

Package ‘brainGraph’

May 29, 2018

Type Package

Version 2.2.0

Date 2018-05-28

Title Graph Theory Analysis of Brain MRI Data

Author Christopher G. Watson <cgwatson@bu.edu>

Maintainer Christopher G. Watson <cgwatson@bu.edu>

Description A set of tools for performing graph theory analysis of brain MRI data. It works with data from a Freesurfer analysis (cortical thickness, volumes, local gyrification index, surface area), diffusion tensor tractography data (e.g., from FSL) and resting-state fMRI data (e.g., from DPABI). It contains a graphical user interface for graph visualization and data exploration, along with several functions for generating useful figures.

URL <https://github.com/cwatson/brainGraph>

BugReports <https://groups.google.com/forum/?hl=en#!forum/brainGraph-help>

LazyData true

Depends R (>= 3.0.0), igraph (>= 1.0.0),

Imports abind, ade4, boot, data.table, expm, foreach, ggplot2, ggrepel, gridExtra, Hmisc, MASS, Matrix, mediation, methods, oro.nifti, permute, parallel, RcppEigen, scales

Suggests RGtk2, cairoDevice

License GPL-3

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-29 04:59:32 UTC

R topics documented:

| | |
|--|----|
| aal116 | 3 |
| aal2.120 | 4 |
| aal2.94 | 5 |
| aal90 | 6 |
| analysis_random_graphs | 7 |
| apply_thresholds | 8 |
| bg_to_mediate | 9 |
| brainGraph_boot | 10 |
| brainGraph_GLM | 11 |
| brainGraph_GLM_design | 13 |
| brainGraph_init | 14 |
| brainGraph_mediate | 16 |
| brainGraph_permute | 19 |
| brainsuite | 20 |
| centr_betw_comm | 21 |
| centr_lev | 22 |
| coeff_var | 23 |
| communicability | 23 |
| contract_brainGraph | 24 |
| cor.diff.test | 25 |
| corr.matrix | 26 |
| CountEdges | 28 |
| craddock200 | 29 |
| create_mats | 30 |
| DataTables | 32 |
| destrieux | 33 |
| destrieux.scgm | 34 |
| dk | 35 |
| dk.scgm | 36 |
| dkt | 37 |
| dkt.scgm | 38 |
| dosenbach160 | 39 |
| edge_asymmetry | 40 |
| efficiency | 41 |
| get.resid | 42 |
| GLMfit | 44 |
| GraphDistances | 45 |
| hoa112 | 46 |
| IndividualContributions | 46 |
| lpba40 | 48 |
| make_brainGraph | 49 |
| make_ego_brainGraph | 50 |
| make_empty_brainGraph | 51 |
| make_glm_brainGraph | 52 |
| make_intersection_brainGraph | 53 |
| make_mediate_brainGraph | 53 |

| | |
|--------------------------------------|----|
| make_nbs_brainGraph | 54 |
| mtpc | 55 |
| NBS | 57 |
| plot.bg_GLM | 59 |
| plot.brainGraph | 60 |
| plot.brainGraph_boot | 61 |
| plot.brainGraph_GLM | 62 |
| plot.brainGraph_mediate | 63 |
| plot.brainGraph_mtpc | 64 |
| plot.brainGraph_NBS | 64 |
| plot.brainGraph_permute | 65 |
| plot.brainGraph_resids | 66 |
| plot.mtpc | 67 |
| plot_brainGraph_gui | 68 |
| plot_brainGraph_list | 69 |
| plot_brainGraph_multi | 70 |
| plot_corr_mat | 71 |
| plot_global | 72 |
| plot_rich_norm | 73 |
| plot_vertex_measures | 74 |
| plot_volumetric | 75 |
| RandomGraphs | 76 |
| RichClub | 77 |
| rich_club_attrs | 79 |
| robustness | 80 |
| set_brainGraph_attr | 81 |
| small.world | 83 |
| summary.bg_GLM | 84 |
| summary.bg_mediate | 84 |
| summary.brainGraph | 85 |
| summary.brainGraph_permute | 85 |
| symmetrize_mats | 86 |
| s_core | 87 |
| VertexRoles | 88 |
| vulnerability | 89 |
| write_brainnet | 90 |
| xfm.weights | 91 |

Index**93**

aal116

*Coordinates for data from the AAL116 atlas***Description**

This is a list of spatial coordinates for the AAL116 atlas, along with indices for the major lobes of the brain.

Usage

```
data("aal116")
```

Format

A data frame with 116 observations on the following 7 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Limbic SCGM Cerebellum

hemi a factor with levels L R

index a numeric vector

References

Tzourio-Mazoyer N., Landeau B., Papathanassiou D., Crivello F., Etard O., Delcroix N., Mazoyer B., Joliot M. (2002) *Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain*. NeuroImage, 15(1):273-289.

Examples

```
data(aal116)
str(aal116)
```

 aal2.120

Coordinates for data from the AAL2 atlas

Description

This is a list of spatial coordinates for the AAL2 atlas, along with indices for the major lobes of the brain.

Usage

```
data("aal2.120")
```

Format

A data frame with 120 observations on the following 7 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)
 lobe a factor with levels Frontal Parietal Temporal Occipital Central Limbic SCGM Cerebellum
 hemi a factor with levels L R
 index a numeric vector

References

Rolls E.T., Joliot M., Tzourio-Mazoyer N. (2015) *Implementation of a new parcellation of the orbitofrontal cortex in the automated anatomical labelling atlas*. *NeuroImage*, 122:1-5. Tzourio-Mazoyer N., Landeau B., Papathanassiou D., Crivello F., Etard O., Delcroix N., Mazoyer B., Joliot M. (2002) *Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain*. *NeuroImage*, 15(1):273-289.

Examples

```
data(aal2.120)
str(aal2.120)
```

 aal2.94

Coordinates for data from the AAL2 atlas

Description

This is a list of spatial coordinates for the AAL2 atlas, along with indices for the major lobes of the brain.

Usage

```
data("aal2.94")
```

Format

A data frame with 94 observations on the following 7 variables.

name a character vector of region names
 x.mni a numeric vector of x-coordinates (in MNI space)
 y.mni a numeric vector of y-coordinates (in MNI space)
 z.mni a numeric vector of z-coordinates (in MNI space)
 lobe a factor with levels Frontal Parietal Temporal Occipital Central Limbic SCGM
 hemi a factor with levels L R
 index a numeric vector

References

Rolls E.T., Joliot M., Tzourio-Mazoyer N. (2015) *Implementation of a new parcellation of the orbitofrontal cortex in the automated anatomical labelling atlas*. *NeuroImage*, 122:1-5. Tzourio-Mazoyer N., Landeau B., Papathanassiou D., Crivello F., Etard O., Delcroix N., Mazoyer B., Joliot M. (2002) *Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain*. *NeuroImage*, 15(1):273-289.

Examples

```
data(aal2.94)
str(aal2.94)
```

| | |
|-------|--|
| aal90 | <i>Coordinates for data from the AAL90 atlas</i> |
|-------|--|

Description

This is a list of spatial coordinates for the AAL90 atlas, along with indices for the major lobes of the brain.

Usage

```
data("aal90")
```

Format

A data frame with 90 observations on the following 7 variables.

name a character vector of region names
 x.mni a numeric vector of x-coordinates (in MNI space)
 y.mni a numeric vector of y-coordinates (in MNI space)
 z.mni a numeric vector of z-coordinates (in MNI space)
 lobe a factor with levels Frontal Parietal Temporal Occipital Insula Limbic SCGM
 hemi a factor with levels L R
 index a numeric vector

References

Tzourio-Mazoyer N., Landeau B., Papathanassiou D., Crivello F., Etard O., Delcroix N., Mazoyer B., Joliot M. (2002) *Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain*. *NeuroImage*, 15(1):273-289.

Examples

```
data(aal90)
str(aal90)
```

`analysis_random_graphs`*Perform an analysis with random graphs for brain MRI data*

Description

This function is not quite a "proper" function. It performs the steps needed for doing typical graph theory analyses with brain MRI data if you need to generate equivalent random graphs. This includes calculating *small world* parameters and normalized *rich club* coefficients.

Usage

```
analysis_random_graphs(g.list, N = 100, savedir = ".", ...)
```

Arguments

| | |
|----------------------|--|
| <code>g.list</code> | List of lists containing igraph graph objects |
| <code>N</code> | Integer specifying number of random graphs to generate per individual graph (default: 100) |
| <code>savedir</code> | Character string specifying the directory in which to save the generated graphs (default: current working directory) |
| <code>...</code> | Other arguments passed to <code>sim.rand.graph.par</code> (e.g. <code>clustering=T</code>) |

Details

The steps that are performed are:

1. `N` random graphs are generated for each group and density/threshold (and subject if you have subject-specific graphs).
2. These graphs are all written to disk in `savedir`. All of these are read back into R and combined into lists; these lists are also written to disk (in a sub-directory named ALL), so you can delete the individual `.rds` files afterwards.
3. *Small world* parameters are calculated, along with values for a few global graph measures that may be of interest.
4. *Normalized rich club coefficients* and associated p-values will be calculated.

Value

A list containing:

| | |
|--------------------|--|
| <code>rich</code> | A data table containing normalized rich-club coefficients and p-values |
| <code>small</code> | A data table with small-world parameters |
| <code>rand</code> | A data table with some global graph measures for all random graphs generated |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[small.world](#)

Other Random graph functions: [RandomGraphs](#), [RichClub](#)

Examples

```
## Not run:
rand_all <- analysis_random_graphs(g.norm, 1e2,
  savedir='/home/cwatson/dti/rand', clustering=F)

## End(Not run)
```

apply_thresholds

Threshold additional set of matrices

Description

apply_thresholds will threshold an additional set of matrices (e.g., FA-weighted matrices for DTI tractography) based on the matrices that have been returned from [create_mats](#). This ensures that the same connections are present in both sets of matrices.

Usage

```
apply_thresholds(sub.mats, group.mats, W.files, inds)
```

Arguments

| | |
|------------|---|
| sub.mats | List (length equal to number of thresholds) of numeric arrays (3-dim) for all subjects |
| group.mats | List (equal to number of thresholds) of lists (equal to number of groups) of numeric matrices for group-level data |
| W.files | Character vector of the filenames of the files containing your connectivity matrices |
| inds | List (length equal to number of groups) of integers; each list element should be an integer vector of length equal to the group sizes |

Value

List containing:

| | |
|-------------|---|
| W | A 3-d array of the raw connection matrices |
| W.norm.sub | List of 3-d arrays of the normalized connection matrices for all given thresholds |
| W.norm.mean | List of lists of numeric matrices averaged for each group |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Matrix functions: [cor.diff.test](#), [create_mats](#), [symmetrize_mats](#)

Examples

```
## Not run:  
W.mats <- apply_thresholds(A.norm.sub, A.norm.mean, f.W, inds)  
  
## End(Not run)
```

| | |
|---------------|---|
| bg_to_mediate | <i>Convert brainGraph results to "mediate" object</i> |
|---------------|---|

Description

[bg_to_mediate](#) converts the results of [brainGraph_mediate](#) into an object of class [mediate](#). In [brainGraph](#), it is only used for the [summary.mediate](#) method, but you can similarly use its output for the [plot.mediate](#) method.

Usage

```
bg_to_mediate(x, region = NULL)
```

Arguments

| | |
|--------|--|
| x | Object output from brainGraph_mediate |
| region | Character string specifying which region's results to summarize; only relevant if level='vertex' (default: NULL) |

Value

An object of class [mediate](#)

See Also

[mediate](#)

brainGraph_boot *Bootstrapping for global graph measures*

Description

Performs bootstrapping to get group standard error estimates of a global graph measure (e.g. modularity).

Print a summary from a bootstrap analysis

Usage

```
brainGraph_boot(densities, resids, R = 1000, measure = c("mod", "E.global",
  "Cp", "Lp", "assortativity", "strength", "mod.wt", "E.global.wt"),
  conf = 0.95, .progress = TRUE)
```

```
## S3 method for class 'brainGraph_boot'
summary(object, ...)
```

Arguments

| | |
|-----------|--|
| densities | Numeric vector of graph densities to loop through |
| resids | An object of class brainGraph_resids (the output from get.resid) |
| R | Integer; the number of bootstrap replicates (default: 1e3) |
| measure | Character string of the measure to test (default: mod) |
| conf | Numeric; the confidence level for calculating confidence intervals (default: 0.95) |
| .progress | Logical indicating whether or not to show a progress bar (default: TRUE) |
| object | A brainGraph_boot object (from brainGraph_boot) |
| ... | Unused |

Details

The confidence intervals are calculated using the normal approximation and at the 95% level.

Value

An object of class brainGraph_boot containing some input variables, in addition to a list of [boot](#) objects (one for each group).

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also[boot](#)Other Group analysis functions: [IndividualContributions](#), [NBS](#), [brainGraph_GLM](#), [brainGraph_mediate](#), [brainGraph_permute](#), [mtpc](#)Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_init](#), [brainGraph_permute](#), [corr.matrix](#), [get.resid](#), [plot.brainGraph_resids](#), [plot_volumetric](#)**Examples**

```
## Not run:
boot.E.global <- brainGraph_boot(densities, resid.all, 1e3, 'E.global')

## End(Not run)
```

| | |
|----------------|--|
| brainGraph_GLM | <i>Fit linear models at each vertex of a graph</i> |
|----------------|--|

Description

brainGraph_GLM specifies and fits a linear model at each vertex for a given vertex measure (e.g. *degree*) or at the graph-level (e.g., *global efficiency*). Given a contrast matrix, it will calculate the associated statistics.

Usage

```
brainGraph_GLM(g.list, covars, measure, con.mat, con.type = c("t", "f"),
  X = NULL, con.name = NULL, alternative = c("two.sided", "less",
  "greater"), alpha = 0.05, level = c("vertex", "graph"), permute = FALSE,
  N = 5000, perms = NULL, long = FALSE, ...)
```

Arguments

| | |
|-------------|--|
| g.list | A list of igraph graph objects for all subjects |
| covars | A data.table of covariates |
| measure | Character string of the graph measure of interest |
| con.mat | Numeric matrix specifying the contrast(s) of interest; if only one contrast is desired, you can supply a vector |
| con.type | Character string; either 't' or 'f' (for t or F-statistics). Default: 't' |
| X | Numeric matrix, if you wish to supply your own design matrix (default: NULL) |
| con.name | Character vector of the contrast name(s); if con.mat has row names, those will be used for reporting results (default: NULL) |
| alternative | Character string, whether to do a two- or one-sided test (default: 'two.sided') |
| alpha | Numeric; the significance level (default: 0.05) |
| level | Character string; either vertex (default) or graph |

| | |
|---------|--|
| permute | Logical indicating whether or not to permute group labels (default: FALSE) |
| N | Integer; number of permutations to create (default: 5e3) |
| perms | Matrix of permutations, if you would like to provide your own (default: NULL) |
| long | Logical indicating whether or not to return all permutation results (default: FALSE) |
| ... | Other arguments passed to <code>brainGraph_GLM_design</code> |

Details

The input list of graphs `g.list` must not be nested; i.e., if you have multiple groups, they will have to be combined into one list. See the code in the Examples below.

A `data.table` of covariates is required input; the first column *must* be named `Study.ID`. Additionally, all graphs must have a `name` attribute (at the graph level) which matches the `Study.ID` for a given subject. If you create the design matrix `X` yourself, you still must supply the covariates table so that subjects can be correctly matched with their network data.

Both t- and F-contrasts are allowed. You may supply a *matrix* to the argument `con.mat`. If you supply a multi-row matrix and you choose `con.type="t"`, then statistics will be calculated for each contrast individually. If you choose `con.type="f"`, in the result data table, ESS stands for "extra sum of squares", the additional variance explained for by the model parameters of interest (as determined by the contrast matrix). Finally, the standard error in these tables is the sum of squared errors of the *full model*.

Finally, you can calculate permutations of the data to build a null distribution of the maximum statistic, to provide control over false positives. The permutation strategy is that of Freedman & Lane (1983), and is the same as that in FSL's *randomise*.

Value

An object of class `bg_GLM` containing some input-specific variables, in addition to:

| | |
|----------------------|---|
| <code>X</code> | A numeric matrix; a copy of the <i>design matrix</i> |
| <code>y</code> | A numeric vector or matrix of the outcome variable |
| <code>DT</code> | A data table with an entry for each vertex (region) containing statistics of interest |
| <code>removed</code> | A character vector of <code>Study.ID</code> 's removed due to incomplete data (if any) |
| <code>perm</code> | A list containing: <i>null.dist</i> (the null distribution of maximum statistics), <i>thresh</i> (the statistic value corresponding to the $100 \times (1 - \alpha)$ th% percentile of the null distribution) |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

- Freedman D & Lane D (1983). *A nonstochastic interpretation of reported significance levels*. J Bus Econ Stat, 1(4):292-298.
- Nichols TE & Holmes AP (2001). *Nonparametric permutation tests for functional neuroimaging: A primer with examples*. Human Brain Mapping, 15(1):1-25.
- Winkler AM, Ridgway GR, Webster MA, Smith SM, Nichols TE (2014). *Permutation inference for the general linear model*. NeuroImage, 92:381-397.

