

# Package ‘anansi’

April 7, 2026

**Type** Package

**Title** Annotation-Based Analysis of Specific Interactions

**Version** 1.1.0

**Description** Studies including both microbiome and metabolomics data are becoming more common. Often, it would be helpful to integrate both datasets in order to see if they corroborate each others patterns. All vs all association is imprecise and likely to yield spurious associations. This package takes a knowledge-based approach to constrain association search space, only considering metabolite-function pairs that have been recorded in a pathway database. This package also provides a framework to assess differential association.

**Depends** R (>= 4.5.0)

**Imports** S7, stats, methods, igraph, Matrix, forcats, S4Vectors, SummarizedExperiment, MultiAssayExperiment, SingleCellExperiment, TreeSummarizedExperiment, rlang, ggplot2, ggforce, patchwork, ggraph, tidygraph

**Suggests** BiocStyle, dplyr, tidyr, graph, mia, KEGGREST, testthat (>= 3.0.0), knitr, rmarkdown

**License** GPL-3

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**biocViews** Microbiome, Metabolomics, Regression, Pathways, KEGG

**URL** <https://github.com/thomazbastiaanssen/anansi>,  
<https://thomazbastiaanssen.github.io/anansi>

**BugReports** <https://github.com/thomazbastiaanssen/anansi/issues>

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE)

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

**git\_branch** devel

**git\_last\_commit** d5ff74a

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2026-04-07

**Author** Thomaz Bastiaanssen [aut, cre] (ORCID:

<https://orcid.org/0000-0001-6891-734X>),

Thomas Quinn [aut] (ORCID: <https://orcid.org/0000-0003-0286-6329>),

Giulio Benedetti [aut] (ORCID: <https://orcid.org/0000-0002-8732-7692>),

Tuomas Borman [aut] (ORCID: <https://orcid.org/0000-0002-8563-8884>),

Leo Lahti [aut] (ORCID: <https://orcid.org/0000-0001-5537-637X>)

**Maintainer** Thomaz Bastiaanssen <[thomazbastiaanssen@gmail.com](mailto:thomazbastiaanssen@gmail.com)>

## Contents

anansi . . . . .	3
anansi-coercion . . . . .	6
anansiCor . . . . .	8
anansiCorPvalue . . . . .	8
anansiCorTestByGroup . . . . .	9
anansiDiffCor . . . . .	10
AnansiTale . . . . .	10
AnansiWeb . . . . .	12
AnansiWeb-methods . . . . .	13
AnansiWeb-pairwise . . . . .	14
dictionary . . . . .	15
ec2ko . . . . .	16
FMT_KOs . . . . .	17
FMT_metab . . . . .	18
FMT_metadata . . . . .	18
getEdgeList . . . . .	19
getFeaturePairs . . . . .	19
getGraph . . . . .	20
krebs . . . . .	20
linkBiobakeryMap . . . . .	21
metadata . . . . .	22
metadata<- . . . . .	22
MultiFactor . . . . .	23
MultiFactor-methods . . . . .	24
pairwiseApply . . . . .	25
plotAnansi . . . . .	26
randomAnansi . . . . .	28
tableX . . . . .	30
tableY . . . . .	30
weaveWeb . . . . .	31
weaveWeb-methods . . . . .	32

---

anansi

*anansi: Annotation-based Analysis of Specific Interactions.*

---

## Description

The anansi package provides tools to prepare and facilitate integrative association analysis between the features of two data sets that are known to interact.

### 1. Input for `anansi()` with `AnansiWeb()` and `MultiFactor()`:

- `randomAnansi`, `kegg_link()`: Generate example input
- `AnansiWeb`, `MultiFactor`: Handle and manipulate input

### 2. Output and cross-compatibility:

- `getGraph()`: Compatibility with `igraph::igraph`
- `plotAnansi()`: Plot output in the style of `miaViz::miaViz`

### 3. Vignettes:

- 1. Getting started with anansi
- 2. Adjacency matrices
- 3. Association testing

### Description: The `anansi()` function:

The main workspider function in the anansi package is called `anansi`. It manages the individual functionalities of anansi, including correlation analysis, correlation by group and differential associations.

## Usage

```
anansi(  
  web,  
  formula,  
  groups = NULL,  
  metadata = NULL,  
  adjust.method = "BH",  
  verbose = TRUE,  
  return.format = "table",  
  ...  
)
```

**Arguments**

web	An AnansiWeb object, containing two tables with 'omics data and a dictionary that links them. See <code>weaveWebFromTables()</code> for how to weave a web.
formula	A formula object. Used to assess differential associations.
groups	A vector of the column names of categorical values in the metadata object to control which groups should be assessed for simple correlations. If no argument provided, anansi will let you know and still to regular correlations according to your dictionary.
metadata	A vector or data.frame of categorical or continuous value necessary for differential correlations. Typically a state or treatment. If no argument provided, anansi will let you know and still to regular correlations according to your dictionary.
adjust.method	Method to adjust p-values for multiple comparisons. <code>adjust.method = "BH"</code> is the default value. See <code>p.adjust()</code> in the base R stats package.
verbose	Logical scalar. Whether to print diagnostic information (Default: TRUE). while running. Useful for debugging errors on large datasets.
return.format	Character scalar. Should be one of "table", "list", or "raw". Should the output of <code>anansi()</code> respectively be a wide data.frame of results, a list containing the results and input, or a list of raw output (used for testing purposes). convenient use. (Default: "table")
...	additional arguments (currently not used).

**Value**

A list of lists containing correlation coefficients, p-values and q-values for all operations.

