Package 'brainwaver'

July 2, 2014

Version 1.6

Date 2010-08-09

Title Basic wavelet analysis of multivariate time series with a visualisation and parametrisation using graph theory.

Author Sophie Achard <sophie.achard@gipsa-lab.inpg.fr>

Maintainer Sophie Achard <sophie.achard@gipsa-lab.inpg.fr>

Depends R (>= 1.10.0), waveslim

ZipData no

Description This package computes the correlation matrix for each scale of a wavelet decomposition, namely the one performed by the R package waveslim (Whitcher, 2000). An hypothesis test is applied to each entry of one matrix in order to construct an adjacency matrix of a graph. The graph obtained is finally analysed using the small-world theory (Watts and Strogatz,1998) and using the computation of efficiency (Latora, 2001),tested using simulated attacks. The brainwaver project is complementary to the camba project for brain-data preprocessing. A collection of scripts (with a makefile) is avalaible to download along with the brainwaver package, see information on the webpage mentioned below.

License GPL (>= 2)

URL http://www.gipsa-lab.grenoble-inp.fr/~sophie.achard/

Repository CRAN

Date/Publication 2012-12-02 07:08:10

NeedsCompilation yes

R topics documented:

`brain` *Time series obtained by an fMRI experiement on the brain*

Description

Time series for each region of interest in the brain. These series are obtained by SPM preprocessing.

Usage

```
data(brain)
```

Format

A data frame with 2048 observations on the following 90 variables.

Source

contact S. Achard (sa428@cam.ac.uk)

References

R. Salvador, J. Suckling, C. Shwarzbauer, E.T. Bullomre (2005) Undirected graphs of frequency-dependent functional connectivity in the whole brain networks. *Philos Trans R Soc Lond B Biol Sci* Vol. 360, pages 937-946.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

Examples

```
data(brain)
## maybe str(brain) ; plot(brain) ...
```

---

choose.thresh.nbedges *Threshold associated to a given number of edges.*

Description

Computes the threshold for the correlation matrix in order to obtain an adjacency matrix with a given number of edges.

Usage

```
choose.thresh.nbedges(cor.mat,  var.ind.mat = 0, n.ind = 0,  thresh = 0.05,
                      nb.edges = 405, test.method = "gaussian",
                      proc.length = 518, num.levels, use.tanh = FALSE,
                      max.iter = 10)
```

Arguments

`cor.mat`	matrix containing the correlation values. (must be diagonal with 1 on the diagonal)
`var.ind.mat`	matrix containing the variance inter individuals of the correlation. Only used with `test.method="t.test"`. (default not used)
`n.ind`	number of individuals to take into account in the test. Only used with `test.method="t.test"`. (default not used)
`thresh`	indicates the rate at which the FDR procedure is controlled. (default 0.05)
`nb.edges`	indicates the exact number of edges that the final graph should contain.
`test.method`	name of the method to be applied. `"gaussian"` assumes a gaussian law for the estimator. `"t.test"` implements a t.test for computing the p-value. (default `"gaussian"`)
`proc.length`	specifies the length of the original processes using to construct the `cor.mat`
`num.levels`	specifies the number of the wavelet scale to take into account in the hypothesis test. Only used with `test.method="gaussian"`

use.tanh	logical. If FALSE take the atanh of the correlation values before applying the hypothesis test, in order to use the Fisher approximation
max.iter	indicates the number of maximum iteration to compute before stopping the loop

Details

In order to compare graphs, the best way to do it is to make sure that all the graphs have the same number of edges!

Value

Real number corresponding to the threshold value.

Note

only in version 2 and higher

Author(s)

S. Achard

References

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

Examples

```
data(brain)
brain<-as.matrix(brain)
# WARNING : To process only the first five regions
brain<-brain[,1:5]

#Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)

#Construction of the adjacency matrice for scale 4

adj.mat.4<-const.adj.mat(wave.cor.list[[4]], sup = 0.44,proc.length=dim(brain)[1],
                         num.levels=4)
nb.edges<-sum(adj.mat.4)/2

sup.thresh<-choose.thresh.nbedges(wave.cor.list[[4]],nb.edges=nb.edges,
                        proc.length=dim(brain)[1],num.levels=4)
```

compute.FDR *False Discovery Rate computation*

Description

Computation of the p-value cut-off which controls the false discovery rate when the test statistics have positive regression dependency on each of the test statistics corresponding to the true null hypotheses.

Usage

```
compute.FDR(pvalue.vec, q)
```

Arguments

pvalue.vec	a vector containing the p-value for each hypothesis test.
q	value of the desired False Discovery Rate, exactly the upper limit for the expectation of the proportion of false positives.

Details

This code implements the FDR procedure described in Benjamini and Yekutieli (2001).

Value

a real giving the p-value cutt-off.

Note

The GeneTS package have also an implementation of this function

Author(s)

S. Achard

References

Benjamini Y. and Yekutieli D. (2001) The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, Vol. 29, No. 4, pages 1165-1188

Examples

```
data(young)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]
```

```
# Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 4,
                              boundary = "periodic", p.corr = 0.975)

# For scale 4
pvalue.cor<-p.value.compute(wave.cor.list[[4]],proc.length=dim(brain)[1],
                            sup=0.44, num.levels=4)

# Computation of the p-value threshold using FDR procedure
pvalue.thresh<-compute.FDR(pvalue.cor,0.05)
```

`const.adj.list` *Computation of the list of adjacency matrices*

Description

Computes the list of the adjacency matrices in terms of the scale of the wavelet decomposition.

