Package ‘SBGNview’

April 15, 2020

Title Overlay omics data onto SBGN pathway diagram

Description SBGNview is an R package for visualizing omics data on SBGN pathway maps. Given omics data and a SBGN-ML file with layout information, SBGNview can display omics data as colors on glyphs and output image files. SBGNview provides extensive options to control glyph and edge features (e.g. color, line width etc.). To facilitate pathway based analysis, SBGNview also provides functions to extract molecule sets from SBGN-ML files. SBGNview can map a large collection of gene, protein and compound ID typs to glyphs.

Version 1.0.0

Author Xiaoxi Dong, Weijun Luo

Maintainer Xiaoxi Dong <dfdongxiaoxi@gmail.com>

Depends R (>= 3.6), pathview, SBGNview.data

Imports Rdpack, grDevices, methods, stats, utils, xml2, rsvg, igraph, rmarkdown, knitr, SummarizedExperiment, AnnotationDbi

RdMacros Rdpack

License AGPL-3

LazyData FALSE

Collate SBGN.to.SVG.R parsing.utilities.R plot.utilities.R data.R SBGNview.R SBGNview.obj.fun.R col.panel.R highlight.utilities.R

RoxygenNote 6.1.1

Suggests testthat, gage

VignetteBuilder knitr

Encoding UTF-8

biocViews GeneTarget, Pathways, GraphAndNetwork, Visualization, GeneSetEnrichment, DifferentialExpression, GeneExpression, Microarray, RNASeq, Genetics, Metabolomics, Proteomics, SystemsBiology, Sequencing, GeneTarget

URL https://github.com/datapplab/SBGNview

BugReports https://github.com/datapplab/SBGNview/issues

git_url https://git.bioconductor.org/packages/SBGNview

git_branch RELEASE_3_10

git_last_commit 8c59137

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

+.SBGNview . . . 2
arc-class . . . 3
changeDataId . . . 4
changeIds . . . 5
downloadSbgnFile . . . 6
findPathways . . . 7
getMolList . . . 8
glyph-class . . . 9
highlightArcs . . . 11
highlightNodes . . . 12
highlightPath . . . 13
loadMappingTable . . . 14
mapped.ids . . . 15
outputFile . . . 16
outputFile<- . . . 16
pathways.info . . . 17
pathways.stat . . . 17
print.SBGNview . . . 18
renderSbgn . . . 18
sbgnNodes . . . 23
SBGNview . . . 24
SBGNview.obj . . . 29
spline.arc-class . . . 29

Index **31**

`+.SBGNview` *Generic function to modify SBGN graph features*

Description

This binary operator ("+") modifies a SBGNview object(first argument) using a function (second argument)

Usage

```
## S3 method for class 'SBGNview'
SBGNview.obj + fn
```

Arguments

`SBGNview.obj`	An object of class SBGNview
`fn`	A character string. The name of any funtion that modifies and returns a SBGNview object. Some functions are available in SBGNview package: `highlightPath`, `highlightNodes`.

Value

The returned value of *fn*

Examples

```
data(SBGNview.obj)
obj.new = SBGNview.obj +
            highlightArcs(class = "production",color = "red")
```

`arc-class` *An object to store arc information.*

Description

An object to store arc information.

Details

Arc information comes from two sources:1. SBGN-ML file's "arc" element ("source", "target", coordinates etc.). 2. Parameters specified when running `SBGNview`. User can modify arc objects to change the way how it is rendered. Some of the slots/parameters in arc object controls similar graph features such as text font, line width/color etc.

The priority is determined in the following sequence:

1. arc@edge
2. Parameters in slot "arc@global.parameters.list"
3. Slots in arc object: stroke.opacity, fill.color etc.

Slots

`target,source,id,arc.class` A character string. Information extracted from element "arc"

`start.x,start.y,end.x,end.y` Numeric. Information extracted from elements "start", "end" or "next"

`stroke.opacity` Numeric. Controls the line of an arc (not tip of arc).

`global.parameters.list` A list. This is a copy of parameters in '...' of SBGNview. It can affect glyph behavior if the element has a corresponding feature of the arc. For example "edge.tip.size" affects the arc but "compartment.opacity" doesn't. See details for more information. This will be improved in future versions to remove unrelated parameters.

`edge` A list. An arc in SBGN map normally consists of a line and a tip shape at the end of the line. This list holds variables that controls arc properties. Available elements can be accessed as follow:

The following three paremeters control the line.
arc@edge$line.stroke.color
arc@edge$line.stroke.opacity
arc@edge$line.width

The following five paremeters control the tip.
arc@edge$tip.stroke.opacity
arc@edge$tip.stroke.color
arc@edge$tip.stroke.width
arc@edge$tip.fill.color
arc@edge$tip.fill.opacity

`changeDataId` *Change the data IDs of input omics data*

Description

This function changes the IDs of input omics data from one type to another.

Usage

```
changeDataId(data.input.id, input.type, output.type, sum.method = "sum",
  org = "hsa", cpd.or.gene, id.mapping.table = NULL,
  SBGNview.data.folder = "./SBGNview.tmp.data")
```

Arguments

`data.input.id` A matrix. Input omics data. Rows are genes or compounds, columns are measurements. Row names are original IDs that need to be transformed.

`input.type` A character string. The type of input IDs. Please check `data("mapped.ids")` for supported types.

`output.type` A character string. The type of output IDs. Please check `data("mapped.ids")` for supported types.

`sum.method` A character string. In some cases multiple input IDs are mapped to one output ID. In this situation ,we may need to derive only one value from them. This parameter is a function that can derive a single numeric value from a vector of numeric values (e.g. "sum","max","min","mean"), including a User Defined Function (UDF).

`org` A character string. The species source of omics data. "changeDataId" uses pathview to map between some gene ID types. Please use "?geneannot.map" to check the detail. Pathview needs species information to do the job. This parameter is a two-letter abbreviation of organism name, or KEGG species code, or the common species name, used to determine the gene annotation package. For all potential values check: data(bods); bods. Default org="Hs", and can also be "hsa" or "human" (case insensitive).

`cpd.or.gene` A character string. Either "compound" or "gene" – the type of input omics data.

`id.mapping.table` A matrix. Mapping table between input.type and output.type. This matrix should have two columns for input.type and output.type, respectively. Column names should be the values of parameters "input.type" and "output.type". See example section for an example.

`SBGNview.data.folder` A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

Details

This function maps between various gene/compound ID types.

1. Map other ID types to glyph IDs in SBGN-ML files of pathwayCommons database,and MetaCyc database: Use output.type = "pathwayCommons" or output.type = "metacyc.SBGN". Please check `data("mapped.ids")` for supported input ID types.

2. Map between other ID types:

2.1 ID types pairs can be mapped by pathview. Currently SBGNview uses pathview to do this mapping. Please check pathview functions "geneannot.map" and "cpdidmap" for more details.

