

Package ‘ConNEcT’

January 20, 2025

Title Contingency Measure-Based Networks for Binary Time Series

Version 0.7.27

Maintainer Nadja Bodner <nadja.bodner@kuleuven.be>

Description The ConNEcT approach investigates the pairwise association strength of binary time series by calculating contingency measures and depicts the results in a network. The package includes features to explore and visualize the data. To calculate the pairwise concurrent or temporal sequenced relationship between the variables, the package provides seven contingency measures (proportion of agreement, classical & corrected Jaccard, Cohen's kappa, phi correlation coefficient, odds ratio, and log odds ratio), however, others can easily be implemented. The package also includes non-parametric significance tests, that can be applied to test whether the contingency value quantifying the relationship between the variables is significantly higher than chance level. Most importantly this test accounts for auto-dependence and relative frequency. See Bodner et al.(2021) <[doi:10.1111/bmsp.12222](https://doi.org/10.1111/bmsp.12222)>. Finally, a network can be drawn. Variables depicted the nodes of the network, with the node size adapted to the prevalence. The association strength between the variables defines the undirected (concurrent) or directed (temporal sequenced) links between the nodes. The results of the non-parametric significance test can be included by depicting either all links or only the significant ones. Tutorial see Bodner et al.(2021) <[doi:10.3758/s13428-021-01760-w](https://doi.org/10.3758/s13428-021-01760-w)>.

Imports qgraph, stats, graphics

Depends R (>= 2.10)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Suggests covr, lintr, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Nadja Bodner [aut, cre],
Eva Ceulemans [ctb, rev]

Repository CRAN

Date/Publication 2022-05-09 07:10:13 UTC

Contents

AttachmentData	2
barplot.conData	3
conData	4
conMx	5
conNEcT	6
conProf	7
conTest	8
FamilyData	9
funClassJacc	10
funCorrJacc	11
funKappa	11
funLogOdds	12
funOdds	13
funPhiCC	13
funPropAgree	14
getProb	15
hist.conTest	15
lagthemats	16
modelAD	17
modelNO	17
permutAD	18
permutNO	19
plot.conData	19
plot.conProf	20
qgraph.conNEcT	21
SymptomData	22
Index	23

AttachmentData	<i>Attachment-related mother-child interaction dataset</i>
----------------	--

Description

During an attachment study, a mother and her child (middle childhood; age between eight and 12) were videotaped while working on a three-minutes stressful puzzle task. The interaction was coded in two-second intervals for the presence and absence of positive, negative, and task related behavior. The dataset contains seven variables and 90 time intervals.

Usage

AttachmentData

Format

A data frame with 90 rows and 7 variables:

Mpos mother's positive behavior

Mneg mother's negative behavior

MAlone mother is working on the task alone

Togth mother and child are working on the task together

Cpos child's positive behavior

Cneg child's negative behavior

CAlone child is working on the task alone

Author(s)

Adinda Dujardin & Guy Bosmans <guy.bosmans@kuleuven.be>

References

Bodner, N., Bosmans, G., Sannen, J., Verhees, M., & Ceulemans, E. (2019). Unraveling middle childhood attachment-related behavior sequences using a micro-coding approach. *PLOS ONE*, 14(10), e0224372. <https://doi.org/10.1371/journal.pone.0224372>

barplot.conData	<i>Depict the relative frequencies (and conditional probabilities) of a binary time series in a barplot</i>
-----------------	---

Description

Depict the relative frequencies (and conditional probabilities) of a binary time series in a barplot

Usage

```
## S3 method for class 'conData'
barplot(height, plottype = "RelFreq", color = NULL, legend = TRUE, ...)
```

Arguments

height	conData object
plottype	needs to be specified if 'RelFreq' only the relative frequency is depicted if 'All' both the relative frequency and the conditional probabilities are depicted
color	string of chars containig as many colornames as variables in the object (default='gray')
legend	indicates whether you want to include a legend or not
...	parameters to be passed on to barplot

