Package ‘edeaR’

December 3, 2015

Type Package

Title Exploratory and Descriptive Event-based data Analysis

Version 0.2

Date 2015-09-21

Author Gert Janssenswillen

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

Description Functions for exploratory and descriptive analysis of event based data

License GPL-3

Depends R(>= 3.0.0), methods, dplyr

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

`activities` *Activities*

Description

Returns a `tbl_df` containing a list of all activity types in the event log, with there absolute and relative frequency

Usage

```
activities(eventlog)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.

See Also

`activity_id`,`activity_instance_id`, `eventlog`

Examples

```
data(example_log)
activities(example_log)
```

`activity_id` *Activity classifier*

Description

Get the activity classifier of an object of class `eventlog`

Usage

```
activity_id(eventlog)
```

Arguments

`eventlog`	An object of class `eventlog`.

See Also

`eventlog`, `case_id`, `activity_instance_id` `timestamp`, `life_cycle_id`

Examples

```
data(example_log)
activity_id(example_log)
```

`activity_instance_id` *Activity instance classifier*

Description

Get the activity instance classifier of an object of class `eventlog`

Usage

```
activity_instance_id(eventlog)
```

Arguments

`eventlog`	An object of class `eventlog`.

See Also

`eventlog`, `activity_id`, `timestamp`, `life_cycle_id`, `case_id`

Examples

```
data(example_log)
activity_instance_id(example_log)
```

`activity_presence` *Metric: Activity Presence*

Description

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

Usage

```
activity_presence(eventlog)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.

See Also

`activity_type_frequency`

Examples

```
data(example_log)
activity_presence_in_cases(example_log)
```

`activity_type_frequency` *Metric: Activity Type Frequency*

Description

Provides summary statistics about the frequency of activity types at the level of traces, cases or activity types '

Usage

```
activity_type_frequency(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of activity type frequency should be performed: trace, case or activity.

`cases` *Cases*

Description

Provides a fine-grained summary of an event log with characteristics for each case: the number of events, the number of activity types, the timespan, the trace, the duration and the first and last event type.

Usage

```
cases(eventlog)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.

Examples

```
data(example_log)
cases(example_log)
```

`case_attributes_from_xes` *Case Attributes from Xes-file*

Description

Case Attributes from Xes-file

Usage

```
case_attributes_from_xes(xesfile = file.choose())
```

`case_id` *Case classifier*

Description

Get the case classifier of an object of class `eventlog`

Usage

```
case_id(eventlog)
```

Arguments

`eventlog`	An object of class `eventlog`.

See Also

`eventlog`, `activity_id`, `start_timestamp`, `complete_timestamp`

Examples

```
data(example_log)
case_id(example_log)
```

`durations` *Durations*

Description

Computes the throughput times of each case. Throughput time is defined as the interval between the start of the first event and the completion of the last event.

Usage

```
durations(eventlog, units = "days")
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`units`	The time unit in which the throughput times should be reported.

Examples

```
data(example_log)
durations(example_log)
```

`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

At log level, computes how many activity types occur as the last event in a case, both absolute and relative. At activity level, shows the activities which occur as last, and how often. The last event in a case is the one which completed the last.

Usage

```
end_activities(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of end activities should be performed: log, case or activity.

`eventlog` *Eventlog*

Description

A function to instantiate an object of class `eventlog` by specifying a `data.frame` or `tbl_df` and appropriate case, activity and timestamp classifiers.

Usage

```
eventlog(eventlog, case_id = NULL, activity_id = NULL,
  activity_instance_id = NULL, lifecycle_id = NULL, timestamp = NULL)
```

Arguments

`eventlog`	The data object to be used as event log. This can be a `data.frame` or `tbl_df`.
`case_id`	The case classifier of the event log.
`activity_id`	The activity classifier of the event log.
`activity_instance_id`	The activity instance classifier of the event log.
`timestamp`	The timestamp of the event log.
`lifecylce_id`	The life cylce classifier of the event log.

See Also

`case_id`, `activity_id`, `activity_instance_id`,`life_cycle_id`, `timestamp`

`eventlog_from_xes` *eventlog_from_xes*

Description

eventlog_from_xes

Usage

```
eventlog_from_xes(xesfile = file.choose())
```

`filter_activity_frequency` *Filter: Activity frequency*

Description

Filters the log based on its most frequent activities, until a specific percentile cut off.

