Package 'provParseR'

October 14, 2022

Title Pulls Information from Prov.Json Files

Version 1.0

Date 2022-08-08

Description R functions to access provenance information collected by 'rdt' or 'rdtLite'. The information is stored inside a 'ProvInfo' object and can be accessed through a collection of functions that will return the requested data. The exact format of the JSON created by 'rdt' and 'rdtLite' is described in <https://github.com/End-to-end-provenance/ExtendedProvJson>.

Depends R (>= 3.3)

License GPL-3 | file LICENSE

Encoding UTF-8

Imports jsonlite, methods

Suggests testthat

URL https://github.com/End-to-end-provenance

RoxygenNote 7.2.0

NeedsCompilation no

Author Orenna Brand [aut],
Joseph Wonsil [aut],
Emery Boose [aut],
Barbara Lerner [cre]

Maintainer Barbara Lerner <blerner@mtholyoke.edu>

Repository CRAN

Date/Publication 2022-08-15 22:20:05 UTC

R topics documented:

get.environment *Provenance access functions*

Description

These functions extract information from a ProvInfo object created by the prov.parse function and return this information as a data frame.

Usage

```
get.environment(prov)

get.libs(prov)

get.tool.info(prov)

get.args(prov)

get.scripts(prov)

get.saved.scripts(prov)

get.proc.nodes(prov)

get.data.nodes(prov)

get.stdout.nodes(prov)

get.error.nodes(prov)

get.func.nodes(prov)

get.proc.proc(prov)

get.data.proc(prov)

get.proc.data(prov)

get.func.proc(prov)

get.func.lib(prov)

get.input.files(prov, only.files = FALSE)

get.urls(prov)

get.output.files(prov)
```

```
get.preexisting(prov)

get.variables.set(prov)

get.variables.used(prov)

get.variable.named(prov, var.name)

get.val.type(prov, node.id = NULL)
```

Arguments

prov	a ProvInfo object created by calling prov.parse.
only.files	If true, the output of get.input.files contains just files. If false, it contains both files and URLs.
var.name	a string containing the name of a variable used in the script the provenance is for
node.id	A vector of node id.

Value

All access functions return NULL if there is no parsed provenance. If parsed provenance exists, but there is no provenance for the type of information requested, such as no input files, an empty data frame is returned.

get.environment returns a data frame containing information about how the provenance was collected. The data frame has 2 columns: label and value. The labels are:

- name - whose value will always be "environment"
- architecture
- operatingSystem
- language
- langVersion
- script - the absolute path to the script executed
- scriptTimeStamp - when the script was last modified
- workingDirectory
- provDirectory - where the provenance is stored
- provTimeStamp - when the provenance was collected
- hashAlgorithm

get.libs returns a data frame describing the libraries used by the script. It contains 3 columns: id, name, and version.

get.tool.info returns a data frame describing the tool that collected the provenance. It contains 3 columns: tool.name, tool.version and json.version.

get.args returns a named list describing the arguments that were passed to prov.run or prov.init when the provenance was collected. Each element is the value of an argument in its original type, each element name is the name of the arguemnt the value corresponds to.

get.scripts returns a data frame identifying all the scripts executed. The main script will be first, followed by all sourced scripts. The data frame contains 2 columns: name and timestamp (when the script was last modified).

get.saved.scripts returns a data frame identifying the location of saved copies of all the scripts executed. The main script will be first, followed by all sourced scripts. The data frame contains 2 columns: name and timestamp (when the script was last modified).

get.proc.nodes returns a data frame identifying all the procedural nodes executed. These are represented in PROV-JSON as activities and include nodes corresponding to lines of code, start or finish nodes that surround blocks of code, and nodes to represent the binding of function arguments to parameters. The data frame contains 8 columns:

- id - a unique id
- name - a description of what the node represents. Often this is a line of code from the script, perhaps shortened
- type - one of Operation, Binding, Start, Finish, or Incomplete
- elapsedTime - when this executed relative to the start of the script
- scriptNum - a number identifing the script it comes from, with script 1 being the main script
- startLine - the line in the script this corresponds to, which may be NA, and the following other position infofmation
- startCol
- endLine
- endCol

get.data.nodes returns a data frame with an entry for each data node in the provenance. The data frame contains the following columns:

- id - a unique id
- name - the descriptive name for the node, which is generally a variable name, file name, or URL
- value - either a text value (possible shortened) or the name of a file where the value is stored
- valType - a description of the value's type, including its container (such as list, vector, etc.), dimensions and member types (such as character, numeric, etc.)
- type - the type of the node, one of Data, Snapshot, File, URL, Exception, Device, Standard-Output, or StandardOutputSnapshot
- scope - a hex number identifying the scope. This is only used for node's with type Data or Snapshot
- fromEnv - a logical value. If true, it means the variable had a value before the script began execution
- hash - the hash value for File nodes
- timestamp - the time at which the node was created

- location - for file nodes, the absolute path to the file

get.stdout.nodes returns a data frame with an entry for each standard output node in the provenance. The data frame contains the following columns:

