

# Package ‘CorNetto’

October 1, 2026

**Title** Knowledge-Guided Multi-Omic Correlation Network Analysis

**Version** 0.99.1

**Description** Builds knowledge-guided multi-omic correlation networks from normalized transcriptomic, proteomic, and metabolomic abundance data. Group-specific correlation networks are compared with the Fisher z-difference test, either across all within-omic pairs in one or every assay without generating cross-omic pairs, or across candidate edges supplied by a prior-knowledge network, and the resulting differential network is summarized into node-level rewiring scores that identify features whose interaction patterns change between groups. Rewiring scores can be compared with group-label permutation reference distributions, and networks can be restricted to pathway-focused neighbourhoods, converted to 'igraph' objects, or exported as Cytoscape-ready node and edge tables. The package is built on Bioconductor containers so that multi-omic assays and sample metadata are managed consistently throughout the workflow.

**License** Artistic-2.0

**Encoding** UTF-8

**Depends** R (>= 4.6.0)

**Imports** BiocParallel, graphics, igraph, methods, MultiAssayExperiment, S4Vectors, stats, SummarizedExperiment, tools, utils, withr

**Suggests** BiocManager, BiocStyle, knitr, qvalue, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**biocViews** GraphAndNetwork, Metabolomics, Network, NetworkInference, Proteomics, SystemsBiology, Transcriptomics

**URL** <https://github.com/bradleyalexward/CorNetto>

**BugReports** <https://github.com/bradleyalexward/CorNetto/issues>

**Config/testthat/edition** 3

**Roxygen** list(markdown = TRUE)

**Config/roxygen2/version** 8.0.0

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

**git\_branch** devel

**git\_last\_commit** 20ca097

**git\_last\_commit\_date** 2026-08-25

**Repository** Bioconductor 3.24

**Date/Publication** 2026-10-01

**Author** Bradley Ward [aut, cre] (ORCID:  
<https://orcid.org/0000-0003-0778-0153>),  
 Jean-Luc Balligand [ctb],  
 Laurence Bamps [ctb],  
 Patrice D. Cani [ctb],  
 Julien De Greef [ctb],  
 Joseph P. Dewulf [ctb],  
 Vincent Haufroid [ctb],  
 Benoît Kabamba [ctb],  
 Sébastien Pyr dit Ruys [ctb],  
 Didier Vertommen [ctb],  
 Jean Cyr Yombi [ctb],  
 Leïla Belkhir [ths, fnd],  
 Laure Elens [ths, fnd],  
 Sofina COVID Solidarity Fund [fnd]

**Maintainer** Bradley Ward <bradleyalexward@gmail.com>

## Contents

|                                                |    |
|------------------------------------------------|----|
| CorNetto-package . . . . .                     | 3  |
| calculateRewiringScores . . . . .              | 4  |
| combineKnowledgeNetworks . . . . .             | 6  |
| combineNetworks . . . . .                      | 6  |
| createAnalysisData . . . . .                   | 8  |
| createCorrelationNetwork . . . . .             | 9  |
| createDifferentialCorrelationNetwork . . . . . | 11 |
| createFocusedNetwork . . . . .                 | 12 |
| createNetworkGraph . . . . .                   | 13 |
| exampleAnalysisData . . . . .                  | 14 |
| exampleKnowledgeNetwork . . . . .              | 15 |
| exampleSeedNodes . . . . .                     | 15 |
| filterFeatures . . . . .                       | 16 |
| filterNetworkByNodes . . . . .                 | 17 |
| filterSamples . . . . .                        | 17 |
| getCorNettoResult . . . . .                    | 18 |
| permuteRewiringScores . . . . .                | 19 |
| plotRewiringScores . . . . .                   | 22 |
| prepareCytoscapeTables . . . . .               | 24 |
| readKnowledgeNetwork . . . . .                 | 24 |
| summarizeAnalysisData . . . . .                | 25 |
| summarizeKnowledgeNetwork . . . . .            | 26 |
| testDifferentialCorrelation . . . . .          | 27 |
| validateAnalysisData . . . . .                 | 29 |
| validateKnowledgeNetwork . . . . .             | 30 |
| writeNetworkTables . . . . .                   | 31 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>32</b> |
|--------------|-----------|

---

CorNetto-package*CorNetto: Knowledge-Guided Multi-Omic Correlation Network Analysis*

---

## Description

CorNetto provides a Bioconductor-oriented workflow for constructing knowledge-guided correlation networks from normalized multi-omic abundance data. The package is designed around `MultiAssayExperiment::MultiAssayExperiment()` so that multiple assays and their shared sample metadata can be analysed consistently.

The main workflow includes:

- validation of normalized assay inputs
- pre-analysis summaries of missingness, sample counts, and feature counts
- sample and feature filtering before network construction
- group-specific correlation networks
- differential correlation, over all pairs in an assay or over prior-supported candidate edges
- rewiring score calculation
- permutation-based assessment of rewiring scores
- pathway-focused network extraction
- graph creation and Cytoscape-ready export

The vignette (`vignette("CorNetto")`) opens with a diagram of how these fit together.

## Value

No return value. This package-level help page documents the CorNetto package and points users to the main workflow functions.

## Author(s)

**Maintainer:** Bradley Ward <bradleyalexward@gmail.com> ([ORCID](#))

Authors:

- Bradley Ward <bradleyalexward@gmail.com> ([ORCID](#))

Other contributors:

- Jean-Luc Balligand [contributor]
- Laurence Bamps [contributor]
- Patrice D. Cani [contributor]
- Julien De Greef [contributor]
- Joseph P. Dewulf [contributor]
- Vincent Haufroid [contributor]
- Benoît Kabamba [contributor]
- Sébastien Pyr dit Ruys [contributor]
- Didier Vertommen [contributor]

- Jean Cyr Yombi [contributor]
- Leïla Belkhir <leila.belkhir@saintluc.uclouvain.be> [thesis advisor, funder]
- Laure Elens <laure.elens@uclouvain.be> [thesis advisor, funder]
- Sofina COVID Solidarity Fund [funder]

## References

Ward B, Balligand J-L, Bamps L, Bommer G, Cani PD, De Greef J, Dewulf JP, Gatto L, Haufroid V, Kabamba B, Pyr dit Ruys S, Vertommen D, Yombi JC, Belkhir L, Elens L (2026). Longitudinal multi-omic network rewiring at the complement-coagulation interface in post-acute sequelae of COVID-19 (PASC). *medRxiv*. doi:10.64898/2026.07.14.26358048

## See Also

Useful links:

- <https://github.com/bradleyalexward/CorNetto>
- Report bugs at <https://github.com/bradleyalexward/CorNetto/issues>

---

calculateRewiringScores

*Calculate Node Rewiring Scores*

---

## Description

Calculate raw, RMS-normalized, and degree-matched rewiring scores from a weighted differential-correlation network.

