Package 'netjack'

October 13, 2022

Type Package

Title Tools for Working with Samples of Networks

Version 1.2.0

Author Teague Henry

Maintainer Teague Henry <trhenry@email.unc.edu>

Imports ggplot2, igraph, brainGraph, methods,Rdpack

Depends R (>= 3.6)

RdMacros Rdpack

Description Tools for managing large sets of network data and performing whole network analysis. This package is focused on the network based statistic jackknife method, and implements a framework that can be extended to other network manipulations and analyses.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-07 17:50:03 UTC

R topics documented:

`as_Net` *Constructor for single Net object*

Description

This function takes a single network, as an adjacency matrix, and returns a `Net` object.

Usage

```
as_Net(matrix, net.name, node.variables)
```

Arguments

`matrix`	Network represented as an adjacency matrix
`net.name`	Name of the network (optional)
`node.variables`	Node level variables (optional)

Value

A `Net` object

Examples

```
data(GroupA)
GroupA1_Net = as_Net(GroupA[[1]], "1", list(group = c(rep(1, 10), rep(2,10))))
```

as_NetSample *Constructor for a* NetSample *object*

Description

This function takes a list of adjacency matrices, and returns a NetSample object.

Usage

```
as_NetSample(matrixList, net.names, node.variables, sample.variables)
```

Arguments

matrixList	A list of adjacency matrices
net.names	A character vector of network names
node.variables	A list of node level variables to be associated with every network in the sample.
sample.variables	A list of network level variables.

Value

A NetSample instance.

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
 sample.variables = list(group = c(rep(1, 10), rep(2,10))))
```

diff_test *Test for differences from original statistic*

Description

This function tests for significant differences from the original network statistic as a result of the network manipulation. If non-parametric is chosen, this is done using the Wilcox test, otherwise, t-test.

Usage

```
diff_test(netSampleStatSet, p.adjust = "BH", non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
p.adjust	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Value

A data frame containing original and adjusted p.values, as well as differences, labeled with manipulation name.

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
diff_test(GlobEff_GroupA_Net)
```

---

diff_test_ggPlot *Difference Test Plots*

Description

This function performs the difference test and generates a ggplot object representing the results.

Usage

```
diff_test_ggPlot(netSampleStatSet, labels, sort = "alpha",
  p.threshold = 0.05, p.adjust = "BH", hide.non.sig = F,
  non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
labels	ggplot2 labs object. Labels for the plot
sort	one of "alpha", "mean", "median". "alpha" sorts in alpha numeric order, while mean and median sort by decreasing values.
p.threshold	Numeric. Threshold by which to highlight results. Defaults to .05
p.adjust	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
hide.non.sig	Logical. If true, non significant (as defined by p.threshold) are not plotted.
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Value

A ggplot object

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
diff_test_ggPlot(GlobEff_GroupA_Net)
```

GroupA *Simulated Dataset of 20 networks. Group A.*

Description

A simulated dataset of 20 binary networks with 20 nodes each.

Usage

```
GroupA
```

Format

A 'list' of 20 'matrix' objects, representing adjacency matrices. Node 10 is simulated to be important for global efficiency

---

GroupB *Simulated Dataset of 20 networks. Group B.*

Description

A simulated dataset of 20 binary networks with 20 nodes each. Node 15 is simulated to be important for global efficiency.

Usage

```
GroupB
```

Format

A 'list' of 20 'matrix' objects, representing adjacency matrices.

`group_diff_test` *Group difference test*

Description

This function implements the group difference test on a network statistic. This test assesses if the change in the network statistic due to the network manipulation is significantly different between groups.

Usage

```
group_diff_test(netSampleStatSet, grouping.variable, p.adjust = "BH",
  non.parametric = F)
```

Arguments

`netSampleStatSet`	Input `NetSampleStatSet`
`grouping.variable`	character name of sample level grouping variable
`p.adjust`	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
`non.parametric`	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Details

If the sample has 2 groups, this test is performed using a t-test or Wilcox test. If the sample has 3 or more groups, the test is performed using a 1-way ANOVA, or Kruskal-Wallis test. Differences are tested at each network manipulation.

Value

A data frame containing original and adjusted p.values.

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_diff_test(GlobEff_GroupA_Net, grouping.variable = "group")
```

group_diff_test_ggPlot *Group Difference Plots*

Description

This function performs the group difference test and generates a ggplot object representing the results.

