Package 'knnIndep'

February 20, 2015

Type Package

Title Independence tests and benchmarks

Version 2.0

Date 2014-09-09

Encoding UTF-8

Author Sebastian Dümcke <duemcke@mpipz.mpg.de>

Maintainer Sebastian Dümcke <duemcke@mpipz.mpg.de>

Description This package provides the implementation of an exact formula of the ith nearest neighbour distance distribution and implementations of tests of independence based on that formula. Furthermore the package provides a general framework to benchmark tests of independence.

Imports parallel

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-11 17:03:09

R topics documented:

knnIndep-package *A package giving the formulas of an exact distribution of ith nearest neighbours and two associated tests for independence*

Description

This package provides the formulas to calculate the probability of observing the ith nearest neighbour given the (i-1)th nearest neighbour. Additionally this formulas is used in independence testing and this package provides implementations for two tests of independence `novelTest.chisq` and `novelTest.extreme`.

This package also provides a mean to benchmark test for independence on many different type of functional dependences and a new type of non-functional dependence.

Details

Package:	knnIndep
Type:	Package
Version:	1.0
Date:	2014-03-06
License:	GPL>=3.0

For benchmarking purposes refer to `run.tests` and `generate.benchmark.data`. The formula is given by `P_ceq`,`P_cge_ale` and `Pc_givena`. The two tests of independence are `novelTest.chisq` and `novelTest.extreme`.

Author(s)

Sebastian Dümcke <`duemcke@mpipz.mpg.de`>

The author is also the maintainer.

benchmark.patchwork.copula *Benchmark function for a new type of non-functional dependence*

Description

This function is used to benchmark test for independence on a new type of non-function dependence called patchwork copula

Usage

```
benchmark.patchwork.copula(fun, args, cvals, n = 320, nsim = 500, bins = 20)
```

Arguments

`fun`	function or character naming a function. A function should have two vectors of coordinates as first two arguments
`args`	list of additional arguments to the functions `fun`. If a function does not need any arguments use an empty list.
`cvals`	target mutual information values vector of concentration factors, these represent mutual information values (see `generate.patchwork.copula`)
`n`	numeric, size of the data sets to generate (default 320 points)
`nsim`	numeric, how many replicate simulations to run under the null model and H1, default 500
`bins`	decimal, number of bins of the bins*bins grid, (see `generate.patchwork.copula`)

Value

This function returns a list data structure that can be further processed with the functions of this package, `calculate.power`,`generate.roc`

Author(s)

Sebastian Dümcke <`duemcke@mpipz.mpg.de`>

See Also

`calculate.power`,`generate.roc`

Examples

```
mycor = function(...) cor(...)^2
copula.vals = benchmark.patchwork.copula(mycor,list(),c(.3,1,10))
drop(calculate.power(copula.vals,.95))
roc.plot(generate.roc(copula.vals))
```

calculate.power *Calculate power at a given significance level*

Description

Function to calculate power at a given significance level. Uses the data structure returned by run.tests

Usage

```
calculate.power(vals, alpha = 0.95, comp = `>`)
```

Arguments

vals	list, values as returned by run.tests
alpha	significance level at which to return power
comp	comparison function, for alpha < .5, it should probably be set to ‘<‘

Details

power is calculated as the fraction of tests that are higher or lower than (according to comp) than the significance level. The significance level is fixed on data generated under the null hypothesis.

Value

returns the power for applicable data from the structure vals, usually for each test it returns the power for all types of dependence and all noise levels.

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

run.tests

Examples

```
mycor = function(...) cor(...)^2
vals = run.tests(mycor,list(),1:2,cbind(c(.3,.4,6),c(.3,.5,4)),100)
drop(calculate.power(vals))
```

`generate.benchmark.data`

Generating functional dependencies

Description

Generate functional dependencies for benchmarking tests of independence. This function can generate 8 types of functional dependence: linear, quadratic, cubic, two sine functions, x^(1/4), step function and a circular dependence.

Usage

```
generate.benchmark.data(typ, noises, n, project = FALSE, windx = 1, windy = 1)
```

Arguments

`typ` decimal, which type of dependence to generate. 1: linear 2: quadratic 3: cubic 4: sine period pi/4 5: sine period pi/16 6: x^(1/4) 7: circle 8: step function

`noises` vector of noise values to apply to the generated dependence. The noise is normally distributed.

`n` decimal, size of sample to return.

