resource. This way, a company gets an overview of the amount of time each resource spends on a case and which resources spend more time on cases than others.
- On the resource-activity level, finally, we can have a look at the efficiency of resources by looking at the combination of each resource with each activity.

Methods (by class)

- `eventlog`: Compute processing time for event log
- `grouped_eventlog`: Compute processing time on grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

redo_repetitions_referral_matrix

Referral matrix repetitons

Description

Provides a list of initatiors and completers of redo repetitons

Usage

```
redo_repetitions_referral_matrix(eventlog)

## S3 method for class 'eventlog'
redo_repetitions_referral_matrix(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.

Methods (by class)

- `eventlog`: Compute matrix for eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

See Also

`number_of_repetitions`

redo_selfloops_referral_matrix

Referral matrix selfloops

Description

Provides a list of initatiors and completers of redo selfloops

Usage

```
redo_selfloops_referral_matrix(eventlog)

## S3 method for class 'eventlog'
redo_selfloops_referral_matrix(eventlog)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.

Methods (by class)

- `eventlog`: Compute matrix for eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

See Also

`number_of_selfloops`

`resource_frequency` *Metric: Resource frequency*

Description

Analyses the frequency of resources at different levels of analysis

Usage

```
resource_frequency(eventlog, level, append, ...)

## S3 method for class 'eventlog'
resource_frequency(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
resource_frequency(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

Comparable to the concept of the activity frequency the frequency of resources in a business process can also be very insightful for companies, e.g., during company restructuring.

- On the level of the complete event log, summary statistics show the number of times a resource executes an activity in the complete event log.
- To get a better view on the variance between the different cases, the summary statistics of the frequency of resources can be calculated on the level of the cases. This way, a company gets an insight in the number of different resources working on each case together with the number of activities a resource executes per case.
- At the level of the different activities, the resource frequency states how many different resources are executing a specific activity in the complete event log.
- At the level of the distinct resources in the event log, this metric simply shows the absolute and relative frequency of occurrences of each resource in the complete event log.
- Finally, at the most specific level of analysis, the absolute and relative number of times each resource-activity level occurs in the complete event log can be calculated. Two different relative numbers are provided here, one from the resource perspective and one from the activity perspective. At the resource perspective, the denominator is the total number of executions by the resource under consideration. At the activity perspective, the denominator is the total number of occurrences of the activity under consideration.

Methods (by class)

- `eventlog`: Resource frequency for eventlog
- `grouped_eventlog`: Resource frequency for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`resource_involvement` *Metric: Resource Involvement*

Description

Calculates for each resource/resource-activity in what percentage of cases it is present.

Usage

```
resource_involvement(eventlog, level, append, ...)

## S3 method for class 'eventlog'
resource_involvement(
  eventlog,
  level = c("case", "resource", "resource-activity"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
resource_involvement(
  eventlog,
  level = c("case", "resource", "resource-activity"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

Next to the resource frequency, the involvement of resources in cases can be of interest to, e.g., decide how "indispensable" they are. This metric is provided on three levels of analysis, which are the cases, the resources, and the resource-activity combinations

- At the level of the specific cases, the absolute and relative number of distinct resources executing activities in each case is calculated. This way a company gets an overview of which cases are handled by a small amount of resources and which cases need more resources, indicating a higher level of variance in the process.
- On the level of the distinct resources, this metric provides the absolute and relative number of cases in which each resource is involved, indicating which resources are more "necessary" within the business process than the others.
- On the level of the specific resource-activity combinations, this metric provides a list of all resource-activity combinations with the absolute and relative number of cases in which each resource-activity combination is involved.

Methods (by class)

- `eventlog`: Resource involvement for eventlog
- `grouped_eventlog`: Resource involvement for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`resource_specialisation` *Metric: Resource Specialisation*

Description

Analyses whether resources specialise in specific activities

Usage

```
resource_specialisation(eventlog, level, append, ...)

resource_specialization(eventlog, level, append, ...)

## S3 method for class 'eventlog'
resource_specialisation(
  eventlog,
  level = c("log", "case", "activity", "resource"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
resource_specialisation(
  eventlog,
  level = c("log", "case", "activity", "resource"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, or resource. For more information, see `vignette("metrics","edeaR")#'`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

This can give a company an overview of which resources are performing certain activities more than others, and which resources are responsible for containing all knowledge or capabilities on one topic.

- On the level of the complete event log, this metric provides summary statistics on the number of distinct activities executed per resource.
- On the level of the cases, this metric provides the number of distinct activities that are executed within each case together with the summary statistics of the distinct activities executed per resource in each case.
- On the level of the distinct activities, this metric provides an overview of the absolute and relative number of different resources executing this activity within the complete event log. This will give a company insights in which activities resources are specialised in.
- Finally, the resource specialisation can also be calculated on the resource level, showing the absolute and relative number of distinct activities that each resource executes.

Methods (by class)

- `eventlog`: Resource specialization for eventlog
- `grouped_eventlog`: Resource specialization for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

---

size_of_repetitions *Metric: Size of repetitions*

Description

Provides summary statistics on the sizes of repetitions.

Usage

```
size_of_repetitions(eventlog, type, level, append, ...)

## S3 method for class 'eventlog'
size_of_repetitions(
  eventlog,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  ...
)

## S3 method for class 'grouped_eventlog'
size_of_repetitions(
  eventlog,
  type = c("repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`type`	The type of repetitions, either all, repeat or redo.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.

