

# Package ‘gatom’

July 11, 2025

**Title** Finding an Active Metabolic Module in Atom Transition Network

**Version** 1.7.0

**Description** This package implements a metabolic network analysis pipeline to identify an active metabolic module based on high throughput data. The pipeline takes as input transcriptional and/or metabolic data and finds a metabolic subnetwork (module) most regulated between the two conditions of interest. The package further provides functions for module post-processing, annotation and visualization.

**biocViews** GeneExpression, DifferentialExpression, Pathways, Network

**Depends** R (>= 4.3.0)

**Imports** data.table, igraph, BioNet, plyr, methods, XML, sna, intergraph, network, GGally, grid, ggplot2, mwcsr, pryr, htmlwidgets, htmltools, shinyCyJS (>= 1.0.0)

**Suggests** testthat, knitr, rmarkdown, KEGGREST, AnnotationDbi, org.Mm.eg.db, reactome.db, fgsea, readr, BiocStyle, R.utils

**License** MIT + file LICENCE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**URL** <https://github.com/ctlab/gatom/>

**BugReports** <https://github.com/ctlab/gatom/issues>

**git\_url** <https://git.bioconductor.org/packages/gatom>

**git\_branch** devel

**git\_last\_commit** 6509ada

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.22

**Date/Publication** 2025-07-11

**Author** Anastasiia Gainullina [aut],  
Mariia Emelianova [aut],  
Alexey Sergushichev [aut, cre]

**Maintainer** Alexey Sergushichev <alsergbox@gmail.com>

## Contents

|                              |           |
|------------------------------|-----------|
| abbreviateLabels             | 2         |
| addHighlyExpressedEdges      | 3         |
| collapseAtomsIntoMetabolites | 3         |
| connectAtomsInsideMetabolite | 4         |
| createShinyCyJSWidget        | 4         |
| gatom                        | 5         |
| gene.de.rawEx                | 6         |
| getGeneDEMeta                | 6         |
| getMetabolicPathways         | 7         |
| getMetDEMeta                 | 8         |
| gEx                          | 9         |
| gsEx                         | 9         |
| makeMetabolicGraph           | 9         |
| makeOrgGatomAnnotation       | 10        |
| met.de.rawEx                 | 12        |
| met.kegg.dbEx                | 12        |
| mEx                          | 12        |
| networkEx                    | 12        |
| org.Mm.eg.gatom.annoEx       | 13        |
| prepareDE                    | 13        |
| saveModuleToDot              | 14        |
| saveModuleToHtml             | 14        |
| saveModuleToPdf              | 15        |
| saveModuleToXgmml            | 16        |
| scoreGraph                   | 16        |
| styleWidget                  | 17        |
| <b>Index</b>                 | <b>18</b> |

---

|                  |                                                 |
|------------------|-------------------------------------------------|
| abbreviateLabels | <i>Abbreviate lipid labels for lipid module</i> |
|------------------|-------------------------------------------------|

---

### Description

Abbreviate lipid labels for lipid module

### Usage

```
abbreviateLabels(module, orig.names, abbrev.names)
```

### Arguments

|              |                                                 |
|--------------|-------------------------------------------------|
| module       | Module to prepare                               |
| orig.names   | whether to use original names from the dataset  |
| abbrev.names | whether to use abbreviated names for all lipids |

### Value

module object with abbreviated labels

---

`addHighlyExpressedEdges`*Add reactions without highly changing genes but with high average expression*

---

**Description**

Add reactions without highly changing genes but with high average expression

**Usage**

```
addHighlyExpressedEdges(m, g, top = 3000)
```

**Arguments**

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| <code>m</code>   | Metabolic module                                                  |
| <code>g</code>   | Scored graph                                                      |
| <code>top</code> | Maximum rank value for the gene to be considered highly expressed |

**Value**

module with added edges that correspond to high average expression

**Examples**

```
data(mEx)
data(gEx)
m <- addHighlyExpressedEdges(m = mEx, g = gEx)
```

---

`collapseAtomsIntoMetabolites`*Collapse atoms belonging to the same metabolite into one vertex*

