

# Package ‘SpiecEasi’

June 5, 2026

**Title** Sparse Inverse Covariance for Ecological Statistical Inference

**Version** 2.1.1

**Date** 2026-04-29

**Description** Estimate networks from the precision matrix of compositional microbial abundance data.

**License** GPL (>= 3)

**URL** <https://github.com/zdk123/SpiecEasi>

**BugReports** <https://github.com/zdk123/SpiecEasi/issues>

**biocViews** Software, Microbiome, Metagenomics, GraphAndNetwork,  
NetworkInference

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Depends** R (>= 4.5.0),

**Imports** stats, methods, graphics, grDevices, huge (>= 1.3.2), pulsar  
(>= 0.3.11), MASS, VGAM, Matrix (>= 1.5), glmnet, phyloseq

**Suggests** parallel, boot, igraph, batchtools, testthat, covr, knitr,  
BiocStyle, rmarkdown, RefManageR, sessioninfo, magick

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**Config/testthat/edition** 3

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

**git\_branch** devel

**git\_last\_commit** 3c74772

**git\_last\_commit\_date** 2026-04-29

**Repository** Bioconductor 3.24

**Date/Publication** 2026-06-04

**Author** Zachary Kurtz [aut, cre],  
Christian Mueller [aut],  
Emily Miraldi [aut],  
Richard Bonneau [aut],  
Laura Tipton [ctb]

**Maintainer** Zachary Kurtz <zdkurtz@gmail.com>

## Contents

SpiecEasi-package	3
adj2igraph	4
AGP	4
alr	5
as.data.frame.graph	6
as.matrix.graph	7
clr	7
coat	8
cor2cov	9
cov2prec	10
ebic	10
edge.diss	11
fitdistr	12
getOptInd	13
get_comm_params	14
graph2prec	14
hmp2	15
make_graph	16
multi.spiec.easi	16
neff	18
neighborhood.net	18
norm_pseudo	19
norm_rdiric	20
norm_to_total	20
prec2cov	21
pulsar.params	21
pval.sparccboot	22
qqdplot_comm	22
rmvnegbin	23
rmvnorm	24
rmvpois	25
rmvzinegbin	25
rmvzipois	26
robustPCA	27
rzipois	28
shannon	28
sparcc	29
sparccboot	29
sparseiCov	30
sparseLowRankiCov	32
spiec.easi	33
stars.roc	34
symBeta	35
synth_comm_from_counts	36
triu	37
<b>Index</b>	<b>38</b>

---

SpiecEasi-package	<i>SpiecEasi: Sparse Inverse Covariance Estimation for Ecological Association Inference</i>
-------------------	---

---

## Description

SpiecEasi provides methods for inferring ecological associations from compositional microbiome data using sparse inverse covariance estimation.

## Details

The package implements SPIEC-EASI (SParse InversE Covariance estimation for Ecological Association Inference) methods for microbiome network inference, including glasso and neighborhood selection approaches with stability selection for model selection.

## Value

This package provides functions for microbiome network inference and analysis

## Author(s)

**Maintainer:** Zachary Kurtz <zdkurtz@gmail.com>

Authors:

- Christian Mueller
- Emily Miraldi
- Richard Bonneau

Other contributors:

- Laura Tipton [contributor]

## See Also

Useful links:

- <https://github.com/zdk123/SpiecEasi>
- Report bugs at <https://github.com/zdk123/SpiecEasi/issues/>

---

`adj2igraph`*Adjacency to igraph*

---

**Description**

Convert an adjacency matrix (ie - from the `sparseiCov` function) to an igraph object

**Usage**

```
adj2igraph(  
  Adj,  
  rmEmptyNodes = FALSE,  
  diag = FALSE,  
  edge.attr = list(),  
  vertex.attr = list(name = seq_len(ncol(Adj)))  
)
```

**Arguments**

<code>Adj</code>	an Adjacency matrix
<code>rmEmptyNodes</code>	should unconnected nodes be removed from the graph
<code>diag</code>	Flag to include self-loops (diagonal of adjacency matrix)
<code>edge.attr</code>	named list of attributes for graph edges
<code>vertex.attr</code>	named list of attributes for graph vertices

**Value**

An igraph object

**Examples**

```
# Create a symmetric adjacency matrix  
adj <- matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), nrow=3, byrow=TRUE)  
  
# Convert to igraph  
g <- adj2igraph(adj, vertex.attr=list(name=c('A', 'B', 'C')))
```

---

`AGP`*American Gut Project*

---

**Description**

Round 1 and 2 community count datasets from the American Gut Project.

**Usage**

```
data(amgut1.filt)  
  
data(amgut2.filt.phy)
```

**Format**

1. amgut1.filt: A matrix with 289 samples (rows) and 127 OTUs (cols).
2. amgut2.filt.phy: A phyloseq object

**Value**

List containing amgut1.filt matrix and amgut2.filt.phy phyloseq object

**Source**

<http://humanfoodproject.com/americangut/>

---

alr

*Additive log-ratio functions*

---

**Description**

Additive log-ratio functions

**Usage**

```
alr(x.f, ...)  
  
## Default S3 method:  
alr(  
  x.f,  
  divcomp = 1,  
  base = exp(1),  
  removeDivComp = TRUE,  
  tol = .Machine$double.eps,  
  ...  
)  
  
## S3 method for class 'matrix'  
alr(  
  x.f,  
  mar = 2,  
  divcomp = 1,  
  base = exp(1),  
  removeDivComp = TRUE,  
  tol = .Machine$double.eps,  
  ...  
)  
  
## S3 method for class 'data.frame'  
alr(x.f, mar = 2, ...)
```

**Arguments**

x.f	input data
...	pass through arguments
divcomp	the index of the component to use as the divisor
base	base for log transformation
removeDivComp	remove the divisor component from the alr result
tol	tolerance for a numerical zero
mar	margin to apply the transformation (rows: 1 or cols: 2)

**Value**

Additive log-ratio transformed data

**Examples**

```
x <- c(1, 2, 3, 4)
alr(x) # Returns additive log-ratio transformation using first component as reference

# Matrix example
mat <- matrix(1:12, nrow=3)
alr(mat) # ALR transformation by columns

# Data frame example
df <- as.data.frame(mat)
alr(df) # ALR transformation by columns
```