`project` boolean (default FALSE), wether to project the generated dependence onto a torus

`windx` decimal, how many times the dependence should wind around the torus in x-direction. Only used if `project` is `TRUE`

`windy` decimal, how many times the dependence should wind around the torus in y-direction. Only used if `project` is `TRUE`

Value

list with two elements

`x` matrix of x-coordinates, each column corresponds to a noise level from `noises`

`y` matrix of y-coordinates, each column corresponds to a noise level from `noises`

Author(s)

Sebastian Dümcke `<duemcke@mpipz.mpg.de>`

See Also

`generate.patchwork.copula` for generating non-functional dependence and `run.tests` for benchmarking tests of independence

Examples

```
#generate a quadratic dependence of 10 points with two noise levels 0.3 and 0.6
generate.benchmark.data(2,c(.3,.6),10)
plot(generate.benchmark.data(4,.2,1000))
```

---

generate.patchwork.copula

Generate data from a non-functional dependence

Description

Generate data from a non-functional dependence called 'patchwork copula'. Like a copula the data is uniform in x and y but it has a dependence between x and yy that has a block like structure

Usage

```
generate.patchwork.copula(p = matrix(rbeta(bins * bins, alpha, beta), ncol = bins),
 alpha = 0.01, beta = 1, c = 1, npoints = 320, bins = 20, returnmi = FALSE,
 plot = FALSE)
```

Arguments

`p`	matrix, starting mass distribution on the grid
`alpha`	decimal, parameter of beta distribution used for `p` (if `p` left as per default)
`beta`	decimal, parameter of beta distribution used for `p` (if `p` left as per default)
`c`	decimal, concentration factor (default `1`), used to stabilize mutual information estimation
`npoints`	decimal, sample size
`bins`	decimal, number of bins of the bins*bins grid
`returnmi`	boolean, whether to return the mutual information
`plot`	boolean, whether to plot the dependence

Value

list with the following elements

`x`	matrix of x-coordinates, each column corresponds to a noise level from `noises`
`y`	matrix of y-coordinates, each column corresponds to a noise level from `noises`
`mi`	mutual information of the dependence, only return if `returnmi` is set to `TRUE`

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

Examples

```
generate.patchwork.copula(bins=20,plot=TRUE)
```

`generate.paths` *Generate all nearest neighbours distances for one point in a sample*

Description

Help function which generates the nearest neighbhour distances for a single point in a sample, assuming rank data on a torus with the maximum distance.

Usage

```
generate.paths(index, rx, ry, N)
```

Arguments

`index`	for which point to calculate the nearest neighbhour distances
`rx`	ranked data (1st dimension)
`ry`	ranked data (2nd dimension)
`N`	Number of points in sample

Value

a vector of length (N-1) containing the sorted distances to the nearest neighbour of point index in the sample

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

Examples

```
x=rank(runif(10))
y=rank(runif(10))
knnIndep:::generate.paths(5,x,y,10)
#for all points in the sample
sapply(1:10,knnIndep:::generate.paths,x,y,10)
```

generate.roc *Generate ROC curve data*

Description

Generate data suitable for ROC curve plotting from the results of run.tests

Usage

```
generate.roc(vals, pval = TRUE)
```

Arguments

vals	list, data structure as returned by run.tests
pval	boolean, whether the values in vals represent pvalues

Details

calculates the power via calculate.power for all significance levels from 0 to 1.

Value

array of values suitable for plotting via roc.plot

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

run.tests and roc.plot for plotting

Examples

```
noises <- cbind(lin=c(.1,.5,.8),circ=c(.2,.4,.6))
mycor <- function(...) cor(...)^2
results.cor <- run.tests(mycor,args=list(),types=c(1,7),noises=noises,nsim=100,size=50)
roc.data <- generate.roc(results.cor,pval=FALSE)
roc.plot(roc.data,legend=noises)
```

novelTest.chisq *A novel test of independence*

Description

This function implements a novel test of independence of bivariate data. It is based on the formula of the exact distribution of the ith nearest neighbour given the previous nearest neighbour (see Pc_givena).

Usage

```
novelTest.chisq(xdata, ydata, maxi = length(xdata) - 1)
```

Arguments

xdata	first dimension of data
ydata	second dimension of data
maxi	up to which ith nearest neighbour to consider

Value

This function returns an object of class htest with:

statistic	The value of the statistic
p.value	p-value of the test

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

Examples

```
set.seed(10)
xylist = generate.benchmark.data(7,.3,100)
x = runif(100)
novelTest.chisq(x,xylist$y,maxi=20)
novelTest.chisq(xylist$x,xylist$y,maxi=20)
```

novelTest.extreme *A novel test of independence*

Description

This function implements a novel test of independence of bivariate data. It is based on the formula of the exact distribution of the ith nearest neighbour given the previous nearest neighbour (see Pc_givena).