See Also

Other GLM functions: [GLMfit](#), [brainGraph_GLM_design](#), [mtpc](#)

Other Group analysis functions: [IndividualContributions](#), [NBS](#), [brainGraph_boot](#), [brainGraph_mediate](#), [brainGraph_permute](#), [mtpc](#)

Examples

```
## Not run:
conmat <- matrix(c(0, 0, 0, 1), nrow=1)
rownames(conmat) <- 'Control > Patient'

## Note that I concatenate the graphs from each group's 6th threshold
g.lm <- brainGraph_GLM(g.list=do.call(Map, c(c, g))[[6]],
  covars=covars.all[tract == 1],
  measure='strength', con.mat=conmat, alt='greater',
  permute=TRUE, long=TRUE)

## End(Not run)
```

brainGraph_GLM_design *Create a design matrix for linear model analysis*

Description

brainGraph_GLM_design takes a data.table of covariates and returns a *design matrix* to be used in linear model analysis.

Usage

```
brainGraph_GLM_design(covars, coding = c("dummy", "effects", "cell.means"),
  factorize = TRUE, mean.center = FALSE, binarize = NULL, int = NULL)
```

Arguments

| | |
|-----------|---|
| covars | A data.table of covariates |
| coding | Character string indicating how factor variables will be coded (default: 'dummy') |
| factorize | Logical indicating whether to convert <i>character</i> columns into <i>factor</i> (default: TRUE) |

| | |
|-------------|--|
| mean.center | Logical indicating whether to mean center non-factor variables (default: FALSE) |
| binarize | Character vector specifying the column name(s) of the covariate(s) to be converted from type factor to numeric (default: NULL) |
| int | Character vector specifying the column name(s) of the covariate(s) to test for an interaction (default: NULL) |

Details

There are three different ways to code factors: *dummy*, *effects*, or *cell-means* (chosen by the argument coding). To understand the difference, see Chapter 8 of the User Guide.

Importantly, the default behavior (as of v2.1.0) is to convert all character columns (excluding the Study ID column and any that you list in the `binarize` argument) to factor variables. To change this, set `factorize=FALSE`. So, if your covariates include multiple character columns, but you want to convert *Scanner* to binary instead of a factor, you may still specify `binarize='Scanner'` and get the expected result. `binarize` will convert the given factor variable(s) into numeric variable(s), which is performed *before* mean-centering.

The argument `mean.center` will mean-center (i.e., subtract the mean of the entire dataset from each variable) any non-factor variables (including any dummy/indicator covariates). This is done *after* "factorizing" and "binarizing".

`int` specifies which variables should interact with one another. This argument accepts both numeric (e.g., *Age*) and factor variables (e.g., *Sex*). All interaction combinations will be generated: if you supply 3 variables, all two-way and the single three-way interaction will be generated. This variable *must* have at least two elements.

Value

A numeric matrix

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other GLM functions: [GLMfit](#), [brainGraph_GLM](#), [mtpc](#)

brainGraph_init

Initialize variables for further use in brainGraph

Description

Initializes some variables that are important for further analysis of structural covariance networks (e.g., *cortical thickness*). This mostly involves loading CSV files (of covariates/demographics, cortical thickness/volumes, etc.) and returning them as data tables.

Usage

```
brainGraph_init(atlas, densities, datadir, modality = "thickness",
  covars = NULL, exclude.subs = NULL, custom.atlas = NULL)
```

Arguments

| | |
|--------------|---|
| atlas | Character string indicating which brain atlas you are using. This can be an atlas present with the package or a custom atlas; in this case you must specify custom here and assign the name to the argument custom.atlas. |
| densities | Numeric vector of the graph densities you would like to investigate |
| datadir | Character string; the filesystem location of your input files |
| modality | Character string indicating the volumetric MRI modality/measure used to create the graphs (default: thickness) |
| covars | A data.table of covariates; specify this if you do not want to load your full covariates file (default: NULL) |
| exclude.subs | Character vector of the Study ID's of subjects who are to be excluded from the analysis (default: NULL) |
| custom.atlas | Character string of the name of the custom atlas you wish to use, if applicable (default: NULL) |

Details

You can use any atlas that is already present in the package; to check the available atlases, you can type `data(package="brainGraph")$results[, 3]` at the R prompt. If you have a custom atlas, specify `atlas="custom"` and supply the R object's name for the argument `custom.atlas`.

The file containing covariates should be named `covars.csv`. However, you may also supply a `data.table` using the function argument `covars`. This is useful if you have multiple covariates in your file and wish to subset the data on your own. It must have, at minimum, columns named `Study.ID` and `Group` (even if you have only 1 group).

The filenames of the structural MRI data should include hemisphere, atlas, and modality separated by the *underscore* character; e.g. `lh_dkt_thickness.csv` contains cortical thickness of left hemisphere regions of the DKT atlas. If you would like to include subcortical gray matter, then you will need files `covars.scgm.csv` and `scgm.csv`.

Value

A list containing:

| | |
|---------------|--|
| atlas | Character string of the brain atlas name |
| densities | Numeric vector of the graph densities |
| modality | Character string of the modality you chose |
| kNumDensities | Integer indicating the number of densities |
| covars | A data.table of covariates |
| groups | Character vector of subject group names |
| kNumGroups | Integer indicating the number of groups |

kNumVertices Integer; the number of vertices in the graphs
 lhrh A data . table of left- and right-hemispheric volumetric data

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_permute](#), [corr.matrix](#), [get.resid](#), [plot.brainGraph_resids](#), [plot_volumetric](#)

Examples

```
## Not run:
init.vars <- brainGraph_init(atlas='dkt', densities=seq(0.07, 0.50, 0.01),
  datadir='/home/cwatson/Data', modality='thickness', exclude.subs=c('Con07',
  'Con23', 'Pat15'))

## End(Not run)
```

brainGraph_mediate *Mediation analysis with brain graph measures as mediator variables*

Description

brainGraph_mediate performs simple mediation analyses in which a given graph- or vertex-level measure (e.g., *weighted global efficiency*) is the mediator M . The outcome (or dependent/response) variable Y can be a neuropsychological measure (e.g., IQ) or can be a disease-specific metric (e.g., recovery time). The treatment variable should be a factor.

Usage

```
brainGraph_mediate(g.list, covars, mediator, treat, outcome, covar.names,
  level = c("graph", "vertex"), boot = TRUE, boot.ci.type = c("perc",
  "bca"), N = 1000, conf.level = 0.95, control.value = 0,
  treat.value = 1, long = TRUE, int = FALSE, ...)
```

Arguments

g.list A list of igraph graph objects for all subjects
 covars A data table containing covariates of interest. It must include columns for *Study.ID*, the treatment variable, covar . names, and the outcome variable.
 mediator Character string; the name of the graph measure acting as the *mediating* variable
 treat Character string; the *treatment* variable (e.g., *Group*)
 outcome Character string; the name of the outcome variable of interest (e.g., full-scale IQ, memory, etc.)

| | |
|---------------|--|
| covar.names | Character vector of the column names in covars to include in the models as pre-treatment covariates. |
| level | Character string; either vertex (default) or graph |
| boot | Logical indicating whether or not to perform bootstrapping (default: TRUE) |
| boot.ci.type | Character string; which type of CI's to calculate (default: perc) |
| N | Integer; the number of bootstrap samples to run (default: 1e3) |
| conf.level | Numeric; the level of the CI's to calculate (default: 0.95 for the 2.5 and 97.5 percentiles) |
| control.value | Value of treat to be used as the control condition (default: 0) |
| treat.value | Value of treat to be used as the treatment condition (default: 1) |
| long | Logical indicating whether or not to return all bootstrap samples (default: TRUE) |
| int | Logical indicating whether or not to include an interaction of the mediator and treatment (default: FALSE) |
| ... | Other arguments passed to brainGraph_GLM_design (e.g., binarize) |

Details

This code was adapted closely from [mediate](#) in the mediation package, and the procedure is exactly the same as theirs (see the references listed below). So, if you use this function, please cite their work.

As of brainGraph v2.0.0, this function has been tested only for a treatment (independent) variable X being a 2-level factor (e.g., disease group, old vs. young, etc.).

Allowing for treatment-mediator interaction (setting `int=TRUE`) currently will only work properly if the mediator is a continuous variable; since the mediator is always a graph metric, this should always be the case.

Value

An object of class `bg_mediate` with elements:

| | |
|--------------|--|
| level | Either graph or vertex. |
| removed | A character vector of Study.ID's removed due to incomplete data |
| X.m, X.y | Design matrices for the model with the mediator as the outcome variable (X.m) and for the model with the mediator as an additional predictor (X.y) |
| y.m, y.y | Outcome variables for the associated design matrices above. <code>y.m</code> will be a matrix of size $\# \text{subj.} \times \# \text{regions}$ |
| res.obs | A data.table of the observed values of the point estimates. |
| res.ci | A data.table of the confidence intervals for the effect estimates. |
| res.p | A data.table of the two-sided p-values for the effect estimates |
| boot | Logical, the boot argument. |
| boot.ci.type | Character string indicating which type of bootstrap confidence intervals were calculated. |
| res.boot | A data.table with N rows of the bootstrap results for all effects. |

| | |
|---------------|---|
| treat | Character string of the treatment variable. |
| mediator | Character string of the mediator variable. |
| outcome | Character string of the outcome variable. |
| covariates | Returns NULL; not used in this package. |
| INT | Logical indicating whether the models included an interaction between treatment and mediator. |
| conf.level | The confidence level. |
| control.value | The value of the treatment variable used as the control condition. |
| treat.value | The value of the treatment variable used as the treatment condition. |
| nobs | Integer; the number of observations in the models. |
| sims | Integer; the number of bootstrap replications. |
| covar.names | The pre-treatment covariate names. |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

- Tingley D, Yamamoto T, Hirose K, Keele L, Imai K (2014). *mediation: R package for causal mediation analysis*. Journal of Statistical Software, 59(5):1-38.
- Imai K, Keele L, Yamamoto T (2010). *Identification, inference, and sensitivity analysis for causal mediation effects*. Statistical Science, 25(1):51-71.
- Imai K, Keele L, Tingley D (2010). *A general approach to causal mediation analysis*. Psychological Methods, 15(4):309-334.
- Imai K, Keele L, Tingley D, Yamamoto T (2011). *Unpacking the black box of causality: learning about causal mechanisms from experimental and observational studies*. American Political Science Review, 105(4):765-789.
- Imai K, Yamamoto T (2013). *Identification and sensitivity analysis for multiple causal mechanisms: revisiting evidence from framing experiments*. Political Analysis, 21(2):141-171.

See Also

Other Group analysis functions: [IndividualContributions](#), [NBS](#), [brainGraph_GLM](#), [brainGraph_boot](#), [brainGraph_permute](#), [mtpc](#)

Examples

```
## Not run:
med.EglobWt.FSIQ <- brainGraph_mediate(dt.G[threshold == thresholds[5]],
  covars.med, 'E.global.wt', 'Group', 'FSIQ', covar.names=c('age', 'gender'),
  boot=TRUE, N=1e4)
med.strength.FSIQ <-
  brainGraph_mediate(dt.V[threshold == thresholds[5] & region == 'lcACC'],
    covars.med, 'strength', 'Group', 'FSIQ',
    covar.names=c('age', 'gender'), N=1e3)
```

```
## End(Not run)
```

```
brainGraph_permute      Permutation test for group difference of graph measures
```

Description

brainGraph_permute draws permutations from linear model residuals to determine the significance of between-group differences of a global or vertex-wise graph measure. It is intended for structural covariance networks (in which there is only one graph per group), but can be extended to other types of data.

Usage

```
brainGraph_permute(densities, resids, N = 5000, perms = NULL, auc = FALSE,
  level = c("graph", "vertex", "other"), measure = c("btwn.cent", "degree",
  "E.nodal", "ev.cent", "knn", "transitivity", "vulnerability"), atlas = NULL,
  .function = NULL)
```

Arguments

| | |
|-----------|--|
| densities | Numeric vector of graph densities |
| resids | An object of class brainGraph_resids (the output from get.resid) |
| N | Integer; the number of permutations (default: 5e3) |
| perms | Numeric matrix of permutations, if you would like to provide your own (default: NULL) |
| auc | Logical indicating whether or not to calculate differences in the area-under-the-curve of metrics (default: FALSE) |
| level | A character string for the attribute "level" to calculate differences (default: graph) |
| measure | A character string specifying the vertex-level metric to calculate, only used if level='vertex' (default: btwn.cent) |
| atlas | Character string of the atlas name; required if level='graph' (default: NULL) |
| .function | A custom function you can pass if level='other' |

Details

If you would like to calculate differences in the area-under-the-curve (AUC) across densities, then specify auc=TRUE.

There are three possible "levels":

1. *graph* Calculate modularity (Louvain algorithm), clustering coefficient, characteristic path length, degree assortativity, global efficiency, lobe assortativity, and edge asymmetry.
2. *vertex* Choose one of: betweenness centrality, degree, nodal efficiency, k-nearest neighbor degree, transitivity, or vulnerability.

3. *other* Supply your own function. This is useful if you want to calculate something that I haven't hard-coded. It must take as its own arguments: `g` (a list of lists of igraph graph objects); and `densities` (numeric vector).

Value

An object of class `brainGraph_permute` with input arguments in addition to:

| | |
|-----------------------|--|
| <code>DT</code> | A data table with permutation statistics |
| <code>obs.diff</code> | A data table of the observed group differences |
| <code>groups</code> | Group names |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Group analysis functions: [IndividualContributions](#), [NBS](#), [brainGraph_GLM](#), [brainGraph_boot](#), [brainGraph_mediate](#), [mtpc](#)

Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_init](#), [corr.matrix](#), [get.resid](#), [plot.brainGraph_resids](#), [plot_volumetric](#)

Examples

```
## Not run:
myResids <- get.resid(lhrh, covars)
myPerms <- shuffleSet(n=nrow(myResids$resids.all), nset=1e3)
out <- brainGraph_permute(densities, m, perms=myPerms, atlas='dk')
out <- brainGraph_permute(densities, m, perms=myPerms, level='vertex')
out <- brainGraph_permute(densities, m, perms=myPerms,
  level='other', .function=myFun)

## End(Not run)
```

brainsuite

Coordinates for data from BrainSuite atlas

Description

This is a list of spatial coordinates for the BrainSuite software, along with indices for the major lobes of the brain.

Usage

```
data("brainsuite")
```

Format

A data frame with 74 observations on the following 7 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate SCGM

hemi a factor with levels L R

index a numeric vector

Source

Shattuck DW and Leahy RM (2002) *BrainSuite: an automated cortical surface identification tool*. Medical Image Analysis, 8(2):129-142.

References

Pantazis D, Joshi AA, Jintao J, Shattuck DW, Bernstein LE, Damasio H, and Leahy RM. (2009) *Comparison of landmark-based and automatic methods for cortical surface registration*. NeuroImage, 49(3):2479-2493.

Examples

```
data(brainsuite)
str(brainsuite)
```

centr_betw_comm *Calculate communicability betweenness centrality*

Description

centr_betw_comm calculates the *communicability betweenness* of the vertices of a graph. The centrality for vertex r is

$$\omega_r = \frac{1}{C} \sum_p \sum_q \frac{(e^{\mathbf{A}})_{pq} - (e^{\mathbf{A} + \mathbf{E}(r)})_{pq}}{(e^{\mathbf{A}})_{pq}}$$

where $C = (n - 1)^2 - (n - 1)$ is a normalization factor.

Usage

```
centr_betw_comm(g, A = NULL)
```

Arguments

g An igraph graph object
A Numeric matrix, the graph's adjacency matrix (default: NULL)

Value

A numeric vector of the centrality for each vertex

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Estrada E., Higham D.J., Hatano N. (2009) *Communicability betweenness in complex networks*. Physica A, 388:764-774.

See Also

Other Centrality functions: [centr_lev](#)

| | |
|-----------|---|
| centr_lev | <i>Calculate a vertex's leverage centrality</i> |
|-----------|---|

Description

Calculates the leverage centrality of each vertex in a graph.

Usage

```
centr_lev(g)
```

Arguments

`g` An igraph graph object

Details

The leverage centrality relates a vertex's degree with the degree of its neighbors. The equation is:

$$l_i = \frac{1}{k_i} \sum_{j \in N_i} \frac{k_i - k_j}{k_i + k_j}$$

where k_i is the degree of the i^{th} vertex and N_i is the set of neighbors of i . This function replaces `NaN` with `NA` (for functions that have the argument `na.rm`).

This function was adapted from the igraph wiki (<http://igraph.wikidot.com>).

Value

A vector of the leverage centrality for all vertices.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Joyce K.E., Laurienti P.J., Burdette J.H., Hayasaka S. (2010) *A new measure of centrality for brain networks*. PLoS One, 5(8):e12200.

See Also

Other Centrality functions: [centr_betw_comm](#)

| | |
|-----------|---|
| coeff_var | <i>Calculate coefficient of variation</i> |
|-----------|---|

Description

Calculates the *coefficient of variation*, defined as

$$CV(x) = \frac{sd(x)}{mean(x)}$$

Usage

coeff_var(x)

Arguments

x Numeric vector

Value

A numeric value

| | |
|-----------------|----------------------------------|
| communicability | <i>Calculate communicability</i> |
|-----------------|----------------------------------|

Description

communicability calculates the communicability of a network, a measure which takes into account all possible paths (including non-shortest paths) between vertex pairs.

