

# Package ‘edeaR’

July 10, 2022

**Type** Package

**Title** Exploratory and Descriptive Event-Based Data Analysis

**Version** 0.9.0

**Date** 2022-07-10

**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(>= 3.5.0)

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

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**URL** <https://bupar.net/>, <https://github.com/bupaverse/edeaR/>

**Suggests** knitr, eventdataR, rmarkdown, covr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**BugReports** <https://github.com/bupaverse/edeaR/issues/>

**Config/testthat/edition** 3

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2022-07-10 18:00:02 UTC

**R topics documented:**

activity_frequency . . . . .	3
activity_presence . . . . .	5
add_fixed_holiday . . . . .	7
add_floating_holiday . . . . .	8
add_holiday_periods . . . . .	8
augment . . . . .	9
calculate_queuing_length . . . . .	10
calculate_queuing_times . . . . .	11
change_day . . . . .	12
create_work_schedule . . . . .	12
edeaR . . . . .	13
end_activities . . . . .	13
filter_activity_instance . . . . .	15
filter_activity_presence . . . . .	17
filter_case_condition . . . . .	19
filter_endpoints_condition . . . . .	20
filter_infrequent_flows . . . . .	22
filter_precedence_condition . . . . .	23
filter_precedence_resource . . . . .	25
filter_resource_frequency . . . . .	27
filter_trace . . . . .	29
filter_trim . . . . .	30
filter_trim_lifecycle . . . . .	32
idle_time . . . . .	34
number_of_repetitions . . . . .	36
number_of_selfloops . . . . .	39
number_of_traces . . . . .	42
plot . . . . .	43
print.work_schedule . . . . .	44
processing_time . . . . .	45
redo_repetitions_referral_matrix . . . . .	48
redo_selfloops_referral_matrix . . . . .	49
resource_frequency . . . . .	50
resource_involvement . . . . .	52
resource_specialisation . . . . .	54
size_of_repetitions . . . . .	56
size_of_selfloops . . . . .	58
start_activities . . . . .	60
throughput_time . . . . .	62
trace_coverage . . . . .	65
trace_length . . . . .	67

---

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(  
  log,  
  level = c("log", "trace", "activity", "case"),  
  append = deprecated(),  
  append_column = NULL,  
  sort = TRUE,  
  eventlog = deprecated()  
)
```

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

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

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

```

)

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

```

### Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "trace", "case", or "activity". For more information, see <code>vignette("metrics", "eadeR")</code> and 'Details' below.
append	<a href="#">logical</a> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<a href="#">logical</a> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	[ <b>Deprecated</b> ]; please use log instead.

### Details

Argument `level` has the following options:

- At `log` level, this metric shows the summary statistics of the frequency of activities throughout the complete log.
- On `case` level, this metric shows the 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 `activity` level, this metric provides the absolute and relative frequency of a specific activity in the complete log.

### Methods (by class)

- `eventlog`: Computes the activity frequency for an [eventlog](#).
- `grouped_eventlog`: Computes the activity frequency for a [grouped\\_eventlog](#).
- `activitylog`: Computes the activity frequency for an [activitylog](#).
- `grouped_activitylog`: Computes the activity frequency for a [grouped\\_activitylog](#).

## References

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

## See Also

Other metrics: [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

activity\_presence      *Metric: Activity Presence*

---

## Description

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

## Usage

```
activity_presence(  
  log,  
  append = deprecated(),  
  append_column = NULL,  
  sort = TRUE,  
  eventlog = deprecated()  
)  
  
## S3 method for class 'eventlog'  
activity_presence(  
  log,  
  append = deprecated(),  
  append_column = NULL,  
  sort = TRUE,  
  eventlog = deprecated()  
)  
  
## S3 method for class 'grouped_eventlog'  
activity_presence(  
  log,  
  append = deprecated(),  
  append_column = NULL,  
  sort = TRUE,  
  eventlog = deprecated()  
)  
  
## S3 method for class 'activitylog'
```

```

activity_presence(
    log,
    append = deprecated(),
    append_column = NULL,
    sort = TRUE,
    eventlog = deprecated()
)

## S3 method for class 'grouped_activitylog'
activity_presence(
    log,
    append = deprecated(),
    append_column = NULL,
    sort = TRUE,
    eventlog = deprecated()
)

```

### Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
append	<a href="#">logical</a> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<a href="#">logical</a> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use <a href="#">log</a> instead.

### 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 an [eventlog](#).
- [grouped\\_eventlog](#): Compute activity presence for a [grouped\\_eventlog](#).
- [activitylog](#): Compute activity presence for an [activitylog](#).
- [grouped\\_activitylog](#): Compute activity presence for a [grouped\\_activitylog](#).

### References

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

**See Also**

Other metrics: [activity\\_frequency\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

**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)
```

---

add\_fixed\_holiday      *Add fixed holiday to work schedule*

---

**Description**

Add fixed holiday to work schedule

**Usage**

```
add_fixed_holiday(work_schedule, name, month, day)
```

**Arguments**

work_schedule	Work schedule created with create_work_schedule
name	Name of holiday
month	Month in which fixed holiday takes place
day	Day of fixed holiday

---

add\_floating\_holiday    *Add floating holiday to work schedule*

---

**Description**

Add floating holiday to work schedule

**Usage**

```
add_floating_holiday(work_schedule, name, dates)
```

**Arguments**

work_schedule	Work schedule created with create_work_schedule
name	Name of holiday
dates	Dates of floating holiday. Make sure to list all dates relevant to your time frame

---

add\_holiday\_periods    *Add holiday period to work schedule*

---

**Description**

Add holiday period to work schedule

**Usage**

```
add_holiday_periods(work_schedule, from, to)
```

**Arguments**

work_schedule	Work schedule created with create_work_schedule
from	Start of holiday period (included)
to	End of holiday period (included)



---

 augment

---

*Augment log with results from metric computation.*


---

### Description

Augment log with results from metric computation.

### Usage

```
augment(metric, log, columns, prefix = "")

## S3 method for class 'log_metric'
augment(metric, log, columns, prefix = "")

## S3 method for class 'case_metric'
augment(metric, log, columns, prefix = "")

## S3 method for class 'activity_metric'
augment(metric, log, columns, prefix = "")

## S3 method for class 'resource_metric'
augment(metric, log, columns, prefix = "")

## S3 method for class 'resource_activity_metric'
augment(metric, log, columns, prefix = "")

## S3 method for class 'trace_metric'
augment(metric, log, columns, prefix = "")
```

### Arguments

metric	Metric computed by edeaR
log	Object of class eventlog or activitylog that was used to compute the metric.
columns	A character vector of column names from the metric that you want to add to the log. If missing, defaults to all columns.
prefix	A character prefix to be added to the newly added metric columns in the log.

### Value

Object of class eventlog or activitylog. Same class as the log input.

### Methods (by class)

- log\_metric: Augment log metric
- case\_metric: Augment case metric
- activity\_metric: Augment activity metric

- resource\_metric: Augment resource metric
- resource\_activity\_metric: Augment resource-activity metric
- trace\_metric: Augment trace metric

---

calculate\_queuing\_length

*Calculate queuing length*

---

## Description

**[Experimental]**

## Usage

```
calculate_queuing_length(
    queuing_times,
    level = c("log", "activity", "resource"),
    time_interval
)
```

## Arguments

- queuing\_times Object of class queuing\_times, returned by [calculate\\_queuing\\_times](#).
- level [character](#) (default "log"): Level of granularity for the analysis: "log", "activity", "resource". For more information, see 'Details' below.
- time\_interval The time interval after which the queue length should be calculated. For more information, see 'Details' below and the by argument of [seq.Date](#).