Value

barplot

Examples

```
ExampleData <- cbind(rep(c(0,1),100),
                    rep(c(0,0,0,0,0,1,1,1,1,1),20),
                    c(rep(c(0,0,0,1,1),20),rep(c(0,1,1,1,1),20)),
                    ifelse(rnorm(200,0,1)<0.95,1,0),
                    c(ifelse(rnorm(100,0,1)<0.7,1,0),ifelse(rnorm(100,0,1)<0.7,0,1)),
                    ifelse(rnorm(200,0,1)<(-0.98),1,0))
colnames(ExampleData) <- c('Var 1','Var 2','Var 3',
                          'Var 4','Var 5','Var 6')
fancy.col <- c('purple','slateblue','royalblue','cyan4',
              'green3','olivedrab3')
PersData <- conData(ExampleData)
barplot(PersData, plottype='RelFreq', color=fancy.col)
barplot(PersData, plottype='All', color=fancy.col)

data(SymptomData)
Sdata <- conData(SymptomData)
FANCY= c('purple','slateblue', 'royalblue', 'cyan4', 'green3',
        'olivedrab3', 'orange', 'orangered')
barplot(Sdata,plottype='RelFreq', color = FANCY)
```

conData

Explore and tidy raw data

Description

Removes not binary columns from multivariate time series data and calculates a table of relative frequency and auto-dependency for each binary variable

Usage

```
conData(data)
```

Arguments

data Binary time-points-by-variable matrix

Value

A conData-object including:

data Binary data in time points to variable format.

probs Table of relative frequency and auto-dependence for each variable.

varNames The names of all variables.

Examples

```

ExampleData <- cbind(rep(c(0,1),100),
                    rep(c(0,0,0,0,0,1,1,1,1,1),20),
                    c(
                      rep(c(0,0,0,1,1),20),
                      rep(c(0,1,1,1,1),20)
                    ),
                    ifelse(rnorm(200,0,1)<0.95,1,0),
                    c(
                      ifelse(rnorm(100,0,1)<0.7,1,0),
                      ifelse(rnorm(100,0,1)<0.7,0,1)
                    ),
                    ifelse(rnorm(200,0,1)<(-0.98),1,0))
colnames(ExampleData) <- c('Var 1', 'Var 2', 'Var 3',
                          'Var 4', 'Var 5', 'Var 6')
conData(ExampleData)

data(SymptomData)
Sdata <- conData(SymptomData)
Sdata$probs

```

conMx

*Calculate contingency measure values of a (lagged) time series matrix***Description**

Calculate contingency measure values of a (lagged) time series matrix

Usage

```
conMx(data, data2 = NULL, lag = 0, conFun)
```

Arguments

data	Binary time-points-by-variable matrix
data2	Second binary time-points-by-variable matrix (optional)
lag	Non-negative integer indicating how many time points the second variable is lagged (default 0)
conFun	Contingency measure function (calculating the contingency value between two binary vectors). Built in: funPropAgree, funClassJacc, funKappa, funCorrJacc, funOdds, funLogOdds, funPhiCC

Value

list with two elements:

value Matrix of pairwise calculated contingency measures

para Parameter settings lag, funName and varNames

Examples

```
conMx(cbind(c(1,0,1,0,1,0,1),c(1,1,1,1,0,0,0)),lag=1,conFun=funCorrJacc)
```

conNEcT	<i>Calculate the link strength between multiple behaviors and return them as a matrix (optionally discarding all non-significant links)</i>
---------	---

Description

Calculate the link strength between multiple behaviors and return them as a matrix (optionally discarding all non-significant links)

Usage

```
conNEcT(
  data,
  lag = 0,
  conFun,
  test = FALSE,
  typeOfTest = "permut",
  adCor = TRUE,
  nBlox = 10,
  nReps = 100,
  signLev = 0.05
)
```

Arguments

data	Binary time-points-by-variable matrix
lag	Non-negative integer indicating how many time points the second variable is lagged (default 0)
conFun	Contingency measure function (calculating the contingency value between two binary vectors). Built in: funPropAgree, funClassJacc, funKappa, funCorrJacc, funOdds, funLogOdds, funPhiCC
test	Logic indicating whether a significance test is executed (TRUE) or not (FALSE;default)
typeOfTest	String indicating whether a model-based ('model') or a permutation-based ('permut'; default) data generation approach is used.
adCor	Logic indicating the auto-dependence correction should be applied (TRUE; default) or not (FALSE)
nBlox	Number indicating the number of segments (default 10). Necessary for permutation-based test, accounting for auto-dependence (typeOfTest='permut'; adCor=TRUE)
nReps	Number of replicates/samples that is used to generate the test distribution
signLev	Significance level of the test (default 0.05)

Value

A conNEcT-object including

- allLinks Matrix of pairwise calculated contingency measures
- signLinksMatrix of pairwise calculated contingency measures containing only significant links (others are set to 0)
- pValue P-values for the one-sided upper significance test
- para Parameter settings containing lag, test, typeOfTest, adCor, nBlox, nReps, funName, and varNames
- probs Table of relative frequency and auto-dependency

