

# Package ‘processmapR’

October 14, 2022

**Type** Package

**Title** Construct Process Maps Using Event Data

**Version** 0.5.2

**Date** 2022-10-03

**Description**

Visualize event logs using directed graphs, i.e. process maps. Part of the 'bupaR' framework.

**License** MIT + file LICENSE

**LinkingTo** Rcpp, BH

**SystemRequirements** C++11

**Depends** R (>= 3.5.0)

**Imports** dplyr, bupaR (>= 0.5.1), edeaR (>= 0.9.0), DiagrammeR (>= 1.0.0), ggplot2, stringr, purrr, data.table, shiny, miniUI, glue, forcats, hms, plotly, rlang (>= 1.0.0), cli (>= 3.2.0), scales, tidyr, htmltools, Rcpp, lifecycle

**Encoding** UTF-8

**RoxygenNote** 7.2.1

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

**VignetteBuilder** knitr

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

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

**Config/testthat/edition** 3

**Collate** 'RcppExports.R' 'create\_base\_precedence.R' 'custom.R' 'deprecated.R' 'dotted\_chart.R' 'dotted\_char\_plotly\_i.R' 'dotted\_chart\_helpers.R' 'export\_graph.R' 'export\_map.R' 'frequency.R' 'get\_meta\_data.R' 'layout.R' 'lined\_chart.R' 'lined\_chart\_helpers.R' 'lined\_chart\_plotly\_i.R' 'performance.R' 'precedence\_matrix.R' 'precedence\_matrix.plot.R' 'processMapOutput.R' 'process\_map.R'

'process\_matrix.R' 'processmapR.R' 'renderProcessMap.R'  
 'render\_map.R' 'resource\_map.R' 'resource\_matrix.R'  
 'trace\_explorer.R' 'utils.R'

**NeedsCompilation** yes

**Author** Gert Janssenswillen [aut, cre],  
 Gerard van Hulzen [ctb],  
 Benoît Depaire [ctb],  
 Felix Mannhardt [ctb],  
 Thijs Beuving [ctb],  
 urvikalia [ctb]

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

**Repository** CRAN

**Date/Publication** 2022-10-03 07:40:02 UTC

## R topics documented:

custom . . . . .	2
dotted_chart . . . . .	4
export_map . . . . .	7
frequency . . . . .	7
get_activities . . . . .	8
layout_pm . . . . .	8
lined_chart . . . . .	9
performance . . . . .	12
plot.process_matrix . . . . .	12
precedence_matrix_absolute . . . . .	13
processMapOutput . . . . .	13
processmapR . . . . .	14
process_map . . . . .	14
process_matrix . . . . .	17
renderProcessMap . . . . .	17
render_map . . . . .	18
resource_map . . . . .	19
resource_matrix . . . . .	20
trace_explorer . . . . .	21
<b>Index</b>	<b>24</b>

---

custom

*Custom map profile*

---

## Description

Function to create a custom map profile based on some event log attribute.

**Usage**

```

custom(
  FUN = mean,
  attribute,
  units = "",
  color_scale = "PuBu",
  color_edges = "dodgerblue4"
)

```

**Arguments**

<code>FUN</code>	A summary function to be called on the provided event attribute, e.g. <code>mean</code> , <code>median</code> , <code>min</code> , <code>max</code> . <code>na.rm = T</code> by default.
<code>attribute</code>	The name of the case attribute to visualize (should be numeric)
<code>units</code>	Character to be placed after values (e.g. EUR for monetary euro values)
<code>color_scale</code>	Name of color scale to be used for nodes. Defaults to PuBu. See <code>Rcolorbrewer::brewer.pal.info()</code> for all options.
<code>color_edges</code>	The color used for edges. Defaults to <code>dodgerblue4</code> .

**Details**

If used for edges, it will show the attribute values which related to the out-going node of the edge.#

**Examples**

```

## Not run:
library(eventdataR)
library(processmapR)
data(traffic_fines)
# make sure the amount attribute is propagated forward in each trace
# using zoo::na.locf instead of tidyr::fill since it is much faster
# still the whole pre-processing is still very slow
library(zoo)

traffic_fines_prepared <- traffic_fines %>%
  filter_trace_frequency(percentage = 0.8) %>%
  group_by_case() %>%
  mutate(amount = na.locf(amount, na.rm = F)) %>%
  ungroup_eventlog()

process_map(traffic_fines_prepared, type_nodes = custom(attribute = "amount", units = "EUR"))

## End(Not run)

```

---

dotted\_chart

*Dotted Chart*


---

### Description

A dotted chart is a graph in which each activity instance is displayed with a point (dot). The x-axis refers to the time aspect, while the y-axis refers to cases.