Usage

communicability(g, weights = NULL)

Arguments

| | |
|---------|---|
| g | An igraph graph object |
| weights | Numeric vector of edge weights; if NULL (the default), and if the graph has edge attribute weight, then that will be used. To avoid using weights, this should be NA. |

Details

The communicability G_{pq} is a weighted sum of the number of walks from vertex p to q and is calculated by taking the exponential of the adjacency matrix A :

$$G_{pq} = \sum_{k=0}^{\infty} \frac{(A^k)_{pq}}{k!} = (e^A)_{pq}$$

where k is *walk* length.

For weighted graphs with $D = \text{diag}(d_i)$ a diagonal matrix of vertex strength,

$$G_{pq} = (e^{D^{-1/2}AD^{-1/2}})_{pq}$$

Value

A numeric matrix of the communicability

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

- Estrada E. & Hatano N. (2008) *Communicability in complex networks*. Physical Review E, 77:036111.
 Crofts J.J. & Higham D.J. (2009) *A weighted communicability measure applied to complex brain networks*. J. R. Soc. Interface, 6:411-414.

contract_brainGraph *Contract graph vertices based on brain lobe and hemisphere*

Description

Create a new graph after merging multiple vertices based on *brain lobe* and *hemisphere* membership.

Usage

```
contract_brainGraph(g)
```

Arguments

| | |
|---|------------------------|
| g | An igraph graph object |
|---|------------------------|

Details

The vertex size of the resultant graph is equal to the number of vertices in each lobe (in the input graph). The x- and y- coordinates of the new vertices are equal to the mean coordinates of the lobe vertices of the original graph. The new edge weight is equal to the number of inter-lobular connections of the original graph.

Value

A new igraph graph object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[contract](#)

cor.diff.test

Calculate the p-value for differences in correlation coefficients

Description

Given two sets of correlation coefficients and sample sizes, this function calculates and returns the *z-scores* and *p-values* associated with the difference between correlation coefficients. This function was adapted from <http://stackoverflow.com/a/14519007/3357706>.

Usage

```
cor.diff.test(r1, r2, n1, n2, alternative = c("two.sided", "less", "greater"))
```

Arguments

| | |
|-------------|--|
| r1 | Numeric (vector or matrix) of correlation coefficients, group 1 |
| r2 | Numeric (vector or matrix) of correlation coefficients, group 2 |
| n1 | Integer; number of observations, group 1 |
| n2 | Integer; number of observations, group 2 |
| alternative | Character string specifying the alternative hypothesis test to use; one of: 'two.sided' (default), 'less', 'greater' |

Value

A list containing:

| | |
|---|--|
| p | The p-values |
| z | The z-score for the difference in correlation coefficients |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Matrix functions: [apply_thresholds](#), [create_mats](#), [symmetrize_mats](#)

Examples

```
## Not run:
kNumSubjs <- summary(covars$Group)
corr.diffs <- cor.diff.test(corr[[1]][[1]]$R, corr[[2]][[1]]$R,
                           kNumSubjs[1], kNumSubjs[2], alternative='two.sided')
edge.diffs <- t(sapply(which(corr.diffs$p < .05), function(x)
                      mapply('[[' ,
                              dimnames(corr.diffs$p),
                              arrayInd(x, dim(corr.diffs$p)))
                      )))

## End(Not run)
```

corr.matrix

Calculate correlation matrix and threshold

Description

`corr.matrix` calculates the correlation between all column pairs of a given data frame, and thresholds the resultant correlation matrix based on a given density (e.g., 0.1 if you want to keep only the 10% strongest correlations). If you want to threshold by a specific correlation coefficient (via the `thresholds` argument), then the `densities` argument is ignored.

Usage

```
corr.matrix(resids, densities, thresholds = NULL, what = c("resids", "raw"),
            exclude.reg = NULL, type = c("pearson", "spearman"), rand = FALSE)
```

Arguments

| | |
|--------------------------|---|
| <code>resids</code> | An object of class <code>brainGraph_resids</code> (the output from get.resid) |
| <code>densities</code> | Numeric vector indicating the resultant network density(ies); keeps the top <i>X</i> % of correlations |
| <code>thresholds</code> | Numeric; absolute correlation value to threshold by (default: NULL) |
| <code>what</code> | Character string indicating whether to correlate the residuals or the raw structural MRI values (default: 'resids') |
| <code>exclude.reg</code> | Character vector of regions to exclude (default: NULL) |
| <code>type</code> | Character string indicating which type of correlation coefficient to calculate (default: 'pearson') |

`rand` Logical indicating whether the function is being called for permutation testing; not intended for general use (default: FALSE)

Details

If you wish to exclude regions from your analysis, you can give the indices of their columns with the `exclude.reg` argument.

By default, the Pearson correlation coefficients are calculated, but you can return Spearman by changing the `type` argument.

Value

A nested list containing a list for all subject groups; each of these has the following components:

| | |
|--------------------------|--|
| <code>R</code> | Numeric matrix of correlation coefficients. |
| <code>P</code> | Numeric matrix of p-values. |
| <code>r.thresh</code> | A 3-d binary array indicating correlations that are above a certain threshold. The length of the 3rd dimension equals the number of thresholds/densities supplied. |
| <code>thresholds</code> | Numeric vector; the thresholds supplied. |
| <code>densities</code> | Numeric vector; the densities supplied. |
| <code>what</code> | Residuals or raw values |
| <code>exclude.reg</code> | Excluded regions (if any) |
| <code>type</code> | Pearson or Spearman |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[rcorr](#)

Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_init](#), [brainGraph_permute](#), [get.resid](#), [plot.brainGraph_resids](#), [plot_volumetric](#)

Examples

```
## Not run:
myResids <- get.resid(lhrh, covars)
corrs <- corr.matrix(myResids, densities=densities))

## End(Not run)
```

| | |
|------------|---|
| CountEdges | <i>Count number of edges of a brain graph</i> |
|------------|---|

Description

count_homologous counts the number of edges between homologous regions in a brain graph (e.g. between L and R superior frontal).

count_interlobar counts the number of edges between all vertices in one major lobe (e.g. Frontal) and all other major lobes.

Usage

```
count_homologous(g)
```

```
count_interlobar(g, lobe)
```

Arguments

g An igraph graph object

lobe A character string indicating the lobe to count from (uppercase)

Value

count_homologous - a named vector of the edge ID's connecting homologous regions

count_interlobar - a data.table of total, intra-, and inter-lobar edge counts

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

Examples

```
## Not run:  
g1.frontal <- count_interlobar(g[[1]][[N]], 'Frontal')  
  
## End(Not run)
```

`craddock200`*Coordinates for data from the Craddock200 atlas*

Description

This is a list of spatial coordinates for the Craddock200 atlas, along with indices for the major lobes of the brain.

Usage

```
data("craddock200")
```

Format

A data frame with 200 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate SCGM Cerebellum Brainstem

`hemi` a factor with levels L R Brainstem

`index` a numeric vector

`name.full` a character vector of longer region names

References

Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., Mayberg, H. S. (2012). *A whole brain fMRI atlas generated via spatially constrained spectral clustering*. Human Brain Mapping, 2012, 33, 1914-1928. doi: 10.1002/hbm.21333.

Examples

```
data(craddock200)
str(craddock200)
```

 create_mats

 Create connection matrices for tractography or fMRI data

Description

create_mats will take a vector of filenames which contain connection matrices (e.g. the *fdt_network_matrix* files from FSL or the *ROICorrelation.txt* files from DPABI) and create arrays of this data. You may choose to normalize these matrices by the *waytotal* or *region size* (tractography), or not at all.

Usage

```
create_mats(A.files, modality = c("dti", "fmri"), divisor = c("none",
  "waytotal", "size", "rowSums"), div.files = NULL,
  threshold.by = c("consensus", "density", "mean", "consistency"),
  mat.thresh = 0, sub.thresh = 0.5, inds = list(1:length(A.files)),
  algo = c("probabilistic", "deterministic"), P = 5000, ...)
```

Arguments

| | |
|--------------|--|
| A.files | Character vector of the filenames with connection matrices |
| modality | Character string indicating data modality (default: dti) |
| divisor | Character string indicating how to normalize the connection matrices; either 'none' (default), 'waytotal', 'size', or 'rowSums' (ignored if modality equals fmri) |
| div.files | Character vector of the filenames with the data to normalize by (e.g. a list of <i>waytotal</i> files) (default: NULL) |
| threshold.by | Character string indicating how to threshold the data; choose density, mean, or consistency if you want all resulting matrices to have the same densities (default: consensus) |
| mat.thresh | Numeric (vector) for thresholding connection matrices (default: 0) |
| sub.thresh | Numeric (between 0 and 1) for thresholding by subject numbers (default: 0.5) |
| inds | List (length equal to number of groups) of integers; each list element should be a vector of length equal to the group sizes |
| algo | Character string of the tractography algorithm used (default: 'probabilistic'). Ignored if <i>modality</i> is fmri. |
| P | Integer; number of samples per seed voxel (default: 5000) |
| ... | Arguments passed to symmetrize_mats |

Details

The argument `threshold.by` has 4 options:

1. `consensus` Threshold based on the raw (normalized, if selected) values in the matrices. If this is selected, it uses the `sub.thresh` value to perform "consensus" thresholding.

2. density Threshold the matrices to yield a specific graph density (given by the `mat.thresh` argument).
3. mean Keep only connections for which the cross-subject mean is at least 2 standard deviations higher than the threshold (specified by `mat.thresh`)
4. consistency Threshold based on the coefficient of variation to yield a graph with a specific density (given by `mat.thresh`). The edge weights will still represent those of the input matrices. See Roberts et al. (2017) for more on "consistency-based" thresholding.

The argument `mat.thresh` allows you to choose a numeric threshold, below which the connections will be replaced with 0; this argument will also accept a numeric vector. The argument `sub.thresh` will keep only those connections for which at least $X\%$ of subjects have a positive entry (the default is 0.5, or 50%).

Value

A list containing:

| | |
|--------------------------|--|
| <code>A</code> | A 3-d array of the raw connection matrices |
| <code>A.norm</code> | A 3-d array of the normalized connection matrices |
| <code>A.bin</code> | A 3-d array of binarized connection matrices |
| <code>A.bin.sums</code> | A list of 2-d arrays of connection matrices, with each entry signifying the number of subjects with a connection present; the number of list elements equals the length of <code>mat.thresh</code> |
| <code>A.ind</code> | A list of arrays of binarized connection matrices, containing 1 if that entry is to be included |
| <code>A.norm.sub</code> | List of 3-d arrays of the normalized connection matrices for all given thresholds |
| <code>A.norm.mean</code> | List of lists of numeric matrices averaged for each group |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Roberts JA, Perry A, Roberts G, Mitchell PB, Breakspear M (2017). *Consistency-based thresholding of the human connectome*. *NeuroImage*, 145:118-129.

See Also

Other Matrix functions: [apply_thresholds](#), [cor.diff.test](#), [symmetrize_mats](#)

Examples

```
## Not run:
thresholds <- seq(from=0.001, to=0.01, by=0.001)
fmri.mats <- create_mats(f.A, modality='fmri', threshold.by='consensus',
  mat.thresh=thresholds, sub.thresh=0.5, inds=inds)
dti.mats <- create_mats(f.A, divisor='waytotal', div.files=f.way,
```

```
mat.thresh=thresholds, sub.thresh=0.5, inds=inds)

## End(Not run)
```

DataTables

Create a data table with graph global and vertex measures

Description

`graph_attr_dt` is a helper function that takes a list of graphs and creates a `data.table` of global measures for each graph. Each row will be for a different graph.

`vertex_attr_dt` is a helper function that creates a `data.table` in which each row is a vertex and each column is a different network measure (degree, centrality, etc.). It is partly a wrapper for [as_data_frame](#).

Usage

```
graph_attr_dt(g.list, group = NULL)

vertex_attr_dt(g, group = NULL)
```

Arguments

| | |
|---------------------|--|
| <code>g.list</code> | A list of igraph graph objects |
| <code>group</code> | A character string indicating group membership (default: NULL) |
| <code>g</code> | An igraph graph object |

Value

A `data.table`

See Also

[graph_attr](#), [graph_attr_names](#)
[vertex_attr](#), [vertex_attr_names](#), [as_data_frame](#)

Examples

```
## Not run:
dt.V <- vertex_attr_dt(g)
setcolorder(dt.V, c('modality', 'atlas', 'Group', names(dt.V)[1:28]))

## End(Not run)
```

| | |
|-----------|--|
| destrieux | <i>Coordinates for data from the Destrieux atlas</i> |
|-----------|--|

Description

This is a list of spatial coordinates for the Destrieux atlas, along with indices for the major lobes of the brain.

Usage

```
data("destrieux")
```

Format

A data frame with 148 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Limbic

`hemi` a factor with levels L R

`index` a numeric vector

`class` a factor with levels G G_and_S S

References

Destrieux C., Fischl B., Dale E. & Halgren E. (2010) *Automatic parcellation of human cortical gyri and sulci using standard anatomic nomenclature*. *NeuroImage*, 53(1):1-15.

Examples

```
data(destrieux)
str(destrieux)
```

| | |
|----------------|--|
| destrieux.scgm | <i>Coordinates for data from the Destrieux atlas</i> |
|----------------|--|

Description

This is a list of spatial coordinates for the Destrieux atlas, along with indices for the major lobes of the brain.

Usage

```
data("destrieux.scgm")
```

Format

A data frame with 162 observations on the following 8 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Limbic Cingulate SCGM

hemi a factor with levels L R

index a numeric vector

class a factor with levels G G_and_S S SCGM

References

Destrieux C., Fischl B., Dale E. & Halgren E. (2010) *Automatic parcellation of human cortical gyri and sulci using standard anatomic nomenclature*. *NeuroImage*, 53(1):1-15.

Examples

```
data(destrieux.scgm)
str(destrieux.scgm)
```

dk *Coordinates for data from the Desikan-Killiany atlas*

Description

This is a list of spatial coordinates for the Desikan-Killiany (DK) atlas, along with indices for the major lobes of the brain.

Usage

```
data("dk")
```

Format

A data frame with 68 observations on the following 8 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate

hemi a factor with levels L R

index a numeric vector

name.full a character vector of full region names

References

Desikan R.S., Segonne F., Fischl B., et al. (2006) *An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest*. NeuroImage, 31:968-980.

Examples

```
data(dk)  
str(dk)
```

`dk.scgm`*Coordinates for data from the Desikan-Killiany atlas*

Description

This is a list of spatial coordinates for the Desikan-Killiany (DK) atlas, along with indices for the major lobes of the brain and subcortical gray matter structures.

Usage

```
data("dk.scgm")
```

Format

A data frame with 82 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate

`hemi` a factor with levels L R

`index` a numeric vector

`name.full` a character vector of full region names

References

Desikan R.S., Segonne F., Fischl B., et al. (2006) *An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest*. NeuroImage, 31:968-980.

Examples

```
data(dk.scgm)
str(dk.scgm)
```

`dkt`*Coordinates for data from the Desikan-Killiany-Tourville atlas*

Description

This is a list of spatial coordinates for the Desikan-Killiany-Tourville (DKT) atlas, along with indices for the major lobes of the brain.

Usage

```
data("dkt")
```

Format

A data frame with 62 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate

`hemi` a factor with levels L R

`index` a numeric vector

`name.full` a character vector of full region names

References

Klein A. and Tourville J. (2012) *101 labeled brain images and a consistent human cortical labeling protocol*. Front Neurosci, doi:10.3389/fnins.2012.00171

Examples

```
data(dkt)
str(dkt)
```

`dkt.scgm`*Coordinates for data from the Desikan-Killiany-Tourville atlas*

Description

This is a list of spatial coordinates for the Desikan-Killiany-Tourville (DKT) atlas, along with indices for the major lobes of the brain and subcortical gray matter structures.

Usage

```
data("dkt.scgm")
```

Format

A data frame with 76 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate

`hemi` a factor with levels L R

`index` a numeric vector

`name.full` a character vector of full region names

References

Klein A. and Tourville J. (2012) *101 labeled brain images and a consistent human cortical labeling protocol*. Front Neurosci, doi:10.3389/fnins.2012.00171

Examples

```
data(dkt.scgm)
str(dkt.scgm)
```

`dosenbach160`*Coordinates for data from the Dosenbach160 atlas*

Description

This is a list of spatial coordinates for the Dosenbach160 atlas, along with indices for the major lobes of the brain and functional networks they specify in their manuscript.

Usage

```
data("dosenbach160")
```

Format

A data frame with 160 observations on the following 8 variables.

`name` a character vector of region names

`x.mni` a numeric vector of x-coordinates (in MNI space)

`y.mni` a numeric vector of y-coordinates (in MNI space)

`z.mni` a numeric vector of z-coordinates (in MNI space)

`lobe` a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate SCGM Cerebellum

`hemi` a factor with levels L R B

`index` a numeric vector

`network` a factor with levels default fronto-parietal cingulo-opercular sensorimotor cerebellum occipital

References

Dosenbach, N. U., Nardos, B., Cohen, A. L., Fair, D. A., Power, J. D., Church, J. A., Nelson, S.M., Wig, G.S., Vogel, A.C., Lessov-Schlaggar, C.N., Barnes, K. A. (2010). *Prediction of individual brain maturity using fMRI*. *Science*, 329(5997), 1358-1361.