---

as.data.frame.graph *s3 method for graph to other data types*

---

**Description**

s3 method for graph to other data types

**Usage**

```
## S3 method for class 'graph'
as.data.frame(x, ...)
```

**Arguments**

x	graph adjacency matrix
...	Arguments to base as.data.frame

**Value**

A data.frame

**Examples**

```
# Create a graph and convert to graph adjacency data.frame
g <- make_graph("erdos_renyi", D=5, e=6)
df <- as.data.frame(g)
```

---

as.matrix.graph	<i>s3 method for graph to other data types</i>
-----------------	--

---

**Description**

s3 method for graph to other data types

**Usage**

```
## S3 method for class 'graph'  
as.matrix(x, ...)
```

**Arguments**

x	graph adjacency matrix
...	Arguments to base as.matrix

**Value**

A matrix

**Examples**

```
# Create a graph and convert to graph adjacency matrix  
g <- make_graph("erdos_renyi", D=5, e=6)  
mat <- as.matrix(g)
```

---

clr	<i>Centered log-ratio functions</i>
-----	-------------------------------------

---

**Description**

Centered log-ratio functions

**Usage**

```
clr(x.f, ...)  
  
## Default S3 method:  
clr(x.f, base = exp(1), tol = .Machine$double.eps, ...)  
  
## S3 method for class 'matrix'  
clr(x.f, mar = 2, ...)  
  
## S3 method for class 'data.frame'  
clr(x.f, mar = 2, ...)
```

**Arguments**

<code>x.f</code>	input data
<code>...</code>	pass through arguments
<code>base</code>	base for log transformation
<code>tol</code>	tolerance for a numerical zero
<code>mar</code>	margin to apply the transformation (rows: 1 or cols: 2)

**Value**

Centered log-ratio transformed data

**Examples**

```
x <- c(1, 2, 3, 4)
clr(x) # Returns centered log-ratio transformation

# Matrix example
mat <- matrix(1:12, nrow=3)
clr(mat) # CLR transformation by columns

# Data frame example
df <- as.data.frame(mat)
clr(df) # CLR transformation by columns
```

---

coat

*COAT*

---

**Description**

Compositional-adjusted thresholding by doi.org/10.1080/01621459.2018.1442340 by Cao, Lin & Li (2018)

**Usage**

```
coat(
  data,
  lambda,
  thresh = "soft",
  adaptive = TRUE,
  shrinkDiag = TRUE,
  ret.icov = FALSE,
  ...
)
```

**Arguments**

<code>data</code>	a clr-transformed data or covariance matrix
<code>lambda</code>	threshold parameter(s)
<code>thresh</code>	"soft" or "hard" thresholding
<code>adaptive</code>	use adative-version of the lambda as in the original COAT paper. See details.

shrinkDiag flag to exclude the covariance diagonal from the shrinkage operation  
 ret.icov flag to also return the inverse covariance matrix (inverse of all thresholded COAT matrices)  
 ... Arguments to automatically calculating the lambda path. See details.

### Details

If adaptive=TRUE, and data is a covariance matrix, the adaptive penalty is calculated by assuming the underlying data is jointly Gaussian in the infinite sample setting. The results may differ from the 'empirical' adaptive setting.

There are a few undocumented arguments useful for computing a lambda path on the fly:

**lambda.max** Maximum lambda. Default: max absolute covariance

**lambda.min.ratio** lambda.min is lambda.min.ratio\*lambda.max is the smallest lambda evaluated.  
 Default: 1e-3

**nlambda** Number of values of lambda between lambda.max and lambda.min. Default: 30

### Value

COAT result object with thresholded covariance matrix

### Examples

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(10)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(10, 0, 2), Sigma))

# normalize
data.clr <- t(clr(data, 1))

# apply COAT
est.coat <- coat(data.clr, lambda=0.15, thresh="soft")
image(as.matrix(est.coat$cov))
```

---

cor2cov	<i>Convert a symmetric correlation matrix to a covariance matrix given the standard deviation</i>
---------	---

---

### Description

Convert a symmetric correlation matrix to a covariance matrix given the standard deviation

### Usage

```
cor2cov(cor, sds)
```

### Arguments

cor a symmetric correlation matrix  
 sds standard deviations of the resulting covariance.

**Value**

Covariance matrix of sample dimension as cor

**Examples**

```
# Create a correlation matrix and standard deviations
cor <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
sds <- c(2, 3, 4)
# Convert to covariance matrix
cov <- cor2cov(cor, sds)
```

---

cov2prec

*Covariance matrix to its matrix inverse (Precision matrix)*

---

**Description**

Covariance matrix to its matrix inverse (Precision matrix)

**Usage**

```
cov2prec(Cov, tol = 1e-04)
```

**Arguments**

Cov                    symmetric covariance matrix (can be correlation also)  
 tol                    tolerance to define a zero eigenvalue (ie - is Prec positive definite)

**Value**

A precision matrix (inverse of the covariance matrix)

**Examples**

```
# Create a simple covariance matrix
cov <- matrix(c(1, 0.5, 0, 0.5, 1, 0.5, 0, 0.5, 1), nrow=3)
# Convert to precision matrix
prec <- cov2prec(cov)
```

---

ebic

*Extended BIC*

---

**Description**

Calculate the extended BIC criterion on a sparse (refit) network and the input data

**Usage**

```
ebic(refit, data, loglik, gamma = 0.5)
```

**Arguments**

refit	adjacency matrix, getOpt from SpiecEasi output
data	input data set used to get the network
loglik	log likelihood of the graphical model
gamma	the model likelihood/complexity tradeoff parameter

**Value**

Extended BIC score

**Examples**

```
# Generate a random adjacency matrix
refit <- matrix(rbinom(100, size=1, prob=0.5), nrow=10)