2.2 Other ID type pairs

In this case, users need to provide id.mapping.table.

Value

A matrix, row names are changed to IDs of "output.type". Note the number of rows may be different from input matrix, because multiple input IDs could be collapsed to a single output ID. Also a single input ID could map to multiple output IDs.

Examples

```
# Change gene ID
data(mapped.ids)
library(pathview)
data("gse16873.d")
gene.data = gse16873.d[c("7157","1032"),]
mapping.table = data.frame(
                   ENTREZID = c("7157","1032")
                   ,SYMBOL = c("TP53","CDKN2D")
                   ,stringsAsFactors=FALSE
)
new.dt = changeDataId(
      data.input.id = gene.data,
      output.type="SYMBOL",
      input.type="ENTREZID",
      cpd.or.gene = "gene",
      id.mapping.table = mapping.table
      )
```

---

changeIds *Change input IDs to another ID type*

Description

Change input IDs to another ID type

Usage

```
changeIds(input.ids, input.type, output.type, cpd.or.gene,
  limit.to.pathways = NULL, org = "hsa",
  SBGNview.data.folder = "./SBGNview.tmp.data")
```

Arguments

`input.ids` A vector of character strings. Input IDs that need to be converted.

`input.type` A character string. The type of input IDs. Supported ID types can be found in data(mapped.ids)

`output.type` A character string. The type of output IDs. Supported ID types can be found in data(mapped.ids)

`cpd.or.gene` A character string. One of "gene" or "compound". Currently only supports macromolecule and simple chemical IDs. This parameter is used to find the ID mapping between input node set IDs and glyph IDs.

`limit.to.pathways` A vector of character strings. A vector of pathways IDs. When *output.type* is one of "pathwayCommons" or "metacyc.SBGN", one input ID(e.g. gene symbol) can map to multiple nodes in different pathways (SBGN-ML files). In this case, we can limit output to the specified pathways. When *output.type* is NOT one of "pathwayCommons" or "metacyc.SBGN", this argument is ignored.

`org` Character string. Three letter KEGG species code.

`SBGNview.data.folder` A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

Value

A list. Each element is the IDs in output.type that are mapped to one input ID.

Examples

```
 data(mapped.ids)
mapping = changeIds(
  input.ids = c(100048912),
  input.type = "ENTREZID",
  output.type = "pathwayCommons",
  cpd.or.gene = "gene",
)
```

---

downloadSbgnFile *Download pre-generated SBGN-ML file from GitHub*

Description

This function can download our pre-generated SBGN-ML files from GitHub repository.

Usage

```
downloadSbgnFile(pathway.id, download.folder = ".")
```

Arguments

`pathway.id` The ID of pathway. It is a pathway database specific ID. For accepted pathway IDs, please check `data("pathways.info")`. IDs are in column "pathway.id" (pathways.info[,"pathway.id"])

`download.folder` The output folder to store downloaded SBGN-ML files.

Value

A character string. The path to the downloaded SBGN-ML file.

Examples

```
data("pathways.info")
data(sbgn.xmls)
input.sbgn = downloadSbgnFile(
                  pathway.id = pathways.info[1,"pathway.id"],
                  download.folder = "./")
```

---

findPathways *Retrieve pathways from keywords*

Description

This function searches for pathways by input keywords.

Usage

```
findPathways(keywords = NULL, keywords.logic = "or",
  keyword.type = "pathway.name", org = NULL,
  pathway.completeness = NULL, mol.name.match = c("exact.match",
  "jaccard", "presence.of.input-string.in.target-name")[1],
  SBGNview.data.folder = "./SBGNview.tmp.data/")
```

Arguments

`keywords` A character string or vector. The search is case in-sensitive.

`keywords.logic` A character string. Options are "and" or "or". This will tell the function if the search require "all" or "any" of the keywords to be present. It only makes difference when keyword.type is "pathway.name".

`keyword.type` A character string. Either "pathway.name" or one of the ID types in `data("mapped.ids")`

`org` A character string. The KEGG species code.

`pathway.completeness` Numeric. The returned pathways need to meet this criteria: completeness in the species(parameter "org") should be larger than this value. Currently it only applies to "pathwayCommons" pathways. Because pathwayCommons only annotated human pathways, we mapped pathwayCommons' nodes to other species using KEGG ortholog annotation. As a result, not all of the nodes have corresponding genes in another species. We call the percentage of mapped nodes the "coverage or completeness" in the species. If "pathway.completeness" is

provided, the function will use this single cutoff for different pathways. If it is not provided, the function will use pre-generated pathway-specific completeness cutoff, that is: use different cutoffs for different pathways. This cutoff is selected using the following approach: 1. A pathway has different completeness in different species thus form a completeness vector across all species (vector C) . 2. Use a completeness cutoff we can define whether this pathway "exists" in a species, thus form a label vector E (a pathway "Exist" or "not Exist" across all species). 3. Use one way ANOVA to calculate F statistic of completeness between the two groups ("Exist" or "not Exist"), thus one cutoff will have one F statistic. 4. Try different cutoffs(unique completeness values in vector C) and select the one with the largest F statistic, i.e. the cutoff the can maximize the difference between Exist" and "not Exist" groups.

`mol.name.match` A character string. How to match molecular names.

`SBGNview.data.folder`
A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

Details

If "keyword.type" is "pathway.name" (default), this function will search for the presence of any keyword in the pathway.name column of data(pathways.info). The search is case in-sensitive. If "keyword.type" is one of the identifier types and "keywords" are corresponding identifiers, this function will return pathways that include nodes mapped to input identifiers. "org" and "pathway.completeness" are used to filter pathways by their completeness is a species. We mapped KEGG ortholog proteins from other species to pathwayCommons' human pathway nodes. Therefore, each pathway has different coverage or completeness in another species. When omics gene data is from a paticular species, "pathway.completeness" can give some information about how confident this pathway exists in the species.

Value

A dataframe. Contains information of pathways found.

Examples

```
data(pathways.info)
input.pathways <- findPathways("Adrenaline and noradrenaline biosynthesis")
```

`getMolList` *Retrieve gene list or compound list from collected databases*

Description

Retrieve gene list or compound list from collected databases

Usage

```
getMolList(database = "pathwayCommons", mol.list.ID.type = "ENTREZID",
  org = "hsa", cpd.or.gene = "gene", output.pathway.name = TRUE,
  combine.duplicated.set = TRUE, truncate.name.length = 50,
  SBGNview.data.folder = "./SBGNview.tmp.data")
```

Arguments

`database` Character string. The database where gene list will be extracted. Acceptable values: "MetaCyc", "pathwayCommons", "MetaCyc". The value is case insensitive.

