Package 'brainGraph'

December 24, 2015

Type Package

Version 0.55.0

Date 2015-12-23

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 is best suited to data from a Freesurfer analysis (cortical thickness, volumes, local gyrification index, surface area), but also works with e.g., tractography data from FSL. It contains a graphical user interface for graph visualization and data exploration.

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), RGtk2, cairoDevice

Imports abind, ade4, boot, data.table, foreach, ggplot2, Hmisc, methods, oro.nifti, parallel, plyr, scales

License GPL-3

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2015-12-24 08:19:03

R topics documented:

`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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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.

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)
```

`aal90` *Coordinates for data from the AAl90 atlas*

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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.

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)
```

---

aop *"Add-one-patient" approach to estimate individual network contribution*

Description

Calculates the individual contribution to group network data for each subject in each group using a "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 compared to the original correlation matrix using the Mantel test.

Usage

```
aop(resids, index, corr.mat, level = c("global", "regional"))
```

Arguments

`resids`	Data table of model residuals
`index`	Integer; the row number (in the residuals data table) of the subject to be added
`corr.mat`	Correlation matrix of the control group
`level`	Character string; the level at which you want to calculate contributions (either *global* or *regional*)

Value

A `data.table` with columns for

`Study.ID`	Subject identifier
`Group`	Group membership
`IC`	The value of the individual contribution

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

Examples

```
## Not run:
IC <- adply(which(resids.all[, Group == groups[2]]), .margins=1, function(x)
            aop(resids.all, x, corrs[[1]][[1]]$R),
            .parallel=T, .id=NULL)

## End(Not run)
```

`assign_lobes` *Give vertices in a graph a* lobe *attribute.*

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 an attribute *circle.layout* for plotting circular graphs.

Usage

```
assign_lobes(g, rand = FALSE)
```

Arguments

`g`	An *igraph* graph object.
`rand`	A character string indicating whether this function is being run for a random graph or a "graph of interest" (default: FALSE).

Details

The input graph `g` *must* have a graph attribute named `atlas`.

Value

An *igraph* graph object with additional vertex attributes:

`lobe`	Character string indicating the lobe
`lobe.hemi`	Integer vector indicating the lobe and hemisphere
`circle.layout`	Integer vector for ordering the vertices for circle plots

Author(s)

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

boot_global *Bootstrapping for global graph measures*

Description

This function performs bootstrapping to get group standard error estimates of a global graph measure (e.g. modularity). It will output a list containing the boot objects and a data.table with standard errors and 95% confidence intervals at each density for each group.

Usage

```
boot_global(densities, resids, R = 1000, measure = c("mod", "E.global",
  "Cp", "Lp", "assortativity"))
```

Arguments

densities	A vector of graph densities to loop through
resids	A data.table of the residuals (from get.resid)
R	The number of bootstrap replicates (default: 1e3)
measure	Character string of the measure to test (default: 'mod')

Details

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

Value

A list with two elements:

g	A list of boot objects (one for each group)
dt	A data table with length *# densities * # groups*

Author(s)

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

See Also

boot, boot.ci, permute.group

Examples

```
## Not run:
boot.E.global <- boot_global(densities, m$resids, 1e3, 'E.global')

## End(Not run)
```

`brainGraph_init` *Initialize variables for further use in brainGraph*

Description

This function initializes some variables that are important for further analysis with the `brainGraph` package. This mostly involves loading CSV files (of covariates/demographics, cortical thickness/volumes, etc.) and returning them as data tables.

Usage

```
brainGraph_init(atlas = c("aal116", "aal90", "brainsuite", "destrieux", "dk",
  "dk.scgm", "dkt", "dkt.scgm", "hoa112", "lpba40"), densities, datadir,
  modality = c("thickness", "volume", "lgi", "area"), use.mean = FALSE,
  exclude.subs = NULL)
```

Arguments

`atlas`	A character string indicating which brain atlas you are using
`densities`	A numeric vector of the graph densities you would like to investigate
`datadir`	A character string; the filesystem location of your input files
`modality`	A character string indicating the volumetric MRI modality/measure you are using to create the graphs ('thickness', 'volume', 'lgi', or 'area')
`use.mean`	A logical indicating whether or not you would like to calculate the mean hemispheric volumetric measure (for later use in linear models) (default: FALSE)
`exclude.subs`	(optional) A character vector of the Study ID's of subjects who are to be excluded from the analysis

Details

The file containing covariates should be names `covars.csv`. The files containing volumetric data should include hemisphere, atlas, and modality, e.g. `lh_dkt_thickness.csv`. 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`	A character string of the brain atlas name
`densities`	A numeric vector of the graph densities
`modality`	A character string of the modality you chose
`kNumDensities`	An integer indicating the number of densities
`covars`	A `data.table` of covariates
`groups`	A character vector of subject group names

kNumGroups	An integer indicating the number of groups
kNumVertices	An integer; the number of vertices in the graphs
lhrh	A data.table of left- and right-hemispheric volumetric data
all.dat	A merged data.table of covars and lhrh
all.dat.tidy	A 'tidied' version of all.dat

Author(s)

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

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'), use.mean=FALSE)

## 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 10 variables.

name a character vector of region names

x a numeric vector of x-coordinates (internal to brainGraph)

y a numeric vector of y-coordinates (internal to brainGraph)

z a numeric vector of z-coordinates (internal to brainGraph)

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_lev` *Calculate a vertex's leverage centrality*

Description

This function calculates the leverage centrality of each vertex in a graph.