# Generate random data
data <- matrix(rnorm(100), nrow=10)

# Calculate log likelihood
loglik <- sum(dnorm(data, mean=0, sd=1, log=TRUE))

# Calculate extended BIC
ebic(refit, data, loglik)
```

---

edge.diss	<i>Edge set dissimilarity</i>
-----------	-------------------------------

---

**Description**

Compute the dissimilarity between the edge sets of two networks via:

1. maximum overlap:  $|x \cap y| / \max\{|x|, |y|\}$
2. jaccard index (default):  $|x \cap y| / (|x \cup y|)$

Input networks do not have to have the same node sets.

**Usage**

```
edge.diss(x, y, metric = "jaccard", otux = NULL, otuy = NULL)
```

**Arguments**

x	pxp adjacency matrix ( <code>Matrix::sparseMatrix</code> class)
y	other qxq adjacency matrix ( <code>Matrix::sparseMatrix</code> class)
metric	'jaccard' or 'max'
otux	taxa names of adjacency x
otuy	taxa names of adjacency y

**Value**

Dissimilarity score between edge sets

**Examples**

```
# Create two sparse adjacency matrices
library(Matrix)
x <- Matrix(c(0,1,0,1,0,1,0,1,0), nrow=3, sparse=TRUE)
y <- Matrix(c(0,1,1,1,0,0,1,0,0), nrow=3, sparse=TRUE)

# Calculate Jaccard dissimilarity
jaccard_sim <- edge.diss(x, y, metric='jaccard')

# Calculate max overlap
max_sim <- edge.diss(x, y, metric='max')
```

fitdistr

*Fit parameters of a marginal distribution to some data vector***Description**

Fit parameters of a marginal distribution to some data vector

**Usage**

```
fitdistr(x, densfun, start, control, ...)
```

**Arguments**

x	data vector
densfun	string giving distribution function name
start	starting guess for the parameters (recommended leaving this out)
control	control parameters to optim
...	further arguments to densfun

**Value**

Fitted distribution parameters

**Examples**

```
# Fit Poisson distribution
x <- rpois(100, lambda=5)
fit_pois <- fitdistr(x, "pois")
fit_pois$par$lambda

# Fit negative binomial distribution
x_nb <- rnbinom(100, size=1, mu=5)
fit_nb <- fitdistr(x_nb, "negbin")
fit_nb$par['mu']

# Fit zero-inflated Poisson
x_zip <- c(rpois(80, lambda=5), rep(0, 20))
fit_zip <- fitdistr(x_zip, "zipois")
fit_zip$par['lambda']
```

---

getOptInd	<i>get StARS-optimal network</i>
-----------	----------------------------------

---

### Description

Get the optimal network, and related structures, when StARS is run.

### Usage

getOptInd(est)

getOptLambda(est)

getOptMerge(est)

getStability(est)

getOptNet(est)

getRefit(est)

getOptBeta(est)

getOptCov(est)

getOptiCov(est)

### Arguments

est                    output from `spiec.easi`

### Details

Use the getter functions to parse `spiec.easi` output:

- getOptLambda: penalty parameter from provided lambda path
- getOptInd: index of the selected lambda from provided lambda path
- getOptNet / getRefit: the optimal (StARS-refit) network
- getStability: average stability at the selected sparsity
- getOptMerge: symmetric matrix with edge-wise stability
- getOptiCov: the optimal inverse covariance matrix (glasso only)
- getOptCov: the optimal covariance matrix associated with the selected network (glasso only)
- getOptBeta: the optimal coefficient matrix (mb only)

### Value

numeric or matrix associated with a StARS solution.

**Examples**

```
# Get optimal index from spiec.easi result
data(amgut1.filt)
est <- spiec.easi(amgut1.filt, method='glasso', nlambda=10)
opt_idx <- getOptInd(est)
```

---

get_comm_params	<i>Get the parameters for the OTUs (along mar) of each community</i>
-----------------	--

---

**Description**

Get the parameters for the OTUs (along mar) of each community

**Usage**

```
get_comm_params(comm, mar = 2, distr, ...)
```

**Arguments**

comm	community: matrix of counts
mar	sample margin (1: "rows", 2: "cols")
distr	distribution to fit (see fitdistr)
...	arguments passed to fitdistr

**Value**

list of parameters

**Examples**

```
# Create a simple community matrix
comm <- matrix(rpois(20, lambda=5), nrow=4, ncol=5)
# Get parameters for Poisson distribution
params <- get_comm_params(comm, distr="pois")
# Get parameters for negative binomial distribution
params_nb <- get_comm_params(comm, distr="negbin")
```

---

graph2prec	<i>Convert a symmetric graph (extension of R matrix class)</i>
------------	--

---

**Description**

Has internal rules for converting various graph topologies into the associated adjacency and, therefore, precision matrix

**Usage**

```
graph2prec(
  Graph,
  posThetaLims = c(2, 3),
  negThetaLims = -posThetaLims,
  targetCondition = 100,
  epsBin = 0.01,
  numBinSearch = 100
)
```

**Arguments**

Graph	graph adjacency matrix
posThetaLims	length 2 vector of lower and upper bound of positive values
negThetaLims	length 2 vector of lower and upper bound of negative values
targetCondition	sets the condition of the precision matrix by modulating the magnitude of the diagonal
epsBin	the convergence tolerance of the condition number binary search
numBinSearch	maximum number of iterations

**Value**

A precision matrix with the specified condition number

**Examples**

```
# Create a simple graph
g <- make_graph("erdos_renyi", D=10, e=15)
# Convert to precision matrix
prec <- graph2prec(g)
```

---

hmp2

*Human Microbiome Project 2*


---

**Description**

Pre-filtered data from the integrated human microbiome project.

**Usage**

```
data(hmp2)
```

**Format**

1. hmp216S: 16S data, phyloseq object 45 taxa and 47 samples.
2. hmp2prot: protein data, A phyloseq object, 43 'taxa' and 47 samples.

**Value**

List containing hmp216S and hmp2prot phyloseq objects

**Source**

<https://www.hmpdacc.org/ihmp/>

---

make_graph	<i>Procedure to generate graph topologies for Gaussian Graphical Models</i>
------------	---

---

**Description**

Procedure to generate graph topologies for Gaussian Graphical Models

**Usage**