## Usage

```
calculateRewiringScores(
  differentialCorrelationNetwork,
  analysisData = NULL,
  resultName = "rewiringScores",
  storeResult = FALSE
)
```

## Arguments

|                                |                                                                  |
|--------------------------------|------------------------------------------------------------------|
| differentialCorrelationNetwork | A standardized differential correlation edge table.              |
| analysisData                   | Optional MultiAssayExperiment used to store the rewiring scores. |
| resultName                     | Name used when storing the rewiring scores.                      |
| storeResult                    | Whether to store the result in analysisData.                     |

## Details

These are descriptive CorNetto scores, not tests. None of them is a p-value, and degreeMatchedZScore is not a z-score against any null model. `permuteRewiringScores()` supplies a randomization p-value only under its documented fixed-universe and exchangeability conditions; otherwise it returns conditional rankings.

For a node with incident differential z-scores  $z_1, \dots, z_k$ :

- `rawRewiringScore` is the L2 norm  $\sqrt{\sum(z_i^2)}$ . It grows with degree: if the  $z_i$  were independent standard normals,  $\sum(z_i^2)$  would follow a chi-square distribution on  $k$  degrees of freedom. Its squared score would then have expectation  $k$ , and the score itself would grow roughly as  $\sqrt{k}$ . Edges sharing a node are not independent, so treat that only as intuition for the scaling.
- `rootMeanSquareRewiringScore` divides by  $\sqrt{k}$ , which removes most of that degree dependence. Under the same rough argument its squared value has expectation 1; the score itself is biased below 1 at small  $k$  and approaches 1 as degree grows. This is the most comparable score across nodes of different degree.
- `degreeMatchedZScore` standardizes `rawRewiringScore` within bins of  $\log_2(\text{degree} + 1)$ . It is a within-bin ranking aid only. Roughly half the nodes in every bin are negative by construction, and the scale is not comparable across bins, so a bin holding few nodes cannot produce a meaningful spread. Bins with fewer than five nodes, or with no spread, return NA.

The network must contain one row per edge and no self-loops. Duplicate or mirrored edges would change both the degree and the score, so the function stops when either is detected. Score the network before calling `combineNetworks()` with `includeReverseEdges = TRUE`.

## Value

A `DataFrame` of rewiring scores or an updated `MultiAssayExperiment`.

## Examples

```
analysisData <- exampleAnalysisData()
differentialTable <- testDifferentialCorrelation(
  analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = c("Recovered", "PASC"),
  assayName = "protein",
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  pAdjustMethod = "fdr",
  storeResult = FALSE
)
differentialNetwork <- createDifferentialCorrelationNetwork(
  differentialTable,
  minimumAbsoluteCorrelation = 0
)
rewiringScores <- calculateRewiringScores(differentialNetwork)
rewiringScores
```

---

 combineKnowledgeNetworks

*Combine Multiple Knowledge Networks*


---

### Description

Row-bind and standardize multiple user-supplied prior-knowledge networks.

### Usage

```
combineKnowledgeNetworks(
  ...,
  removeDuplicates = TRUE,
  analysisData = NULL,
  resultName = "combinedKnowledgeNetwork",
  storeResult = FALSE
)
```

### Arguments

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| ...              | Knowledge-network objects or lists of knowledge-network objects.  |
| removeDuplicates | Whether to remove duplicated interactions.                        |
| analysisData     | Optional MultiAssayExperiment used to store the combined network. |
| resultName       | Name used when storing the combined network.                      |
| storeResult      | Whether to store the result in analysisData.                      |

### Value

A standardized DataFrame or an updated MultiAssayExperiment.

### Examples

```
knowledgeNetwork <- exampleKnowledgeNetwork()
combined <- combineKnowledgeNetworks(
  knowledgeNetwork[1:3, ],
  knowledgeNetwork[4:6, ]
)
combined
```

---

 combineNetworks

*Combine Network Edge Tables*


---

### Description

Combine knowledge edges, correlation edges, differential-correlation edges, or other standardized CorNetto edge tables into one network.

## Usage

```
combineNetworks(  
  analysisData = NULL,  
  knowledgeNetwork = NULL,  
  correlationNetwork = NULL,  
  differentialCorrelationNetwork = NULL,  
  networkList = NULL,  
  includeReverseEdges = TRUE,  
  removeDuplicates = TRUE,  
  resultName = "combinedNetwork",  
  storeResult = FALSE  
)
```

## Arguments

|                                |                                                                                                                                            |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| analysisData                   | Optional MultiAssayExperiment used to retrieve stored CorNetto results and optionally store the combined network.                          |
| knowledgeNetwork               | Optional knowledge-network edge table or list of edge tables.                                                                              |
| correlationNetwork             | Optional correlation edge table or list of edge tables.                                                                                    |
| differentialCorrelationNetwork | Optional differential correlation edge table or list of edge tables.                                                                       |
| networkList                    | Optional additional edge table or list of edge tables.                                                                                     |
| includeReverseEdges            | Whether to duplicate undirected edges to support directed graph traversal. See Details before enabling it for a table you intend to score. |
| removeDuplicates               | Whether to remove exactly duplicated edge rows after standardization.                                                                      |
| resultName                     | Name used when storing the combined network.                                                                                               |
| storeResult                    | Whether to store the result in analysisData.                                                                                               |

## Details

includeReverseEdges = TRUE appends a reversed copy of every undirected edge. That is what you want for traversal and for export to tools that expect an explicit adjacency list, and it is the default for that reason.

It is not what you want for anything that treats the table as a set of edges. `calculateRewiringScores()` rejects duplicate or mirrored edges because they would double every node's degree and multiply rawRewiringScore by  $\sqrt{2}$ . An undirected `createNetworkGraph()` built from a mirrored table gains parallel edges. Score first, then combine with reverse edges for export, or pass includeReverseEdges = FALSE and let `createNetworkGraph()` handle direction.

## Value

A standardized edge DataFrame or an updated MultiAssayExperiment.

## Examples

```
analysisData <- exampleAnalysisData()
knowledgeNetwork <- exampleKnowledgeNetwork()
differentialResults <- testDifferentialCorrelation(
  analysisData = analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = c("Recovered", "PASC"),
  assayName = "protein",
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  pAdjustMethod = "fdr",
  storeResult = FALSE
)
differentialNetwork <- createDifferentialCorrelationNetwork(
  differentialResults,
  minimumAbsoluteCorrelation = 0
)
combinedNetwork <- combineNetworks(
  knowledgeNetwork = knowledgeNetwork,
  differentialCorrelationNetwork = differentialNetwork
)
combinedNetwork
```

---

|                    |                                          |
|--------------------|------------------------------------------|
| createAnalysisData | <i>Create a CorNetto Analysis Object</i> |
|--------------------|------------------------------------------|

---

## Description

Create a MultiAssayExperiment from normalized assay objects and sample metadata.

## Usage

```
createAnalysisData(assayList, sampleData)
```

## Arguments

|            |                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------|
| assayList  | A named list of assays. Each element may be a matrix, data frame, or SummarizedExperiment.                                   |
| sampleData | Sample metadata with row names corresponding to sample identifiers. A sampleId column may be used when row names are absent. |

## Details

CorNetto expects abundances that are already normalized; no transformation is applied here. Each assay contributes exactly one matrix. A SummarizedExperiment carrying several matrices must either name the one to use abundance or be subset beforehand, otherwise createAnalysisData() stops rather than guessing. Only the selected matrix and the featureIdentifier/featureName columns of rowData are retained; other rowData columns, assay-level colData, and rowRanges are dropped.