`mol.list.ID.type` Character string. The ID type of output gene list. One of the supported types in `data("mapped.ids")`

`org` Character string. The three letter species code used by KEGG. E.g. "hsa","mmu"

`cpd.or.gene` Character string. One of "gene" or "compound"

`output.pathway.name` Logical. If set to "TRUE", the names of returned list are in the format: "pathway.id::pathway.name". If set to "FALSE", the format is "pahtway.id"

`combine.duplicated.set` Logical. Some pathways have the same geneset. If this parameter is set to "TRUE", the output list will combine pathways that have the same gene set. The name in the list will be pathway names concatinated with "||"

`truncate.name.length` Integer. The pathway names will be truncated to at most that length.

`SBGNview.data.folder` A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

Value

A list. Each element is a genelist of a pathway.

Examples

```
data(pathways.info)
mol.list <- getMolList(
                database = "pathwayCommons",
                mol.list.ID.type = "ENTREZID",
                org = "hsa"
)
```

---

glyph-class *An object to store glyph information*

Description

An object to store glyph information

Details

User can modify glyph objects to change the way how it is rendered.

Some of the slots/parameters in a glyph object control similar graph features (e.g. text font, line width/color etc). The priority is determined in the following sequence:

1. glyph@text, glyph@shape.

2. Fill color derived from omics data

3. Parameters in slot "glyph@global.parameters.list"

4. Slots in glyph object: stroke.opacity, fill.color etc.

For example, if both of the following parameters are set:

1). glyph@text$font.size = 5

2). lobal.parameters.list$font.size = 10

Then the font size is set by "glyph@text$font.size = 5".

Slots

`compartment` A character string. The compartment this glyph belongs to.

`x,y,h,w` Numeric. The x,y location and hight, width of the glyph

`id` A character string. The ID of the glyph (determined by the "id" attribute in element "glyph")

`label` A character string. label of the glyph. Extracted from the element "label"

`glyph.class` A character string. Class of the glyph

`user.data` A numeric vector. The omics data mapped to this glyph.

`stroke.width,stroke.opacity,fill.opacity` Numeric. Controls glyph border width or opacity of node fill and border.

`fill.color,stroke.color` A character string. The fill color and stroke color of glyph.

`label_location` A character string. One of "center" or "top". If set to "center", the glyph label text will be displayed at the center of the glyph. If set to "top", the label will be displayed at the top of the glyph.

`orientation` A character string. One of "left","right","up","down". This is specific to glyphs of classes "terminal" and "tag". It will change the orientation of the node.

`global.parameters.list` A list. This is a copy of parameters in '...' of function `SBGNview`.

`shape` A list. Parameters controlling how the node is rendered. Available elements can be accessed as follow:

glyph@shape$stroke.color
-Character (e.g. "red"). The color of glyph border.

glyph@shape$stroke.opacity
-Numeric (e.g. 0.8). The opacity of glyph border. Must be between 0 and 1.

glyph@shape$stroke.width
-Numeric. The width of glyph border

glyph@shape$fill.opacity
-Numeric. The opacity of glyph fill. Must be between 0 and 1.

glyph@shape$fill.color
-Numeric. The glyph fill color. Note that this will overwrite colors generated from omics data!

`text` A list. Parameters controlling how node label is rendered. Available elements can be accessed as follow:

glyph@text$y.shift
-Numeric. Move glyph label vertically by this value.

glyph@text$x.shift
-Numeric. Move glyph label horizontally by this value.

glyph@text$color
-A character string. The color of glyph label.

glyph@text$font.size
-A character string. The font size of glyph label.

`highlightArcs` *Highlight arcs by arc class*

Description

This function creates another function that can modify a SBGNview object and modify arc. Normally we use it as an argument to `+.SBGNview` and modify a SBGNview object

Usage

```
highlightArcs(class = "all", color = "black", line.width = 2,
  tip.size = 6)
```

Arguments

`class`	String. The arc class to modify.
`color`	String.
`line.width`	Numeric.
`tip.size`	Numeric

Value

A function that modify a SBGNview object to change arc color.

Examples

```
data(SBGNview.obj)
obj.new = SBGNview.obj +
            highlightArcs(class = "production",color = "red")
```

highlightNodes *Highlight input nodes*

Description

Change node properties such as border color and width to highlight a list of input nodes. This function should be used as the second argument to function `+.SBGNview`.

Usage

```
highlightNodes(node.set = "all", node.set.id.type = "CompoundName",
  glyphs.id.type = "pathwayCommons", cpd.or.gene = "compound",
  stroke.color = "black", stroke.width = 10, stroke.opacity = 1,
  show.glyph.id = FALSE, select.glyph.class = c(), label.x.shift = 0,
  label.y.shift = 0, label.color = "black", label.font.size = NULL,
  label.spliting.string = NULL, labels = NULL)
```

Arguments

`node.set`	A vector of character strings. Input node set that need to be highlighted. It can be any ID types supported by SBGNview.
`node.set.id.type`	A character string. ID type of input nodes.
`glyphs.id.type`	A character string. ID type of nodes in SBGN-ML file, which is used as node IDs in the SBGNview object returned by the SBGNview() function.
`cpd.or.gene`	A character string. One of "gene" or "compound". Currently highlight.node() only supports highlighting macromolecule and simple chemical nodes. This parameter is used to find the ID mapping between input node set IDs and glyph IDs.
`stroke.color`	A character string. The vorder color of highlighted nodes.
`stroke.width`	A character string. The border width of highlighted nodes.
`stroke.opacity`	Numeric
`show.glyph.id`	Logic
`select.glyph.class`	Character
`label.x.shift`	Numeric
`label.y.shift`	Numeric
`label.color`	Character
`label.font.size`	Numeric
`label.spliting.string`	Character
`labels`	Character

Value

A SBGNview obj

Examples

```
data(SBGNview.obj)
obj.new = SBGNview.obj +
       highlightNodes(node.set = c("tyrosine", "(+-)-epinephrine"),
                      stroke.width = 4, stroke.color = "green")
```

`highlightPath` *Given two nodes, find the shortest path between them and highlight the path. Other molecules/nodes and edges involved in reactions in the path are also highlighted.*

Description

Given two nodes, find the shortest path between them and highlight the path. Other molecules/nodes and edges involved in reactions in the path are also highlighted.

Usage

```
highlightPath(from.node = NULL, to.node = NULL,
  directed.graph = TRUE, node.set.id.type = "CompoundName",
  glyphs.id.type = "pathwayCommons", cpd.or.gene = "compound",
  input.node.stroke.width = 10, from.node.color = "red",
  to.node.color = "blue", path.node.color = "black",
  path.node.stroke.width = 10, n.shortest.paths.plot = 1,
  shortest.paths.cols = c("purple"), path.stroke.width = 2,
  tip.size = 4)
```

Arguments

`from.node` A character string. The ID of source node.

