Package: WayFindR (via r-universe)

August 9, 2024

Version 0.1.2

Date 2024-04-09

Title Computing Graph Structures on WikiPathways

Description Converts pathways from 'WikiPathways' GPML format into 'igraph' objects. Includes tools to find all cycles in the resulting graphs and determine which ones involve negative feedback (inhibition).

Depends R (>= 3.5)

Imports graphics, utils, XML, igraph

Suggests knitr, rmarkdown, Polychrome

License Artistic-2.0

URL http://oompa.r-forge.r-project.org/

VignetteBuilder knitr

NeedsCompilation no

Author Kevin R. Coombes [aut, cre], Polina Bombina [aut]

Maintainer Kevin R. Coombes <krc@silicovore.com>

Repository CRAN

Date/Publication 2024-04-10 19:50:06 UTC

Contents

---

gpml-cycles   *Finding Cycles in Biological Pathways*

Description

Tools to find and interpret cycles in graphs derived from pathways in WikiPathways.

Usage

```
findCycles(graph)
interpretCycle(v, graph)
cycleSubgraph(graph, cycles)
```

Arguments

graph   An igraph object produced by [GPMLtoIgraph](#) from a pathway file in GPML
             format.

v   One of the cycles produced by the findCycles function.

cycles   The list of cycles produced by the findCycles function.

Details

The implementation of the findCycles function, while provably correct, makes no concessions
to efficiency. It is likely to be adequate for the vast majority of biological pathways present in
WikiPathways, but may prove incapable of handling large complex graphs in general. The algorithm
is a sraightforward double loop. It first iterates over all nodes, and uses the neighbors function
from igraph to find all nodes that are directly connected to the start node. It then iterates over those
neighbors and uses the all_simple_paths function from igraph to find all paths that lead back
to the starting node. One aspect of its lack of efficiency is that each cycle with N nodes is found
N times, once for each place you could start traversing the cycle. Before returning the value to the
user, it chooses a unique representative for each such cycle by finding the "earliest" node, based on
its index, to start from.

Value

The findCycles function returns a list. Each element of the list is a cycle, repreented by a named
character vector specifying the nodes in the order that they can be traversed.

The interpretCycle function returns a matrix with two columns, genes and arrows. Each row
contains the gene name (or label) of a node and the form of the interaction arrow relating it to the
next node in the cycle. Likely to be most often used inside an lapply function in order to interpret
all cycles at once.

The cycleSubgraph function returns an igraph object. This value repesents the subgraph of the
full graph defined by all nodes that are part of at least one cycle.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Polina Bombina <pbombina@augusta.edu>

Examples

```
xmlfile <- system.file("pathways/WP3850.gpml", package = "WayFindR")
graf <- GPMLtoIgraph(xmlfile)
cyc <- findCycles(graf)
cyc
CS <- cycleSubgraph(graf, cyc)
plot(CS)
```

---

gpml-data   *GPML GraphingR Data*

Description

Our standard colors, line types (for edges), and shapes (for nodes) to display graph features from WikiPathways GPML files.

Usage

```
data(edgeColors)
data(edgeTypes)
data(nodeColors)
data(nodeShapes)
```

Format

There are four different objects.

edgeColors A named character vector defining the colors used to display 17 different kinds of edges.

edgeTypes A named character vector defining the line types (solid, dashed, etc.) used to display 17 different kinds of edges.

nodeColors A named character vector defining the colors used to display 16 different kinds of nodes.

nodeShapes A named character vector defining the symbols used to display 16 different kinds of nodes.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Polina Bombina <pbombina@augusta.edu>

Source

We downloaded the complete set of 889 human (Homo sapiens) pathways from WikiPathways, in GPML format, at the end of March 2024. We wrote and ran perl scripts to iterate over this set of pathways and extract all the node types and all the edge types used anywhere in the set. We added a handful of node types (Undefined, Shape, Label, EDGE) in order to enable almost all of the pathways to be converted into mathematical graphs. We then defined colors and shapes to be used to distinguish different types in plots.

