

R documentation

of 'Rdrools6-internal.Rd' etc.

July 27, 2015

Rdrools6-package

a rules engine for R based on the popular Java Drools6 engine

Description

Rdrools6 is a robust rules engine based on the popular rules engine Drools6 written in Java. Rules make it easy to express solutions to different problems. They are often expressed as declarative elements of logic and are held in a file repository. In Rdrools, rules are written in the native Drools language. If you are not familiar, please go through the Drools documentation provided in the list of references below.

Details

Package:	Rdrools6
Type:	Package
Version:	0.0.1
Date:	2013-12-07
License:	LGPL-3
LazyLoad:	yes
LazyData:	yes

Rdrools6 is fairly straightforward to use. It expects the user to provide a data frame as an input, a rules file, and a comma separated list of input and output columns. Rdrools then picks a row from the data frame, applies the rules provided to it and maps the output to the list of expected output columns. Incorrect inputs would throw a meaningful error.

Note

For applying a conditional rule, you can use the eval statement in your rules syntax. Please see a detailed example below.

```
import java.util.HashMap;  
global java.util.HashMap output;  
rule"xx"
```

```

dialect "mvel"
  salience3
  when
    input:HashMap()
    eval(input.get("currentAction")==input.get("previousAction"))
  then
    output.put("send", "false")
end
rule "xy"
dialect "mvel"
  salience2
  when
    input:HashMap()
    eval(input.get("currentAction")!=input.get("previousAction"))
  then
    output.put("send", "true")
end

```

Author(s)

Stefano Picozzi <StefanoPicozzi@gmail.com>, SMS Chauhan <smschauhan@gmail.com>, Zubin Dowlaty <zubin@dowlaty.com>

Maintainer: Stefano Picozzi <StefanoPicozzi@gmail.com>

References

Wikipedia:<http://en.wikipedia.org/wiki/Drools>

Drools Documentation:http://docs.jboss.org/drools/release/6.0.0.Final/drools-docs/html_single/index.html

See Also

[rulesSession](#), [runRules](#)

Examples

```

library(Rdrools6)
data(class)
data(rules)
mode<-"STATELESS"
input.columns<-c("name", "class", "grade", "email")
output.columns<-c("address", "subject", "body")
rules.session<-rulesSession(mode, rules, input.columns, output.columns)
output.df<-runRules(rules.session, class)

```

class

Sample data set on which rules from the [rules](#) dataset can be applied

Description

The dataset describes a class of students, their grades and contact details.

Usage

```
data(class)
```

Format

A data frame with 15 observations on the following 5 variables.

id a numeric vector

name a factor with levels James Johnny Joseph

class a factor with levels English History Math Science Spanish

grade a numeric vector

email a factor with levels james@school.edu johhny@school.edu johnny@school.edu joseph@school.edu

Examples

```
data(class)
str(class)
```

rules	<i>a set of rules, for the <code>class</code> dataset</i>
-------	---

Description

a character vector of lines read using `readLines()` from a drools rule file(text files with drools rules)

Usage

```
data(rules)
```

Format

The format is: chr [1:12] "import java.util.HashMap;" "global java.util.HashMap output;" "" ...

Details

You could create a similar rules character vector by using
`rules<-readLines("rules.drl")`

Examples

```
data(rules)
str(rules)
```

rulesSession	<i>Creates a session of the rules engine</i>
--------------	--

Description

The rulesSession creates a session that interfaces between R and the Drools engine. The session is utilized by the runRules function for executing a data frame against a set of rules.

Usage

```
rulesSession(mode, rules, input.columns, output.columns)
```

Arguments

mode	a string indicating whether to create a "STATELESS", "STATEFUL" or "STREAM" session mode<-"STATELESS"
rules	a character vector consisting of lines read from a rules file of format drl(drools rules file) This character vector is eventually collapsed into a character vector of length 1, so the way you read the file could potentially be just about anything
input.columns	a character vector of a set of input column, for example input.columns<-c("name", "class", "grade", "email")
output.columns	a character vector of a set of expected output columns, for example output.columns<-c("address", "subject", "body")

Details

An active drools rules session. This promotes re-usability of a session, i.e. you can utilize the same session repetitively for different data sets of the same format.

