

# Package ‘epiNEM’

May 15, 2022

**Type** Package

**Title** epiNEM

**Version** 1.21.0

**Author** Madeline Diekmann & Martin Pirkl

**Maintainer** Martin Pirkl <martinpirkl@yahoo.de>

**Description** epiNEM is an extension of the original Nested Effects Models (NEM). EpiNEM is able to take into account double knockouts and infer more complex network signalling pathways. It is tailored towards large scale double knock-out screens.

**Depends** R (>= 4.1)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**biocViews** Pathways, SystemsBiology, NetworkInference, Network

**RoxygenNote** 7.1.1

**Imports** BoolNet, e1071, gtools, stats, igraph, utils, lattice, latticeExtra, RColorBrewer, pcalg, minet, grDevices, graph, mnem, latex2exp

**VignetteBuilder** knitr

**Suggests** knitr, RUnit, BiocGenerics, STRINGdb, devtools, rmarkdown, GOSemSim, AnnotationHub, org.Sc.sgd.db

**BugReports** <https://github.com/cbg-ethz/epiNEM/issues>

**URL** <https://github.com/cbg-ethz/epiNEM/>

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

**git\_branch** master

**git\_last\_commit** 15310ae

**git\_last\_commit\_date** 2022-04-26

**Date/Publication** 2022-05-15

**R topics documented:**

|                                   |           |
|-----------------------------------|-----------|
| AddLogicGates . . . . .           | 2         |
| CreateExtendedAdjacency . . . . . | 3         |
| CreateRandomGraph . . . . .       | 4         |
| CreateTopology . . . . .          | 4         |
| epiAnno . . . . .                 | 5         |
| epiNEM . . . . .                  | 5         |
| epiScreen . . . . .               | 7         |
| ExtendTopology . . . . .          | 8         |
| GenerateData . . . . .            | 8         |
| HeatmapOP . . . . .               | 9         |
| Mll . . . . .                     | 11        |
| perm.rank.test . . . . .          | 12        |
| plot.epiNEM . . . . .             | 13        |
| plot.epiScreen . . . . .          | 14        |
| plot.epiSim . . . . .             | 15        |
| rank.enrichment . . . . .         | 15        |
| sameith_GO . . . . .              | 17        |
| sameith_string . . . . .          | 17        |
| samscreen . . . . .               | 17        |
| sim . . . . .                     | 18        |
| SimEpiNEM . . . . .               | 18        |
| wageningen_GO . . . . .           | 19        |
| wageningen_string . . . . .       | 19        |
| wagscreen . . . . .               | 20        |
| <b>Index</b>                      | <b>21</b> |

---

|               |                   |
|---------------|-------------------|
| AddLogicGates | <i>Add logic.</i> |
|---------------|-------------------|

---

**Description**

extend model with node representing logic gate

**Usage**

```
AddLogicGates(child, logic, model)
```

**Arguments**

|       |                         |
|-------|-------------------------|
| child | define the child        |
| logic | define the logical gate |
| model | normal model            |

**Value**

model list with additional logic gate

**Examples**

```
model <- CreateRandomGraph(c("Ikk1", "Ikk2", "Re1A"))
model2 <- AddLogicGates("Re1A", "OR", model)
```

---

CreateExtendedAdjacency

*Create an extended adjacency matrix*

---

**Description**

extend adjacency matrices taking cycles and logics into account. For every given start state, the final state is computed using BoolNet.

**Usage**

```
CreateExtendedAdjacency(network, mutants, experiments)
```

**Arguments**

|             |                                      |
|-------------|--------------------------------------|
| network     | network created by BoolNet from file |
| mutants     | vector of single knockouts           |
| experiments | vector of all knockouts              |

**Value**

extended adjacency matrix

**Examples**

```
library(BoolNet)
data(cellcycle)
extModel <- CreateExtendedAdjacency(cellcycle,
c(cellcycle$genes, "CycD.Rb"), cellcycle$genes)
```

CreateRandomGraph      *Create a random graph*

---

**Description**

Returns a model graph with randomly sampled edges. Every possible edge has a probability to exist in the graph.

