Package ‘processcheckR’

November 5, 2019

Type Package
Title Rule-Based Conformance Checking of Business Process Event Data
Version 0.1.2
Date 2019-11-05
Description Check compliance of event-data from (business) processes with respect to specified rules. Rules supported are of three types: frequency (activities that should (not) happen x number of times), order (succession between activities) and exclusiveness (and and exclusive choice between activities).
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports dplyr, bupaR, rlang, edeaR, stringr, glue
RoxygenNote 6.1.1
Suggests knitr, rmarkdown, eventdataR
VignetteBuilder knitr
BugReports https://github.com/bupaverse/processcheckr/issues
NeedsCompilation no
Author Gert Janssenswillen [aut, cre]
Maintainer Gert Janssenswillen <gert.janssenswillen@uhasselt.be>
Repository CRAN
Date/Publication 2019-11-05 12:40:02 UTC

R topics documented:

absent .................................................. 2
and ....................................................... 3
check_rule .............................................. 4
check_rules ............................................ 4
absent

Description

The ‘absent’ rule can be used to check whether an activity is absent in a case or not. The ‘n’ parameter can be configured to create a different level of absence. When n = 1, an activity is not allowed to occur even a single time. The maximum number of times it is allowed to occur is ‘n-1’.

Usage

absent(activity, n = 0)

Arguments

- **activity**: The activity to check. Character vector of length one.
- **n**: n is the allowed number of occurrences of the activity. E.g. n = 0 means the activity should be absent, n = 1 means it is allowed to occur once.

See Also

Other Declarative Rules: and, contains_between, contains_exactly, contains, ends, precedence, responded_existence, response, starts, succession, xor

Examples

```r
library(bupaR)
library(eventdataR)

# Check for which patients the activity "MRI SCAN" is absent.
patients %>%
check_rule(absent("MRI SCAN"))

# Check for which patients the activity "Blood test"
```
and

# occurs maximum a single time, but not 2 times or more.
patients %>%
check_rule(absent("Blood test", n = 1))

---

and  

Check for co-existence of two activities

Description

The `and` rule checks whether two activities both occur in a case (or are both absent). If activity A exists, Activity B should also exist, and vice versa.

Usage

```r
and(activity_a, activity_b)
```

Arguments

- **activity_a**: Activity A. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.
- **activity_b**: Activity B. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.

See Also

Other Declarative Rules: `absent`, `contains_between`, `contains_exactly`, `contains`, `ends`, `precedence`, `responded_existence`, `response`, `starts`, `succession`, `xor`

Examples

```r
library(bupaR)
library(eventdataR)

# Check that if a patients is registered, he's also checked-out, and vice versa.
patients %>%
check_rule(and("Registration","Check-out"))
```
check_rule

Check declarative rules.

Description

This function can be used to check rules on event data. It needs an event log and a rule. Rules can be made with the following functions: absent(), and(), contains(), contains_between(), contains_exactly(), ends(), precedence(), response(), responded_existence(), starts(), succession(), xor().

Usage

check_rule(eventlog, rule, label = NULL)

Arguments

eventlog  
Eventlog object

rule  
A rule create by a rule function.

label  
Optionally, the variable name under which the result of the rule should be stored.

Value

An annotated event log, where a new column indicates whether the rule holds or not. The name of the new column can optionally be set using the "label" argument.

Examples

library(bupaR)
library(eventdataR)

# check whether MRI Scan is preceded by Blood test.
patients %>%
  check_rule(precedence("Blood test","MRI SCAN"))

check_rules

Check multiple declarative rules.

Description

This function can be used to check several rules on event data. It needs an event log and a rule. Rules can be made with the following functions: absent(), and(), contains(), contains_between(), contains_exactly(), ends(), precedence(), response(), responded_existence(), starts(), succession(), xor().

Usage

check_rules(eventlog, ...)

contains

Arguments
  eventlog  Eventlog object
  ...  Name-rule pairs.

Value
  An annotated event log, where - for every rule - a new column indicates whether the rule holds or
  not. The name of each rule becomes the name of the column.