Methods (by class)

- `eventlog`: Size of repetitions for eventlog
- `grouped_eventlog`: Size of repetitions for grouped event log

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

See Also

`number_of_repetitions`

`size_of_selfloops` *Metric: Size of selfloops*

Description

Provides summary statistics on the sizes of selfloops

Usage

```
size_of_selfloops(eventlog, type, level, append, ...)

## S3 method for class 'eventlog'
size_of_selfloops(
  eventlog,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  ...
)

## S3 method for class 'grouped_eventlog'
size_of_selfloops(
  eventlog,
  type = c("repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-acitivty"),
  append = FALSE,
  append_column = NULL,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`type`	The type of repetitions, either all, repeat or redo.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.

Methods (by class)

- `eventlog`: Size of selfloops for eventlog
- `grouped_eventlog`: Size of selfloops for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

See Also

`number_of_selfloops`

`start_activities` *Metric: Start activities*

Description

Analyse the start activities in the process

Usage

```
start_activities(eventlog, level, append, ...)

## S3 method for class 'eventlog'
start_activities(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

```
## S3 method for class 'grouped_eventlog'
start_activities(
  eventlog,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = FALSE,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort output on count. Defaults to TRUE. Only for levels with frequency count output.

Details

- On log levels, this metric shows the absolute and relative number of activities that are the first activity in one or more of the cases.
- On the level of the specific cases, this metric provides an overview of the start activity of each case.
- On the activity level This metric calculates for each activity the absolute and relative number of cases that start with this activity type. The relative number is calculated as a portion of the number of cases, being the number of "opportunities" that an activity could be the start activity. The cumulative sum is added to have an insight in the number of activities that is required to cover a certain part of the total.
- On the level of the distinct resources, an overview of which resources execute the first activity per case can be of interest for a company. Probably this person is responsible for the correct communication to the customer.
- Finally, on the resource-activity level, this metric shows for each occurring resource-activity combination the absolute and relative number of times this resource executes this activity as an start activity in a case.

Methods (by class)

- `eventlog`: Start activities for eventlog
- `grouped_eventlog`: Start activities for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

---

throughput_time *Metric: Throughput time of cases*

Description

Provides summary statistics concerning the throughput times of cases.

Usage

```
throughput_time(eventlog, level, append, append_column, units, ...)

## S3 method for class 'eventlog'
throughput_time(
  eventlog,
  level = c("log", "trace", "case"),
  append = FALSE,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
throughput_time(
  eventlog,
  level = c("log", "trace", "case"),
  append = FALSE,
  append_column = NULL,
  units = c("days", "hours", "mins", "secs", "weeks"),
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.

`units`	Time units to be used
`...`	Deprecated arguments
`sort`	Sort by decreasing throughput time. Defaults to true. Only relevant for case level.

Details

- The throughput time of a case is the total duration of the case, or the difference between the timestamp of the end event and the timestamp of the start event of the case. Possible idle time is also included in this calculation.
- On log level, the summary statistics of these throughput to describe the throughput time of cases in an aggregated fashion.
- Instead of looking at all cases in the log, it can be interesting to analyse the different process variants or traces in the log

Methods (by class)

- `eventlog`: Throughput time for eventlog
- `grouped_eventlog`: Throughput time for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`trace_coverage` *Metric: Trace coverage*

Description

Analyses the structuredness of an event log by use of trace frequencies. Applicable at logn case and trace level

Usage

```
trace_coverage(eventlog, level, append, ...)

## S3 method for class 'eventlog'
trace_coverage(
  eventlog,
  level = c("log", "trace", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

```
## S3 method for class 'grouped_eventlog'
trace_coverage(
  eventlog,
  level = c("log", "trace", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort by decreasing throughput time. Defaults to true. Only relevant for case level.

Details

- Trace: The absolute and relative frequency of each trace is returned
- Case: for each case, the coverage of the corresponding trace is returned
- Log: Summary statistics of the coverage of traces is returned.

Methods (by class)

- `eventlog`: Trace coverage metric for eventlog
- `grouped_eventlog`: Trace coverage metric for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

`trace_length` *Metric: Trace length*

Description

Analysis of trace lengths

Usage

```
trace_length(eventlog, level, append, ...)

## S3 method for class 'eventlog'
trace_length(
  eventlog,
  level = c("log", "trace", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)

## S3 method for class 'grouped_eventlog'
trace_length(
  eventlog,
  level = c("log", "trace", "case"),
  append = F,
  append_column = NULL,
  sort = TRUE,
  ...
)
```

Arguments

`eventlog`	The dataset to be used. Should be a (grouped) eventlog object. `eventlog`.
`level`	Level of granularity for the analysis: log, case, activity, resource or resource-activity. For more information, see `vignette("metrics","edeaR")`
`append`	Logical, indicating whether to append results to original event log. Ignored when level is log or trace.
`...`	Deprecated arguments
`append_column`	Which of the output columns to append to log, if append = T. Default column depends on chosen level.
`sort`	Sort by decreasing throughput time. Defaults to true. Only relevant for case level.

Details

This metric provides an overview of the number of activities that occur in each trace. In this metric, instances of an activity, as opposed to the actual activities, are calculated.

- On the level of the log, the number of actual transactions in a trace are calculated and aggregated on the log level.
- On the level of the cases, this metric calculates the number of activity instances in each case.
- This metric shows the number of activity instances executed in each trace. #'

Methods (by class)

- `eventlog`: Trace length for eventlog
- `grouped_eventlog`: Trace length for grouped eventlog

References

Swennen, M. (2018). Using Event Log Knowledge to Support Operational Exellence Techniques (Doctoral dissertation). Hasselt University.

Index