**Usage**

```
CreateRandomGraph(pathwayGenes, edgeProb = 0.5)
```

**Arguments**

pathwayGenes      vector of genes in the pathway  
edgeProb            probability of random edge

**Value**

adjacency matrix

**Examples**

```
graph <- CreateRandomGraph(c("Ikk1", "Ikk2", "RelA"))
```

---

CreateTopology      *Create Topology.*

---

**Description**

Create topology for a randomly generated pathway topology

**Usage**

```
CreateTopology(single, double, force = TRUE)
```

**Arguments**

single              number of single knockouts  
double              number of double knockouts  
force                if true the random model will have a sophisticated logical gate

**Value**

adjacency matrix

**Examples**

```
model <- CreateTopology(3, 1)
```

---

epiAnno                      *Gate visualisation.*

---

**Description**

Plots logical gate data annotation. The 8 heatmaps visualize what perfect data would look like in respective to each logical gate. Perfect data is equivalent to Boolean truth tables.

**Usage**

```
epiAnno()
```

**Value**

plot of heatmaps showing the silencing scheme (=expected data, truth tables)

**Author(s)**

Martin Pirkl

**References**

[https://en.wikipedia.org/wiki/Boolean\\_algebra](https://en.wikipedia.org/wiki/Boolean_algebra)

**Examples**

```
epiAnno()
```

---

epiNEM                      *Epistatic NEMs - main function.*

---

**Description**

This function contains the inference algorithm to learn logical networks from knock-down data including double knock-downs.

**Usage**

```
epiNEM(
  filename = "random",
  method = "greedy",
  nIterations = 10,
  nModels = 0,
  random = list(single = 4, double = 1, reporters = 100, FPrate = 0.1, FNrate = 0.1,
    replicates = 1),
  ltype = "marginal",
  para = c(0.13, 0.05),
  init = NULL
)
```

**Arguments**

|             |  |
|-------------|--|
| filename    | A binary, tab-delimited matrix. Columns: single and double knockdowns. Rows: genes showing effect or not? Default: random; artificial data is generated to 'random' specifications |
| method      | greedy or exhaustive search. Default: greedy   |
| nIterations | number of iterations. Default: 10  |
| nModels     | number of Models. Default: 0   |
| random      | list specifying how the data should be generated: no. of single mutants, no. of double mutants, no. of reporterGenes, FP-rate, FN-rate, no. of replicates                          |
| ltype       | likelihood either "marginal" or "maximum"  |
| para        | false positive and false negative rates  |
| init        | adjacency matrix to initialise the greedy search   |

**Value**

List object with an adjacency matrix denoting the network, the model of the silencing scheme (rows are knock-downs, columns are signalling genes), a string with the inferred logical gates, a column indices denoting position of logical gates, the log transformed likelihood and the effect reporter distribution (rows are the signalling genes including the null node).

**Author(s)**

Madeline Diekmann

**See Also**

nem

**Examples**

```
data <- matrix(sample(c(0,1), 100*4, replace = TRUE), 100, 4)
colnames(data) <- c("A", "A.B", "B", "C")
rownames(data) <- paste("E", 1:100, sep = "_")
```

```
res <- epiNEM(data, method = "exhaustive")  
plot(res)
```

---

epiScreen

*Analyse large double knock-out screen.*

---

## Description

This function is used to analyse knock-out screens with multiple double and single knock-outs combined in one data set.

## Usage

```
epiScreen(data, ...)
```

## Arguments

|      |   |
|------|---|
| data | data matrix containing multiple single and double knock-downs in columns and effect reporters in the rows |
| ...  | additional parameters, e.g. for the main epiNEM function  |

## Value

list object with vectors of double knock-downs, single knock-downs and two matrices with doubles in the columns and singles in the rows. The first matrix denotes the respective logical gate for the triple and the second matrix the log-likelihood

## Author(s)

Martin Pirkl

## Examples

```
data <- matrix(sample(c(0,1), 100*9, replace = TRUE), 100, 9)  
colnames(data) <- c("A.B", "A.C", "B.C", "A", "B", "C", "D", "E", "G")  
rownames(data) <- paste("E", 1:100, sep = "_")  
res <- epiScreen(data)
```

ExtendTopology            *Extending topology of normal "nem"*

---

**Description**

Extending topology of normal "nem"

**Usage**

```
ExtendTopology(topology, nReporters)
```

**Arguments**

topology            model of a topology from CreateTopology  
nReporters          number of effects reporters

**Value**

extended topology in which reporters are linked to pathway genes

**Author(s)**

Madeline Diekmann

**See Also**

CreateTopology

**Examples**

```
topology <- CreateTopology(3, 1, force = TRUE)  
topology <- unlist(unique(topology), recursive = FALSE)  
extTopology <- ExtendTopology(topology$model, 100)
```

---

GenerateData            *Generate data from extended model.*

---

**Description**

Given a model created from CreateTopology and ExtendTopology, this function creates a corresponding artificial data matrix, which is used as a ground truth for simulation studies.