Usage

```
const.adj.list(wave.cor.list, wave.var.ind = 0, n.ind = 0, thresh = 0.05,
                 sup = 0, test.method = "gaussian", proc.length,
                 use.tanh = FALSE)
```

Arguments

`wave.cor.list`	object of class `"Wave Correlation"` containing the correlation matrices to be analysed
`wave.var.ind`	object of class `"Wave Correlation"` containing the inter individuals variance of the correlation. Only used with `test.method="t.test"`. (default not used)
`n.ind`	number of individuals to take into account in the test. Only used with `test.method="t.test"`. (default not used)
`thresh`	indicates the rate at which the FDR procedure is controlled. (default 0.05)
`sup`	indicates the correlation threshold to consider in each hypothesis test
`test.method`	name of the method to be applied. `"gaussian"` assumes a gaussian law for the estimator. `"t.test"` implements a t.test for computing the p-value. (default `"gaussian"`).
`proc.length`	specifies the length of the original processes using to construct the `wave.cor.list`
`use.tanh`	logical. If FALSE take the `atanh` of the correlation values before applying the hypothesis test, in order to use the Fisher approximation

Details

Each hypothesis test is written as :

H_0 : "|correlation| $<=$ sup"

H_1 : "|correlation| $>$ sup"

Value

Object of class `"Wave Adjacency matrix"`, basically, a list with the following components

d?	Adjacency matrix for each scale of the wavelet decomposition

Author(s)

S. Achard

References

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

`const.adj.mat`

Examples

```
data(brain)
brain<-as.matrix(brain)
# WARNING : To process only the first five regions
brain<-brain[,1:5]


# Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)

# Construction of the adjacency matrices associated to each level of the
# wavelet decomposition
wave.adj.list<-const.adj.list(wave.cor.list, sup = 0.44, proc.length=dim(brain)[1])

par(mfrow=c(3,2))

for(i in 1:4)
{
name.txt<-paste("Level ",i,sep="")
image(wave.adj.list[[i]],col=gray((0:20)/20),main=name.txt)
}
```

`const.adj.mat` *Computation of the adjacency matrix*

Description

Computes the adjacency matrix for a given correlation matrix.

Usage

```
const.adj.mat(cor.mat, var.ind.mat = 0, n.ind = 0, thresh = 0.05, sup = 0,
              test.method = "gaussian", proc.length, num.levels,
              use.tanh = FALSE)
```

Arguments

`cor.mat`	matrix containing the correlation values. (must be diagonal with 1 on the diagonal)
`var.ind.mat`	matrix containing the variance inter individuals of the correlation. Only used with `test.method="t.test"`. (default not used)
`n.ind`	number of individuals to take into account in the test. Only used with `test.method="t.test"`. (default not used)
`thresh`	indicates the rate at which the FDR procedure is controlled. (default 0.05)
`sup`	indicates the correlation threshold to consider in each hypothesis test.
`test.method`	name of the method to be applied. `"gaussian"` assumes a gaussian law for the estimator. `"t.test"` implements a t.test for computing the p-value. (default `"gaussian"`)
`proc.length`	specifies the length of the original processes using to construct the `cor.mat`
`num.levels`	specifies the number of the wavelet scale to take into account in the hypothesis test. Only used with `test.method="gaussian"`
`use.tanh`	logical. If FALSE take the `atanh` of the correlation values before applying the hypothesis test, in order to use the Fisher approximation

Details

Each hypothesis test is written as :

H_0 : "|correlation| $<=$ sup"

H_1 : "|correlation| $>$ sup"

Value

Binary matrix.

Author(s)

S. Achard

References

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

const.adj.list

Examples

```
data(brain)
brain<-as.matrix(brain)
# WARNING : To process only the first five regions
brain<-brain[,1:5]


#Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)

#Construction of the adjacency matrice for scale 4

adj.mat.4<-const.adj.mat(wave.cor.list[[4]], sup = 0.44,proc.length=dim(brain)[1],
                         num.levels=4)

image(adj.mat.4,col=gray((0:20)/20))
```

---

const.cor.list *Computation of the list of correlation matrices*

Description

Computes the list of the correlation matrices in terms of the scale of the wavelet decomposition.

Usage

```
const.cor.list(data.mat, names.data = 0, method = "modwt", wf = "la8",
               n.levels = 4, boundary = "periodic", p.corr = 0.975,
               save.wave = FALSE, export.data = FALSE)
```

Arguments

data.mat	matrix containing the data time series. Each column of the matrix represents one time series.
names.data	optional character vector containing the name associated to the column of the matrix data.mat.

`method`	wavelet decomposition to be used, algorithm implemented in the `waveslim` package (Whitcher, 2000). By default, the Maximal Overlap Discrete Wavelet Transform is used `"modwt"`. It is also possible to use the classical Discrete Wavelet Transform `"dwt"`.
`wf`	name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`n.levels`	specifies the depth of the decomposition. This must be a number less than or equal to $\log_2(\text{length}(x))$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.
`p.corr`	(one minus the) two-sided p-value for the confidence interval
`save.wave`	logical. If TRUE all the wavelet coefficient are saved.
`export.data`	logical. If TRUE the correlation matrices with the upper and lower bound are exported to text file.

Details

This function uses the wavelet decomposition implemented in the R package `waveslim`, (whitcher, 2000).

Value

Object of class `"Wave Correlation"`, basically, a list with the following components

`d?`	Correlation matrix for each scale of the wavelet decomposition.
`lowerd?`	matrix containing the lower bound of the correlation for each scale of the wavelet decomposition.
`upperd?`	matrix containing the upper bound of the correlation for each scale of the wavelet decomposition.

Note

change between version 1.1 and 1.2, now the length of the time series is saved with the values of the correlation.