- id - a unique id
- value - either a text value (possible shortened) or the name of a file where the value is stored
- timestamp - the time at which the node was created

get.error.nodes returns a data frame with an entry for each error node in the provenance. The data frame contains the following columns:

- id - a unique id
- value - either a text value (possible shortened) or the name of a file where the value is stored
- timestamp - the time at which the node was created

get.func.nodes returns a data frame containing information about the functions used from other libraries within the script. The data frame has 2 columns: id (a unique id) and name (the name of the function called).

get.proc.proc returns a data frame containing information about the edges that go between two procedural nodes. These edges indicate a control-flow relationship between the two activities. The data frame has 3 columns: id (a unique id), informant (the tail of the edge), and informed (the head of the edge).

get.data.proc returns a data frame containing information about the edges that go from data nodes to procedural nodes. These edges indicate an input relationship where the data is used by the activity. The data frame has 3 columns: id (a unique id), entity (the input data), and activity (the procedural node that uses the data).

get.proc.data returns a data frame containing information about the edges that go from procedural nodes to data nodes. These edges indicate an output relationship where the data is produed by the activity. The data frame has 3 columns: id (a unique id), entity (the output data), and activity (the procedural node that produces the data).

get.proc.func returns a data frame containing information about where externally-defined functions are used in the script. The data frame has 3 columns: func_id (the id of the function node), activity (the procedural node that calls the function) and function (the function's name).

get.func.lib returns a data frame containing information about what libraries externally-defined functions come from. The data frame has 3 columns: func_id (the id of the function node), library (a library node) and function (the name of a function).

get.input.files returns a data frame containing a subset of the data nodes that correspond to files that are read by the script. If only.files is False, the data frame contains information about both input files and URLs.

get.urls returns a data frame containing a subset of the data nodes that correspond to urls used in the script.

get.output.files returns a data frame containing a subset of the data nodes that correspond to files that are written by the script.

get.preexisting returns a data frame containing variables in the global environment that are used but not set by a script or a console session.

get.variables.set returns a data frame containing a subset of the data nodes that correspond to variables assigned to in the script.

get.variables.used returns a data frame containing a subset of the data nodes that correspond to variables whose values are used in the script.

get.variable.named returns a data frame containing a subset of the data nodes that correspond to variables with the specified name.

A data frame containing the valType of the specified data node, or the valTypes of all data nodes if no data node is specified. Return NULL if there are no data nodes or if the specified data node is not found. If not NULL, the data frame will contain 4 columns in the following order:

- id - The data node id.
- container - The type for the data's container, such as list, vector, etc. NA in cases such as environment and function where the original valType is not a json object.
- dimension - The size of the data, represented as a string list when there are 2 or more dimensions. NA in cases such as environment and function where the original valType is not a json object.
- type - The type(s) contained within the container, represented as a string list in containers such as data frames when there are multiple types. NA in cases like lists where the type of each element can be complex.

See Also

prov.parse

Examples

```
prov <- prov.parse(system.file ("testdata", "prov.json", package="provParseR", mustWork=TRUE))
get.proc.nodes(prov)
get.input.files(prov)
get.urls(prov)
get.output.files(prov)
get.variables.set(prov)
get.variables.used(prov)
get.variable.named(prov, "z")
get.data.nodes(prov)
get.error.nodes(prov)
get.func.nodes(prov)
get.proc.proc(prov)
get.data.proc(prov)
get.proc.data(prov)
get.func.proc(prov)
get.func.lib(prov)
get.libs(prov)
get.scripts(prov)
get.environment(prov)
get.val.type(prov, "d1")
get.tool.info(prov)
get.args(prov)
get.stdout.nodes(prov)
```

`prov.parse` *Provenance parser*

Description

The prov.parse function parses the provenance collected by rdt or rdtLite. This provenance can be stored in a prov.json file or passed to prov.parse as a string. The provParseR package also defines a number of functions that extract and return information from the parsed provenance.

Usage

```
prov.parse(prov.input, isFile = T)
```

Arguments

`prov.input`	A path to a json file that has been created by rdt or rdtLite or a string that is in prov.json format.
`isFile`	A logical value that indicates whether the provenance information is stored in a file (isFile=T) or in a string (isFile=F).

Value

A ProvInfo object that can be passed to the access functions provided by the provParseR package.

See Also

The access functions, including `get.environment`

Examples

```
prov <- prov.parse(system.file ("testdata", "prov.json", package="provParseR", mustWork=TRUE))
```

`ProvInfo-class` *Collection of information gathered from parsing a PROV file*

Description

This is the class that stores provenance information. It is created by prov.parse. Rather than access the slots directly, it is better to use the access functions the package provides.

Slots

`proc.nodes` the procedure nodes

`data.nodes` the data nodes

`func.nodes` the function nodes

`proc.proc.edges` control flow edges

`proc.data.edges` output data edges

`data.proc.edges` input data edges

`func.proc.edges` function use edges

`func.lib.edges` function library edges

`agents` tool that created the provenance

`args` arguments passed when provenance was created

`envi` environmental information

`libs` libraries

`scripts` scripts executed

See Also

The parse function, which creates the ProvInfo object, `prov.parse`

The access functions, including `get.environment`

Index