`to.node` A character string. The ID of target node.

`directed.graph` Logical. If treat the SBGN map as a directed graph. If treat it as directed graph, the direction of an arc is determined by the <arc> XML element in the input SBGN-ML file: from "source" node to "target" node.

`node.set.id.type` A character string. ID type of input nodes.

`glyphs.id.type` A character string. ID type of nodes in SBGN-ML file, which is used as node IDs in the SBGNview object returned by the SBGNview() function.

`cpd.or.gene` A character string. One of "gene" or "compound". Currently highlight.node() only supports highlighting macromolecule and simple chemical nodes. This parameter is used to find the ID mapping between input node set IDs and glyph IDs.

`input.node.stroke.width` A character string. Border stroke width of input nodes (affects both from.node and to.node)

`from.node.color` A character string. Color of "source"/from.node

`to.node.color` A character string. Color of "target"/to.node

`path.node.color`
: String

`path.node.stroke.width`
: Numeric

`n.shortest.paths.plot`
: Integer. There could be more than one shortest paths between two nodes. User can choose how many of them to plot.

`shortest.paths.cols`
: A character string. The colors of arcs in different shortest paths. The length of this vector (number of colors) should be the value of n.shortest.paths.plot. If one arc is in multiple shortest paths, its color will be set to the color that appears later in the vector.

`path.stroke.width`
: Numeric. The width of line in edges in the shortest paths.

`tip.size`
: Numeric. The size of arc tips in edges of the shortest paths.

Value

A SBGNview obj.

Examples

```
 data(SBGNview.obj)
## Not run:
data("SBGNview.obj" )
obj.new = SBGNview.obj +
              highlightPath(from.node = c("tyrosine")
                 , to.node = c("dopamine")
                ,from.node.color = "red"
                ,to.node.color = "blue"
  )

## End(Not run)
```

`loadMappingTable` *Generate ID mapping table from input and output ID types. If provided a vector of input IDs (limit.to.ids argument), the function will output mapping table only containing the input IDs. Otherwise, the function will output all IDs of input and output types (restricted to a species if working on gene types and specified the "species" parameter).*

Description

Generate ID mapping table from input and output ID types. If provided a vector of input IDs (limit.to.ids argument), the function will output mapping table only containing the input IDs. Otherwise, the function will output all IDs of input and output types (restricted to a species if working on gene types and specified the "species" parameter).

Usage

```
loadMappingTable(output.type, input.type, species = NULL, cpd.or.gene,
  limit.to.ids = NULL, SBGNview.data.folder = "./SBGNview.tmp.data")
```

Arguments

- `output.type` A character string. Gene or compound ID type
- `input.type` A character string. Gene or compound ID type
- `species` A character string. Three letter KEGG species code.
- `cpd.or.gene` A character string. Either "gene" or "compound"
- `limit.to.ids` vector. Molecule IDs of "input.type".
- `SBGNview.data.folder` A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

Value

A list containing the mapping table.

Examples

```
data(mapped.ids)
entrez.to.pathwayCommons = loadMappingTable(
                                input.type = "ENTREZID"
                                ,output.type = "pathwayCommons"
                                ,species = "hsa"
                                ,cpd.or.gene = "gene"
                              )
```

`mapped.ids` *IDs mappable by SBGNview*

Description

IDs mappable by SBGNview

Usage

```
mapped.ids
```

Format

A list with two elements: A vector of mappable gene ID types, a vector of mappable compound ID types.

Details

ID types mappable to glyph IDs in SBGNhub SBGN-ML file collection: https://github.com/datapplab/SBGNhub/tree/ma

outputFile — *Retrieve output file information from a SBGNview object*

Description

Retrieve output file information from a SBGNview object

Usage

```
outputFile(obj)
```

Arguments

obj	A SBGNview object.

Details

This function prints the output file path recorded in a SBGNview object. When printing the SBGNview object, output image files will be generated using the output file.

Value

A string. The output file information. The same as parameter "output.file" when running SBGNview

Examples

```
data("SBGNview.obj" )
outputFile(SBGNview.obj)
```

outputFile<- — *Set output file information for a SBGNview object*

Description

Set output file information for a SBGNview object

Usage

```
outputFile(obj) <- value
```

Arguments

obj	No need to provide
value	No need to provide

Details

This function sets the output file path that can be recorded in a SBGNview object. When printing the SBGNview object, output image files will be generated using the output file.

Value

A SBGNview object

Examples

```
data("SBGNview.obj" )
outputFile(SBGNview.obj) = "./test.output"
```

---

pathways.info *Information of collected pathways*

Description

Information of collected pathways

Usage

```
pathways.info
```

Format

A data.frame

Details

The information of pre-collected pathways (SBGNhub: https://github.com/datapplab/SBGNhub/tree/master/data/SBGN. and their SBGN-ML file information, such as pathway ID, name, source database, glyph ID types and URLs to the original pathway webpages.

---

pathways.stat *Number of pathways collected*

Description

Number of pathways collected

Usage

```
pathways.stat
```

Format

A data.frame

Details

The number of pathways in SBGNhub from each source database. It is calculated from data 'pathways.info'.

print.SBGNview — *A wrapper to run function renderSbgn for all pathways in a SBGNview object and generate image files.*

Description

A wrapper to run function renderSbgn for all pathways in a SBGNview object and generate image files.

Usage

```
## S3 method for class 'SBGNview'
print(x, ...)
```

Arguments

x	An object of class SBGNview
...	Other parameters passed to print.

Value

None

Examples

```
### use simulated data. Please see vignettes for more examples
data("pathways.info","sbgn.xmls")
SBGNview.obj = SBGNview(
              simulate.data = TRUE
              ,sbgn.dir = "./"
              ,input.sbgn = "P00001"

              ,output.file = "./test.local.file"
              ,output.formats = c("pdf")

              ,min.gene.value = -1
              ,max.gene.value = 1
            )
 print(SBGNview.obj)
```

renderSbgn — *Overlay omics data on a SBGN pathway graph and output image files.*

Description

This function is not intended to be used directly. Use SBGNview instead. Some input arguments can be better prepared by SBGNview.