## Details

Argument level has the following options:

- At log level, this metric calculates the total number of activity instances that are queued at a given moment in time.
- At resource level, this metric calculates the total number activity instances that are queued for a given resource.
- On activity level, this metric calculates the total number of activity instances that are queue for a given activity type.

Argument time\_interval has the following options (see also the by argument of [seq.Date](#)):

- A [numeric](#) as number of days.
- An object of class [difftime](#).
- A [character](#) string, which could be one of "day", "week", "month", "quarter", or "year". The first day for which queue length is calculated, is the first timestamp found in the log.

## See Also

[calculate\\_queuing\\_times](#), [seq.Date](#)

---

```
calculate_queuing_times
    Calculate queuing times
```

---

## Description

**[Experimental]**

## Usage

```
calculate_queuing_times(
    log,
    units = c("auto", "secs", "mins", "hours", "days", "weeks"),
    eventlog = deprecated()
)

## S3 method for class 'eventlog'
calculate_queuing_times(
    log,
    units = c("auto", "secs", "mins", "hours", "days", "weeks"),
    eventlog = deprecated()
)

## S3 method for class 'activitylog'
calculate_queuing_times(
    log,
    units = c("auto", "secs", "mins", "hours", "days", "weeks"),
    eventlog = deprecated()
)
```

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
units	<a href="#">character</a> (default "auto"): The time unit in which the throughput times should be reported. Should be one of the following values: "auto" (default), "secs", "mins", "hours", "days", "weeks". See also the units argument of <a href="#">difftime</a> .
eventlog	<b>[Deprecated]</b> ; please use log instead.

## Value

Returns a list of all the activity instances, with the time they started, and the time since they were queued. Notice that this does not take into account any process model notion! The time since they are queued is the completion time of the previous activity in the log.

**Methods (by class)**

- eventlog: Calculate queueing times for [eventlog](#) and [grouped\\_eventlog](#).
- activitylog: Calculate queueing times for [activitylog](#) and [grouped\\_activitylog](#).

**References**

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

**See Also**

[difftime](#)

---

change_day	<i>Adjust days in work schedule</i>
------------	-------------------------------------

---

**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	<i>Create work schedule</i>
----------------------	-----------------------------

---

**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)

---

edeR

*edeR - 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

*End activities*

---

### Description

Analyse the end activities in the process.

### Usage

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

```

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

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

```

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see vignette("metrics", "edeaR") and 'Details' below.
append	<b>logical</b> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	[ <b>Deprecated</b> ]; please use log instead.

### Details

Argument level has the following options:

- At log level, this metric shows the absolute and relative number of activities that are the last activity in one or more of the cases.
- On case level, this metric provides an overview of the end activity of each case.
- On 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.

- At resource level, an overview of which resources execute the last activity per case is provided.
- On 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: Computes the end activities for an [eventlog](#).
- grouped\_eventlog: Computes the end activities for a [grouped\\_eventlog](#).
- activitylog: Computes the end activities for an [activitylog](#).
- grouped\_activitylog: Computes the end activities for a [grouped\\_activitylog](#).

### References

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

### See Also

[start\\_activities](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

filter\_activity\_instance

*Filter Activity Instance*

---

### Description

Filters the log based on activity instance identifier. This method has an `activity_instances` argument, to which a vector of identifiers can be given. The selection can be negated with the `reverse` argument.

### Usage

```
filter_activity_instance(
  log,
  activity_instances,
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'eventlog'
```

```

filter_activity_instance(
  log,
  activity_instances,
  reverse = FALSE,
  eventlog = deprecated()
)

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

ifilter_activity_instance(eventlog)

```

### Arguments

**log**                    **eventlog**: Object of class **eventlog** or derivatives (**grouped\_eventlog**).  
**activity\_instances**                    A vector of activity instance identifiers.  
**reverse**                    **logical** (default FALSE): Indicating whether the selection should be reversed.  
**eventlog**                    **[Deprecated]**; please use **log** instead.

### Value

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

### Methods (by class)

- **eventlog**: Filters activities for an **eventlog**.
- **grouped\_eventlog**: Filters activities for a **grouped\_eventlog**.

### References

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

### See Also

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\\_condition\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\\_resource\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\\_frequency\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trace\(\)](#), [filter\\_trim\\_lifecycle\(\)](#), [filter\\_trim\(\)](#)



---

```
filter_activity_presence
      Filter Activity Presence
```

---

## Description

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

## Usage

```
filter_activity_presence(
  log,
  activities = NULL,
  method = c("all", "none", "one_of", "exact", "only"),
  reverse = FALSE,
  eventlog = deprecated()
)

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

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

ifilter_activity_presence(eventlog)
```

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
activities	<a href="#">character</a> vector: Containing one or more activity identifiers.
method	<a href="#">character</a> (default "all"): Filter method: "all" (default), "none", "one_of", "exact", or "only". For more information, see 'Details' below.
reverse	<a href="#">logical</a> (default FALSE): Indicating whether the selection should be reversed.
eventlog	<b>[Deprecated]</b> ; please use log instead.

## 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 with following options:

- "all" 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.
- "exact" means that only exactly these activities can be present (although multiple times and in random orderings).
- "only" means that only (a set of) these activities are allowed to be present.

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

## Value

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

## Methods (by class)

- `log`: Filters activities for a `log`.
- `grouped_log`: Filters activities for a `grouped_log`.

## References

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

## See Also

Other filters: `filter_activity_frequency()`, `filter_activity_instance()`, `filter_activity()`, `filter_case_condition()`, `filter_case()`, `filter_endpoints_condition()`, `filter_endpoints()`, `filter_infrequent_flows()`, `filter_lifecycle_presence()`, `filter_lifecycle()`, `filter_precedence_condition()`, `filter_precedence_resource()`, `filter_precedence()`, `filter_processing_time()`, `filter_resource_frequency()`, `filter_resource()`, `filter_throughput_time()`, `filter_time_period()`, `filter_trace_frequency()`, `filter_trace_length()`, `filter_trace()`, `filter_trim_lifecycle()`, `filter_trim()`

---

 filter\_case\_condition *Filter Case Condition*


---

**Description**

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

**Usage**

```
filter_case_condition(
  log,
  ...,
  condition = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'log'
filter_case_condition(
  log,
  ...,
  condition = deprecated(),
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'grouped_log'
filter_case_condition(
  log,
  ...,
  condition = deprecated(),
  reverse = FALSE,
  eventlog = deprecated()
)
```

**Arguments**

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
...	<a href="#">data-masking</a> : Expressions that return a logical value, and are defined in terms of the variables in <a href="#">log</a> . If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept. For more information, see <a href="#">filter</a> .
condition	<b>[Deprecated]</b> ; please use <a href="#">data-masking</a> expressions instead.
reverse	<a href="#">logical</a> (default FALSE): Indicating whether the selection should be reversed.
eventlog	<b>[Deprecated]</b> ; please use <a href="#">log</a> instead.

**Value**

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

**Methods (by class)**

- `log`: Filters cases for a `log`.
- `grouped_log`: Filters cases for a `grouped_log`.