Usage

```
group_diff_test_ggPlot(netSampleStatSet, grouping.variable, labels,
  sort = "alpha", p.threshold = 0.05, p.adjust = "BH",
  hide.non.sig = F, non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
grouping.variable	character name of sample level grouping variable
labels	ggplot2 labs object. Labels for the plot
sort	one of "alpha", "mag"; "alpha" sorts in alpha numeric order, while "mag" sorts in order of decreasing effect size
p.threshold	Numeric. Threshold by which to highlight results. Defaults to .05
p.adjust	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
hide.non.sig	Logical. If true, non significant (as defined by p.threshold) are not plotted.
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Value

A ggplot object

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_diff_test_ggPlot(GlobEff_GroupA_Net, "group")
```

`group_perc_diff_test` *Group percentage difference test*

Description

This function implements the group percentage difference test on a network statistic. This test assesses if the percent change in the network statistic due to the network manipulation is significantly different between groups. Percent change is calculated as the difference between the target and original statistic divided by the original statistic.

Usage

```
group_perc_diff_test(netSampleStatSet, grouping.variable,
  p.adjust = "BH", non.parametric = F)
```

Arguments

`netSampleStatSet`	Input `NetSampleStatSet`
`grouping.variable`	character name of sample level grouping variable
`p.adjust`	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
`non.parametric`	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Details

If the sample has 2 groups, this test is performed using a t-test or Wilcox test. If the sample has 3 or more groups, the test is performed using a 1-way ANOVA, or Kruskal-Wallis test. Differences are tested at each network manipulation.

Value

A data frame containing original and adjusted p.values.

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_diff_test(GlobEff_GroupA_Net, grouping.variable = "group")
```

group_perc_diff_test_ggPlot *Group Percentage Difference Plots*

Description

This function performs the group percentage difference test and generates a ggplot object representing the results.

Usage

```
group_perc_diff_test_ggPlot(netSampleStatSet, grouping.variable, labels,
  sort = "alpha", p.threshold = 0.05, p.adjust = "BH",
  hide.non.sig = F, non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
grouping.variable	character name of sample level grouping variable
labels	ggplot2 labs object. Labels for the plot
sort	one of "alpha", "mag"; "alpha" sorts in alpha numeric order, while "mag" sorts in order of decreasing effect size
p.threshold	Numeric. Threshold by which to highlight results. Defaults to .05
p.adjust	character string for requested multiple comparisons adjustment. Defaults to Benjamani-Hochberg
hide.non.sig	Logical. If true, non significant (as defined by p.threshold) are not plotted.
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Value

A ggplot object

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_perc_diff_test_ggPlot(GlobEff_GroupA_Net, "group")
```

group_test *Group test*

Description

This function implements the group test on a network statistic. This test assesses if the network statistic is significantly different between groups, at each network manipulation.

Usage

```
group_test(netSampleStatSet, grouping.variable, p.adjust = "none",
  non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
grouping.variable	character name of sample level grouping variable
p.adjust	character string for requested multiple comparisons adjustment. Defaults to none.
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Details

If the sample has 2 groups, this test is performed using a t-test or Wilcox test. If the sample has 3 or more groups, the test is performed using a 1-way ANOVA, or Kruskal-Wallis test. Differences are tested at each network manipulation.

Value

A data frame containing original and adjusted p.values.

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_test(GlobEff_GroupA_Net, grouping.variable = "group")
```

group_test_ggPlot *Group Test Plots*

Description

This function performs the group test and generates a ggplot object representing the results.

Usage

```
group_test_ggPlot(netSampleStatSet, grouping.variable, labels,
  sort = "alpha", p.threshold = 0.05, p.adjust = "BH",
  hide.non.sig = F, non.parametric = F)
```

Arguments

netSampleStatSet	Input NetSampleStatSet
grouping.variable	character name of sample level grouping variable
labels	ggplot2 labs object. Labels for the plot
sort	one of "alpha", "mag"; "alpha" sorts in alpha numeric order, while "mag" sorts in order of decreasing effect size
p.threshold	Numeric. Threshold by which to highlight results. Defaults to .05
p.adjust	character string for requested multiple comparisons adjustment. Defaults to none.
hide.non.sig	Logical. If true, non significant (as defined by p.threshold) are not plotted.
non.parametric	Logical. if true, test is performed using Wilcox test. If false, t-test. Defaults to false.

Value

A ggplot object

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
group_test_ggPlot(GlobEff_GroupA_Net, "group")
```

Net-class *An S4 class to represent a single network.*

Description

This class represents a single observation of a network, with associated node level variables.