### Usage

```
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("auto", "start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  add_end_events = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  plotly = FALSE,
  eventlog = deprecated()
)
```

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

```
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("auto", "start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  add_end_events = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  plotly = FALSE,
  eventlog = deprecated()
)
```

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

```
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("auto", "start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  add_end_events = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  plotly = FALSE,
  eventlog = deprecated()
)
```

```
## S3 method for class 'grouped_eventlog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("auto", "start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  add_end_events = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  plotly = FALSE,
  eventlog = deprecated()
)
```

```
## S3 method for class 'grouped_activitylog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("auto", "start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  add_end_events = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  plotly = 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.).
x	<b>character</b> (default "absolute"): Value to plot on x-axis: "absolute" time or "relative" time (since start of week: "relative_week", since start of day: "relative_day").
sort	<b>character</b> (default "auto"): Ordering of the cases on y-axis: "auto" (default, see <b>Details</b> ), "start", "end", "duration", "start_week", or "start_day".
color	<b>character</b> (default <b>NULL</b> ): Attribute to use for coloring the activity instances (dots). This attribute should be present in log. Default ( <b>NULL</b> ) is the activity identifier ( <b>activity_id()</b> ). Use <b>NA</b> for no colors.
units	<b>character</b> (default "auto"): Time units to use on the x-axis in case of relative time: "auto" (default, see <b>Details</b> ), "secs", "mins", "hours", "days", or "weeks".
add_end_events	<b>logical</b> (default <b>FALSE</b> ): Whether to add dots for the complete lifecycle event with a different shape.
scale_color	<b>ggplot2</b> scale function (default <b>scale_color_discrete_bupaR</b> ): Set color scale. Defaults to <b>scale_color_discrete_bupaR</b> . Replaced with <b>scale_color_discrete</b> when more than 26 activities are present.

plotly            [logical](#) (default FALSE): Return a [plotly](#) object, instead of a [ggplot2](#).  
eventlog        **[Deprecated]**; please use log instead.

## Details

When setting sort to "auto", the ordering of cases is done automatically, based on the specified value of x:

- x = "absolute": sort = "start",
- x = "relative": sort = "duration",
- x = "relative\_week": sort = "start\_week",
- x = "relative\_day": sort = "start\_day".

When setting units to "auto", the time units on the x-axis is done automatically, based on the specified value of x:

- x = "absolute": units = "weeks",
- x = "relative": units = "weeks",
- x = "relative\_week": units = "secs",
- x = "relative\_day": units = "secs".

## Methods (by class)

- `dotted_chart(eventlog)`: Create dotted chart for an [eventlog](#).
- `dotted_chart(activitylog)`: Create dotted chart for an [activitylog](#).
- `dotted_chart(grouped_eventlog)`: Create dotted chart for a [grouped\\_eventlog](#).
- `dotted_chart(grouped_activitylog)`: Create dotted chart for a [grouped\\_activitylog](#).

## Examples

```
library(processmapR)
library(eventdataR)

patients %>%
  dotted_chart(x = "absolute", sort = "start", color = "employee")
```

---

export_map	<i>Export process map to pdf, png, ps or svg.</i>
------------	---

---

### Description

Export process map to pdf, png, ps or svg.

### Usage

```
export_map(
  map,
  file_name = NULL,
  file_type = NULL,
  title = NULL,
  width = NULL,
  height = NULL
)
```

### Arguments

map	A process_map created with <a href="#">process_map</a> and argument render = F.
file_name	The name of the exported file (including it's extension).
file_type	The type of file to be exported. Options for graph files are: png, pdf, svg, and ps.
title	An optional title for the output graph.
width	Output width in pixels or NULL for default. Only useful for export to image file formats png, pdf, svg, and ps.
height	Output height in pixels or NULL for default. Only useful for export to image file formats png, pdf, svg, and ps.

---

frequency	<i>Frequency map profile</i>
-----------	------------------------------

---

### Description

Function to create a frequency profile for a process map.

### Usage

