Package 'processmapR'

July 13, 2022

Type Package

Title Construct Process Maps Using Event Data

Version 0.5.1

Date 2022-07-13

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 (>= 0.4.0), edeaR (>= 0.8.0), DiagrammeR (>= 1.0.0), ggplot2, stringr, purrr, data.table, shiny, miniUI, glue, forcats, hms, plotly, rlang, scales, tidyr, htmltools, Rcpp, lifecycle

Encoding UTF-8

RoxygenNote 7.2.0

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

VignetteBuilder knitr

URL https://bupar.net/, https://github.com/bupaverse/processmapr/

BugReports https://github.com/bupaverse/processmapr/issues/

Config/testthat/edition 3

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 2022-07-13 14:50:02 UTC

R topics documented:

`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

`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

```
## 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)
```

`frequency` *Frequency map profile*

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 '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.

idotted_chart *Dotted chart*

Description

Create a dotted chart to view all events in a glance

Usage

```
idotted_chart(eventlog, plotly = FALSE)

iplotly_dotted_chart(eventlog)

plotly_dotted_chart(
  eventlog,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = c("start", "end", "duration", "start_week", "start_day"),
  color = NULL,
  units = c("weeks", "days", "hours", "mins", "secs"),
  ...
)

dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = NULL,
  color = NULL,
```

```
  units = NULL,
  add_end_events = F,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'eventlog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = NULL,
  color = NULL,
  units = NULL,
  add_end_events = F,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'activitylog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = NULL,
  color = NULL,
  units = NULL,
  add_end_events = F,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'grouped_eventlog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = NULL,
  color = NULL,
  units = NULL,
  add_end_events = F,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'grouped_activitylog'
dotted_chart(
  log,
  x = c("absolute", "relative", "relative_week", "relative_day"),
  sort = NULL,
  color = NULL,
```

```
  units = NULL,
  add_end_events = F,
  eventlog = deprecated(),
  ...
)
```

Arguments

eventlog	Deprecated. Use log instead.
plotly	Return plotly object
x	Value for plot on x-axis: absolute time or relative time (since start, since start of week, since start of day)
sort	Ordering of the cases on y-axis: start, end or duration, start_week, start_day
color	Optional, variable to use for coloring dots. Default is the activity identifier. Use NA for no colors.
units	Time units to use on x-axis in case of relative time.
...	Deprecated arguments
log	log: Object of class log or derivatives (grouped_log, eventlog, activitylog, etc.).
add_end_events	Whether to add dots for the complete lifecycle event with a different shape.

Methods (by class)

- eventlog: Create dotted chart for eventlog
- activitylog: Create dotted chart for activitylog
- grouped_eventlog: Create dotted chart for grouped eventlog
- grouped_activitylog: Create dotted chart for grouped activitylog

---

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	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 choosen too high. Note that this has no effect when using the 'fixed_positions' parameters.

---

lined_chart *Lined chart*

Description

Create a lined chart to view all cases at a glance

Usage

```
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = NULL,
  color,
  units,
  line_width = 2,
  plotly,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'eventlog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = NULL,
  color = NULL,
  units = c("weeks", "days", "hours", "mins", "secs"),
  line_width = 2,
  plotly = FALSE,
  eventlog = deprecated(),
```

```
  ...
)

## S3 method for class 'activitylog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = NULL,
  color = NULL,
  units = c("weeks", "days", "hours", "mins", "secs"),
  line_width = 2,
  plotly = FALSE,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'grouped_eventlog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = NULL,
  color = NULL,
  units = c("weeks", "days", "hours", "mins", "secs"),
  line_width = 2,
  plotly = FALSE,
  eventlog = deprecated(),
  ...
)

## S3 method for class 'grouped_activitylog'
lined_chart(
  log,
  x = c("absolute", "relative"),
  sort = NULL,
  color = NULL,
  units = c("weeks", "days", "hours", "mins", "secs"),
  line_width = 2,
  plotly = FALSE,
  eventlog = deprecated(),
  ...
)

ilined_chart(eventlog, plotly = FALSE)

iplotly_lined_chart(eventlog)

plotly_lined_chart(eventlog, color = NULL, ...)
```