**Author(s)**

**Maintainer:** Thomaz Bastiaanssen <thomazbastiaanssen@gmail.com> ([ORCID](#))

Authors:

- Thomas Quinn <contacttomquinn@gmail.com> ([ORCID](#))
- Giulio Benedetti <giulio.benedetti@utu.fi> ([ORCID](#))
- Tuomas Borman <tuomas.v.borman@utu.fi> ([ORCID](#))
- Leo Lahti <leo.lahti@utu.fi> ([ORCID](#))

**See Also**

Useful links:

- <https://github.com/thomazbastiaanssen/anansi>
- <https://thomazbastiaanssen.github.io/anansi>
- Report bugs at <https://github.com/thomazbastiaanssen/anansi/issues>

**Examples**

```
# Load example data:

data(FMT_data)

# Make sure that columns are features and rows are samples.

t1 <- t(FMT_metab)
t2 <- t(FMT_KOs)

# Run anansi pipeline.

web <- weaveWeb(
  cpd ~ ko,
  tableY = t1,
  tableX = t2,
  link = kegg_link()
)

anansi_out <- anansi(
  web = web,
  formula = ~Legend,
  groups = "Legend",
  metadata = FMT_metadata,
  adjust.method = "BH",
  verbose = TRUE
)

library(tidyr)

# Use tidyr to wrangle the correlation r-values to a single column
anansiLong <- anansi_out |>
  pivot_longer(starts_with("All") | contains("FMT")) |>
  separate_wider_delim(
    name,
    delim = "_", names = c("cor_group", "param")
  ) |>
  pivot_wider(names_from = param, values_from = value)

# Only consider interactions where the entire model fits well enough.
library(ggplot2)
anansiLong <- anansiLong[anansiLong$full_p.values < 0.05, ]

ggplot(
  data = anansiLong,
  aes(
    x = r.values,
    y = feature_X,
    fill = cor_group,
```

```

    alpha = disjointed_Legend_p.values < 0.05
  )
) +

# Make a vertical dashed red line at x = 0
geom_vline(xintercept = 0, linetype = "dashed", colour = "red") +

# Points show raw correlation coefficients
geom_point(shape = 21, size = 3) +

# facet per compound
ggforce::facet_col(~feature_Y, space = "free", scales = "free_y") +
scale_fill_manual(values = c(
  "Young yFMT" = "#2166ac",
  "Aged oFMT" = "#b2182b",
  "Aged yFMT" = "#ef8a62",
  "All" = "gray"
)) +
theme_bw()

# Using miaViz style function:

p <- plotAnansi(anansi_out,
  association.type = "disjointed",
  model.var = "Legend",
  fill_by = "group",
  signif.threshold = 0.05,
  x_lab = "Pearson's rho"
)
p <- p +
  scale_fill_manual(values = c(
    "Young yFMT" = "#2166ac",
    "Aged oFMT" = "#b2182b",
    "Aged yFMT" = "#ef8a62",
    "All" = "gray"
  )) +
  theme_bw()

p

```

## Description

Coercion functions for anansi

**Usage**

```
## S3 method for class '`anansi::AnansiWeb`'  
as.list(x, ...)  
  
## S3 method for class '`anansi::MultiFactor`'  
as.list(x, ..., use.names = TRUE)  
  
## S3 method for class '`anansi::AnansiWeb`'  
as.data.frame(x, row.names, optional, ...)  
  
asMAE(x)  
  
asTSE(x)
```

**Arguments**

x	input object
...	additional arguments (currently not used).
use.names	Logical scalar, whether output list should contain character (Default) or integer data frame. If FALSE, returns <code>unfactor(x)</code> .
row.names, optional	Not used. See <code>?base::data.frame</code>

**Value**

An object of the desired class.

**See Also**

[unfactor\(\)](#)

**Examples**

```
# AnansiWeb  
x <- randomWeb(  
  n_samples = 5,  
  n_features_x = 4,  
  n_features_y = 6  
)  
  
as.list(x)  
as.data.frame(x)  
  
# AnansiWeb to MultiAssayExperiment  
asMAE(x)  
asTSE(x)  
  
# MultiFactor  
x <- randomMultiFactor(n_types = 3, n_features = 3)  
as.list(x, use.names = TRUE)
```

---

anansiCor	<i>Compute r-statistics for each featureY-featureX pair in the dictionary. Typically, the main anansi() function will run this for you.</i>
-----------	---

---

### Description

Compute r-statistics for each featureY-featureX pair in the dictionary. Typically, the main anansi() function will run this for you.

### Usage

```
anansiCor(web, group.bool)
```

### Arguments

web	An AnansiWeb object, containing two tables with omics data and a dictionary that links them. See weaveWebFromTables() for how to weave a web.
group.bool	A boolean vector used to select which samples should be included in the correlations.

### Value

A matrix of r-statistics.

### See Also

[anansi\(\)](#)  
[anansiCorTestByGroup\(\)](#)

---

anansiCorPvalue	<i>Compute r-statistics for each featureY-featureX pair in the dictionary. Typically, the main anansi() function will run this for you.</i>
-----------------	---

---

### Description

Compute r-statistics for each featureY-featureX pair in the dictionary. Typically, the main anansi() function will run this for you.

### Usage

```
anansiCorPvalue(web, group.bool, verbose)
```

**Arguments**

web	An AnansiWeb object, containing two tables with omics data and a dictionary that links them. See <code>weaveWebFromTables()</code> for how to weave a web.
group.bool	A categorical or continuous value necessary for differential correlations. Typically a state or treatment score. If no argument provided, anansi will let you know and still to regular correlations according to your dictionary.
verbose	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets.

**Value**

An AnansiTale result object.

**See Also**

[anansi\(\)](#)  
[anansiCorTestByGroup\(\)](#)

---

`anansiCorTestByGroup` *Run correlations for all interacting metabolites and functions.*

---

**Description**

If the groups argument is suitable, will also run correlation analysis per group. Typically, the main `anansi()` function will run this for you.