Missing values are permitted and are never imputed. Correlations are computed on pairwise complete observations, so each edge carries its own effective sample size in the sampleCount, group1SampleCount, and group2SampleCount columns. Non-finite values (Inf, NaN) are rejected.

**Value**

A validated MultiAssayExperiment.

**Examples**

```
protein <- matrix(
  seq_len(12),
  nrow = 3,
  dimnames = list(paste0("P", 1:3), paste0("S", 1:4))
)
sampleData <- data.frame(
  sampleId = paste0("S", 1:4),
  group = rep(c("A", "B"), each = 2)
)
analysisData <- createAnalysisData(list(protein = protein), sampleData)
analysisData
```

---

createCorrelationNetwork

*Create a Group-Specific Correlation Network*

---

**Description**

Create a within-omic correlation network for one assay and one sample group. All pairs in the assay are tested, which is intended for small-to-medium assays. For large assays, restrict the feature set with `filterFeatures()` or `featureSubset`, or work from prior-supported pairs with `testDifferentialCorrelation(candidateEdgeTable = ...)`.

**Usage**

```
createCorrelationNetwork(
  analysisData,
  assayName,
  groupColumn = NULL,
  groupLevel = NULL,
  sampleIds = NULL,
  featureSubset = NULL,
  correlationMethod = c("pearson", "spearman", "kendall"),
  minimumAbsoluteCorrelation = 0.3,
  adjustedPValueThreshold = 0.05,
  pAdjustMethod = "fdr",
  featureNameColumn = "featureName",
  resultName = NULL,
  storeResult = TRUE
)
```

**Arguments**

|                           |                                                  |
|---------------------------|--------------------------------------------------|
| <code>analysisData</code> | A MultiAssayExperiment.                          |
| <code>assayName</code>    | Name of the assay to analyse.                    |
| <code>groupColumn</code>  | Sample metadata column used to define the group. |

|                            |                                                                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| groupLevel                 | Single group label to analyse.                                                                                                                     |
| sampleIds                  | Optional explicit sample identifiers.                                                                                                              |
| featureSubset              | Optional feature identifiers to retain.                                                                                                            |
| correlationMethod          | Correlation method. One of "pearson", "spearman", or "kendall".                                                                                    |
| minimumAbsoluteCorrelation | Minimum absolute correlation retained in the final network.                                                                                        |
| adjustedPValueThreshold    | Maximum adjusted p-value retained in the final network.                                                                                            |
| pAdjustMethod              | Multiple-testing correction method. Use one of <code>stats::p.adjust.methods</code> or "qvalue"; the latter requires the suggested qvalue package. |
| featureNameColumn          | Row-data column containing display names.                                                                                                          |
| resultName                 | Optional name used when storing the result.                                                                                                        |
| storeResult                | Whether to store the result in <code>metadata(analysisData)\$cornetto</code> .                                                                     |

## Details

Pearson p-values come from the t statistic  $t = r \cdot \sqrt{(n-2)/(1-r^2)}$  on  $n-2$  degrees of freedom, matching `stats::cor.test()`. Spearman and Kendall p-values are taken from `stats::cor.test()` with `exact = FALSE`, using its asymptotic approximations. Tied ranks are permitted; Kendall's estimate is tau-b.

Correlations use pairwise complete observations, so  $n$  varies by pair and is reported per edge in `sampleCount`. Pairs with fewer than three shared observations, and pairs where either feature is constant, have no test and are dropped rather than reported as non-significant.

The Pearson t test assumes independent observations and approximate bivariate normality for each pair. Pairwise deletion is defensible only when the missingness process is ignorable for the correlation of interest; abundance-dependent missingness in proteomics or metabolomics can violate that condition. Rank correlations reduce sensitivity to outliers but do not remove the independence or missingness assumptions.

## Value

A standardized edge `DataFrame` or an updated `MultiAssayExperiment`.

## References

- Benjamini Y, Hochberg Y (1995). Controlling the false discovery rate. *Journal of the Royal Statistical Society B*, 57, 289-300. doi:[10.1111/j.25176161.1995.tb02031.x](https://doi.org/10.1111/j.25176161.1995.tb02031.x)
- Storey JD, Tibshirani R (2003). Statistical significance for genomewide studies. *PNAS*, 100, 9440-9445. doi:[10.1073/pnas.1530509100](https://doi.org/10.1073/pnas.1530509100)

## Examples

```
analysisData <- exampleAnalysisData()
correlationNetwork <- createCorrelationNetwork(
  analysisData,
  assayName = "protein",
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  storeResult = FALSE
```

```
)
correlationNetwork
```

---

```
createDifferentialCorrelationNetwork
```

*Create a Weighted Differential Correlation Network*

---

## Description

Convert the output of `testDifferentialCorrelation()` into a weighted edge table with rewiring-oriented labels.

## Usage

```
createDifferentialCorrelationNetwork(
  differentialCorrelationTable,
  differenceAdjustedPValueThreshold = NULL,
  minimumAbsoluteCorrelation = 0.3,
  edgeWeightMethod = c("signedZScore", "absoluteZScore"),
  analysisData = NULL,
  resultName = "differentialCorrelationNetwork",
  storeResult = FALSE
)
```

## Arguments

|                                                |                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>differentialCorrelationTable</code>      | Output of <code>testDifferentialCorrelation()</code> .                                                                                                                                                                                                                                                         |
| <code>differenceAdjustedPValueThreshold</code> | Optional maximum adjusted differential-correlation p-value retained in the network. The default NULL keeps all rewiring-scoring edges returned by <code>testDifferentialCorrelation()</code> .                                                                                                                 |
| <code>minimumAbsoluteCorrelation</code>        | Minimum absolute correlation used to classify gains, losses, sign flips, and strengthening or weakening. Note that this is independent of the prefilter applied by <code>testDifferentialCorrelation()</code> : if you change one, change the other, or the labels will disagree with which edges are present. |
| <code>edgeWeightMethod</code>                  | Method used to generate the edge weight. "signedZScore" keeps the sign of <code>zScoreDifference</code> ; "absoluteZScore" takes its magnitude. See Details for what the sign means.                                                                                                                           |
| <code>analysisData</code>                      | Optional <code>MultiAssayExperiment</code> used to store the weighted differential network.                                                                                                                                                                                                                    |
| <code>resultName</code>                        | Name used when storing the weighted differential network.                                                                                                                                                                                                                                                      |
| <code>storeResult</code>                       | Whether to store the result in <code>analysisData</code> .                                                                                                                                                                                                                                                     |

## Details

edgeWeightMethod does not affect any rewiring score, because `calculateRewiringScores()` squares the weight. It matters only when you read or export edgeWeight directly.

Under "signedZScore" the weight inherits the Fisher statistic's group-2-minus-group-1 orientation: a **positive** weight means the pair is more strongly correlated in the second of the groupLevels passed to `testDifferentialCorrelation()`. The edgeDirection labels run the other way, being named from group 1's point of view, so gainInGroup1 edges carry negative weights.

edgeDirection is NA when either group has no correlation to compare, which happens when a feature is constant within that group.

## Value

A standardized edge DataFrame, or the updated MultiAssayExperiment when storeResult = TRUE.