**References**

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

**See Also**

`filter`

Other filters: `filter_activity_frequency()`, `filter_activity_instance()`, `filter_activity_presence()`, `filter_activity()`, `filter_case()`, `filter_endpoints_condition()`, `filter_endpoints()`, `filter_infrequent_flows()`, `filter_lifecycle_presence()`, `filter_lifecycle()`, `filter_precedence_condition()`, `filter_precedence_resource()`, `filter_precedence()`, `filter_processing_time()`, `filter_resource_frequency()`, `filter_resource()`, `filter_throughput_time()`, `filter_time_period()`, `filter_trace_frequency()`, `filter_trace_length()`, `filter_trace()`, `filter_trim_lifecycle()`, `filter_trim()`

---

filter\_endpoints\_condition

*Filter Start and End Conditions*

---

**Description**

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

**Usage**

```
filter_endpoints_condition(
  log,
  start_condition = NULL,
  end_condition = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'eventlog'
filter_endpoints_condition(
  log,
```

```
    start_condition = NULL,
    end_condition = NULL,
    reverse = FALSE,
    eventlog = deprecated()
)

## S3 method for class 'grouped_log'
filter_endpoints_condition(
  log,
  start_condition = NULL,
  end_condition = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'activitylog'
filter_endpoints_condition(
  log,
  start_condition = NULL,
  end_condition = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)

filter_endpoints_conditions(
  log,
  start_condition = NULL,
  end_condition = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)
```

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
start_condition, end_condition	A logical condition.
reverse	<a href="#">logical</a> (default FALSE): Indicating whether the selection should be reversed.
eventlog	<b>[Deprecated]</b> ; please use <a href="#">log</a> instead.

## Value

When given an object of type [log](#), it will return a filtered [log](#). When given an object of type [grouped\\_log](#), the filter will be applied in a stratified way (i.e. each separately for each group). The returned log will be grouped on the same variables as the original log.

**Methods (by class)**

- eventlog: Filters cases for an [eventlog](#).
- grouped\_log: Filters cases for a [grouped\\_log](#).
- activitylog: Filters cases for an [activitylog](#).

**References**

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

**See Also**

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_instance\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\\_resource\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\\_frequency\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trace\(\)](#), [filter\\_trim\\_lifecycle\(\)](#), [filter\\_trim\(\)](#)

---

filter\_infrequent\_flows

*Filter Infrequent Flows*

---

**Description****[Experimental]**

Filter cases based on infrequent flows.

**Usage**

```
filter_infrequent_flows(log, min_n, eventlog = deprecated())

## S3 method for class 'eventlog'
filter_infrequent_flows(log, min_n, eventlog = deprecated())

## S3 method for class 'grouped_eventlog'
filter_infrequent_flows(log, min_n, eventlog = deprecated())

## S3 method for class 'activitylog'
filter_infrequent_flows(log, min_n, eventlog = deprecated())

## S3 method for class 'grouped_activitylog'
filter_infrequent_flows(log, min_n, eventlog = deprecated())
```

**Arguments**

log	log: Object of class <code>log</code> or derivatives ( <code>grouped_log</code> , <code>eventlog</code> , <code>activitylog</code> , etc.).
min_n	numeric: Cases containing a flow that occurs less than <code>min_n</code> times are discarded.
eventlog	<b>[Deprecated]</b> ; please use <code>log</code> instead.

**Value**

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

**Methods (by class)**

- `eventlog`: Filters infrequent flows for an `eventlog`.
- `grouped_eventlog`: Filters infrequent flows for a `grouped_eventlog`.
- `activitylog`: Filters infrequent flows for an `activitylog`.
- `grouped_activitylog`: Filters infrequent flows for a `grouped_activitylog`.

**References**

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

**See Also**

Other filters: `filter_activity_frequency()`, `filter_activity_instance()`, `filter_activity_presence()`, `filter_activity()`, `filter_case_condition()`, `filter_case()`, `filter_endpoints_condition()`, `filter_endpoints()`, `filter_lifecycle_presence()`, `filter_lifecycle()`, `filter_precedence_condition()`, `filter_precedence_resource()`, `filter_precedence()`, `filter_processing_time()`, `filter_resource_frequency()`, `filter_resource()`, `filter_throughput_time()`, `filter_time_period()`, `filter_trace_frequency()`, `filter_trace_length()`, `filter_trace()`, `filter_trim_lifecycle()`, `filter_trim()`

---

filter\_precedence\_condition

*Filter Precedence Relations*

---

**Description**

Filters cases based on the precedence relations between two sets of activities. For more information, see `filter_precedence`.

**Usage**

```

filter_precedence_condition(
  log,
  antecedent_condition,
  consequent_condition,
  precedence_type = c("directly_follows", "eventually_follows"),
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'log'
filter_precedence_condition(
  log,
  antecedent_condition,
  consequent_condition,
  precedence_type = c("directly_follows", "eventually_follows"),
  reverse = FALSE,
  eventlog = deprecated()
)

## S3 method for class 'grouped_log'
filter_precedence_condition(
  log,
  antecedent_condition,
  consequent_condition,
  precedence_type = c("directly_follows", "eventually_follows"),
  reverse = FALSE,
  eventlog = deprecated()
)

```

**Arguments**

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
antecedent_condition, consequent_condition	The antecedent and consequent conditions.
precedence_type	<b>character</b> (default "directly_follows"): When "directly_follows", the consequent activity should happen immediately after the antecedent activities. When "eventually_follows", other events are allowed to happen in between.
reverse	<b>logical</b> (default FALSE): Indicating whether the selection should be reversed.
eventlog	<b>[Deprecated]</b> ; please use log instead.

**Value**

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



**Methods (by class)**

- `log`: Filters cases for a `log`.
- `grouped_log`: Filters cases for a `grouped_log`.

**References**

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

**See Also**

Other filters: `filter_activity_frequency()`, `filter_activity_instance()`, `filter_activity_presence()`, `filter_activity()`, `filter_case_condition()`, `filter_case()`, `filter_endpoints_condition()`, `filter_endpoints()`, `filter_infrequent_flows()`, `filter_lifecycle_presence()`, `filter_lifecycle()`, `filter_precedence_resource()`, `filter_precedence()`, `filter_processing_time()`, `filter_resource_frequency()`, `filter_resource()`, `filter_throughput_time()`, `filter_time_period()`, `filter_trace_frequency()`, `filter_trace_length()`, `filter_trace()`, `filter_trim_lifecycle()`, `filter_trim()`

---

filter\_precedence\_resource

*Filter Precedence Relations with Identical Resources*

---

**Description**

Filters cases based on the precedence relations between two sets of activities, where both antecedent and consequent have to be executed by the same resource. For more information, see [filter\\_precedence](#).

**Usage**

```
filter_precedence_resource(
  log,
  antecedents,
  consequents,
  precedence_type = c("directly_follows", "eventually_follows"),
  filter_method = c("all", "one_of", "none"),
  reverse = FALSE,
  eventlog = deprecated()
)
```

```
## S3 method for class 'log'
filter_precedence_resource(
  log,
  antecedents,
  consequents,
  precedence_type = c("directly_follows", "eventually_follows"),
  filter_method = c("all", "one_of", "none"),
```

```

reverse = FALSE,
eventlog = deprecated()
)

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

```

### Arguments

**log** **log**: Object of class **log** or derivatives (**grouped\_log**, **eventlog**, **activitylog**, etc.).

**antecedents, consequents** **character** vector: The set of antecedent and consequent activities. Both are **character** vectors containing at least one activity identifier. All pairs of antecedents and consequents are turned into separate precedence rules.

**precedence\_type** **character** (default "directly\_follows"): 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** **character** (default "all"): 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** (default FALSE): Indicating whether the selection should be reversed.

**eventlog** **[Deprecated]**; please use **log** instead.

### Value

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

### Methods (by class)

- **log**: Filters cases for a **log**.
- **grouped\_log**: Filters cases for a **grouped\_log**.