Author(s)

S. Achard

References

R. Gencay, F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

D. B. Percival and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

const.var.list, read.cor.txt, save.cor.txt

Examples

```
data(brain)
brain<-as.matrix(brain)
# WARNING : To process only the first five regions
brain<-brain[,1:5]


n.levels<-4
wave.cor.list<-const.cor.list(brain,n.levels=n.levels)

tot.regions <- dim(brain)[2]
n.series <- dim(brain)[1]
col.regions<-1

nb.comp.regions <- 8
    comp.regions <- round(runif(nb.comp.regions,2,tot.regions))
    sym.region <- col.regions+1

comp.regions <- c(sym.region,comp.regions)

    name.ps <- "example-1.ps"
    postscript(name.ps)
    par(mfrow=c(3,3))
    for(k in 1:(nb.comp.regions+1)){

        reg <- comp.regions[k]


       cor.vector<-matrix(0,4,3)
for(i in 1:n.levels){

       cor.vector[i,1]<-(wave.cor.list[[i]])[1,reg]
       cor.vector[i,2]<-(wave.cor.list[[i+n.levels]])[1,reg]
       cor.vector[i,3]<-(wave.cor.list[[i+2*n.levels]])[1,reg]

}
```

```
      title <- paste("1 -- ",comp.regions[k],sep="")
     matplot(2^(0:(n.levels-1)),cor.vector,main=title,type="b",
             log="x", pch="*LU", xaxt="n", lty=1, col=c(1,4,4),
             xlab="Wavelet Scale",ylab="Wavelet Covariance",ylim=c(-0.5,1))
     axis(side=1, at=2^(0:7))
     abline(h=0)
   }
 dev.off()
```

`const.var.list` *Computation of the list of variance vectors*

Description

Computes the list of the variance vectors in terms of the scale of the wavelet decomposition.

Usage

```
const.var.list(data.mat, names.data = 0, method = "modwt", wf = "la8",
               n.levels = 4, boundary = "periodic", save.wave = FALSE,
               export.data = FALSE)
```

Arguments

`data.mat`	matrix containing the data time series. Each column of the matrix represents one time series.
`names.data`	optional character vector containing the name associated to the column of the matrix `data.mat`.
`method`	wavelet decomposition to be used, algorithm implemented in the `waveslim` package (Whitcher, 2000). By default, the Maximal Overlap Discrete Wavelet Transform is used `"modwt"`. It is also possible to use the classical Discrete Wavelet Transform `"dwt"`.
`wf`	name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`n.levels`	specifies the depth of the decomposition. This must be a number less than or equal to $\log_2(\text{length}(x))$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.

save.wave	logical. If TRUE all the wavelet coefficient are saved.
export.data	logical. If TRUE the variance vectors with the upper and lower bound are exported to text file.

Details

This function uses the wavelet decomposition implemented in the R package waveslim, (whitcher, 2000).

Value

Object of class "Wave Variance", basically, a list with the following components

d?	Variance vectors for each scale of the wavelet decomposition.
lowerd?	vector containing the lower bound of the variance for each scale of the wavelet decomposition.
upperd?	vector containing the upper bound of the variance for each scale of the wavelet decomposition.

Author(s)

S. Achard

References

Gencay, R., F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

Percival, D. B. and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

const.cor.list, read.var.txt, save.var.txt

Examples

```
data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]



n.levels<-4
wave.var.list<-const.var.list(brain,n.levels=n.levels)
```

```
tot.regions <- dim(brain)[2]
n.series <- dim(brain)[1]

nb.num.regions <- 9
   num.regions <- round(runif(nb.num.regions,2,tot.regions))
    par(mfrow=c(3,3))
    for(k in 1:(nb.num.regions)){

        reg <- num.regions[k]

       var.vector<-matrix(0,4,3)
for(i in 1:n.levels){

var.vector[i,1]<-(wave.var.list[[i]])[reg]
        var.vector[i,2]<-(wave.var.list[[i+n.levels]])[reg]
        var.vector[i,3]<-(wave.var.list[[i+2*n.levels]])[reg]
      }
        title <- num.regions[k]
        matplot(2^(0:(n.levels-1)),var.vector,main=title,type="b",
                log="x", pch="*LU", xaxt="n", lty=1, col=c(1,4,4),
                xlab="Wavelet Scale",ylab="Wavelet Variance")
        axis(side=1, at=2^(0:7))
        abline(h=0)
      }
```

`correlations.to.adjacencies` *Produce adjencency matrices for a given number of edges*

Description

Given a correlations thingy as produced by `const.cor.list`, produce a list of adjacency matrices fiddled to have a preferred number of edges Actually this is not quite possible, but come as close as `choose.thresh.nbedges` will allow us. A functional parameter allows us to say things like produce the graphs with n log n edges where n is the number of nodes

Usage

```
correlations.to.adjacencies(correlations, edge.func)
ideal.wavelet.levels(brain)
distance(x,y,z)
```

Arguments

`correlations`	a list of correlation matrices produced by `const.cor.list`

edge.func	a function to mention the way to choose the number of edges given the number of nodes in the graph. In the companion scripts files, the small-limit is used and by default edge.func=(function(x){x*log(x)})
brain	matrix containing the data time series. Each column of the matrix represents one time series.
x	x coordinate
y	y coordinate
z	z coordinate

Details

Functions produced to manipulate better nice outputs of the package

Value

correlations.to.adjacencies	Description of 'comp1'
ideal.wavelets.levels	number indicating up to each wavelet scale it is possible to go given the length of the time series
disctance	the euclidean distance in 3D

Author(s)

John Aspden, external collaborator of the brainwaver package

References

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

const.cor.list

Examples

```
data(brain)
brain<-as.matrix(brain)
# WARNING : To process only the first five regions
brain<-brain[,1:5]

n.levels<-4
wave.cor.list<-const.cor.list(brain,n.levels=n.levels)
adj.mat<-correlations.to.adjacencies(wave.cor.list,edge.func=(function(x){x*log(x)}))
```

`efficiency` *Graph efficiency*

Description

Computes various measures of efficiency of a graph using the definition given by (Latora, 2001 and 2003)

Usage

```
global.efficiency(adj.mat, weight.mat)
local.efficiency(adj.mat, weight.mat)
global.cost(adj.mat, weight.mat)
cost.evaluator(x)
```

Arguments

`adj.mat`	adjacency matrix of the graph
`weight.mat`	weighted matrix associated to the graph
`x`	real

Details

Formula for the global efficiency :

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

where $L_{i,j}$ is the minimum path length between each pair of nodes, and G the graph.