## Examples

```
analysisData <- exampleAnalysisData()
differentialTable <- testDifferentialCorrelation(
  analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = c("Recovered", "PASC"),
  assayName = "protein",
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  pAdjustMethod = "fdr",
  storeResult = FALSE
)
differentialNetwork <- createDifferentialCorrelationNetwork(
  differentialTable,
  minimumAbsoluteCorrelation = 0
)
differentialNetwork
```

---

createFocusedNetwork    *Create a Focused Network Around Seed Nodes*

---

## Description

Build a neighborhood-induced subnetwork around a set of seed nodes. Focused-network outputs retain an explicit node table in metadata so isolated retained nodes remain available to downstream graph and export helpers.

## Usage

```
createFocusedNetwork(
  networkEdgeTable,
  seedNodes,
  neighborhoodOrder = 1L,
  mode = c("all", "out", "in"),
  nodeIdType = c("identifier", "key"),
  dropIsolatedNodes = FALSE
)
```

**Arguments**

|                   |                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------|
| networkEdgeTable  | A standardized edge table.                                                                                     |
| seedNodes         | Seed-node identifiers or node keys.                                                                            |
| neighborhoodOrder | Number of graph steps to expand.                                                                               |
| mode              | Neighborhood mode passed to <code>igraph::ego()</code> . One of "all", "out", or "in".                         |
| nodeIdType        | Whether seedNodes are assay-local feature identifiers ("identifier") or assay-qualified node keys ("key").     |
| dropIsolatedNodes | Whether to remove nodes retained in the focused subgraph that do not participate in any of the returned edges. |

**Value**

A standardized edge DataFrame.

**Examples**

```
knowledgeNetwork <- exampleKnowledgeNetwork()
focusedNetwork <- createFocusedNetwork(
  knowledgeNetwork,
  seedNodes = exampleSeedNodes()
)
focusedNetwork
```

---

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| createNetworkGraph | <i>Create an igraph Object from a CorNetto Edge Table</i> |
|--------------------|-----------------------------------------------------------|

---

**Description**

Stored node metadata is used when present so isolated nodes can be preserved across graph creation and export.

**Usage**

```
createNetworkGraph(
  networkEdgeTable,
  nodeTable = NULL,
  directed = NULL,
  mirrorUndirectedEdges = TRUE
)
```

**Arguments**

|                  |                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------|
| networkEdgeTable | A standardized edge table.                                                                                       |
| nodeTable        | Optional node table. When NULL, the stored node table is used when present and otherwise derived from the edges. |

**directed** Optional logical value. When NULL, the graph is directed when any edge is directed.

**mirrorUndirectedEdges** Whether to add a reverse copy of each undirected edge when the graph is directed. `igraph` cannot mix directed and undirected edges in one object, so without this an undirected edge in a directed graph is only traversable one way.

## Details

`igraph` graphs are either directed or undirected as a whole. When a table mixes both, the graph is built directed and each undirected edge is added in both directions so that `mode = "in"` and `mode = "out"` traversal treat it symmetrically. Set `mirrorUndirectedEdges = FALSE` to keep a one-to-one correspondence between rows and graph edges, at the cost of asymmetric neighbourhoods.

All standardized edge columns become edge attributes. Nonstandard columns in a supplied or stored node table are retained as vertex attributes.

Note that an edge table already carrying reverse copies, as produced by `combineNetworks(includeReverseEdges = TRUE)`, will yield parallel edges in an undirected graph. Prefer `includeReverseEdges = FALSE` and let this function handle direction.

## Value

An `igraph` object.

## Examples

```
knowledgeNetwork <- exampleKnowledgeNetwork()
graph <- createNetworkGraph(knowledgeNetwork)
graph
```

---

`exampleAnalysisData`      *Load the Synthetic Example Analysis Data*

---

## Description

Load the Synthetic Example Analysis Data

## Usage

```
exampleAnalysisData()
```

## Value

A `MultiAssayExperiment`.

## Examples

```
analysisData <- exampleAnalysisData()
names(MultiAssayExperiment::experiments(analysisData))
```

---

`exampleKnowledgeNetwork`*Load the Synthetic Example Knowledge Network*

---

**Description**

Load the Synthetic Example Knowledge Network

**Usage**

```
exampleKnowledgeNetwork()
```

**Value**

A standardized knowledge-network DataFrame.

**Examples**

```
knowledgeNetwork <- exampleKnowledgeNetwork()
nrow(knowledgeNetwork)
```

---

`exampleSeedNodes`*Load the Synthetic Example Seed Nodes*

---

**Description**

Load the Synthetic Example Seed Nodes

**Usage**

```
exampleSeedNodes()
```

**Value**

A character vector of seed-node identifiers.

**Examples**

```
seedNodes <- exampleSeedNodes()
seedNodes
```

---

|                |                                                    |
|----------------|----------------------------------------------------|
| filterFeatures | <i>Filter Features Before Network Construction</i> |
|----------------|----------------------------------------------------|

---

## Description

Filter one or more assays to reduce network complexity.

## Usage

```
filterFeatures(
  analysisData,
  assayNames = NULL,
  featureSubset = NULL,
  topVariableFeatures = NULL,
  minimumVariance = NULL,
  removeZeroVariance = FALSE
)
```

## Arguments

|                     |                                                                                                                                                                       |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analysisData        | A MultiAssayExperiment.                                                                                                                                               |
| assayNames          | Optional assay names to filter. When NULL, all assays are filtered.                                                                                                   |
| featureSubset       | Optional vector or named list of feature identifiers to retain.                                                                                                       |
| topVariableFeatures | Optional scalar or named list giving the number of most variable features to retain per assay.                                                                        |
| minimumVariance     | Optional scalar or named list defining the minimum variance required for retention. Features are retained when their variance is greater than or equal to this value. |
| removeZeroVariance  | Logical scalar or named list. When TRUE, remove features with zero or undefined variance before applying minimumVariance or topVariableFeatures.                      |

## Value

A filtered MultiAssayExperiment.

## Examples

```
analysisData <- exampleAnalysisData()
filtered <- filterFeatures(
  analysisData,
  assayNames = "protein",
  topVariableFeatures = 3
)
filtered
```

---

|                      |                                            |
|----------------------|--------------------------------------------|
| filterNetworkByNodes | <i>Filter a Network by Node Membership</i> |
|----------------------|--------------------------------------------|

---

**Description**

Retain only edges touching selected nodes or only edges where both endpoints belong to a selected node set.

**Usage**

```
filterNetworkByNodes(
  networkEdgeTable,
  nodes,
  mode = c("either", "both"),
  nodeIdType = c("identifier", "key")
)
```

**Arguments**

|                  |                                                                                                        |
|------------------|--------------------------------------------------------------------------------------------------------|
| networkEdgeTable |                                                                                                        |
| nodes            | A standardized edge table.                                                                             |
| mode             | Node identifiers or node keys to retain.                                                               |
| nodeIdType       | Either "either" or "both".                                                                             |
|                  | Whether nodes are assay-local feature identifiers ("identifier") or assay-qualified node keys ("key"). |

**Value**

A standardized edge DataFrame.

**Examples**

```
knowledgeNetwork <- exampleKnowledgeNetwork()
filterNetworkByNodes(knowledgeNetwork, nodes = c("P1", "P2"))
```

---

|               |                                                   |
|---------------|---------------------------------------------------|
| filterSamples | <i>Filter Samples Before Network Construction</i> |
|---------------|---------------------------------------------------|

---

**Description**

Filter a CorNetto analysis object to selected samples while allowing assays to retain different overlapping subsets of those samples.