---

**Description**

Collapse atoms belonging to the same metabolite into one vertex

**Usage**

```
collapseAtomsIntoMetabolites(m)
```

**Arguments**

|                |                  |
|----------------|------------------|
| <code>m</code> | Metabolic module |
|----------------|------------------|

**Value**

module in which atoms of the same metabolite are collapsed into one

## Examples

```
data(mEx)
m <- collapseAtomsIntoMetabolites(m = mEx)
```

---

connectAtomsInsideMetabolite

*Connect atoms belonging to the same metabolite with edges*

---

## Description

Connect atoms belonging to the same metabolite with edges

## Usage

```
connectAtomsInsideMetabolite(m)
```

## Arguments

m                      Metabolic module

## Value

module in which atoms of the same metabolite are connected

## Examples

```
data(mEx)
m <- connectAtomsInsideMetabolite(m = mEx)
```

---

createShinyCyJSWidget    *Creates shinyCyJS widget from module*

---

## Description

Creates shinyCyJS widget from module

## Usage

```
createShinyCyJSWidget(
  module,
  layout = list(name = "cose-bilkent", animate = FALSE, randomize = FALSE,
    nodeDimensionsIncludeLabels = TRUE),
  ...
)
```

## Arguments

|        |                       |
|--------|-----------------------|
| module | Module                |
| layout | Layout for the module |
| ...    | Other parameters      |

## Value

html widget of input module

## Examples

```
data(mEx)
hw <- createShinyCyJSWidget(module = mEx)
```

---

|       |                                                                                           |
|-------|-------------------------------------------------------------------------------------------|
| gatom | <i>gatom: a package for finding an active metabolic module in atom transition network</i> |
|-------|-------------------------------------------------------------------------------------------|

---

## Description

This package implements a metabolic network analysis pipeline to identify an active metabolic module based on high throughput data. The pipeline takes as input transcriptional and/or metabolic data and finds a metabolic subnetwork (module) most regulated between the two conditions of interest. The package further provides functions for module post-processing, annotation and visualization.

## Functions

Data preprocessing: [prepareDE](#), [getMetDEMeta](#), [getGeneDEMeta](#)

Graph creation: [makeMetabolicGraph](#)

Graph scoring: [scoreGraph](#)

Module postprocessing: [collapseAtomsIntoMetabolites](#), [connectAtomsInsideMetabolite](#), [addHighlyExpressed](#), [abbreviateLabels](#)

Plotting module: [createShinyCyJSWidget](#)

Exporting module: [saveModuleToHtml](#), [saveModuleToDot](#), [saveModuleToPdf](#), [saveModuleToXgmm1](#)

For detailed pipeline analysis, see gatom vignette: `vignette("gatom-tutorial", package = "gatom")`

## Example Data

Example data provided by gatom consists of: metabolite differential abundance data ([met.de.rawEx](#)), gene differential expression data ([gene.de.rawEx](#)), KEGG-based network object ([networkEx](#)), KEGG-based metabolite database object ([met.kegg.dbEx](#)), Example organism annotation object ([org.Mm.eg.gatom.annoEx](#)), metabolic graph with atom topology ([gEx](#)), scored metabolic graph with atom topology ([gsEx](#)), and metabolic module ([mEx](#)).

---

|               |                                                   |
|---------------|---------------------------------------------------|
| gene.de.rawEx | <i>Example gene differential expression data.</i> |
|---------------|---------------------------------------------------|

---

### Description

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

### Format

tibble/data.frame object

---

|               |                                                                                        |
|---------------|----------------------------------------------------------------------------------------|
| getGeneDEMeta | <i>Finds columns in gene differential expression table required for gatom analysis</i> |
|---------------|----------------------------------------------------------------------------------------|

---

### Description

Default values for all columns are NULL which mean they are determined automatically.