Examples

```
data(dosenbach160)
str(dosenbach160)
```

| | |
|----------------|---|
| edge_asymmetry | Calculate an asymmetry index based on edge counts |
|----------------|---|

Description

Calculate an *asymmetry index*, a ratio of intra-hemispheric edges in the left to right hemisphere of a graph for brain MRI data.

Usage

```
edge_asymmetry(g, level = c("hemi", "vertex"))
```

Arguments

| | |
|--------------------|--|
| <code>g</code> | An igraph graph object |
| <code>level</code> | Character string indicating whether to calculate asymmetry for each region, or the hemisphere as a whole (default: 'hemi') |

Details

The equation is:

$$A = \frac{E_{lh} - E_{rh}}{0.5 \times (E_{lh} + E_{rh})}$$

where *lh* and *rh* are left and right hemispheres, respectively. The range of this measure is $[-2, 2]$ (although the limits will only be reached if all edges are in one hemisphere), with negative numbers indicating more edges in the right hemisphere, and a value of 0 indicating equal number of edges in each hemisphere.

The `level` argument specifies whether to calculate asymmetry for each vertex, or for the whole hemisphere.

Value

A data table with edge counts for both hemispheres and the asymmetry index; if `level` is *vertex*, the data table will have `vcount(g)` rows.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

efficiency Calculate graph global, local, or nodal efficiency

Description

This function calculates the global efficiency of a graph or the local or nodal efficiency of each vertex of a graph.

Usage

```
efficiency(g, type = c("local", "nodal", "global"), weights = NULL,
  use.parallel = TRUE, A = NULL)
```

Arguments

| | |
|---------------------------|---|
| <code>g</code> | An igraph graph object |
| <code>type</code> | Character string; either local, nodal, or global (default: local) |
| <code>weights</code> | Numeric vector of edge weights; if NULL (the default), and if the graph has edge attribute weight, then that will be used. To avoid using weights, this should be NA. |
| <code>use.parallel</code> | Logical indicating whether or not to use foreach (default: TRUE) |
| <code>A</code> | Numeric matrix; the (weighted or unweighted) adjacency matrix of the input graph (default: NULL) |

Details

Local efficiency for vertex i is:

$$E_{local}(i) = \frac{1}{N} \sum_{i \in G} E_{global}(G_i)$$

where G_i is the subgraph of neighbors of i , and N is the number of vertices in that subgraph.

Nodal efficiency for vertex i is:

$$E_{nodal}(i) = \frac{1}{N-1} \sum_{j \in G} \frac{1}{d_{ij}}$$

Global efficiency for graph G with N vertices is:

$$E_{global}(G) = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{d_{ij}}$$

where d_{ij} is the shortest path length between vertices i and j . Alternatively, global efficiency is equal to the mean of all nodal efficiencies.

Value

A numeric vector of the efficiencies for each vertex of the graph (if *type* is `local` | `nodal`) or a single number (if *type* is `global`).

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Latora V., Marchiori M. (2001) *Efficient behavior of small-world networks*. Phys Rev Lett, 87.19:198701.
 Latora V., Marchiori M. (2003) *Economic small-world behavior in weighted networks*. Eur Phys J B, 32:249-263.

get.resid

Linear model residuals in structural covariance networks

Description

Runs linear models across brain regions listed in a `data.table` (e.g. cortical thickness), adjusting for variables in `covars` (e.g. age, sex, etc.), and calculates the *externally Studentized* (or *leave-one-out*) residuals.

The `[]` method will let you reorder or subset residuals based on a given numeric vector. However, this is used in bootstrap and permutation analysis and should generally not be called directly by the user.

Print a summary of residuals for structural covariance data

Usage

```
get.resid(dt.vol, covars, method = c("comb.groups", "sep.groups"),
  use.mean = FALSE, exclude = NULL, ...)
```

```
## S3 method for class 'brainGraph_resids'
x[i, g = NULL]
```

```
## S3 method for class 'brainGraph_resids'
summary(object, regions = NULL, ...)
```

Arguments

| | |
|---------------------|---|
| <code>dt.vol</code> | A <code>data.table</code> containing all the volumetric measure of interest (i.e., the object <code>lhrh</code> as output by <code>brainGraph_init</code>) |
| <code>covars</code> | A <code>data.table</code> of the covariates of interest |
| <code>method</code> | Character string indicating whether to test models for subject groups separately or combined (default: <code>comb.groups</code>) |

| | |
|----------|--|
| use.mean | Logical should we control for the mean hemispheric brain value (e.g. mean LH/RH cortical thickness) (default: FALSE) |
| exclude | Character vector of covariates to exclude (default: NULL) |
| ... | Arguments passed to brainGraph_GLM_design (optional) |
| x | A brainGraph_resids object |
| i | Numeric vector of the indices |
| g | Character string indicating the group (default: NULL) |
| object | A brainGraph_resids object |
| regions | Character vector of region(s) to focus on; default behavior is to show summary for all regions |

Details

You can choose to run models for each of your subject groups separately or combined (the default) via the method argument. You may also choose whether or not to include the mean, per-hemisphere structural measure in the models. Finally, you can list variables that are present in covars but you would like to exclude from the models.

Value

An object of class brainGraph_resids with elements:

| | |
|--------------|---|
| X | The <i>design matrix</i> |
| method | The input argument method |
| use.mean | The input argument use.mean |
| all.dat.tidy | The tidied data.table of volumetric data (e.g., mean regional cortical thickness) and covariates, along with <i>resids</i> column added |
| resids.all | The "wide" data.table of residuals |
| groups | Group names |

[summary.brainGraph_resids](#) returns a list with two data tables, one of the residuals, and one of only the outlier regions

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[rstudent](#)

Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_init](#), [brainGraph_permute](#), [corr.matrix](#), [plot.brainGraph_resids](#), [plot_volumetric](#)

Description

`brainGraph_GLM_fit_t` fits a linear model for t-contrasts (i.e., uni-dimensional contrasts) and returns the contrasts of parameter estimates, standard errors, t-statistics, and p-values. If a contrast matrix is supplied, it will return the above values for each row of the matrix.

`brainGraph_GLM_fit_f` fits a linear model for f-contrasts (i.e., multi-dimensional contrasts) and returns the *extra sum of squares* due to the full model, the sum of squared errors of the full model, the f-statistic, and associated p-value.

Usage

```
brainGraph_GLM_fit_t(X, y, XtX, con.mat)
```

```
brainGraph_GLM_fit_f(X, y, dfR, con.mat, rkC, CXtX)
```

Arguments

| | |
|---------|---|
| X | Numeric matrix, if you wish to supply your own design matrix (default: NULL) |
| y | Numeric vector; the outcome variable |
| XtX | Numeric matrix |
| con.mat | Numeric matrix specifying the contrast(s) of interest; if only one contrast is desired, you can supply a vector |
| dfR | Integer; residual degrees of freedom |
| rkC | Integer; rank of the contrast matrix |
| CXtX | Numeric matrix |

Details

For speed purposes (if it is called from `brainGraph_GLM` and permutation testing is done), this function does not do argument checking.

Value

`brainGraph_GLM_fit_t` - A list containing:

| | |
|----------|-------------------------------------|
| gamma | The contrast of parameter estimates |
| se | The standard error |
| contrast | The contrast number(s) |

`brainGraph_GLM_fit_f` - A list containing:

| | |
|----------|---|
| numer | The extra sum of squares due to the full model divided by the rank of the contrast matrix |
| se | The sum of squared errors of the full model |
| contrast | The contrast number; defaults to 1 |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other GLM functions: [brainGraph_GLM_design](#), [brainGraph_GLM](#), [mtpc](#)

GraphDistances

Calculate Euclidean distance of edges and vertices

Description

`edge_spatial_dist` calculates the Euclidean distance of an `igraph` graph object's edges. The distances are in *mm* and based on MNI space. These distances are *NOT* along the cortical surface, so can only be considered approximations, particularly concerning inter-hemispheric connections. The input graph must have *atlas* as a graph-level attribute.

`vertex_spatial_dist` calculates, for each vertex of a graph, the average Euclidean distance across all of that vertex's connections.

Usage

```
edge_spatial_dist(g)
```

```
vertex_spatial_dist(g)
```

Arguments

`g` An `igraph` graph object

Value

`edge_spatial_dist` - a numeric vector with length equal to the edge count of the input graph, consisting of the Euclidean distance (in *mm*) of each edge

`vertex_spatial_dist` - a named numeric vector with length equal to the number of vertices, consisting of the average distance (in *mm*) for each vertex

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Alexander-Bloch A.F., Vertes P.E., Stidd R. et al. (2013) *The anatomical distance of functional connections predicts brain network topology in health and schizophrenia*. *Cerebral Cortex*, 23:127-138.

hoa112 *Coordinates for data from Harvard-Oxford atlas*

Description

This is a list of spatial coordinates for the Harvard-Oxford atlas, along with indices for the major lobes of the brain.

Usage

```
data("hoa112")
```

Format

A data frame with 112 observations on the following 7 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate SCGM

hemi a factor with levels L R

index a numeric vector

References

Makris N., Goldstein J.M., Kennedy D. et al. (2006) *Decreased volume of left and total anterior insular lobule in schizophrenia*. Schizophr Res, 83(2-3):155-171.

Examples

```
data(hoa112)
str(hoa112)
```

IndividualContributions

Approaches to estimate individual network contribution

Description

loo calculates the individual contribution to group network data for each subject in each group using a "leave-one-out" approach. The residuals of a single subject are excluded, and a correlation matrix is created. This is compared to the original correlation matrix using the Mantel test.

aop calculates the individual contribution using an "add-one-patient" approach. The residuals of a single patient are added to those of a control group, and a correlation matrix is created. This is repeated for all individual patients and each patient group.

Usage

```
loo(resids, corrs, level = c("global", "regional"))

aop(resids, corr.mat, level = c("global", "regional"), control.value = 1)
```

Arguments

| | |
|---------------|---|
| resids | An object of class <code>brainGraph_resids</code> (the output from <code>get.resid</code>) |
| corrs | List of lists of correlation matrices (as output by <code>corr.matrix</code>). |
| level | Character string; the level at which you want to calculate contributions (either <code>global</code> or <code>regional</code>) |
| corr.mat | Numeric; correlation matrix of the <i>control</i> group |
| control.value | Integer or character string specifying the control group (default: 1) |

Value

A `data.table` with columns for

| | |
|----------|--|
| Study.ID | Subject identifier |
| Group | Group membership |
| region | If <code>level='regional'</code> |
| IC,RC | The value of the individual/regional contributions |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Saggar M., Hosseini S.M.H., Buno J.L., Quintin E., Raman M.M., Kesler S.R., Reiss A.L. (2015) *Estimating individual contributions from group-based structural correlations networks*. *NeuroImage*, 120:274-284. doi:10.1016/j.neuroimage.2015.07.006

See Also

Other Structural covariance network functions: [brainGraph_boot](#), [brainGraph_init](#), [brainGraph_permute](#), [corr.matrix](#), [get.resid](#), [plot.brainGraph_resids](#), [plot_volumetric](#)

Other Group analysis functions: [NBS](#), [brainGraph_GLM](#), [brainGraph_boot](#), [brainGraph_mediate](#), [brainGraph_permute](#), [mtpc](#)

Examples

```
## Not run:
IC <- loo(resids.all, corrs)
RC <- loo(resids.all, corrs, level='regional')

## End(Not run)
## Not run:
```

```
IC <- aop(resids.all, corrs[[1]]$R)
RC <- aop(resids.all, corrs[[1]]$R, level='regional')

## End(Not run)
```

lpba40

Coordinates for data from the LONI probabilistic brain atlas

Description

This is a list of spatial coordinates for the LPBA40 atlas, along with indices for the major lobes of the brain. The coordinates were obtained from some colleagues.

Usage

```
data("lpba40")
```

Format

A data frame with 56 observations on the following 7 variables.

name a character vector of region names

x.mni a numeric vector of x-coordinates (in MNI space)

y.mni a numeric vector of y-coordinates (in MNI space)

z.mni a numeric vector of z-coordinates (in MNI space)

lobe a factor with levels Frontal Parietal Temporal Occipital Insula Cingulate SCGM

hemi a factor with levels L R

index a numeric vector

References

Shattuck DW, Mirza M, Adisetiyo V, Hojatkashani C, Salamon G, Narr KL, Poldrack RA, Bilder RM, Toga AW (2007) *Construction of a 3D probabilistic atlas of human cortical structures*. NeuroImage, doi:10.1016/j.neuroimage.2007.09.031

Examples

```
data(lpba40)
str(lpba40)
```

| | |
|-----------------|-----------------------------------|
| make_brainGraph | <i>Create a brainGraph object</i> |
|-----------------|-----------------------------------|

Description

This function will assign vertex attributes *lobe* and *lobe.hemi* for all vertices in a graph, given a specific atlas. It will also add attributes *circle.layout*, *x*, *y*, *z* for plotting.

Usage

```
make_brainGraph(g, atlas, rand = FALSE, modality = NULL, weighting = NULL,
  threshold = NULL, subject = NULL, group = NULL)
```

Arguments

| | |
|------------------|--|
| <i>g</i> | An <i>igraph</i> graph object. |
| <i>atlas</i> | Character string specifying the brain atlas |
| <i>rand</i> | A character string indicating whether this function is being run for a random graph or a "graph of interest" (default: FALSE). |
| <i>modality</i> | Character vector indicating imaging modality (e.g. 'dti') (default: NULL) |
| <i>weighting</i> | Character string indicating how the edges are weighted (e.g., 'fa', 'pearson', etc.) (default: NULL) |
| <i>threshold</i> | Numeric indicating the level at which the matrices were thresholded (if at all) (default: NULL) |
| <i>subject</i> | Character vector indicating subject ID (default: NULL) |
| <i>group</i> | Character vector indicating group membership (default: NULL) |

Details

The input graph *g* *must* have a graph attribute named *atlas*, and will assign other attributes specific to the atlas.

For the *modality* argument, you can choose anything you like, but the `summary.brainGraph` knows about *dti*, *fmri*, *thickness*, *area*, and *volume*.

For the *weighting* argument, you can choose anything you like, but `summary.brainGraph` knows about *fa*, *sld* (streamline density, tractography), *pearson*, *spearman*, *kendall*, and *partial* (partial correlation coefficient).

Value

A *brainGraph* graph object with additional attributes:

| | |
|----------------|--|
| <i>version</i> | (graph-level) The current version of <code>brainGraph</code> |
| <i>atlas</i> | (graph-level) |
| <i>lobe</i> | (vertex-level) Character vector of lobe names |

| | |
|------------------------------|---|
| hemi | (vertex-level) Character vector of hemispheres ('L', 'R', or 'B') |
| lobe.hemi | Integer vector indicating the lobe and hemisphere |
| class | (vertex-level) Character vector of class names (if applicable) |
| network | (vertex-level) Character vector of network names (if applicable) |
| modality | (graph-level) |
| weighting | (graph-level) |
| threshold | (graph-level) |
| name | (graph-level) The subject ID |
| Group | (graph-level) |
| x, y, z, x.mni, y.mni, z.mni | Spatial coordinates |
| color.lobe | (vertex- and edge-level) Colors based on <i>lobe</i> |
| color.class, color.network | (vertex- and edge-level) If applicable |
| circle.layout | Integer vector for ordering the vertices for circle plots |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Graph creation functions: [make_ego_brainGraph](#), [make_empty_brainGraph](#), [make_glm_brainGraph](#), [make_mediate_brainGraph](#), [make_nbs_brainGraph](#)

make_ego_brainGraph *Create a graph of the union of multiple vertex neighborhoods*

Description

This function accepts multiple vertices, creates graphs of their neighborhoods (of order 1), and returns the union of those graphs.

Usage

```
make_ego_brainGraph(g, vs)
```

Arguments

| | |
|----|---|
| g | An igraph graph object |
| vs | Either a character or integer vector (vertex names or indices, respectively) for the vertices of interest |

Value

An igraph graph object containing the union of all edges and vertices in the neighborhoods of the input vertices; only the vertex attribute *name* will be present

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[make_ego_graph](#)

Other Graph creation functions: [make_brainGraph](#), [make_empty_brainGraph](#), [make_glm_brainGraph](#), [make_mediate_brainGraph](#), [make_nbs_brainGraph](#)

Examples

```
## Not run:
subg <- make_ego_brainGraph(g1[[N]], c(24, 58))
subg <- make_ego_brainGraph(g1[[N]], c('IPCUN', 'rPCUN'))

## End(Not run)
```

`make_empty_brainGraph` *Create an empty graph with attributes for brainGraph*

Description

This function creates an empty undirected graph with vertex count equal to the atlas specified, and includes some graph-, vertex-, and edge-level attributes that are important for brainGraph functions. Basically a wrapper for [make_empty_graph](#).