**Usage**

```
filterSamples(
  analysisData,
  sampleIds = NULL,
  groupColumn = NULL,
  groupLevels = NULL,
  dropUnusedLevels = TRUE
)
```

**Arguments**

analysisData    A MultiAssayExperiment.  
 sampleIds       Optional sample identifiers to retain.  
 groupColumn     Optional sample metadata column used to retain samples by group membership.  
 groupLevels     Optional group labels to retain from groupColumn. When NULL and groupColumn is supplied, samples with non-missing and non-empty values in groupColumn are retained.  
 dropUnusedLevels    Whether to drop unused factor levels in sample metadata after filtering.

**Value**

A filtered MultiAssayExperiment.

**Examples**

```

analysisData <- exampleAnalysisData()
filtered <- filterSamples(
  analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = "PASC"
)
summarizeAnalysisData(
  filtered,
  groupColumn = "clinicalGroup",
  quiet = TRUE
)$samples
  
```

---

|                   |                                    |
|-------------------|------------------------------------|
| getCorNettoResult | <i>Get Stored CorNetto Results</i> |
|-------------------|------------------------------------|

---

**Description**

Retrieve the complete CorNetto result store, a result-type slot, or a single named stored result from a MultiAssayExperiment.

**Usage**

```

getCorNettoResult(analysisData, resultType = NULL, resultName = NULL)

knowledgeNetworks(analysisData)

correlationResults(analysisData)

differentialCorrelationResults(analysisData)

differentialCorrelationNetworks(analysisData)

rewiringResults(analysisData)

validationResults(analysisData)

integratedNetworks(analysisData)
  
```

**Arguments**

|              |                                                                                                                                                                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analysisData | A MultiAssayExperiment object.                                                                                                                                                                                                                                                                           |
| resultType   | Optional result type. When NULL, the complete result store is returned unless resultName is supplied. Common values include "knowledgeNetworks", "correlationResults", "differentialCorrelationResults", "differentialCorrelationNet", "rewiringResults", "validationResults", and "integratedNetworks". |
| resultName   | Optional stored result name. When supplied with resultType, a single result is returned. When supplied without resultType, all result slots are searched and the result name must be unique.                                                                                                             |

**Value**

A named list of stored CorNetto results, a single result-type list, or one stored result object.

**Examples**

```
analysisData <- exampleAnalysisData()
analysisData <- validateKnowledgeNetwork(
  exampleKnowledgeNetwork(),
  analysisData = analysisData,
  storeResult = TRUE
)
names(getCorNettoResult(analysisData))
names(knowledgeNetworks(analysisData))
getCorNettoResult(analysisData, "knowledgeNetworks", "knowledgeNetwork")
```

---

permuteRewiringScores *Permute Rewiring Scores*

---

**Description**

Permute group labels, rerun differential correlation testing, rebuild the differential network, and compare observed node scores with their permutation distributions. When both assayName and candidateEdgeTable are NULL, every assay is tested densely within-omic; no cross-omic pairs are generated.

**Usage**

```
permuteRewiringScores(
  analysisData,
  groupColumn,
  groupLevels,
  assayName = NULL,
  candidateEdgeTable = NULL,
  featureSubset = NULL,
  correlationMethod = c("pearson", "spearman"),
  minimumAbsoluteCorrelation = 0.3,
  adjustedPValueThreshold = 0.05,
  pAdjustMethod = "fdr",
  featureNameColumn = "featureName",
  differenceAdjustedPValueThreshold = NULL,
```

```

edgeWeightMethod = c("signedZScore", "absoluteZScore"),
scoreColumn = "rawRewiringScore",
nPermutations = 1000L,
blockColumn = NULL,
seed = NULL,
keepPermutationScores = FALSE,
resultName = NULL,
storeResult = FALSE,
verbose = FALSE,
progressEvery = 300L,
BPPARAM = BiocParallel::SerialParam()
)

```

## Arguments

|                                   |                                                                                                                                                                                                                                                    |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analysisData                      | A MultiAssayExperiment.                                                                                                                                                                                                                            |
| groupColumn                       | Sample metadata column used to define the two groups.                                                                                                                                                                                              |
| groupLevels                       | A character vector of length two giving the group labels. The first group is treated as the reference for gain or loss labels.                                                                                                                     |
| assayName                         | Optional assay name used when testing all within-omic pairs. When NULL, every assay is tested and the results are combined. Ignored when candidateEdgeTable is supplied.                                                                           |
| candidateEdgeTable                | Optional candidate edges to test. May be a standardized CorNetto edge table or a validated knowledge network. Candidate self-loops are omitted with a warning because a feature's correlation with itself is not an informative differential edge. |
| featureSubset                     | Optional feature subset. In all-assay mode, a vector is intersected with each assay, or a named list may supply a separate subset for each assay. Assays absent from a named list use all their features.                                          |
| correlationMethod                 | Correlation method. Differential correlation currently supports "pearson" and "spearman".                                                                                                                                                          |
| minimumAbsoluteCorrelation        | Minimum absolute correlation that must be present in at least one group for edge retention. P-values are adjusted over the full testable family before this filter is applied.                                                                     |
| adjustedPValueThreshold           | Maximum within-group adjusted p-value that must be reached in the same group as the correlation threshold before an edge is kept. Set NULL to skip within-group significance filtering after the correlation prefilter.                            |
| pAdjustMethod                     | Multiple-testing correction method. Use one of stats::p.adjust.methods or "qvalue"; the latter requires the suggested qvalue package.                                                                                                              |
| featureNameColumn                 | Row-data column containing display names.                                                                                                                                                                                                          |
| differenceAdjustedPValueThreshold | Optional maximum adjusted differential-correlation p-value retained in the network. The default NULL keeps all rewiring-scoring edges returned by testDifferentialCorrelation.                                                                     |
| edgeWeightMethod                  | Method used to generate the edge weight. "signedZScore" keeps the sign of zScoreDifference; "absoluteZScore" takes its magnitude. See Details for what the sign means.                                                                             |

|                       |                                                                                                                                                                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| scoreColumn           | Rewiring-score column used to compute permutation tail probabilities.                                                                                                                                                                        |
| nPermutations         | Number of group-label permutations.                                                                                                                                                                                                          |
| blockColumn           | Optional sample metadata column. When supplied, group labels are permuted within each block.                                                                                                                                                 |
| seed                  | Optional random seed. Label allocations are generated before parallel work is dispatched, so the same seed gives the same allocations for serial and parallel backends.                                                                      |
| keepPermutationScores | Whether to return the node-by-permutation score table.                                                                                                                                                                                       |
| resultName            | Optional storage name.                                                                                                                                                                                                                       |
| storeResult           | Whether to store the result in analysisData.                                                                                                                                                                                                 |
| verbose               | Whether to report permutation progress.                                                                                                                                                                                                      |
| progressEvery         | Number of completed permutations between progress messages. This also sets the size of each parallel batch.                                                                                                                                  |
| BPPARAM               | A <a href="#">BiocParallel::BiocParallelParam</a> object controlling execution. The default <a href="#">BiocParallel::SerialParam()</a> runs sequentially. On Windows, use <a href="#">BiocParallel::SnowParam()</a> for parallel execution. |

## Details

Group labels are permuted and the differential-correlation statistics are recomputed over a fixed edge universe. An edge can still be unscored in a permutation when a group has fewer than four pairwise-complete observations or a feature is constant, so realized node degree can fall in some permutations.

