Package ‘heuristicsmineR’

April 4, 2023

Type  Package
Title  Discovery of Process Models with the Heuristics Miner
Version  0.3.0
Description  Provides the heuristics miner algorithm for process discovery as proposed by Weijters et al. (2011) <doi:10.1109/CIDM.2011.5949453>. The algorithm builds a causal net from an event log created with the ‘bupaR’ package. Event logs are a set of ordered sequences of events for which ‘bupaR’ provides the S3 class eventlog(). The discovered causal nets can be visualised as ‘htmlwidgets’ and it is possible to annotate them with the occurrence frequency or processing and waiting time of process activities.
License  MIT + file LICENSE
Encoding  UTF-8
LazyData  true
LinkingTo  Rcpp, BH
SystemRequirements  C++
Depends  R (>= 2.10)
Imports  bupaR, processmapR (>= 0.3.1), rlang, magrittr, dplyr, tidyr, DiagrammeR (>= 1.0.0), petrinetR (>= 0.3.0), purrr, scales, Rcpp, ggplot2, ggthemes, data.table, stringr
Suggests  eventdataR, svgPanZoom, DiagrammeRsvg
RoxygenNote  7.2.3
URL  https://github.com/bupaverse/heuristicsmineR
BugReports  https://github.com/bupaverse/heuristicsmineR/issues
NeedsCompilation  yes
Author  Felix Mannhardt [aut, cre],
        Gert Janssenswillen [ctb]
Maintainer  Felix Mannhardt <f.mannhardt@tue.nl>
Repository  CRAN
Date/Publication  2023-04-04 13:20:06 UTC
as.petrinet

R topics documented:

as.petrinet ..................................................... 2
causalBindings .................................................... 3
causalCustom ..................................................... 3
causalFrequency .................................................. 4
causalNet ........................................................ 5
causalPerformance ............................................... 7
dependencyMatrix ................................................. 7
dependencyTypeFHM ............................................... 8
dependencyTypeLifecycle ......................................... 10
hospitalMultiPerspective ....................................... 11
LHeur1 .......................................................... 11
LHeur2 .......................................................... 12
parallelMatrixLifecycle ......................................... 12
plotDependencyMatrix ............................................ 13
precedenceMatrix ............................................... 13
precedenceMatrixAbsolute ...................................... 14
precedenceMatrixLengthTwoLoops ................................ 14
precedenceMatrixLifecycle .................................... 15
printCausalNet .................................................. 15
printDependencyMatrix .......................................... 16
renderCausalNet ................................................ 16
renderDependencyMatrix ......................................... 17

Index

as.petrinet ................................. Converts the object to a Petrinet

Description

Converts the object to a Petrinet

Usage

as.petrinet(obj)

Arguments

obj The event log to be used. An object of class

Examples

data(LHeur1)
cn <- causalNet(LHeur1, threshold = .8)
pn <- as.petrinet(cn)
petrinetR::render_PN(pn)
causal_bindings Compute input and output bindings

Description

Computes the input- and output bindings for use in a causal map. Several heuristics may be used to determine the activities that are activated or consumed by an event. The Flexible Heuristic Miner (FHM) paper describes a heuristic that looks ahead (or looks back) until the end of the trace and determines those activities as activated for which no other cause (activity in a causal dependency) is found. This approach is implemented as type nearest.

Usage

causal_bindings(eventlog, dependencies, type = c("nearest"))

Arguments

eventlog The bupaR event log.
dependencies A dependency matrix obtained, for example, through dependency_matrix.
type The heuristic used to determine the bindings. Currently only nearest is available.

Value

A data frame

Examples

causal_bindings(L_heur_1, dependencies = dependency_matrix(L_heur_1))

causal_custom Custom map profile

Description

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

causal_custom(
  FUN = mean,
  attribute,
  units = "",
  color_scale = "RdPu",
  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
attribute The name of the case attribute to visualize (should be numeric)
units   Character to be placed after values (e.g. EUR for monetary euro values)
color_scale Name of color scale to be used for nodes. Defaults to RdPu See
            Rcolorbrewer::brewer_pal.info() for all options.
color_edges The color used for edges. Defaults to red4.
...     Additional arguments forwarded to FUN

Details

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

Examples

causal_net(L_heur_1,
  type_nodes = causal_custom(attribute = "timestamp"),
  type_edges = causal_custom(attribute = "timestamp"))

causal_frequency Frequency map profile

Description

Function to create a frequency profile for a process map.