**Usage**

```
anansiCorTestByGroup(web, group.vec, verbose = TRUE)
```

**Arguments**

web	An AnansiWeb object, containing two tables with omics data and a dictionary that links them. See <code>weaveWebFromTables()</code> for how to weave a web.
group.vec	A character vector denoting group membership. Typically a state or treatment score.
verbose	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets.

**Value**

a list of AnansiTale result objects, one for the total dataset and per group if applicable.

**See Also**

[anansi\(\)](#)

---

anansiDiffCor	<i>Run differential correlation analysis for all interacting metabolites and functions.</i>
---------------	---

---

### Description

Can either take continuous or categorical data. Typically, the main `anansi()` function will run this for you.

### Usage

```
anansiDiffCor(web, sat_model, errorterm, int.terms, metadata, verbose)
```

### Arguments

web	An AnansiWeb object, containing two tables with omics data and a dictionary that links them. See <code>weaveWebFromTables()</code> for how to weave a web.
sat_model	A formula object, containing the full model
errorterm	A character vector, containing the metadata col.name denoting repeated measures.
int.terms	A character vector, containing the metadata col.names denoting covariates interacting with X to be tested for differential associations.
metadata	A vector or data.frame of categorical or continuous value necessary for differential correlations. Often a state or treatment score. If no argument provided, anansi will let you know and still to regular correlations according to your dictionary.
verbose	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets.

### Value

a list of AnansiTale result objects, one for the total model, one for emergent correlations and one for disjointed correlations.

---

AnansiTale	<i>AnansiTale S7 container class. Not intended for general use.</i>
------------	---

---

### Description

AnansiTale is the main container that will hold your stats output data coming out of the anansi pipeline.

**Usage**

```
AnansiTale(
  subject = character(0),
  type = character(0),
  df = integer(0),
  estimates = integer(0),
  f.values = integer(0),
  t.values = integer(0),
  p.values = integer(0)
)
```

**Arguments**

subject	A character that describes the data that was queried.
type	A character that describes type of parameter contained in the estimates slot. For example r.values for correlations or r.squared for models.
df	a vector of length 2, containing df1 and df2 corresponding to the F-ratio considered.
estimates	A matrix containing the estimates for the parameters named in the type slot.
f.values	A matrix containing the f-values, for least-squares.
t.values	A matrix containing the t-values, for correlations.
p.values	A matrix containing the p.values for the parameters named in the type slot.

**Value**

an AnansiTale

**Slots**

subject	A character that describes the data that was queried.
type	A character that describes type of parameter contained in the estimates slot. For example r.values for correlations or r.squared for models.
df	a vector of length 2, containing df1 and df2 corresponding to the F-ratio considered.
estimates	A matrix containing the estimates for the parameters named in the type slot.
f.values	A matrix containing the f-values, for least-squares.
t.values	A matrix containing the t-values, for correlations.
p.values	A matrix containing the p.values for the parameters named in the type slot.

**Examples**

```
AnansiTale
```

AnansiWeb

*AnansiWeb S7 container class***Description**

AnansiWeb is an S7 class containing two feature tables as well as a dictionary to link them. AnansiWeb is the main container that will hold your input data throughout the anansi pipeline.

Typical use of the anansi package will involve generating an AnansiWeb object using the `weaveWeb()` function.

The function `AnansiWeb()` constructs an AnansiWeb object from two feature tables and an adjacency matrix.

**Usage**

```
## Constructor for `AnansiWeb` objects
AnansiWeb(tableX, tableY, dictionary, metadata = data.frame())
```

**Arguments**

<code>tableY, tableX</code>	A table containing features of interest. Rows should be samples and columns should be features. Y and X refer to the position of the features in a formula: $Y \sim X$ .
<code>dictionary</code>	A binary adjacency matrix of class <code>Matrix</code> , or coercible to <code>Matrix</code>
<code>metadata</code>	list of metadata. Optional.

**Value**

an AnansiWeb object, with sparse binary biadjacency matrix with features from y as rows and features from x as columns in dictionary slot.

**Slots**

<code>tableY, tableX</code>	Two matrix objects of measurements, data. Rows are samples and columns are features. Access with <code>@tableY</code> and <code>@tableX</code> .
<code>dictionary</code>	<code>Matrix</code> , binary adjacency matrix. Optionally sparse. Typically generated using the <code>weaveWeb()</code> function. Access with <code>@dictionary</code> .
<code>metadata</code>	Optional <code>data.frame</code> of sample metadata. Access with <code>@metadata</code> .

**See Also**

- [AnansiWeb-methods](#)
- `randomAnansi`: For generation of random AnansiWeb objects.
- `kegg_link()`: For examples of input for link argument.

## Examples

```
# Use AnansiWeb() to construct an AnansiWeb object from components:
tX <- `dimnames<-`(replicate(5, (rnorm(36))),
  value = list(
    as.character(seq_len(36)),
    letters[1:5]
  )
)
tY <- `dimnames<-`(replicate(3, (rnorm(36))),
  value = list(
    as.character(seq_len(36)),
    LETTERS[1:3]
  )
)
d <- matrix(TRUE,
  nrow = NCOL(tY), ncol = NCOL(tX),

  # Note: Dictionary should have named dimensions
  dimnames = list(
    y_names = colnames(tY),
    x_names = colnames(tX)
  )
)
web <- AnansiWeb(tableX = tX, tableY = tY, dictionary = d)
```

---

AnansiWeb-methods

*Methods for AnansiWeb S7 container class*

---

## Description

Methods for AnansiWeb S7 container class

## Arguments

x                   input, AnansiWeb object

## Value

The desired information from an AnansiWeb object

## See Also

- [weaveWeb\(\)](#): for general use.
- [AnansiWeb-pairwise](#): for methods for pairwise operations