## References

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

## See Also

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_instance\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\\_condition\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\\_frequency\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trace\(\)](#), [filter\\_trim\\_lifecycle\(\)](#), [filter\\_trim\(\)](#)

---

filter\_resource\_frequency

*Filter Resource Frequency*

---

## Description

Filters the log based on frequency of resources

## Usage

```
filter_resource_frequency(  
  log,  
  interval = NULL,  
  percentage = NULL,  
  reverse = FALSE,  
  eventlog = deprecated()  
)  
  
## S3 method for class 'log'  
filter_resource_frequency(  
  log,  
  interval = NULL,  
  percentage = NULL,  
  reverse = FALSE,  
  eventlog = deprecated()  
)  
  
## S3 method for class 'grouped_log'  
filter_resource_frequency(  
  log,  
  interval = NULL,  
  percentage = NULL,  
  reverse = FALSE,  
  eventlog = deprecated()  
)
```

```
)
ifilter_resource_frequency(eventlog)
```

### Arguments

`log`                    **log**: Object of class **log** or derivatives (**grouped\_log**, **eventlog**, **activitylog**, etc.).

`percentage, interval`                    The target coverage of activity instances. Provide either `percentage` or `interval`.  
`percentage` (**numeric**): A percentile of `p` will return the most common resource types of the log, which account for at least `p%` of the activity instances.  
`interval` (**numeric** vector of length 2): A resource frequency interval. Half open interval can be created using **NA**.  
For more information, see 'Details' below.

`reverse`                    **logical** (default **FALSE**): Indicating whether the selection should be reversed.

`eventlog`                    **[Deprecated]**; please use `log` instead.

### Details

Filtering the log based on 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%` of the activity instances, starting from the resource labels with the highest 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.

### Value

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

### Methods (by class)

- `log`: Filters resources for a **log**.
- `grouped_log`: Filters resources for a **grouped\_log**.

### References

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

**See Also**

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_instance\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\\_condition\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\\_resource\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trace\(\)](#), [filter\\_trim\\_lifecycle\(\)](#), [filter\\_trim\(\)](#)

filter\_trace

*Filter Trace***Description**

Filters the log based on trace identifier.

This method can be used to filter on trace identifier, which can be obtained from [case\\_list](#). It has a `trace_ids` argument, to which a vector of identifiers can be given. The selection can be negated with the `reverse` argument.

**Usage**

```
filter_trace(log, trace_ids, reverse = FALSE, eventlog = deprecated())
```

```
## S3 method for class 'log'
```

```
filter_trace(log, trace_ids, reverse = FALSE, eventlog = deprecated())
```

```
## S3 method for class 'grouped_log'
```

```
filter_trace(log, trace_ids, reverse = FALSE, eventlog = deprecated())
```

**Arguments**

<code>log</code>	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
<code>trace_ids</code>	<a href="#">character</a> vector: A vector of trace identifiers
<code>reverse</code>	<a href="#">logical</a> (default FALSE): Indicating whether the selection should be reversed.
<code>eventlog</code>	<b>[Deprecated]</b> ; please use <code>log</code> instead.

**Value**

When given an object of type [log](#), it will return a filtered [log](#). When given an object of type [grouped\\_log](#), the filter will be applied in a stratified way (i.e. each separately for each group). The returned log will be grouped on the same variables as the original log.

**Methods (by class)**

- `log`: Filters cases for a [log](#).
- `grouped_log`: Filters cases for a [grouped\\_log](#).

**References**

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

**See Also**

[case\\_list](#)

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_instance\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\\_condition\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\\_resource\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\\_frequency\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trim\\_lifecycle\(\)](#), [filter\\_trim\(\)](#)

---

filter\_trim

*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.

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.

**Usage**

```
filter_trim(
  log,
  start_activities = NULL,
  end_activities = NULL,
  reverse = FALSE,
  eventlog = deprecated()
)

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

```

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

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

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

ifilter_trim(eventlog)

```

### Arguments

**log**                    **log**: Object of class **log** or derivatives (**grouped\_log**, **eventlog**, **activitylog**, etc.).  
**start\_activities**, **end\_activities**                    **character** vector (default **NULL**): A vector of activity identifiers, or **NULL**.  
**reverse**                    **logical** (default **FALSE**): Indicating whether the selection should be reversed.  
**eventlog**                    **[Deprecated]**; please use **log** instead.

### Value

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

### Methods (by class)

- **eventlog**: Filters activity instances for an **eventlog**.

- `grouped_eventlog`: Filters activity instances for a `grouped_eventlog`.
- `activitylog`: Filters activity instances for an `activitylog`.
- `grouped_activitylog`: Filters activity instances for a `grouped_activitylog`.

## References

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

## See Also

Other filters: `filter_activity_frequency()`, `filter_activity_instance()`, `filter_activity_presence()`, `filter_activity()`, `filter_case_condition()`, `filter_case()`, `filter_endpoints_condition()`, `filter_endpoints()`, `filter_infrequent_flows()`, `filter_lifecycle_presence()`, `filter_lifecycle()`, `filter_precedence_condition()`, `filter_precedence_resource()`, `filter_precedence()`, `filter_processing_time()`, `filter_resource_frequency()`, `filter_resource()`, `filter_throughput_time()`, `filter_time_period()`, `filter_trace_frequency()`, `filter_trace_length()`, `filter_trace()`, `filter_trim_lifecycle()`

---

`filter_trim_lifecycle` *Filter Trim Life Cycle*

---

## 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.

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.

## Usage

```
filter_trim_lifecycle(
  log,
  start_lifecycles = NULL,
  end_lifecycles = NULL,
  reverse = FALSE,
  start_lifecycle = deprecated(),
  end_lifecycle = deprecated(),
  eventlog = deprecated()
)
```



```

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

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

ifilter_trim_lifecycle(eventlog)

```

### Arguments

**log**            **eventlog**: Object of class **eventlog** or derivatives (**grouped\_eventlog**).  
**start\_lifecycles**, **end\_lifecycles**    **character** vector (default **NULL**): A vector of life cycle identifiers, or **NULL**.  
**reverse**        **logical** (default **FALSE**): Indicating whether the selection should be reversed.  
**start\_lifecycle**                    **[Deprecated]**; please use **start\_lifecycles** instead.  
**end\_lifecycle**    **[Deprecated]**; please use **end\_lifecycles** instead.  
**eventlog**        **[Deprecated]**; please use **log** instead.

### Value

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

### Methods (by class)

- **eventlog**: Filters activity instances for an **eventlog**.
- **grouped\_eventlog**: Filters activity instances for a **grouped\_eventlog**.

## References

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

## See Also

[lifecycle\\_id](#)

Other filters: [filter\\_activity\\_frequency\(\)](#), [filter\\_activity\\_instance\(\)](#), [filter\\_activity\\_presence\(\)](#), [filter\\_activity\(\)](#), [filter\\_case\\_condition\(\)](#), [filter\\_case\(\)](#), [filter\\_endpoints\\_condition\(\)](#), [filter\\_endpoints\(\)](#), [filter\\_infrequent\\_flows\(\)](#), [filter\\_lifecycle\\_presence\(\)](#), [filter\\_lifecycle\(\)](#), [filter\\_precedence\\_condition\(\)](#), [filter\\_precedence\\_resource\(\)](#), [filter\\_precedence\(\)](#), [filter\\_processing\\_time\(\)](#), [filter\\_resource\\_frequency\(\)](#), [filter\\_resource\(\)](#), [filter\\_throughput\\_time\(\)](#), [filter\\_time\\_period\(\)](#), [filter\\_trace\\_frequency\(\)](#), [filter\\_trace\\_length\(\)](#), [filter\\_trace\(\)](#), [filter\\_trim\(\)](#)

---

idle\_time

*Idle Time*

---

## Description

Calculates the amount of time that no activity occurs.

## Usage

```
idle_time(
  log,
  level = c("log", "trace", "case", "resource"),
  append = deprecated(),
  append_column = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  sort = TRUE,
  eventlog = deprecated()
)
```

```
## S3 method for class 'eventlog'
```

```
idle_time(
  log,
  level = c("log", "trace", "case", "resource"),
  append = deprecated(),
  append_column = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  sort = TRUE,
  eventlog = deprecated()
)
```

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