Usage

causal_frequency(
  value = c("absolute", "relative"),
  color_scale = "PuBu",
  color_edges = "dodgerblue4"
)
causal_net

Arguments

value The type of frequency value to be used: absolute, relative (percentage of activity instances).

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.

Examples

causal_net(L_heur_1,
  type = causal_frequency("relative"))

causal_net eventlog Dependencies Create a Causal net (also Heuristics net)

Description

Creates a Causal net, also known as Heuristics net. This is similar to a processmapR process map. However, the causal map deals with parallelism by trying to identifying causal dependencies between activities by using different heuristics as documented in dependency_matrix.

Usage

causal_net(
  eventlog = NULL,
  dependencies = dependency_matrix(eventlog = eventlog, threshold = threshold,
    threshold_frequency = threshold_frequency, ...),
  bindings = causal_bindings(eventlog, dependencies),
  threshold = 0.9,
  threshold_frequency = 0,
  type = causal_frequency("absolute"),
  sec = NULL,
  type_nodes = type,
  type_edges = type,
  sec_nodes = sec,
  sec_edges = sec,
  ...)

Arguments

eventlog The event log for which a causal map should be computed. Can be left NULL for more control if parameters dependencies and bindings are provided directly.

dependencies A dependency matrix created for the event log, for example, by dependency_matrix.

bindings Causal bindings created by causal_bindings.
threshold

The dependency threshold to be used when using the default dependency matrix computation.

threshold_frequency

The frequency threshold to be used when using the default dependency matrix computation.

type

A causal map type. For example, causal_frequency or causal_performance.

sec

A causal process map type. Values are shown between brackets.

type_nodes

A causal map type to be used for nodes only.

type_edges

A causal map type to be used for edges only.

sec_nodes

A secondary causal map type for nodes only.

sec_edges

A secondary causal map type for edges only.

... Further parameters forwarded to the default dependency_matrix function.

Details

Warning: Projected frequencies are heuristically determined and counts may not add up.

Value

A DiagrammeR graph of the causal map.

Examples

# Causal map with default parameters
causal_net(L_heur_1)

# Causal map with lower dependency threshold
causal_net(L_heur_1, threshold = .8)

# For even more control omit the \code{eventlog} parameter
# and provide \code{dependencies} and \code{bindings} directly.
# d <- dependency_matrix(L_heur_1, threshold = .8)
causal_net(dependencies = d,
        bindings = causal_bindings(L_heur_1, d, "nearest"))

# The returned DiagrammeR object can be further augmented with
# panning and zooming before rendering:
library(magrittr)
causal_net(L_heur_1) %>%
    render_causal_net(render = TRUE) %>%
    DiagrammeRsvg::export_svg() %>%
svgPanZoom::svgPanZoom()
causal_performance  

**Performance map profile**

**Description**

Function to create a performance profile for a causal map.

**Usage**

```r
causal_performance(
  FUN = mean,
  units = c("mins", "secs", "hours", "days", "weeks", "months", "quarters", "semesters", "years"),
  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
- **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 forwarded to `FUN`

**Examples**