Formula for the nodal efficiency for the node i:

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

Formula for the local efficiency for node i:

$$E_{local} = \frac{1}{N_{G_i}(N_{G_i}-1)} \sum_{j,k \in G_i} \frac{1}{L_{j,k}}$$

where G_i is the set of nodes which are nearest-neighbours of the ith node, i.e., the nodes in G_i are each directly connected by a single edge to the index node i and N_{G_i} is the number of nodes in the sub-graph G_i.

The computation of the cost requires the definition of an internal function, called `cost.evaluator`. For the moment, the `cost.evaluator` is the identity. Refer to Latora (2001) for the exact definition and usage of this function.

Value

`global.efficiency$nodal.eff`
: vector containing the nodal efficiency for each node of the graph (equal to the inverse of the harmonic mean of the path length, when two nodes are disconnected, the path length is taken to be infinity so the inverse is 0)

`global.efficiency$eff`
: real corresponding to the mean of the nodal efficiency for the whole graph

`local.efficiency$loc.eff`
: vector containing the local efficiency for each node of the graph (see details for the exact fomula)

`global.efficiency$eff`
: real corresponding to the mean of the local efficiency for the whole graph

`global.cost`
: real corresponding to the mean of the cost for the whole graph

Note

only in version 2 and higher

Author(s)

S. Achard

References

V. Latora, M. Marchiori (2001) Efficient Behavior of Small-World Networks. *Phys. Rev. Lett.*, Vol. 87, N. 19, pages 1-4.

V. Latora, and M. Marchiori (2003) Economic Small-World Behavior in Weighted Networks. *Europ. Phys. Journ. B*, Vol. 32, pages 249-263.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

`const.adj.list`, `small.world`

Examples

```
data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]



n.regions<-dim(brain)[2]
```

```
#Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)


sup.seq<-((1:10)/10) #sequence of the correlation threshold
nmax<-length(sup.seq)
Eglob<-matrix(0,6,nmax)
Eloc<-matrix(0,6,nmax)
Cost<-matrix(0,6,nmax)

n.levels<-6

#For each value of the correlation thrashold
for(i in 1:nmax){
n.sup<-sup.seq[i]

#Construction of the adjacency matrices associated to each level of the wavelet decomposition
wave.adj.list<-const.adj.list(wave.cor.list, sup = n.sup)


#For each level of the wavelet decomposition
for(j in 1:n.levels){

Eglob.brain<-global.efficiency(wave.adj.list[[j]],
weight.mat=matrix(1,n.regions,n.regions))
Eglob[j,i]<-Eglob.brain$eff

Eloc.brain<-local.efficiency(wave.adj.list[[j]],
weight.mat=matrix(1,n.regions,n.regions))
Eloc[j,i]<-Eloc.brain$eff

Cost.brain<-global.cost(wave.adj.list[[j]],
weight.mat=matrix(1,n.regions,n.regions))
Cost[j,i]<-Cost.brain

}}


plot(sup.seq,(1:nmax)/2,type='n',xlab='Correlation threshold, R',ylab='Global efficiency',
     cex.axis=2,cex.lab=2,xlim=c(0,1),ylim=c(0,1))


for(i in 1:n.levels){
lines(sup.seq,Eglob[i,],type='l',col=i,lwd=2)
}
legend(x="topright",legend=c("Level 1","Level 2","Level 3","Level 4",
"Level 5","Level 6"),fill=TRUE,col=(1:n.levels),lwd=2)
```

`equadist.rand.sw` *Small-world parameters for a simulated graph given its degree distribution*

Description

Computes the degree, the minimum path length and the clustering coefficient for a simulated graph with a known degree distribution.

Usage

```
equadist.rand.sw(nsim, dat = "reduced", dist.mat, degree.dist)
```

Arguments

`nsim`	number of simulated graphs to use for the computation of the small-world parameters.
`dat`	character string specifying if all the small-world parameters have to be returned. If `"reduced"`, only the mean of the parameters for the whole graph is returned.
`dist.mat`	matrix with a distance associated to each pair of nodes of the graph to take into account in the computation of the small-world parameters.
`degree.dist`	vector describing the degree distribution verified by the nodes of the simulated graph.

Details

Because of the problem for the simulation of a random graph with exactly the same degree distribution, each simulation applied in this function is subject to the result to get a graph with the same number of edges as expected. So, the number of wanted simulations can be very different from the number of simulations taken into account in the computation of the small-world parameters.

Value

`in.degree`	mean of the degree for the whole graph.
`Lp.rand`	mean of the minimum path length for the whole graph.
`Cp.rand`	mean of the clustering coefficient for the whole graph.
`in.degree.all`	vector of the degree of each node of the graph.
`Lp.rand.all`	vector of the minimum path length of each node of the graph.
`Cp.rand.all`	vector of the clustering coefficient of each node of the graph.

Author(s)

S. Achard

References

S. H. Strogatz (2001) Exploring complex networks. *Nature*, Vol. 410, pages 268-276.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

sim.equadist, rand.sw, reg.sw

Examples

```
#For a scale-free graph

x<-1:50
probx<-x^(-1.4)
n.nodes<-50
n.edges<-250

stop<-0

while(stop==0){
write.table(stop)
r<-sample(x,n.nodes,prob=probx,replace=TRUE)
if(sum(r)==n.edges) stop<-1
  }

sf.degree<-r

mat<-sim.equadist(sf.degree)

result<-equadist.rand.sw(10, dat = "reduced", dist.mat=matrix(1,50,50),
                  degree.dist=sf.degree)
```

`fitting` *Fitting laws with maximum likelihood*

Description

Fits three laws (exponential, power and truncated power laws) to an empirical distribution using the maximum likelihood estimators.

Usage

```
fitting(degree.dist, nmax)
```

Arguments

degree.dist	vector of the distribution to be fitted.
nmax	maximum of the value of degree.dist.

Details

The fitted laws are : exponential law, power law and truncated power law.

This function plots the histogram of degree.dist (dist.ps). This function plots also the cumulative distributions of the empirical and three fitted laws in a log-log scale (fitting.ps). Finally, all the parameters are exported to the file fitting.txt.