**Usage**

```
GenerateData(model, extTopology, FPrate, FNrate, replicates)
```



**Arguments**

|             |   |
|-------------|---|
| model       | model of a topology from CreateTopology |
| extTopology | extended topology                       |
| FPrate      | false positive rate                     |
| FNrate      | false negative rate                     |
| replicates  | number of replicates                    |

**Value**

data matrix with effect reporters as rows and knock-downs (including double knock-downs) as columns.

**Author(s)**

Madeline Diekmann

**See Also**

CreateTopology

**Examples**

```
topology <-  
CreateTopology(3, 1, force = TRUE)  
topology <-  
unlist(unique(topology), recursive = FALSE)  
extTopology <-  
ExtendTopology(topology$model, 100)  
sortedData <-  
GenerateData(topology$model, extTopology, 0.05, 0.13, 3)
```

---

HeatmapOP

*Heatmap.*

---

**Description**

Heatmap function based on the lattice package more information: ?xyplot

**Usage**

```
HeatmapOP(  
  x,  
  col = "RdYlGn",  
  colNA = "grey",  
  coln = 11,  
  bordercol = "grey",  
  borderwidth = 0.1,
```

```

breaks = "sym",
main = "",
sub = "",
dendrogram = "none",
colorkey = "right",
Colv = TRUE,
Rowv = TRUE,
xrot = 90,
yrot = 0,
shrink = c(1, 1),
cexCol = 1,
cexRow = 1,
cexMain = 1,
cexSub = 1,
colSideColors = NULL,
aspect = "fill",
contour = FALSE,
useRaster = FALSE,
xlab = NULL,
ylab = NULL,
colSideColorsPos = "top",
clust = NULL,
clusterx = NULL,
...
)

```

### Arguments

|             |  |
|-------------|--|
| x           | Matrix.  |
| col         | Color. See <a href="#">brewer.pal.info</a> for all available color schemes. Alternatively, any number of colors, which are then used to create a color gradient. E.g., <code>c('blue', 'red')</code> produces a color scheme with a gradient from blue to red. |
| colNA       | color for NAs; default is grey   |
| coln        | Number of colors.  |
| bordercol   | Border color.  |
| borderwidth | Border width.  |
| breaks      | Defines the breaks in the color range. "sym" makes the breaks symmetric around 0.  |
| main        | Main title.  |
| sub         | Subtitle.  |
| dendrogram  | Draw dendrogram with "both", "col" or "row", or do not draw with "none".   |
| colorkey    | Draw colorkey "left", "right" (default), "top", "bottom" or NULL for no colorkey. See <code>?lattice::levelplot</code> for more complex colorkey options.  |
| Colv        | Cluster columns (TRUE) or not (FALSE).   |
| Rowv        | Cluster rows (TRUE) or not (FALSE).  |

|                  |  |
|------------------|--|
| xrot             | Rotate the column names by degree.   |
| yrot             | Rotate the row names by degree.  |
| shrink           | c(x,y) defines a range of size for the data boxes from low to high.  |
| cexCol           | Font size of column names.   |
| cexRow           | Font size of row names.  |
| cexMain          | Font size of main title.   |
| cexSub           | Font size of subtitle.   |
| colSideColors    | Defines a numeric vector to annotate columns with different colors.  |
| aspect           | "iso" for quadratic boxes or "fil" for stretched boxes.  |
| contour          | TRUE adds a contour plot.  |
| useRaster        | TRUE to add raster visuals   |
| xlab             | Label for the x-axis.  |
| ylab             | Label for the y-axis.  |
| colSideColorsPos | Place colSideColors at the "top" or "bottom".  |
| clust            | p, s, or k for correlation clustering  |
| clusterx         | Optional data matrix y with the same dimensions as x. x is columns or rows are sorted by the cluster information of y. |
| ...              | Optional arguments.  |

