Package ‘processmapR’

July 15, 2024

Type   Package
Title  Construct Process Maps Using Event Data
Version 0.5.4
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++
Depends  R (>= 3.5.0)
Imports  dplyr, bupaR (>= 0.5.1), edcaR (>= 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, htmlwidgets
Encoding  UTF-8
RoxygenNote  7.2.3
Suggests  knitr, rmarkdown, eventdataR, testthat (>= 3.0.0), rsvg, DiagrammeRsvg, covr
VignetteBuilder  knitr
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_chart_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'
custom

Custom map profile

Description

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

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

**Arguments**

- **FUN**
  A summary function to be called on the provided event attribute, e.g. mean, median, min, max. `na.rm = T` by default.

- **attribute**
  The name of the case attribute to visualize (should be numeric)

- **units**
  Character to be placed after values (e.g. EUR for monitary euro values)

- **color_scale**
  Name of color scale to be used for nodes. Defaults to PuBu. See `Rcolorbrewer::brewer_pal.info()` for all options.

- **color_edges**
  The color used for edges. Defaults to dodgerblue4.

**Details**

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

**Examples**

```r
## 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**

```r
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()
)
```

```r
## 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()
)
```

```r
## 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()
)
```
## Arguments

**log**

Object of class `log` or derivatives (grouped_log, eventlog, activitylog, etc.).

**x**

character (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**

character (default "auto"): Ordering of the cases on y-axis: "auto" (default, see Details), "start", "end", "duration", "start_week", or "start_day".

**color**

character (default NULL): Attribute to use for coloring the activity instances (dots). This attribute should be present in log. Default (NULL) is the activity identifier (activity_id()). Use NA for no colors.

**units**

character (default "auto"): Time units to use on the x-axis in case of relative time: "auto" (default, see Details), "secs", "mins", "hours", "days", or "weeks".

**add_end_events**

logical (default FALSE): Whether to add dots for the complete lifecycle event with a different shape.

**scale_color**

ggplot2 scale function (default scale_color_discrete_bupaR): Set color scale. Defaults to scale_color_discrete_bupaR. Replaced with scale_color_discrete when more than 26 activities are present.
dotted_chart

plotly logical (default FALSE): Return a plotly object, instead of a ggplot2.

eventlog [Deprecation]: 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**

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

**Description**

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

**Usage**

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

**Arguments**

- `map`: A `process_map` created with `process_map` 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**

Frequency map profile

**Description**

Function to create a frequency profile for a process map.

**Usage**

```r
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 Rcolorbrewer::brewer_pal.info() for all options.

color_edges The color used for edges. Defaults to dodgerblue4.

get_activities Get data values for activities and flows from process map

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 Configure layout parameters for process map

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 (numeric) Number between 0 and 1. Edges with a relative frequency below the cut off are not considered at all when calculating the layout. This may create very long and complicated edge routings when choosen too high. Note that this has no effect when using the `fixed_positions` parameters.

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,
lined_chart

## 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: Object of class log or derivatives (grouped_log, eventlog, activitylog, etc.).

x: character (default "absolute"): Value to plot on x-axis: "absolute" time or "relative" time.
lined_chart

sort character (default "auto"): Ordering of the cases on y-axis: "auto" (default, see Details), "start", "end", or "duration".

color character (default NULL): Attribute to use for coloring the activity instances (dots). This attribute should be present in log. Default (NULL) is the activity identifier (activity_id()). Use NA for no colors.

units character (default "auto"): Time units to use on the x-axis in case of relative time: "auto" (default, see Details), "secs", "mins", "hours", "days", or "weeks".

line_width numeric (default 2): The width of lines.

plotly logical (default FALSE): Return a plotly object, instead of a ggplot2.

scale_color ggplot2 scale function (default scale_color_discrete_bupaR): Set color scale. Defaults to scale_color_discrete_bupaR. Replaced with scale_color_discrete when more than 26 activities are present.

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".