Dense all-pairs mode follows [testDifferentialCorrelation\(\)](#): with assayName = NULL, feature pairs are generated separately within every assay and the assay-specific edge sets are combined for scoring. It never creates cross-omic pairs. The resulting observed network defines the combined within-omic edge universe reused throughout permutation validation. Tail probabilities use the Phipson and Smyth (2010) estimator,  $(1 + r) / (1 + B_i)$ , where  $B_i$  is the number of permutations that produced a score for the node. contributingPermutations reports  $B_i$ . Nodes with few contributing permutations have a thin null and should be read as descriptive.

inferenceStatus is "randomization p-value" only when a supplied candidate universe is fixed before the observed analysis, no correlation or p-value filter is active, and missingness and ties cannot make a candidate edge unscorable under any permitted allocation. The code also checks that every edge was scored in the observed data and every sampled permutation. The selected node score must be finite in the observed data and all requested permutations. The user must still ensure that the candidate universe was prespecified and that labels are exchangeable globally, or within each level of blockColumn.

In all other settings, including all-pairs mode and the function's default filtering settings, inferenceStatus is "conditional ranking". The two tail-probability columns then rank nodes against a label-dependent or incomplete reference distribution; they do not carry p-value or false discovery rate guarantees. Applying pAdjustMethod remains useful as a monotone ranking adjustment, but does not restore those guarantees.

Permutations are sampled independently and assignments can repeat. With  $B = nPermutations$  Monte Carlo draws, the smallest tail probability for a fully scored node is  $1 / (B + 1)$ . Exact enumeration of distinct assignments is not implemented. With 1,000 draws the minimum is about 0.001, which is often too coarse after adjustment across many nodes.

When seed is supplied, it is applied with [withr::local\\_seed\(\)](#), which restores the calling RNG state on exit. With seed = NULL, the label draws use and advance the caller's current RNG stream.

Label allocations are generated in permutation-index order before scoring begins. Parallel workers therefore perform deterministic calculations, and a fixed seed gives the same result regardless of worker count.

Parallel scoring is opt-in through BPPARAM; the default remains serial. When verbose = TRUE, the manager reports progress after each batch of up to progressEvery permutations. A stopped backend is started for the call and stopped on exit. A backend that was already running is left running. Each socket worker receives its own copy of the analysis inputs, so memory use should be considered when choosing the worker count. Interrupted runs cannot currently be resumed from a partial batch.

## Value

A named list containing the observed differential-correlation table, observed differential network, rewiring validation table, and optionally the permutation score table. The rewiring table includes permutationTailProbability, adjustedPermutationTailProbability, and inferenceStatus; the same status is also a scalar element of the result list so it remains available when the table has no rows. When storeResult = TRUE, the list is stored in metadata(analysisData)\$cornetto\$validationResults and the updated MultiAssayExperiment is returned.

## References

Phipson B, Smyth GK (2010). Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9, 39. doi:[10.2202/15446115.1585](https://doi.org/10.2202/15446115.1585)

## Examples

```
analysisData <- exampleAnalysisData()
validation <- permuteRewiringScores(
  analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = c("Recovered", "PASC"),
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  pAdjustMethod = "fdr",
  nPermutations = 2,
  seed = 1
)
names(validation)
```

---

|                    |                             |
|--------------------|-----------------------------|
| plotRewiringScores | <i>Plot Rewiring Scores</i> |
|--------------------|-----------------------------|

---

## Description

Plot Rewiring Scores

**Usage**

```
plotRewiringScores(
  rewiringTable,
  scoreColumn = "rootMeanSquareRewiringScore",
  topN = 20L,
  label = NULL,
  main = "Top rewired nodes"
)
```

**Arguments**

|                            |                                                                                                                                                                                                                                                     |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rewiringTable</code> | Output of <code>calculateRewiringScores()</code> .                                                                                                                                                                                                  |
| <code>scoreColumn</code>   | Score column to plot. See <a href="#">calculateRewiringScores()</a> for what the three rewiring scores mean. The tail-probability columns from <a href="#">permuteRewiringScores()</a> are also supported and are ordered from smallest to largest. |
| <code>topN</code>          | Number of top nodes to display.                                                                                                                                                                                                                     |
| <code>label</code>         | Optional label source. When <code>NULL</code> , <code>nodeKey</code> is used when present and <code>nodeIdentifier</code> otherwise. Use "identifier", "name", or "key" to select a specific label source.                                          |
| <code>main</code>          | Plot title.                                                                                                                                                                                                                                         |

**Details**

The default is `rootMeanSquareRewiringScore` because it is comparable across nodes of different degree and is defined for every node. `degreeMatchedZScore` is only a within-bin ranking aid and is NA for sparsely populated degree bins, so it is a poor default for small networks. Rewiring scores are ordered decreasingly. Permutation tail probabilities are ordered increasingly because smaller values are more extreme.

**Value**

Invisibly returns the plotted subset.

**Examples**

```
rewiringTable <- S4Vectors::DataFrame(
  nodeKey = c("protein:P1", "protein:P2"),
  nodeIdentifier = c("P1", "P2"),
  nodeName = c("CFH", "C3"),
  rootMeanSquareRewiringScore = c(2, 1),
  check.names = FALSE
)
plotRewiringScores(rewiringTable, topN = 2)
```

---

```
prepareCytoscapeTables
```

*Prepare Cytoscape Node and Edge Tables*

---

### Description

Prepare Cytoscape Node and Edge Tables

### Usage

```
prepareCytoscapeTables(networkEdgeTable, rewiringTable = NULL)
```

### Arguments

networkEdgeTable

A standardized edge table.

rewiringTable

Optional rewiring table to merge into the node table. Stored node metadata is retained when present so isolated nodes can be exported.

### Value

A named list with nodes and edges.

### Examples

```
knowledgeNetwork <- exampleKnowledgeNetwork()
networkTables <- prepareCytoscapeTables(knowledgeNetwork)
names(networkTables)
```

---

```
readKnowledgeNetwork
```

*Read a Knowledge Network from Disk*

---

### Description

Read a delimited prior-knowledge table, optionally remap the input column names, and standardize it for downstream CorNetto functions.

### Usage

```
readKnowledgeNetwork(
  filePath,
  columnMapping = NULL,
  delimiter = NULL,
  analysisData = NULL,
  resultName = "knowledgeNetwork",
  storeResult = FALSE,
  edgeType = "association",
  edgeDirection = "undirected",
  knowledgeSource = "userSupplied"
)
```

**Arguments**

|                 |                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------|
| filePath        | Path to a delimited text file.                                                                         |
| columnMapping   | Optional named character vector mapping standard CorNetto column names to columns present in the file. |
| delimiter       | Optional delimiter. When NULL, the delimiter is inferred from the file extension.                      |
| analysisData    | Optional MultiAssayExperiment used to store the standardized network.                                  |
| resultName      | Name used when storing the knowledge network.                                                          |
| storeResult     | Whether to store the result in analysisData.                                                           |
| edgeType        | Default edge type when the column is missing.                                                          |
| edgeDirection   | Default edge direction when the column is missing.                                                     |
| knowledgeSource | Default source label when the column is missing.                                                       |

**Value**

A standardized DataFrame or an updated MultiAssayExperiment.