**Examples**

```

# Setup
web <- randomWeb(n_samp = 36)

# Accessors
dimnames(web)
dim(web)
names(web)

# Getters and setters:

tableX(web)[1:5, 1:5]
tableY(web)[1:5, 1:5]
dictionary(web)
head(metadata(web))

# Assign some random metadata
metadata(web) <- data.frame(
  id = row.names(tableY(web)),
  a = rnorm(36),
  b = sample(c("a", "b"), 36, TRUE),
  row.names = "id"
)

# Coerce to list
weblist <- as.list(web)

# Coerce to Data.frame
webdf <- as.data.frame(web)

# Coerce to MultiAssayExperiment
mae <- asMAE(web)

# Coerce to TreeSummarizedExperiment
tse <- asTSE(web)

```

---

AnansiWeb-pairwise      *Methods for pairwise operations on AnansiWeb objects*

---

**Description**

Methods to run pairwise analysis on multi-modal data.

**Arguments**

X, x	input, AnansiWeb object
...	additional arguments
FUN	a function with at least two arguments. The variables x and y, in order, refer to the corresponding values of feature pairs in tableX and tableY.

MoreArgs, SIMPLIFY, USE.NAMES  
 see ?base::mapply  
 which integer matrix, indicating pair positions in x@tableY and x@tableX, respectively. If NULL (default): Matrix::which(x@dictionary, TRUE).  
 with.metadata Logical scalar whether to append metadata to output

**Value**

pairs: A two-column array index, corresponding to i,j coordinates in matrix notation.  
 getFeaturePairs: A list of data.frames with the paired data

**Examples**

```
web <- randomWeb()
# Extract data.frames in pairs (only show first)
getFeaturePairs(web)[1L]

pairs(x = web)

getFeaturePairs(x = web, which = NULL, with.metadata = FALSE)

pairwiseApply(
  X = web,
  FUN = function(x, y) cor(x, y),
  MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE
)
```

---

 dictionary

*Get dictionary*


---

**Description**

Get dictionary

**Usage**

```
dictionary(x, ...)
```

**Arguments**

x input object  
 ... additional arguments

**Value**

dictionary slot of x

## Examples

```
x <- randomWeb(10)
dictionary(x)
```

---

ec2ko

*Use linking data from the KEGG database.*

---

## Description

`kegg_link()` is a convenience function to return a list containing two `data.frames`; `ec2cpd` and `ec2ko`. This will be their most likely use. `ec2cpd` and `ec2ko` are two `data.frames`, used to link `ko`, `ecs` and `cpd` identifiers in the KEGG database.

## Usage

```
data("ec2ko", package = "anansi")
data("ec2cpd", package = "anansi")
kegg_link()
```

## Format

`ec2ko`: a `data.frame` of two columns, named `"ec"` and `"ko"`. The IDs refer to KEGG orthologues. Enzyme commission numbers, `ecs`, typically describe reactions captured by them.

`ec2cpd`: a `data.frame` of two columns, named `"ec"` and `"cpd"`. The IDs refer to compounds in the KEGG database. Enzyme commission numbers, `ecs`, typically describe reactions either producing or requiring them.

## Value

`kegg_link()` returns a list containing the two aforementioned `data.frames`, `ec2cpd` and `ec2ko`.

## Source

`ec2ko`: Adapted from <https://www.genome.jp/kegg/>, using KEGGREST. Script to generate available in example.

`ec2cpd`: Adapted from <https://www.genome.jp/kegg/> using KEGGREST. Script to generate available in example.

## Examples

```
kegg_link()

# Generate ec2ko and ec2cpd:
# Don't download during tests. set to `TRUE` to download.
dry_run <- TRUE

if (!dry_run) {
  ec2ko <- KEGGREST::keggLink("ec", "ko")
  ec2ko <- data.frame(
    ec = gsub("ec:", "", x = ec2ko, fixed = TRUE),
    ko = gsub("ko:", "", x = names(ec2ko), fixed = TRUE),
    row.names = NULL
  )

  ec2cpd <- KEGGREST::keggLink("ec", "cpd")
  ec2cpd <- data.frame(
    ec = gsub("ec:", "", x = ec2cpd, fixed = TRUE),
    cpd = gsub("cpd:", "", x = names(ec2cpd), fixed = TRUE),
    row.names = NULL
  )
}
```

---

FMT\_KOs

*Snippet of the CLR-transformed inferred functional data from the FMT Aging study.*

---

## Description

Piphillin was used to infer functions from the 16S sequencing data in terms of KOs. Unfortunately, the Piphillin algorithm is proprietary and has since been taken down.

## Usage

```
data("FMT_data", package = "anansi")
```

## Format

A matrix object with 6468 rows, KOs, and 36 columns, samples.

## Source

[doi:10.1038/s43587021000939](https://doi.org/10.1038/s43587021000939)

---

FMT_metab	<i>Snippet of the CLR-transformed hippocampal metabolomics data from the FMT Aging study.</i>
-----------	---

---

**Description**

Snippet of the CLR-transformed hippocampal metabolomics data from the FMT Aging study.

**Usage**

```
data("FMT_data", package = "anansi")
```

**Format**

A matrix object with three rows, compounds, and 36 columns, samples.

**Source**

[doi:10.1038/s43587021000939](https://doi.org/10.1038/s43587021000939)

---

FMT_metadata	<i>Snippet of the metadata from the FMT Aging study.</i>
--------------	--

---

**Description**

There were three treatment groups in the study that all received faecal microbiota transplantation (FMT). Young mice that received FMT from young mice (Young yFMT), aged mice that received FMT from aged mice (Aged oFMT) and aged mice that received FMT from young mice (Aged yFMT).