```
make_graph(method, D, e, enforce = TRUE, ...)
```

**Arguments**

method	Type of graph to make
D	Number of nodes/OTUs (Graph dimension)
e	Number of edges (preferably sparse, must be at least 1/2 D)
enforce	add/remove edges to enforce graph has e edges
...	additional options to graph method

**Value**

A symmetric adjacency matrix representing the graph topology

**Examples**

```
# Generate different types of graphs
g1 <- make_graph("erdos_renyi", D=10, e=15)
g2 <- make_graph("hub", D=10, e=15, numHubs=2)
g3 <- make_graph("scale_free", D=10, e=15)
g4 <- make_graph("cluster", D=10, e=15)
g5 <- make_graph("band", D=10, e=15)
g6 <- make_graph("block", D=10, e=15, numHubs=2)
```

---

multi.spiec.easi	<i>multi domain SPIEC-EASI</i>
------------------	--------------------------------

---

**Description**

A SPIEC-EASI pipeline for inferring a sparse inverse covariance matrix within and between multiple compositional datasets, under joint sparsity penalty.

**Usage**

```
multi.spiec.easi(
  datalist,
  method = "glasso",
  sel.criterion = "stars",
  verbose = TRUE,
  pulsar.select = TRUE,
  pulsar.params = list(),
  ...
)

## S3 method for class 'list'
spiec.easi(data, ...)
```

**Arguments**

<code>datalist</code>	list of non-normalized count OTU/data tables (stored in a matrix, data.frame or phyloseq/otu_table) with samples on rows and features/OTUs in columns
<code>method</code>	estimation method to use as a character string. Currently either 'glasso' or 'mb' (meinshausen-buhlmann's neighborhood selection)
<code>sel.criterion</code>	character string specifying criterion/method for model selection. Accepts 'stars' and 'bstars' [default]
<code>verbose</code>	flag to show progress messages
<code>pulsar.select</code>	flag to perform model selection. Choices are TRUE/FALSE/'batch'
<code>pulsar.params</code>	list of further arguments to pulsar model selection. See the documentation for <a href="#">pulsar.params</a> .
<code>...</code>	further arguments to sparse inverse covariance estimation
<code>data</code>	non-normalized count OTU/data table with samples on rows and features/OTUs in columns. Can also be list of phyloseq objects.

**Details**

Can also run `spiec.easi` on a list and S3 will dispatch the proper function.

**Value**

a list of pulsar parameters.

**See Also**

[spiec.easi](#)

**Examples**

```
# Generate random data
data <- exp(matrix(rnorm(100), nrow=10))
data2 <- exp(matrix(rnorm(100, sd=2, mean=20), nrow=10))
datalist <- list(data, data2)
# Run SPIEC-EASI
result <- spiec.easi(datalist)
```

---

neff	<i>N_effective: Compute the exponential of the shannon entropy. linearizes shannon entropy, for a better diveristy metric (effective number of species)</i>
------	---

---

**Description**

*N\_effective*: Compute the exponential of the shannon entropy. linearizes shannon entropy, for a better diveristy metric (effective number of species)

**Usage**

```
neff(x)
```

**Arguments**

x	data vector
---	-------------

**Value**

*N\_eff* in base e

**Examples**

```
x <- c(1, 2, 3, 4)
neff(x) # Returns effective number of species
```

---

neighborhood.net	<i>Neighborhood net estimates</i>
------------------	-----------------------------------

---

**Description**

Select a sparse inverse covariance matrix using neighborhood selection and glmnet from various exponential models.

**Usage**

```
neighborhood.net(data, lambda, method = "ising", ncores = 1, sym = "or", ...)
```

**Arguments**

data	n x p input (pre-transformed) data
lambda	the lambda path
method	ising and poisson models currently supported.
ncores	number of cores for distributing the model fitting
sym	symmetrize the neighborhood using the 'or' (default)/'and' rule
...	further arguments to glmnet

**Value**

A sparse inverse covariance matrix estimated using neighborhood selection

**Examples**

```
# Generate binary data for Ising model
set.seed(123)
data <- matrix(rbinom(100, 1, 0.5), nrow=20, ncol=5)
lambda <- c(0.1, 0.2, 0.3)

# Fit neighborhood selection model
result <- neighborhood.net(data, lambda, method="ising")

# Check adjacency matrices
length(result$path) # Number of lambda values
```

---

norm_pseudo	<i>Normalize w/ Pseudocount</i>
-------------	---------------------------------

---

**Description**

add pseudocount before normalizing a count vector

**Usage**

```
norm_pseudo(x)
```

**Arguments**

x                    count data vector

**Value**

A normalized vector with pseudo-count added

**Examples**

```
x <- c(1, 2, 0, 4)
norm_pseudo(x) # Adds 1 to each value before normalizing
```

---

norm_rdiric	<i>Normalize via dirichlet sampling</i>
-------------	---

---

**Description**

"Normalize" a count vector by drawing a single sample from a Dirichlet distribution, using the count vector as the prior.

**Usage**

```
norm_rdiric(x)
```

**Arguments**

x                    count data vector

**Value**

A single sample from Dirichlet distribution

**Examples**

```
x <- c(1, 2, 3, 4)
norm_rdiric(x) # Returns a single sample from Dirichlet distribution
```

---

norm_to_total	<i>Total Sum Normalize</i>
---------------	----------------------------

---

**Description**

Normalize a count vector by the total sum of that vector

**Usage**

```
norm_to_total(x)
```

**Arguments**

x                    count data vector

**Value**

A normalized vector (values sum to 1)

**Examples**

```
x <- c(1, 2, 3, 4)
norm_to_total(x) # Divides each value by sum(x) = 10
```

---

```
prec2cov
```

*Precision matrix (inverse covariance) to a covariance matrix*

---

**Description**

Precision matrix (inverse covariance) to a covariance matrix

**Usage**

```
prec2cov(Precision, tol = 1e-04)
```

**Arguments**

Precision	symmetric precision matrix
tol	tolerance to define a zero eigenvalue (ie - is Prec positive definite)

**Value**

A covariance matrix (inverse of the precision matrix)

**Examples**