The situation with edges was slightly more complicated. The pathway editor (PathVisio; https://pathvisio.org/) typically used to create pathways in WikiPathways supports two different biologically meaningful vocabularies for edges: Molecular Interaction Maps (MIM; https://discover.nci.nih.gov/mim/formal_mim_spec.pdf) and System Biology Graphical Notation (SBGN; https://sbgn.github.io/), along with a very simple descrption as "Arrows" or "TBars". To make things more consistent, we converted both the SBGN and simple systems to match the MIM specification. We then defined colors and edge types so we could distinguish different edge types in plots.

Examples

```
data(edgeColors)
data(edgeTypes)
data(nodeColors)
data(nodeShapes)
if (require(Polychrome)) {
  swatch(edgeColors)
  swatch(nodeColors)
}
plot(0,0, type = "n", xlab="", ylab = "")
legend("center", legend = names(edgeColors),  lwd = 3,
       col = edgeColors,  lty = edgeTypes)
num <- c(rectangle = 15, circle = 16)
plot(0,0, type = "n", xlab="", ylab = "")
legend("center", legend = names(nodeColors),  cex = 1.5,
       col = nodeColors,  pch = num[nodeShapes])
```

---

gpml-utility   *Utility Functions to Parse GPML FIles*

Description

Extract entities of different types from GPML files in order to convert the pathway to a mathematical graph that we can compute on.

Usage

```
collectEdges(xmldoc)
collectNodes(xmldoc)
collectGroups(xmldoc, allnodes)
collectAnchors(xmldoc)
collectLabels(xmldoc)
collectShapes(xmldoc)
```

Arguments

xmldoc   Either the name of an XML file meeting the specifications of the Genomic Path-
                way Markup Language (GPML), or an object of class XMLInternalDocument
                obtained by running such a file through the [xmlParseDoc](#) function of the XML
                package. (All of the functions described here will call xmlParseDoc if it hasn't
                already been used.)

allnodes   A data frame containing node information, in the format produced by the collectNodes
                function.

Details

These functions are primarily intended as utility functions that implement processes required by the
main function in the package, [GPMLtoIgraph](#). They have been made accessible to the end user for
use in debugging problematic GPML files or to reuse the GPML files in contexts other than the one
we focus on in this package.

While the meaning of nodes (known as DataNodes in GPML) and edges (known as Interactions
in GPML) should be obvious, some of the other objects are less so. For example, an Anchor in
GPML is an invisible object used to allow an edge to point to another edge instead of to a node.
That structure isn't allowed in graphs in mathematics or computer science. WayFindR handles this
by creating a new node type to represent the anchor position, breaking the target edge into two
pieces separated by the anchor, and adding an edge from the source of the anchored edge to the new
node.

In GPML, a Label is a text box allowing extra information to be placed on a pathway, and a Shape
is a graphical display object. The definition type document (DTD) for GPML describes both of
these entities as non-semantic, intending them for display purposes only. However, some authors of
pathways in the WikiPathways database use such objects as the (usually, final or "leaf") targets of
interaction edges. When that happens, the WayFindR package deals with it by creating actual nodes
to represent such labels or shapes. Other labels and shapes are ignored.

GPML also uses the idea of a Group as a first class object in their DTD. These are defined as
"A collection of structurally or functionally similar or related pathway elements." The GPML file
subclassifies some groups as "Complexes", indicating that they represent physical interactions and
bindings between two or more molecules. Other groups may simply indicate that there is a related
set of molecules (for example, STAT2 and STA3) that play the same role at this point in the pathway.
WayFindR deals with groups by creating a new node to represent the group as a whole and expanding
the component genes into nodes with a single "contained" edge that points to the new group node.

Value

The collectEdges function returns a data frame with three columns (Source, Target, and MIM),
where each row describes one edge (or "Interaction" in the GPML terminology) of the pathway/graph.
The Source and Target columns are the alphanumeric identifiers of items decribing nodes. The
MIM column is the edge type in GPML, which often contains terminology based on the "Molecular
Interaction Map" standard. When creating an [igraph](#) object from a pathway, the first two columns
are used as identifiers to define the nodes at each end of the edge.

The collectNodes function returns a data frame with three columns (GraphId, label, and Type),
where each row describes node or vertex of the pathway/graph. The GraphId column is a unique
alphanumeric identifier. The label column is a human-readable name for the node, often the official

gene symbol. When creating an `igraph` object from a pathway, the first column is used as identifier to define the node. Also, the `plot` method for `igraphs` recognizes the term `label` as a column that defines the text that should be displayed in a node.