```r
causal_net(L_heur_1, type = causal_performance())
```

---

dependency_matrix  

**Create a dependency matrix**

**Description**

Creates a dependency matrix from a precedence matrix (`precedence_matrix`) based on different approaches.
Usage

dependency_matrix(
  eventlog = NULL,
  dependency_type = dependency_type_fhm(threshold_dependency = threshold,
    threshold_frequency = threshold_frequency, ...),
  threshold = 0.9,
  threshold_frequency = 0,
  ...
)

Arguments

  eventlog A bupaR event log, may be NULL when a precedence matrix is provided.
  dependency_type Which approach to use for calculation of the dependency matrix. Currently only (dependency_type_fhm) is available.
  threshold A dependency threshold, usually in the interval [0,1], filtering out dependencies below the threshold.
  threshold_frequency An absolute frequency threshold filtering dependencies which are observed infrequently.
  ... Parameters forwarded to (dependency_type_fhm).

Value

A square matrix with class dependency_matrix containing the computed dependency values between all activities.

See Also

precedence_matrix

Examples

d <- dependency_matrix(L_heur_1)
print(d)
as.matrix(d)

dependency_type_fhm Dependency type based on Flexible Heuristics Miner (FHM)

Description

Computes the dependencies based on the approach known as Flexible Heuristics Miner.
Usage

dependency_type_fhm(
  threshold_dependency = 0.9,
  threshold_l1 = threshold_dependency,
  threshold_l2 = threshold_dependency,
  threshold_frequency = 0,
  all_connected = FALSE,
  endpoints_connected = FALSE
)

Arguments

threshold_dependency
  A dependency threshold, usually in the interval \([0,1]\), filtering out dependencies below the threshold.

threshold_l1
  A dependency threshold, usually in the interval \([0,1]\), filtering out self-loop dependencies below the threshold.

threshold_l2
  A dependency threshold, usually in the interval \([0,1]\), filtering out length-two loop dependencies below the threshold.

threshold_frequency
  An absolute frequency threshold filtering dependencies which are observed infrequently.

all_connected
  If TRUE the best antecedent and consequent (as determined by the dependency measure) are going to be added regardless of the threshold value.

endpoints_connected
  If TRUE the start/end activity is added as antecedent/consequent when an activity would not be connected according to the threshold value.

Value

A dependency type.

References


Examples

dependency_matrix(L_heur_1,
  dependency_type = dependency_type_fhm(all_connected = TRUE))
dependency_type_lifecycle

_Dependency type based on time intervals_

**Description**

Computes the dependencies based on the approach taking into account activity durations based on life-cycle transitions.

**Usage**

```r
dependency_type_lifecycle(
  threshold_dependency = 0.9,
  threshold_l1 = threshold_dependency,
  threshold_frequency = 0,
  all_connected = FALSE,
  endpoints_connected = FALSE
)
```

**Arguments**

- `threshold_dependency`  
  A dependency threshold, usually in the interval $[0,1]$, filtering out dependencies below the threshold.
- `threshold_l1`  
  A dependency threshold, usually in the interval $[0,1]$, filtering out self-loop dependencies below the threshold.
- `threshold_frequency`  
  An absolute frequency threshold filtering dependencies which are observed infrequently.
- `all_connected`  
  If TRUE the best antecedent and consequent (as determined by the dependency measure) are going to be added regardless of the threshold value.
- `endpoints_connected`  
  If TRUE the start/end activity is added as antecedent/consequent when an activity would not be connected according to the threshold value.

**Value**

A dependency type.

**References**

**Examples**

```r
dependency_matrix(L_heur_1, 
    dependency_type = dependency_type_fhm(all_connected = TRUE))
```

**Description**

*Hospital example event log capturing multi-perspectives*

Sample of 10,000 traces from an artificial eventlog from the PhD thesis ‘Multi-perspective Process Mining’ used to illustrate the Data-aware Heuristics Miner algorithm.

**Usage**

```r
hospital_multi_perspective
```

**Format**

Eventlog containing a sample of 10,000 cases

**Source**

`doi:10.4121/uuid:32cad43f8bb946af833348aae2bea037`

**References**


**Description**

*Heuristics miner example log #1*

Artificial eventlog for illustrating Heuristics Miner published as supplementary material to the book *Process Mining: Discovery, Conformance and Enhancement of Business Processes*.

**Usage**