Usage

```
centr_lev(g, .parallel = TRUE)
```

Arguments

`g`	The igraph graph object
`.parallel`	Logical indicating whether or not to use *foreach* (default: TRUE)

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.

---

check.resid *Check model residuals for each brain region*

Description

This function checks the model residuals for each brain region in the analysis. It simply does a qqplot of the studentized residuals (but uses ggplot2 functions).

Usage

```
check.resid(resids, cols = FALSE)
```

Arguments

resids	Data table of model residuals for all brain regions
cols	Logical indicating whether to color by group (default: FALSE)

Value

A list of ggplot objects

Author(s)

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

See Also

qqnorm

Examples

```
## Not run:
p.resids <- check.resid(resids.all)
lapply(p.resids, function(x) {dev.new(); print(x)})

## End(Not run)
```

choose.edges *Select edges for re-wiring.*

Description

This function selects edges to be re-wired when simulating random graphs. It is based on the algorithm by Bansal et al. (2009), BMC Bioinformatics.

Usage

```
choose.edges(g)
```

Arguments

g	The random graph that has been generated

Value

A data frame with four elements; two edges will be removed and two edges will be added between the four vertices.

References

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

color.edges *Color graph edges*

Description

This function takes the community membership of a given graph, and assigns to the edges a specific color (the same as the vertex membership colors). Edges that connect vertices of two different groups are colored gray. Also works for the major lobes of the brain, plus insula, subcortical gray matter, cingulate, limbic lobe (if included in the specific brain atlas).

Usage

```
color.edges(g, memb)
```

Arguments

g	The graph to get its edges colored
memb	An integer vector indicating vertex group membership

Value

A character vector of colors for each edge in the graph

color.vertices *Color graph vertices*

Description

This function takes an integer vector (representing membership of a community or component) and creates a character vector of colors for each community/module, component, etc. This only assigns a color to groups with at least 2 members; isolated vertices will be colored 'gray'.

Usage

```
color.vertices(memb)
```

Arguments

memb	An integer vector representing membership of e.g. a community

Value

A character vector with length equal to the number of communities, lobes, components, etc.

corr.matrix *Calculate correlation matrix and threshold*

Description

This function does a column-by-column correlation of a given data frame, and will threshold the matrix based on a given density; e.g. 0.1 if you want to keep only the 10% strongest correlations.

Usage

```
corr.matrix(dat, thresh = NULL, density = 0.1, exclusions = NULL, ...)
```

Arguments

dat	Matrix or data frame of the columns to correlate
thresh	Absolute correlation value to threshold by
density	Keeps the top $X\%$ of correlations
exclusions	Vector of indices (columns) to exclude (optional)
...	Other arguments to be passed to rcorr

Details

If you wish to exclude regions from your analysis, you can give the indices of their columns. Also returns the p-values. Essentially a wrapper for rcorr, with some added functionality to work with this type of data more easily. By default, the Pearson correlation coefficients are calculated, but can return Spearman by passing an additional argument.

Value

A list with the following components:

R	Correlation coefficients (default: Pearson).
P	Associated p-values.
r.thresh	Binary matrix indicating correlations that are above a certain threshold.
threshold	The threshold used.

See Also

rcorr

---

count_homologous *Count number of edges between homologous regions of a brain graph*

Description

This function will count the number of edges between homologous regions in a brain graph (e.g. between L and R superior frontal).

Usage

```
count_homologous(g)
```

Arguments

g	An igraph graph object

Value

A named vector of the edge ID's connecting homologous regions

Author(s)

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

count_interlobar *Count number of inter-lobar connections from a given major lobe*

Description

This function will count the number of edges between all vertices in one major lobe (e.g. Frontal) and all other major lobes.

Usage

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

Arguments

g	The igraph graph object
lobe	A character string indicating the lobe to count from (uppercase)

Value

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(g1[[N]], 'Frontal')

## End(Not run)
```

destrieux *Coordinates for data from the Destrieux atlas*

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 11 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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`

Source

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.

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)
```

---

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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.

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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.