Arguments

log	log: Object of class log or derivatives (grouped_log, eventlog, activitylog, etc.).
x	Value for plot on x-axis: absolute time or relative time (since start, since start of week, since start of day)
sort	Ordering of the cases on y-axis: start, end or duration, start_week, start_day
color	Optional. Should be a case attribute! No coloring applied by default.
units	Time units to use on x-axis in case of relative time.
line_width	The width of lines
plotly	Return plotly object
eventlog	Eventlog object
...	Deprecated arguments

Methods (by class)

- eventlog: Create lined chart for eventlog
- activitylog: Create lined chart for activity log
- grouped_eventlog: Create lined chart for grouped eventlog
- grouped_activitylog: Create lined chart for grouped activitylog

---

performance *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

---

plot.process_matrix *Process Matrix Plot*

Description

Visualize a precendence 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%", 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.
`heigth`	A valid CSS unit for the height or a number, which will be coerced to a string and have px appended.

`processmapR` *processmapR - Process Maps in R*

Description

This package provides several useful techniques process visualization.

process_map *Process Map*

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 log or derivatives (grouped_log, eventlog, activitylog, 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.
fixed_node_pos	Deprecated, please use the 'layout' parameter instead.
eventlog	Deprecated. Use log instead.
...	Deprecated arguments

Methods (by class)

- eventlog: Process map for event log
- grouped_eventlog: Process map for event log
- 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. Use log instead.

Methods (by class)

- eventlog: Process matrix for event log
- 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.

resource_map *Resource Map*

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	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. Use log instead.

Methods (by class)

- eventlog: Create resource map for eventlog
- activitylog: Create resource map for activity log

Examples

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

## End(Not run)
```

`resource_matrix` *Resource Matrix*

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`	`log`: Object of class `log` or derivatives (`grouped_log`, `eventlog`, `activitylog`, etc.).
`type`	The type of resource matrix, which can be absolulte, 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 antecendent, i.e. showing the relative proportion of consequents within each antecedent. Relative_consequent will do the reverse.
`eventlog`	Deprecated. Use log instead.

Methods (by class)

- `eventlog`: Resource matrix of event log
- `activitylog`: Resource matrix of activity log

Examples

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

## End(Not run)
```

`trace_explorer` *Trace explorer*

Description

Explore traces, ordered by relative trace frequency

Usage

```
trace_explorer(
  log,
  coverage = NULL,
  n_traces = NULL,
  type = c("frequent", "infrequent"),
  coverage_labels = c("relative", "absolute", "cumulative"),
  .abbreviate = T,
  show_labels = T,
  label_size = 3,
  scale_fill = processmapR:::scale_fill_discrete_bupaR(),
  raw_data = F,
  eventlog = 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 = T,
  show_labels = T,
  label_size = 3,
```

```
  scale_fill = processmapR:::scale_fill_discrete_bupaR(),
  raw_data = F,
  eventlog = 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 = T,
  show_labels = T,
  label_size = 3,
  scale_fill = processmapR:::scale_fill_discrete_bupaR(),
  raw_data = F,
  eventlog = deprecated()
)

plotly_trace_explorer(
  eventlog,
  coverage = NULL,
  n_traces = NULL,
  type = c("frequent", "infrequent"),
  coverage_labels = c("relative", "absolute", "cumulative"),
  .abbreviate = T,
  show_labels = T,
  label_size = 3,
  scale_fill = processmapR:::scale_fill_discrete_bupaR(),
  raw_data = F
)
```

Arguments

log	log: Object of class log or derivatives (grouped_log, eventlog, activitylog, etc.).
coverage	The percentage coverage of the trace to explore. Default is 20% most (in)frequent
n_traces	Instead of setting coverage, you can set an exact number of traces. Should be an integer larger than 0.
type	Frequent traces first, or infrequent traces first?
coverage_labels	Change the labels to be shown on the right of the process variants. These can be relative frequency (default), absolute, or cumulative.
.abbreviate	If TRUE, abbreviate activity labels
show_labels	If False, activity labels are not shown.
label_size	Font size of labels

`scale_fill`	Set color scale
`raw_data`	Return raw data
`eventlog`	Deprecated. Use log instead.

Methods (by class)

- `eventlog`: Trace explorer eventlog
- `activitylog`: Trace explorer activity log

Index