**Usage**

```
data("FMT_data", package = "anansi")
```

**Format**

A data.frame object with 36 rows, samples, and two columns, denoting sample ID and treatment group, respectively.

**Source**

[doi:10.1038/s43587021000939](https://doi.org/10.1038/s43587021000939)

---

getEdgeList	<i>Get a list of edges</i>
-------------	----------------------------

---

**Description**

Get a list of edges

**Usage**

```
getEdgeList(x, ...)
```

**Arguments**

x	input object
...	additional arguments

**Value**

a two-column data.frame that lists the content of each entry in the input MultiFactor

**Examples**

```
x <- randomMultiFactor(n_features = 10)
getEdgeList(x)
```

---

getFeaturePairs	<i>Get a list of all pairs of features</i>
-----------------	--

---

**Description**

Get a list of all pairs of features

**Usage**

```
getFeaturePairs(x, ...)
```

**Arguments**

x	input object
...	additional arguments for specific methods

**Value**

an list of two-column data.frames that represent all feature pairs.

**Examples**

```
x <- randomWeb(10)
head(getFeaturePairs(x), 3)
```

---

getGraph	<i>Get a graph object.</i>
----------	----------------------------

---

**Description**

Get a graph object.

**Usage**

```
getGraph(x, ...)
```

**Arguments**

x	input
...	additional arguments (currently not used).

**Value**

a specified graph object.

**See Also**

[MultiFactor-methods](#)

**Examples**

```
# Show methods
getGraph
```

---

krebs	<i>Simplified snippet of the Krebs cycle</i>
-------	--

---

**Description**

Enzymes and metabolites that make up the Krebs cycle. Features share a row when they play a role in the same reaction. Used in vignettes, exported to facilitate user exploration.

**Usage**

```
data("krebs", package = "anansi")
```

**Format**

A data.frame of two factors, named "Enzyme" and "Metabolite", which contain these respective components of the krebs cycle.

**Source**

Generated by hand based on the krebs cycle.

**See Also**

[krebsDemoWeb\(\)](#)

---

linkBiobakeryMap	<i>make a link data.frame for biobakery mapping files input</i>
------------------	---

---

**Description**

make a link data.frame for biobakery mapping files input

**Usage**

```
linkBiobakeryMap(map)
```

**Arguments**

map                    Character, result from readLines() on uncompressed humann mapping files.

**Value**

a two-column data.frame that can be converted into an adjacency matrix used as input for weaveWeb().

**See Also**

[weaveWeb\(\)](#)

**Examples**

```
# some dummy input, as a character vector of IDs separated by tabs.  
x <- c("x_1\ty_1\ty_2\ty_4", "x_2\ty_1\ty_3", "x_3\ty_2\ty_y")  
linkBiobakeryMap(x)
```

---

metadata	<i>Get metadata.</i>
----------	----------------------

---

**Description**

Get metadata.

**Arguments**

x	input object
...	additional arguments

**Details**

Compatible with S4Vectors generic.

**Value**

metadata slot of x

**Examples**

```
x <- randomWeb(10)
metadata(x)
```

---

metadata<-	<i>Set metadata.</i>
------------	----------------------

---

**Description**

Set metadata.

**Arguments**

x	input object
...	additional arguments
value	replacement value, coerced to data.frame

**Details**

Compatible with S4Vectors generic.

**Value**

x with modified metadata slot.

**Examples**

```
x <- randomWeb(10)
metadata(x) <- cbind(
  metadata(x),
  new_groups = c("A", "B")
)
```

---

MultiFactor

*MultiFactor S7 container class*


---

**Description**

MultiFactor is an S7 class to organize and manage multiple sets of factors, for instance when tracing or converting feature IDs across databases. Methods for MultiFactor aim to follow factor behaviour.

**Usage**

```
MultiFactor(x, levels = NULL, drop.unmatched = FALSE)
```

```
## Constructor for `MultiFactor` objects
MultiFactor(x, levels = NULL, drop.unmatched = FALSE)
```

**Arguments**

x	MultiFactor
levels	an optional named list of vectors of the unique values (as character strings) that x might have taken. The default is the unique set of values taken by lapply(x, as.character), sorted into increasing order of x.
drop.unmatched	Logical scalar If TRUE (Default), for feature types that are seen at least twice, exclude features that only present in one of their respective link data frames.

**Details**

The most straightforward way to construct a MultiFactor object is as a named list of named data.frames. The columns of the data.frames indicate the category of factor in that column.

A MultiFactor object presents itself similar to a data.frame, in the sense that level types can be called as columns and individual data.frame components can be called as rows.

**Value**

a MultiFactor object.

**Slots**

`index` Named list of named integer data frames of at least two columns each. The column names correspond to names in the `levels` slot. Similar to factors, the integers in those columns correspond to the characters in that level. Accessed through regular list methods (e.g., `[`, `[[`).

`levels` Named list of character vectors. Accessed through `levels(x)`

`map` (sparse) Matrix specifying which elements contain which levels. Accessed through `x@dictionary`.

**See Also**

[MultiFactor-methods\(\)](#)

- [kegg\\_link\(\)](#): for an example of valid input.

**Examples**

```
# Generate some random linkage input
x <- data.frame(
  a = sample(letters[seq(3)], 10, replace = TRUE),
  A = sample(LETTERS[seq(3)], 10, replace = TRUE)
)
MultiFactor(x)
```

---

MultiFactor-methods    *Methods for MultiFactor S7 container class*

---

**Description**

`droplevels()` `droplevels(MultiFactor)` returns a `MultiFactor` with unused levels removed, Analogous to the `factor` method.