### Usage

```
getGeneDEMeta(
  gene.de.raw,
  org.gatom.anno,
  idColumn = NULL,
  idType = NULL,
  pvalColumn = NULL,
  logPvalColumn = NULL,
  log2FCColumn = NULL,
  baseMeanColumn = NULL,
  signalColumn = NULL,
  signalRankColumn = NULL
)
```

### Arguments

|                |                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------|
| gene.de.raw    | A table with differential expression results, an object convertible to data.frame.                  |
| org.gatom.anno | Organsim-specific annotation obtained from makeOrgGatomAnnotation function.                         |
| idColumn       | Specifies column name with gene identifiers.                                                        |
| idType         | Specifies type of gene IDs (one of the supported by annotation).                                    |
| pvalColumn     | Specifies column with p-values.                                                                     |
| logPvalColumn  | Specifies column with log p-values, if there is no such column one will be generated automatically. |
| log2FCColumn   | Specifies column with log2-fold changes.                                                            |
| baseMeanColumn | Specifies column with average expression across samples.                                            |

- signalColumn** Specifies column with identifier of the measured entity (such as gene ID for RNA-seq and probe ID for microarrays). Could be NULL (automatic, set from based on pval and log2FC columns), character (column name), or function (evaluated in a scope of original data frame)
- signalRankColumn** Specifies how the genes are ranked from highly to lowly expressed, used in 'addHighlyExpressedEdges' function. Could be NULL (automatic), character (column name) function (evaluated in a scope of original data frame).

**Value**

object with prepared columns for the analysis for gene data

**Examples**

```
data("org.Mm.eg.gatom.annoEx")
data("gene.de.rawEx")
de.meta <- getGeneDEMeta(gene.de.rawEx, org.gatom.anno = org.Mm.eg.gatom.annoEx)
```

---

`getMetabolicPathways`    *Generate list of metabolic pathways from Reactome and KEGG databases*

---

**Description**

Generate list of metabolic pathways from Reactome and KEGG databases

**Usage**

```
getMetabolicPathways(
  universe,
  metGenes,
  keggOrgCode,
  threshold = 0.01,
  includeReactome = TRUE,
  includeKEGG = TRUE
)
```

**Arguments**

- universe** list of genes
- metGenes** list of metabolic genes
- keggOrgCode** KEGG organism code, like mmu or hsa
- threshold** threshold for Fisher test to filter out non-metabolic pathways
- includeReactome** whether to include Reactome pathways (only works for Entrez ID universe)
- includeKEGG** whether to include KEGG pathways and modules

**Value**

list of metabolic pathways for given organism and list of genes

---

|              |                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------|
| getMetDEMeta | <i>Finds columns in differential expression table for metabolites required for gatom analysis</i> |
|--------------|---------------------------------------------------------------------------------------------------|

---

## Description

Finds columns in differential expression table for metabolites required for gatom analysis

## Usage

```
getMetDEMeta(  
  met.de.raw,  
  met.db,  
  idColumn = NULL,  
  idType = NULL,  
  pvalColumn = NULL,  
  logPvalColumn = NULL,  
  log2FCColumn = NULL,  
  signalColumn = NULL  
)
```

## Arguments

|               |                                                                                                                                                                                                                  |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| met.de.raw    | A table with differential expression results, an object convertible to data.frame.                                                                                                                               |
| met.db        | Metabolite database                                                                                                                                                                                              |
| idColumn      | Specifies column name with metabolite identifiers.                                                                                                                                                               |
| idType        | Specifies type of metabolite IDs (one of the supported by annotation).                                                                                                                                           |
| pvalColumn    | Specifies column with p-values.                                                                                                                                                                                  |
| logPvalColumn | Specifies column with log p-values, if there is no such column one will be generated automatically.                                                                                                              |
| log2FCColumn  | Specifies column with log2-fold changes.                                                                                                                                                                         |
| signalColumn  | Specifies column with identifier of the measured entity Could be NULL (automatic, set from based on pval and log2FC columns), character (column name), or function (evaluated in a scope of original data frame) |

## Value

object with prepared columns for the analysis for metabolite data

## Examples

```
data("met.kegg.dbEx")  
data("met.de.rawEx")  
de.meta <- getMetDEMeta(met.de.rawEx, met.db = met.kegg.dbEx)
```