Examples

```
netdata=cbind(rep(c(1,1,1,1,1,0,0,0,0),100),
rep(c(0,0,1,1,1,1,0,0,0),100))
conNEcT(netdata,lag=1,conFun=funKappa,test=TRUE,nBlox=5)
```

conProf

*Compare different lags in a contingency profile***Description**

Compare different lags in a contingency profile

Usage

```
conProf(data, maxlag, conFun)
```

Arguments

data	Binary time-points-by-variable matrix
maxlag	Positive integer indicating how many lags should be investigated
conFun	Contingency measure function (calculating the contingency value between two binary vectors). Built in: funPropAgree, funClassJacc, funKappa, funCorrJacc, funOdds, funLogOdds, funPhiCC

Value

A conProf-object consisting of

- value Contingency matrices for different lags
- para Parameters including maxlag, funName and varNames

Examples

```
IntData <- cbind(rep(rep(c(0,0,1,0,1,0,1,0,0,0),each=5),times=5),
                rep(rep(c(1,0,0,0), each=10), times=25))
colnames(IntData) <- c('Var1','Var2')
conProf(IntData,maxlag=10,conFun=funClassJacc)
```

conTest

*Test significance***Description**

Test significance

Usage

```
conTest(
  data,
  lag = 0,
  conFun,
  typeOfTest = "permut",
  adCor = TRUE,
  nBlox = 10,
  nReps = 100
)
```

Arguments

data	Binary time-points-by-variable matrix
lag	Non-negative integer indicating how much the second variable is lagged (default 0)
conFun	Contingency measure function (calculating the contingency value between two binary vectors). Built in: FunClassJacc, FunCorrJacc, FunKappa, FunOdds, FunLogOdds, FunPropAgree, FunPhiCC
typeOfTest	String indicating whether a model-based ('model') or a permutation-based ('permut'; default) data generation approach is used.
adCor	Logic indicating the auto-dependence correction should be applied (TRUE; default) or not (FALSE)
nBlox	Number indicating the number of segments (default 10). Necessary for permutation-based test, accounting for auto-dependence (typeOfTest='permut'; adCor=TRUE)
nReps	Number of replicates/samples that is used to generate the test distribution

Value

A conTest-object including

allLinks Matrix of pairwise calculated contingency measures

percentile Matrix of raw percentiles, situating the observed value in the sample distribution

pValue Matrix of the p-values (upper one-sided significance test) calculated by subtracting the percentile from 1.

para: Saving the parameter settings for typeOfTest, adCor, nBlox, nReps, funName, lag, varNames

samples Saved generated replicates/samples for each variable combination under \$NameVariable1\$NameVariable2

Examples

```
signdata=cbind(c(1,0,1,0,1,0,1,0),c(1,1,1,1,0,0,0,0),c(0,0,0,0,0,0,1,1))
colnames(signdata) <-c ('momangry', 'momsad', 'adoangry')
conTest(data=signdata, lag=1, conFun=funClassJacc, typeOfTest='model',
adCor=FALSE)
```

FamilyData

Affective family interaction dataset

Description

These data was collected during a nine-minutes problem-solving family interaction between two parents and their adolescent son or daughter During this interaction, the presence and absence of expressions of ‘anger’, ‘dysphoric’ feelings and ‘happiness’ were coded for each family member in an event-basis way (i.e., noting when a certain behavior starts and when it stops). The codes were subsequently restructured into second-to-second interval data, resulting in a 540 seconds by nine variables binary dataset.

Usage

FamilyData

Format

A data frame with 540 rows and 9 variables:

moanger mother expressing anger

faanger father expressing anger

adanger adolescent expressing anger

modysph mother expressing dysphoric feelings

fadysph father expressing dysphoric feelings

addysph adolescent expressing dysphoric feelings

mohappy mother expressing happy feelings

fahappy father expressing happy feelings

adhappy adolescent expressing happy feelings

Author(s)

Lisa Sheeber <lshieber@ori.org>

References

Sheeber, L. B., Kuppens, P., Shortt, J. W., Katz, L. F., Davis, B., & Allen, N. B. (2012). Depression is associated with the escalation of adolescents' dysphoric behavior during interactions with parents. *Emotion*, 12(5), 913–918. <https://doi.org/10.1037/a0025784>