Usage

```
renderSbgn(input.sbgn, output.file, if.write.files = TRUE,
  output.formats, sbgn.id.attr, glyphs.user = list(),
  arcs.user = list(), arcs.info = "straight",
  compartment.layer.info = "original",
  user.data = matrix("no.user.data", nrow = 1), if.plot.svg = TRUE,
  key.pos = "topright", color.panel.scale = 0.7,
  color.panel.n.grid = 21, col.gene.low = "green",
  col.gene.high = "red", col.gene.mid = "gray", col.cpd.low = "blue",
  col.cpd.high = "yellow", col.cpd.mid = "gray", min.gene.value = -1,
  max.gene.value = 1, mid.gene.value = 0, min.cpd.value = -1,
  max.cpd.value = 1, mid.cpd.value = 0, pathway.name = "",
  pathway.name.font.size = 1, if.plot.cardinality = FALSE,
  multimer.margin = 5, compartment.opacity = 1,
  auxiliary.opacity = 1, if.plot.annotation.nodes = FALSE,
  inhibition.edge.end.shift = 5, edge.tip.size = 6,
  if.use.number.for.long.label = FALSE, label.spliting.string = c(" ",
  ":", "-", ";", "/", "_"), complex.compartment.label.margin = 8,
  if.write.shorter.label.mapping = TRUE, font.size = 3,
  logic.node.font.scale = 3, status.node.font.scale = 3,
  node.width.adjust.factor = 2, font.size.scale.gene = 3,
  font.size.scale.cpd = 3, font.size.scale.complex = 1.1,
  font.size.scale.compartment = 1.6,
  if.scale.complex.font.size = FALSE,
  if.scale.compartment.font.size = FALSE,
  node.width.adjust.factor.compartment = 1,
  node.width.adjust.factor.complex = 3, text.length.factor = 2,
  text.length.factor.macromolecule = 2,
  text.length.factor.compartment = 2, text.length.factor.complex = 2,
  space.between.color.panel.and.entity = 100,
  global.parameters.list = NULL)
```

Arguments

`input.sbgn` A character string. The path to a local SBGN-ML file.

`output.file, output.formats, sbgn.id.attr`
These parameters are the same as the ones in `SBGNview`. Please see `SBGNview` for more details.

`if.write.files` Logical. If generate image files.

`glyphs.user` A list, optional. Each element is a "glyph" object. The element names are glyph IDs (attribute "id" of XHTML element "glyph"). Note this is not affected by parameter "sbgn.id.attr". The glyph elements contain glyph meta-data for plotting (e.g. text size, border width, border color etc.). Please see the glyph object documentation for more information. By default, SBGNview will run without this argument and return a glyph list extracted from the SBGN file. User can then customize this glyph list and assign it to "glyphs.user" in the next SBGNview run to update the graph.

`arcs.user` A list, optional. Each member is an "arc" object. The element names are arc IDs (the value of "id" attribute in XHTML element "arc" or "arc.spline" in the SBGN-ML file). Some SBGN-ML files have no arc IDs, in this case SBGNview will create an arc ID using "source" and "target" node IDs). The arc object

contains arc meta-data for plotting (e.g. arc line width, line color etc.). Please see the arc object documentation for more information. By default, SBGNview() will run without this argument and return an arc list. User can then customize this arc list and assign it to "arcs.user" in the next SBGNview() run to update the arcs.

`arcs.info` A character string. It should be one of the following: "parse splines", "straight" or a string of svg code of arcs. If it is "parse splines", this function will look for XML element "arc.spline" in the SBGN-ML file and plot spline arcs. If it is "straight", the function will look for element "arc" and plot straight line arcs. If it is a string of svg code, it will write this code directly to the output svg file.

`compartment.layer.info` A character vector. It is a vector containing the IDs of all compartment glyphs. It determins the layer arrangement of compartments. Compartments will be drawn following their sequence in this vector. Therefore, a compartment that appears later in the vector will be on the front layer and covers the compartments that are before it in this vector. This is important. In some cases compartments have overlap. A glyph laying in the overlapped region belongs to the compartment on the top layer.

`user.data` A matrix. It holds both gene/protein data and compound data. Rows are gene or compounds, columns are experiments. Row names are gene IDs or compound IDs. The ID types must be the same as the type of glyph ID in the SBGN-ML file.

`if.plot.svg` Logical. If generate svg or only parse SBGN-ML file.

`key.pos` A character string. -Default: "topright" One of "bottomleft" , "bottomright", "topright", "topleft". The location of color panel: lower left, lower right, upper right, upper left

`color.panel.scale` Numeric. -Default: 1.5 It controls the relative size of color scheme panel.

`color.panel.n.grid` Numeric. -Default: 101 How many colors does the color scheme show.

`col.gene.low` A character string. -Default: "green"

`col.gene.high` A character string. -Default: "red"

`col.gene.mid` A character string. -Default: "gray"

`col.cpd.low` A character string. -Default: "blue"

`col.cpd.high` A character string. -Default: "yellow"

`col.cpd.mid` A character string. -Default: "gray"

`min.gene.value` Numeric. -Default: -1 Color panel's min value for gene. Values smaller than this will have the same color as min.value.

`max.gene.value` Numeric. -Default: 1

`mid.gene.value` Numeric. - Default: 0

`min.cpd.value` Numeric. -Default: -1 Color panel's min value for compound. Values smaller than this will have the same color as min.value.

`max.cpd.value` Numeric. -Default: 1

`mid.cpd.value` Numeric. -Default: 0

`pathway.name` List Containing two elements: 1. pathway name 2. stamp information

`pathway.name.font.size` Numeric

`if.plot.cardinality`
: Logical. If plot cardinality glyphs.

`multimer.margin`
: Numeric. -Default: 5 For multimers, they are represented by two partly overlapped shapes (rectangle, ellipse etc.). This parameter controls how much the two shapes overlap.

`compartment.opacity`
: Numeric. -Default: 0.9 How transparent the compartments are.

`auxiliary.opacity`
: Numeric. -Default: 1 Opacity of auxiliary glyphs.

`if.plot.annotation.nodes`
: Logical. -Default: F Some SBGN-ML files have "annotation" glyphs. By default we don't plot them.

`inhibition.edge.end.shift`
: Numeric. -Default: 4 The tip of "inhibition" arcs is a line segment. Sometimes it overlaps with target glyph's border. We can shift it along the arc to prevent the overlap.

`edge.tip.size`
: Numeric. -Default: 4

`if.use.number.for.long.label`
: Logical. -Default: T If the label is too long, we can create a shorter name for it. e.g. "macromolecule_1".

`label.spliting.string`
: A character vector. -Default: c(" ","-",";","/","_") When we split text into multiple lines, these characters will be used to split label (where a new line can be created).

`complex.compartment.label.margin`
: -Default: 8 Move the label text vertically for compartment and complex.

`if.write.shorter.label.mapping`
: Logical. -Default: T If if.use.number.for.long.label is "T", we can write the mapping between shorter name and the original label to a text file.

`font.size`
: Numeric. -Default: 6 Affects font size of all types of glyphs.

`logic.node.font.scale`
: Numeric. -Default: 6 Controls the size of logical glyphs ("and", "or", "not" etc.).

`status.node.font.scale`
: Numeric. Scale the font size for status variable and unit of information nodes.

`node.width.adjust.factor`
: Numeric. -Default: 3 Change font size according to the width of its glyph. If the glyph is too large (e.g. a compartment), its label may look too small. Then we can enlarge the label in proportion to the width of the glyph. It affects all types of glyphs.