Value

```
rules.session.object
```

Returns a session to the rules engine

Note

Please have a look at the examples provided in the "examples" section of the [Rdrools6](#) man page. A sample data set and a set of rules have been supplied help you understand the package usage.

Author(s)

SMS Chauhan <smschauhan@gmail.com>

See Also

[runRules](#), [Rdrools6](#)

Examples

```
library(Rdrools6)
data(rules)
mode<-"STATELESS"
input.columns<-c("name", "class", "grade", "email")
output.columns<-c("address", "subject", "body")
rules.session<-rulesSession(mode, rules, input.columns, output.columns)

## The function is currently defined as
function(mode,rules,input.columns, output.columns)
{
  mode <- paste(mode)
  rules <- paste(rules, collapse="\n")
  input.columns<-paste(input.columns,collapse=",")
  output.columns<-paste(output.columns,collapse=",")
  droolsSession<-.jnew('org/math/r/drools/DroolsService',rules,input.columns, output.columns)
  return(droolsSession)
}
```

runRules

apply a set of rule transformations to a data frame

Description

This function is the core of the Rdrools6 package. Rules are applied on an input data frame and the results are returned as the output of the function. The columns on which the rules need to be applied have to be provided explicitly. Additionally, the new columns that would be created based on the rules have to be provided explicitly as well.

The rules engine picks up a row from the dataframe, applies the transformation to it based on rules provided and saves the result in an output dataframe.

Usage

```
runRules(rules.session, input.df)
```

Arguments

rules.session	a session of the rules engine created using the the rulesSession
input.df	a dataframe consisting of a set of rows you wish to transform, and columns you wish to use in the transformation

Details

If you are not familiar with the drools file format, please have a look at the references provided in the [Rdrools6](#) man page. More details on how conflicting rules are resolved using either salience or the Rete algorithm are also present in the references.

Value

output.df	a dataframe which is the result of transformations applied to the input dataframe(<code>input.df</code>), the columns being the list provided through the <code>output.columns</code> parameter in rulesSession .
-----------	---

Warning**Transformation policy**

Transformations are applied row by row, iteratively. That is to say, all inputs required for a rule transformation should be present in columns as a part of that row itself. Each row should be considered independent of another; all input values required for a transformation should be available in that row itself. The expectation from rules engines are often misplaced.

Column Mismatch

Please make sure that the list of output columns provided through the `output.columns` parameter is exhaustive. Any additional column which is created through the rules transformation but is not present in the list would inhibit proper functioning. In most cases, an error should be thrown.

Author(s)

SMS Chauhan <smschauhan@gmail.com>

See Also

[Rdrools6](#), [rulesSession](#)

Examples

```
library(Rdrools6)
data(class)
data(rules)
mode<-"STATELESS"
input.columns<-c("name", "class", "grade", "email")
output.columns<-c("address", "subject", "body")
rules.session<-rulesSession(mode,rules, input.columns, output.columns)
output.df<-runRules(rules.session, class)

## The function is currently defined as
function(rules.session,input.df)
{
  conn<-textConnection('input.csv.string','w')
  write.csv(input.df,file=conn)
  close(conn)
  input.csv.string<-paste(input.csv.string, collapse="\n")
  output.csv.string <- .jcall(rules.session, "S", "execute",input.csv.string)
  conn <- textConnection(output.csv.string, "r")
  output.df<-read.csv(file=conn, header=T)
  close(conn)
  return(output.df)
}
```

Index

*Topic **Rdrools6**

Rdrools6-package, [1](#)
rulesSession, [4](#)
runRules, [5](#)

*Topic **datasets**

class, [2](#)
rules, [3](#)

*Topic **rulesSession**

Rdrools6-package, [1](#)
rulesSession, [4](#)
runRules, [5](#)

*Topic **runRules**

Rdrools6-package, [1](#)
rulesSession, [4](#)
runRules, [5](#)

class, [2](#), [3](#)

Rdrools6, [4–6](#)

Rdrools6 (Rdrools6-package), [1](#)

Rdrools6-package, [1](#)

rules, [2](#), [3](#)

rulesSession, [2](#), [4](#), [5](#), [6](#)

runRules, [2](#), [4](#), [5](#)