```

idle_time(
  log,
  level = c("log", "case", "trace", "resource"),
  append = deprecated(),
  append_column = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  sort = TRUE,
  eventlog = deprecated()
)

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

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

```

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "trace", "case", or "resource". For more information, see vignette("metrics", "eDear") and 'Details' below.
append	<a href="#">logical</a> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
units	<a href="#">character</a> (default "auto"): The time unit in which the throughput times should be reported. Should be one of the following values: "auto" (default), "secs", "mins", "hours", "days", "weeks". See also the units argument of <a href="#">difftime</a> .
sort	<a href="#">logical</a> (default TRUE): Sort by decreasing idle time. Only relevant for "trace" and "resource" level.

eventlog **[Deprecated]**; please use log instead.

### Details

Argument level has the following options:

- At "log" level, the idle time metric provides an overview of summary statistics of the idle time per case, aggregated over the complete log.
- On "trace" level, the idle time metric provides an overview of the summary statistics of the idle time for each trace in the log.
- On "case" level, the idle time metric provides an overview of the total idle time per case
- On "resource" level, this metric can be used to get an insight in the amount of time each resource "wastes" during the process.

### Methods (by class)

- eventlog: Computes the idle time for an [eventlog](#).
- grouped\_eventlog: Computes the idle time for a [grouped\\_eventlog](#).
- activitylog: Computes the idle time for an [activitylog](#).
- grouped\_activitylog: Computes the idle time for a [grouped\\_activitylog](#).

### References

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

### See Also

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

number\_of\_repetitions *Number of Repetitions*

---

### Description

Provides information statistics on the number of repetitions

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.

**Usage**

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

```

level = c("log", "case", "activity", "resource", "resource-activity"),
append = deprecated(),
append_column = NULL,
sort = TRUE,
eventlog = deprecated()
)

```

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
type	<b>character</b> (default "all"): The type of repetitions: "all" (default), "repeat", or "redo". For more information, see 'Details' below.
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see <code>vignette("metrics", "e4eaR")</code> and 'Details' below.
append	<b>logical</b> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use log instead.

### Details

Argument level has the following options:

- At "log" level, this metric shows the summary statistics of the number of repetitions within a case, which can provide insights in the amount of waste in a 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 "activity" level, 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.
- On "resource" level, 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.
- On "resource-activity" level, this metric provides 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 different relative numbers are provided, one relative to the total number of executions of the activity in the complete log, and one relative to the total number of executions performed by the resource throughout the complete log.

Similar to the [self-loop](#) metric, a distinction should be made between "repeat" and "redo" repetitions, as can be set by the type argument:

- "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.

### Methods (by class)

- `eventlog`: Computes the number of repetitions for an [eventlog](#).
- `grouped_eventlog`: Computes the number of repetitions for a [grouped\\_eventlog](#).
- `activitylog`: Computes the number of repetitions for an [activitylog](#).
- `grouped_activitylog`: Computes the number of repetitions for a [grouped\\_activitylog](#).

### References

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

### See Also

[number\\_of\\_selfloops](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

number\_of\_selfloops    *Number of Self-loops*

---

### Description

Provides information statistics on the number of self-loops in a trace.

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".

**Usage**

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



```

    level = c("log", "case", "activity", "resource", "resource-activity"),
    append = deprecated(),
    append_column = NULL,
    sort = TRUE,
    eventlog = deprecated()
)

```

## Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
type	<b>character</b> (default "all"): The type of repetitions: "all" (default), "repeat", or "redo". For more information, see 'Details' below.
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see <code>vignette("metrics", "edeaR")</code> and 'Details' below.
append	<b>logical</b> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	[ <b>Deprecated</b> ]; please use log instead.

## Details

Two types of self-loops are defined, which can be chosen using the type argument:

- "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.

Argument level has the following options:

- At "log" level, the summary statistics of the number of self-loops within a trace can give a first insight in the amount of waste in a 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.
- On "case" level, an overview is provided 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 "activity" level, the absolute and relative number of self-loops per activity can be an indication for which activities are causing the most waste in the process.
- On "resource" level, this metric can give insights into which resources needs to repeat their work most often within a case, or for which resource the work they did should be redone by another resource within the same case. This metric shows the absolute and relative number of both repeat and redo self-loops for each resource in the log.

- On "resource-activity" level, this metric can be used 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 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`: Computes the number of self-loops for an [eventlog](#).
- `grouped_eventlog`: Computes the number of self-loops for a [grouped\\_eventlog](#).
- `activitylog`: Computes the number of self-loops for an [activitylog](#).
- `grouped_activitylog`: Computes the number of self-loops for a [grouped\\_activitylog](#).

### References

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

### See Also

[number\\_of\\_repetitions](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

number_of_traces	<i>Number of Traces</i>
------------------	-------------------------

---

### Description

Computes how many traces there are.

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

### Usage

```
number_of_traces(log, eventlog = deprecated())

## S3 method for class 'log'
number_of_traces(log, eventlog = deprecated())

## S3 method for class 'grouped_log'
number_of_traces(log, eventlog = deprecated())
```

**Arguments**

- log                    [log](#): Object of class [log](#) or derivatives ([grouped\\_log](#), [eventlog](#), [activitylog](#), etc.).
- eventlog              **[Deprecated]**; please use [log](#) instead.

**Methods (by class)**

- [log](#): Number of traces in a [log](#).
- [grouped\\_log](#): Number of traces in a [grouped\\_log](#).

**References**

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

**See Also**

[traces](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

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.work\_schedule    *Print work schedule*

---

### Description

Print work schedule

**Usage**

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

**Arguments**

```
x          Work schedule to print
...       Additional arguments (ignored)
```

---

processing_time	<i>Processing Time</i>
-----------------	------------------------

---

**Description**

Provides summary statistics about the processing time of the process.

In contrast to the [throughput\\_time](#) of the cases in a 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 log, the specific cases, traces, the activities, and the resource-activity combinations.

**Usage**

```
processing_time(
  log,
  level = c("log", "trace", "case", "activity", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  sort = TRUE,
  work_schedule = NULL,
  eventlog = deprecated()
)
```

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

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

```

    log,
    level = c("log", "trace", "case", "activity", "resource", "resource-activity"),
    append = deprecated(),
    append_column = NULL,
    units = c("auto", "secs", "mins", "hours", "days", "weeks"),
    sort = TRUE,
    work_schedule = NULL,
    eventlog = deprecated()
)

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

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

```

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "trace", "case", "activity", "resource", or "resource-activity". For more information, see <code>vignette("metrics", "eideaR")</code> and 'Details' below.
append	<a href="#">logical</a> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
units	<a href="#">character</a> (default "auto"): The time unit in which the processing times should be reported. Should be one of the following values: "auto" (default), "secs",

	"mins", "hours", "days", "weeks". See also the units argument of <a href="#">difftime</a> .
sort	<a href="#">logical</a> (default TRUE): 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.
eventlog	<b>[Deprecated]</b> ; please use <code>log</code> instead.

## Details

Argument `level` has the following options:

- At "log" level, this metric calculates the summary statistics of the actual processing time per case, summarised over the complete event log.
- On "trace" level, the summary statistics of processing time can be calculated for each possible sequence of activities that appears in the event log.
- On "case" level, a list of cases with their processing time are provided.
- On "activity" level, an overview of the average processing time -or the service time- of each activity can be calculated.
- At "resource" level, this metric calculates the processing time per resource.
- On "resource-activity" level, the efficiency of resources by looking at the combination of each resource with each activity can be investigated.

## Methods (by class)

- `eventlog`: Computes processing time for an [eventlog](#).
- `grouped_eventlog`: Computes processing time for a [grouped\\_eventlog](#).
- `activitylog`: Computes processing time for an [activitylog](#).
- `grouped_activitylog`: Computes processing time for a [grouped\\_activitylog](#).

## References

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

## See Also

[throughput\\_time](#), [difftime](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

redo\_repetitions\_referral\_matrix  
*Referral matrix repetitons*

---

### Description

Provides a list of initiators and completers of redo repetitons

### Usage