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 map profile

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 Rcolorbrewer::brewer_pal.info() for all options.
- **color_edges**: The color used for edges. Defaults to red4.
- **...**: Additional arguments too FUN

Process Matrix Plot

Description

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

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

**Arguments**

- `x`: Precedence matrix
- `...`: Additional parameters

**Value**

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

---

**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**

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

**Arguments**

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

---

**Description**

Widget output function for use in Shiny

**Usage**

```r
processMapOutput(outputId, width = "100\%", heigth = "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.

**Description**

This package provides several useful techniques for process visualization.

**Process Map**

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

**Usage**

```r
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(),
  eventlog = deprecated(),
  ...
)
```

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

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

Arguments

- **log**: Object of class `log` or derivatives (`grouped_log`, `eventlog`, `activitylog`, ...).
process_map etc.

type A process map type, which can be created with the functions frequency, performance and custom. 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 frequency and performance. 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 frequency and performance. 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 dgr_graph 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 layout_pm to create a suitable parameter list.

eventlog [Deprecated]: please use log instead.

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

Create process matrix

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 log: Object of class log or derivatives (grouped_log, eventlog, activitylog, etc.).
type A process matrix type, which can be created with the functions frequency, performance and custom. 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 [Deprecated]: please use log instead.

Methods (by class)
• process_matrix(eventlog): Process matrix for event log
• process_matrix(activitylog): Process matrix for activity log

renderProcessMap Widget render function for use in Shiny

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 process_map 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 grViz() function and visNetwork renders the graph using the visnetwork() 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.
Description

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

Usage

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

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

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

Arguments

- `log` Object of class `log` or derivatives (`grouped_log`, `eventlog`, `activitylog`, etc.).
- `type` A process map type, which can be created with the functions `frequency` and `performance`. 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 `dgr_graph` should be returned.
- `...` Deprecated arguments
- `eventlog` [Deprecated]: please use `log` instead.

Methods (by class)

- `resource_map(eventlog)`: Create resource map for eventlog
- `resource_map(activitylog)`: Create resource map for activity log
Examples

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

## End(Not run)
```

---

### resource_matrix

**Description**

Construct a resource matrix, showing how work is handed over

**Usage**

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

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

Examples

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

## End(Not run)
```

**trace_explorer**  
*Trace Explorer*

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)frequent traces covering 20% of the log.

Usage

```r
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,
)
trace_explorer

```r
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

- **log** (`log`): Object of class `log` or derivatives (eventlog or activitylog).
- **coverage** (`numeric` (default 0.2)): The percentage coverage of the trace to explore. Defaults to 0.2 (0.05) most (in)frequent.
- **n_traces** (`integer`): Instead of setting coverage, an exact number of traces can be set. Should be an integer larger than 0.
- **type** (`character` (default "frequent")): "frequent" traces first, or "infrequent" traces first?
- **coverage_labels** (`character` (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.
- **abbreviate** (`logical` (default TRUE)): If TRUE, abbreviate activity labels.
- **show_labels** (`logical` (default TRUE)): If FALSE, activity labels are not shown.
- **label_size** (`numeric` (default 3)): Font size of labels.
- **scale_fill** (`ggplot2` scale function (default `scale_fill_discrete_bupaR`): Set color scale. Defaults to `scale_fill_discrete_bupaR`. Replaced with `scale_fill_discrete` when more than 26 activities are present.
trace_explorer

- `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**

```r
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, 15, 17, 19, 20, 22, 23
character, 5, 10, 11, 22
custom, 2
dotted_chart, 4
dotted_chart(), 11
eventlog, 5, 6, 10, 11, 15, 17, 19, 20, 22, 23
export_map, 7
frequency, 7
get_activities, 8
get_flows (get_activities), 8
grouped_activitylog, 6, 11
grouped_eventlog, 6, 11
grouped_log, 5, 10, 15, 17, 19, 20
grViz(), 18
integer, 22
layout_pm, 8, 16
lined_chart, 9
log, 5, 10, 15, 17, 19, 20, 22
logical, 5, 10, 11, 22, 23
NA, 5, 11
NULL, 5, 11
numeric, 9, 11, 22
performance, 12
plot.process_matrix, 12
precedence_matrix, 13
precedence_matrix_absolute, 13
process_matrix, 7, 14, 18
process_map, 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