Value

mu	parameter of the exponential law.
gamma	parameter of the power law.
alpha, beta	parameter of the truncated power law.
AIC	vector containing the Akaike's criterion for the fitting of the three laws.

Author(s)

S. Achard

References

Akaike, H. (1974) A new look at the statistical model identification *IEEE Transactions on Automatic Control* Vol. 19, N. 6,pages 716-723.

Examples

```
data(brain)

brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]



n.regions<-dim(brain)[2]

#Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)

#Construction of the adjacency matrices associated to each level of the wavelet decomposition
wave.adj.list<-const.adj.list(wave.cor.list, sup = 0.44, proc.length=dim(brain)[1])

# For scale 4
degree.dist<-rowSums(wave.adj.list[[4]])
```

```
par(cex=1.5,cex.lab=1.2,font.lab=2)
hist(degree.dist,xlab="Degree", ylab="Number of regions",main=NULL,col=1,border=8)

nmax<-50
tmp<-hist(degree.dist,breaks=c(0:nmax))
cum.dist<-1-cumsum(tmp$counts)/n.regions
# cumulative distribution of degree.dist


d<-fitting(degree.dist,nmax)

exp.trace<-exp(-d$mu*(0:nmax))
# cumulative distribution of the exponential law

power.trace<-(1:(nmax+1))^(-d$gamma+1)
# cumulative distribution of the power law

gamma.trace<-1-pgamma((0:nmax),shape=d$alpha,scale=d$beta)
# cumulative distribution of the truncated power law

par(cex=1.5,cex.lab=1.2,font.lab=2)

plot(log(1:(nmax)),log(cum.dist),pch=3,xlab="log(k)",ylab="log(cumulative distribution)")
lines(log(1:(nmax+1)),log(exp.trace),lty=3,lwd=2)
lines(log(1:(nmax+1)),log(power.trace),lty=2,lwd=2)
lines(log(1:(nmax+1)),log(gamma.trace),lty=1,lwd=2)
#text(c(0.5,0.5,0.5,0.5),c(-3,-3.5,-4,-4.5),labels=c("+ data","-- power law",
#        ".. exponential law","- truncated power law"),pos=4)
```

`graph.attack` *Attack of graphs*

Description

Computes the evolution of the size of the largest connex component and its mean minimum path length under a random or targeted attack of a graph.

Usage

```
random.attack(adj.mat, max.remove, nsim)
node.attack(adj.mat, node.sup)
targeted.attack(adj.mat, max.remove)
```

Arguments

`adj.mat`	adjacency matrix of the graph.

max.remove	maximum number of nodes to be removed
nsim	number of simulation of random attack to be processed
node.sup	number of the node to be removed

Details

An attack consists in removing a node and all its connections from a given graph. random.attack removes max.remove nodes by selecting one regional node at random and removing it (and all its connections) from the graph. targeted.attack removes max.remove nodes : the first node to be eliminated was the hub with the largest degree, and nodes were subsequently eliminated in rank order of decreasing degree.

Value

size.large.connex
: vector containing the evolution of the size of the largest connexe component during an attack.

charac.path.length
: vector containing the evolution of the mean minimum path length of the largest connexe component during an attack.

Author(s)

S. Achard

References

Albert R., Barabasi A. L. (2002) Statistical mechanics of complex networks. *Rev Mod Physics* 74 pages 47-97.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

Examples

```
set2<-array(c(5,6.5,7,6.5,5,3.5,3,3.5,1,1.5,3,4.5,5,4.5,3,1.5),dim=c(8,2))
names<-c(1:8)

mat<-sim.rand(8,20)

#simulated attack on node 1
plot(set2[,1], set2[,2], type = "n",xlab="", ylab="",cex.lab=1.5)
text(set2[2:8,1], set2[2:8,2], names[2:8], cex = 1.5)
text(set2[1,1], set2[1,2], 1 , cex = 1.5,col="gray")

for(k in 2:8){
                for(q in 1:(k-1)){

                if(mat[k,q]==1)
```

```
                {
                if(q==1) visu <- "gray" else visu <- "red"
                lines(c(set2[k,1], set2[q,1]), c(set2[k,2], set2[q,2]), col = visu)
        }

}
}

ra<-random.attack(mat,8,10)
na<-node.attack(mat, 1)
ta<-targeted.attack(mat,8)
```

`p.value.compute` *Computation of the p-value for a given hypotheses test*

Description

Computes the p-values for all the entries in the matrix `test.mat` using the asymtotic properties of the estimator of the wavelet correlation given in (Whitcher, 2000).

Usage

```
p.value.compute(test.mat, var.ind.mat = 0, n.ind = 0, test.method = "gaussian",
                  proc.length, sup, num.levels, use.tanh = FALSE)
```

Arguments

`test.mat`	matrix containing the wavelet correlation to be tested
`var.ind.mat`	matrix containing the variance inter individuals of the correlation. Only used with `test.method="t.test"`. (default not used)
`n.ind`	number of individuals to take into account in the test. Only used with `test.method="t.test"`. (default not used)
`test.method`	name of the method to be applied. `"gaussian"` assumes a gaussian law for the estimator. `"t.test"` implements a t.test for computing the p-value. (default `"gaussian"`)
`proc.length`	specifies the length of the original processes using to construct the `cor.mat`
`num.levels`	specifies the number of the wavelet scale to take into account in the hypothesis test. Only used with `test.method="gaussian"`
`use.tanh`	logical. If FALSE take the `atanh` of the correlation values before applying the hypothesis test, in order to use the Fisher approximation
`sup`	indicates the correlation threshold to consider in each hypothesis test.

Details

Each hypothesis test is written as : H_0 : "|correlation| <= sup" H_1 : "|correlation| > sup" This function is essentially an internal function called by `const.adj.mat`.

Value

Vector with the p-value for each entry of the matrix.