`font.size.scale.gene`
: Numeric. -Default: 3 Only affect font size of "macromolecule" glyphs.

`font.size.scale.cpd`
: Numeric. -Default: 3 Only affects font size of "simple chemical" glyphs.

`font.size.scale.complex`
: Numeric

`font.size.scale.compartment`
: Numeric

`if.scale.complex.font.size`
: Logical. -Default: T Whether we want to scale complex font size according to its width.

`if.scale.compartment.font.size`
: Logical. -Default: T Whether scale compartment font size according to its width.

`node.width.adjust.factor.compartment`
: Numeric. -Default: 1 How much the font size should change in proportion to the width of compartment.

`node.width.adjust.factor.complex`
: Numeric. -Default: 4 How much the font size should change in proportion to the width of complex.

`text.length.factor`
: Numeric. -Default: 2 How wide the wrapped text should be. If text is longer than the width controled by this parameter, the text is split into a new line, but only at characters in "label.spliting.string". Controls all glyphs.

`text.length.factor.macromolecule`
: Numeric. Controls macromolecles glyphs.

`text.length.factor.compartment`
: Numeric

`text.length.factor.complex`
: Numeric

`space.between.color.panel.and.entity`
: Numeric. -Default: 10 The minimum space between color panel and any other object in the graph. The function will always try to find a location of the color panel to minimize empty space on the whole graph.

`global.parameters.list`
: List. A record of parameters fed to "renderSbgn" for reuse. It will over-write other parameters. It is not designed to be used directly.

Value

A list of three elements: glyphs.list, arcs.list, global.parameters.list

Examples

```
 data(pathways.info)
## Not run:
SBGNview.obj = SBGNview(
              simulate.data = TRUE
              ,sbgn.dir = "./"
              ,input.sbgn = "P00001"

              ,output.file = "./test.local.file"
              ,output.formats = c("pdf")

              ,min.gene.value = -1
              ,max.gene.value = 1
            )

## End(Not run)
```

sbgnNodes *Extract glyph information*

Description

This function will extract node information such as complex members, compartment members, node class, nodes with state variables etc.

Usage

```
sbgnNodes(input.sbgn, output.gene.id.type = NA,
  output.cpd.id.type = NA, database = NA, species = NA,
  show.ids.for.multiHit = NULL,
  SBGNview.data.folder = "./SBGNview.tmp.data",
  sbgn.dir = "./SBGNview.tmp.data")
```

Arguments

`input.sbgn` A character string. required. The path to input SBGN-ML file.

`output.gene.id.type` A character string. The desired output gene ID type. It only works when the SBGN-ML file is from one of these databases: "MetaCyc" and "pathwayCommons". Currently, only "macromolecule" glyphs are supported. Please check `data("mapped.ids")` for the types accepted. If omitted, the output IDs are the oritinal IDs in the SBGN-ML file.

`output.cpd.id.type` A character string. The desired output compound ID type. It only works when the SBGN-ML file is from one of these databases: "MetaCyc" and "pathwayCommons". Currently, only "simple chemical" glyphs are supported. Please check `data("mapped.ids")` for the types accepted. If omitted, the output IDs are the oritinal IDs in the SBGN-ML file.

`database` A character string. If the SBGN-ML file is from one of these databases: "MetaCyc" and "pathwayCommons", this parameter should be set to the corresponding string. For these two databases, this function can output other ID types instead of the original IDs in the SBGN-ML files. Otherwise, the output IDs are the oritinal IDs in the "id" attribure in the "glyph" element.

`species` A character string. Only output IDs from this particular species. It only works when the SBGN-ML file is from one of these databases: "MetaCyc" and "pathwayCommons". Please check data("supported.species") for supported species. If omitted, the function will output IDs from all species.

`show.ids.for.multiHit` Vector

`SBGNview.data.folder` A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

`sbgn.dir` A character string. The path to a folder that will hold download SBGN-ML files, if the input.sbgn are IDs of pre-collected SBGN-ML files.

Details

The following glyph information is extracted: complex members, compartment members,submap members, node class, nodes with state variables, class of state variables, edges with cardinality, nodes with ports, "source and sink" nodes, process nodes.
When trying to output other ID types, sometimes multiple output IDs are mapped to one glyph. In this situation, the IDs are concatenated by "; " to represent the glyph.

Value

A list, containing extracted glyph information.

Examples

```
 data(mapped.ids)
 data(sbgn.xmls)
 data(pathways.info)
node.list = sbgnNodes(
                      input.sbgn = "P00001",
                      output.gene.id.type = "ENTREZID",
                      output.cpd.id.type = "CompoundName",
                      species = "hsa"
                    )
```

SBGNview — *Overlay omics data on SBGN pathway diagram and output image files.*

Description

This is the main function to map, integrate and render omics data on pathway graphs. Two inputs are needed: 1. A pathway file in SBGN-ML format and 2. gene and/or compound omics data. The function generates image file of a pathway graph with the omics data mapped to the glyphs and rendered as pseudo-colors. It is similar to pathview except the pathways are rendered with SBGN notation. In addition, users can control more graph properties including node/edge attributes. We collected SBGN-ML files from several pathway databases: Reactome, MetaCyc, MetaCrop, PANTHER and SMPDB. Given a vector of patway IDs, SBGNview can automatically download and use these SBGN-ML files. To map omics data to glyphs, user just needs to specify omics data ID types. When using user customized SBGN-ML files, users need to provide a mapping file from omics data's molecule IDs to SBGN-ML file's glyph IDs.

Usage

```
SBGNview(gene.data = NULL, cpd.data = NULL, simulate.data = FALSE,
  input.sbgn = NULL, sbgn.dir = ".", output.file = "./output.svg",
  node.sum = "sum", gene.id.type = NA, cpd.id.type = NA,
  sbgn.id.attr = "id", sbgn.gene.id.type = NULL,
  sbgn.cpd.id.type = NA, id.mapping.gene = NULL,
  id.mapping.cpd = NULL, org = "hsa", output.formats = c("svg"),
  show.pathway.name = FALSE,
  SBGNview.data.folder = "./SBGNview.tmp.data", ...)
```

Arguments

`gene.data` A matrix, vector or SummarizedExperiment object. The same as "gene.data" argument in package "pathview", it is either a vector (single measurement) or a matrix-like data (multiple measurements). If the data is a vector, the entries should be numeric and names of entries should be gene IDs. Matrix data structure has genes as rows and samples as columns. Row names should be gene IDs. Here gene ID is a generic concept, including multiple types: gene, transcript or protein. Default gene.data=NULL.

`cpd.data` A matrix, vector or SummarizedExperiment object. The same as "gene.data", excpet named with compound IDs. Default cpd.data=NULL.