```
frequency(
  value = c("absolute", "relative", "absolute-case", "relative-case",
    "relative-antecedent", "relative-consequent"),
  color_scale = "PuBu",
  color_edges = "dodgerblue4"
)
```

**Arguments**

value	The type of frequency value to be used: absolute, relative (percentage of activity instances) or relative_case (percentage of cases the activity occurs in).
color_scale	Name of color scale to be used for nodes. Defaults to PuBu. See <code>Rcolorbrewer::brewer.pal.info()</code> for all options.
color_edges	The color used for edges. Defaults to dodgerblue4.

---

get_activities	<i>Get data values for activities and flows from process map</i>
----------------	--

---

**Description**

Get data values for activities and flows from process map

**Usage**

```
get_activities(process_map)
```

```
get_flows(process_map)
```

**Arguments**

process_map	An object created using process_map function. Can both be a rendered or not rendered object.
-------------	--

---

layout_pm	<i>Configure layout parameters for process map</i>
-----------	--

---

**Description**

Configure layout parameters for process map

**Usage**

```
layout_pm(fixed_positions = NULL, edge_weight = FALSE, edge_cutoff = 0)
```



**Arguments**

fixed_positions	When specified as a data.frame with three columns 'act', 'x', and 'y' the position of nodes is fixed. Note that using this option switches to the 'neato' layout engine.
edge_weight	When TRUE then the frequency with which an edge appears in the process map has influence on the process map layout. Edges with higher frequency get higher priority in the layout algorithm, which increases the visibility of 'process highways'. Note that this has no effect when using the 'fixed_positions' parameters.
edge_cutoff	Edges that appear in the process map below this frequency are not considered at all when calculating the layout. This may create very long and complicated edge routings when chosen too high. Note that this has no effect when using the 'fixed_positions' parameters.

lined\_chart

*Lined Chart***Description**

A lined chart is a graph in which each activity instance is displayed with a line. The x-axis refers to the time aspect, while the y-axis refers to cases.

**Usage**

```
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = c("auto", "start", "end", "duration"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  line_width = 2,
  plotly = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  eventlog = deprecated()
)

## S3 method for class 'eventlog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = c("auto", "start", "end", "duration"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  line_width = 2,
  plotly = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
```

```

    eventlog = deprecated()
)

## S3 method for class 'activitylog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = c("auto", "start", "end", "duration"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  line_width = 2,
  plotly = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  eventlog = deprecated()
)

## S3 method for class 'grouped_eventlog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = c("auto", "start", "end", "duration"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  line_width = 2,
  plotly = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  eventlog = deprecated()
)

## S3 method for class 'grouped_activitylog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = c("auto", "start", "end", "duration"),
  color = NULL,
  units = c("auto", "secs", "mins", "hours", "days", "weeks"),
  line_width = 2,
  plotly = FALSE,
  scale_color = bupaR::scale_color_discrete_bupaR,
  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.).
x	<a href="#">character</a> (default "absolute"): Value to plot on x-axis: "absolute" time or "relative" time.

sort	<b>character</b> (default "auto"): Ordering of the cases on y-axis: "auto" (default, see <b>Details</b> ), "start", "end", or "duration".
color	<b>character</b> (default <b>NULL</b> ): Attribute to use for coloring the activity instances (dots). This attribute should be present in log. Default ( <b>NULL</b> ) is the activity identifier ( <code>activity_id()</code> ). Use <b>NA</b> for no colors.
units	<b>character</b> (default "auto"): Time units to use on the x-axis in case of relative time: "auto" (default, see <b>Details</b> ), "secs", "mins", "hours", "days", or "weeks".
line_width	<b>numeric</b> (default 2): The width of lines.
plotly	<b>logical</b> (default <b>FALSE</b> ): Return a <b>plotly</b> object, instead of a <b>ggplot2</b> .
scale_color	<b>ggplot2</b> scale function (default <code>scale_color_discrete_bupaR</code> ): Set color scale. Defaults to <code>scale_color_discrete_bupaR</code> . Replaced with <code>scale_color_discrete</code> when more than 26 activities are present.
eventlog	<b>[Deprecated]</b> ; please use <code>log</code> instead.

### Details

When setting sort to "auto", the ordering of cases is done automatically, based on the specified value of x:

- x = "absolute": sort = "start",
- x = "relative": sort = "duration".