Usage

```
make_empty_brainGraph(atlas, ...)
```

Arguments

| | |
|--------------------|---|
| <code>atlas</code> | Character string of the atlas to create a graph from |
| <code>...</code> | Other arguments passed to make_brainGraph |

Value

An empty brainGraph graph object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[make_empty_graph](#)

Other Graph creation functions: [make_brainGraph](#), [make_ego_brainGraph](#), [make_glm_brainGraph](#), [make_mediate_brainGraph](#), [make_nbs_brainGraph](#)

make_glm_brainGraph *Create a graph with GLM-specific attributes*

Description

make_glm_brainGraph will create graphs with attributes specific to the results of [brainGraph_GLM](#) or [mtpc](#). The function returns a list, with one element for each specified contrast.

Usage

```
make_glm_brainGraph(res.glm, atlas, ...)
```

Arguments

| | |
|---------|---|
| res.glm | List as output by brainGraph_GLM or by mtpc . |
| atlas | Character string specifying the brain atlas to use |
| ... | Other arguments passed to make_brainGraph |

Details

This function only creates a graph for *vertex*-level analyses.

Value

A list of igraph graph objects (length equal to the number of contrasts) with additional attributes:

| | |
|--------|---|
| Graph | <i>name</i> (contrast name), <i>outcome</i> (the outcome variable), <i>alpha</i> (the significance level); for MTPC: <i>tau.mtpc</i> , <i>S.mtpc</i> , <i>S.crit</i> , <i>A.crit</i> |
| Vertex | <i>size2</i> (t-statistic), <i>size</i> (the t-stat transformed for visualization purposes), <i>p</i> (equal to $1 - p$), <i>p.fdr</i> (equal to $1 - p_{FDR}$, the FDR-adjusted p-value), <i>gamma</i> (the contrast of parameter estimates, <i>se</i> (the standard error of <i>gamma</i>); <i>A.mtpc</i> , <i>sig</i> (binary indicating whether $A.mtpc > A.crit$) (for MTPC) |

See Also

[brainGraph_GLM](#), [mtpc](#)

Other Graph creation functions: [make_brainGraph](#), [make_ego_brainGraph](#), [make_empty_brainGraph](#), [make_mediate_brainGraph](#), [make_nbs_brainGraph](#)

`make_intersection_brainGraph`*Create the intersection of graphs based on a logical condition*

Description

Create the intersection of graphs based on a logical condition

Usage

```
make_intersection_brainGraph(..., subgraph)
```

Arguments

| | |
|-----------------------|--|
| <code>...</code> | Graph objects or lists of graph objects |
| <code>subgraph</code> | Character string specifying an equation (logical condition) for the vertices to subset |

Value

An igraph graph object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

Examples

```
## Not run:
res.mtpc <- mtpc(g, covars, ...)
g.mtpc <- make_glm_brainGraph(res.mtpc, atlas)
g.mtpc.int <- make_intersection_brainGraph(g.mtpc,
  subgraph='sig == 1')

## End(Not run)
```

`make_mediate_brainGraph`*Create a graph with mediation-specific attributes*

Description

This function only creates a graph for *vertex*-level analyses.

Usage

```
make_mediate_brainGraph(res.med, atlas, ...)
```

Arguments

| | |
|---------|---|
| res.med | List object output by brainGraph_mediate |
| atlas | Character string specifying the brain atlas to use |
| ... | Other arguments passed to make_brainGraph |

Value

A brainGraph_mediate graph object with attributes:

| | |
|--------|---|
| Graph | <i>mediator, treat, outcome, nobis</i> |
| Vertex | <i>b?.acme, p?.acme, b?.ade, p?.ade, b?.prop, p?.prop, b.tot, p.tot</i> |

See Also

Other Graph creation functions: [make_brainGraph](#), [make_ego_brainGraph](#), [make_empty_brainGraph](#), [make_glm_brainGraph](#), [make_nbs_brainGraph](#)

make_nbs_brainGraph *Create a graph with NBS-specific attributes*

Description

Create a graph with NBS-specific attributes

Usage

```
make_nbs_brainGraph(res.nbs, atlas, ...)
```

Arguments

| | |
|---------|---|
| res.nbs | List that is output by NBS |
| atlas | Character string specifying the brain atlas to use |
| ... | Other arguments passed to make_brainGraph |

Value

A list of igraph graph objects (length equal to the number of contrasts) with additional attributes:

| | |
|--------|---|
| Graph | <i>name</i> (contrast name) |
| Vertex | <i>comp</i> (integer vector indicating connected component membership), <i>p.nbs</i> (P-value for each component) |
| Edge | <i>stat</i> (the test statistic for each connection), <i>p</i> (the P-value) |

See Also

Other Graph creation functions: [make_brainGraph](#), [make_ego_brainGraph](#), [make_empty_brainGraph](#), [make_glm_brainGraph](#), [make_mediate_brainGraph](#)

| | |
|------|---|
| mtpc | <i>Multi-threshold permutation correction</i> |
|------|---|

Description

Applies the *multi-threshold permutation correction (MTPC)* method to perform inference in graph theory analyses of brain MRI data.

Print a summary of MTPC results

Usage

```
mtpc(g.list, thresholds, covars, measure, con.mat, con.type = c("t", "f"),
     con.name = NULL, level = c("vertex", "graph"), clust.size = 3L,
     N = 500L, perms = NULL, alpha = 0.05, res.glm = NULL, long = TRUE,
     ...)
```

```
## S3 method for class 'mtpc'
summary(object, contrast = NULL, digits = max(3L,
      getOption("digits") - 2L), print.head = TRUE, ...)
```

Arguments

| | |
|-------------------------|--|
| <code>g.list</code> | A list of lists of <code>igraph</code> graph objects for all thresholds and subjects |
| <code>thresholds</code> | Numeric vector of the thresholds applied to the raw connectivity matrices. |
| <code>covars</code> | A <code>data.table</code> of covariates |
| <code>measure</code> | Character string of the graph measure of interest |
| <code>con.mat</code> | Numeric matrix specifying the contrast(s) of interest; if only one contrast is desired, you can supply a vector |
| <code>con.type</code> | Character string; either 't' or 'f' (for t or F-statistics). Default: 't' |
| <code>con.name</code> | Character vector of the contrast name(s); if <code>con.mat</code> has row names, those will be used for reporting results (default: NULL) |
| <code>level</code> | Character string; either <code>vertex</code> (default) or <code>graph</code> |
| <code>clust.size</code> | Integer indicating the size of "clusters" (i.e., consecutive thresholds for which the observed statistic exceeds the null) (default: 3L) |
| <code>N</code> | Integer; number of permutations to create (default: 5e3) |
| <code>perms</code> | Matrix of permutations, if you would like to provide your own (default: NULL) |
| <code>alpha</code> | Numeric; the significance level (default: 0.05) |
| <code>res.glm</code> | A list of <code>bg_GLM</code> objects, as output by a previous run of <code>mtpc</code> . Useful if you want to change the cluster size without re-running all of the GLM's and permutations (default: NULL) |
| <code>long</code> | Logical indicating whether or not to return all permutation results (default: FALSE) |
| <code>...</code> | Other arguments passed to <code>brainGraph_GLM</code> and/or <code>brainGraph_GLM_design</code> |

| | |
|------------|--|
| object | A mtpc object |
| contrast | Integer specifying the contrast to summarize; defaults to showing results for all contrasts |
| digits | Integer specifying the number of digits to display for p-values |
| print.head | Logical indicating whether or not to print only the first and last 5 rows of the statistics tables (default: TRUE) |

Details

This is a multi-step procedure: (steps 3-4 are the time-consuming steps)

1. Apply thresholds τ to the networks, and compute network metrics for all networks and thresholds. (already done beforehand)
2. Compute test statistics S_{obs} for each threshold. (done by [brainGraph_GLM](#))
3. Permute group assignments and compute test statistics for each permutation and threshold. (done by [brainGraph_GLM](#))
4. Build a null distribution of the maximum statistic across thresholds (and across brain regions) for each permutation. (done by [brainGraph_GLM](#))
5. Determine the critical value, S_{crit} from the null distribution of maximum statistics.
6. Identify clusters where $S_{obs} > S_{crit}$ and compute the AUC for these clusters (denoted A_{MTPC}).
7. Compute a critical AUC (A_{crit}) from the mean of the supra-critical AUC's for the permuted tests.
8. Reject H_0 if $A_{MTPC} > A_{crit}$.

Value

An object of class mtpc with some input arguments plus the following elements:

| | |
|------------|---|
| res.glm | List with length equal to the number of thresholds; each list element is the output from brainGraph_GLM |
| DT | A data.table for all thresholds, combined from the outputs of brainGraph_GLM |
| stats | A data.table containing S.mtpc (the max. observed statistic), tau.mtpc (the threshold of the max. observed statistic), S.crit (the critical statistic value), and A.crit (the critical AUC) |
| null.dist | Numeric matrix with N rows and number of columns equal to the number of thresholds. Each element is the maximum statistic for that permutation and threshold. |
| perm.order | Numeric matrix; the permutation set applied for all thresholds (each row is a separate permutation) |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Drakesmith M, Caeyenberghs K, Dutt A, Lewis G, David AS, Jones DK (2015). *Overcoming the effects of false positives and threshold bias in graph theoretical analyses of neuroimaging data*. *NeuroImage*, 118:313-333.

See Also

Other Group analysis functions: [IndividualContributions](#), [NBS](#), [brainGraph_GLM](#), [brainGraph_boot](#), [brainGraph_mediate](#), [brainGraph_permute](#)

Other GLM functions: [GLMfit](#), [brainGraph_GLM_design](#), [brainGraph_GLM](#)

Examples

```
## Not run:
diffs.mtpc <- mtpc(g.list=g.norm, thresholds=thresholds, N=N,
  covars=covars.dti, measure='E.nodal.wt', coding='effects',
  con.mat=c(0, 0, 0, 0, -2), alt='greater',
  binarize=c('Sex', 'Scanner'), con.name='Group 1 > Group 2')
sig.regions <- diffs.mtpc$DT[A.mtpc > A.crit]

## End(Not run)
```

NBS

Network-based statistic for brain MRI data

Description

Calculates the *network-based statistic (NBS)*, which allows for family-wise error (FWE) control over network data, introduced for brain MRI data by Zalesky et al. Accepts a three-dimensional array of all subjects' connectivity matrices and a `data.table` of covariates, and creates a null distribution of the largest connected component size by permuting subjects across groups. The covariates `data.table` must have (at least) a *Group* column.

Print a summary of NBS analysis

Usage

```
NBS(A, covars, con.mat, con.type = c("t", "f"), X = NULL, con.name = NULL,
  p.init = 0.001, N = 1000, perms = NULL, symm.by = c("max", "min",
  "avg"), alternative = c("two.sided", "less", "greater"), long = FALSE,
  ...)
```

```
## S3 method for class 'NBS'
summary(object, contrast = NULL, digits = max(3L,
  getOption("digits") - 2L), ...)
```

Arguments

| | |
|-------------|--|
| A | Three-dimensional array of all subjects' connectivity matrices |
| covars | A data.table of covariates |
| con.mat | Numeric matrix specifying the contrast(s) of interest; if only one contrast is desired, you can supply a vector |
| con.type | Character string; either 't' or 'f' (for t or F-statistics). Default: 't' |
| X | Numeric matrix, if you wish to supply your own design matrix (default: NULL) |
| con.name | Character vector of the contrast name(s); if con.mat has row names, those will be used for reporting results (default: NULL) |
| p.init | Numeric; the initial p-value threshold (default: 0.001) |
| N | Integer; number of permutations to create (default: 5e3) |
| perms | Matrix of permutations, if you would like to provide your own (default: NULL) |
| symm.by | Character string; how to create symmetric off-diagonal elements (default: max) |
| alternative | Character string, whether to do a two- or one-sided test (default: 'two.sided') |
| long | Logical indicating whether or not to return all permutation results (default: FALSE) |
| ... | Other arguments passed to brainGraph_GLM_design |
| object | A NBS object |
| contrast | Integer specifying the contrast to summarize; defaults to showing results for all contrasts |
| digits | Integer specifying the number of digits to display for p-values |

Details

The graph that is returned by this function will have a `t.stat` edge attribute which is the t-statistic for that particular connection, along with a `p` edge attribute, which is the p-value for that connection. Additionally, each vertex will have a `p.nbs` attribute representing $1 -$ the p-value associated with that vertex's component.

Value

An object of class NBS with some input arguments in addition to:

| | |
|------------|---|
| X | The design matrix |
| removed | Character vector of subject ID's removed due to incomplete data (if any) |
| T.mat | List of numeric matrices (symmetric) containing the statistics for each edge |
| p.mat | List of numeric matrices (symmetric) containing the P-values |
| components | List containing data tables of the observed and permuted connected component sizes and P-values |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Zalesky A., Fornito A., Bullmore E.T. (2010) *Network-based statistic: identifying differences in brain networks*. NeuroImage, 53(4):1197-1207.

See Also

[brainGraph_GLM_design](#), [brainGraph_GLM_fit_t](#)

Other Group analysis functions: [IndividualContributions](#), [brainGraph_GLM](#), [brainGraph_boot](#), [brainGraph_mediate](#), [brainGraph_permute](#), [mtpc](#)

Examples

```
## Not run:
max.comp.nbs <- NBS(A.norm.sub[[1]], covars.dti, N=5e3)

## End(Not run)
```

plot.bg_GLM

Plot GLM diagnostics for a brain network

Description

Plots the GLM diagnostics (similar to that of [plot.lm](#)) for the output of [brainGraph_GLM](#). There are a total of 6 possible plots, specified by the `which` argument; the behavior is the same as in [plot.lm](#). Please see the help for that function.

Usage

```
## S3 method for class 'bg_GLM'
plot(x, region = NULL, which = c(1L:3L, 5L), ...)
```

Arguments

| | |
|---------------------|--|
| <code>x</code> | A <code>bg_GLM</code> object |
| <code>region</code> | Character string specifying which region's results to plot; only relevant if <code>level='vertex'</code> (default: <code>NULL</code>) |
| <code>which</code> | Integer vector indicating which of the 6 plots to print to the plot device (default: <code>c(1:3, 5)</code>) |
| <code>...</code> | Unused |

Value

A list of [ggplot](#) objects

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also[plot.lm](#)**Examples**

```
## Not run:
## Save objects and then to multipage PDF
lmPlots <- plot(x)
ggsave('lmPlots.pdf', lmPlots)

## Save all the GLM sub-objects from MTPC analysis
res.mtpc <- mtpc(...)
glmPlots <- lapply(res.mtpc$res.glm, plot, which=1:6)
m1 <- marrangeGrob(glmPlots, nrow=1, ncol=1)
ggsave('glmPlots.pdf', m1, width=8.5, height=11)

## End(Not run)
```

`plot.brainGraph`*Plot a brain graph with a specific spatial layout*

Description

`plot.brainGraph` plots a graph in which the spatial layout of the nodes is important. The network itself is plotted over a brain MRI slice from the MNI152 template if `mni=TRUE`.

Usage

```
## S3 method for class 'brainGraph'
plot(x, plane = c("axial", "sagittal", "circular"),
     hemi = c("both", "L", "R"), subgraph = NULL, show.legend = FALSE,
     rescale = FALSE, asp = 0, main = NULL, subt = "default", mni = TRUE,
     ...)

plot_brainGraph(x, plane = c("axial", "sagittal", "circular"),
               hemi = c("both", "L", "R"), subgraph = NULL, show.legend = FALSE,
               rescale = FALSE, asp = 0, main = NULL, subt = "default", mni = TRUE,
               ...)
```

Arguments

| | |
|--------------------------|--|
| <code>x</code> | A <code>brainGraph</code> graph object |
| <code>plane</code> | Character string indicating which orientation to plot (default: 'axial') |
| <code>hemi</code> | Character string indicating which hemisphere to plot (default: 'both') |
| <code>subgraph</code> | Character string specifying an equation for vertices to plot (default: NULL) |
| <code>show.legend</code> | Logical indicating whether or not to show a legend (default: FALSE) |

| | |
|---------|---|
| rescale | Logical, whether to rescale the coordinates (default: FALSE) |
| asp | Numeric constant; the aspect ratio (default: 0) |
| main | Character string; the main title (default: NULL) |
| subt | Character string; the subtitle (default: default) |
| mni | Logical indicating whether or not to plot over a slice of the brain (default: TRUE) |
| ... | Other parameters (passed to plot.igraph). See igraph.plotting for details. |

Details

With the argument `subgraph`, you can specify a simple logical equation for which vertices to show. For example, `'degree > 10'` will plot only vertices with a *degree* greater than 10. Combinations of *AND* (i.e., `&`) and *OR* (i.e., `|`) are allowed.

To remove the subtitle at the bottom, simply specify `subt=NULL`.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

Examples

```
## Not run:
plot(g[[1]], hemi='R')
plot(g[[1]], subgraph='degree > 10 | btwn.cent > 50')

## End(Not run)
```

`plot.brainGraph_boot` *Plot bootstrap output of global graph measures across densities*

Description

This function plots the output of [brainGraph_boot](#), returning two `ggplot` objects: one with shaded regions based on the standard error, and the other based on confidence intervals (calculated using the normal approximation).