`simulate.data` Logical. SBGNview can simulate a dataset. If set to true, SBGNview will simulate a gene data set and a compound dataset and user input "gene.data" and "cpd.data" are ignored.

`input.sbgn` A vector. Can be either names of local SBGN files or pathway IDs of our pre-collected pathways.

`sbgn.dir` A character string. The path to the folder that holds SBGN-ML files. If "input.sbgn" is a vector of pathway IDs in data "pathways.info", the SBGN-ML files will be downloaded into this folder.

`output.file` A character string. Path to the output image files. Because we often work with multiple pathways, each pathway will have its own image files. Each string in "input.sbgn" will be added to the end of "output.file". Depending on the image format specified by the "output.formats" parameter, extentions will be added to the end (e.g. .pdf, .png etc.).

`node.sum` A character string. Sometimes multiple omics genes/compounds are mapped to one SBGN glyph. Therefore multiple values will be mapped to one measurement/slice on the glyph. In this situation ,we may need to derive a single value for the slice on the glyph. This function can be any R function that takes a numeric vector as input and output a single numeric value (e.g. "sum","max","min","mean"). It can also be a User Defined Function (UDF).

`gene.id.type` A character string. The type of gene ID in "gene.data". This parameter is used for ID mapping. It should be one of the IDs in data "mapped.ids". For details, run: `data("mapped.ids")`

`cpd.id.type` A character string. The type of compound ID in "cpd.data". For details, run: `data("mapped.ids")`

`sbgn.id.attr` A character string. This tells SBGNview where to find the ID of a glyph in SBGN-ML file for ID mapping. This ID is used to map omics data to the glyph. It is normally the name of an attribute in the "glyph" element . For example : <glyph class="macromolecule" id="p53"> </glyph>. We can specify: sbgn.id.attr = "id"; sbgn.gene.id.type = "SYMBOL". For our pre-generated SBGN-ML files, the ID attribute will be determined automatically thus can be omitted.

Accepted values:

1. Any attribute name in element "glyph" For example : <glyph class="macromolecule" id="p53" protein="P04637"> </glyph>. We can specify: sbgn.id.attr = "protein"; sbgn.gene.id.type = "UNIPROT", then "P04637" will be the glyph ID.

2. The string "label", this will make SBGNview use the glyph label as glyph ID. For example: <glyph id="glyph14" class="simple chemical"> <label text="L-alanine"/> </glyph>. We can specify: sbgn.id.attr = "label"; sbgn.gene.id.type = "CompoundName", then "L-alanine" will be used as glyph ID.

`sbgn.gene.id.type`
: A character string. The type of gene ID in SBGN-ML file (recorded in the "id" attribute in XML tag "glyph" of "macromolecule"s. See parameter "sbgn.id.attr" for more details). This parameter is used for ID mapping,i.e. either use our pre-generated mapping tables or find corresponding columns in user defined mapping tables in "id.mapping.gene". For our pre-generated SBGN-ML files, this will be determined automatically according to the pathway IDs thus can be omitted. For user defined SBGN-ML file, this parameter should be one of the column names of the matrix "id.mapping.gene".

`sbgn.cpd.id.type`
: A character string. See "sbgn.gene.id.type". The compond IDs are recorded in the "id" attribute in XML tag "glyph" of "simple chemical"s. See "sbgn.id.attr" for more details.

`id.mapping.gene`
: A matrix. Mapping table between gene.id.type and sbgn.gene.id.type. This table is needed if the ID pair of gene.id.type and sbgn.gene.id.type is NOT included in data "mapped.ids" or not mappable by package "pathview". This matrix should have two columns for gene.id.type and sbgn.gene.id.type, respectively. Column names should be the values of parameters "sbgn.gene.id.type" and "gene.id.type". See example section for an example.

`id.mapping.cpd`
: A matrix. See id.mapping.gene.

`org`
: A character string. The species of the gene omics data. It is used for species specific gene ID mapping. Currently only supports three letters KEGG code (e.g. hsa, mmu, ath). For a complete list of KEGG codes, see this page: https://www.genome.jp/kegg/catalog/org_list.html

`output.formats`
: A vector. It specifies the formats of output image files. The vector should be a subset of c("pdf" , "ps", "png"). By default the function will always output a svg file. SBGNview uses rsvg to convert svg file to other formats. If other "output.formats" is set but "rsvg" package is not installed, an error will occur. See this page for how to install "rsvg": https://github.com/jeroen/rsvg

`show.pathway.name`
: Logical. If set to TRUE and "input.sbgn" are pre-collected pathway IDs, the pathway name will be added to the output file.

`SBGNview.data.folder`
: A character string. The path to a folder that will hold download ID mapping files and pathway information data files. The data can be reused once downloaded.

`...`
: Other parameters passed to function `renderSbgn`, including:

Details

1. About SBGNview()

This function extracts glyph (node) and arc (edge) data from a SBGN-ML file and creates a SBGN graph from the extracted data (draws shapes etc. in SVG format). Then it maps omics data to the glyphs and renders data as colors. Currently it maps gene/protein omics data to "macromolecule" glyphs and maps compound omics data to "simple chemical" glyphs.

2. About SBGN-ML files and curved arcs encoding

SBGNview can parse SBGN-ML files with standard SBGN PD syntax. For arcs, SBGNview can handle both straight lines (standard SBGN PD syntax) and spline curves (a syntax add by us). Current SBGN-ML syntax supports straight lines. The coordinates of line start or end points are stored in element "arc". But straight lines often produce node-edge or edge-edge crossings. Therefore, we generated SBGN-ML files with pre-routed spline edges.

To store the routed splines, we added an XHTML element called "edge.spline.info", which has children elements called "arc.spline" . Each "arc.spline" element has three types of children elements: "source.arc", "target.arc" and "spline". "source.arc" and "target.arc" will be rendered as straight line arcs controlled by attributes "start.x","start.y", "end.x", "end.y" (line ends' coordinates) and "class" (type of the straight line arc). These two arcs ensure the notation of the spline arc comply with its class. "spline" will be rendered as splines connecting "source.arc" and "target.arc". Each "spline" is defined by coordinates of four points: s (starting point of spline), c1 (the first control point of spline), c2 (the second control point of spline) and e (ending point of spline). In case of complicated routing, there could be multiple "splines" in an "arc.spline".

The function first checks if the SBGN-ML file has spline arcs (XHTML element "edge.spline.info") and use it if found. When there are no spline arcs, it will use straight line arcs (XHTML element "arc"). Please check out examples in our SBGN-ML file collection.

3. About ID mapping

SBGNview can automatically map several ID types to glyphs of pathwayCommons, MetaCyc and MetaCrop. For user defined SBGN-ML file, users need to provide information about how to map their omics data to glyphs.

