Package ‘processanimateR’

July 20, 2022

Type Package

Title Process Map Token Replay Animation

Version 1.0.5

Date 2022-07-14

Description Provides animated process maps based on the ‘procesmapR’ package. Cases stored in event logs created with ‘bupaR’ S3 class eventlog() are rendered as tokens (SVG shapes) and animated according to their occurrence times on top of the process map. For rendering SVG animations (‘SMIL’) and the ‘htmlwidget’ package are used.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports bupaR, procesmapR (>= 0.3.1), rlang, magrittr, dplyr, tidyr, htmlwidgets, DiagrammeR (>= 1.0.0), grDevices, stringr, htmltools

Suggests eventdataR, edeaR, testthat, knitr, rmarkdown, shiny, RColorBrewer, lubridate

RoxygenNote 7.2.1

URL https://github.com/bupaverse/processanimateR/

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

VignetteBuilder knitr

NeedsCompilation no

Author Felix Mannhardt [aut, cre], Gert Janssenswillen [ctb]

Maintainer Felix Mannhardt <f.mannhardt@tue.nl>

Repository CRAN

Date/Publication 2022-07-20 12:40:03 UTC
activity_select_decoration

Description

Decoration callback for activity selection

Usage

```r
activity_select_decoration(
  stroke_dasharray = "2",
  stroke_width = "2",
  stroke = "black"
)
```

Arguments

- **stroke_dasharray**: Sets the 'stroke-dasharray' attribute for selected activities.
- **stroke_width**: Sets the 'stroke-width' attribute for selected activities.
- **stroke**: Sets the 'stroke' attribute for selected activities.

Value

A JavaScript callback function called when activity selection changes.

See Also

animate_process
Examples

# Create a decoration callback that increases the activity stroke width
activity_select_decoration(stroke_width = "5")

animate_process
Animate cases on a process map

Description

This function animates the cases stored in a ‘bupaR‘ event log on top of a process model. Each case is represented by a token that travels through the process model according to the waiting and processing times of activities. Currently, animation is only supported for process models created by `process_map` of the ‘processmapR’ package. The animation will be rendered as SVG animation (SMIL) using the ‘htmlwidgets‘ framework. Each token is a SVG shape and customizable.

Usage

```r
animate_process(
  eventlog,
  processmap = process_map(eventlog, render = F, ...),
  renderer = renderer_graphviz(),
  mode = c("absolute", "relative", "off"),
  duration = 60,
  jitter = 0,
  timeline = TRUE,
  legend = NULL,
  initial_state = c("playing", "paused"),
  initial_time = 0,
  repeat_count = 1,
  repeat_delay = 0.5,
  epsilon_time = duration/1000,
  mapping = token_aes(),
  token_callback_onclick = c("function(svg_root, svg_element, case_id) {", "}"),
  token_callback_select = token_select_decoration(),
  activity_callback_onclick = c("function(svg_root, svg_element, activity_id) {", "}"),
  activity_callback_select = activity_select_decoration(),
  elementId = NULL,
  preRenderHook = NULL,
  width = NULL,
  height = NULL,
  sizingPolicy = htmlwidgets::sizingPolicy(browser.fill = TRUE, viewer.fill = TRUE,
    knitr.figure = FALSE, knitr.defaultWidth = "100%", knitr.defaultHeight = "300"),
  ...
)
```
Arguments