```
redo_repetitions_referral_matrix(log, eventlog = deprecated())
```

```
## S3 method for class 'eventlog'  
redo_repetitions_referral_matrix(log, eventlog = deprecated())
```

```
## S3 method for class 'activitylog'  
redo_repetitions_referral_matrix(log, eventlog = deprecated())
```

### Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
eventlog	<b>[Deprecated]</b> ; please use log instead.

### Methods (by class)

- eventlog: Compute matrix for eventlog
- activitylog: Compute matrix for activitylog

### References

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

### See Also

[number\\_of\\_repetitions](#)



---

redo\_selfloops\_referral\_matrix  
*Referral matrix selfloops*

---

### Description

Provides a list of initiators and completers of redo selfloops

### Usage

```
redo_selfloops_referral_matrix(log, eventlog = deprecated())

## S3 method for class 'eventlog'
redo_selfloops_referral_matrix(log, eventlog = deprecated())

## S3 method for class 'activitylog'
redo_selfloops_referral_matrix(log, eventlog = deprecated())
```

### Arguments

`log`            [log](#): Object of class [log](#) or derivatives ([grouped\\_log](#), [eventlog](#), [activitylog](#), etc.).

`eventlog`       **[Deprecated]**; please use `log` instead.

### Methods (by class)

- `eventlog`: Compute matrix for `eventlog`
- `activitylog`: Compute matrix for `activitylog`

### References

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

### See Also

[number\\_of\\_selfloops](#)

---

resource_frequency	<i>Resource Frequency</i>
--------------------	---------------------------

---

**Description**

Analyses the frequency of resources at different levels of analysis.

**Usage**

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

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

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see <code>vignette("metrics", "edeaR")</code> and 'Details' below.
append	<b>logical</b> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if <code>append = TRUE</code> . Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	[ <b>Deprecated</b> ]; please use <code>log</code> instead.

### Details

Argument `level` has the following options:

- At "log" level, summary statistics show the number of times a resource executes an activity in the complete log.
- On "case" level, summary statistics of the frequency of resources can be used to get a better view on the variance between the different cases, to get an insight into the number of different resources working on each case together with the number of activities a resource executes per case.
- On "activity" level, the resource frequency states how many different resources are executing a specific activity in the complete log.
- On "resource" level, this metric simply shows the absolute and relative frequency of occurrences of each resource in the complete log.
- On "resource-activity" level, the absolute and relative number of times each resource-activity combination occurs in the complete 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: Computes the resource frequency for an [eventlog](#).
- grouped\_eventlog: Computes the resource frequency for a [grouped\\_eventlog](#).
- activitylog: Computes the resource frequency for an [activitylog](#).
- grouped\_activitylog: Computes the resource frequency for a [grouped\\_activitylog](#).

**References**

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

**See Also**

[resource\\_involvement](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

resource\_involvement    *Resource Involvement*

---

**Description**

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

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.

**Usage**

```
resource_involvement(
  log,
  level = c("case", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)

## S3 method for class 'log'
resource_involvement(
  log,
  level = c("case", "resource", "resource-activity"),
```

```

    append = deprecated(),
    append_column = NULL,
    sort = TRUE,
    eventlog = deprecated()
)

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

```

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "case"): Level of granularity for the analysis: "case" (default), "resource", or "resource-activity". For more information, see vignette("metrics", "eDeaR") and 'Details' below.
append	<b>logical</b> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	[ <b>Deprecated</b> ]; please use log instead.

### Details

Argument level has the following options:

- On "case" level, the absolute and relative number of distinct resources executing activities in each case is calculated, to get 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 "resource" level, this metric provides the absolute and relative number of cases in which each resource is involved, indicating which resources are more "necessary" within the process than the others.
- On "resource-activity" level, 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)**

- `log`: Computes the resource involvement for a `log`.
- `grouped_log`: Computes the resource involvement for a `grouped_log`.

**References**

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

**See Also**

[resource\\_frequency](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

resource\_specialisation

*Resource Specialisation*

---

**Description**

Analyses whether resources specialise in specific activities.

This metric can give 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.

**Usage**

```
resource_specialisation(
  log,
  level = c("log", "case", "activity", "resource"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)
```

```
resource_specialization(
  log,
  level = c("log", "case", "activity", "resource"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)
```

```

)

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

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

```

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", or "resource". For more information, see <code>vignette("metrics", "eデアR")</code> and 'Details' below.
append	<b>logical</b> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use log instead.

### Details

Argument level has the following options:

- At "log" level, this metric provides summary statistics on the number of distinct activities executed per resource.
- On "case" level, 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 "activity" level, this metric provides an overview of the absolute and relative number of different resources executing this activity within the complete log. This will give insights into which activities resources are specialised in.
- On "resource" level, this metric shows the absolute and relative number of distinct activities that each resource executes.

### Methods (by class)

- log: Computes the resource specialisation for a [log](#).
- grouped\_log: Computes the resource specialisation for a [grouped\\_log](#).

### References

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

### See Also

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

size\_of\_repetitions    *Metric: Size of repetitions*

---

### Description

Provides summary statistics on the sizes of repetitions.

### Usage

```
size_of_repetitions(
  log,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  eventlog = deprecated()
)

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



```

    append_column = NULL,
    eventlog = deprecated()
)

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

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

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

```

## Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
type	<b>character</b> (default "all"): The type of repetitions: "all" (default), "repeat", or "redo". For more information, see 'Details' below.
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see vignette("metrics", "eDeaR") and 'Details' below.
append	<b>logical</b> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
eventlog	[ <b>Deprecated</b> ]; please use log instead.

**Methods (by class)**

- eventlog: Size of repetitions for eventlog
- grouped\_eventlog: Size of repetitions for grouped event log
- activitylog: Size of repetitions for activitylog
- grouped\_activitylog: Size of repetitions for grouped activitylog

**References**

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

**See Also**

[number\\_of\\_repetitions](#)

---

size_of_selfloops	<i>Metric: Size of selfloops</i>
-------------------	----------------------------------

---

**Description**

Provides summary statistics on the sizes of selfloops

**Usage**

```
size_of_selfloops(
  log,
  type = c("all", "repeat", "redo"),
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  eventlog = deprecated()
)

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

## S3 method for class 'grouped_eventlog'
size_of_selfloops(
  log,
```

```

    type = c("repeat", "redo"),
    level = c("log", "case", "activity", "resource", "resource-activity"),
    append = deprecated(),
    append_column = NULL,
    eventlog = deprecated()
)

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

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

```

### Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
type	<a href="#">character</a> (default "all"): The type of repetitions: "all" (default), "repeat", or "redo". For more information, see 'Details' below.
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see <code>vignette("metrics", "eDeaR")</code> and 'Details' below.
append	<a href="#">logical</a> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
eventlog	[ <b>Deprecated</b> ]; please use <a href="#">log</a> instead.

### Methods (by class)

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

**References**

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

**See Also**

[number\\_of\\_selfloops](#)

---

start_activities	<i>Start Activities</i>
------------------	-------------------------

---

**Description**

Analyse the start activities in the process.

**Usage**

```
start_activities(
  log,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)

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

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

## S3 method for class 'activitylog'
```

```

start_activities(
  log,
  level = c("log", "case", "activity", "resource", "resource-activity"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)

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

```

## Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "case", "activity", "resource", or "resource-activity". For more information, see vignette("metrics", "edeaR") and 'Details' below.
append	<b>logical</b> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use log instead.