Usage

```
filter_activity_frequency(eventlog, percentile_cut_off = 0.8, reverse = F)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`reverse`	A logical parameter depicting whether the selection should be reversed.
`percentile`	cut off The target coverage of events A percentile of 0.9 will return the most common activity types of the eventlog, which account for 90% of the events.

`filter_endpoints` *Filter: Filter based on percentile of start and end activities*

Description

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

Usage

```
filter_endpoints(eventlog, start_activities = NULL, end_activities = NULL,
  percentile_cut_off = NULL, reverse = F)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`start_activities`	Start activities used for filtering.
`end_activities`	End activities used for filtering.
`percentile_cut_off`	Alternatively to using (sets of) start or end activities, a percentile cut off can be provided. A percentile cut off value of 0.9 will return the cases starting and ending with the 90% most common start and end activities. When `reverse` is set to TRUE, it will return the 10% cases with the least common start and end activivities.
`reverse`	A logical parameter depicting whether the selection should be reversed.

`filter_precedence` *Filter: precedence relations*

Description

Filters cases based on the precedence relations between two sets of activities: antecedents and consequent. The filter can detect directly following activities as well as eventually following activites.

Usage

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

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`antecedents,`	consequents The set of antecendent and consequent activities. All pairs of antecedents and consequents are checked for.
`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 `each`, 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.
`reverse`	A logical parameter depicting whether the selection should be reversed.

filter_throughput_time *Filter: Throughput Time*

Description

Filters cases based on their throughput time.

Usage

```
filter_throughput_time(eventlog, lower_threshold = NULL,
  upper_threshold = NULL, percentile_cut_off = NULL, reverse = F)
```

Arguments

eventlog	The event log to be used. An object of class eventlog.
lower_threshold	The lower duration threshold, specified in number of days. When reverse is FALSE, all cases with a lower duration are discarded.
upper_threshold	The upper duration threshold, specified in number of days. When reverse is FALSE, all cases with a higher duration are discarded.
percentile_cut_off	Alternatively to providing thresholds, a percentile cut off can be provided. A percentile cut off value of 0.9 will return the 90% shortest cases. When reverse is set to TRUE, it will return the 10% longest cases.
reverse	A logical parameter depicting whether the selection should be reversed.

filter_time_period *Filter: Time Period*

Description

Function to filter eventlog using a time period.

Usage

```
filter_time_period(eventlog, start_point, end_point,
  filter_method = "contained", reverse = FALSE)
```

Arguments

eventlog	The event log to be used. An object of class eventlog.
start_point	Start timestamp of the time period. This should be a date object.
end_point	End timestamp of the time period. This should be a data object.

`filter_method`
: Can be `contained, start, complete, intersecting` or `trim`. `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.

`reverse`
: A logical parameter depicting whether the selection should be reversed.

`filter_trace_frequency` *Filter: Trace frequency percentile*

Description

Filters the log based the frequency of traces, using an upper and lower threshold or a percentile cut off.

Usage

```
filter_trace_frequency(eventlog, lower_threshold = NULL,
  upper_threshold = NULL, percentile_cut_off = NULL, reverse = F)
```

Arguments

`eventlog`
: The event log to be used. An object of class `eventlog`.

`lower_threshold`
: The lower frequency threshold. When `reverse` is FALSE, all traces with a lower frequency are discarded.

`upper_threshold`
: The upper frequency threshold. When `reverse` is FALSE, all traces with a higher frequency are discarded.

`percentile_cut_off`
: Alternatively to providing thresholds, a percentile cut off can be provided. A percentile cut off value of 0.9 will return the most common traces, accounting for 90% of the cases. When `reverse` is set to TRUE, it will return the least common traces, acoounting for 10% of the cases.

`reverse`
: A logical parameter depicting whether the selection should be reversed.

`filter_trace_length` *Filter: Trace length percentile*

Description

Filters cases on length, using a percentile threshold.

Usage

```
filter_trace_length(eventlog, lower_threshold = NULL,
  upper_threshold = NULL, percentile_cut_off = NULL, reverse = F)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`percentile_cut_off`	Alternatively to providing thresholds, a percentile cut off can be provided. A percentile cut off value of 0.9 will return the 90% shortest cases. When `reverse` is set to TRUE, it will return the 10% longest cases.
`reverse`	A logical parameter depicting whether the selection should be reversed.

`filter_trim` *Filter: Trim cases*

Description

Trim all cases from the first event of a set of start activities to the last event of a set of end activities. Traces that don't have at least one event of both sets are discarded.

Usage

```
filter_trim(eventlog, start_activities = NULL, end_activities = NULL,
  reverse = F)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`start_activities`	Start activities used for trimming. If not provided, the start of the cases is not trimmed.
`end_activities`	End activities used for trimming. If not provided, the end of the cases or not trimmed.
`reverse`	A logical parameter depicting whether the selection should be reversed.

`lifecycle_id` *Life cycle classifier*

Description

Get the life_cycle_id of an object of class `eventlog`

Usage

```
lifecycle_id(eventlog)
```

Arguments

`eventlog`	An object of class `eventlog`.

See Also

`eventlog`, `activity_instance_id`

`mapping` *Mapping*

Description

Prints the mapping of an event log object.

Usage

```
mapping(eventlog)
```

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

Description

Returns the number of selfloops in each trace. Can be performed at the level of traces, activities, or the level of the event log.