Usage

```
novelTest.extreme(xdata, ydata, maxi = length(xdata) - 1)
```

Arguments

xdata	first dimension of data
ydata	second dimension of data
maxi	up to which ith nearest neighbour to consider

Value

This function returns the aggregated test statistic

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

Examples

```
set.seed(10)
xylist = generate.benchmark.data(7,.3,50)
x = runif(50)
novelTest.extreme(x,xylist$y,maxi=20)
novelTest.extreme(xylist$x,xylist$y,maxi=20)
```

optimise.copula.mi *optimize the parameter* c *of* generate.patchwork.copula

Description

Find the correct c parameter for the patchwork copula (generate.patchwork.copula) to reach a certain mutual information value

Usage

```
optimise.copula.mi(mis, distribution, interval = c(-10, 5), npoints)
```

Arguments

`mis`	traget mutual information values
`distribution`	matrix. Choices of alpha and beta parameter of `generate.patchwork.copula` e.g. `matrix(rbeta(bins*bins,.01,1),ncol=bins)`
`interval`	search interval for solution
`npoints`	sample size

Value

vector of values to be used as concentration facor `c` in `generate.patchwork.copula` to achive the input MI value

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

`generate.patchwork.copula`

Examples

```
bins=10
knnIndep:::optimise.copula.mi(c(0.001,.01,.5,2),matrix(rbeta(bins*bins,.01,1),ncol=bins),npoints=10)
```

`parameters` *Central probabilty*

Description

Probability of observing r NN distances at distance c, all previous NN distances at distance < c and all following NN distances at a distance > c

Usage

```
parameters(r, i0, c, N)
kr(r, i0, c)
```

Arguments

`r`	the number of points that are at the same distance c
`i0`	which i0-th nearest neighbour we are considering.
`c`	the distance of the i-th nearest neighbour
`N`	sample size

Value

for `kr` the number of possibilities to place r points onto the same distance when we already observed i0 points at a smaller distance

for `parameters` the probability of observing r NN distances at distance c, all previous NN distances at distance < c and all following NN distances at a distance > c

Author(s)

Sebastian Dümcke <`duemcke@mpipz.mpg.de`>

Examples

```
knnIndep:::kr(3,5,6)
knnIndep:::parameters(3,5,6,20)
```

`Pc_givena` *Probability of observing the ith nearest neighbour at distance greater or equal c given the (i-1)th nearest neighbour at distance a*

Description

This function gives the probability of observing the ith nearest neighbour at distance c given the (i-1)th nearest neighbour at distance a, $P(d_i >= x | d_(i-1) = a)$

Usage

```
Pc_givena(i, c, a, N)
```

Arguments

`i`	numeric, which nearest neighbour to consider
`c`	vector, the distance at which the ith NN was observed
`a`	vector, the distance at which the (i-1)th NN was observed, a <= c
`N`	numeric, size of the dataset

Details

The probability is calculated by ranking the data and assuming that the data lie on a torus. For details see Dümcke et al. "A novel test for independence derived from an exact distribution of ith nearest neighbours" (manuscript in preparation)

Value

Probability vector

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

P_cge_ale, P_ceq

Examples

```
Pc_givena(10,2:7,1:6,20)
```

Pc_givena4nn *Probability of observing the ith nearest neighbour at distance greater or equal c given the 4 previous nearest neighbours*

Description

This function gives the probability of observing the ith nearest neighbour at distance c given the previous 4 nearest neighbour distances, $P(d_i >= x | d_(i-1), d_(i-2), d_(i-3), d_(i-4))$

Usage

```
Pc_givena4nn(i, c, a, k1, k2, N)
```

Arguments

i	numeric, which nearest neighbour to consider
c	vector, the distance at which the ith NN was observed
a	vector, the distance at which the (i-1)th NN was observed, a <= c
k1	vector, number of previous neighbhour at distance d_i
k2	vector, number of preivous neighbhours at distance d_(i-1)
N	numeric, size of the dataset

Details

The probability is calculated by ranking the data and assuming that the data lie on a torus. For details see Dümcke et al. "A novel test for independence derived from an exact distribution of ith nearest neighbours" (PLoS ONE 2014)

Value

Probability vector

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

P_cge_ale, P_ceq

Examples

```
Pc_givena4nn(10,2:7,1:6,rep(0,6),rep(1,6),20)
```

power.plot *Plot power of benchmarked tests of independence*

Description

This functions plots the results of the benchmark. Input are the estimated powers at a certin significance level from calculate.power.