3.1 How SBGNview finds glyph IDs in SBGN-ML file: Glyph IDs are recorded in attribute "id" in XHTML element "glyph". But for ID mapping, user can use other attributes by changing parameter "sbgn.id.attr".

3.2 For our SBGN-ML file collection., SBGNview can do ID mapping automatically. It uses extracted mapping between 1) UNIPROT/uniref and "macromolecule" glyph IDs and 2) ChEBI and "simple chemical" glyph IDs from biopax files in pathwayCommons and MetaCyc. For other ID types, we used pathview (gene/protein) and UniChem (compound) to map to UNIPROT and ChEBI, respectively, then map them to glyph IDs. Please see the README.txt file for more details. For MetaCrop, we used pathview for ID mapping.

4. Two common scenarios of using SBGNview

4.1 Using our pre-generated SBGN-ML files.

Supported pathways can be found using `data("pathways.info")`. This is a collection of SBGN-ML files for these databases: MetaCyc, MetaCrop and three databases collected by pathwayCommons (panther, Reactome and smpdb). For each file, the glyph layout is based on fdp and optimized to eliminate glyph-glyph overlaps. The arcs are splines that are routed to eliminate arc-glyph crossings.

To use these data, SBGNview needs the following parameters:

-gene.id.type and/or cpd.id.type (at least one should be provided)

SBGNview can map omics data to SBGN-ML glyphs automatically. Supported ID types can be found in `data("mapped.ids")`

Input SBGN-ML files can be obtained by using function "download.sbgn.file". It uses a pathway ID to download the SBGN-ML file. Available pathway IDs can be found in `data("pathways.info")`, in column "pathway.id". database should one of the following "MetaCrop", "MetaCyc" or "pathwayCommons".

The pathway database information for each pathway ID can be found in column "database".

4.2 Using SBGN-ML files from other sources.

In this scenario, database can be any character, but should NOT be set to any of the following "MetaCrop", "MetaCyc" or "pathwayCommons". Because these three values will overwrite other parameters such as "sbgn.gene.id.type" and "sbgn.cpd.id.type"

4.2.1 Input omics data have the SAME ID type as the glyph ID type in SBGN-ML file:

In this scenario, SBGNview needs the following information to map omics data to SBGN-ML glyphs:

-ID type of input omics data (gene.id.type and/or cpd.id.type)

-ID type of glyphs of input SBGN-ML file (sbgn.gene.id.type and/or sbgn.cpd.id.type).

These ID types can be any characters, but gene.id.type must be the same as sbgn.gene.id.type, and cpd.id.type must be the same as sbgn.cpd.id.type.

Users can use the function "change.data.id" to change the omics IDs to the glyph IDs in SBGN-ML file. This is recommended when the users want to overlay omics data on a large number of pathways. Because we only need to do ID mapping once, instead of letting SBGNview to do ID mapping for every pathway.

4.2.2 Input omics data have DIFFERENT ID type as the glyph ID type in SBGN-ML file:

In this scenario, SBGNview needs the following information to map omics data to SBGN-ML glyphs:

-ID type of input omics data (gene.id.type and/or cpd.id.type)

-ID type of glyphs of input SBGN-ML file (sbgn.gene.id.type and/or sbgn.cpd.id.type).

-A mapping table between input omics IDs and SBGN-ML glyph IDs (id.mapping.gene and/or id.mapping.cpd).

For user's convinience, pathview can generate such tables for several ID types (functions "geneannot.map" and "cpdidmap"). But column names need to be changed to the values of "gene.id.type" and "sbgn.gene.id.type", as described in the "parameters" section.

Value

A SBGNview object.

Examples

```
### use simulated data. Please see vignettes for more examples
data("pathways.info","sbgn.xmls")
SBGNview.obj = SBGNview(
              simulate.data = TRUE
              ,sbgn.dir = "./"
              ,input.sbgn = "P00001"

              ,output.file = "./test.local.file"
              ,output.formats = c("pdf")

              ,min.gene.value = -1
              ,max.gene.value = 1
            )
 SBGNview.obj
```

SBGNview.obj — *Demo SBGNview object*

Description

Demo SBGNview object

Usage

```
SBGNview.obj
```

Format

A SBGNview object

Details

An example SBGNview object

spline.arc-class — *An object to store information for spline formated arcs.*

Description

An object to store information for spline formated arcs.

Details

Arc information comes from two sources:1. SBGN-ML file's "arc" element ("source", "target", coordinates etc.). 2. Parameters specified when running `SBGNview`. User can modify arc objects to change the way how it is rendered. Some of the slots/parameters in arc object controls similar graph features such as text font, line width/color etc.

The priority is determined in the following sequence:

1. arc@edge
2. Parameters in slot "arc@global.parameters.list"
3. Slots in arc object: stroke.opacity, fill.color etc.

Slots

`target,source,id,arc.class` A character string. Information extracted from element "arc.spline"

`start.x,start.y,end.x,end.y` Numeric. Information extracted from elements "start", "end" or "next"

`stroke.opacity` Numeric. Controls the line of an arc (not tip of arc).

`components` A list of "arc" and "spline" objects. A spline arc is represented by several components: 1. The two ends of the arc are represented by straight line "arc" objects. 2. splines connecting the ends are represented by "spline" objects.

`global.parameters.list` A list. This is a copy of parameters in '...' of SBGNview. It can affect glyph behavior if the element has a corresponding feature of the arc. For example "edge.tip.size" affects the arc but "compartment.opacity" doesn't. See details for more information. This will be improved in future versions to remove unrelated parameters.

`edge` A list. An arc in SBGN map normally consists of a line and a tip shape at the end of the line. This list holds variables that controls arc properties. Available elements can be accessed as follow:

The following three paremeters control the line.
arc@edge$line.stroke.color
arc@edge$line.stroke.opacity
arc@edge$line.width

The following five paremeters control the tip.
arc@edge$tip.stroke.opacity
arc@edge$tip.stroke.color
arc@edge$tip.stroke.width
arc@edge$tip.fill.color
arc@edge$tip.fill.opacity

Index

∗Topic **datasets**
    mapped.ids, 15
    pathways.info, 17
    pathways.stat, 17
    SBGNview.obj, 29
+.SBGNview, 2, *11*, *12*

arc-class, 3

changeDataId, 4
changeIds, 5

downloadSbgnFile, 6

findPathways, 7

getMolList, 8
glyph-class, 9

highlightArcs, 11
highlightNodes, *2*, 12
highlightPath, *2*, 13

loadMappingTable, 14

mapped.ids, 15

outputFile, 16
outputFile<-, 16

pathways.info, 17
pathways.stat, 17
print.SBGNview, 18

renderSbgn, *18*, 18, *26*

sbgnNodes, 23
SBGNview, *3*, *10*, *16*, *18*, *19*, 24, *29*
SBGNview.obj, 29
spline.arc-class, 29