**Arguments**

<code>...</code>	<code>i, j</code> indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or <code>NULL</code> . Numeric values are coerced to integer or whole numbers as by <code>as.integer</code> or for large values by <code>trunc</code> (and hence truncated towards zero). Character vectors will be matched to the names of the object.
<code>value</code>	Replacement value, typically of same type as that which is to be replaced.
<code>exclude</code>	<code>NULL</code> or Named character list of similar structure as <code>levels(MultiFactor)</code> . Which levels to drop from output.
<code>select</code>	<code>NULL</code> or Named character list of similar structure as <code>levels(MultiFactor)</code> . Which levels to keep in output.
<code>use.names, ignore.mcols</code>	For compatibility, not used.
<code>x</code>	<code>MultiFactor</code>
<code>format</code>	Character scalar, controls output format by package name. "igraph" and "graph" are supported.

**Details**

Only one of select and exclude should be provided, as they are each others complement.

**Value**

A MultiFactor  
a specified graph object.

**See Also**

`igraph::graph_from_data_frame()` and `igraph::as_graphnel()`, which are used under the hood, from `igraph::igraph()` package.

**Examples**

```
# Setup
x <- MultiFactor(kegg_link())
x

# Basic properties
dim(x)
dimnames(x)

# Factor-like properties
head(levels(x)$ko)
droplevels(x)
head(unfactor(x)$ec2ko)

# Extract common output formats
getEdgeList(x)

droplevels(x, exclude = list(ko = "K00001"))
droplevels(x, select = list(ko = "K00001"))
# Generate an igraph object
g <- getGraph(x = kegg_link(), format = "igraph")
plot(g)
```

---

pairwiseApply

*Apply a function on each pair of features*

---

**Description**

Apply a function on each pair of features

**Usage**

```
pairwiseApply(X, ...)
```

**Arguments**

X                   input object  
 ...                 additional arguments

**Value**

a list containing the output of applying the function to each feature pair. See `?base::mapply()`

**Examples**

```
web <- randomWeb(10)

# For each feature pair, was the value for x higher than the value for y?
pairwise_gt <- pairwiseApply(
  X = web,
  FUN = function(x, y) x > y,
  MoreArgs = NULL, SIMPLIFY = FALSE, USE.NAMES = TRUE
)

head(pairwise_gt)

# Run cor.test() on each pair of features
pairwise_cor <- pairwiseApply(
  X = web,
  FUN = function(x, y) cor.test(x, y),
  MoreArgs = NULL, SIMPLIFY = FALSE, USE.NAMES = TRUE
)

pairwise_cor[1]
```

---

plotAnansi

*Dissociation plot*

---

**Description**

`plotAnansi` generates an association plot from the output of `anansi()` in the table format. It provides a convenient way to visually assess relevant results from the `anansi` analysis, either in the form of a dotplot or a graph.

**Arguments**

x                   a data.frame object output of `anansi()` in the table format.  
 ...                 additional parameters  
 layout             Character scalar. Specifies the plot layout to generate. It must be one of `c("dotplot", "graph")`. (Default: `dotplot`)

association.type	Character scalar. Specifies the type of association to show in the plot. One of "disjointed", "emergent" and "full". (Default: NULL)
model.var	Character scalar. Specifies the name of a variable in the anansi model. It is relevant only when association.type is "disjointed" or "emergent". (Default: NULL)
group	Character scalar. Selects one of the groups included in the anansi model. It is relevant only when layout is graph. (Default: All)
signif.threshold	Numeric scalar. Specifies the threshold to mark the significance of association.type. (Default: NULL)
colour_by	Character scalar. Specifies one of the groups terms used in the original anansi call, x by which points should be coloured. (Default: NULL)
color_by	Character scalar. Alias to colour_by.
fill_by	Character scalar. Specifies one of the groups terms used in the original anansi call, x by which points should be filled (Default: "group")
size_by	Character scalar. Specifies one of the groups terms used in the original anansi call, x by which points should be sized. (Default: NULL)
shape_by	Character scalar. Specifies one of the groups terms used in the original anansi call, x by which points should be shaped. (Default: NULL)
x_lab	Character scalar. Specifies the label of the x axis. (Default: "cor")
y_lab	Character scalar. Specifies the label of the y axis. (Default: "")
y_position	Character scalar. Specifies the position of the y labels. It should be either "left" or "right". (Default: "right")
show.cor	Logical scalar. Whether correlation edges should be labelled with correlation coefficients when layout is graph. (Default: FALSE)

## Details

plotAnansi provides a standardised method to visualise the results of anansi by means of a differential association plot. The input for this function should be generated from `anansi()` or `anansi()`, with `return.format = "table"`

## Value

a figure that can be further modified using the ggplot2 suite  
A ggplot2 object.

## See Also

[anansi\(\)](#)

## Examples

```
# Import libraries
library(mia)
library(TreeSummarizedExperiment)
library(MultiAssayExperiment)
```

```

library(ggraph)

web <- randomWeb(n_samples = 100)
mae <- asMAE(web)

# Perform anansi analysis
out <- weaveWeb(mae,
  tableY = "y", tableX = "x"
) |> anansi(formula = ~group_ab)

# Select significant interactions
out <- out[out$full_p.values < 0.05, ]

# Visualise disjointed associations filled by group
plotAnansi(out,
  association.type = "disjointed",
  model.var = "group_ab",
  signif.threshold = 0.05,
  fill_by = "group"
)

# Visualise full associations filled by group
plotAnansi(out,
  association.type = "full",
  signif.threshold = 0.05,
  fill_by = "group"
)

# Visualise full associations as graph
plotAnansi(out,
  layout = "graph",
  association.type = "full",
  signif.threshold = 0.05,
  show.cor = TRUE
)

# Visualise disjointed associations as graph
plotAnansi(out,
  layout = "graph",
  association.type = "disjointed",
  model.var = "group_ab",
  signif.threshold = 0.05
)

```

---

randomAnansi

*Generate a random AnansiWeb or MultiFactor*


---

### Description

Randomly generate a valid AnansiWeb or MultiFactor object.