- **eventlog**: The ‘bupaR’ event log object that should be animated.
- **processmap**: A process map created with ‘processmapR’ (process_map) on which the event log will be animated. If not provided a standard process map will be generated from the supplied event log.
- **renderer**: Whether to use Graphviz (renderer_graphviz) to layout and render the process map, or to render the process map using Leaflet (renderer_leaflet) on a geographical map.
- **mode**: Whether to animate the cases according to their actual time of occurrence (‘absolute’) or to start all cases at once (‘relative’).
- **duration**: The overall duration of the animation, all times are scaled according to this overall duration.
- **jitter**: The magnitude of a random coordinate translation, known as jitter in scatterplots, which is added to each token. Adding jitter can help to disambiguate tokens drawn on top of each other.
- **timeline**: Whether to render a timeline slider in supported browsers (Work only on recent versions of Chrome and Firefox).
- **legend**: Whether to show a legend for the ‘size’ or the ‘color’ scale. The default is not to show a legend.
- **initial_state**: Whether the initial playback state is ‘playing’ or ‘paused’. The default is ‘playing’.
- **initial_time**: Sets the initial time of the animation. The default value is ‘0’.
- **repeat_count**: The number of times the process animation is repeated.
- **repeat_delay**: The seconds to wait before one repetition of the animation.
- **epsilon_time**: A (small) time to be added to every animation to ensure that tokens are visible.
- **mapping**: A list of aesthetic mappings from event log attributes to certain visual parameters of the tokens. Use token_aes to create a suitable mapping list.
- **token_callback_onclick**: A JavaScript function that is called when a token is clicked. The function is parsed by JS and received three parameters: ‘svg_root’, ‘svg_element’, and ‘case_id’.
- **token_callback_select**: A JavaScript callback function called when token selection changes.
- **activity_callback_onclick**: A JavaScript function that is called when an activity is clicked. The function is parsed by JS and received three parameters: ‘svg_root’, ‘svg_element’, and ‘activity_id’.
- **activity_callback_select**: A JavaScript callback function called when activity selection changes.
- **elementId**: passed through to createWidget. A custom elementId is useful to capture the selection events via input$elementId_tokens and input$elementId_activities when used in Shiny.
- **preRenderHook**: passed through to createWidget.
width, height  Fixed size for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.

sizingPolicy  Options that govern how the widget is sized in various containers (e.g. a standalone browser, the RStudio Viewer, a knitr figure, or a Shiny output binding). These options can be specified by calling the `sizingPolicy` function.

... Options passed on to `process_map`.

See Also

`process_map`, `token_aes`

Examples

data(example_log)

# Animate the process with default options (absolute time and 60s duration)
animate_process(example_log)

# Animate the process with default options (relative time, with jitter, infinite repeat)
animate_process(example_log, mode = "relative", jitter = 10, repeat_count = Inf)

attribution_osm

<table>
<thead>
<tr>
<th>Standard attribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribution_osm</td>
</tr>
</tbody>
</table>

Description

This is the standard attribution advised for OpenStreetMap tiles.

Usage

`attribution_osm()`

Value

The attribution character vector.

Examples

`attribution_osm()`
example_log  

Example event log used in documentation

**Description**

Example event log used in documentation

**Usage**

```python
example_log
```

**Format**

An bupaR event log

icon_circle  

Standard circle marker

**Description**

The marker is based on Material Design (Apache 2.0 License): https://material.io/

**Usage**

```javascript
icon_circle()
```

**Value**

SVG code for a map marker.

**Examples**

```javascript
icon_circle()
```
**icon_marker**  

*Standard map marker*

**Description**

The marker is based on Material Design (Apache 2.0 License): https://material.io/

**Usage**

```javascript
icon_marker()
```

**Value**

SVG code for a map marker.

**Examples**

```javascript
icon_marker()
```

---

**processanimaterOutput**  

*Create a process animation output element*

**Description**

Renders a renderProcessanimater within an application page.

**Usage**

```javascript
processanimaterOutput(outputId, width = "100\%", height = "400px")
```

**Arguments**

- `outputId`  
  Output variable to read the animation from

- `width, height`  
  Must be a valid CSS unit (like 100 which will be coerced to a string and have px appended.
renderer_graphviz  

Render as a plain graph

Description

This renderer uses viz.js to render the process map using the DOT layout.

Usage

renderer_graphviz(
  svg_fit = TRUE,
  svg_contain = FALSE,
  svg_resize_fit = TRUE,
  zoom_controls = TRUE,
  zoom_initial = NULL
)

Arguments

svg_fit  Whether to scale the process map to fully fit in its container. If set to ‘TRUE’ the process map will be scaled to be fully visible and may appear very small.

svg_contain  Whether to scale the process map to use all available space (contain) from its container. If set to ‘FALSE’, if ‘svg_fit’ is set this takes precedence.

svg_resize_fit  Whether to (re)-fit the process map to its container upon resize.

zoom_controls  Whether to show zoom controls.

zoom_initial  The initial zoom level to use.