Author(s)

S. Achard

References

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

codeconst.adj.mat

Examples

```
data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]


# Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 4,
                              boundary = "periodic", p.corr = 0.975)

# For scale 4
pvalue.cor<-p.value.compute(wave.cor.list[[4]],proc.length=dim(brain)[1], sup=0.44,
                            num.levels=4)
```

---

rand.eff *Efficiency for simulated graphs*

Description

Computes the local, global efficiency and cost for simulated random and regular graphs.

Usage

```
rand.eff(nsim, n.nodes.rand, n.edges.rand, dist.mat, dat = "reduced")
reg.eff(n.nodes.rand, n.edges.rand, dist.mat)
```

Arguments

nsim	number of simulated graphs to use for the computation of the small-world parameters.
dat	character string specifying if all the small-world parameters have to be returned. If "reduced", only the mean of the parameters for the whole graph is returned.
n.nodes.rand	number of nodes of the simulated graphs
n.edges.rand	number of edges of the simulated graphs
dist.mat	matrix with a distance associated to each pair of nodes of the graph to take into account in the computation of the efficiency values.

Value

eff	global efficiency for the whole graph
loc	local efficiency for th whole graph
cost	cost for th whole graph

Note

only in version 1.2 and after

Author(s)

S. Achard

References

V. Latora, M. Marchiori (2001) Efficient Behavior of Small-World Networks. *Phys. Rev. Lett.*, Vol. 87, N. 19, pages 1-4.

V. Latora, and M. Marchiori (2003) Economic Small-World Behavior in Weighted Networks. *Europ. Phys. Journ. B*, Vol. 32, pages 249-263.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

const.adj.list, small.world

Examples

```
result<-rand.eff(10,8,20,dist.mat=matrix(1,8,8))
result<-reg.eff(8,20,dist.mat=matrix(1,8,8))
```

`rand.sw` *Small-world parameters for simulated random graphs*

Description

Computes the degree, the minimum path length and the clustering coefficient for simulated random graphs.

Usage

```
rand.sw(nsim, n.nodes.rand, n.edges.rand, dist.mat, dat = "reduced")
```

Arguments

`nsim`	number of simulated graphs to use for the computation of the small-world parameters.
`dat`	character string specifying if all the small-world parameters have to be returned. If `"reduced"`, only the mean of the parameters for the whole graph is returned.
`n.nodes.rand`	number of nodes of the simulated graphs
`n.edges.rand`	number of edges of the simulated graphs
`dist.mat`	matrix with a distance associated to each pair of nodes of the graph to take into account in the computation of the small-world parameters.

Value

`in.degree`	mean of the degree for the whole graph.
`Lp.rand`	mean of the minimum path length for the whole graph.
`Cp.rand`	mean of the clustering coefficient for the whole graph.
`in.degree.all`	vector of the degree of each node of the graph.
`Lp.rand.all`	vector of the minimum path length of each node of the graph.
`Cp.rand.all`	vector of the clustering coefficient of each node of the graph.

Author(s)

S. Achard

References

S. H. Strogatz (2001) Exploring complex networks. *Nature*, Vol. 410, pages 268-276.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

equadist.rand.sw,reg.sw

Examples

```
mat<-sim.rand(8,20)

result<-rand.sw(10,8,20,dist.mat=matrix(1,8,8))
```

---

read.convert_1_2 *Conversion of already edited objects from version 1 to 2*

Description

Reads text files ("wave_cor_mat_level_[1:n.levels].txt") from version 1 and add the length of the time series in the object of class "Wave Correlation".

Usage

```
read.convert_1_2(proc.length)
```

Arguments

proc.length	specifies the length of the original processes using to construct the cor.mat~~Describe proc.length

Value

Object of class "Wave Correlation", basically, a list with the following components

d?	Correlation matrix for each scale of the wavelet decomposition.
lowerd?	matrix containing the lower bound of the correlation for each scale of the wavelet decomposition.
upperd?	matrix containing the upper bound of the correlation for each scale of the wavelet decomposition.

Note

only in version 1.2

Author(s)

S. Achard

See Also

read.cor.txt, save.cor.txt

---

read.cor.txt *Exportation and importation of internal objects.*

Description

Reads text files and imports them in object of class "Wave Correlation". Exports object of class "Wave Correlation".

Usage

```
read.cor.txt()
save.cor.txt(wave.cor.list)
```

Arguments

wave.cor.list	object of class "Wave Correlation" containing the correlation matrices to be exported.

Details

These two functions cannot be used separately. The names of the files used in read.cor.txt are given by the save.cor.txt functions

Value

For read.cor.txt function : Object of class "Wave Correlation", basically, a list with the following components

d?	Correlation matrix for each scale of the wavelet decomposition.
lowerd?	matrix containing the lower bound of the correlation for each scale of the wavelet decomposition.
upperd?	matrix containing the upper bound of the correlation for each scale of the wavelet decomposition.

Note

change between version 1 and 2, now the length of the time series and the number of time series are saved with the values of the correlation.

Author(s)

S. Achard

See Also

read.var.txt, save.var.txt

Examples

```
data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]



# Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 4,
                              boundary = "periodic", p.corr = 0.975,export.data=FALSE)

#Export the data
save.cor.txt(wave.cor.list)

#Import the data
read.cor.txt()
```

---

read.var.txt *Exportation and importation internal objects.*

Description

Reads text files and imports them in object of class "Wave Variance". Exports object of class "Wave Variance".

Usage

```
read.var.txt()
save.var.txt(wave.var.list)
```

Arguments

wave.var.list	object of class "Wave Variance" containing the vectors of variance to be exported.

Details

These two functions cannot be used separately. The names of the files used in read.var.txt are given by the save.var.txt functions

Value

For `read.var.txt` function : Object of class `"Wave Variance"`, basically, a list with the following components

`d?`	vectors of variance for each scale of the wavelet decomposition.
`lowerd?`	vector containing the lower bound of the variance for each scale of the wavelet decomposition.
`upperd?`	vector containing the upper bound of the variance for each scale of the wavelet decomposition.

Note

change between version 1 and 2, now the length of the time series and the number of the time series are saved with the values of the correlation.