Allen, N. B., Kuppens, P., & Sheeber, L. B. (2012). Heart rate responses to parental behavior in depressed adolescents. *Biological Psychology*, 90(1), 80–87. <https://doi.org/10.1016/j.biopsycho.2012.02.013>

Bodner, N., Kuppens, P., Allen, N. B., Sheeber, L. B., & Ceulemans, E. (2018). Affective family interactions and their associations with adolescent depression: A dynamic network approach. *Development and Psychopathology*, 30(4), 1459–1473. <https://doi.org/10.1017/S0954579417001699>

funClassJacc

Calculate the classic Jaccard index between two vectors

Description

Calculate the classic Jaccard index between two vectors

Usage

```
funClassJacc(vec1, vec2)
```

Arguments

vec1 Vector of binary time series (no missing values)

vec2 Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements

value of the classic Jaccard index and

funName name of the function

Examples

```
data1<-rep(c(1,0,1,1),25)
data2<-ifelse(rnorm(100,0,1)<0.7,0,1)
funClassJacc(data1,data2)
```

funCorrJacc	<i>Calculate the corrected Jaccard index between two vectors</i>
-------------	--

Description

Calculate the corrected Jaccard index between two vectors

Usage

```
funCorrJacc(vec1, vec2)
```

Arguments

vec1	Vector of binary time series (no missing values)
vec2	Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements
value of the corrected Jaccard index and
funName name of the function

Examples

```
data1<-rep(c(1,0,1,1),25)  
data2<-ifelse(rnorm(100,0,1)<0.7,0,1)  
funCorrJacc(data1,data2)
```

funKappa	<i>Calculate Cohen's kappa between two vectors</i>
----------	--

Description

Calculate Cohen's kappa between two vectors

Usage

```
funKappa(vec1, vec2)
```

Arguments

vec1	Vector of binary time series (NA not allowed)
vec2	Vector of binary time series (equal length as vec1, NA not allowed)

Value

list with two elements
value of the Cohen's kappa and
funName name of the function

Examples

```
data1<-rep(c(1,0,1,1),25)
data2<-ifelse(rnorm(100,0,1)<0.7,0,1)
funKappa(data1,data2)
```

funLogOdds

Calculate the log odds ratio between two vectors

Description

Calculate the log odds ratio between two vectors

Usage

```
funLogOdds(vec1, vec2)
```

Arguments

vec1 Vector of binary time series (no missing values)
vec2 Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements
value of the log odds ratio and
funName name of the function

Examples

```
data1 <- rep(c(1,0,1,1),25)
data2 <- ifelse(rnorm(100,0,1)<0.7,0,1)
funLogOdds(data1,data2)
```

funOdds	<i>Calculate the odds ratio between two vectors</i>
---------	---

Description

Calculate the odds ratio between two vectors

Usage

```
funOdds(vec1, vec2)
```

Arguments

vec1	Vector of binary time series (no missing values)
vec2	Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements
value of the odds ratio and
funName name of the function

Examples

```
data1 <- rep(c(1,0,1,1),25)
data2 <- ifelse(rnorm(100,0,1)<0.7,0,1)
funOdds(data1,data2)
```

funPhiCC	<i>Calculate the phi correlation coefficient index between two vectors</i>
----------	--

Description

Calculate the phi correlation coefficient index between two vectors

Usage

```
funPhiCC(vec1, vec2)
```

Arguments

vec1	Vector of binary time series (no missing values)
vec2	Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements
value of the phi correlation coefficient and
funName name of the function

Examples

```
set.seed(1234)
data1<-rep(c(1,0,1,1),25)
data2<-ifelse(rnorm(100,0,1)<0.7,0,1)
funPhiCC(data1,data2)
```

funPropAgree

Calculate the proportion of agreement between two vectors

Description

Calculate the proportion of agreement between two vectors

Usage

```
funPropAgree(vec1, vec2)
```

Arguments

vec1 Vector of binary time series (no missing values)
vec2 Vector of binary time series (equal length as vec1, no missing values)

Value

list with two elements
value of the proportion of agreement and
funName name of the function

Examples

```
data1 <- rep(c(1,0,1,1),25)
data2 <- ifelse(rnorm(100,0,1)<0.7,0,1)
funPropAgree(data1,data2)
```

getProb	<i>Retrieve (conditional) probabilities from binary time series</i>
---------	---