Value

A rendering function to be used with animate_process

See Also

animate_process

Examples

data(example_log)

# Animate the process with the default GraphViz DOT renderer
animate_process(example_log, renderer = renderer_graphviz())
renderer_leaflet  Render as graph on a geographical map

Description

This renderer uses Leaflet to draw the nodes and edges of the process map on a geographical map.

Usage

```r
renderer_leaflet(
  node_coordinates,
  edge_coordinates = data.frame(act_from = character(0), act_to = character(0), lat = numeric(0), lng = numeric(0), stringsAsFactors = FALSE),
  layer = c(paste0("new L.TileLayer('", tile, ",',"), paste0("{ attribution : '",
                                 attribution_osm(), "'},")),
  tile = "http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
  options = list(),
  grayscale = TRUE,
  icon_act = icon_marker(),
  icon_start = icon_circle(),
  icon_end = icon_circle(),
  scale_max = 4,
  scale_min = 0.25
)
```

Arguments

- **node_coordinates**
  A data frame with node coordinates in the format ‘act’, ‘lat’, ‘lng’.
- **edge_coordinates**
  A data frame with additional edge coordinates in the format ‘act_from’, ‘act_to’, ‘lat’, ‘lng’.
- **layer**
  The JavaScript code used to create a Leaflet layer. A TileLayer is used as default value.
- **tile**
  The URL to be used for the standard Leaflet TileLayer.
- **options**
  A named list of leaflet options, such as the center point of the map and the initial zoom level.
- **grayscale**
  Whether to apply a grayscale filter to the map.
- **icon_act**
  The SVG code used for the activity icon.
- **icon_start**
  The SVG code used for the start icon.
- **icon_end**
  The SVG code used for the end icon.
- **scale_max**
  The maximum factor to be used to scale the process map with when zooming out.
- **scale_min**
  The minimum factor to be used to scale the process map with when zooming in.
Value

A rendering function to be used with `animate_process`.

See Also

`animate_process`.

Examples

data(example_log)

# Animate the example process with activities placed in some locations
animate_process(example_log,
    renderer = renderer_leaflet(
        node_coordinates = data.frame(
            act = c("A", "B", "C", "D", "ARTIFICIAL_START", "ARTIFICIAL_END"),
            lat = c(63.443680, 63.426925, 63.409207, 63.422336, 63.450950, 63.419706),
            lng = c(10.383625, 10.396972, 10.406418, 10.432119, 10.383368, 10.252347),
            stringsAsFactors = FALSE),
        edge_coordinates = data.frame(
            act_from = c("B"),
            act_to = c("C"),
            lat = c(63.419207),
            lng = c(10.386418),
            stringsAsFactors = FALSE),
        options = list(center = c(63.412273, 10.399590), zoom = 12),
        duration = 5, repeat_count = Inf))


renderProcessanimater  Renders process animation output

Description

Renders a SVG process animation suitable to be used by `processanimaterOutput`.

Usage

renderProcessanimater(expr, env = parent.frame(), quoted = FALSE)

Arguments

expr  The expression generating a process animation (animate_process).
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.
**token_aes**  
*Tokens aesthetics mapping*

**Description**

Tokens aesthetics mapping

**Usage**

```
token_aes(
    size = token_scale(),
    color = token_scale(),
    image = token_scale(),
    opacity = token_scale(),
    shape = "circle",
    attributes = list()
)
```

**Arguments**

- **size**: The scale used for the token size.
- **color**: The scale used for the token color.
- **image**: The scale used for the token image.
- **opacity**: The scale used for the token opacity.
- **shape**: The (fixed) SVG shape to be used to draw tokens. Can be either 'circle' (default), 'rect' or 'image'. In the latter case the image URL needs to be specified as parameter 'token_image'.
- **attributes**: A list of additional (fixed) SVG attributes to be added to each token.