Usage

```
## S3 method for class 'brainGraph_boot'
plot(x, ..., alpha = 0.4)
```

Arguments

| | |
|-------|--|
| x | The object output from brainGraph_boot |
| ... | Not used |
| alpha | A numeric indicating the opacity for geom_ribbon |

Value

A *list* with the following elements:

| | |
|----|---|
| se | A ggplot object with ribbon representing standard error |
| ci | A ggplot object with ribbon representing confidence intervals |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[brainGraph_boot](#), [boot.ci](#)

plot.brainGraph_GLM *Plot a graph with results from brainGraph_GLM*

Description

This is a convenience function for plotting a graph based on results from [brainGraph_GLM](#). There are a few argument defaults: to plot only those vertices for which $p < \alpha$; a plot title with the outcome measure and contrast name, and to omit the plot subtitle.

Usage

```
## S3 method for class 'brainGraph_GLM'
plot(x, p.sig = c("p", "p.fdr", "p.perm"),
     subgraph = NULL, main = paste0("\n\n", x$outcome, ": ", x$name),
     subt = NULL, cex.main = 2, ...)
```

Arguments

| | |
|----------|---|
| x | A brainGraph_GLM graph object (from make_glm_brainGraph) |
| p.sig | Character string indicating which p-value to use for determining significance (default: p) |
| subgraph | Character string specifying an equation for vertices to plot (default: NULL) |
| main | Character string; the main title (default: NULL) |
| subt | Character string; the subtitle (default: default) |
| cex.main | Numeric indicating the scaling for plot title size (see par). |
| ... | Other parameters (passed to plot.igraph). See igraph.plotting for details. |

See Also

Other Plotting functions: [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

plot.brainGraph_mediate

Plot a graph with results from a mediation analysis

Description

Plot a graph with results from a mediation analysis

Usage

```
## S3 method for class 'brainGraph_mediate'
plot(x, subgraph = "p.acme > 0.95",
     main = sprintf("\n\nEffect of \"%s\" on\n\"%s\"\nmediated by \"%s\"",
                    x$treat, x$outcome, x$mediator), subt = NULL, cex.main = 1, ...)
```

Arguments

| | |
|----------|---|
| x | A brainGraph_mediate graph object (from make_mediate_brainGraph) |
| subgraph | Character string specifying an equation for vertices to plot (default: NULL) |
| main | Character string; the main title (default: NULL) |
| subt | Character string; the subtitle (default: default) |
| cex.main | Numeric indicating the scaling for plot title size (see par). |
| ... | Other parameters (passed to plot.igraph). See igraph.plotting for details. |

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

plot.brainGraph_mtpc *Plot a graph with results from MTPC*

Description

This is a convenience function for plotting a graph based on results from [mtpc](#). There are a few argument defaults: to plot only those vertices for which $A_{mtpc} > A_{crit}$; a plot title with the outcome measure and contrast name, and to omit the plot subtitle.

Usage

```
## S3 method for class 'brainGraph_mtpc'
plot(x, subgraph = "sig == 1",
     main = paste0("\n\n", x$outcome, ": ", x$name), subt = NULL,
     cex.main = 2, ...)
```

Arguments

| | |
|----------|---|
| x | A brainGraph_mtpc graph object (from make_glm_brainGraph) |
| subgraph | Character string specifying an equation for vertices to plot (default: NULL) |
| main | Character string; the main title (default: NULL) |
| subt | Character string; the subtitle (default: default) |
| cex.main | Numeric indicating the scaling for plot title size (see par). |
| ... | Other parameters (passed to plot.igraph). See igraph.plotting for details. |

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

plot.brainGraph_NBS *Plot a graph with results from the network-based statistic*

Description

This is a convenience function for plotting a graph based on results from [NBS](#). There are several default arguments that are set: vertex/edge colors will correspond to connected component membership, and only those vertices in which $V(g)_{p.nbs} > 1 - \alpha$ will be shown. Finally, vertex names will be omitted.

Usage

```
## S3 method for class 'brainGraph_NBS'
plot(x, alpha = 0.05, subgraph = paste("p.nbs >", 1
  - alpha), vertex.label = NA, vertex.color = "color.comp",
  edge.color = "color.comp", subt = NULL, main = paste0("\n\nNBS: ",
  x$name), cex.main = 2, ...)
```

Arguments

| | |
|--------------|---|
| x | A brainGraph_NBS graph object (from make_nbs_brainGraph) |
| alpha | Numeric; the significance level (default: 0.05) |
| subgraph | Character string specifying the condition for subsetting the graph. By default, it will show only the vertices which are members of components determined to be significant based on alpha. |
| vertex.label | Character vector of the vertex labels to be displayed. Default behavior is to omit them. |
| vertex.color | Character string specifying the vertex attribute to color the vertices by (default: color.comp, which groups vertices by connected component) |
| edge.color | Character string specifying the edge attribute to color the edges by (default: color.comp, which groups edges by connected component) |
| subt | Character string; the subtitle (default: default) |
| main | Character string; the main title (default: NULL) |
| cex.main | Numeric; the scaling factor for text size; see par (default: 2) |
| ... | Other arguments passed to plot.brainGraph |

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

plot.brainGraph_permute

Plot results from permutation testing

Description

Plot results from permutation testing

Usage

```
## S3 method for class 'brainGraph_permute'
plot(x, measure = NULL,
  alternative = c("two.sided", "less", "greater"), alpha = 0.05,
  p.sig = c("p", "p.fdr"), ptitle = NULL, ...)
```

Arguments

| | |
|-------------|--|
| x | A brainGraph_permute object (output by brainGraph_permute). |
| measure | Character string of the measure to focus on (if level='graph') |
| alternative | Character string, whether to do a two- or one-sided test (default: 'two.sided') |
| alpha | Numeric; the significance level (default: 0.05) |
| p.sig | Character string specifying which p-value to use for displaying significant results (default: p) |
| ptitle | Character string specifying a title for the plot (default: NULL) |
| ... | Unused |

Value

A list of `ggplot` objects

plot.brainGraph_resids

Plot model residuals for each brain region

Description

Check the model residuals for each brain region in a structural covariance analysis. It shows a *qqplot* of the studentized residuals, as output from [get.resid](#).

Usage

```
## S3 method for class 'brainGraph_resids'
plot(x, regions = NULL, cols = FALSE, ...)
```

Arguments

| | |
|---------|--|
| x | A brainGraph_resids object (from get.resid) |
| regions | Character vector of region(s) to focus on; default behavior is to show summary for all regions |
| cols | Logical indicating whether to color by group (default: FALSE) |
| ... | Arguments passed to brainGraph_GLM_design (optional) |

Value

A list of `ggplot` objects

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also[qqnorm](#)Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_init](#), [brainGraph_permute](#), [corr.matrix](#), [get.resid](#), [plot_volumetric](#)**Examples**

```
## Not run:
myresids <- get.resids(lhrh, covars)
residPlots <- plot(myresids, cols=TRUE)

## Save as a multi-page PDF
m1 <- marrangeGrob(residPlots, nrow=3, ncol=3)
ggsave('residuals.pdf', m1)

## End(Not run)
```

plot.mtpc

*Plot statistics from an MTPC analysis***Description**

Plot the statistics from an MTPC analysis, along with the maximum permuted statistics. The output is similar to Figure 11 in Drakesmith et al. (2015).

Usage

```
## S3 method for class 'mtpc'
plot(x, contrast = 1L, region = NULL,
     only.sig.regions = TRUE, show.null = TRUE, caption.stats = FALSE, ...)
```

Arguments

| | |
|------------------|---|
| x | A mtpc object |
| contrast | Integer specifying which contrast to plot results for (default: 1) |
| region | Character string specifying which region's results to plot; only relevant if level='vertex' (default: NULL) |
| only.sig.regions | Logical indicating whether to plot only significant regions (default: TRUE) |
| show.null | Logical indicating whether to plot points of the maximum null statistics (per permutation) |
| caption.stats | Logical indicating whether to print the MTPC statistics in the caption of the plot (default: FALSE) |
| ... | Unused |

Value

A list of `ggplot` objects

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Drakesmith M, Caeyenberghs K, Dutt A, Lewis G, David AS, Jones DK (2015). *Overcoming the effects of false positives and threshold bias in graph theoretical analyses of neuroimaging data*. *NeuroImage*, 118:313-333.

Examples

```
## Not run:
mtpcPlots <- plot(mtpc.diffs)

## Arrange plots into 3x3 grids
m1 <- marrangeGrob(mtpcPlots, nrow=3, ncol=3)
ggsave('mtpc.pdf', m1)

## End(Not run)
```

plot_brainGraph_gui *GUI for plotting graphs overlaid on an MNI152 image or in a circle.*

Description

This function creates a GUI for plotting graphs over an image from the MNI152 template. It gives the user control over several plotting parameters. Also possible is a circular plot (in addition to the axial and sagittal views). It is necessary for the graphs to have an *atlas* attribute, and several vertex- and edge-level attributes (set by [set_brainGraph_attr](#)).

Usage

```
plot_brainGraph_gui()
```

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_list](#), [plot_brainGraph_multi](#)

plot_brainGraph_list *Write PNG files for a list of graphs*

Description

This function takes a list of igraph graph objects and plots them over an axial slice of the brain. A png file is written for each element of the list, which can be joined as a gif or converted to video using a tool outside of R.

Usage

```
plot_brainGraph_list(g.list, fname.base, diffs = FALSE, ...)
```

Arguments

| | |
|-------------------------|---|
| <code>g.list</code> | A list of igraph graph objects |
| <code>fname.base</code> | A character string specifying the base of the filename for <i>png</i> output |
| <code>diffs</code> | A logical, indicating whether or not to highlight edge differences (default: FALSE) |
| <code>...</code> | Other parameters (passed to plot_brainGraph) |

Details

You can choose to highlight edge differences between subsequent list elements; in this case, new/different edges are colored pink.

This function may be particularly useful if the graph list contains graphs of a single subject group at incremental densities, or if the graph list contains graphs of each subject in a group.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_multi](#)

plot_brainGraph_multi *Save PNG of three views of a brain graph*

Description

This function will save a PNG file to disk containing three views (columns) of a brain graph (from left-to-right): left sagittal, axial, and right sagittal. The number of rows in the figure will equal the number of groups to plot.

Usage

```
plot_brainGraph_multi(g.list, groups = 1, N = 1, filename = "tmp.png",
  subgraph = NULL, main = NULL, ...)
```

Arguments

| | |
|-----------------------|--|
| <code>g.list</code> | A list of lists of igraph graph objects |
| <code>groups</code> | An integer vector indicating which groups to plot; corresponds to the first element of the list <code>g.list</code> (default: 1) |
| <code>N</code> | Integer corresponding to the second element of the list <code>g.list</code> (default: 1) |
| <code>filename</code> | Character string of the filename of the PNG to be written (default: 'tmp.png') |
| <code>subgraph</code> | A list of character strings to (optionally) subset the graph(s), possibly by multiple conditions (default: NULL) |
| <code>main</code> | A list of character strings to be placed in the main title of the center plot for each group (default: NULL) |
| <code>...</code> | Other arguments passed to plot_brainGraph |

Details

The function argument `N` tells the function to use the `N`-th element of the input list `g.list` for each group. So, for example, if `g.list` consists of lists of graphs for two groups, and `N` is 4, then the plots for `g.list[[1]][[4]]` and `g.list[[2]][[4]]` will be written to the file.

The `subgraph` argument can be used to apply one or more conditions for subsetting the graph. If you would like multiple conditions, then it must be a list variable that equals in length to the number of groups. For a single group and multiple conditions, simply write e.g., `groups=c(1,1)`. The `main` argument has the same rule except it controls the main plot title, which appears in the *axial* view along with the *Group* name.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Plotting functions: [plot.brainGraph_GLM](#), [plot.brainGraph_NBS](#), [plot.brainGraph_mediate](#), [plot.brainGraph_mtpc](#), [plot.brainGraph](#), [plot_brainGraph_gui](#), [plot_brainGraph_list](#)

Examples

```
## Not run:
plot_brainGraph_multi(g.hubs, groups=1:2, filename='Figure01_hubs.png',
  subgraph='N > 0', vertex.color='color.lobe', vertex.size=15,
  show.legend=TRUE, vertex.label.cex=1.5)
## Single group, different subgraphs for each plot
plot_brainGraph_multi(g, groups=c(1, 1), N=5, filename='5_6core.png',
  vertex.color='color.lobe', edge.color='color.lobe', vertex.label=NA,
  subgraph=list('coreness > 5', 'coreness > 6'),
  main=list('k-core 5', 'k-core 6'))

## End(Not run)
```

| | |
|---------------|----------------------------------|
| plot_corr_mat | <i>Plot a correlation matrix</i> |
|---------------|----------------------------------|

Description

This function will plot a correlation matrix in the form of a “heatmap”. You have the choice to plot the vertices in an order based on either community or lobe membership, and they will be colored accordingly.

Usage

```
plot_corr_mat(corrs, ordered = TRUE, type = c("comm", "comm.wt", "lobe",
  "network"), g = NULL, group = NULL)
```

Arguments

| | |
|---------|--|
| corrs | The correlation matrix |
| ordered | A logical indicating whether or not to order vertices (default:TRUE) |
| type | Character string, one of: 'comm', 'comm.wt', 'lobe', or 'network' |
| g | An igraph graph object; not required if <i>ordered</i> is FALSE |
| group | A character vector of the group name (default: NULL) |

Value

A ggplot object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[geom_tile](#)

Examples

```
## Not run:
matplot1 <- plot_corr_mat(corrmat[[1]]$r.thresh[, , N], g=g[[1]][[N]],
                        group=groups[1])

## End(Not run)
```

plot_global

Plot global graph measures across densities

Description

Create a faceted line plot of global graph measures across a range of graph densities. Given a "tidied" `data.table`, you can choose to insert a dashed vertical line at a density of interest, rename the variable levels (which become the facet titles), exclude certain variables, and include a `data.table` of permutation data to add asterisks indicating significant group differences.

Usage

```
plot_global(tidy.dt, xvar = c("density", "threshold"), vline = NULL,
            level.names = NULL, exclude = NULL, perms = NULL, g = NULL,
            alt = NULL)
```

Arguments

| | |
|--------------------------|---|
| <code>tidy.dt</code> | A <code>data.table</code> that has been "tidied", containing global graph measures for all densities and subject groups |
| <code>xvar</code> | A character string indicating whether the variable of interest is "density" or "threshold" (e.g. with DTI data) |
| <code>vline</code> | Numeric; required to plot a dashed vertical line (default: <code>NULL</code>) |
| <code>level.names</code> | Character vector of facet label names, if you wish to change them (default: <code>NULL</code>) |
| <code>exclude</code> | Character vector of variables to exclude (default: <code>NULL</code>) |
| <code>perms</code> | A <code>data.table</code> of permutation group differences (default: <code>NULL</code>) |
| <code>g</code> | A list of lists of <code>igraph</code> graph objects; required if <code>perms</code> is provided (default: <code>NULL</code>) |
| <code>alt</code> | Character vector of alternative hypotheses; required if <code>perms</code> is provided, but defaults to "two.sided" for all variables |

Value

A `ggplot` object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

| | |
|----------------|--|
| plot_rich_norm | <i>Plot normalized rich club coefficients against degree threshold</i> |
|----------------|--|

Description

Returns a [ggplot](#) object of a line plot of the normalized rich club coefficient for up to two subject groups. Optionally will include a shaded region demarcating the [rich_core](#) cutoff.

Usage

```
plot_rich_norm(rich.dt, facet.by = c("density", "threshold"), densities,
  alpha = 0.05, fdr = TRUE, g = NULL)
```

Arguments

| | |
|------------------------|--|
| <code>rich.dt</code> | A <code>data.table</code> with rich-club coefficients |
| <code>facet.by</code> | A character string indicating whether the variable of interest is "density" or "threshold" (e.g. with DTI data) |
| <code>densities</code> | A numeric vector of the densities to plot |
| <code>alpha</code> | The significance level (default: 0.05) |
| <code>fdr</code> | A logical, indicating whether or not to use the FDR-adjusted p-value for determining significance (default: TRUE) |
| <code>g</code> | A list (of lists) of <code>igraph</code> graph objects; required if you want to plot a shaded region demarcating the rich_core |

Value

A [ggplot](#) object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Rich-club functions: [RichClub](#), [rich_club_attr](#)s

Examples

```
## Not run:
plot_rich_norm(rich.dt, facet.by='density', densities[N:(N+1)], g=g)

## End(Not run)
```

plot_vertex_measures *Plot vertex-level graph measures at a single density or threshold*

Description

This function creates boxplots of a single vertex-level graph measure at a single density or threshold, grouped by the variable specified by `facet.by` (e.g., *lobe* or *network*).

Usage

```
plot_vertex_measures(tidy.dt, facet.by = "lobe", measure = "btwn.cent",  
  show.points = FALSE, ylabel = NULL)
```

Arguments

| | |
|--------------------------|---|
| <code>tidy.dt</code> | A “tidied” data.table of vertex-level graph measures |
| <code>facet.by</code> | Character string indicating whether the data should be plotted separately by a certain variable (default: 'lobe') |
| <code>measure</code> | A character string of the graph measure to plot (default: 'btwn.cent') |
| <code>show.points</code> | Logical indicating whether or not to show individual data points (default: FALSE) |
| <code>ylabel</code> | A character string for the y-axis label |

Value

A ggplot object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

Examples

```
## Not run:  
ggp.btwn <- plot_vertex_measures(dt.V.tidy, facet.by='network',  
  measure='E.nodal')  
  
## End(Not run)
```

| | |
|-----------------|---|
| plot_volumetric | <i>Plot group distributions of volumetric measures for a given brain region</i> |
|-----------------|---|

Description

This function takes a "tidied" dataset of cortical volumetric measures (thickness, volume, LGI, etc.) and plots a histogram or violin plot for 1 or more groups, and of 1 or more brain regions.