**Value**

lattice object/matrix

**Author(s)**

Martin Pirkl & Oscar Perpinan at <http://oscarperpinan.github.io/rastervis/>

**Examples**

```
x <- matrix(rnorm(50), 10, 5)
HeatmapOP(x, dendrogram = "both", aspect = "iso", xrot = 45)
```

**Description**

Computes marginal log-likelihood for model Phi given observed data matrix D1

**Usage**

```
M11(Phi, D1, D0, ltype = "marginal", para = c(0.13, 0.05))
```

**Arguments**

|       |  |
|-------|--|
| Phi   | model to be evaluated                          |
| D1    | observed data matrix                           |
| D0    | complementary D1                               |
| ltype | likelihood type either "marginal" or "maximum" |
| para  | false positive and false negative rates        |

**Value**

list with likelihood poster probability, egene positions

**Examples**

```
Phi <- matrix(sample(c(0,1), 9, replace = TRUE), 3, 3)
data <- matrix(sample(c(0,1), 3*10, replace = TRUE), 10, 3)
rownames(Phi) <- colnames(Phi) <- colnames(data) <- c("Ikk1", "Ikk2", "RelA")
score <- M11(Phi, D1 <- data, D0 <- 1 - data)
```

---

|                |                             |
|----------------|-----------------------------|
| perm.rank.test | <i>AUC permutation test</i> |
|----------------|-----------------------------|

---

**Description**

computes the area under the rank enrichment score curve and does a permutation test to compute the p-value

**Usage**

```
perm.rank.test(
  x,
  y = NULL,
  alternative = c("two.sided", "less", "greater"),
  iter = 1000
)
```

**Arguments**

|             |   |
|-------------|---|
| x           | numeric vector of ranks                                 |
| y           | numeric vector of the superset of x                     |
| alternative | character for test type: 'less', 'greater', 'two.sided' |
| iter        | integer number of iterations                            |

**Value**

p-value

**Author(s)**

Martin Pirkl

**Examples**

```
x <- 1:10
y <- 1:100
perm.rank.test(x,y,alternative='less')
perm.rank.test(x,y,alternative='greater')
```

---

`plot.epiNEM`*Plot pathway.*

---

**Description**

Plots the winning pathway structure

**Usage**

```
## S3 method for class 'epiNEM'
plot(x, ...)
```

**Arguments**

|                  |                                     |
|------------------|-------------------------------------|
| <code>x</code>   | object of class <code>epiNEM</code> |
| <code>...</code> | other arguments                     |

**Value**

plot of the logical network

**Examples**

```
data <- matrix(sample(c(0,1), 100*4, replace = TRUE), 100, 4)
colnames(data) <- c("A", "A.B", "B", "C")
rownames(data) <- paste("E", 1:100, sep = "_")
res <- epiNEM(data, method = "exhaustive")
plot(res)
```

---

|                |                     |
|----------------|---------------------|
| plot.epiScreen | <i>Plot screen.</i> |
|----------------|---------------------|

---

### Description

Plots the results of a systematic knock-out screen

### Usage

```
## S3 method for class 'epiScreen'
plot(
  x,
  global = TRUE,
  ind = NULL,
  colorkey = TRUE,
  cexGene = 1,
  off = 0.05,
  cexLegend = 1,
  ...
)
```

### Arguments

|           |   |
|-----------|---|
| x         | object of class epiScreen   |
| global    | plot global distribution or for each pair (FALSE)                   |
| ind       | index of pairs to plot  |
| colorkey  | if TRUE prints colorkey   |
| cexGene   | size of modulator annotation  |
| off       | relative distance from the gene names to the respective likelihoods |
| cexLegend | font size of the legend   |
| ...       | other arguments   |

### Value

plot(s) of an epiNEM screen analysis

### Examples

```
data <- matrix(sample(c(0,1), 100*9, replace = TRUE), 100, 9)
colnames(data) <- c("A.B", "A.C", "B.C", "A", "B", "C", "D", "E", "G")
rownames(data) <- paste("E", 1:100, sep = "_")
res <- epiScreen(data)
plot(res)
plot(res, global = FALSE, ind = 1:3)
```

---

|             |                          |
|-------------|--------------------------|
| plot.epiSim | <i>Plot simulations.</i> |
|-------------|--------------------------|

---

**Description**

Plots the simulation results

**Usage**

```
## S3 method for class 'epiSim'  
plot(x, ...)
```

**Arguments**

|     |                        |
|-----|------------------------|
| x   | object of class epiSim |
| ... | other arguments        |

**Value**

plot(s) of an epiNEM simulation analysis

**Examples**

```
res <- SimEpiNEM(runs = 1)  
plot(res)
```

---

|                 |                        |
|-----------------|------------------------|
| rank.enrichment | <i>Rank enrichment</i> |
|-----------------|------------------------|

---

**Description**

Infers a signalling pathway from peerturbation experiments.