---

`gEx`*Example metabolic graph with atom topology.*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

igraph object

---

`gsEx`*Example scored metabolic graph with atom topology.*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

igraph object

---

`makeMetabolicGraph`*Creates metabolic graph based on specified data*

---

**Description**

Creates metabolic graph based on specified data

**Usage**

```
makeMetabolicGraph(  
  network,  
  topology = c("atoms", "metabolites"),  
  org.gatom.anno,  
  gene.de,  
  gene.de.meta = getGeneDEMeta(gene.de, org.gatom.anno),  
  gene.keep.top = 12000,  
  met.db,  
  met.de,  
  met.de.meta = getMetDEMeta(met.de, met.db),  
  met.to.filter = fread(system.file("extdata", "mets2mask.lst", package = "gatom"))$ID,  
  gene2reaction.extra = NULL,  
  keepReactionsWithoutEnzymes = FALSE,  
  largest.component = TRUE  
)
```

**Arguments**

|                             |                                                                                                                                                                   |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| network                     | Network object                                                                                                                                                    |
| topology                    | Way to determine network vertices                                                                                                                                 |
| org.gatom.anno              | Organism annotation object                                                                                                                                        |
| gene.de                     | Table with the differential gene expression, set to NULL if absent                                                                                                |
| gene.de.meta                | Annotation of 'gene.de' table                                                                                                                                     |
| gene.keep.top               | Only the 'gene.keep.top' of the most expressed genes will be kept for the network                                                                                 |
| met.db                      | Metabolite database                                                                                                                                               |
| met.de                      | Table with the differential expression for metabolites, set to NULL if absent                                                                                     |
| met.de.meta                 | Annotation of 'met.de' table                                                                                                                                      |
| met.to.filter               | List of metabolites to filter from the network                                                                                                                    |
| gene2reaction.extra         | Additional gene to reaction mappings. Should be a data.table with 'gene' and 'reaction' columns                                                                   |
| keepReactionsWithoutEnzymes | If TRUE, keep reactions that have no annotated enzymes, thus expanding the network but including some reactions which are not possible in the considered species. |
| largest.component           | If TRUE, only the largest connected component is returned                                                                                                         |

**Value**

igraph object created from input data

**Examples**

```
data("gene.de.rawEx")
data("met.de.rawEx")
data("met.kegg.dbEx")
data("networkEx")
data("org.Mm.eg.gatom.annoEx")
g <- makeMetabolicGraph(network = networkEx, topology = "atoms",
                        org.gatom.anno = org.Mm.eg.gatom.annoEx,
                        gene.de = gene.de.rawEx, met.db = met.kegg.dbEx,
                        met.de = met.de.rawEx)
```

---

makeOrgGatomAnnotation

*Create an organism annotation object for network analysis*

---

**Description**

Create an organism annotation object for network analysis

**Usage**

```
makeOrgGatomAnnotation(
  org.db,
  idColumns = c(Entrez = "ENTREZID", RefSeq = "REFSEQ", Ensembl = "ENSEMBL", Symbol =
    "SYMBOL"),
  nameColumn = "SYMBOL",
  enzymeColumn = "ENZYME",
  appendEnzymesFromKegg = TRUE,
  appendOrthologiesFromKegg = TRUE,
  filterNonSpecificEnzymes = TRUE,
  keggOrgCode = NULL
)
```

**Arguments**

|                                        |                                                                                                                                                                      |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>org.db</code>                    | Bioconductor <code>org.db</code> object, e.g. <code>org.Mm.eg.db</code>                                                                                              |
| <code>idColumns</code>                 | vector of column names from 'org.db' object to creat ID mappings. First ID will be used as a base identifier, should be compatible with KEGG and Reactome databases. |
| <code>nameColumn</code>                | column with a human readable gene symbol. Default to "SYMBOL".                                                                                                       |
| <code>enzymeColumn</code>              | column with an Enzyme Commission ID. Default to "ENZYME".                                                                                                            |
| <code>appendEnzymesFromKegg</code>     | if TRUE, KEGG databases will be sued to extend gene to enzyme mappings obtained from <code>org.db</code> package.                                                    |
| <code>appendOrthologiesFromKegg</code> | if TRUE, KEGG database will be sued to extend gene to orthology mappings obtained from <code>org.db</code> package                                                   |
| <code>filterNonSpecificEnzymes</code>  | if TRUE, will filter out non-specific enzymes from gene to enzyme mappings obtained from <code>org.db</code> package                                                 |
| <code>keggOrgCode</code>               | KEGG organism code, e.g. "mmu". If set to NULL, the code is determined automatically.                                                                                |