```
# Create a simple precision matrix
prec <- matrix(c(2, -1, 0, -1, 2, -1, 0, -1, 2), nrow=3)
# Convert to covariance matrix
cov <- prec2cov(prec)
```

---

```
pulsar.params
```

*pulsar params*

---

**Description**

The values to the `pulsar.params/icov.select.params` argument in the [spiec.easi](#) function must be a list with values for pulsar model selection parameters.

List of arguments, data type, default. Description

- `thresh`, numeric, 0.05. Threshold for StARS criterion.
- `subsample.ratio`, numeric, 0.8. Subsample size for StARS.
- `rep.num`, numeric, 20. Number of subsamples for StARS.
- `seed`, numeric, NULL. Set the random seed for subsample set.
- `ncores`, numeric, 1. Number of cores for parallel.

With `pulsar.select='batch'`, additional arguments:

- `wkdir`, dir path, current directory. Working directory for process running jobs.
- `regdir`, dir path, temp directory. Directory for storing the registry files.
- `init`, string, 'init'. String for differentiating the init registry for batch mode pulsar.
- `conffile`, string / file path, ". Path to config file or string that identifies a default config file.
- `job.res`, list, empty list. Named list to specify job resources for an hpc.
- `cleanup`, boolean, FALSE. Remove registry files.

**Value**

A list of parameters for pulsar model selection

**See Also**

[spiec.easi](#)

---

pval.sparccboot	<i>SparCC p-vals</i>
-----------------	----------------------

---

**Description**

Get empirical p-values from bootstrap SparCC output.

**Usage**

```
pval.sparccboot(x, sided = "both")
```

**Arguments**

**x** output from sparccboot  
**sided** type of p-value to compute. Only two sided (sided="both") is implemented.

**Examples**

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(10)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(10, 0, 2), Sigma))

# estimate
est.sparcc <- sparccboot(data, R=100)
# find significant correlations
out <- pval.sparccboot(est.sparcc)
out$cors[out$pvals < .05]
```

---

qqdplot_comm	<i>qq-plot for theoretical vs observed communities</i>
--------------	--

---

**Description**

qq-plot for theoretical vs observed communities

**Usage**

```
qqdplot_comm(comm, distr, param, plot = TRUE, ...)
```

**Arguments**

comm	commutity count matrix
distr	character specifying target distribution
param	parameter list for fitting the data. Output from get_comm_params
plot	graph the output
...	pass arguments to qqplot

**Value**

QQ plot object or fitted parameters

**Examples**

```
# Create a simple community matrix
comm <- matrix(rpois(100, lambda=5), nrow=10, ncol=10)
# Get parameters for Poisson distribution
params <- get_comm_params(comm, distr="pois")
# Create QQ plot
qqdplot_comm(comm, distr="pois", param=params)
```

---

rmvnegbin	<i>Generate multivariate, Zero-inflated negative binomial data, with counts approximately correlated according to Sigma</i>
-----------	---

---

**Description**

Generate multivariate, Zero-inflated negative binomial data, with counts approximately correlated according to Sigma

**Usage**

```
rmvnegbin(n, mu, Sigma, ks, ...)
```

**Arguments**

n	number of samples to draw
mu	mean vector for variables (of length $D$ )
Sigma	$D \times D$ covariance or correlation matrix
ks	shape parameter
...	other arguments to the negative binomial distribution

**Value**

$D \times n$  matrix with zi-poisson data

**Examples**

```
# Generate 50 samples from 3 correlated negative binomial variables
mu <- c(2, 5, 8)
Sigma <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
data <- rmvnegbin(50, mu=mu, Sigma=Sigma)

# Generate with explicit shape parameters
ks <- c(2, 3, 4)
data2 <- rmvnegbin(50, mu=mu, Sigma=Sigma, ks=ks)
```

rmvnorm

*Draw samples from multivariate, correlated normal distribution with counts correlated according to Sigma*

**Description**

Draw samples from multivariate, correlated normal distribution with counts correlated according to Sigma

**Usage**

```
rmvnorm(
  n = 100,
  mu = rep(0, 10),
  Sigma = diag(10),
  tol = 1e-06,
  empirical = TRUE
)
```

**Arguments**

n	number of samples to draw
mu	mean vector for variables (of length $D$ )
Sigma	$D \times D$ covariance or correlation matrix
tol	numerical tolerance for a zero eigenvalue (check for PD of Sigma)
empirical	is Sigma the empirical correlation?

**Value**

$D \times n$  matrix with Gaussian data

**Examples**

```
# Generate 50 samples from 3 correlated normal variables
mu <- c(0, 0, 0)
Sigma <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
data <- rmvnorm(50, mu=mu, Sigma=Sigma)

# Generate with different mean vector
mu2 <- c(1, 2, 3)
data2 <- rmvnorm(50, mu=mu2, Sigma=Sigma)
```

---

rmvpois	<i>Generate multivariate poisson data, with counts approximately correlated according to Sigma</i>
---------	--

---

**Description**

Generate multivariate poisson data, with counts approximately correlated according to Sigma

**Usage**

```
rmvpois(n, mu, Sigma, ...)
```

**Arguments**

n	number of samples to draw
mu	mean vector for variables (of length $D$ )
Sigma	$D \times D$ covariance or correlation matrix
...	Arguments passed to qpois

**Value**

$D \times n$  matrix with zi-poisson data

**Examples**

```
# Generate 50 samples from 3 correlated Poisson variables
mu <- c(2, 5, 8)
Sigma <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
data <- rmvpois(50, mu=mu, Sigma=Sigma)
```

---

rmvzinegbin	<i>Generate multivariate, negative binomial data, with counts approximately correlated according to Sigma</i>
-------------	---

---

**Description**

Generate multivariate, negative binomial data, with counts approximately correlated according to Sigma

**Usage**

```
rmvzinegbin(n, mu, Sigma, munbs, ks, ps, ...)
```

**Arguments**

n	number of samples to draw
mu	mean vector for variables (of length $D$ )
Sigma	$D \times D$ covariance or correlation matrix
munbs	Rate/mean parameter (instead of mu)
ks	shape parameter
ps	probability of zero inflation
...	other arguments to the negative binomial distribution