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

Source

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

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)
```

---

dti_create_mats *Create connection matrices for tractography analysis*

Description

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

Usage

```
dti_create_mats(A.files, Nv, divisor = c("none", "waytotal", "size",
  "rowSums"), div.files, mat.thresh = 0, sub.thresh = 0.5, groups = NULL,
  P = 5000)
```

Arguments

A.files	A character vector of the filenames with connection matrices
Nv	Integer representing the number of vertices (rows/columns)
divisor	A character string indicating how to normalize the connection matrices; either 'none', 'waytotal', 'size', or 'rowSums'
div.files	A character vector of the filenames with the data to normalize by (e.g. a list of *waytotal* files)
mat.thresh	A numeric (vector) for thresholding connection matrices
sub.thresh	A numeric (between 0 and 1) for thresholding by subject numbers
groups	A character vector of group names
P	Number of samples generated using FSL (default: 5000)

Details

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:

A	A 3-d array of the raw connection matrices
A.norm	A 3-d array of the normalized connection matrices
A.bin	A 3-d array of binarized connection matrices
A.bin.sums	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 `mat.thresh`
A.inds	A list of arrays of binarized connection matrices, containing 1 if that entry is to be included
A.norm.sub	A list of 3-d arrays of the normalized connection matrices for all given thresholds
A.norm.mean	A list of 2-d arrays of the normalized connection matrices averaged for each group

Author(s)

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

Examples

```
## Not run:
thresholds <- seq(from=0.001, to=0.01, by=0.001)
my.mats <- dti_create_mats(f.A, Nv, 'waytotal', f.way, thresholds,
  sub.thresh=0.5, groups)
my.mats <- dti_create_mats(f.A, Nv, 'size', f.size, thresholds,
  sub.thresh=0.5, groups, P=5000)

## End(Not run)
```

`edge_asymmetry` *Calculate an asymmetry index based on edge counts*

Description

This function will calculate an asymmetry index that is a measure of whether or not more edges are present in the left or right hemisphere of a graph for brain MRI data. You can choose a value for each vertex, or for the whole hemisphere.

Usage

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

Arguments

`g`	The igraph graph object
`level`	A character string indicating whether to calculate asymmetry for each region, or the hemisphere as a whole (default: 'hemi')
`.parallel`	Logical indicating whether or not to use *foreach* (default: TRUE)

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.

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>

get.resid *Linear model residuals across brain regions*

Description

This function runs linear models across brain regions listed in a `data.table` (e.g. cortical thickness), in order to adjust for relevant variables (e.g. age, sex, group, etc.). It adds the *studentized* residuals as a column and returns the data table.

Usage

```
get.resid(tidy.dt, covars, use.mean = FALSE, exclude = NULL)
```

Arguments

`tidy.dt`	A `data.table` that has been "tidied", containing all covariates and the brain measure of interest
`covars`	A `data.table` of covariates
`use.mean`	A logical indicating whether to control for the mean hemispheric brain value (e.g. mean LH/RH cortical thickness) (default: NULL)
`exclude`	A character vector of columns to exclude (default: NULL)

Value

A list with components:

`all.dat.tidy`	The tidied `data.table` with 'resids' column added
`formulas`	Character string of the `lm` formulas used
`resids.all`	The "wide" `data.table` of residuals

See Also

`rstudent`

graph.contract.brain *Contract graph vertices based on brain lobe and hemisphere*

Description

This function creates a new graph after merging multiple vertices based on brain lobe and hemisphere membership. The new vertex size is equal to the number of vertices in each lobe. The x- and y- coordinates of the new vertices are equal to the mean 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.

Usage

```
graph.contract.brain(g)
```

Arguments

g	The graph to contract

Value

A new graph

See Also

contract.vertices

---

graph.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. The global efficiency is equal to the mean of all nodal efficiencies.

Usage

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

Arguments

g	The graph on which to calculate efficiency
type	A character string; either 'local', 'nodal', or 'global'
weights	A numeric vector of edge weights; if 'NULL', and if the graph has edge attribute 'weight', then that will be used. To avoid using weights, this should be 'NA'
.parallel	Logical indicating whether or not to use foreach (default: TRUE)

Details

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*.

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:

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

Value

A vector of the local efficiencies for each vertex of the graph (if *type* is 'locallnodal') or a 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.

`graph_attr_dt` *Create a data table with graph global measures*

Description

This is just a helper function that takes a list of graphs and creates a data table of global measures for each graph, ordered by graph density.

Usage

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

Arguments

`g.list`	A list of igraph graph objects
`group`	A character string indicating group membership (default:NULL)

Value

A data table with several columns (equal to the number of graph attributes) and row number equal to the number of graphs in the input list

See Also

`graph_attr`, `graph_attr_names`

graph_neighborhood_multiple *Take the union of multiple neighborhood graphs*

Description

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

Usage

```
graph_neighborhood_multiple(g, vs)
```

Arguments

g	The 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

Examples

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

## End(Not run)
```

group.graph.diffs *Do between-group tests at each vertex for a given graph measure*

Description

This function takes two lists of graphs (the length of each equaling the number of subjects per group) and performs either a linear model, 2-sample t-test, or 2-sample Wilcoxon test at each vertex for a given network measure (e.g. vertex degree).