Details

For constructor see: as_Net

Slots

`net` matrix. The network represented as an adjacency matrix.

`net.name` character. The name of the network (e.g. subject ID, school name)

`node.variables` list. A named list of node variables, in the same order as the adjacency matrix.

NetSample-class *An S4 class to represent a sample of networks*

Description

This class represents a collection of networks, with associated network level variables.

Details

For constructor see: as_NetSample

Slots

`nets` list. A list of Net objects

`net.names` character. A character vector representing the names of the net objects

`sample.variables` list. A named list of network level variables.

`NetSampleSet-class` *An S4 class representing a sample of networks with a network permutation function applied to it.*

Description

This class represents the results of applying a network permutation function, such as a jackknife, or rewiring algorithm, to a sample of networks.

Slots

`net.sets` list. A list of NetSet objects, reach representing a network, and the results of applying the permutation function.

`net.names` character. A character vector representing the names of the original network in the sample

`sample.variables` list. A list representing sample level variables.

`NetSampleStatSet-class` *An S4 class representing the results of applying a network statistic function to a NetSampSet object.*

Description

This class represents the results of applying a network statistic function to a NetSampSet object. This class contains the results for the original networks, as well as for each instance of the permuted/manipulated networks.

Slots

`stat.fun` function. The network statistic function applied

`stat.fun.name` character. The name of the network statistic function

`stat.fun.args` list. Additional arguments the network statistic function took

`orig.net.name` character. The name of the original network.

`orig.net.stat` numeric. The value of the network statistic calculated on the original network.

`nets.stat` list. A list of values of the network statistic applied to the manipulated networks

`nets.names` character. Names of the manipulated networks.

`sample.variables` list. A list of sample level variables.

NetSet-class *An S4 class representing a single network with a network permutation function applied to it.*

Description

This class represents the results of applying a network permutation function, such as a jackknife, or rewiring algorithm, to a single network.

Details

For constructor see: For constructor see: net_apply

Slots

`fun.name` character. The name of the network permutation function applied

`fun` function. The permutation function applied

`fun.args` list. The arguments supplied to the permutation function

`orig.net` Net. The original network

`orig.net.name` character. The name of the original network

`nets` list. A list of Net objects, each corresponding to a instance of the manipulated original network

`nets.names` character. The names of the manipulated networks.

`node.variables` list. Node variables of the original network

`iter` logical. A flag to indicate that the permutation function was repeated with the same arguements. Currently unused.

NetStatSet-class *An S4 class representing the results of applying a network statistic function to a single NetSet object.*

Description

This class represents the results of applying a network statistic function to a single NetSet object. This class contains the results for the original network, as well as for each instance of the permuted/manipulated networks.

Slots

`stat.fun` function. The network statistic function applied

`stat.fun.name` character. The name of the network statistic function

`stat.fun.args` list. Additional arguments the network statistic function took

`orig.net.name` character. The name of the original network.

`orig.net.stat` numeric. The value of the network statistic calculated on the original network.

`nets.stat` list. A list of values of the network statistic applied to the manipulated networks

`nets.names` character. Names of the manipulated networks.

`network_functions` *Network Manipulation Functions*

Description

These functions take a `Net` object, manipulate the network in some way, and return a `list` of modified `Net` objects.

Usage

```
node_jackknife(Net)

network_jackknife(Net, network.variable)

absolute_threshold(Net, thresholds)

relative_threshold(Net, percentiles)
```

Arguments

`Net`	Network to jackknife
`network.variable`	Character name of node variable containing network labels
`thresholds`	Vector of thresholds to use
`percentiles`	Vector of densities to threshold at

Value

A `list` of `Net` objects

Examples

```
data(GroupA)
GroupA1_Net = as_Net(GroupA[[1]], "1", list(community = c(rep(1, 10), rep(2,10))))
node_jackknife(GroupA1_Net)
network_jackknife(GroupA1_Net, "community")
```

network_statistics *Network Statistic Functions*

Description

These functions compute a variety of network statistics on single Net objects.

Usage

```
global_efficiency(Net)

modularity(Net, community.variable)
```

Arguments

Net	Input Net object
community.variable	character name of the node variable that represents the partition.