```r
L_heur_1
```

**Format**

Eventlog containing 40 cases
References


L_heur_2

*Heuristics miner example log #2*

Description

Artificial eventlog for illustrating Heuristics Miner published as supplementary material to the book Process Mining: Discovery, Conformance and Enhancement of Business Processes.

Usage

L_heur_2

Format

Eventlog containing 85 cases

References


parallel_matrix_lifecycle

*Parallel Matrix with Lifecycle*

Description

Parallel Matrix with Lifecycle

Usage

parallel_matrix_lifecycle(eventlog)

Arguments

eventlog The event log object to be used.

Examples

parallel_matrix_lifecycle(L_heur_1)
plot.dependency_matrix

Dependency matrix plot

Description
Visualize a dependency matrix. A generic plot function for dependency matrices.

Usage
## S3 method for class 'dependency_matrix'
plot(x, ...)

Arguments
x Dependency matrix
...
Additional parameters

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

precedence_matrix

Precedence Matrix

Description
Construct a precedence matrix, showing how activities are followed by each other. This is a performance improved variant of precedence_matrix in the processmapR package.

Usage
precedence_matrix(
  eventlog,
  type = c("absolute", "relative", "relative-antecedent", "relative-consequent", "relative-case")
)

Arguments
eventlog The event log object to be used
type The type of precedence 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.
Examples

m <- precedence_matrix(hospital_multi_perspective, type = "absolute")
print(m)
as.matrix(m)

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.

Examples

library(eventdataR)
data(traffic_fines)
m <- precedence_matrix_absolute(traffic_fines)
print(m)
as.matrix(m)

precedence_matrix_length_two_loops

Length Two Loop Precedence Matrix

Description

Construct a precedence matrix counting how often pattern aba occurs.

Usage

precedence_matrix_length_two_loops(eventlog)
precedence_matrix_lifecycle

Arguments
  eventlog  The event log object to be used.

Examples
  m <- precedence_matrix_length_two_loops(hospital_multi_perspective)
  print(m)
  as.matrix(m)

precedence_matrix_lifecycle
  Precedence Matrix with Lifecycle

Description
  Precedence Matrix with Lifecycle

Usage
  precedence_matrix_lifecycle(eventlog)

Arguments
  eventlog  The event log object to be used.

Examples
  precedence_matrix_lifecycle(L_heur_1)

print.causal_net
  Generic print function for a Causal net

Description
  Generic print function for a Causal net

Usage
  # S3 method for class 'causal_net'
  print(x, ...)

Arguments
  x  Causal net object
  ...  Additional Arguments
print.dependency_matrix

*Generic print function for a dependency matrix*

**Description**

Generic print function for a dependency matrix

**Usage**

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

**Arguments**

- `x` dependency matrix object
- `...` Additional Arguments

---

render_causal_net

*Renders a Causal net as graph*

**Description**

Renders a Causal net as graph

**Usage**

```r
render_causal_net(
  causal_net,
  rankdir = "LR",
  layout = "dot",
  render = T,
  fixed_edge_width = F,
  fixed_node_pos = NULL,
  ...
)
```

**Arguments**

- `causal_net` A causal net created by `causal_net`
- `rankdir` Rankdir to be used for DiagrammeR.
- `layout` Layout to be used for DiagrammeR.
- `render` Whether to directly render the DiagrammeR graph or simply return it.
**render_dependency_matrix**

Renders a dependency matrix as dependency graph

**Description**

Creates a dependency graph visualizing the contents of a dependency matrix.

**Usage**

```r
render_dependency_matrix(
  dependencies,
  rankdir = "LR",
  layout = "dot",
  render = T
)
```

**Arguments**

- **dependencies**: A dependency matrix created by `dependency_matrix`
- **rankdir**: Rankdir to be used for DiagrammeR.
- **layout**: Layout to be used for DiagrammeR.
- **render**: Whether to directly render the DiagrammeR graph or simply return it.

**Value**

A DiagrammeR graph of the (filtered) dependency matrix.

**Examples**

```r
render_dependency_matrix(dependency_matrix(L_heur_1))
```
Index

* datasets
  - hospital_multi_perspective, 11
  - L_heur_1, 11
  - L_heur_2, 12

as.petrinet, 2

causal_bindings, 3, 5
causal_custom, 3
causal_frequency, 4, 6
causal_net, 5, 16
causal_performance, 6, 7

dependency_matrix, 3, 5, 6, 7, 17
dependency_type_fhm, 8, 8
dependency_type_lifecycle, 10

hospital_multi_perspective, 11

L_heur_1, 11
L_heur_2, 12

parallel_matrix_lifecycle, 12
plot.dependency_matrix, 13
precedence_matrix, 7, 8, 13, 13, 14
precedence_matrix_absolute, 14
precedence_matrix_length_two_loops, 14
precedence_matrix_lifecycle, 15
print.causal_net, 15
print.dependency_matrix, 16

render_causal_net, 16
render_dependency_matrix, 17