Usage

```
group.graph.diffs(g1, g2, measure, test = c("t.test", "wilcox.test", "lm"),
  covars = NULL, permute = FALSE, perm.order = NULL)
```

Arguments

g1	A list of igraph graph objects for group 1
g2	A list of igraph graph objects for group 2
measure	A character string of the measure to test
test	A character string for the test to use, either 't.test' (default) or 'wilcox.test'
covars	A data frame of covariates; only needed if using *lm* (default: NULL)
permute	Logical; should be *TRUE* if being called from permute.vertex (default: FALSE)
perm.order	A vector indicating the order that permuted subjects are in; only necessary if being called from permute.vertex

Details

The linear model choice is currently pretty inflexible. You will need to provide a data frame of covariates, of which *Study.ID* and *Group* need to be column names. Additionally, all graphs must have a *name* attribute (at the graph level) which matches the *Study.ID* for a given subject. This function will then return the p-value, t-statistic, and parameter estimate related to the *Group* covariate.

Value

A graph with vertex attributes:

size2	equals the t-statistic
size	*size2* transformed to be positive values (for visualization purposes)
p	Equal to $1 - p$
p.fdr	Equal to $1 - p_{FDR}$ (the FDR-adjusted p-value)

Author(s)

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

See Also

t.test, wilcox.test,p.adjust, vec.transform

Examples

```
## Not run:
g.diffs.btwn <- group.graph.diffs(g1, g2, 'btwn.cent', test='wilcox.test')
g.diffs.btwn <- group.graph.diffs(g.way[[1]][[5]], g.way[[2]][[5]], 'btwn.cent',
                                  test='lm', covars=covars.dti)

## End(Not run)
```

---

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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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

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.

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)
```

`loo` *"Leave-one-out" approach to estimate individual network contribution*

Description

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.

Usage

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

Arguments

`resids`	Data table of model residuals
`corrs`	List of lists of correlation matrices (as output by
`level`	Character string; the level at which you want to calculate contributions (either *global* or *regional*) `corr.matrix`). The length should equal the number of groups.

Value

A `data.table` with columns for

`Study.ID`	Subject identifier
`Group`	Group membership
`IC`	The value of the individual contribution

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

Examples

```
## Not run:
IC <- loo(resids.all, corrs)
RC <- loo(resids.all, corrs, 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 10 variables.

`name` a character vector of region names

`x` a numeric vector of x-coordinates (internal to `brainGraph`)

`y` a numeric vector of y-coordinates (internal to `brainGraph`)

`z` a numeric vector of z-coordinates (internal to `brainGraph`)

`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, 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

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)
```

`part.coeff` *Calculate vertex participation coefficient*

Description

This function calculates the participation coefficient of each vertex in a graph, based on community membership.

Usage

```
part.coeff(g, memb)
```

Arguments

`g`	The graph
`memb`	The community membership indices of each vertex

Details

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.

Value

A vector of the participation coeff's for each vertex of the graph.

Author(s)

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

References

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.

`permute.group` *Permutation test for group difference of graph measures*

Description

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

Usage

```
permute.group(permSet, density, resids, level = c("graph", "vertex", "lobe",
  "other"), atlas, measure = c("btwn.cent", "degree", "E.nodal", "knn",
  "transitivity", "vulnerability"), .function = NULL)
```

Arguments

`permSet`	A matrix of the set of permutations to loop through; the number of rows equals the desired number of permutations and the number of columns equals the total number of subjects across groups
`density`	Numeric; the density of the resultant graphs
`resids`	A data table of the residuals (from `get.resid`)
`level`	A character string for the attribute level to calculate differences; either 'graph', 'vertex', 'lobe', or 'other'
`atlas`	Character string of the atlas name
`measure`	A character string, either 'btwn.cent', 'degree', 'E.nodal', 'knn', or 'transitivity', 'vulnerability' (specific to the vertex *level*)
`.function`	A custom function you can pass (if *level* is 'other')

Details

The *graph* "level" will calculate modularity (Louvain algorithm), clustering coefficient, average path length, degree assortativity, global efficiency, lobe assortativity, and edge asymmetry.

The *vertex* "level" will calculate a vertex-wise measure. Currently, you can choose betweenness centrality, degree, nodal efficiency, k-nearest neighbor degree, transitivity, or vulnerability.

The *lobe* "level" is intended to test for group differences in number of inter-lobar connections, e.g. from the temporal lobe to the rest of the brain.

The *other* "level" allows you to pass your own function to do permutations with. This is useful if you want to calculate something that I haven't hard-coded (e.g. number of hubs between groups). It must take as its own arguments: "g1", "g2", and "density".

Value

A data table with values for group differences in modularity, global efficiency, clustering, average path length, and assortativity (etc.)

Author(s)

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

See Also

centr_betw, vulnerability,count_interlobar, edge_asymmetry, graph.efficiency

Examples