Author(s)

S. Achard

See Also

`read.cor.txt`, `save.cor.txt`

Examples

```
data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]


# Construction of the correlation matrices for each level of the wavelet decomposition
wave.var.list<-const.var.list(brain, method = "modwt" ,wf = "la8", n.levels = 4,
                              boundary = "periodic", export.data=FALSE)

#Export the data
save.var.txt(wave.var.list)

#Import the data
read.var.txt()
```

reg.sw *Small-world parameters for a lattice graph*

Description

Computes the degree, the minimum path length and the clustering coefficient for a lattice graph.

Usage

```
reg.sw(n.nodes.rand, n.edges.rand, dist.mat)
```

Arguments

n.nodes.rand	number of nodes of the graph
n.edges.rand	number of edges of the graph
dist.mat	matrix with a distance associated to each pair of nodes of the graph to take into account in the computation of the small-world parameters.

Value

in.degree	mean of the degree for the whole graph.
Lp.rand	mean of the minimum path length for the whole graph.
Cp.rand	mean of the clustering coefficient for the whole graph.
in.degree.all	vector of the degree of each node of the graph.
Lp.rand.all	vector of the minimum path length of each node of the graph.
Cp.rand.all	vector of the clustering coefficient of each node of the graph.

Author(s)

S. Achard

References

S. H. Strogatz (2001) Exploring complex networks. *Nature*, Vol. 410, pages 268-276.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

See Also

rand.sw,equadist.rand.sw

Examples

```
mat<-sim.reg(8,20)
result<-reg.sw(8,20,dist.mat=matrix(1,8,8))
```

sim.graph *Simulation of graphs*

Description

Simulates four different types of graphs, random, lattice, scale-free and random with a given degree distribution.

Usage

```
sim.rand(n.nodes, n.edges)
sim.equadist(degree)
sim.reg(n.nodes, n.edges)
```

Arguments

n.nodes	number of nodes of the simulated graph
n.edges	number of edges of the simulated graph
degree	degree distribution of the simulated graph. Only for the sim.equadist function.

Details

The simulation of a graph with a given degree distribution is not always possible. Sometimes the random choice of the connected nodes will cause an impossible construction of the wanted graph with a given number of nodes and edges, because we do not allow to connect a node to itself. Becareful with this function and check always if the returned graph have the exact number of edges!

Value

A matrix containing the adjacency matrix of the simulated graph.

Author(s)

S. Achard

Examples

```
#Coordinates of the nodes of the graph

set2<-array(c(5,6.5,7,6.5,5,3.5,3,3.5,1,1.5,3,4.5,5,4.5,3,1.5),dim=c(8,2))
names<-c(1:8)


# For a random graph

mat<-sim.rand(8,20)
```

```
plot(set2[,1], set2[,2], type = "n",xlab="", ylab="",cex.lab=1.5)
text(set2[,1], set2[,2], names, cex = 1.5)

for(k in 2:8){
                for(q in 1:(k-1)){

                if(mat[k,q]==1)
                {

                visu <- "red"
                lines(c(set2[k,1], set2[q,1]), c(set2[k,2], set2[q,2]), col = visu)
        }

}
}

# For a lattice graph

mat<-sim.reg(8,20)

plot(set2[,1], set2[,2], type = "n",xlab="", ylab="",cex.lab=1.5)
text(set2[,1], set2[,2], names, cex = 1.5)

for(k in 2:8){
                for(q in 1:(k-1)){

                if(mat[k,q]==1)
                {

                visu <- "red"
                lines(c(set2[k,1], set2[q,1]), c(set2[k,2], set2[q,2]), col = visu)
        }

}
}

# For a graph with a given degree distribution

degree<-c(1,2,3,4,5,6,7,8)
mat<-sim.equadist(degree)

plot(set2[,1], set2[,2], type = "n",xlab="", ylab="",cex.lab=1.5)
text(set2[,1], set2[,2], names, cex = 1)

for(k in 2:8){
                for(q in 1:(k-1)){

                if(mat[k,q]==1)
                {

                visu <- "red"
                lines(c(set2[k,1], set2[q,1]), c(set2[k,2], set2[q,2]), col = visu)
```

```
        }

}
}

# For a scale-free graph

# Simulation of a scale-free degree distribution

x<-1:50
probx<-x^(-1.4)
n.nodes<-8
n.edges<-25
sf.degree<-rep(0,n.nodes)

stop<-0

while(stop==0){

r<-sample(x,n.nodes,prob=probx,replace=TRUE)
if(sum(r)==n.edges) stop<-1
  }

sf.degree<-r

mat<-sim.equadist(sf.degree)

plot(set2[,1], set2[,2], type = "n",xlab="", ylab="",cex.lab=1.5)
text(set2[,1], set2[,2], names, cex = 1)

for(k in 2:8){
                for(q in 1:(k-1)){

                if(mat[k,q]==1)
                {

                visu <- "red"
                lines(c(set2[k,1], set2[q,1]), c(set2[k,2], set2[q,2]), col = visu)
        }

}
}
```

`small.world` *Computation of some quantitive parameters given in the framework of the small-world graph theory.*

Description

For a given graph, computes the size of the largest connex component, degree, minimum path length and clustering coefficient.

Usage

```
small.world(wave.adj.mat, dat = "reduced", distance = "norm",
            coord = 0, export.data = FALSE)
```

Arguments

wave.adj.mat	adjacency matrix of the graph to be analysed.
dat	character, if dat = "all", the degree, minimum path length and clustering coefficient are computed for each node of the graph. If dat = "reduced", only the mean on all the nodes is computed.
distance	matrix with a distance associated to each pair of nodes of the graph to take into account in the computation of the small-world parameters. By default, the matrix of the distance has a value of 1 in each entry.
coord	optional vector containing the coordinate of the nodes of the graph.
export.data	logical. If TRUE the correlation matrices with the upper and lower bound are exported to text file.