**Examples**

```
knowledgePath <- system.file(
  "extdata", "example_knowledge_network.csv", package = "CorNetto"
)
knowledgeNetwork <- readKnowledgeNetwork(knowledgePath)
knowledgeNetwork
```

---

`summarizeAnalysisData` *Summarize a CorNetto Analysis Object*

---

**Description**

Summarize assay dimensions, missingness, low-variance features, sample overlap, and optional group sizes before network construction.

**Usage**

```
summarizeAnalysisData(
  analysisData,
  groupColumn = NULL,
  missingnessWarningThreshold = 0.2,
  lowSampleWarningThreshold = 10L,
  highFeatureWarningThreshold = 3000L,
  quiet = FALSE
)
```

**Arguments**

|                             |                                                                                            |
|-----------------------------|--------------------------------------------------------------------------------------------|
| analysisData                | A MultiAssayExperiment.                                                                    |
| groupColumn                 | Optional sample metadata column used to summarize group sizes.                             |
| missingnessWarningThreshold | Fraction of missing assay values above which a warning is reported for an assay.           |
| lowSampleWarningThreshold   | Group or assay sample count below which a warning is reported.                             |
| highFeatureWarningThreshold | Feature count above which dense all-pairs correlation is flagged as computationally heavy. |
| quiet                       | Whether to suppress emitted warnings and only return them in the result object.            |

**Value**

A named list with assays, samples, and warnings.

**Examples**

```
analysisData <- exampleAnalysisData()
summary <- summarizeAnalysisData(
  analysisData,
  groupColumn = "clinicalGroup",
  quiet = TRUE
)
summary$assays
```

---

summarizeKnowledgeNetwork

*Summarize a Knowledge Network*


---

**Description**

Summarize prior-network size, edge annotations, assay-pair coverage, duplicate edges, and optionally how much of the network is measurable in a CorNetto analysis object.

**Usage**

```
summarizeKnowledgeNetwork(knowledgeNetwork, analysisData = NULL, quiet = FALSE)
```

**Arguments**

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| knowledgeNetwork | A standardized knowledge network or any table accepted by validateKnowledgeNetwork().                             |
| analysisData     | Optional MultiAssayExperiment used to count edges whose assay and feature endpoints are present in measured data. |
| quiet            | Whether to suppress emitted warnings and only return them in the result object.                                   |

**Value**

A named list with summary DataFrame objects and warnings.

**Examples**

```
analysisData <- exampleAnalysisData()
knowledgeNetwork <- exampleKnowledgeNetwork()
summary <- summarizeKnowledgeNetwork(knowledgeNetwork, analysisData)
summary$overall
summary$assayPairs
```

---

testDifferentialCorrelation

*Test Differential Correlation Between Two Groups*


---

**Description**

Compute differential correlation statistics for either all within-omic feature pairs or a supplied set of candidate edges. In all-pairs mode, omitting assayName tests every assay while keeping pairs within their assay; no cross-omic pairs are generated.

**Usage**

```
testDifferentialCorrelation(
  analysisData,
  groupColumn,
  groupLevels,
  assayName = NULL,
  candidateEdgeTable = NULL,
  featureSubset = NULL,
  correlationMethod = c("pearson", "spearman"),
  minimumAbsoluteCorrelation = 0.3,
  adjustedPValueThreshold = 0.05,
  pAdjustMethod = "fdr",
  featureNameColumn = "featureName",
  resultName = NULL,
  storeResult = TRUE
)
```

**Arguments**

|                    |                                                                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analysisData       | A MultiAssayExperiment.                                                                                                                                                                                                                            |
| groupColumn        | Sample metadata column used to define the two groups.                                                                                                                                                                                              |
| groupLevels        | A character vector of length two giving the group labels. The first group is treated as the reference for gain or loss labels.                                                                                                                     |
| assayName          | Optional assay name used when testing all within-omic pairs. When NULL, every assay is tested and the results are combined. Ignored when candidateEdgeTable is supplied.                                                                           |
| candidateEdgeTable | Optional candidate edges to test. May be a standardized CorNetto edge table or a validated knowledge network. Candidate self-loops are omitted with a warning because a feature's correlation with itself is not an informative differential edge. |

|                            |                                                                                                                                                                                                                         |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| featureSubset              | Optional feature subset. In all-assay mode, a vector is intersected with each assay, or a named list may supply a separate subset for each assay. Assays absent from a named list use all their features.               |
| correlationMethod          | Correlation method. Differential correlation currently supports "pearson" and "spearman".                                                                                                                               |
| minimumAbsoluteCorrelation | Minimum absolute correlation that must be present in at least one group for edge retention. P-values are adjusted over the full testable family before this filter is applied.                                          |
| adjustedPValueThreshold    | Maximum within-group adjusted p-value that must be reached in the same group as the correlation threshold before an edge is kept. Set NULL to skip within-group significance filtering after the correlation prefilter. |
| pAdjustMethod              | Multiple-testing correction method. Use one of <code>stats::p.adjust.methods</code> or "qvalue"; the latter requires the suggested qvalue package.                                                                      |
| featureNameColumn          | Row-data column containing display names.                                                                                                                                                                               |
| resultName                 | Optional name used when storing the result.                                                                                                                                                                             |
| storeResult                | Whether to store the result in <code>analysisData</code> .                                                                                                                                                              |

## Details

Within-group and differential p-values are adjusted over all testable pairs before any correlation-strength filter is applied. When `assayName = NULL`, this is one combined multiple-testing family across every within-omic pair from every assay. Edges are then filtered to pairs with sufficient absolute correlation in at least one group. When `adjustedPValueThreshold` is not NULL, an edge is retained only if the same group is both sufficiently correlated and significant after multiple-testing correction. Differential-correlation adjusted p-values are reported but are not used for retention unless requested later by `createDifferentialCorrelationNetwork()`.

Candidate edges are treated as undirected hypotheses. Self-loops are removed with a warning, and duplicate or mirrored rows are tested once, using the first occurrence.

The statistic is the Fisher z difference between two independent correlations, evaluated against a standard normal:

$$z = (\operatorname{atanh}(r_2) - \operatorname{atanh}(r_1)) / \sqrt{1/(n_1 - 3) + 1/(n_2 - 3)}$$

For `correlationMethod = "spearman"` the variance term is inflated by 1.06 following Fieller, Hartley and Pearson (1957).

Sample correlations of exactly -1 or 1 make the Fisher transform infinite. CorNetto bounds them to  $-1 + 1e-6$  or  $1 - 1e-6$  before the transform. This keeps the statistic finite but makes results for perfectly collinear pairs sensitive to that numerical convention.