**Usage**

```
randomWeb(
  n_samples = 10,
  n_reps = 1L,
  n_features_x = 8,
  n_features_y = 12,
  sparseness = 0.5,
  tableY = NULL,
  tableX = NULL,
  dictionary = NULL
)
```

```
randomMultiFactor(n_types = 6, n_features = 100, sparseness = 0.5)
```

```
krebsDemoWeb(n_samples = 100, n_reps = 4L)
```

**Arguments**

<code>n_samples, n_reps</code>	Numeric scalar Number of samples and repeated measures of those samples to be generated. Ignored if <code>tableY</code> and <code>tableX</code> are provided. (defaults: 10 samples without repeats)
<code>n_features_y, n_features_x</code>	Numeric scalar Number of features to be generated. Ignored if <code>tableY</code> and <code>tableX</code> are provided.
<code>sparseness</code>	Numeric scalar, proportion: How rare are connections
<code>tableY, tableX</code>	A table containing features of interest. Rows should be samples and columns should be features. Y and X refer to the position of the features in a formula: $Y \sim X$ .
<code>dictionary</code>	A binary adjacency matrix of class <code>Matrix</code> , or coercible to <code>Matrix</code> .
<code>n_types</code>	Numeric scalar, number of types of features to generate
<code>n_features</code>	Numeric scalar, number of features per type

**Value**

a randomly generated object of the specified class.

**See Also**

[AnansiWeb\(\)](#), [MultiFactor\(\)](#)

**Examples**

```
# Make a random AnansiWeb object
randomWeb()
krebsDemoWeb()
randomMultiFactor()
```

---

tableX	<i>Get tableX</i>
--------	-------------------

---

**Description**

Get tableX

**Usage**

```
tableX(x, ...)
```

**Arguments**

x	input object
...	additional arguments

**Value**

tableX slot of x

**Examples**

```
x <- randomWeb(10)
tableX(x)
```

---

tableY	<i>Get tableY</i>
--------	-------------------

---

**Description**

Get tableY

**Usage**

```
tableY(x, ...)
```

**Arguments**

x	input object
...	additional arguments

**Value**

tableY slot of x

**Examples**

```
x <- randomWeb(10)
tableY(x)
```

---

weaveWeb	<i>Weave an AnansiWeb object</i>
----------	----------------------------------

---

### Description

Weave an AnansiWeb object

### Usage

```
weaveWeb(x, ...)
```

```
weaveKEGG(x, ...)
```

### Arguments

x	input object
...	additional arguments

### Value

an AnansiWeb object, with sparse binary biadjacency matrix with features from y as rows and features from x as columns in dictionary slot.

### See Also

[weaveWeb-methods\(\)](#)

### Examples

```
# Setup demo tables
ec2ko <- kegg_link()[["ec2ko"]]
ec2cpd <- kegg_link()[["ec2cpd"]]

# Basic usage
weaveWeb(cpd ~ ko, link = kegg_link())
weaveWeb(x = "ko", y = "ec", link = ec2ko)
weaveWeb(ec ~ cpd, link = ec2cpd)

# A wrapper is available for kegg ko, ec and cpd data
generic <- weaveWeb(cpd ~ ko, link = kegg_link())
kegg_wrapper <- weaveKEGG(cpd ~ ko)

identical(generic, kegg_wrapper)

# The following are equivalent to transposition:
a <- weaveWeb(ko ~ cpd, link = kegg_link()) |> dictionary()
b <- weaveWeb(cpd ~ ko, link = kegg_link()) |> dictionary()

identical(a, Matrix::t(b))
```

**Description**

Generate a biadjacency matrix, linking the features between two tables. Return an AnansiWeb object which contains all three.

weaveWeb() is for general use and has flexible default settings.

weaveKEGG() is a wrapper that sets link to kegg\_link(). All variants are special cases of weaveWeb().

**Arguments**

x, y	Character scalar, names of feature types that should be linked. Should be found in the column names of link.
link	One of the following: <ul style="list-style-type: none"> <li>• Character scalar with value "none".</li> <li>• data.frame with two columns</li> <li>• list with two such data.frames.</li> </ul>
tableY, tableX	A table containing features of interest. Rows should be samples and columns should be features. Y and X refer to the position of the features in a formula: Y ~ X. < For Bioconductor S4 objects > Character scalar or numeric scalar. Selects experiment corresponding to tableY and tableX from experiments(x) of MultiAssayExperiment object by name or index, name is recommended. (Default slots: Y = 1L, X = 2L).
metadata	Optional data.frame of sample metadata, to be included with output. Can be accessed from AnansiWeb generated by weaveWeb() with output@metadata.
verbose	Logical scalar. Whether to print diagnostic information (Default: TRUE).# @param force_new boolean If x already has a dictionary Matrix in metadata, ignore it and generate a new object anyway? (Default: FALSE).
typeY, typeX	Character scalar or numeric scalar. Selects assay from experiments to tableY and tableX from experiments(x). (Default: 1L - the first assay in that experiment).
experiment1, experiment2	synonymous args to tableY, tableX for compatibility with mia argument style.
assay.type1, assay.type2	synonymous args to typeY, typeX for compatibility with mi argument style.

**Details**

If the link argument is "none", all features will be considered linked. If one or more data.frames, colnames should be as specified in x and y.

**Value**

an AnansiWeb object, with sparse binary biadjacency matrix with features from y as rows and features from x as columns in dictionary slot.

**See Also**

- [AnansiWeb](#): For general constructor and methods.
- [kegg\\_link\(\)](#): For examples of input for link argument.

