Package ‘provGraphR’

August 17, 2022

Title Creates Adjacency Matrices for Lineage Searches

Version 1.0.1

Date 2022-08-17

Copyright President and Fellows of Harvard College, Trustees of Mount Holyoke College

Description Creates and manages a provenance graph corresponding to the provenance created by the 'rdtLite' package, which collects provenance from R scripts. 'rdtLite' is available on CRAN. The provenance format is an extension of the W3C PROV JSON format (<https://www.w3.org/Submission/2013/SUBM-prov-json-20130424/>). The extended JSON provenance format is described in <https://github.com/End-to-end-provenance/ExtendedProvJson>.

Depends R (>= 3.5.0)

Imports igraph, Matrix, methods, provParseR (>= 0.2)

Suggests testthat

License GPL-3

Encoding UTF-8

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 2022-08-17 15:30:02 UTC

R topics documented:

`create.graph` *Create provenance graph*

Description

create.graph uses saved provenance to create an adjacency graph that captures the dependencies between data and the R statements that use or modify the data.

get.prov returns the provenance that corresponds with the given adjacency graph

Usage

```
create.graph(prov.input = NULL, isFile = TRUE)

get.prov(adj.graph)
```

Arguments

`prov.input`	This is either a file name, a string containing provenance collected by rdt or rdtLite, or parsed provenance. The exact format of the JSON files is described in ExtendedProvJson.md.
`isFile`	A logical value indicating whether prov.input should be treated as a file name (isFile=TRUE) or a string containing provenance (isFile=False). If prov.input is not a string, this parameter is ignored.
`adj.graph`	the adjacency graph

Details

The graph contains a node for each R statement that is executed, for each variable set, and for each file read or written, and for each URL read. There is an edge from each R statement node to the nodes representing variables set in that statement, or files written by the statement. There is an edge from each variable node to the statement nodes that use the variable with that value. There is also an edge from each input file or URL to the statement node that performs the input operation.

The lineage of any data value can be traced through this graph by calling `get.lineage`.

Value

create.graph returns an object that contains the parsed provenance and a matrix representation of the graph. In the matrix, there is a row and a column for each data and procedure node in the graph. The values in the matrix are either 1 or 0. A 1 indicates that there is an edge for the column node to the row node. create.graph returns NULL if there is no provenance available.

Examples

```
adj.graph <- create.graph(system.file("testdata", "basic.json", package = "provGraphR"))
```

`get.creator` *Get creators and users of data*

Description

get.creator finds the node that creates the given data node

A data node can represent a variable or a file. The users of the data node will be procedure nodes representing the statment that used the variable in an expression or read from the file.

Usage

```
get.creator(adj.graph, data.node.id)

get.users(adj.graph, data.node.id)
```

Arguments

`adj.graph`	the adjacency matrix
`data.node.id`	the id of the data node.

Details

A data node can represent a variable, a file, a plot, or a warning or error. The creator of the data node will be a procedure node representing the statment that assigned the variable, wrote to the file, created the plot, or resulted in the error or warning.

Value

the id of the procedure node that created the specified data node. Returns NULL if there is no node with the given id, the id is not for a data node, or the data node does not have a creator. The last case can occur, for example, if the data node represents an input file.

the id of the procedure node that created the specified data node. Returns NULL if there is no node with the given id, the id is not for a data node, or the data node does not have any users. The last case can occur, for example, if the data node represents an output file.

See Also

`create.graph`

`create.graph`

Examples

```
adj.graph <- create.graph(system.file("testdata", "basic.json", package = "provGraphR"))
get.creator (adj.graph, "d1")

get.users (adj.graph, "d1")
```

---

get.lineage *Calculate lineage of a node*

Description

get.lineage returns either the list of nodes that the provided node depends on (backward lineage) or the list of nodes that depend on the provided node (forward lineage).

Usage

```
get.lineage(adj.graph, node.id, forward = FALSE)
```

Arguments

`adj.graph`	An adjacency graph to get the lineage from, or a ProvGraphInfo object. The object can be created by a call to create.graph.
`node.id`	The string id for a node that the lineage is being requested for
`forward`	Logical that states whether the search is going forward through the graph from the provided node, or backwards.

Details

Most commonly, the node passed in is a data node representing either a variable, a file, or a plot. Forward lineage reports everything computed from that variable or file. Backward lineage reports everything that contributed to the variable's value, the contents of an output file or plot.

Value

get.lineage returns the forward or backward lineage of the specified node. The lineage is represented as a vector of strings, with each string being the id of a node in the lineage. The first entry in the returned vector is the node.id passed in. The remaining entries form a path either forward or backward through the adjacency graph.

See Also

`create.graph`

Examples

```
adj.graph <- create.graph(system.file("testdata", "basic.json", package = "provGraphR"))
get.lineage (adj.graph, "d24")
```

`get.used.data` *Get data used and created by a statement*

Description

get.used.data returns the data nodes that this procedure node uses

A procedure node represents a top-level statement. The data created by the statement can be the variables set, output files written to, plots created, error or warning messages created.

Usage

```
get.used.data(adj.graph, proc.node.id)

get.created.data(adj.graph, proc.node.id)

get.updated.data(adj.graph, proc.node.id)
```

Arguments

`adj.graph`	the adjacency matrix
`proc.node.id`	the id of the procedure node.

Details

A procedure node represents a top-level statement. The data used by the statement are the variables used in expressions or input files or URLs read from.

Value

the ids of the data nodes that are used by the specified procedure node. Returns NULL if there is no node with the given id, the id is not for a procedure node, or the procedure node does not use any data nodes. The last case can occur, for example, if the procedure node represents a statement where a constant is assigned to a variable.

the ids of the data nodes that are created by the specified procedure node. Returns NULL if there is no node with the given id, the id is not for a procedure node, or the procedure node does not create any data nodes. The last case can occur, for example, if the procedure node represents a statement where the statement prints a constant string.

the ids of the data nodes that are updated by the specified procedure node. Returns NULL if there is no node with the given id, the id is not for a procedure node, or the procedure node does not update any data.

See Also

`create.graph`

`create.graph`

`create.graph`

Examples

```
adj.graph <- create.graph(system.file("testdata", "basic.json", package = "provGraphR"))
get.used.data (adj.graph, "p11")

get.created.data (adj.graph, "p11")

get.updated.data (adj.graph, "p5")
```

Index