Examples

library(eventdataR)

# check whether MRI Scan is preceded by Blood test, and the case starts with Registration
check.rules(patients,
    rule1 = precedence("Blood test","MRI SCAN"),
    rule2 = starts("Registration"))

contains  Check if activity is present (contained) in a case

Description
  This rules examines whether the supplied activity is present in a case or not. The argument ‘n’ can
  be used to set a minimum number of occurences that should be present in each case. Using the
  function ‘check_rule’, this information can be added to the event log.

Usage
  contains(activity, n = 1)

Arguments
  activity  Activity to check. A character vector of length one. Should be an activity of the
            eventlog supplied with check_rule.
  n  The minimum number of times the activity should be present.

See Also
  Other Declarative Rules: absent, and, contains_between, contains_exactly, ends, precedence,
  responded_existence, response, starts, succession, xor
contains_between

Examples

```r
library(bupaR)
library(eventdataR)

# Each patient should be registered at least once.
patients %>%
  check_rule(contains("Registration"))

# Check whether some patients have received 2 or more blood tests.
patients %>%
  check_rule(contains("Blood test", n = 2))
```

Description

This rule examines whether the supplied activity is present in a case for a certain interval of times. The arguments ‘min’ and ‘max’ can be used to specify the allowed interval of occurrences.

Usage

```r
contains_between(activity, min = 1, max = 1)
```

Arguments

- `activity`: Activity to check. Character vector of length one. This should be an activity of the event log supplied with `check_rule`.
- `min`: The minimum number of times the activity should be present.
- `max`: The maximum number of times the activity should be present.

See Also

Other Declarative Rules: `absent, and, contains_exactly, contains, ends, precedence, responded_existence, response, starts, succession, xor`

Examples

```r
library(bupaR)
library(eventdataR)

# A patients should have between 0 and 4 blood tests (including 0 and 4).
patients %>%
  check_rule(contains_between("Blood test", min = 0, max = 4))
```
contains_exactly

Description

This rule examines whether the supplied activity is present in a case for an exact number of times.

Usage

contains_exactly(activity, n = 1)

Arguments

- activity: Activity to check. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.
- n: The exact number of times the activity should be present.

See Also

Other Declarative Rules: absent, and, contains_between, contains, ends, precedence, responded_existence, response, starts, succession, xor

Examples

```r
library(bupaR)
library(eventdataR)

# Each patient should have exactly one registration activity instance.
patients %>%
check_rule(contains_exactly("Registration", n = 1))
```

ends

Description

Check if cases end with an activity.

Usage

ends(activity)
Arguments

activity: The end activity. Character vector of length one. This should be an activity of the event log supplied to 'check_rule'.

See Also

Other Declarative Rules: absent, and, contains_between, contains_exactly, contains, precedence, responded_existence, response, starts, succession, xor

Examples

```r
library(bupaR)
library(eventdataR)

# A patient's last activity should be the Check-out
patients %>%
  check_rule(ends("Check-out"))
```

Description

This function can be used to filter event data using declaritive rules. It needs an event log and a rule. Rules can be made with the following functions: absent(), and(), contains(), contains_between(), contains_exactly(), ends(), precedence(), response(), responded_existence(), starts(), succession(), xor().

Usage

```r
filter_rules(eventlog, ...)
```

Arguments

- `eventlog`: Eventlog object
- `...`: rules

Value

A filtered event log.

Examples

```r
library(eventdataR)

# Filter where Blood test precedes MRI SCAN and Registration is the start of the case.
filter_rules(patients, precedence("Blood test","MRI SCAN"),
            starts("Registration"))
```
precedence

Check for precedence between two activities.

Description

If activity B occurred, it should be preceded by activity A in the same case.

Usage

precedence(activity_a, activity_b)

Arguments

activity_a Activity A. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.
activity_b Activity B. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.

See Also

Other Declarative Rules: absent, and, contains_between, contains_exactly, contains, ends, responded_existence, response, starts, succession, xor

Examples

library(bupaR)
library(eventdataR)

# A MRI Scan should be preceded by a Blood test.
patients %>%
check_rule(precedence("Blood test","MRI SCAN"))

processcheckR

processcheckR - Check rules in event data

Description

Tools to check declarative rules in event logs.


responded_existence  
*Check for responded existence between two activity*

**Description**
If activity A occurs in a case, activity B should also occur (before or after).

**Usage**

```r
responded_existence(activity_a, activity_b)
```

**Arguments**
- `activity_a`: Activity A. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.
- `activity_b`: Activity B. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.