When setting units to "auto", the time units on the x-axis is done automatically, based on the specified value of x:

- x = "absolute": units = "weeks",
- x = "relative": units = "weeks".

### Methods (by class)

- `lined_chart(eventlog)`: Create lined chart for an **eventlog**.
- `lined_chart(activitylog)`: Create lined chart for an **activitylog**.
- `lined_chart(grouped_eventlog)`: Create lined chart for a **grouped\_eventlog**.
- `lined_chart(grouped_activitylog)`: Create lined chart for a **grouped\_activitylog**.

### See Also

[dotted\\_chart\(\)](#)

### Examples

```
library(processmapR)
library(eventdataR)

patients %>%
  lined_chart(x = "absolute", color = "employee")
```

---

performance	<i>Performance map profile</i>
-------------	--------------------------------

---

### Description

Function to create a performance map profile to be used as the type of a process map. It results in a process map describing process time.

### Usage

```
performance(
  FUN = mean,
  units = c("mins", "secs", "hours", "days", "weeks", "months", "quarters", "semesters",
            "years"),
  flow_time = c("idle_time", "inter_start_time"),
  color_scale = "Reds",
  color_edges = "red4",
  ...
)
```

### Arguments

FUN	A summary function to be called on the process time of a specific activity, e.g. mean, median, min, max
units	The time unit in which processing time should be presented (mins, hours, days, weeks, months, quarters, semesters, years. A month is defined as 30 days. A quarter is 13 weeks. A semester is 26 weeks and a year is 365 days
flow_time	The time to depict on the flows: the inter start time is the time between the start timestamp of consecutive activity instances, the idle time is the time between the end and start time of consecutive activity instances.
color_scale	Name of color scale to be used for nodes. Defaults to Reds. See <code>Rcolorbrewer::brewer.pal.info()</code> for all options.
color_edges	The color used for edges. Defaults to red4.
...	Additional arguments too FUN

---

plot.process_matrix	<i>Process Matrix Plot</i>
---------------------	----------------------------

---

### Description

Visualize a precedence matrix. A generic plot function for precedences matrices.

**Usage**

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

**Arguments**

x	Precedence matrix
...	Additional paramters

**Value**

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

---

```
precedence_matrix_absolute
```

*Precedence Matrix*

---

**Description**

Construct a precedence matrix, showing how activities are followed by each other. This function computes the precedence matrix directly in C++ for efficiency. Only the type absolute of ([precedence\\_matrix](#)) is supported.

**Usage**

```
precedence_matrix_absolute(eventlog, lead = 1)
```

**Arguments**

eventlog	The event log object to be used.
lead	The distance between activities following/preceding each other.

---

```
processMapOutput
```

*Widget output function for use in Shiny*

---

**Description**

Widget output function for use in Shiny

**Usage**

```
processMapOutput(outputId, width = "100%", height = "400px")
```

**Arguments**

outputId	Output variable to read from.
width	A valid CSS unit for the width or a number, which will be coerced to a string and have px appended.
height	A valid CSS unit for the height or a number, which will be coerced to a string and have px appended.

---

processmapR	<i>processmapR - Process Maps in R</i>
-------------	--

---

**Description**

This package provides several useful techniques process visualization.

---

process_map	<i>Process Map</i>
-------------	--------------------

---

**Description**

A function for creating a process map of an event log.

**Usage**

```
process_map(
  log,
  type = frequency("absolute"),
  sec = NULL,
  type_nodes = type,
  type_edges = type,
  sec_nodes = sec,
  sec_edges = sec,
  rankdir = "LR",
  render = T,
  fixed_edge_width = F,
  layout = layout_pm(),
  fixed_node_pos = NULL,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'eventlog'
process_map(
  log,
  type = frequency("absolute"),
```

```
    sec = NULL,
    type_nodes = type,
    type_edges = type,
    sec_nodes = sec,
    sec_edges = sec,
    rankdir = "LR",
    render = T,
    fixed_edge_width = F,
    layout = layout_pm(),
    fixed_node_pos = NULL,
    eventlog = deprecated(),
    ...
)

## S3 method for class 'grouped_eventlog'
process_map(
  log,
  type = frequency("absolute"),
  sec = NULL,
  type_nodes = type,
  type_edges = type,
  sec_nodes = sec,
  sec_edges = sec,
  rankdir = "LR",
  render = T,
  fixed_edge_width = F,
  layout = layout_pm(),
  fixed_node_pos = NULL,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'activitylog'
process_map(
  log,
  type = frequency("absolute"),
  sec = NULL,
  type_nodes = type,
  type_edges = type,
  sec_nodes = sec,
  sec_edges = sec,
  rankdir = "LR",
  render = T,
  fixed_edge_width = F,
  layout = layout_pm(),
  fixed_node_pos = NULL,
  eventlog = deprecated(),
  ...
)
```