Usage

```
plot_volumetric(dat, regions, type = c("violin", "histogram"),
  all.vals = TRUE, modality = c("thickness", "volume", "lgi", "area"))
```

Arguments

| | |
|----------|--|
| dat | A data table of volumetric data; needs columns for 'Group', 'region', and 'value' |
| regions | A vector of character strings or integers of the brain region(s) to plot; if integer, the region(s) is/are chosen from the input data table based on the index |
| type | A character string indicating the plot type; either 'histogram' or 'violin' |
| all.vals | A logical indicating whether or not to plot horizontal lines for all observations (only valid for 'violin' plots) (default: TRUE) |
| modality | A character string indicating the type of volumetric measure ('thickness', 'volume', 'lgi', or 'area') |

Value

A ggplot object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[geom_histogram](#), [geom_vline](#)

Other Structural covariance network functions: [IndividualContributions](#), [brainGraph_boot](#), [brainGraph_init](#), [brainGraph_permute](#), [corr.matrix](#), [get.resid](#), [plot.brainGraph_resids](#)

| | |
|--------------|--|
| RandomGraphs | <i>Simulate N random graphs w/ same clustering and degree sequence as the input.</i> |
|--------------|--|

Description

`sim.rand.graph.par` simulates N simple random graphs with the same clustering (optional) and degree sequence as the input. Essentially a wrapper for `sample_degseq` (or, if you want to match by clustering, `sim.rand.graph.clust`) and `set_brainGraph_attr`. It uses `foreach` for parallel processing.

`sim.rand.graph.clust` simulates a random graph with a given degree sequence *and* clustering coefficient. Increasing the `max.iters` value will result in a closer match of clustering with the observed graph.

Usage

```
sim.rand.graph.par(g, N = 100, clustering = FALSE, ...)
```

```
sim.rand.graph.clust(g, rewire.iters = 10000, cl = g$transitivity,
  max.iters = 100)
```

Arguments

| | |
|---------------------------|---|
| <code>g</code> | An igraph graph object |
| <code>N</code> | Integer; the number of random graphs to simulate (default: 100) |
| <code>clustering</code> | Logical; whether or not to control for clustering (default: FALSE) |
| <code>...</code> | Other parameters (passed to <code>sim.rand.graph.clust</code>) |
| <code>rewire.iters</code> | Integer; number of rewiring iterations for the initial graph randomization (default: 1e4) |
| <code>cl</code> | The clustering measure (default: transitivity) |
| <code>max.iters</code> | The maximum number of iterations to perform; choosing a lower number may result in clustering that is further away from the observed graph's (default: 100) |

Details

If you do not want to match by clustering, then simple rewiring of the input graph is performed (the number of rewires equaling the larger of 1e4 and $10 \times m$, where m is the graph's edge count).

Value

`sim.rand.graph.par` - a *list* of N random graphs with some additional vertex and graph attributes
`sim.rand.graph.clust` - A single igraph graph object

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Bansal S., Khandelwal S., Meyers L.A. (2009) *Exploring biological network structure with clustered random networks*. BMC Bioinformatics, 10:405-421.

See Also

[rewire](#), [sample_degseq](#), [keeping_degseq](#)
[transitivity](#)

Other Random graph functions: [RichClub](#), [analysis_random_graphs](#)

Examples

```
## Not run:  
rand1 <- sim.rand.graph.par(g[[1]][[N]], N=1e3)  
rand1.cl <- sim.rand.graph.par(g[[1]][[N]], N=1e2,  
  clustering=T, max.iters=1e3)  
  
## End(Not run)
```

RichClub

Rich club calculations

Description

`rich_club_coeff` calculates the *rich club* of a graph, returning the rich-club coefficient, ϕ , and the subgraph of rich club vertices.

`rich_club_norm` will (optionally) generate a number of random graphs, calculate their rich club coefficients (ϕ), and return ϕ_{norm} of the graph of interest, which is the observed rich-club coefficient divided by the mean across the random graphs.

`rich_core` finds the boundary of the rich core of a graph, based on the decreasing order of vertex degree. It also calculates the degree that corresponds to that rank, and the core size relative to the total number of vertices in the graph.

Usage

```
rich_club_coeff(g, k = 1, weighted = FALSE)  
  
rich_club_norm(g, N = 100, rand = NULL, ...)  
  
rich_core(g)
```

Arguments

| | |
|-----------------------|--|
| <code>g</code> | An igraph graph object |
| <code>k</code> | Integer; the minimum degree for including a vertex (default: 1) |
| <code>weighted</code> | Logical indicating whether or not edge weights should be used (default: FALSE) |
| <code>N</code> | Integer; the number of random graphs to generate (default: 100) |
| <code>rand</code> | A list of igraph graph objects, if random graphs have already been generated (default: NULL) |
| <code>...</code> | Other parameters (passed to <code>sim.rand.graph.par</code>) |

Details

If random graphs have already been generated, you can supply a list as an argument (since graph generation is time consuming).

Value

`rich_club_coeff` - a list with components:

| | |
|--------------------|--|
| <code>phi</code> | The rich club coefficient, ϕ . |
| <code>graph</code> | A subgraph containing only the rich club vertices. |
| <code>Nk</code> | The number of vertices in the rich club graph. |
| <code>Ek</code> | The number of edges in the rich club graph. |

`rich_club_norm` - a data table with columns:

| | |
|------------------------|--|
| <code>k</code> | Sequence of degrees |
| <code>rand</code> | Rich-club coefficients for the random graphs |
| <code>orig</code> | Rich-club coefficients for the original graph. |
| <code>norm</code> | Normalized rich-club coefficients. |
| <code>p</code> | The P-values based on the distribution of rich-club coefficients from the random graphs. |
| <code>p.fdr</code> | The FDR-adjusted P-values |
| <code>density</code> | The observed graph's density |
| <code>threshold</code> | |
| <code>Group</code> | |
| <code>name</code> | |

`rich_core` - a data frame with columns:

| | |
|------------------------|--|
| <code>density</code> | The density of the graph. |
| <code>rank</code> | The rank of the boundary for the rich core. |
| <code>k.r</code> | The degree of the vertex at the boundary. |
| <code>core.size</code> | The size of the core relative to the graph size. |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Zhou S., Mondragon R.J. (2004) *The rich-club phenomenon in the internet topology*. IEEE Comm Lett, 8:180-182.

Opsahl T., Colizza V., Panzarasa P., Ramasco J.J. (2008) *Prominence and control: the weighted rich-club effect*. Physical Review Letters, 101.16:168702.

Colizza V., Flammini A., Serrano M.A., Vespignani A. (2006) *Detecting rich-club ordering in complex networks*. Nature Physics, 2:110-115.

Ma A & Mondragon R.J. (2015) *Rich-cores in networks*. PLoS One, 10(3): e0119678. doi: 10.1371/journal.pone.0119678

See Also

Other Rich-club functions: [plot_rich_norm](#), [rich_club_attr](#)

Other Random graph functions: [RandomGraphs](#), [analysis_random_graphs](#)

rich_club_attr *Assign graph attributes based on rich-club analysis*

Description

This function will assign vertex- and edge-level attributes based on the results of a *rich-club* analysis, based on a range of vertex degrees in which the rich-club coefficient was determined to be significantly greater than that of a set of random graphs (see [rich_club_norm](#)).

Usage

```
rich_club_attr(g, deg.range = NULL, adj.vsize = FALSE)
```

Arguments

| | |
|------------------------|---|
| <code>g</code> | An igraph graph object |
| <code>deg.range</code> | Integer vector of the range of degrees indicating inclusion in the rich-club; if the default <i>NULL</i> , it will be from 1 to the maximum degree in the graph |
| <code>adj.vsize</code> | Logical indicating whether to adjust vertex size proportional to degree (default: <i>FALSE</i>) |

Details

Vertices which are in the rich club will be assigned an attribute `rich`, taking on a binary value. Their colors (attribute `color.rich`) will be either *red* or *gray*. Their sizes (attribute `size.rich`) will either be 10 or will be proportional to their degree.

Edge attribute `type.rich` takes on three values: *rich-club* (if it connects two rich-club vertices), *feeder* (if it connects a rich- to a non-rich-club vertex), and *local* (if it connects two non-rich-club vertices). They will also be given a `color.rich` attribute (either *red*, *orange*, or *green*). Edge sizes (`size.rich`) will be largest for *rich-club* connections, then smaller for *feeder*, and smallest for *local*.

Value

An igraph graph object with additional attributes:

| | |
|-------------------------|--|
| <code>rich</code> | Binary indicating membership in the rich-club |
| <code>type.rich</code> | Edge attribute indicating the type of connection |
| <code>color.rich</code> | Edge and vertex attributes |
| <code>size.rich</code> | Edge and vertex attributes |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

Other Rich-club functions: [RichClub](#), [plot_rich_norm](#)

Examples

```
## Not run:
g <- rich_club_attrs(g, rich.dt[density == densities[N] & p.fdr < .01,
                             range(k)])

## End(Not run)
```

robustness

Analysis of network robustness

Description

This function performs a "targeted attack" of a graph or a "random failure" analysis, calculating the size of the largest component after edge or vertex removal.

Usage

```
robustness(g, type = c("vertex", "edge"), measure = c("btwn.cent", "degree",
              "random"), N = 1000)
```


Arguments

| | |
|---------|---|
| g | An igraph graph object |
| type | Character string; either 'vertex' or 'edge' removals (default: vertex) |
| measure | Character string; sort by either 'btwn.cent' or 'degree', or choose 'random' (default: btwn.cent) |
| N | Integer; the number of iterations if <i>random</i> is chosen (default: 1e3) |

Details

In a targeted attack, it will sort the vertices by either degree or betweenness centrality (or sort edges by betweenness), and successively remove the top vertices/edges. Then it calculates the size of the largest component.

In a random failure analysis, vertices/edges are removed in a random order.

Value

Numeric vector representing the ratio of maximal component size after each removal to the observed graph's maximal component size

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Albert R., Jeong H., Barabasi A. (2000) *Error and attack tolerance of complex networks*. Nature, 406:378-381.

set_brainGraph_attr *Set graph, vertex, and edge attributes common in MRI analyses*

Description

This function sets a number of graph, vertex, and edge attributes for a given igraph graph object. These are all measures that are common in MRI analyses of brain networks.

Usage

```
set_brainGraph_attr(g, atlas = NULL, rand = FALSE, use.parallel = TRUE,
  A = NULL, xfm.type = c("1/w", "-log(w)", "1-w"), ...)
```

Arguments

| | |
|---------------------------|---|
| <code>g</code> | An igraph graph object |
| <code>atlas</code> | Character vector indicating which atlas was used (default: NULL) |
| <code>rand</code> | Logical indicating if the graph is random or not (default: FALSE) |
| <code>use.parallel</code> | Logical indicating whether or not to use <i>foreach</i> (default: TRUE) |
| <code>A</code> | Numeric matrix; the (weighted) adjacency matrix, which can be used for faster calculation of local efficiency (default: NULL) |
| <code>xfm.type</code> | Character string indicating how to transform edge weights (default: 1/w [reciprocal]) |
| <code>...</code> | Other arguments passed to make_brainGraph |

Details

`xfm.type` allows you to choose from 3 options for transforming edge weights when calculating distance-based metrics (e.g., shortest paths). There is no "best-practice" for choosing one over the other, but the reciprocal is probably most common.

- 1/w: reciprocal (default)
- $-\log(w)$: the negative (natural) logarithm
- 1-w: subtract weights from 1

Value

`g` An igraph graph object with the following attributes:

| | |
|--------------|--|
| Graph-level | Density, connected component sizes, diameter, \# of triangles, transitivity, average path length, assortativity, global & local efficiency, modularity, vulnerability, hub score, rich-club coefficient, \# of hubs, edge asymmetry, and modality |
| Vertex-level | Degree, strength; betweenness, eigenvector, and leverage centralities; hubs; transitivity (local); k-core, s-core; local & nodal efficiency; color (community, lobe, component); membership (community, lobe, component); gateway and participation coefficients, within-module degree z-score; vulnerability; and coordinates (x, y, and z) |
| Edge-level | Color (community, lobe, component), edge betweenness, Euclidean distance (in mm), weight (if weighted) |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[components](#), [diameter](#), [clique_num](#), [centr_betw](#), [part_coeff](#), [edge_betweenness](#), [centr_eigen](#), [gateway_coeff](#), [transitivity](#), [mean_distance](#), [assortativity_degree](#), [efficiency](#), [assortativity_nominal](#), [coreness](#), [cluster_louvain](#), [set_edge_color](#), [rich_club_coeff](#), [s_core](#), [centr_lev](#), [within_module_deg_z_score](#), [edge_spatial_dist](#), [vulnerability](#), [edge_asymmetry](#), [graph.knn](#), [vertex_spatial_dist](#)

| | |
|-------------|--|
| small.world | <i>Calculate graph small-worldness</i> |
|-------------|--|

Description

This function will calculate the characteristic path length and clustering coefficient, which are used to calculate small-worldness.

Usage

```
small.world(g, rand)
```

Arguments

| | |
|------|---|
| g | The graph (or list of graphs) of interest |
| rand | List of (lists of) equivalent random graphs (output from sim.rand.graph.par) |

Value

A data frame with the following components:

| | |
|---------|--|
| density | The range of density thresholds used. |
| N | The number of random graphs that were generated. |
| Lp | The characteristic path length. |
| Cp | The clustering coefficient. |
| Lp.rand | The mean characteristic path length of the random graphs with the same degree distribution as g. |
| Cp.rand | The mean clustering coefficient of the random graphs with the same degree distribution as g. |
| Lp.norm | The normalized characteristic path length. |
| Cp.norm | The normalized clustering coefficient. |
| sigma | The small-world measure of the graph. |

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Watts D.J., Strogatz S.H. (1998) *Collective dynamics of 'small-world' networks*. Nature, 393:440-442.

summary.bg_GLM *Print a summary from brainGraph_GLM analysis*

Description

summary prints the results from a `brainGraph_GLM` analysis. It will only print results for which $p < \alpha$; you may change this to the FDR-adjusted or permutation p-values via the function argument `p.sig`.

Usage

```
## S3 method for class 'bg_GLM'
summary(object, p.sig = c("p", "p.fdr", "p.perm"),
        contrast = NULL, digits = max(3L, getOption("digits") - 2L),
        print.head = TRUE, ...)
```

Arguments

| | |
|-------------------------|--|
| <code>object</code> | A <code>bg_GLM</code> object |
| <code>p.sig</code> | Character string specifying which p-value to use for displaying significant results (default: p) |
| <code>contrast</code> | Integer specifying the contrast to summarize; defaults to showing results for all contrasts |
| <code>digits</code> | Integer specifying the number of digits to display for p-values |
| <code>print.head</code> | Logical indicating whether or not to print only the first and last 5 rows of the statistics tables (default: TRUE) |
| <code>...</code> | Unused |

summary.bg_mediate *Print a summary from a brainGraph mediation analysis*

Description

Print a summary from a brainGraph mediation analysis

Usage

```
## S3 method for class 'bg_mediate'
summary(object, mediate = FALSE, region = NULL,
        digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

| | |
|---------|--|
| object | A bg_mediate object |
| mediate | Logical indicating whether or not to use the summary method from mediate (default: FALSE). If TRUE, only a single region can be printed. |
| region | Character string specifying which region's results to summarize; only relevant if level='vertex' (default: NULL) |
| digits | Integer specifying the number of digits to display for p-values |
| ... | Unused |

summary.brainGraph *Print a summary of a brainGraph object*

Description

Print a summary of a brainGraph object

Usage

```
## S3 method for class 'brainGraph'
summary(object, print.attrs = c("all", "none"), ...)
```

Arguments

| | |
|-------------|---|
| object | A brainGraph object |
| print.attrs | Character string indicating whether or not to list the object's attributes (default: all) |
| ... | Unused |

summary.brainGraph_permute
Print a summary from a permutation analysis

Description

Print a summary from a permutation analysis

Usage

```
## S3 method for class 'brainGraph_permute'
summary(object, measure = NULL,
        alternative = c("two.sided", "less", "greater"), alpha = 0.05,
        p.sig = c("p", "p.fdr"), ...)
```

Arguments

| | |
|-------------|--|
| object | A brainGraph_permute object (output by brainGraph_permute). |
| measure | Character string of the measure to focus on (if level='graph') |
| alternative | Character string, whether to do a two- or one-sided test (default: 'two.sided') |
| alpha | Numeric; the significance level (default: 0.05) |
| p.sig | Character string specifying which p-value to use for displaying significant results (default: p) |
| ... | Unused |

| | |
|-----------------|----------------------------------|
| symmetrize_mats | <i>Create a symmetric matrix</i> |
|-----------------|----------------------------------|

Description

symmetrize_mats will symmetrize a numeric matrix by assigning the off-diagonal elements values of either the max, min, or average of $\{A(i,j), A(j,i)\}$. The default is max because that is the default for [graph_from_adjacency_matrix](#).

symmetrize_array is a convenience function which applies [symmetrize_mats](#) along the 3rd dimension of an array.

Usage