**Value**

$D \times n$  matrix with zi-poisson data

**Examples**

```
# Generate 50 samples from 3 correlated ZINB variables
mu <- c(2, 5, 8)
Sigma <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
data <- rmvzinegbin(50, mu=mu, Sigma=Sigma)

# Generate with explicit parameters
munbs <- c(2, 5, 8)
ks <- c(2, 3, 4)
ps <- c(0.1, 0.2, 0.3)
data2 <- rmvzinegbin(50, Sigma=Sigma, munbs=munbs, ks=ks, ps=ps)
```

---

rmvzipois	<i>Generate multivariate, Zero-inflated poisson data, with counts approximately correlated according to Sigma</i>
-----------	---

---

**Description**

Generate multivariate, Zero-inflated poisson data, with counts approximately correlated according to Sigma

**Usage**

```
rmvzipois(n, mu, Sigma = diag(length(mu)), lambdas, ps, ...)
```

**Arguments**

n	number of samples to draw
mu	mean vector for variables (of length $D$ )
Sigma	$D \times D$ covariance or correlation matrix
lambdas	supply rate parameter (instead of mu)
ps	probability of zeros (instead of mu)
...	arguments passed to VGAM::qzipois

**Value**

*Dxn* matrix with zi-poisson data

**Examples**

```
# Generate 50 samples from 3 correlated ZIP variables
mu <- c(2, 5, 8)
Sigma <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.3, 0.2, 0.3, 1), nrow=3)
data <- rmvzipois(50, mu=mu, Sigma=Sigma)

# Generate using explicit lambda and zero-inflation parameters
lambdas <- c(2, 5, 8)
ps <- c(0.1, 0.2, 0.3)
data2 <- rmvzipois(50, Sigma=Sigma, lambdas=lambdas, ps=ps)
```

---

robustPCA

*robust PCA*


---

**Description**

Form a robust PCA from clr-transformed data and [the low rank component of] an inverse covariance matrix

**Usage**

```
robustPCA(X, L, inverse = TRUE)
```

**Arguments**

X	the n x p [clr-transformed] data
L	the p x p rank-r ('residual') inverse covariance matrix from <code>spiec.easi</code> run argument <code>method='slr'</code> .
inverse	flag to indicate the L is the inverse covariance matrix

**Value**

a named list with n x r matrix of scores and r x r matrix of loadings

**Examples**

```
# Create sample data
data(amgut1.filt)
data.clr <- t(clr(t(amgut1.filt), 1))
# Perform robust PCA
pca_result <- robustPCA(data.clr, diag(ncol(data.clr)))
```

---

rzipois	<i>Draw samples from a zero-inflated poisson distribution</i>
---------	---

---

**Description**

Draw samples from a zero-inflated poisson distribution

**Usage**

```
rzipois(n, lambda, pstr0 = 0)
```

**Arguments**

n	the number of samples to draw
lambda	The poisson rate parameter
pstr0	probability of drawing a zero

**Value**

Poisson counts of length  $n$

**Examples**

```
# Draw 10 samples from ZIP with lambda=5 and 20% zero inflation
rzipois(10, lambda=5, pstr0=0.2)

# Draw 100 samples with different parameters
rzipois(100, lambda=c(2,5,8), pstr0=c(0.1,0.2,0.3))
```

---

shannon	<i>compute the shannon entropy from a vector (normalized internally)</i>
---------	--

---

**Description**

Shannon entropy is:  $\sum [ x_i \log(x_i) ]$

**Usage**

```
shannon(x)
```

**Arguments**

x	data vector
---	-------------

**Value**

shannon entropy in base e

**Examples**

```
x <- c(1, 2, 3, 4)
shannon(x) # Returns Shannon entropy of normalized vector
```

---

sparcc	<i>sparcc wrapper</i>
--------	-----------------------

---

### Description

A reimplementation of SparCC algorithm (Friedman et Alm 2012, PLoS Comp Bio, 2012).

### Usage

```
sparcc(data, iter = 20, inner_iter = 10, th = 0.1)
```

### Arguments

data	Community count data matrix
iter	Number of iterations in the outer loop
inner_iter	Number of iterations in the inner loop
th	absolute value of correlations below this threshold are considered zero by the inner SparCC loop.

### See Also

[sparccboot](#)

### Examples

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(10)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(10, 0, 2), Sigma))

# estimate
est.sparcc <- sparcc(data)
est.sparcc$Cor[1,2]
```

---

sparccboot	<i>Bootstrap SparCC</i>
------------	-------------------------

---

### Description

Get bootstrapped estimates of SparCC correlation coefficients. To get empirical p-values, pass this output to `pval.sparccboot`.

**Usage**

```
sparccboot(
  data,
  sparcc.params = list(),
  statisticboot = function(data, indices) triu(do.call("sparcc", c(list(data[indices, ],
    drop = FALSE])), sparcc.params))$Cor),
  statisticperm = function(data, indices) triu(do.call("sparcc",
    c(list(apply(data[indices, ], 2, sample)), sparcc.params))$Cor),
  R,
  ncpus = 1,
  ...
)
```

**Arguments**

data	Community count data
sparcc.params	named list of parameters to pass to sparcc
statisticboot	function which takes data and bootstrap sample indices and results the upper triangle of the bootstapped correlation matrix
statisticperm	function which takes data and permuted sample indices and results the upper triangle of the null correlation matrix
R	number of bootstraps
ncpus	number of cores to use for parallelization
...	additional arguments that are passed to boot::boot

**Examples**

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(10)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(10, 0, 2), Sigma))

# estimate
est.sparcc <- sparccboot(data, R=100)
mean(est.sparcc$t[,1]) # bootstrap estimate of true correlation
```

---

 sparseiCov

*Sparse/penalized estimators of covariance matrices*


---

**Description**

This function estimates the sparse inverse covariance matrix/matrices given data (typically after clr transformation) and further arguments to huge package functions.