)

**Arguments**

log	log: Object of class <code>log</code> or derivatives ( <code>grouped_log</code> , <code>eventlog</code> , <code>activitylog</code> , etc.).
type	A process map type, which can be created with the functions <code>frequency</code> , <code>performance</code> and <code>custom</code> . The first type focusses on the frequency aspect of a process, while the second one focussed on processing time. The third one allows custom attributes to be used.
sec	A secondary process map type. Values are shown between brackets.
type_nodes	A process map type to be used for nodes only, which can be created with the functions <code>frequency</code> and <code>performance</code> . The first type focusses on the frequency aspect of a process, while the second one focussed on processing time.
type_edges	A process map type to be used for edges only, which can be created with the functions <code>frequency</code> and <code>performance</code> . The first type focusses on the frequency aspect of a process, while the second one focussed on processing time.
sec_nodes	A secondary process map type for nodes only.
sec_edges	A secondary process map type for edges only.
rankdir	The direction in which to layout the graph: "LR" (default), "TB", "BT", "RL", corresponding to directed graphs drawn from top to bottom, from left to right, from bottom to top, and from right to left, respectively.
render	Whether the map should be rendered immediately (default), or rather an object of type <code>dgr_graph</code> should be returned.
fixed_edge_width	If TRUE, don't vary the width of edges.
layout	List of parameters influencing the (automatic) layout of the process map. Use <code>layout_pm</code> to create a suitable parameter list.
fixed_node_pos	Deprecated, please use the 'layout' parameter instead.
eventlog	<b>[Deprecated]</b> ; please use <code>log</code> instead.
...	Deprecated arguments

**Methods (by class)**

- `process_map(eventlog)`: Process map for event log
- `process_map(grouped_eventlog)`: Process map for event log
- `process_map(activitylog)`: Process map for activitylog

**Examples**

```
## Not run:
library(eventdataR)
data(patients)
process_map(patients)

## End(Not run)
```



---

process_matrix	<i>Create process matrix</i>
----------------	------------------------------

---

**Description**

Create process matrix

**Usage**

```
process_matrix(log, type, ..., eventlog = deprecated())

## S3 method for class 'eventlog'
process_matrix(log, type = frequency(), ..., eventlog = deprecated())

## S3 method for class 'activitylog'
process_matrix(log, type = frequency(), ..., 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 process matrix type, which can be created with the functions <a href="#">frequency</a> , <a href="#">performance</a> and <a href="#">custom</a> . The first type focusses on the frequency aspect of a process, while the second one focussed on processing time. The third one allows custom attributes to be used.
...	Other arguments
eventlog	<b>[Deprecated]</b> ; please use <a href="#">log</a> instead.

**Methods (by class)**

- `process_matrix(eventlog)`: Process matrix for event log
- `process_matrix(activitylog)`: Process matrix for activity log

---

renderProcessMap	<i>Widget render function for use in Shiny</i>
------------------	--

---

**Description**

Widget render function for use in Shiny

**Usage**

```
renderProcessMap(expr, env = parent.frame(), quoted = FALSE)
```

**Arguments**

expr	an expression that generates a DiagrammeR graph.
env	the environment in which to evaluate expr.
quoted	is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

---

render\_map

*Render process map*


---

**Description**

Render process map

**Usage**

```
render_map(
  map,
  layout = NULL,
  output = NULL,
  as_svg = FALSE,
  title = NULL,
  width = NULL,
  height = NULL
)
```

**Arguments**

map	A process_map created with <a href="#">process_map</a> and argument render = F.
layout	A string specifying a layout type to use for node placement in this rendering. Possible layouts include: nicely, circle, tree, kk, and fr.
output	A string specifying the output type; graph (the default) renders the graph using the <a href="#">grViz()</a> function and visNetwork renders the graph using the <a href="#">visnetwork()</a> function.
as_svg	An option to render the graph as an SVG document.
title	An optional title for a graph when using output = "graph".
width	An optional parameter for specifying the width of the resulting graphic in pixels.
height	An optional parameter for specifying the height of the resulting graphic in pixels.

---

resource_map	<i>Resource Map</i>
--------------	---------------------

---

### Description

A function for creating a resource map of an event log based on handover of work.

### Usage