```
## Not run:
m <- get.resid(all.thick, covars)
myPerms <- shuffleSet(n=nrow(m$resids), nset=1e3)
out <- permute.group(myPerms, densities[N], m$resids, 'graph', atlas='dk')
out <- permute.group(myPerms, densities[N], m$resids, 'vertex')
out <- permute.group(myPerms, densities[N], m$resids, 'other',
  .function=myFun)

## End(Not run)
```

---

permute.vertex *Permutation test for group difference of graph vertex measures*

Description

This function performs a permutation test looking at the max t-statistic for a given vertex-level graph attribute (e.g. degree). This is the same principle as that of Nichols & Holmes (2001) used in voxelwise MRI analysis.

Usage

```
permute.vertex(g1, g2, measure, test = c("t.test", "wilcox.test", "lm"),
  alpha = 0.05, N = 1000, ...)
```

Arguments

g1	A list of igraph graph objects for group 1
g2	A list of igraph graph objects for group 2
measure	A character string of the measure to test
test	Character string for the test to use (passed to group.graph.diffs); one of 't.test', 'wilcox.test', or 'lm'
alpha	The significance level (default: 0.05)
N	The number of permutations (default: 1e3)
...	Other arguments passed to group.graph.diffs

Details

By default, a t-test is used when calling group.graph.diffs; if you would like to do e.g. a linear model, then see that function's help section and pass the appropriate arguments.

Value

A list with the following elements:

g	A graph representing group differences, with *p.perm* included as a vertex-level attribute (it is actually 1 - *p.perm*)
p.max	The proportion of the number of times the maximum t-statistic of the permuted groups was greater than the observed maximum t-statistic
thresh	The $1 - \alpha$ %ile maximum t-statistic of all permuted values

Author(s)

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

References

Nichols TE & Holmes AP (2001). *Nonparametric permutation tests for functional neuroimaging: A primer with examples.* Human Brain Mapping, 15(1):1-25.

See Also

sample, group.graph.diffs

Examples

```
## Not run:
g.diff <- permute.vertex(g.wt[[1]][[3]], g.wt[[2]][[3]], measure='degree')
g.diff <- permute.vertex(g[[1]][[5]], g[[2]][[5]], measure='degree',
    test='lm', covars=covars.dti)

## End(Not run)
```

---

plot_boot *Plot global graph measures with shaded regions calculated from bootstrapping*

Description

This function takes a list of boot objects (the number of elements equals the number of subject groups) and plots the observed value across all graph densities. It returns a list containing: a data.table with standard errors and 95% confidence intervals at each density, and 2 ggplot objects with shaded regions surrounding the observed values.

Usage

```
plot_boot(boot.dt, ylabel = NULL, alpha = 0.4, ...)
```

Arguments

boot.dt	A data.table output from boot_global
ylabel	A character string to place on the y-axis label (default: NULL)
alpha	A numeric indicating the opacity for geom_ribbon
...	Other parameters passed to geom_ribbon

Details

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

Value

A list with the following elements:

p1	A ggplot object with ribbon representing standard error
p2	A ggplot object with ribbon representing 95% confidence interval

Author(s)

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

See Also

boot_global

Examples

```
## Not run:
boot.mod.plots <- plot_boot(boot.mod$dt, ylab='Modularity')

## End(Not run)
```

---

plot_brainGraph *Plot a graph with a specific spatial layout*

Description

This function plots a graph when the spatial layout of the nodes is important (e.g. in the brain). It is really just a wrapper for plot.igraph, with some options pre-specified that work for plotting in the brain's layout. This means that set.brainGraph.attributes needs to be run on the graph, and a valid set of coordinates provided for the vertices. Most of the parameters valid here can be seen in igraph.plotting.

Usage

```
plot_brainGraph(g, rescale = F, ylim = c(-1.5, 1.5), asp = 0,
  main = NULL, ...)
```

Arguments

g	The graph to plot
rescale	A logical, whether to rescale the coordinates
ylim	A vector giving limits for the vertical axis
asp	A numeric constant for the aspect ratio
main	Character string for the main title
...	Other parameters (passed to plot).

---

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.attributes).

Usage

```
plot_brainGraph_gui()
```

---

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.

Usage

```
plot_brainGraph_list(g.list, fname.base, diffs = FALSE, cols = c("none",
  "lobe", "comm"))
```

Arguments

g.list	A list of igraph graph objects
fname.base	A character string specifying the base of the filename for *png* output
diffs	A logical, indicating whether or not to highlight edge differences (default: FALSE)
cols	A character string indicating how to color the vertices (default: 'none')

Details

You can choose to highlight edge differences between subsequent list elements, and whether to color vertices by *lobe*, *community* membership, or *lightblue* (the default). By default, the vertex sizes are equal to vertex degree, and max out at 20.

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>

---

plot_brainGraph_mni *Draw an axial or sagittal slice of the MNI152 T1 image*

Description

This function draws an axial or sagittal slice from the MNI152 T1 image, to plot the vertices of a graph over it. It will optionally write to a filename for output.

Usage