Value

Network statistic value

References

There are no references for Rd macro \insertAllCites on this help page.

Examples

```
data(GroupA)
GroupA1_Net = as_Net(GroupA[[1]], "1", list(community = c(rep(1, 10), rep(2,10))))
global_efficiency(GroupA1_Net)
modularity(GroupA1_Net, "community")
```

net_apply *Apply a network manipulation function to a single network, or to a sample of networks*

Description

This function applies a network manipulation function to a single network or sample of networks, and returns a NetSet, or NetSampleSet containing the results.

Usage

```
net_apply(network, net.function, net.function.args, orig.net.name)

## S4 method for signature 'Net,ANY,ANY,ANY'
net_apply(network, net.function,
  net.function.args, orig.net.name)

## S4 method for signature 'NetSample,ANY,ANY,missing'
net_apply(network, net.function,
  net.function.args, orig.net.name)
```

Arguments

network	An Net object or a NetSample object
net.function	A network manipulation function (reference or character)
net.function.args	A labeled list containing arguments to the net.function
orig.net.name	The original network name, when applying net_apply to a Net

Value

A NetSet or NetSampleSet object

Methods (by class)

- network = Net,net.function = ANY,net.function.args = ANY,orig.net.name = ANY: net_apply for Net
- network = NetSample,net.function = ANY,net.function.args = ANY,orig.net.name = missing: net_apply for NetSample

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
 sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
```

---

net_stat_apply *Apply a network statistic function to a* NetSet *or* NetSampleSet *object.*

Description

This function applies a network statistic function to a NetSet or NetSampleSet object, and returns the calculated network statistics.

Usage

```
net_stat_apply(netSet, net.stat.fun, net.stat.fun.args, net.stat.name)

## S4 method for signature 'NetSet'
net_stat_apply(netSet, net.stat.fun, net.stat.fun.args,
  net.stat.name)

## S4 method for signature 'NetSampleSet'
net_stat_apply(netSet, net.stat.fun,
  net.stat.fun.args, net.stat.name)
```

Arguments

netSet	A NetSet or NetSampleSet object.
net.stat.fun	The network statistic function
net.stat.fun.args	A list of additional arguments to the network statistic function
net.stat.name	A descriptive name for the network statistic (defaults to deparsed name of statistic function)

Value

A NetStatSet or NetSampleStatSet

Methods (by class)

- NetSet: net_stat_apply for NetSet
- NetSampleSet: Converter for NetSampleSet

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
 sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
```

---

show,Net-method *Convenience Functions for printing information about Net-type objects*

Description

Convenience Functions for printing information about Net-type objects

Usage

```
## S4 method for signature 'Net'
show(object)

## S4 method for signature 'NetSample'
show(object)

## S4 method for signature 'NetSet'
show(object)

## S4 method for signature 'NetSampleSet'
show(object)

## S4 method for signature 'NetSet'
names(x)

## S4 method for signature 'NetStatSet'
show(object)

## S4 method for signature 'NetSampleStatSet'
show(object)
```

Arguments

object, x	Object to print

---

to_data_frame *Network statistics to long format dataframe*

Description

This function converts a NetStatSet or NetSampleStatSet into a long format dataframe

Usage

```
to_data_frame(netStatSet)

## S4 method for signature 'NetStatSet'
to_data_frame(netStatSet)

## S4 method for signature 'NetSampleStatSet'
to_data_frame(netStatSet)
```

Arguments

netStatSet	A NetStatSet or NetSampleStatSet object

Value

A long format dataframe containing the name of the original network, the original network network statistic, the name of the manipulated network, the manipulated network network statistic and the name of the network statistic.

Methods (by class)

- `NetStatSet`: Converter for NetSampleStatSet
- `NetSampleStatSet`: Converter for NetSampleStatSet

Examples

```
data(GroupA)
GroupA_Net = as_NetSample(GroupA, 1:20, node.variables = list(community = c(rep(1, 10), rep(2,10))),
  sample.variables = list(group = c(rep(1, 10), rep(2,10))))
Jackknife_GroupA_Net = net_apply(GroupA_Net, node_jackknife)
GlobEff_GroupA_Net = net_stat_apply(Jackknife_GroupA_Net, global_efficiency)
head(to_data_frame(GlobEff_GroupA_Net))
```

Index