**Usage**

```
sparseiCov(data, method, npn = FALSE, verbose = FALSE, cov.output = TRUE, ...)
```

**Arguments**

data	data matrix with features/OTUs in the columns and samples in the rows. Should be transformed by clr for meaningful results, if the data is compositional
method	estimation method to use as a character string. Currently either 'glasso' or 'mb' (meinshausen-buhlmann)
nbn	perform Nonparanormal (nbn) transformation before estimation?
verbose	print progress to standard out
cov.output	return the covariance matrix as well.
...	further arguments to huge/estimation functions. See details.

**Details**

This is a wrapper function for sparse iCov estimations performed by glasso in the huge package.

Therefore, arguments ... should be named. Typically, these are for specifying a penalty parameter, lambda, or the number of penalties to use. By default 10 penalties are used, ranging logarithmically between  $\lambda_{\min} \cdot \text{ratio} \cdot \text{MAX}$  and MAX. Max is the theoretical upper bound on lambda and is  $\max |S|$ , the maximum absolute value in the data correlation matrix.  $\lambda_{\min} \cdot \text{ratio}$  is  $1e-3$  by default. Lower values of lambda require more memory/cpu time to compute, and sometimes huge will throw an error.

The argument nlambda determines the number of penalties - somewhere between 10-100 is usually good, depending on how the values of empirical correlation are distributed.

**Value**

Sparse inverse covariance estimation result object

**Examples**

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(50)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(50, 0, 2), Sigma))

# normalize
data.f <- t(apply(data, 1, norm_to_total))
data.clr <- t(clr(data.f, 1))

# estimate
est.clr <- sparseiCov(data.clr, method='glasso')
est.f <- sparseiCov(data.f, method='glasso')
est.log <- sparseiCov(log(data), method='glasso')

# visualize results
par(mfrow=c(1,3))
image(as.matrix(est.log$path[[3]][1:5,1:5]))
image(as.matrix(est.clr$path[[3]][1:5,1:5]))
image(as.matrix(est.f$path[[3]][1:5,1:5]))
```

---

sparseLowRankiCov      *Sparse plus Low Rank inverse covariance*

---

### Description

Select an inverse covariance matrix that is a sparse plus low rank decomposition.

### Usage

```
sparseLowRankiCov(data, npn = FALSE, verbose = FALSE, cor = FALSE, ...)
```

### Arguments

data	the n x p data matrix
npn	flag to first fit nonparametric normal transform to the data
verbose	flag to turn on verbose output
cor	flag to use correlation matrix as the input (default: false - uses covariance)
...	arguments to override default algorithm settings (see details)

### Details

This is a wrapper function for sparse plus low rank iCov estimations performed by a custom ADMM algorithm.

Therefore, arguments ... should be named. Typically, these are for specifying a penalty parameter, lambda, or the number of penalties to use. By default 10 penalties are used, ranging logarithmically between lambda.min.ratio\*MAX and MAX. Max is the theoretical upper bound on lambda and is max|S|, the maximum absolute value in the data correlation matrix. lambda.min.ratio is 1e-3 by default. Lower values of lambda require more memory/cpu time to compute, and sometimes huge will throw an error.

The argument nlambda determines the number of penalties - somewhere between 10-100 is usually good, depending on how the values of empirical correlation are distributed. #' @export

One of beta (penalty for the nuclear norm) or r (number of ranks) should be supplied or r=2 is chosen by default.

### Examples

```
# simulate data with 1 negative correlation
set.seed(10010)
Sigma <- diag(10)*2
Sigma[1,2] <- Sigma[2,1] <- -.9
data <- exp(rmvnorm(50, runif(10, 0, 2), Sigma))

# normalize
data.f <- t(apply(data, 1, norm_to_total))
data.clr <- t(cclr(data.f, 1))

# estimate
est.clr <- sparseLowRankiCov(data.clr, cor=TRUE, r=2)
est.f <- sparseLowRankiCov(data.f, cor=TRUE, r=2)
est.log <- sparseLowRankiCov(log(data), cor=TRUE, r=2)
```

```
# visualize results
par(mfrow=c(1,3))
image(as.matrix(est.log$path[[6]][1:5,1:5]))
image(as.matrix(est.clr$path[[6]][1:5,1:5]))
image(as.matrix(est.f$path[[6]][1:5,1:5]))
```

---

spiec.easi

*SPIEC-EASI pipeline*


---

## Description

Run the whole SPIEC-EASI pipeline, from data transformation, sparse inverse covariance estimation and model selection. Inputs are a non-normalized OTU table and pipeline options.

## Usage

```
spiec.easi(data, ...)

## S3 method for class 'phyloseq'
spiec.easi(data, ...)

## S3 method for class 'otu_table'
spiec.easi(data, ...)

## Default S3 method:
spiec.easi(
  data,
  method = "glasso",
  sel.criterion = "stars",
  verbose = TRUE,
  pulsar.select = TRUE,
  pulsar.params = list(),
  icov.select = pulsar.select,
  icov.select.params = pulsar.params,
  lambda.log = TRUE,
  ...
)
```

## Arguments

data	For a matrix, non-normalized count OTU/data table with samples on rows and features/OTUs in columns. Can also be phyloseq or otu_table object.
...	further arguments to sparse inverse covariance estimation
method	estimation method to use as a character string. Currently either 'glasso' or 'mb' (meinshausen-buhlmann's neighborhood selection)
sel.criterion	character string specifying criterion/method for model selection. Accepts 'stars' [default], 'bstars' (Bounded StARS)
verbose	flag to show progress messages
pulsar.select	flag to perform model selection. Choices are TRUE/FALSE/'batch'

`pulsar.params` list of further arguments to pulsar model selection. See the documentation for [pulsar.params](#).  
`icov.select` deprecated.  
`icov.select.params` deprecated.  
`lambda.log` should values of lambda be distributed logarithmically (TRUE) or linearly (FALSE) between `lambda.min` and `lambda.max`?

**Value**

SPIEC-EASI result object

**See Also**

[multi.spiec.easi](#)

**Examples**

```

# Generate random data
data <- exp(matrix(rnorm(100), nrow=10))

# Run SPIEC-EASI
result <- spiec.easi(data)
  
```

---

stars.roc

*stars.roc, stars.pr*

---

**Description**

Plot a ROC (receiver operator characteristic) or a Precision-Recall curve along the stars 'confidence path'. Each edge is a number in [0,1], which is on the fraction of inferred graphs over subsamples in which that edge appeared in stars.

**Usage**