Value

in.degree.mean	real corresponding to the mean of the degree of all the nodes of the graph.
in.degree	vector containing the degree of each node of the graph.
Lp.mean	real corresponding to the mean of minimum path length for each node belonging to the largest connex component of the graph only.
Lp	vector containing the mean minimum path length for each node of the graph. (the computaion of Lp requires at least one neighbourgh per node, if it is not the case the value of Lp is equal to -1)
Cp.mean	real corresponding to the mean of clustering coefficient for each node belonging to the largest connex component of the graph only.
Cp	vector containing the clustering coefficient for each node of the graph. (the computaion of Cp requires at least two neighbourghs per node, if it is not the case the value of Cp is equal to -1)
size.large.connex	real corresponding to the size of the largest connex component.

Author(s)

S. Achard

References

D. J. Watts and S. H. Strogatz (1998) Collective dynamics of "small-world" networks. *Nature*, Vol. 393, pages 440-442.

S. H. Strogatz (2001) Exploring complex networks. *Nature*, Vol. 410, pages 268-276.

S. Achard, R. Salvador, B. Whitcher, J. Suckling, Ed Bullmore (2006) A Resilient, Low-Frequency, Small-World Human Brain Functional Network with Highly Connected Association Cortical Hubs. *Journal of Neuroscience*, Vol. 26, N. 1, pages 63-72.

Examples

```
# fig 3 of Achard (2006)


data(brain)
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]



#Construction of the correlation matrices for each level of the wavelet decomposition
wave.cor.list<-const.cor.list(brain, method = "modwt" ,wf = "la8", n.levels = 6,
                              boundary = "periodic", p.corr = 0.975)

sup.seq<-((1:10)/10) #sequence of the correlation threshold
nmax<-length(sup.seq)
in.degree.mean<-matrix(0,6,nmax)
n.levels<-6

#For each value of the correlation thrashold
for(i in 1:nmax){
n.sup<-sup.seq[i]

#Construction of the adjacency matrices associated to each level of the wavelet decomposition
wave.adj.list<-const.adj.list(wave.cor.list, sup = n.sup, proc.length=dim(brain)[1])


#For each level of the wavelet decomposition
for(j in 1:n.levels){

param.sw.brain<-small.world(wave.adj.list[[j]],dat="reduced")
in.degree.mean[j,i]<-param.sw.brain$in.degree.mean

}}

#Plots of the average in-degree in terms of the scale

n.regions<-dim(brain)[2]

plot(sup.seq,(1:nmax)/2,type='n',xlab='Correlation threshold, R',ylab='Mean degree, k',
```

```
     cex.axis=2,cex.lab=2,xlim=c(0,1),ylim=c(0,90))


for(i in 1:n.levels){
lines(sup.seq,in.degree.mean[i,],type='l',col=i,lwd=2)
}
lines(sup.seq,rep(log(n.regions),nmax))
legend(x="topright",legend=c("Level 1","Level 2","Level 3","Level 4",
"Level 5","Level 6"),col=(1:n.levels),lwd=2)
```

`wave.trans` *Computation of the wavelet transform*

Description

Uses the wavelet decomposition implemented by Whitcher in the library `waveslim`. See all the details there.

Usage

```
wave.trans(x, method = "modwt", wf = "la8", n.levels = 4, boundary = "periodic")
```

Arguments

`x`	original vector to be decomposed
`method`	wavelet decomposition to be used, algorithm implemented in the `waveslim` package (Whitcher, 2000). By default, the Maximal Overlap Discrete Wavelet Transform is used `"modwt"`. It is also possible to use the classical Discrete Wavelet Transform `"dwt"`.
`wf`	name of the wavelet filter to use in the decomposition. By default this is set to `"la8"`, the Daubechies orthonormal compactly supported wavelet of length $L = 8$ (Daubechies, 1992), least asymmetric family.
`n.levels`	specifies the depth of the decomposition. This must be a number less than or equal to $\log_2(\text{length}(x))$.
`boundary`	Character string specifying the boundary condition. If `boundary=="periodic"` the default, then the vector you decompose is assumed to be periodic on its defined interval, if `boundary=="reflection"`, the vector beyond its boundaries is assumed to be a symmetric reflection of itself.

Details

See the library package `waveslim` (Whitcher, 2000).

Value

Object of class "modwt", basically, a list with the following components

d?	Wavelet coefficient vectors.
s?	Scaling coefficient vector.
wavelet	Name of the wavelet filter used.
boundary	How the boundaries were handled.

Author(s)

S. Achard

References

R. Gencay, F. Selcuk and B. Whitcher (2001) *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press.

D. B. Percival and A. T. Walden (2000) *Wavelet Methods for Time Series Analysis*, Cambridge University Press.

Examples

```
data(brain) # the result brain is a matrix
brain<-as.matrix(brain)

# WARNING : To process only the first five regions
brain<-brain[,1:5]

PreCG.R<-brain[,1]
# LA(8)
PreCG.R.la8 <- wave.trans(PreCG.R, wf="la8")
names(PreCG.R.la8) <- c("w1", "w2", "w3", "w4", "v4")
## plot partial MODWT for PreCG.R data
par(mfcol=c(6,1), pty="m", mar=c(5-2,4,4-2,2))
plot.ts(PreCG.R, axes=FALSE, ylab="", main="(a)")
for(i in 1:5)
  plot.ts(PreCG.R.la8[[i]], axes=FALSE, ylab=names(PreCG.R.la8)[i])
axis(side=1, at=seq(0,518,by=50),
  labels=c(0,"",100,"",200,"",300,"",400,"",500))
```

---

young *Time series obtained by an fMRI experiement on the brain*

Description

Time series for each region of interest in the brain. These series are obtained by SPM preprocessing from young healthy individual. The length of the time series for each region of the brain is equal to 518.

Usage

```
data(young)
```

Format

A data frame with 518 observations on the following 90 variables.

Source

contact S. Achard (sa428@cam.ac.uk)

References

to appear

Examples

```
data(young)
## maybe str(brain) ; plot(brain) ...
```

Index