**Usage**

```
rank.enrichment(  
  data,  
  list,  
  list2 = NULL,  
  n = 1000,  
  main = NULL,  
  col1 = "RdBu",  
  col2 = rgb(1, 0, 0, 0.75),  
  col3 = rgb(0, 0, 1, 0.75),  
  blim = NULL,  
  p = NULL,
```

```

    lwd = 3,
    test = wilcox.test,
    vis = "matrix",
    verbose = FALSE,
    ...
)

```

### Arguments

|         |  |
|---------|--|
| data    | m times l matrix with m observed genes and l variables with numeric values to rank the genes   |
| list    | list of of vectors of genes  |
| list2   | optional list with same length as list   |
| n       | length of the gradient (maximum: m)  |
| main    | character string for main header; if NULL uses the column names of data by default   |
| col1    | color of the gradient  |
| col2    | color of the first list  |
| col3    | color of the second list2  |
| blim    | numeric vector of length two with the lower and upper bounds for the gradient  |
| p       | numeric adjustment (length four) of the left side of the gradient (low means more to the left, high more to the right) the right side of the enrichment lines and the top positions of the additional matrices in case of vis='matrices' |
| lwd     | line width of the enrichment lines   |
| test    | test function for the enrichment p-value; must have input argument and output values same as perm.rank.test; e.g., wilcox.test or ks.test (here 'less' and 'greater' are switched!)  |
| vis     | method for visualisation: 'matrix' uses one matrix heatmap for; 'matrices' uses several matrices (experimental), 'colside' uses the colSideColors argument for the ticks of genes in list/list2 (can use a lot of memory; experimental)  |
| verbose | if TRUE gives prints additional output   |
| ...     | additional arguments for epiNEM::HeatmapOP   |

### Value

transitively closed matrix or graphNEL

### Author(s)

Martin Pirkl



**Examples**

```
data <- matrix(rnorm(100*2),100,2)
rownames(data) <- 1:100
colnames(data) <- LETTERS[1:2]
list <- list(first = as.character(sample(1:100, 10)), second = as.character(sample(1:100, 20)))
rank.enrichment(data,list)
```

---

|            |  |
|------------|--|
| sameith_GO | <i>graph-based GO similarity scores, string GO annotations for Sameith et al., 2015 data</i> |
|------------|--|

---

**Description**

The data consists of lists including epiNEM identified and general similarity scores and GO annotations for each triple. For details see the vignette.

**Examples**

```
data(sameith_GO)
```

---

|                |  |
|----------------|--|
| sameith_string | <i>sig. of string interaction scores for Sameith et al., 2015 data</i> |
|----------------|--|

---

**Description**

The data consists of a list including a vectors of pairs (for interactions) and a corresponding list of interaction scores derived from the string database. For details see the vignette.

**Examples**

```
data(sameith_string)
```

---

|           |   |
|-----------|---|
| samscreen | <i>Example data: epiNEM results for the Sameith et al., 2015 knock-out screen</i> |
|-----------|---|

---

**Description**

The result of the epiNEM analysis of the data from "[http://www.holstegelab.nl/publications/sv/signaling\\_redundancy/download](http://www.holstegelab.nl/publications/sv/signaling_redundancy/download)". The data consists of a list of matrices with the likelihoods (ll) for each analysed triple of signalling genes and the inferred logic (logic) for each triple. The signalling genes or modulators C are the rows and the signalling genes from the double knock-downs are in the columns. For details see the vignette.