The `collectAnchors` function returns a list containing a `nodes` element (in the same format returned by `collectNodes`) and an `edges` element (in the same format returned by `collectEdges`). The `collectGroups` function returns a list with `nodes` and `edges` components, just like the one from `collectAnchors`.

Both the `collectLabels` and `collectShapes` functions return the same kind of data frame that is returned by `collectNodes`.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Polina Bombina <pbombina@augusta.edu>

Examples

```
xmlfile <- system.file("pathways/WP3850.gpml", package = "WayFindR")
xmldoc <- XML::xmlParseDoc(xmlfile)
edges <- collectEdges(xmldoc)
nodes <- collectNodes(xmldoc)
anchors <- collectAnchors(xmldoc)
labels <- collectLabels(xmldoc)
edges <- collectShapes(xmldoc)
groups <- collectGroups(xmldoc, nodes)
```

GPMLtoIgraph   *Converting GPML FIles to Igraph Objects*

Description

Takes a GPML file from WikiPathways, extracts the entities therein, and makes minor adjustments necessary to convert it into an [igraph](#) object. Along the way, it assigns a consistent set of colors, line types, and shapes.

Usage

```
GPMLtoIgraph(xmldoc, returnLists = FALSE, debug = FALSE)
nodeLegend(x, graph)
edgeLegend(x, graph)
```

Arguments

`xmldoc`	Either the name of an XML file meeting the specifications of the Genomic Pathway Markup Language (GPML), or an object of class `XMLInternalDocument` obtained by running such a file through the `xmlParseDoc` function of the `XML` package.
`returnLists`	A logical value; should the return value include the node list and edge list matrices?

debug	A logical value; should debugging progress information be printed? Probably best to leave it equal to FALSE.
x	A character string, such as "topleft" indicating where to place the legend.
graph	An `igraph` object as produced by the function`GPMLtoIgraph`.

Details

`GPMLtoIgraph` is the main function of the `WayFindR`package. It achieves the primary goal of converting pathways from the GPML file format to a mathematical graph, in the format defined by the igraph package, at which point we can apply a wide variety of graph algorithms from computer science in order to "compute on biological pathways".

The implementation of `GPMLtoIgraph` relies on the utility functions described in gpml-utility. Briefly, the algorithm starts by collecting all nodes (`DataNodes` in GPML) and edges (`Interactions` in GPML) from the GPML input file. However, GPML includes two other structures with (semantic) biological meaning. First, the GPML description includes the idea of an (invisible) "Anchor" that allows one edge to point to another edge. We expand those invisible target locations into full-fledged nodes in the final graph. Second, GPML includes "Groups" that represent protein complexes or sets of closely related genes. In `WayFindR`, we represent such groups as their own nodes in the final graph, and add "contained" edges linking in the group members. The transformations of Anchors and Groups do not change the fnudamental topology (in particular, the existence of cycles) of the resulting graph.

Finally, GPML includes non-semantic features (including "Labels" and "Shapes") that are (mis)used by some pathway authors as the targets of edges. `WayFindR` converts any targeted non-semantic features into nodes in order to preserve as much information as possible from the original pathway in WikiPathways.

Value

The `GMLtoIgraph` function usually returns an `igraph` object that represents the pathway defined by the input `xmlfile`. If the argument `returnLists = TRUE`, then it returns a list containing three components; `graph` is the `igraph` object, `nodes` is a data frame containing node information where each row is a node, and `edges` is a matrix containing edge information where each row is an edge. The node and edge information can be used to reproduce the graph in any network or graph visualization tool that accepts such matrices to describe the graph. The `nodes` data frame includes columns for `color` and `shape`, and the `edges` data frame includes columns for `color` and `lty` that are recognized and used by the plot.igraph function.

Both `nodeLegend` and `edgeLegend` invisibly return the same value that is returned by the legend function that is used in the implementation.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Polina Bombina <pbombina@augusta.edu>

Examples

```
xmlfile <- system.file("pathways/WP3850.gpml", package = "WayFindR")
graf <- GPMLtoIgraph(xmlfile)
set.seed(13579)
```

```
L <- igraph::layout_with_graphopt
plot(graf, layout=L)
nodeLegend("topleft", graf)
edgeLegend("bottomright", graf)
```

Index