```
resource_map(log, type, render, ..., eventlog = deprecated())
```

```
## S3 method for class 'eventlog'
resource_map(
  log,
  type = frequency("absolute"),
  render = T,
  ...,
  eventlog = deprecated()
)
```

```
## S3 method for class 'activitylog'
resource_map(
  log,
  type = frequency("absolute"),
  render = T,
  ...,
  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	A process map type, which can be created with the functions <b>frequency</b> and <b>performance</b> . The first type focusses on the frequency aspect of a process, while the second one focussed on processing time.
render	Whether the map should be rendered immediately (default), or rather an object of type <b>dgr_graph</b> should be returned.
...	Deprecated arguments
eventlog	<b>[Deprecated]</b> ; please use <b>log</b> instead.

### Methods (by class)

- **resource\_map(eventlog)**: Create resource map for eventlog
- **resource\_map(activitylog)**: Create resource map for activity log

**Examples**

```
## Not run:
library(eventdataR)
data(patients)
resource_map(patients)

## End(Not run)
```

---

resource_matrix	<i>Resource Matrix</i>
-----------------	------------------------

---

**Description**

Construct a resource matrix, showing how work is handed over

**Usage**

```
resource_matrix(log, type, eventlog = deprecated())

## S3 method for class 'eventlog'
resource_matrix(
  log,
  type = c("absolute", "relative", "relative-antecedent", "relative-consequent"),
  eventlog = deprecated()
)

## S3 method for class 'activitylog'
resource_matrix(
  log,
  type = c("absolute", "relative", "relative-antecedent", "relative-consequent"),
  eventlog = deprecated()
)
```

**Arguments**

log	<b>log</b> : Object of class <code>log</code> or derivatives ( <code>grouped_log</code> , <code>eventlog</code> , <code>activitylog</code> , etc.).
type	The type of resource matrix, which can be <code>absolute</code> , <code>relative</code> , <code>relative_antecedent</code> or <code>relative_consequent</code> . <code>Absolute</code> will return a matrix with absolute frequencies, <code>relative</code> will return global relative frequencies for all antecedent-consequent pairs. <code>Relative_antecedent</code> will return relative frequencies within each antecedent, i.e. showing the relative proportion of consequents within each antecedent. <code>Relative_consequent</code> will do the reverse.
eventlog	<b>[Deprecated]</b> ; please use <code>log</code> instead.

**Methods (by class)**

- `resource_matrix(eventlog)`: Resource matrix of event log
- `resource_matrix(activitylog)`: Resource matrix of activity log

**Examples**

```
## Not run:
library(eventdataR)
data(patients)
precedence_matrix(patients)

## End(Not run)
```

---

trace_explorer	<i>Trace Explorer</i>
----------------	-----------------------

---

**Description**

Different activity sequences in the log can be visualized with `trace_explorer()`. With the `type` argument, it can be used to explore frequent as well as infrequent traces. The `coverage` argument specifies how much of the log you want to explore. By default it is set at 0.2, meaning that it will show the most (in)frequency traces covering 20% of the log.

**Usage**

```
trace_explorer(
  log,
  coverage = NULL,
  n_traces = NULL,
  type = c("frequent", "infrequent"),
  coverage_labels = c("relative", "absolute", "cumulative"),
  abbreviate = TRUE,
  show_labels = TRUE,
  label_size = 3,
  scale_fill = bupaR::scale_fill_discrete_bupaR,
  raw_data = FALSE,
  plotly = FALSE,
  eventlog = deprecated(),
  .abbreviate = deprecated()
)

## S3 method for class 'eventlog'
trace_explorer(
  log,
  coverage = NULL,
  n_traces = NULL,
```

```

type = c("frequent", "infrequent"),
coverage_labels = c("relative", "absolute", "cumulative"),
abbreviate = TRUE,
show_labels = TRUE,
label_size = 3,
scale_fill = bupaR::scale_fill_discrete_bupaR,
raw_data = FALSE,
plotly = FALSE,
eventlog = deprecated(),
.abbreviate = deprecated()
)

## S3 method for class 'activitylog'
trace_explorer(
  log,
  coverage = NULL,
  n_traces = NULL,
  type = c("frequent", "infrequent"),
  coverage_labels = c("relative", "absolute", "cumulative"),
  abbreviate = TRUE,
  show_labels = TRUE,
  label_size = 3,
  scale_fill = bupaR::scale_fill_discrete_bupaR,
  raw_data = FALSE,
  plotly = FALSE,
  eventlog = deprecated(),
  .abbreviate = deprecated()
)