## Details

Argument level has the following options:

- On "log" level, this metric shows the absolute and relative number of activities that are the first activity in one or more of the cases.
- On "case" level, this metric provides an overview of the start activity of each case.
- On "activity" level, this metric calculates for each activity the absolute and relative number of cases that start with this activity type. Similar to the **end\_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 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 "resource" level, an overview of which resources execute the first activity per case are provided.
- On "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: Computes the start activities for an [eventlog](#).
- grouped\_eventlog: Computes the start activities for a [grouped\\_eventlog](#).
- activitylog: Computes the start activities for an [activitylog](#).
- grouped\_activitylog: Computes the start activities for a [grouped\\_activitylog](#).

### References

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

### See Also

[end\\_activities](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

throughput_time	<i>Throughput Time of Cases</i>
-----------------	---------------------------------

---

### Description

Provides summary statistics concerning the throughput times of cases.

### Usage

```
throughput_time(
  log,
  level = c("log", "trace", "case"),
  append = deprecated(),
  append_column = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  sort = TRUE,
  work_schedule = NULL,
  eventlog = deprecated()
)
```

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

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

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

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

**Arguments**

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "trace", or "case". For more information, see <code>vignette("metrics", "eDeaR")</code> and 'Details' below.
append	<a href="#">logical</a> (default FALSE) [ <b>Deprecated</b> ]: Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	[ <b>Deprecated</b> ] Which of the output columns to append to log, if <code>append = TRUE</code> . Default column depends on chosen level.
units	<a href="#">character</a> (default "auto"): The time unit in which the throughput times should be reported. Should be one of the following values: "auto" (default), "secs", "mins", "hours", "days", "weeks". See also the units argument of <a href="#">difftime</a> .
sort	<a href="#">logical</a> (default TRUE): Sort output on count. Only for levels with frequency count output.
work_schedule	A schedule of working hours. If provided, only working hours are counted as processing time.
eventlog	[ <b>Deprecated</b> ]; please use <code>log</code> instead.

**Details**

Argument `level` has the following options:

- At "log" level, the summary statistics describing the throughput time of cases in an aggregated fashion.
- On "trace" level, the throughput time of the different process variants or traces in the log are calculated.
- On "case" level, the throughput time is defined as 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.

For other levels (e.g. "activity", "resource", or "resource-activity"), the throughput time is equal to the [processing\\_time](#) and are, therefore, not supported by this method.

**Methods (by class)**

- `eventlog`: Computes throughput time for an [eventlog](#).
- `grouped_eventlog`: Computes throughput time for a [grouped\\_eventlog](#).
- `activitylog`: Computes throughput time for an [activitylog](#).
- `grouped_activitylog`: Computes throughput time for a [grouped\\_activitylog](#).

**References**

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



**See Also**

[idle\\_time](#), [processing\\_time](#), [difftime](#)

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [trace\\_coverage\(\)](#), [trace\\_length\(\)](#)

---

trace_coverage	<i>Trace Coverage</i>
----------------	-----------------------

---

**Description**

Analyses the structuredness of a log by use of trace frequencies.

**Usage**

```
trace_coverage(
  log,
  level = c("log", "trace", "case"),
  append = deprecated(),
  append_column = NULL,
  sort = TRUE,
  eventlog = deprecated()
)

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

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

## Arguments

log	<a href="#">log</a> : Object of class <a href="#">log</a> or derivatives ( <a href="#">grouped_log</a> , <a href="#">eventlog</a> , <a href="#">activitylog</a> , etc.).
level	<a href="#">character</a> (default "log"): Level of granularity for the analysis: "log" (default), "trace", or "case". For more information, see <a href="#">vignette("metrics", "edeaR")</a> and 'Details' below.
append	<a href="#">logical</a> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<a href="#">logical</a> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use <a href="#">log</a> instead.

## Details

Argument `level` has the following options:

- At "log" level, summary statistics of the coverage of traces are returned.
- On "trace" level, the absolute and relative frequency of each trace are returned.
- On "case" level, the coverage of the corresponding trace is returned for each case.

## Methods (by class)

- `log`: Calculates trace coverage metric for a [log](#).
- `grouped_log`: Calculates trace coverage metric for a [grouped\\_log](#).

## References

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

## See Also

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_length\(\)](#)

---

trace_length	<i>Trace Length</i>
--------------	---------------------

---

### Description

Analysis of trace lengths

This metric provides an overview of the number of activities that occur in each trace.

An important remark is that this metric takes into account each instance of an activity, but not the individual lifecycle events.

### Usage

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

```

    append_column = NULL,
    sort = TRUE,
    eventlog = deprecated()
)