```
symmetrize_mats(A, symm.by = c("max", "min", "avg"))
symmetrize_array(A, ...)
```

Arguments

| | |
|---------|--|
| A | Numeric matrix |
| symm.by | Character string; how to create symmetric off-diagonal elements (default: max) |
| ... | Arguments passed to symmetrize_mats |

Value

Either a single symmetrized matrix, or an (3D) array

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

See Also

[graph_from_adjacency_matrix](#)

Other Matrix functions: [apply_thresholds](#), [cor.diff.test](#), [create_mats](#)

`s_core`*Calculate the s-core of a network*

Description

Calculates the *s-core* decomposition of a network. This is analogous to the *k-core* decomposition, but takes into account the *strength* of vertices (i.e., in weighted networks). If an unweighted network is supplied, then the output of the function [coreness](#) is returned.

Usage

```
s_core(g, W = NULL)
```

Arguments

| | |
|----------------|--|
| <code>g</code> | The igraph graph object of interest |
| <code>W</code> | Numeric matrix of edge weights (default: NULL) |

Details

The *s-core* consists of all vertices i with $s_i > s$, where s is some threshold value. The s_0 core is the entire network, and the threshold value of the s_n core is

$$s_{n-1} = \min_i s_i$$

for all vertices i in the s_{n-1} core.

Note that in networks with a wide distribution of vertex strengths, in which there are almost as many unique values as there are vertices, then several separate cores will have a single vertex. See the reference provided below.

Value

Integer vector of the vertices' *s-core* membership

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Eidsaa M & Almaas E. (2013) *s-core network decomposition: a generalization of k-core analysis to weighted networks*. Physical Review E, 88:062819.

See Also

[coreness](#)

| | |
|-------------|--|
| VertexRoles | <i>Gateway coefficient, participation coefficient, and within-mod degree z-score</i> |
|-------------|--|

Description

`gateway_coeff` calculates the gateway coefficient of each vertex, based on community membership.

`part_coeff` calculates the participation coefficient of each vertex, based on community membership.

`within_module_deg_z_score` is a measure of the connectivity from a given vertex to other vertices in its module/community.

Usage

```
gateway_coeff(g, memb, centr = c("btwn.cent", "degree", "strength"))
```

```
part_coeff(g, memb)
```

```
within_module_deg_z_score(g, memb)
```

Arguments

| | |
|--------------------|--|
| <code>g</code> | An igraph graph object |
| <code>memb</code> | A numeric vector of membership indices of each vertex |
| <code>centr</code> | Character string; the type of centrality to use in calculating GC (default: <code>btwn.cent</code>) |

Details

The gateway coefficient G_i of vertex i is:

$$G_i = 1 - \sum_{S=1}^{N_M} \left(\frac{\kappa_{iS}}{\kappa_i} \right)^2 (g_{iS})^2$$

where κ_{iS} is the number of edges from vertex i to vertices in module S , and κ_i is the degree of vertex i . N_M equals the number of modules. g_{iS} is a weight, defined as:

$$g_{iS} = 1 - \kappa_{iS} \bar{c}_{iS}$$

where

$$\bar{\kappa}_{iS} = \frac{\kappa_{iS}}{\sum_j \kappa_{jS}}$$

for all nodes j in node i 's module, and

$$\bar{c}_{iS} = c_{iS} / \max(c_n)$$

The participation coefficient P_i of vertex i is:

$$P_i = 1 - \sum_{s=1}^{N_M} \left(\frac{\kappa_{is}}{\kappa_i} \right)^2$$

where κ_{is} is the number of edges from vertex i to vertices in module s , and κ_s is the degree of vertex i . N_M equals the number of modules.

As discussed in Guimera et al., $P_i = 0$ if vertex i is connected only to vertices in the same module, and $P_i = 1$ if vertex i is equally connected to all other modules.

The within-module degree z-score is:

$$z_i = \frac{\kappa_i - \bar{\kappa}_{s_i}}{\sigma_{\kappa_{s_i}}}$$

where κ_i is the number of edges from vertex i to vertices in the same module s_i , $\bar{\kappa}_{s_i}$ is the average of κ over all vertices in s_i , and $\sigma_{\kappa_{s_i}}$ is the standard deviation.

Value

A vector of the participation coefficients, within-module degree z-scores, or gateway coefficients for each vertex of the graph.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Vargas E.R. & Wahl L.M. (2014) The gateway coefficient: a novel metric for identifying critical connections in modular networks. *Eur Phys J B*, 87:161-170.

Guimera, R. and Amaral, L.A.N. (2005) Cartography of complex networks: modules and universal roles, *Journal of Statistical Mechanics: Theory and Experiment*, 02, P02001.

vulnerability

Calculate graph vulnerability

Description

This function calculates the *vulnerability* of the vertices of a graph. Here, vulnerability is considered to be the proportional drop in global efficiency when a given node is removed from the graph. The vulnerability of the graph is considered the maximum across all vertices.

Usage

`vulnerability(g, use.parallel = TRUE, weighted = FALSE)`

Arguments

| | |
|---------------------------|--|
| <code>g</code> | An igraph graph object |
| <code>use.parallel</code> | Logical indicating whether or not to use <i>foreach</i> (default: TRUE) |
| <code>weighted</code> | Logical indicating whether weighted efficiency should be calculated (default: FALSE) |

Value

A numeric vector of length equal to the vertex count of *g*

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Latora V., Marchiori M. (2005) *Variability and protection of infrastructure networks*. Physical Review E, 71:015103.

See Also

[efficiency](#)

| | |
|----------------|--|
| write_brainnet | <i>Write files to be used for visualization with BrainNet Viewer</i> |
|----------------|--|

Description

This function will write the *.node* and *.edge* files necessary for visualization with the BrainNet Viewer software (see Reference below).

Usage

```
write_brainnet(g, node.color = "none", node.size = "constant",
              edge.wt = NULL, file.prefix = "")
```

Arguments

| | |
|--------------------------|---|
| <code>g</code> | The igraph graph object of interest |
| <code>node.color</code> | Character string indicating whether to color the vertices or not (default: 'none') |
| <code>node.size</code> | Character string indicating what size the vertices should be; can be any vertex-level attribute (default: 'constant') |
| <code>edge.wt</code> | Character string indicating the edge attribute to use to return a weighted adjacency matrix (default: NULL) |
| <code>file.prefix</code> | Character string for the basename of the <i>.node</i> and <i>.edge</i> files that are written |

Details

For the *.node* file, there are 6 columns:

- *Column 1*: x-coordinates
- *Column 2*: y-coordinates
- *Column 3*: z-coordinates
- *Column 4*: Vertex color
- *Column 5*: Vertex size
- *Column 6*: Vertex label

The *.edge* file is the graph's associated adjacency matrix; a weighted adjacency matrix can be returned by using the *edge.wt* argument.

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

References

Xia M, Wang J, He Y (2013). *BrainNet Viewer: a network visualization tool for human brain connectomics*. PLoS One, 8(7):e68910.

Examples

```
## Not run:
write_brainnet(g, node.color='community', node.size='degree',
  edge.wt='t.stat')

## End(Not run)
```

xfm.weights

Transform edge weights

Description

For distance-based measures, it is important to transform the edge weights so that the *strongest* connections are re-mapped to having the *lowest* weights. Then you may calculate e.g., the *shortest path length* which will include the strongest connections.

Usage

```
xfm.weights(g, xfm.type = c("1/w", "-log(w)", "1-w"), invert = FALSE)
```

Arguments

| | |
|-----------------------|---|
| <code>g</code> | An igraph graph object |
| <code>xfm.type</code> | Character string specifying how to transform the weights (default: 1/w) |
| <code>invert</code> | Logical indicating whether or not to invert the transformation (default: FALSE) |

Details

There are 3 options for the type of transform to apply:

1. $1/w$: calculate the inverse
2. $-\log(w)$: calculate the negative (natural) logarithm
3. $1-w$: subtract each weight from 1

To transform the weights back to original values, specify `invert=TRUE`.

Value

An `igraph` graph object with transformed edge weights and a graph attribute, `xfm.type`, of the type of transform

Author(s)

Christopher G. Watson, <cgwatson@bu.edu>

Index

*Topic **datasets**

- aal116, 3
- aal2.120, 4
- aal2.94, 5
- aal90, 6
- brainsuite, 20
- craddock200, 29
- destrieux, 33
- destrieux.scgm, 34
- dk, 35
- dk.scgm, 36
- dkt, 37
- dkt.scgm, 38
- dosenbach160, 39
- hoa112, 46
- lpba40, 48
- [.brainGraph_resids (get.resid), 42

- aal116, 3
- aal2.120, 4
- aal2.94, 5
- aal90, 6
- analysis_random_graphs, 7, 77, 79
- aop (IndividualContributions), 46
- apply_thresholds, 8, 26, 31, 86
- as_data_frame, 32
- assortativity_degree, 82
- assortativity_nominal, 82

- bg_to_mediate, 9, 9
- boot, 10, 11
- boot.ci, 62
- brainGraph_boot, 10, 10, 13, 16, 18, 20, 27, 43, 47, 57, 59, 61, 62, 67, 75
- brainGraph_GLM, 11, 11, 14, 18, 20, 44, 45, 47, 52, 55–57, 59, 62, 84
- brainGraph_GLM_design, 12, 13, 13, 17, 43, 45, 55, 57–59, 66
- brainGraph_GLM_fit_f (GLMfit), 44
- brainGraph_GLM_fit_t, 59

- brainGraph_GLM_fit_t (GLMfit), 44
- brainGraph_init, 11, 14, 20, 27, 42, 43, 47, 67, 75
- brainGraph_mediate, 9, 11, 13, 16, 20, 47, 54, 57, 59
- brainGraph_permute, 11, 13, 16, 18, 19, 27, 43, 47, 57, 59, 66, 67, 75, 86
- brainsuite, 20

- centr_betw, 82
- centr_betw_comm, 21, 23
- centr_eigen, 82
- centr_lev, 22, 22, 82
- clique_num, 82
- cluster_louvain, 82
- coeff_var, 23
- communicability, 23
- components, 82
- contract, 25
- contract_brainGraph, 24
- cor.diff.test, 9, 25, 31, 86
- coreness, 82, 87
- corr.matrix, 11, 16, 20, 26, 43, 47, 67, 75
- count_homologous (CountEdges), 28
- count_interlobar (CountEdges), 28
- CountEdges, 28
- craddock200, 29
- create_mats, 8, 9, 26, 30, 86

- data.table, 72
- DataTables, 32
- destrieux, 33
- destrieux.scgm, 34
- diameter, 82
- dk, 35
- dk.scgm, 36
- dkt, 37
- dkt.scgm, 38
- dosenbach160, 39

- edge.betweenness, [82](#)
- edge_asymmetry, [40](#), [82](#)
- edge_spatial_dist, [82](#)
- edge_spatial_dist (GraphDistances), [45](#)
- efficiency, [41](#), [82](#), [90](#)
- Extract.brainGraph_resids (get.resid), [42](#)
- foreach, [76](#)
- gateway_coeff, [82](#)
- gateway_coeff (VertexRoles), [88](#)
- geom_histogram, [75](#)
- geom_ribbon, [62](#)
- geom_tile, [71](#)
- geom_vline, [75](#)
- get.resid, [10](#), [11](#), [16](#), [19](#), [20](#), [26](#), [27](#), [42](#), [47](#), [66](#), [67](#), [75](#)
- ggplot, [59](#), [66](#), [68](#), [72](#), [73](#)
- GLMfit, [13](#), [14](#), [44](#), [57](#)
- graph.knn, [82](#)
- graph_attr, [32](#)
- graph_attr_dt (DataTables), [32](#)
- graph_attr_names, [32](#)
- graph_from_adjacency_matrix, [86](#)
- GraphDistances, [45](#)
- hoa112, [46](#)
- igraph.plotting, [61–64](#)
- IndividualContributions, [11](#), [13](#), [16](#), [18](#), [20](#), [27](#), [43](#), [46](#), [57](#), [59](#), [67](#), [75](#)
- keeping_degseq, [77](#)
- loo (IndividualContributions), [46](#)
- lpba40, [48](#)
- make_brainGraph, [49](#), [51](#), [52](#), [54](#), [82](#)
- make_ego_brainGraph, [50](#), [50](#), [52](#), [54](#)
- make_ego_graph, [51](#)
- make_empty_brainGraph, [50](#), [51](#), [51](#), [52](#), [54](#)
- make_empty_graph, [51](#), [52](#)
- make_glm_brainGraph, [50–52](#), [52](#), [54](#), [62](#), [64](#)
- make_intersection_brainGraph, [53](#)
- make_mEDIATE_brainGraph, [50–52](#), [53](#), [54](#), [63](#)
- make_nbs_brainGraph, [50–52](#), [54](#), [54](#), [65](#)
- mean_distance, [82](#)
- mediate, [9](#), [17](#), [85](#)
- mtpc, [11](#), [13](#), [14](#), [18](#), [20](#), [45](#), [47](#), [52](#), [55](#), [59](#), [64](#)
- NBS, [11](#), [13](#), [18](#), [20](#), [47](#), [54](#), [57](#), [57](#), [64](#)
- par, [62–65](#)
- part_coeff, [82](#)
- part_coeff (VertexRoles), [88](#)
- plot.bg_GLM, [59](#)
- plot.brainGraph, [60](#), [63–65](#), [68–70](#)
- plot.brainGraph_boot, [61](#)
- plot.brainGraph_GLM, [61](#), [62](#), [63–65](#), [68–70](#)
- plot.brainGraph_mEDIATE, [61](#), [63](#), [63](#), [64](#), [65](#), [68–70](#)
- plot.brainGraph_mtpc, [61](#), [63](#), [64](#), [65](#), [68–70](#)
- plot.brainGraph_NBS, [61](#), [63](#), [64](#), [64](#), [68–70](#)
- plot.brainGraph_permute, [65](#)
- plot.brainGraph_resids, [11](#), [16](#), [20](#), [27](#), [43](#), [47](#), [66](#), [75](#)
- plot.igraph, [61–64](#)
- plot.lm, [59](#), [60](#)
- plot.mEDIATE, [9](#)
- plot.mtpc, [67](#)
- plot_brainGraph, [69](#), [70](#)
- plot_brainGraph (plot.brainGraph), [60](#)
- plot_brainGraph_gui, [61](#), [63–65](#), [68](#), [69](#), [70](#)
- plot_brainGraph_list, [61](#), [63–65](#), [68](#), [69](#), [70](#)
- plot_brainGraph_multi, [61](#), [63–65](#), [68](#), [69](#), [70](#)
- plot_corr_mat, [71](#)
- plot_global, [72](#)
- plot_rich_norm, [73](#), [79](#), [80](#)
- plot_vertex_measures, [74](#)
- plot_volumetric, [11](#), [16](#), [20](#), [27](#), [43](#), [47](#), [67](#), [75](#)
- qqnorm, [67](#)
- RandomGraphs, [8](#), [76](#), [79](#)
- rcorr, [27](#)
- rewire, [77](#)
- rich_club_attrs, [73](#), [79](#), [79](#)
- rich_club_coeff, [78](#), [82](#)
- rich_club_coeff (RichClub), [77](#)
- rich_club_norm, [78](#), [79](#)
- rich_club_norm (RichClub), [77](#)
- rich_core, [73](#), [78](#)
- rich_core (RichClub), [77](#)

RichClub, [8](#), [73](#), [77](#), [77](#), [80](#)
robustness, [80](#)
rstudent, [43](#)

s_core, [82](#), [87](#)
sample_degseq, [76](#), [77](#)
set_brainGraph_attr, [68](#), [76](#), [81](#)
set_edge_color, [82](#)
sim.rand.graph.clust, [76](#)
sim.rand.graph.clust (RandomGraphs), [76](#)
sim.rand.graph.par, [7](#), [78](#), [83](#)
sim.rand.graph.par (RandomGraphs), [76](#)
small.world, [8](#), [83](#)
summary.bg_GLM, [84](#)
summary.bg_mediate, [84](#)
summary.brainGraph, [85](#)
summary.brainGraph_boot
 (brainGraph_boot), [10](#)
summary.brainGraph_permute, [85](#)
summary.brainGraph_resids, [43](#)
summary.brainGraph_resids (get.resid),
 [42](#)
summary.mediate, [9](#)
summary.mtpc (mtpc), [55](#)
summary.NBS (NBS), [57](#)
symmetrize_array (symmetrize_mats), [86](#)
symmetrize_mats, [9](#), [26](#), [30](#), [31](#), [86](#), [86](#)

transitivity, [77](#), [82](#)

vertex_attr, [32](#)
vertex_attr_dt (DataTables), [32](#)
vertex_attr_names, [32](#)
vertex_spatial_dist, [82](#)
vertex_spatial_dist (GraphDistances), [45](#)
VertexRoles, [88](#)
vulnerability, [82](#), [89](#)

within_module_deg_z_score, [82](#)
within_module_deg_z_score
 (VertexRoles), [88](#)
write_brainnet, [90](#)

xfm.weights, [91](#)