```
stars.roc(optmerge, theta, verbose = TRUE, plot = TRUE, ll = 15)
```

```
stars.pr(optmerge, theta, verbose = TRUE, plot = TRUE, ll = 15)
```

**Arguments**

`optmerge` the optimal 'merge' matrix selected by stars  
`theta` the true graph or precision matrix  
`verbose` display messages  
`plot` graph the output  
`ll` number of points for the plot

**Value**

ROC curve object

**Examples**

```
# Create sample data and run spiec.easi
data(amgut1.filt)
est <- spiec.easi(amgut1.filt, method='glasso', nlambda=10)
# Create a simple true graph for demonstration
true_graph <- matrix(0, ncol(amgut1.filt), ncol(amgut1.filt))
true_graph[1,2] <- true_graph[2,1] <- 1
# Plot ROC curve
roc_result <- stars.roc(getOptMerge(est), true_graph)
```

symBeta

*sym beta***Description**

Symmetrize a beta (coefficient) matrix, ie. selected from MB neighborhood selection

**Usage**

```
symBeta(beta, mode = "ave")
```

**Arguments**

beta	square coefficient matrix
mode	how to symmetrize, see details

**Details**

Mode can be:

**ave** Arithmetic average between the two possible values of beta

**maxabs** The maximum [absolute] value between the two values

**upper** Take the values from the upper triangle

**lower** Take the values from the lower triangle

**Value**

a symmetric coefficient matrix

**Examples**

```
# Create an asymmetric coefficient matrix
beta <- matrix(c(0, 0.5, 0.2, 0.3, 0, 0.1, 0, 0.4, 0), nrow=3)

# Symmetrize using different methods
sym_ave <- symBeta(beta, mode='ave') # Average
sym_max <- symBeta(beta, mode='maxabs') # Maximum absolute
sym_upper <- symBeta(beta, mode='upper') # Upper triangle
```

---

 synth\_comm\_from\_counts

*synth\_comm\_from\_counts*


---

### Description

from count data (ex HMP) fit parameters to OTU margins and simulate a new community with those properties

### Usage

```
synth_comm_from_counts(
  comm,
  mar = 2,
  distr,
  Sigma = cov(comm),
  params,
  n = nrow(comm),
  retParams = FALSE,
  ...
)
```

### Arguments

comm	community: matrix of counts
mar	the sample margin of the community data matrix (1: rows, 2: cols)
distr	distribution to fit (see fitdistr)
Sigma	covariance structure (defaults to empirical cov of comm)
params	optionally supply already fitted parameters
n	number of samples (defaults to comm samples)
retParams	if TRUE, return the fitted parameters
...	additional parameters to parameter fitting

### Value

community

### Examples

```
# Create a simple community matrix
comm <- matrix(rpois(20, lambda=5), nrow=4, ncol=5)
# Simulate new community using Poisson distribution
new_comm <- synth_comm_from_counts(comm, distr="pois")
# Simulate using negative binomial with custom parameters
params <- get_comm_params(comm, distr="negbin")
new_comm_nb <- synth_comm_from_counts(comm, distr="negbin", params=params)
```

---

`triu`*Functions for triangular matrices*

---

**Description**

Get or symmeterize the upper/lower triangle of a symmetric matrix with the other side zeroed out

**Usage**

```
triu(x, k = 1)
```

```
triu(x, k = 1)
```

```
tril(x, k = 1)
```

```
triu2diag(x, diagval = 0)
```

**Arguments**

`x` the data matrix or vector

`k` (0/1 flag indicate diagonal should be selected)

`diagval` value to be added to the diagonal if converting from upper triangular matrix.

**Value**

Upper triangular elements of matrix

Upper triangular elements of matrix

**Examples**

```
# Create a symmetric matrix
mat <- matrix(1:16, 4, 4)
mat <- mat + t(mat) # Make symmetric
# Get upper triangular elements
upper_elements <- triu(mat)
```

# Index

## \* internal

- SpiecEasi-package, 3
- triu, 37
  
- adj2igraph, 4
- AGP, 4
- alr, 5
- amgut1.filt (AGP), 4
- amgut2.filt.phy (AGP), 4
- as.data.frame.graph, 6
- as.matrix.graph, 7
  
- clr, 7
- coat, 8
- cor2cov, 9
- cov2prec, 10
  
- ebic, 10
- edge.diss, 11
  
- fitdistr, 12
  
- get\_comm\_params, 14
- getOptBeta (getOptInd), 13
- getOptCov (getOptInd), 13
- getOptiCov (getOptInd), 13
- getOptInd, 13
- getOptLambda (getOptInd), 13
- getOptMerge (getOptInd), 13
- getOptNet (getOptInd), 13
- getRefit (getOptInd), 13
- getStability (getOptInd), 13
- graph2prec, 14
  
- hmp2, 15
- hmp216S (hmp2), 15
- hmp2prot (hmp2), 15
  
- make\_graph, 16
- multi.spiec.easi, 16, 34
  
- neff, 18
- neighborhood.net, 18
- norm\_pseudo, 19
- norm\_rdiric, 20
  
- norm\_to\_total, 20
  
- prec2cov, 21
- pulsar.params, 17, 21, 34
- pval.sparccboot, 22
  
- qqdplot\_comm, 22
  
- rmvnegbin, 23
- rmvnorm, 24
- rmvpois, 25
- rmvzinegbin, 25
- rmvzipois, 26
- robustPCA, 27
- rzipois, 28
  
- shannon, 28
- sparcc, 29
- sparccboot, 29, 29
- sparseiCov, 30
- sparseLowRankiCov, 32
- spiec.easi, 13, 17, 21, 22, 33
- spiec.easi.list (multi.spiec.easi), 16
- SpiecEasi (SpiecEasi-package), 3
- SpiecEasi-package, 3
- stars.pr (stars.roc), 34
- stars.roc, 34
- symBeta, 35
- synth\_comm\_from\_counts, 36
  
- tril (triu), 37
- triu, 37
- triu2diag (triu), 37