Description

Retrieve (conditional) probabilities from binary time series

Usage

```
getProb(ts)
```

Arguments

ts	Binary time series vector
----	---------------------------

Value

List of three elements

p1 the prevalence $p(X(t)=1)$ and

$p1|1$ and $p1|0$ the two auto-conditional probabilities $p(X(t)=1|X(t-1)=1)$ & $p(X(t)=1|X(t-1)=0)$

Examples

```
getProb(c(1,0,1,0,1,0,1,1,1,1,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0))
```

hist.conTest	<i>Plot histogram matrix of the significance test</i>
--------------	---

Description

Plot histogram matrix of the significance test

Usage

```
## S3 method for class 'conTest'
hist(x, signLev = 0.05, ...)
```

Arguments

x	Object of the class conTest
signLev	Significance level (default .05)
...	Graphical parameters to be passed to hist()

Value

Histogram matrix with sample distribution and value from observed data for each variable combination

Examples

```
data(SymptomData)
test.result <- conTest(SymptomData[,c(2,6,8)], conFun=funClassJacc,
  typeOfTest='permut', nBlox=10, adCor=TRUE, nReps=100)
hist(test.result)
```

 lagthemats

Lag a matrix

Description

Lag a matrix

Usage

```
lagthemats(data, lag)
```

Arguments

data	Binary time-points-by-variable matrix
lag	Non-negative integer indicating the number of time points the second variable is lagged (default 0)

Value

laggeddata Matrix in which all variables are lagged lag time points

Examples

```
lagthemats(cbind(c(1,0,1,0,1,0,1),c(1,1,1,1,0,0,0)), lag=2)
```

modelAD	<i>Generate data with model-based approach accounting for auto-dependence</i>
---------	---

Description

This function generates a new time serie that is similar to the original one in relative frequency and auto-dependence, by drawing samples time point per time point from a Bernoulli distribution with the different conditional probabilities as parameter, depending on the state of the previous time point.

Usage

```
modelAD(vec)
```

Arguments

vec	Time series vector
-----	--------------------

Value

Time series vector that is similar to the original one in relative frequency and auto-dependence

Examples

```
ts=rep(c(1,1,1,1,1,0,0,0,0,0),15)
modelAD(ts)
```

modelNO	<i>Generate data with model-based approach ignoring auto-dependence</i>
---------	---

Description

This function generates a new time serie that is similar to the original one in relative frequency, but not in auto-dependence by drawing from a Bernoulli distribution with the relative frequency as parameter.

Usage

```
modelNO(vec)
```

Arguments

vec	Time series vector
-----	--------------------

Value

Time series vector that is similar to the original one considering relative frequency

Examples

```
ts=rep(c(1,1,1,1,1,0,0,0,0,0),15)
modelNO(ts)
```

permutAD	<i>Generate data with permutation-based approach accounting for auto-dependence</i>
----------	---

Description

This function generates a new time serie with exactly the same relative frequency as the original one, and a similar auto-dependence by cutting the original variable in segments and shuffling these segements.

Usage

```
permutAD(vec, nBlokS = 10)
```

Arguments

vec	Time series vector
nBlokS	positive integer indicating the number of segments/blocks (default 10)

Value

Time series vector with exactly the same relative frequency and a similar auto-dependence as the original vector

Examples

```
ts=rep(c(1,1,1,1,1,0,0,0),15)
permutAD(ts,nBlokS=11)
```

permutNO	<i>Generate data with permutation-based approach ignoring auto-dependence</i>
----------	---

Description

This function generates a new time serie with exactly the same relative frequency as the original one, but with a lower auto-dependence by shuffling all time points.

Usage

```
permutNO(vec)
```

Arguments

vec	Time series vector
-----	--------------------

Value

Time series vector with exactly the same relative frequency as the original vector

Examples

```
ts=rep(c(1,1,1,1,1,0,0,0,0),15)
permutNO(ts)
```

plot.conData	<i>Visualize the course of the variables over time</i>
--------------	--

Description

Visualize the course of the variables over time

Usage

```
## S3 method for class 'conData'
plot(x, plottype = "interval", color = NULL, ...)
```

Arguments

x	conData object
plottype	Character specified as 'interval', 'line', or 'both'
color	Character string of colornames for all variables (default='black')
...	Parameters to be transfered to the plot function