```
plot_brainGraph_mni(plane = c("axial", "sagittal"), slice, hemi = c("left",
  "right"), save = FALSE, fname = NULL)
```

Arguments

plane	Character string, either 'axial' or 'sagittal'
slice	The x or z-coordinate of the slice to use
hemi	Character string, either 'left' or 'right'
save	Logical indicating whether or not a png file should be saved (default: FALSE)
fname	The name of the file to be saved

See Also

image.nifti

`plot_corr_mat` *Plot a correlation matrix*

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", "lobe"), g = NULL,
  group = NULL)
```

Arguments

`corrs`	The correlation matrix
`ordered`	A logical indicating whether or not to order vertices (default:TRUE)
`type`	Character string, either 'comm' or 'lobe'
`g`	An igraph graph object; not required if *ordered* 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(corrs[[1]][[N]]$r.thresh, g=g[[1]][[N]],
                          group=groups[1])

## End(Not run)
```

plot_group_means *Plot group distributions of volumetric measures for a given brain region*

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_group_means(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

plot_perm_diffs *Calculate permutation p-values and plot group differences*

Description

For a given (global- or vertex-level) graph measure, determine the permutation p-value and create a plot showing group differences, either across densities or regions. You may specify the α-level; a red asterisk is added if $p < \alpha$ and a blue asterisk is added if $\alpha < p < 0.1$ (i.e. a "trend"). You may also choose whether you want a one- or two-sided test.

Usage

```
plot_perm_diffs(g1, g2, perm.dt, measure, level = c("graph", "vertex"),
  alternative = c("two.sided", "less", "greater"), alpha = 0.05,
  groups = NULL, ylabel = NULL)
```

Arguments

g1	List of igraph graph objects for group 1
g2	List of igraph graph objects for group 2
perm.dt	Data table with the permutation results
measure	Character string for the graph measure of interest
level	Character string, either 'graph' or 'vertex'
alternative	Character string, whether to do a two- or one-sided test (default: 'two.sided')
alpha	Significance level (default: 0.05)
groups	Character vector of group names (default: NULL)
ylabel	Character string for y-axis label (default: NULL)

Value

A list with three elements:

dt	A data table with p-values for each density/region
p1	A ggplot plotting object
p2	A ggplot plotting object

Author(s)

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

See Also

permute.group

Examples

```
## Not run:
perms.mod.sig <- perms.sig(g[[1]], g[[2]], perms.all, 'mod', level='graph',
  'less', groups, ylabel='Modularity')
perms.mod.btwn <- perms.sig(g[[1]], g[[2]], perms.btwn, 'btwn.cent',
  level='vertex')

## End(Not run)
```

---

plot_rich_norm *Plot normalized rich club coefficients against degree threshold*

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, densities, alpha = 0.05, g = NULL)
```

Arguments

rich.dt	A data.table with rich-club coefficients
densities	A numeric vector of the densities to plot
alpha	The significance level (DEFAULT: 0.05)
g	A list (of lists) of igraph 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>

Examples

```
## Not run:
richPlot <- plot_rich_norm(rich.dt, densities[N:(N+1)], g=g)

## End(Not run)
```

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

Description

This function creates boxplots of a single vertex-level graph measure at a single density, grouped by *lobe*; each lobe has a separate *facet* in a ggplot object.

Usage

```
plot_vertex_measures(dat, cur.density = 0.1, measure = "btwn.cent",
  ylabel = NULL)
```

Arguments

dat	A "tidied" data table of vertex-level graph measures
cur.density	A numeric indicating the graph density
measure	A character string of the graph measure to plot (default: 'btwn.cent')
ylabel	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(net.meas.tidy, densities[N], 'btwn.cent')

## End(Not run)
```

rich.club.coeff *Calculate the rich club of a graph*

Description

This function calculates the rich club of a graph, both the coefficient ϕ and the nodes that make up this subgraph.

Usage

```
rich.club.coeff(g, k = 1, weighted = FALSE)
```

Arguments

g	The graph of interest
k	The minimum degree for including a vertex (default: 1)
weighted	A logical indicating whether or not edge weights should be used (default: FALSE)

Value

A list with the following components:

phi	The rich club coefficient, ϕ.
graph	A subgraph containing only the rich club nodes.
Nk	The number of vertices in the rich club graph.
Ek	The number of edges in the rich club graph.

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.

See Also

rich.club.norm

---

rich.club.norm *Calculate the normalized rich club coefficient*

Description

This function will generate a number of random graphs, calculate their rich club coefficients (ϕ), and return ϕ of the graph of interest divided by the mean across random graphs, i.e. ϕ_{norm}. If random graphs have already been generated, you can supply a list as an argument (since graph generation is time consuming).

Usage