Usage

```
number_of_selfloops(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of selfloops should be performed: log, trace, case, activity

`number_of_traces` *Metric: Number of traces*

Description

Computes how many traces there are. The result is returned as absolute number as well as a relative number. The relative number refers to the number of traces per 100 cases.

Usage

```
number_of_traces(eventlog)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.

`number_of_traces_with_selfloop` *Metric: Number of Traces with Selfloop*

Description

Returns the number of traces in which one or more selfloops occur, both in absolute and relative numbers.

Usage

```
number_of_traces_with_selfloop(eventlog)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.

`n_activities` *n_activities*

Description

n_activities

Usage

```
n_activities(eventlog)
```

`n_activity_instances` *n_activity_instances*

Description

n_activity_instances

Usage

```
n_activity_instances(eventlog)
```

`n_cases` *n_cases*

Description

n_cases

Usage

```
n_cases(eventlog)
```

`n_events` *n_events*

Description

n_events

Usage

```
n_events(eventlog)
```

`n_traces` *n_traces*

Description

n_traces

Usage

```
n_traces(eventlog)
```

`print.eventlog` *Generic print function for eventlog*

Description

Generic print function for eventlog

Usage

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

`processing_time` *Metric: Processing time*

Description

Provides summary statistics about the processing time of events on the level of activities, traces, cases or log.

Usage

```
processing_time(eventlog, level_of_analysis, units = "days")
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of processing times should be performed: log, trace, case or activity.
`units`	The time unit in which the throughput times should be reported.

`repetitions` *Metric: Repetitions*

Description

Provides summuary statistics on the number of repetitions, at the level of activity types, traces, cases and the eventlog.

Usage

```
repetitions(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of repetitions should be performed: log, case, trace or activity.

`size_of_selfloops` *Metric: Size of selfloops*

Description

Provides summary statistics on the sizes of selfloops at the level of activity types, cases, traces or log. A selfloop of size x refers to the occurrence of x consecutive events of that activity type.

Usage

```
size_of_selfloops(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of selfloops should be performed: log, case, trace or activity.

`start_activities` *Metric: Start activities*

Description

At log level, computes how many activity types occur as the first event in a case, both absolute and relative. At activity level, shows the activities which occur as first, and how often. The first event in a case is the one which started the first. #'

Usage

```
start_activities(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of start activities should be performed: log, case or activity.

`summary.eventlog` *Generic summary function for eventlog class*

Description

Generic summary function for eventlog class

Usage

```
## S3 method for class 'eventlog'
summary(object, ...)
```

`throughput_time` *Metric: Throughput time of cases*

Description

Provides summary statistics concerning the throughput times of cases. The throughput time of cases is defined as the time between the start of the first event and the completion of the last event. Can be performed at the level of the log as well as the level of traces and cases.

Usage

```
throughput_time(eventlog, level_of_analysis, units = "days")
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of throughput times should be performed: log, case or trace.
`units`	The time unit in which the throughput times should be reported.

`timestamp` *Timestamp classifier*

Description

Get the timestamp classifier of an object of class `eventlog`

Usage

```
timestamp(eventlog)
```

Arguments

`eventlog`	An object of class `eventlog`.

See Also

`eventlog`

Examples

```
data(example_log)
timestamp(example_log)
```

`traces` *Traces*

Description

`traces` computes the different activity sequences of an event log together with their absolute and relative frequencies. Activity sequences are based on the start timestamp of activities.

Usage

```
traces(eventlog, output_traces = TRUE, output_cases = FALSE)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`output_traces,output_cases`	Logicals specifying what should be returned, a list of traces or a list of cases. If both are TRUE, a list of both is returned.

See Also

`cases`, `eventlog`

Examples

```
data(example_log)
traces(example_log)
```

`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

Trace: The absolute and relative frequency of each trace is returned

Case: for each case, the coverage of the corresponding trace is returned

Log: The number of traces to cover a certain percentage (default is 80%) of a log is computed. If a tie exists, the two nearest points are returned.

Usage

```
trace_coverage(eventlog, level_of_analysis, threshold = NULL)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of coverage should be performed: log, case or trace.
`threshold`	The threshold to be used for the analysis at log level. Default is at 0.8 (80%)

`trace_length` *Metric: Trace length*

Description

Computes the length of each trace, in terms of the number of events, at the level of the eventlog or the level of a trace. The relative numbers at trace level measure trace length compared to the average trace length of the top 80

Usage

```
trace_length(eventlog, level_of_analysis)
```

Arguments

`eventlog`	The event log to be used. An object of class `eventlog`.
`level_of_analysis`	At which level the analysis of trace_length should be performed: log, case or trace.

`write_xes` *Write XES file*

Description

Write XES file

Usage

```
write_xes(eventlog, case_attributes = NULL, file)
```