**Value**

An aesthetics mapping for 'animate_process'.

**See Also**

`animate_process`, `token_scale`

**Examples**

```
data(example_log)

# Change default token sizes / shape
animate_process(example_log, mapping = token_aes(size = token_scale(12), shape = "rect"))

# Change default token color
animate_process(example_log, mapping = token_aes(color = token_scale("red")))
```
# Change default token opacity
animate_process(example_log, mapping = token_aes(opacity = token_scale("0.2")))

# Change default token image (GIFs work too)
animate_process(example_log,
    mapping = token_aes(shape = "image",
    size = token_scale(10),
    image = token_scale("https://upload.wikimedia.org/wikipedia/en/5/5f/Pacman.gif"))

# A more elaborate example with a secondary data frame
library(eventdataR)
data(traffic_fines)
# Change token color based on a numeric attribute, here the nonsensical 'time' of an event
animate_process(edeaR::filter_trace_frequency(bupaR::sample_n(traffic_fines,1000),percentage=0.95),
    legend = "color", mode = "relative",
    mapping = token_aes(color = token_scale("amount",
        scale = "linear",
        range = c("yellow","red"))))

---

**token_scale**

*Token scale mapping values to aesthetics*

**Description**

Creates a ‘list’ of parameters suitable to be used as token scale in (**token_aes**) for mapping values to certain aesthetics of the tokens in a process map animation. Refer to the d3-scale documentation (https://github.com/d3/d3-scale) for more information about how to set 'domain' and 'range' properly.

**Usage**

```r
token_scale(attribute = NULL,
    scale = c("identity", "linear", "sqrt", "log", "quantize", "ordinal", "time"),
    domain = NULL,
    range = NULL
)
```

**Arguments**

| attribute | This may be (1) the name of the event attribute to be used as values, (2) a data frame with three columns (case, time, value) in which the values in the case column are matching the case identifier of the supplied event log, or (3) a constant value that does not change over time. |
The domain of the D3 scale function. Can be left NULL in which case it will be automatically determined based on the values.

range

The range of the D3 scale function. Should be a vector of two or more numerical values.

Value

A scale to be used with 'token_mapping'

See Also

animate_process, token_aes

Examples

data(example_log)

# (1) Change token color based on a factor attribute
animate_process(example_log, legend = "color", mapping = token_aes(color = token_scale("res", scale = "ordinal", range = RColorBrewer::brewer.pal(8, "Paired"))))

# (2) Change token color based on second data frame
x <- data.frame(case = as.character(rep(c(1,2,3), 2)),
               time = seq(from = as.POSIXct("2018-10-03 03:41:00"), to = as.POSIXct("2018-10-03 06:00:00"), length.out = 6),
               value = rep(c("orange", "green"), 3), stringsAsFactors = FALSE)
animate_process(example_log, mode = "relative", jitter = 10, legend = "color", mapping = token_aes(color = token_scale(x)))

# (3) Constant token color
animate_process(example_log, legend = "color", mapping = token_aes(color = token_scale("red")))
Description
Decoration callback for token selection

Usage
	token_select_decoration(stroke = "black")

Arguments
stroke Sets the ‘stroke’ attribute of selected tokens.

Value
A JavaScript callback function called when token selection changes.

See Also
animate_process

Examples
	# Create a decoration callback that paints tokens red
	token_select_decoration("red")
Index

* datasets
  example_log, 6
  activity_select_decoration, 2
  animate_process, 3, 8, 10, 11, 13
  attribution_osm, 5
createWidget, 4
example_log, 6
icon_circle, 6
icon_marker, 7
JS, 4
process_map, 3–5
processanimaterOutput, 7
renderer_graphviz, 4, 8
renderer_leaflet, 4, 9
renderProcessanimater, 10
sizingPolicy, 5
token_aes, 4, 5, 11, 12, 13
token_scale, 11, 12
token_select_decoration, 13