**Value**

organism annotation object that will be used for network analysis

**Examples**

```
library(org.Mm.eg.db)
org.Mm.eg.gatom.anno <- makeOrgGatomAnnotation(org.db = org.Mm.eg.db)
```

---

`met.de.rawEx`*Example metabolite differential abundance data.*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

tibble/data.frame object

---

`met.kegg.dbEx`*Example KEGG-based metabolite database object*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

list object

---

`mEx`*Example metabolic module.*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

igraph object

---

`networkEx`*Example KEGG-based network object*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

list object

---

`org.Mm.eg.gatom.annoEx`*Example organism annotation object*

---

**Description**

See file <https://github.com/ctlab/gatom/blob/master/inst/scripts/example.R> for details.

**Format**

list object

---

`prepareDE`*Makes data.table with differential expression results containing all columns required for gatom and in the expected format based on meta-data object*

---

**Description**

Makes data.table with differential expression results containing all columns required for gatom and in the expected format based on metadata object

**Usage**

```
prepareDE(de.raw, de.meta)
```

**Arguments**

|                      |                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>de.raw</code>  | Table with differential expression results, an object convertible to data.frame                                                    |
| <code>de.meta</code> | Object with differential expression table metadata acquired with <code>getGeneDEMeta</code> or <code>getMetDEMeta</code> functions |

**Value**

data.table object with converted differential expression table

**Examples**

```
data("org.Mm.eg.gatom.annoEx")
data("gene.de.rawEx")
de.meta <- getGeneDEMeta(gene.de.rawEx, org.gatom.anno = org.Mm.eg.gatom.annoEx)
de <- prepareDE(gene.de.rawEx, de.meta)
```

---

|                 |                                           |
|-----------------|-------------------------------------------|
| saveModuleToDot | <i>Save module to a graphviz dot file</i> |
|-----------------|-------------------------------------------|

---

### Description

Save module to a graphviz dot file

### Usage

```
saveModuleToDot(
  module,
  file,
  name = NULL,
  extra.node.attrs = NULL,
  extra.edge.attrs = NULL
)
```

### Arguments

|                  |                                                                           |
|------------------|---------------------------------------------------------------------------|
| module           | Module to save                                                            |
| file             | File to save to                                                           |
| name             | Name of the module                                                        |
| extra.node.attrs | Table with additional node attributes to be written to the dot file as is |
| extra.edge.attrs | Table with additional edge attributes to be written to the dot file as is |

### Value

Returns NULL

### Examples

```
data(mEx)
saveModuleToDot(module = mEx, file = "module.dot")
```

---

|                  |                                     |
|------------------|-------------------------------------|
| saveModuleToHtml | <i>Save module to a html widget</i> |
|------------------|-------------------------------------|

---

### Description

Save module to a html widget

**Usage**

```

saveModuleToHtml(
    module,
    file,
    name = "",
    sizingPolicy = htmlwidgets::sizingPolicy(defaultWidth = "100%", defaultHeight =
        "90vh", padding = 10),
    ...
)

```

**Arguments**

|              |                        |
|--------------|------------------------|
| module       | Module to save         |
| file         | File to save to        |
| name         | Name of the module     |
| sizingPolicy | A widget sizing policy |
| ...          | Other parameters       |

**Value**

Returns NULL

**Examples**

```

data(mEx)
saveModuleToHtml(module = mEx, file = "module.html")

```

---

|                 |                                       |
|-----------------|---------------------------------------|
| saveModuleToPdf | <i>Save module to a nice pdf file</i> |
|-----------------|---------------------------------------|

---

**Description**

Save module to a nice pdf file

**Usage**

```
saveModuleToPdf(module, file, name = NULL, n_iter = 100, force = 1e-05)
```

**Arguments**

|        |                                      |
|--------|--------------------------------------|
| module | Module to save                       |
| file   | File to save to                      |
| name   | Name of the module                   |
| n_iter | Number of repel algorithm iterations |
| force  | Value of repel force                 |