```

### Arguments

<code>log</code>	<b>log</b> : Object of class <code>log</code> or derivatives ( <code>eventlog</code> or <code>activitylog</code> ).
<code>coverage</code>	<b>numeric</b> (default 0.2): The percentage coverage of the trace to explore. Defaults to 0.2 (0.05) most (in)frequent.
<code>n_traces</code>	<b>integer</b> : Instead of setting coverage, an exact number of traces can be set. Should be an <b>integer</b> larger than 0.
<code>type</code>	<b>character</b> (default "frequent"): "frequent" traces first, or "infrequent" traces first?
<code>coverage_labels</code>	<b>character</b> (default "relative"): Change the labels to be shown on the right of the process variants. These can be "relative" frequency (default), "absolute", or "cumulative". Multiple labels can be selected at the same time.
<code>abbreviate</code>	<b>logical</b> (default TRUE): If TRUE, abbreviate activity labels.
<code>show_labels</code>	<b>logical</b> (default TRUE): If FALSE, activity labels are not shown.
<code>label_size</code>	<b>numeric</b> (default 3): Font size of labels.
<code>scale_fill</code>	<b>ggplot2</b> scale function (default <code>scale_fill_discrete_bupaR</code> ): Set color scale. Defaults to <code>scale_fill_discrete_bupaR</code> . Replaced with <code>scale_fill_discrete</code> when more than 26 activities are present.

`raw_data`      **logical** (default FALSE): Return raw data instead of graph.  
`plotly`        **logical** (default FALSE): Return a **plotly** object, instead of a **ggplot2**.  
`eventlog`      **[Deprecated]**; please use `log` instead.  
`.abbreviate`   **[Deprecated]**; please use `abbreviate` instead.

**Methods (by class)**

- `trace_explorer(eventlog)`: Trace explorer for an **eventlog**.
- `trace_explorer(activitylog)`: Trace explorer for an **activitylog**.

**Examples**

```
library(processmapR)
library(eventdataR)

patients %>%
  trace_explorer(coverage = 0.8)
```

# Index

- \* **interval**
  - precedence\_matrix\_absolute, 13
- activity\_id(), 5, 11
- activitylog, 5, 6, 10, 11, 16, 17, 19, 20, 22, 23
- character, 5, 10, 11, 22
- custom, 2
- dotted\_chart, 4
- dotted\_chart(), 11
- eventlog, 5, 6, 10, 11, 16, 17, 19, 20, 22, 23
- export\_map, 7
- frequency, 7
- get\_activities, 8
- get\_flows (get\_activities), 8
- ggplot2, 5, 6, 11, 22, 23
- grouped\_activitylog, 6, 11
- grouped\_eventlog, 6, 11
- grouped\_log, 5, 10, 16, 17, 19, 20
- grViz(), 18
- integer, 22
- layout\_pm, 8, 16
- lined\_chart, 9
- log, 5, 10, 16, 17, 19, 20, 22
- logical, 5, 6, 11, 22, 23
- NA, 5, 11
- NULL, 5, 11
- numeric, 11, 22
- performance, 12
- plot.process\_matrix, 12
- plotly, 6, 11, 23
- precedence\_matrix, 13
- precedence\_matrix\_absolute, 13
- process\_map, 7, 14, 18
- process\_matrix, 17
- processMapOutput, 13
- processmapR, 14
- render\_map, 18
- renderProcessMap, 17
- resource\_map, 19
- resource\_matrix, 20
- scale\_color\_discrete, 5, 11
- scale\_color\_discrete\_bupaR, 5, 11
- scale\_fill\_discrete, 22
- scale\_fill\_discrete\_bupaR, 22
- trace\_explorer, 21
- trace\_explorer(), 21
- visnetwork(), 18