```
rich.club.norm(g, N = 100, rand = NULL, ...)
```

Arguments

g	The igraph graph object of interest
N	The number of random graphs to generate (default: 100)
rand	A list of igraph graph objects, if random graphs have already been generated (default: NULL)
...	Other parameters (passed to *rich.club.coeff*)

Value

A list with two elements:

phi.rand	A matrix with *N* rows and *max(degree(g))* columns, where each row contains the coefficients for a given random graph.
phi.orig	A vector of the rich-club coefficients for the original graph.
phi.norm	A named vector of the normalized rich club coefficients.
p	The p-value based on the *N* random graphs generated.

Author(s)

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

References

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

See Also

rich.club.coeff, sim.rand.graph.par

---

rich.core *Calculate the rich core of a graph*

Description

This function 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 that that rank, and the core size relative to the total number of vertices in the graph.

Usage

```
rich.core(g)
```

Arguments

g	The graph of interest

Value

A data frame with the following components:

density	The density of the graph.
rank	The rank of the boundary for the rich core.
k.r	The degree of the vertex at the boundary.
core.size	The size of the core relative to the graph size.

Author(s)

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

References

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

See Also

rich.club.coeff, rich.club.norm

---

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	The igraph graph object of interest
type	A character string; either 'vertex' or 'edge' removals
measure	A character string; sort by either 'btwn.cent' or 'degree', or choose 'random'
N	Integer; the number of iterations if *random* is chosen

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

A vector representing the ratio of maximal component size after each removal to the graph's original maximal component

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.

`rotation` *Apply a rotation matrix to a set of points*

Description

This function takes a set of points and applies a rotation matrix (e.g. will rotate points 90 deg. if given "pi/2" as input)

Usage

```
rotation(x, theta)
```

Arguments

`x`	A matrix with 2 columns of the points to rotate
`theta`	The angle to apply

Value

A matrix with 2 columns of the points' new locations

Author(s)

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

set.brainGraph.attributes

Set a number of graph and vertex attributes useful in MRI analyses

Description

This function will set a number of graph, vertex, and edge attributes of a given igraph object.

Usage

```
set.brainGraph.attributes(g, atlas = NULL, modality = NULL,
  subject = NULL, group = NULL, rand = FALSE)
```

Arguments

g	An igraph object
atlas	A character vector indicating which atlas was used for the nodes
modality	A character vector indicating imaging modality (e.g. 'dti')
subject	A character vector indicating subject ID (default: NULL)
group	A character vector indicating group membership (default: NULL)
rand	Logical indicating if the graph is random or not (default: FALSE)

Value

g A copy of the same graph, with the following attributes:

Graph-level	Package version, atlas, density, connected component sizes, diameter, \# of triangles, transitivity, average path length, assortativity, clique number, 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), coreness, local & nodal efficiency, color (community), color (lobe), color (component), membership (community), membership (component), participation coefficient, within-module degree z-score, vulnerability, and coordinates (x, y, and z)
Edge-level	Color (community), color (lobe), color (component), edge betweenness, Euclidean distance (in mm)

Author(s)

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

See Also

components, diameter, clique_num, centr_betw, part.coeff, edge.betweenness, centr_eigen, hub.score, auth

`sim.rand.graph.clust` *Simulate a random graph with given degree sequence and clustering.*

Description

This function will simulate a random graph with a given degree sequence and clustering coefficient. This function calls choose.edges to decide which edges will be re-wired.

Usage

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

Arguments

`graph`	The graph from which null graphs are simulated
`cl`	The clustering measure (default: transitivity)
`max.iters`	The maximum number of iterations to perform (default: 100)

Value

A list with components:

`g`	The random graph that was generated
`iters`	The total number of iterations performed
`cl`	The clustering coefficient at each step

See Also

`choose.edges`, `rewire`,`transitivity`, `keeping_degseq`

`sim.rand.graph.par` *Simulate N random graphs w/ same clustering and degree sequence as the input.*

Description

This function will simulate N simple random graphs with the same clustering and degree sequence as the input. Essentially a wrapper for `sim.rand.graph.clust` and `set.brainGraph.attributes`. It uses `foreach` to speed it up. If you do not want to match by clustering, then it will do a simple rewiring of the given graph (the number of rewire's equaling the larger of 1e4 and 10 * number of edges).

Usage

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

Arguments

g	A graph with the characteristics for simulation of random graphs
N	The number of iterations
clustering	Logical for whether or not to control for clustering
...	Other parameters (passed to sim.rand.graph.clust)

Value

A list of *N* random graphs with vertex and graph attributes.

See Also

sim.rand.graph.clust, rewire

Examples