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

```

### Arguments

log	<b>log</b> : Object of class <b>log</b> or derivatives ( <b>grouped_log</b> , <b>eventlog</b> , <b>activitylog</b> , etc.).
level	<b>character</b> (default "log"): Level of granularity for the analysis: "log" (default), "trace", or "case". For more information, see vignette("metrics", "edeaR") and 'Details' below.
append	<b>logical</b> (default FALSE) <b>[Deprecated]</b> : Indicating whether to append results to original log. Ignored when level is "log" or "trace".
append_column	<b>[Deprecated]</b> Which of the output columns to append to log, if append = TRUE. Default column depends on chosen level.
sort	<b>logical</b> (default TRUE): Sort output on count. Only for levels with frequency count output.
eventlog	<b>[Deprecated]</b> ; please use log instead.

### Details

Argument level has the following options:

- At "log" level, the summary statistics describing the trace length of cases in an aggregated fashion.
- On "trace" level, the trace length of the different process variants or traces in the log are calculated.
- On "case" level, the trace lengths for each case are computed.

### Methods (by class)

- eventlog: Computes trace length for an **eventlog**.
- grouped\_eventlog: Computes trace length for a **grouped\_eventlog**.
- activitylog: Computes trace length for an **activitylog**.
- grouped\_activitylog: Computes trace length for a **grouped\_activitylog**.

## **References**

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

## **See Also**

Other metrics: [activity\\_frequency\(\)](#), [activity\\_presence\(\)](#), [end\\_activities\(\)](#), [idle\\_time\(\)](#), [number\\_of\\_repetitions\(\)](#), [number\\_of\\_selfloops\(\)](#), [number\\_of\\_traces\(\)](#), [processing\\_time\(\)](#), [resource\\_frequency\(\)](#), [resource\\_involvement\(\)](#), [resource\\_specialisation\(\)](#), [start\\_activities\(\)](#), [throughput\\_time\(\)](#), [trace\\_coverage\(\)](#)

# Index

- \* **filters**
  - filter\_activity\_instance, 15
  - filter\_activity\_presence, 17
  - filter\_case\_condition, 19
  - filter\_endpoints\_condition, 20
  - filter\_infrequent\_flows, 22
  - filter\_precedence\_condition, 23
  - filter\_precedence\_resource, 25
  - filter\_resource\_frequency, 27
  - filter\_trace, 29
  - filter\_trim, 30
  - filter\_trim\_lifecycle, 32
- \* **internals**
  - filter\_activity\_instance, 15
  - filter\_activity\_presence, 17
  - filter\_resource\_frequency, 27
  - filter\_trim, 30
  - filter\_trim\_lifecycle, 32
- \* **metrics**
  - activity\_frequency, 3
  - activity\_presence, 5
  - end\_activities, 13
  - idle\_time, 34
  - number\_of\_repetitions, 36
  - number\_of\_selfloops, 39
  - number\_of\_traces, 42
  - processing\_time, 45
  - resource\_frequency, 50
  - resource\_involvement, 52
  - resource\_specialisation, 54
  - start\_activities, 60
  - throughput\_time, 62
  - trace\_coverage, 65
  - trace\_length, 67
- activity\_frequency, 3, 7, 15, 36, 39, 42, 43, 47, 52, 54, 56, 62, 65, 66, 69
- activity\_presence, 5, 5, 15, 36, 39, 42, 43, 47, 52, 54, 56, 62, 65, 66, 69
- activitylog, 4, 6, 11, 12, 14, 15, 17, 19, 21–24, 26, 28, 29, 31, 32, 35, 36, 38, 39, 41–43, 46–49, 51–53, 55, 57, 59, 61, 62, 64, 66, 68
- add\_fixed\_holiday, 7
- add\_floating\_holiday, 8
- add\_holiday\_periods, 8
- augment, 9
- calculate\_queuing\_length, 10
- calculate\_queuing\_times, 10, 11
- case\_list, 29, 30
- change\_day, 12
- character, 4, 10, 11, 14, 17, 24, 26, 29, 31, 33, 35, 38, 41, 46, 51, 53, 55, 57, 59, 61, 64, 66, 68
- create\_work\_schedule, 12
- difftime, 10–12, 35, 47, 64, 65
- edeaR, 13
- end\_activities, 5, 7, 13, 36, 39, 42, 43, 47, 52, 54, 56, 61, 62, 65, 66, 69
- eventlog, 4, 6, 11, 12, 14–17, 19, 21–24, 26, 28, 29, 31, 33, 35, 36, 38, 39, 41–43, 46–49, 51–53, 55, 57, 59, 61, 62, 64, 66, 68
- filter, 19, 20
- filter\_activity, 16, 18, 20, 22, 23, 25, 27, 29, 30, 32, 34
- filter\_activity\_frequency, 16, 18, 20, 22, 23, 25, 27, 29, 30, 32, 34
- filter\_activity\_instance, 15, 18, 20, 22, 23, 25, 27, 29, 30, 32, 34
- filter\_activity\_presence, 16, 17, 20, 22, 23, 25, 27, 29, 30, 32, 34
- filter\_case, 16, 18, 20, 22, 23, 25, 27, 29, 30, 32, 34
- filter\_case\_condition, 16, 18, 19, 22, 23, 25, 27, 29, 30, 32, 34

- filter\_endpoints, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_endpoints\_condition, [16](#), [18](#), [20](#), [20](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_endpoints\_conditions (filter\_endpoints\_condition), [20](#)
- filter\_infrequent\_flows, [16](#), [18](#), [20](#), [22](#), [22](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_lifecycle, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_lifecycle\_presence, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_precedence, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_precedence\_condition, [16](#), [18](#), [20](#), [22](#), [23](#), [23](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_precedence\_resource, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [25](#), [29](#), [30](#), [32](#), [34](#)
- filter\_processing\_time, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_resource, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_resource\_frequency, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [27](#), [30](#), [32](#), [34](#)
- filter\_throughput\_time, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_time\_period, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_trace, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [29](#), [32](#), [34](#)
- filter\_trace\_frequency, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_trace\_length, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [34](#)
- filter\_trim, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [30](#), [34](#)
- filter\_trim\_lifecycle, [16](#), [18](#), [20](#), [22](#), [23](#), [25](#), [27](#), [29](#), [30](#), [32](#), [32](#)
- grouped\_activitylog, [4](#), [6](#), [12](#), [15](#), [23](#), [32](#), [36](#), [39](#), [42](#), [47](#), [52](#), [62](#), [64](#), [68](#)
- grouped\_eventlog, [4](#), [6](#), [12](#), [15](#), [16](#), [23](#), [32](#), [33](#), [36](#), [39](#), [42](#), [47](#), [52](#), [62](#), [64](#), [68](#)
- grouped\_log, [4](#), [6](#), [11](#), [14](#), [16–26](#), [28](#), [29](#), [31](#), [33](#), [35](#), [38](#), [41](#), [43](#), [46](#), [48](#), [49](#), [51](#), [53–57](#), [59](#), [61](#), [64](#), [66](#), [68](#)
- idle\_time, [5](#), [7](#), [15](#), [34](#), [39](#), [42](#), [43](#), [47](#), [52](#), [54](#), [56](#), [62](#), [64–66](#), [69](#)
- ifilter\_activity\_instance (filter\_activity\_instance), [15](#)
- ifilter\_activity\_presence (filter\_activity\_presence), [17](#)
- ifilter\_resource\_frequency (filter\_resource\_frequency), [27](#)
- ifilter\_trim (filter\_trim), [30](#)
- ifilter\_trim\_lifecycle (filter\_trim\_lifecycle), [32](#)
- lifecycle\_id, [34](#)
- log, [4](#), [6](#), [11](#), [14](#), [16–21](#), [23–26](#), [28](#), [29](#), [31](#), [33](#), [35](#), [38](#), [41](#), [43](#), [46](#), [48](#), [49](#), [51](#), [53–57](#), [59](#), [61](#), [64](#), [66](#), [68](#)
- logical, [4](#), [6](#), [14](#), [16](#), [17](#), [19](#), [21](#), [24](#), [26](#), [28](#), [29](#), [31](#), [33](#), [35](#), [38](#), [41](#), [46](#), [47](#), [51](#), [53](#), [55](#), [57](#), [59](#), [61](#), [64](#), [66](#), [68](#)
- NA, [28](#)
- NULL, [31](#), [33](#)
- number\_of\_repetitions, [5](#), [7](#), [15](#), [36](#), [36](#), [42](#), [43](#), [47](#), [48](#), [52](#), [54](#), [56](#), [58](#), [62](#), [65](#), [66](#), [69](#)
- number\_of\_selfloops, [5](#), [7](#), [15](#), [36](#), [39](#), [39](#), [43](#), [47](#), [49](#), [52](#), [54](#), [56](#), [60](#), [62](#), [65](#), [66](#), [69](#)
- number\_of\_traces, [5](#), [7](#), [15](#), [36](#), [39](#), [42](#), [42](#), [47](#), [52](#), [54](#), [56](#), [62](#), [65](#), [66](#), [69](#)
- numeric, [10](#), [28](#)
- plot, [43](#)
- print.work\_schedule, [44](#)
- processing\_time, [5](#), [7](#), [15](#), [36](#), [39](#), [42](#), [43](#), [45](#), [52](#), [54](#), [56](#), [62](#), [64–66](#), [69](#)
- redo\_repetitions\_referral\_matrix, [48](#)
- redo\_selfloops\_referral\_matrix, [49](#)
- resource\_frequency, [5](#), [7](#), [15](#), [36](#), [39](#), [42](#), [43](#), [47](#), [50](#), [52](#), [54](#), [56](#), [62](#), [65](#), [66](#), [69](#)
- resource\_involvement, [5](#), [7](#), [15](#), [36](#), [39](#), [42](#), [43](#), [47](#), [52](#), [52](#), [56](#), [62](#), [65](#), [66](#), [69](#)
- resource\_specialisation, [5](#), [7](#), [15](#), [36](#), [39](#), [42](#), [43](#), [47](#), [52](#), [54](#), [54](#), [62](#), [65](#), [66](#), [69](#)
- resource\_specialization (resource\_specialisation), [54](#)
- self-loop, [39](#)
- seq.Date, [10](#)
- size\_of\_repetitions, [56](#)

size\_of\_selfloops, 58  
start\_activities, 5, 7, 14, 15, 36, 39, 42,  
43, 47, 52, 54, 56, 60, 65, 66, 69  
throughput\_time, 5, 7, 15, 36, 39, 42, 43, 45,  
47, 52, 54, 56, 62, 62, 66, 69  
trace\_coverage, 5, 7, 15, 36, 39, 42, 43, 47,  
52, 54, 56, 62, 65, 65, 69  
trace\_length, 5, 7, 15, 36, 39, 42, 43, 47, 52,  
54, 56, 62, 65, 66, 67  
traces, 43