Mind the direction. The subtraction is group 2 minus group 1, so a **positive** `zScoreDifference` means the pair is more strongly correlated in `groupLevels[2]`. This is the opposite orientation to the `edgeDirection` labels from `createDifferentialCorrelationNetwork()`, which are named from group 1's point of view (`gainInGroup1` marks a pair correlated in group 1 and not in group 2, and therefore carries a negative z). The magnitude, the p-value, and every rewiring score are unaffected by the convention, since they square or take the absolute value of z; only `edgeWeight` under `edgeWeightMethod = "signedZScore"` exposes the sign.

This carries assumptions that omic data routinely violate, and the package does not test them for you:

- Pearson z tests assume each pair is approximately bivariate normal. Log-abundance proteomics and metabolomics are often heavy-tailed or left-censored, where the approximation is anticonservative. Use `correlationMethod = "spearman"` for heavy-tailed, monotone non-linear, or rank-robust analyses; Pearson remains the default because it is conventional for normalized log-scale omics.
- The two groups must be independent. Repeated measurements on the same subject in both groups break this.
- The variance term divides by  $n - 3$ , so four samples per group is the arithmetic floor and the function stops below it. That floor is not a recommendation: at four per group the standard error of the z difference is  $\sqrt{2}$ . A warning is emitted below ten samples per group, and results there should be read as descriptive.

Correlations use pairwise complete observations, so each edge carries its own effective sample size in `group1SampleCount` and `group2SampleCount`. Pairs where a feature is constant within a group yield no correlation and are dropped.

### Value

A standardized edge `DataFrame` or an updated `MultiAssayExperiment`.

### References

- Fisher RA (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron*, 1, 3-32.
- Fieller EC, Hartley HO, Pearson ES (1957). Tests for rank correlation coefficients I. *Biometrika*, 44, 470-481. doi:10.1093/biomet/44.34.470
- Benjamini Y, Hochberg Y (1995). Controlling the false discovery rate. *Journal of the Royal Statistical Society B*, 57, 289-300. doi:10.1111/j.25176161.1995.tb02031.x

### Examples

```
analysisData <- exampleAnalysisData()
differentialTable <- testDifferentialCorrelation(
  analysisData,
  groupColumn = "clinicalGroup",
  groupLevels = c("Recovered", "PASC"),
  minimumAbsoluteCorrelation = 0,
  adjustedPValueThreshold = 1,
  pAdjustMethod = "fdr",
  storeResult = FALSE
)
differentialTable
```

---

`validateAnalysisData`    *Validate a CorNetto Analysis Object*

---

### Description

Validate that a `MultiAssayExperiment` is compatible with CorNetto.

**Usage**

```
validateAnalysisData(analysisData, quiet = FALSE)
```

**Arguments**

|              |                                          |
|--------------|------------------------------------------|
| analysisData | A MultiAssayExperiment.                  |
| quiet        | Whether to suppress the success message. |

**Value**

The validated MultiAssayExperiment.

**Examples**

```
analysisData <- exampleAnalysisData()
validated <- validateAnalysisData(analysisData)
validated
```

---

```
validateKnowledgeNetwork
```

*Validate a Knowledge Network*

---

**Description**

Validate and standardize a user-supplied prior-knowledge network.

**Usage**

```
validateKnowledgeNetwork(
  knowledgeNetwork,
  analysisData = NULL,
  resultName = "knowledgeNetwork",
  storeResult = FALSE,
  edgeType = "association",
  edgeDirection = "undirected",
  knowledgeSource = "userSupplied",
  quiet = FALSE
)
```

**Arguments**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| knowledgeNetwork | A data frame-like object containing prior interactions.                           |
| analysisData     | Optional MultiAssayExperiment used to store the standardized network.             |
| resultName       | Name used when storing the knowledge network in metadata(analysisData)\$cornetto. |
| storeResult      | Whether to store the result in analysisData.                                      |
| edgeType         | Default edge type when the column is missing.                                     |
| edgeDirection    | Default edge direction when the column is missing.                                |
| knowledgeSource  | Default source label when the column is missing.                                  |
| quiet            | Whether to suppress the success message.                                          |

**Value**

A standardized DataFrame or an updated MultiAssayExperiment when storeResult = TRUE.

**Examples**

```
knowledgeNetwork <- exampleKnowledgeNetwork()
validated <- validateKnowledgeNetwork(knowledgeNetwork)
validated
```

---

|                    |                                             |
|--------------------|---------------------------------------------|
| writeNetworkTables | <i>Write Cytoscape-Ready Network Tables</i> |
|--------------------|---------------------------------------------|

---

**Description**

Write Cytoscape-Ready Network Tables

**Usage**

```
writeNetworkTables(
  networkTables,
  directoryPath,
  prefix = "cornettoNetwork",
  fileFormat = c("tsv", "csv")
)
```

**Arguments**

|               |                                                                                 |
|---------------|---------------------------------------------------------------------------------|
| networkTables | Output of prepareCytoscapeTables(), or a named list containing nodes and edges. |
| directoryPath | Output directory. Required, and created if it does not exist.                   |
| prefix        | File-name prefix.                                                               |
| fileFormat    | Either "tsv" or "csv".                                                          |

**Details**

directoryPath has no default so that the destination is always a deliberate choice rather than the current working directory.

**Value**

Invisibly returns the written file paths.

**Examples**

```
knowledgeNetwork <- exampleKnowledgeNetwork()
networkTables <- prepareCytoscapeTables(knowledgeNetwork)
outputDir <- file.path(tempdir(), "cornetto-example")
writtenFiles <- writeNetworkTables(networkTables, outputDir)
file.exists(writtenFiles)
```

# Index

## \* internal

CorNetto-package, 3

BiocParallel::BiocParallelParam, 21

BiocParallel::SerialParam(), 21

BiocParallel::SnowParam(), 21

calculateRewiringScores, 4

calculateRewiringScores(), 7, 12, 23

combineKnowledgeNetworks, 6

combineNetworks, 6

combineNetworks(), 5

CorNetto (CorNetto-package), 3

CorNetto-package, 3

correlationResults (getCorNettoResult),  
18

createAnalysisData, 8

createCorrelationNetwork, 9

createDifferentialCorrelationNetwork,  
11

createDifferentialCorrelationNetwork(),  
28

createFocusedNetwork, 12

createNetworkGraph, 13

createNetworkGraph(), 7

differentialCorrelationNetworks  
(getCorNettoResult), 18

differentialCorrelationResults  
(getCorNettoResult), 18

exampleAnalysisData, 14

exampleKnowledgeNetwork, 15

exampleSeedNodes, 15

filterFeatures, 16

filterFeatures(), 9

filterNetworkByNodes, 17

filterSamples, 17

getCorNettoResult, 18

integratedNetworks (getCorNettoResult),  
18

knowledgeNetworks (getCorNettoResult),  
18

permuteRewiringScores, 19

permuteRewiringScores(), 5, 23

plotRewiringScores, 22

prepareCytoscapeTables, 24

readKnowledgeNetwork, 24

rewiringResults (getCorNettoResult), 18

stats::cor.test(), 10

summarizeAnalysisData, 25

summarizeKnowledgeNetwork, 26

testDifferentialCorrelation, 27

testDifferentialCorrelation(), 11, 12,  
21

validateAnalysisData, 29

validateKnowledgeNetwork, 30

validationResults (getCorNettoResult),  
18

withr::local\_seed(), 21

writeNetworkTables, 31