Value

Plot visualizing the course of the variables over time

Examples

```
ExampleData <- cbind(rep(c(0,1),100),
                    rep(c(0,0,0,0,0,1,1,1,1,1),20),
                    c(rep(c(0,0,0,1,1),20),
                      rep(c(0,1,1,1,1),20)),
                    ifelse(rnorm(200,0,1)<0.95,1,0),
                    c(
                      ifelse(rnorm(100,0,1)<0.7,1,0),
                      ifelse(rnorm(100,0,1)<0.7,0,1)
                    ),
                    ifelse(rnorm(200,0,1)<(-0.98),1,0))
colnames(ExampleData) <- c('Var 1', 'Var 2', 'Var 3',
                          'Var 4', 'Var 5', 'Var 6')
PersData <- conData(ExampleData)
fancy.col=c('purple', 'slateblue', 'royalblue', 'cyan4',
            'green3', 'olivedrab3', 'orange', 'orangered')
plot(PersData, plottype='line', color=fancy.col)

data(SymptomData)
Sdata <- conData(SymptomData)
fancy.col=c('purple', 'slateblue', 'royalblue', 'cyan4',
            'green3', 'olivedrab3', 'orange', 'orangered')
plot(Sdata, plottype='interval', color=fancy.col)
```

plot.conProf

Draw contingency profiles

Description

Draw contingency profiles

Usage

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

Arguments

x Object of a conProf class
 ... Parameters to be transferred to the plot function

Value

Contingency profile matrix

Examples

```
IntData <- cbind(rep(rep(c(0,0,1,0,1,0,1,0,0,0),each=5),times=5),
                rep(rep(c(1,0,0,0), each=10), times=25))
colnames(IntData) <- c('Var1','Var2')
CP <- conProf(IntData,maxlag=10,conFun=funClassJacc)
plot(CP)
```

qgraph.conNEcT	<i>Draws a Network figure</i>
----------------	-------------------------------

Description

Draws a Network figure

Usage

```
qgraph.conNEcT(x, signOnly = TRUE, ...)
```

Arguments

x	A conNEcT-object
signOnly	Logic indicating whether significant links should be depicted (TRUE; default) or not (FALSE)
...	Parameter settings for qgraph

Value

A network plot

Examples

```
AD0angry=rep(c(1,1,1,1,1,0,0,0,0),100)
MAangry=rep(c(0,0,1,1,1,1,0,0,0),100)
MASad=rep(c(0,0,1,1,1,1,0,0,0),100)
netdata <- cbind(AD0angry,MAangry,MASad)
netnet <- conNEcT(netdata,lag=1,conFun=funKappa,
                  test=TRUE,nBlox=5)
qgraph.conNEcT(netnet,signOnly=FALSE)
```

SymptomData

Depression symptom dataset

Description

In this depression symptoms data, a patient reported for each week on the presence and absence of eight depression symptoms. The dataset contains the reports of the eight variables for 145 subsequent weeks.

Usage

SymptomData

Format

A data frame with 145 rows and 8 variables:

Core core depression symptoms including depressed mood and/or diminished interest

Energy lack of energy

Eat eating problems and weight loss or gain

Sleep sleeping problems including hyposomnia or hypersomnia

Motor psychomotor problems

Guilt feelings of guilt

Cogn cognitive problems

Death preoccupation with death

Author(s)

Bettina Hosenfeld & Peter de Jonge <peter.de.jonge@rug.nl>

References

Hosenfeld, B., Bos, E. H., Wardenaar, K. J., Conradi, H. J., van der Maas, H. L. J., Visser, I., & de Jonge, P. (2015). Major depressive disorder as a nonlinear dynamic system: Bimodality in the frequency distribution of depressive symptoms over time. *BMC Psychiatry*, 15(1), 222. <https://doi.org/10.1186/s12888-015-0596-5>

Bodner, N., Bringmann, L., Tuerlinckx, F., De Jonge, P., & Ceulemans, E. (in press). ConNEcT: A novel network approach for investigating the co-occurrence of binary psychopathological symptoms over time. *Psychometrika*. <https://doi.org/10.1007/s11336-021-09765-2>

Index

* datasets

AttachmentData, 2
FamilyData, 9
SymptomData, 22

AttachmentData, 2

barplot.conData, 3

conData, 4

conMx, 5

conNEcT, 6

conProf, 7

conTest, 8

FamilyData, 9

funClassJacc, 10

funCorrJacc, 11

funKappa, 11

funLogOdds, 12

funOdds, 13

funPhiCC, 13

funPropAgree, 14

getProb, 15

hist.conTest, 15

lagthemats, 16

modelAD, 17

modelNO, 17

permutAD, 18

permutNO, 19

plot.conData, 19

plot.conProf, 20

qgraph.conNEcT, 21

SymptomData, 22