```
## Not run:
rand1 <- sim.rand.graph.par(g1[[N]], N=1e3, clustering=F)
rand1.cl <- sim.rand.graph.par(g1[[N]], N=1e2, max.iters=1e3)

## End(Not run)
```

---

small.world *Calculate graph small-worldness*

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.

---

spatial.dist *Calculate Euclidean distance between vertices (MNI space)*

Description

This function calculates the Euclidean distance of edges between vertices of a graph. The distances are in mm and based on MNI space. The 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.

Usage

```
spatial.dist(g)
```

Arguments

g	An igraph graph object

Value

A numeric vector with length equal to the edge count of the input graph

Author(s)

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

`update_brainGraph_gui` *Function to dynamically plot a graph*

Description

This function is called by plot_brainGraph_gui to update a plot on-the-fly. It updates by calling the helper function `make.plot`.

Usage

```
update_brainGraph_gui(plotDev, graph1, graph2, plotFunc, vertSize, edgeWidth,
  vertColor, hemi, lobe, orient, vertSize.min, edgeWidth.min,
  vertSize.const = NULL, edgeWidth.const = NULL, vertLabels = NULL,
  comm = NULL, kNumComms = NULL, neighb = NULL, neighbMult = NULL,
  slider = NULL, vertSize.other = NULL, vertSize.eqn = NULL,
  showDiameter = NULL, edgeDiffs = NULL)
```

Arguments

`plotDev`	A Cairo device for the plotting area
`graph1`	An igraph graph object for the first plotting area
`graph2`	An igraph graph object for the second plotting area
`plotFunc`	A function specifying which type of plot to use
`vertSize`	A GTK combo box for scaling vertex size
`edgeWidth`	A GTK entry for changing edge width
`vertColor`	A GTK combo box for changing vertex colors
`hemi`	A GTK combo box for plotting individual hemispheres
`lobe`	A GTK combo box for plotting individual lobes
`orient`	A GTK combo box for plotting a specific orientation
`vertSize.min`	A GTK spin button for minimum vertex size
`edgeWidth.min`	A GTK spin button for minimum edge width
`vertSize.const`	A GTK entry for constant vertex size
`edgeWidth.const`	A GTK entry for constant width
`vertLabels`	A GTK check button for showing vertex labels
`comm`	A GTK combo box for plotting individual communities
`kNumComms`	Integer indicating the number of total communities (optional)
`neighb`	A GTK combo box for plotting individual neighborhoods

neighbMult	A GTK entry for joint neighborhoods of multiple vertices
slider	A GTK horizontal slider widget for changing edge curvature
vertSize.other	A GTK entry for vertex size (other attributes)
vertSize.eqn	A GTK entry for equations to exclude vertices
showDiameter	A GTK check button for showing the graph's diameter
edgeDiffs	A GTK check button for showing edge diffs between graphs

Author(s)

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

`vec.transform` *Transform a vector to have a different range*

Description

This function takes a vector and transforms it to have a new range, given the input, or the default values of [0, 1].

Usage

```
vec.transform(x, min.val = 0, max.val = 1)
```

Arguments

x	the vector to transform
min.val	the minimum value of the new range
max.val	the maximum value of the new range

Value

A vector of the transformed input.

vertex_attr_dt *Create a data table with graph vertex measures*

Description

This is just 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.).

Usage

```
vertex_attr_dt(g, group = NULL)
```

Arguments

g	An igraph graph object
group	A character string indicating group membership (default:NULL)

Value

A data table with 18-19 columns and row number equal to the number of vertices in the graph

See Also

vertex_attr, vertex_attr_names,as_data_frame

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, .parallel = TRUE, weighted = FALSE)
```

Arguments

g	The igraph graph object of interest
.parallel	Logical indicating whether or not to use *foreach* (default: TRUE)
weighted	Logical indicating whether weighted efficiency should be calculated (default: FALSE)

Value

A vector of length equal to the number of vertices in *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

graph.efficiency

`within_module_deg_z_score` *Calculate vertex within-module degree z-score*

Description

This function calculates the within-module degree z-score of each vertex in a graph, based on some module membership. This is a measure of the connectivity from a given vertex to other vertices in its module.

Usage

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

Arguments

g	The graph
memb	The community membership indices of each vertex

Details

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 within-module degree z-scores for each vertex of the graph.

Author(s)

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

References

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.

---

write.brainnet *Write files to be used for visualization with BrainNet Viewer*

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 = c("none", "community", "lobe"),
  node.size = "constant", file.prefix = "")
```

Arguments

g	A graph
node.color	Character string indicating whether to color the nodes or not; can be 'none', 'lobe', or 'community'
node.size	Character string indicating what size the nodes should be; can be any vertex-level attribute (default: 'constant')
file.prefix	Character string for the basename of the *.node* and *.edge* 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.

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.

Index