**Value**

Returns NULL

**Examples**

```
data(mEx)
saveModuleToPdf(module = mEx, file = "module.pdf")
```

---

|                   |                                     |
|-------------------|-------------------------------------|
| saveModuleToXgmml | <i>Save module to an XGMML file</i> |
|-------------------|-------------------------------------|

---

**Description**

Save module to an XGMML file

**Usage**

```
saveModuleToXgmml(module, file, name = NULL)
```

**Arguments**

|        |                    |
|--------|--------------------|
| module | Module to save     |
| file   | File to save to    |
| name   | Name of the module |

**Value**

Returns NULL

**Examples**

```
data(mEx)
saveModuleToXgmml(module = mEx, file = "module.xgmml")
```

---

|            |                              |
|------------|------------------------------|
| scoreGraph | <i>Score metabolic graph</i> |
|------------|------------------------------|

---

**Description**

Score metabolic graph

**Usage**

```
scoreGraph(
  g,
  k.gene,
  k.met,
  vertex.threshold.min = 0.1,
  edge.threshold.min = 0.1,
  met.score.coef = 1,
  show.warnings = TRUE,
  raw = FALSE
)
```

**Arguments**

|                                   |                                                                                                                                                                                                   |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>g</code>                    | Metabolic graph obtained with <code>makeMetabolic</code> graph function                                                                                                                           |
| <code>k.gene</code>               | Number of gene signals to be scored positively, the higher is the number, the larger will be the resulting module. If set to <code>NULL</code> , genes will not be used for scoring.              |
| <code>k.met</code>                | Number of metabolite signals to be scored positively, the higher is the number, the larger will be the resulting module. If set to <code>NULL</code> , metabolites will not be used for scoring.  |
| <code>vertex.threshold.min</code> | The worst acceptable estimated FDR for vertices. If necessary number of positive metabolite signals will be decreased from ' <code>k.met</code> ' to reach this threshold. Default value is 0.1.  |
| <code>edge.threshold.min</code>   | The worst acceptable estimated FDR for vertices. If necessary number of positive metabolite signals will be decreased from ' <code>k.gene</code> ' to reach this threshold. Default value is 0.1. |
| <code>met.score.coef</code>       | Coefficient on which all vertex weights are multiplied. Can be used to balance vertex and edge weights. Default values is 1.                                                                      |
| <code>show.warnings</code>        | whether to show warnings                                                                                                                                                                          |
| <code>raw</code>                  | whether to return raw scored graph, not a SGMWCS instance. Default to <code>FALSE</code> .                                                                                                        |

**Value**

SGMWCS instance or scored igraph object

**Examples**

```
data("gEx")
gs <- scoreGraph(g = gEx, k.gene = 25, k.met = 25)
```

---

styleWidget

---

*code adopted from <https://github.com/ramnathv/htmlwidgets/issues/231>*


---

**Description**

code adopted from <https://github.com/ramnathv/htmlwidgets/issues/231>

**Usage**

```
styleWidget(hw, style = "", addl_selector = "", elementId = NULL)
```

**Value**

styled html widget

# Index

## \* internal

styleWidget, 17

abbreviateLabels, 2, 5

addHighlyExpressedEdges, 3, 5

collapseAtomsIntoMetabolites, 3, 5

connectAtomsInsideMetabolite, 4, 5

createShinyCyJSWidget, 4, 5

gatom, 5

gene.de.rawEx, 5, 6

getGeneDEMeta, 5, 6

getMetabolicPathways, 7

getMetDEMeta, 5, 8

gEx, 5, 9

gsEx, 5, 9

makeMetabolicGraph, 5, 9

makeOrgGatomAnnotation, 10

met.de.rawEx, 5, 12

met.kegg.dbEx, 5, 12

mEx, 5, 12

networkEx, 5, 12

org.Mm.eg.gatom.annoEx, 5, 13

prepareDE, 5, 13

saveModuleToDot, 5, 14

saveModuleToHtml, 5, 14

saveModuleToPdf, 5, 15

saveModuleToXgmml, 5, 16

scoreGraph, 5, 16

styleWidget, 17