**Examples**

```
# Setup demo tables, see first vignette.
data(FMT_data)

t1 <- t(FMT_metab)
t2 <- t(FMT_KOs)

# Input objects and syntax:
## define `x` and `y` as characters
web <- weaveWeb(
  x = "ko", y = "cpd", link = kegg_link(),
  tableX = t2, tableY = t1,
  metadata = NULL, verbose = TRUE
)

## define `x` and `y` with a formula
web2 <- weaveWeb(
  x = cpd ~ ko, link = kegg_link(),
  tableX = t2, tableY = t1,
  metadata = NULL, verbose = TRUE
)

identical(web, web2)

# Method for MultiAssayExperiment S4 object
mae <- asMAE(web)
weaveWeb(
  x = mae, link = kegg_link(),
  tableY = "cpd", tableX = "ko",
  force_new = FALSE
)

# Method for TreeSummarizedExperiment S4 object
tse <- asTSE(web)
weaveWeb(
  x = tse, link = kegg_link(),
  tableY = "cpd", tableX = "ko",
  force_new = FALSE
)
```

# Index

- \* **datasets**
  - ec2ko, 16
  - FMT\_KOs, 17
  - FMT\_metab, 18
  - FMT\_metadata, 18
  - krebs, 20
- \* **internal**
  - anansi, 3
  - anansiCor, 8
  - anansiCorPvalue, 8
  - anansiCorTestByGroup, 9
  - anansiDiffCor, 10
- anansi, 3
- anansi(), 4, 8, 9, 26, 27
- anansi-coercion, 6
- anansi-package (anansi), 3
- anansiCor, 8
- anansiCorPvalue, 8
- anansiCorTestByGroup, 9
- anansiCorTestByGroup(), 8, 9
- anansiDiffCor, 10
- AnansiTale, 10
- AnansiTale-class (AnansiTale), 10
- AnansiWeb, 3, 12, 33
- AnansiWeb(), 3, 29
- AnansiWeb-methods, 12, 13
- AnansiWeb-pairwise, 13, 14
- as.data.frame.anansi::AnansiWeb (anansi-coercion), 6
- as.list.anansi::AnansiWeb (anansi-coercion), 6
- as.list.anansi::MultiFactor (anansi-coercion), 6
- as.MAE (anansi-coercion), 6
- as.MultiAssayExperiment (anansi-coercion), 6
- as.TreeSummarizedExperiment (anansi-coercion), 6
- as.TSE (anansi-coercion), 6
- asMAE (anansi-coercion), 6
- asMultiAssayExperiment (anansi-coercion), 6
- asMultiFactor (MultiFactor), 23
- asTreeSummarizedExperiment (anansi-coercion), 6
- asTSE (anansi-coercion), 6
- dictionary, 15
- dictionary-generic (dictionary), 15
- dim.anansi::AnansiWeb (AnansiWeb-methods), 13
- dimnames.anansi::AnansiWeb (AnansiWeb-methods), 13
- ec2cpd (ec2ko), 16
- ec2ko, 16
- FMT\_KOs, 17
- FMT\_metab, 18
- FMT\_metadata, 18
- getEdgeList, 19
- getFeaturePairs, 19
- getGraph, 20
- getGraph(), 3
- getGraph.anansi::MultiFactor (MultiFactor-methods), 24
- igraph::as\_graphnel(), 25
- igraph::graph\_from\_data\_frame(), 25
- igraph::igraph, 3
- igraph::igraph(), 25
- kegg\_link (ec2ko), 16
- kegg\_link(), 3, 12, 24, 33
- krebs, 20
- krebsDemoWeb (randomAnansi), 28
- krebsDemoWeb(), 21
- levels.anansi::MultiFactor (MultiFactor-methods), 24

linkBiobakeryMap, 21

metadata, 22

metadata, anansi::AnansiWeb-method  
(metadata), 22

metadata.set (metadata<-), 22

metadata<-, 22

metadata<-, anansi::AnansiWeb-method  
(metadata<-), 22

miaViz::miaViz, 3

MultiFactor, 3, 23

MultiFactor(), 3, 29

MultiFactor-methods, 24

names.anansi::AnansiWeb  
(AnansiWeb-methods), 13

pairs.anansi::AnansiWeb  
(AnansiWeb-pairwise), 14

pairs.AnansiWeb (AnansiWeb-pairwise), 14

pairwiseApply, 25

pairwiseApply-generic (pairwiseApply),  
25

pairwiseApply.anansi::AnansiWeb  
(AnansiWeb-pairwise), 14

pairwiseApply.AnansiWeb  
(AnansiWeb-pairwise), 14

plotAnansi, 26

plotAnansi(), 3

plotAnansi-generic (plotAnansi), 26

plotAnansi-methods (plotAnansi), 26

randomAnansi, 3, 12, 28

randomMultiFactor (randomAnansi), 28

randomWeb (randomAnansi), 28

show.anansi::AnansiWeb  
(AnansiWeb-methods), 13

tableX, 30

tableX-generic (tableX), 30

tableY, 30

tableY-generic (tableY), 30

unfactor (MultiFactor-methods), 24

unfactor(), 7

unfactor, anansi::MultiFactor-method  
(MultiFactor-methods), 24

weaveKEGG (weaveWeb), 31

weaveWeb, 31

weaveWeb(), 13, 21

weaveWeb-methods, 32

weaveWeb.character (weaveWeb-methods),  
32

weaveWeb.formula (weaveWeb-methods), 32

weaveWeb.MultiAssayExperiment  
(weaveWeb-methods), 32

weaveWeb.SingleCellExperiment  
(weaveWeb-methods), 32

weaveWeb.TreeSumarizedExperiment  
(weaveWeb-methods), 32