**See Also**
Other Declarative Rules: `absent`, `and`, `contains_between`, `contains_exactly`, `contains`, `ends`, `precedence`, `response`, `starts`, `succession`, `xor`

**Examples**

```r
library(bupaR)
library(eventdataR)

# When a Blood test occurs, a MRI Scan should also have
# happened for this patient (before or after the test).

patients %>%
  check_rule/responded_existence("Blood test","MRI SCAN")
```

---

response  
*Check for response between two activities*

**Description**
If activity A is executed, it should be eventually followed by activity B.

**Usage**

```r
response(activity_a, activity_b)
```
starts

Arguments

activity_a  Activity A. A character vector of length one. This should be an activity of the
             event log supplied to `check_rule`.
activity_b  Activity B. A character vector of length one. This should be an activity of the
             event log supplied to `check_rule`.

See Also

Other Declarative Rules: absent, and, contains_between, contains_exactly, contains, ends,
precedence, responded_existence, starts, succession, xor

Examples

library(bupaR)
library(eventdataR)

# A blood test should eventually be followed by Discuss Results
patients %>%
  check_rule(response("Blood test","Discuss Results"))

starts  Check if cases start with an activity

Description

Check if cases start with an activity

Usage

starts(activity)

Arguments

activity  The start activity. Character vector of length one. This should be an activity of
           the event log supplied to `check_rule`.

See Also

Other Declarative Rules: absent, and, contains_between, contains_exactly, contains, ends,
precedence, responded_existence, response, succession, xor
succession

Examples

```r
library(bupaR)
library(eventdataR)

# Each patients should first be registered.
patients %>%
  check_rule(starts("Registration"))
```

---

succession | Check succession between two activities

Description

If activity A happens, it should be eventually followed by activity B. If activity B happens, it should be preceded by activity A.

Usage

```r
succession(activity_a, activity_b)
```

Arguments

- `activity_a` | Activity A. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.
- `activity_b` | Activity B. A character vector of length one. This should be an activity of the event log supplied to `check_rule`.

See Also

Other Declarative Rules: `absent`, `and`, `contains_between`, `contains_exactly`, `contains`, `ends`, `precedence`, `responded_existence`, `response`, `starts`, `xor`

Examples

```r
library(bupaR)
library(eventdataR)

# Blood test should always happen before a MRI Scan,
# and both should happen when one of them happens.
patients %>%
  check_rule(succession("Blood test","MRI SCAN"))
```
xor

Check for exclusiveness of two activities

Description

If activity A exists, Activity B should not exist, and vice versa.

Usage

xor(activity_a, activity_b)

Arguments

activity_a  Activity A. A character vector of length one. This should be an activity of the event log supplied to 'check_rule'.
activity_b  Activity B. A character vector of length one. This should be an activity of the event log supplied to 'check_rule'.

See Also

Other Declarative Rules: absent, and, contains_between, contains_exactly, contains, ends, precedence, responded_existence, response, starts, succession

Examples

library(bupaR)
library(eventdataR)

# A patient should not receive both an X-Ray and MRI Scan
patients %>%
  check_rule(xor("X-Ray","MRI SCAN"))
Index

absent, 2, 3, 5–13
and, 2, 3, 5–13

check_rule, 4
check_rules, 4
contains, 2, 3, 5, 6–13
contains_between, 2, 3, 5, 6, 7–13
contains_exactly, 2, 3, 5, 6, 7, 8–13
ends, 2, 3, 5–7, 7, 9–13

filter_rules, 8

precedence, 2, 3, 5–8, 9, 10–13
processcheckR, 9
processcheckR-package (processcheckR), 9

responded_existence, 2, 3, 5–9, 10, 11–13
response, 2, 3, 5–10, 10, 11–13

starts, 2, 3, 5–11, 11, 12, 13
succession, 2, 3, 5–11, 12, 13

xor, 2, 3, 5–12, 13