**Examples**

```
data(samscreen)
```

---

|     |   |
|-----|---|
| sim | <i>Example data: simulation results</i> |
|-----|---|

---

### Description

Contains simulation results. How they were acquired is explained in the vignette. The data consists of a list of data matrices holding sensitivity and specificity (spec, sens) of network edges for the various methods compared to the ground truth, sensitivity and specificity (sens2, spec2) of the expected data for epiNEM and Boolean NEMs and accuracy of the inferred logics for both. The different methods are in the rows and the columns denote the different independent simulation runs.

### Examples

```
data(sim)
```

---

|           |                            |
|-----------|----------------------------|
| SimEpiNEM | <i>Compare algorithms.</i> |
|-----------|----------------------------|

---

### Description

Compares different network reconstruction algorithm on simulated data.

### Usage

```
SimEpiNEM(
  runs = 10,
  do = c("n", "e"),
  random = list(FPrate = 0.1, FNrate = c(0.1, 0.5), single = 3, double = 1, reporters =
    10, replicates = 2),
  maxTime = FALSE,
  forceLogic = TRUE,
  epiNEMsearch = "greedy",
  bNEMsearch = "genetic",
  ...
)
```

### Arguments

|        |   |
|--------|---|
| runs   | number simulation runs  |
| do     | string vector of algorithms to compare: e (epiNEM), n (Nested Effects Models), b (B-NEM), p (PC algorithm), a (Aracne), e.g. c("e", "n", "p")   |
| random | list of false positive rate FPrate, false negative rates FNrate, number of single knock-downs single, number of double knock-downs double, number of effect reporters reporters and number of replicates replicates |

|              |   |
|--------------|---|
| maxTime      | TRUE if the algorithms are bound to a maximum running time in respect to epiNEM               |
| forcelogic   | if TRUE the randomly sampled ground truth network includes a complex logic with probability 1 |
| epinemsearch | greedy or exhaustive search for epiNEM  |
| bnemsearch   | genetic or greedy search for B-NEM  |
| ...          | additional parameters   |

**Value**

returns list of specificity and sensitivity of inferred edges (spec, sens) and inferred expected data (spec2, sens2) and accuracy of logics (logics) and running time (time)

**Author(s)**

Martin Pirkl

**Examples**

```
res <- SimEpiNEM(runs = 1)
```

---

|               |   |
|---------------|---|
| wageningen_GO | <i>graph-based GO similarity scores, string GO annotations for van Wageningen et al., 2015 data</i> |
|---------------|---|

---

**Description**

The data consists of lists including epiNEM identified and general similarity scores and GO annotations for each triple. For details see the vignette.

**Examples**

```
data(wageningen_GO)
```

---

|                   |   |
|-------------------|---|
| wageningen_string | <i>sig. of string interaction scores for van Wageningen et al., 2010 data</i> |
|-------------------|---|

---

**Description**

The data consists of a list including a vectors of pairs (for interactions) and a corresponding list of interaction scores derived from the string database. For details see the vignette.

**Examples**

```
data(wageningen_string)
```

---

wagscreen

*Example data: epiNEM results for the Wageningen et al., 2010 knock-out screen*  
*"[http://www.holstegelab.nl/publications/GSTF\\_geneticinteractions/downloads/del\\_mutants\\_limma.txt](http://www.holstegelab.nl/publications/GSTF_geneticinteractions/downloads/del_mutants_limma.txt)"*

---

### **Description**

The data consists of a list of matrices with the likelihoods (ll) for each analysed triple of signalling genes and the inferred logic (logic) for each triple. The signalling genes or modulators C are the rows and the signalling genes from the double knock-downs are in the columns. For details see the vignette.

### **Examples**

```
data(wagscreen)
```

# Index

AddLogicGates, [2](#)

CreateExtendedAdjacency, [3](#)

CreateRandomGraph, [4](#)

CreateTopology, [4](#)

epiAnno, [5](#)

epiNEM, [5](#)

epiScreen, [7](#)

ExtendTopology, [8](#)

GenerateData, [8](#)

HeatmapOP, [9](#)

M11, [11](#)

perm.rank.test, [12](#)

plot.epiNEM, [13](#)

plot.epiScreen, [14](#)

plot.epiSim, [15](#)

rank.enrichment, [15](#)

sameith\_GO, [17](#)

sameith\_string, [17](#)

samscreen, [17](#)

sim, [18](#)

SimEpiNEM, [18](#)

wageningen\_GO, [19](#)

wageningen\_string, [19](#)

wagscreen, [20](#)