Usage

```
power.plot(powers, num.noise = seq(from = 0.1, to = 3, by = 0.1), mains = c("Linear",
"Quadratic", "Cubic", expression("Sine: period 4" * pi),
expression("Sine: period 16" * pi), "X^(1/4)", "Circle", "Step function",
 "Torus"), col = c("black", "red", "blue", "green", "cyan", "brown", "pink"),
 labels = TRUE,which = 1:nrow(powers[[1]]), show.legend = "bottomright")
```

Arguments

powers	named list of matrices one for each method with dimension, with one row for each type of dependence and a column for each noise level
num.noise	matrix, noise levels at which the test were run (see run.tests)
mains	character vector, title of each dependence type
col	character vector, specify the colours, one for each test
labels	labels to plot at the x axis, or TRUE (default) for standard label plotting (see axis)
which	numeric vector, which type of dependence to plot
show.legend	character, either ("bottomright", "topleft", "topright", or "bottomleft") indicates where to place the legend (see legend). NULL (default) to disable plotting a legend

Value

Does not return a value, used for the side-effect of plotting

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

calculate.power, run.tests

Examples

```
mycor = function(...) cor(...)^2
noises = cbind(c(.3,.4,6),c(.3,.5,4))
colnames(noises) = c("1",".2") #mutual information of the noise levels
vals = run.tests(mycor,list(),1:2,noises,100)
power.cor = drop(calculate.power(vals))
power.plot(list(cor=power.cor),t(noises))
```

P_ceq *Probability of observing the ith nearest neighbour at the same distance or larger as the (i-1)th nearest neighbour*

Description

This function gives the probability of observing the ith nearest neighbour distance larger or equal to c, and the (i-1)th nearest neighbour at distance c, $P(d_i >= c, d_(i-1) = c)$.

Usage

```
P_ceq(i, c, N)
```

Arguments

`i`	numeric, which nearest neighbour to consider
`c`	vector, the distance at which the ith NN was observed
`N`	numeric, size of the dataset

Details

The probability is calculated by ranking the data and assuming that the data lie on a torus. For details see Dümcke et al. "A novel test for independence derived from an exact distribution of ith nearest neighbours" (manuscript in preparation)

Value

Probability vector

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

P_cge_ale, Pc_givena

Examples

```
P_ceq(10,1:10,25)
```

P_cge_aeq *Probability of observing the ith nearest neighbour at a distance greater or equal to c and the (i-1)th nearest neighbour was observed at distance a*

Description

This function gives the probability of observing the ith nearest neighbour at a distance greater or equal to c and the (i-1)th nearest neighbour at distance a $P(d_i >= c, d_(i-1) = a)$

Usage

```
P_cge_aeq(i, c, a, k, N)
```

Arguments

`i`	numeric, which nearest neighbour to consider
`c`	vector, the distance at which the ith NN was observed
`a`	vector, the distance at which the ith NN was observed. `a <= c`
`k`	vector, number of previous NNs at distance a
`N`	numeric, size of the dataset

Details

The probability is calculated by ranking the data and assuming that the data lie on a torus. For details see Dümcke et al. "A novel test for independence derived from an exact distribution of ith nearest neighbours" (manuscript in preparation)

Value

Probability vector, entries with value -1 if the probability does not exist

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

P_ceq, Pc_givena

Examples

```
P_cge_aeq(10,4:8,2:6,rep(1,5),30)
```

P_cge_ale *Probability of observing the ith nearest neighbour at a distance greater or equal to c and the (i-1)th nearest neighbour was observed at distance smaller or equal a*

Description

This function gives the probability of observing the ith nearest neighbour at a distance greater or equal to c and the (i-1)th nearest neighbour was observed at distance smaller or equal a $P(d_i >= c, d_(i-1) <= a)$

Usage

```
P_cge_ale(i, c, a, N)
```

Arguments

`i`	numeric, which nearest neighbour to consider
`c`	vector, the distance at which the ith NN was observed
`a`	vector, the distance at which the ith NN was observed. `a <= c`
`N`	numeric, size of the dataset

Details

The probability is calculated by ranking the data and assuming that the data lie on a torus. For details see Dümcke et al. "A novel test for independence derived from an exact distribution of ith nearest neighbours" (manuscript in preparation)

Value

Probability vector, entries with value -1 if the probability does not exist

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

P_ceq, Pc_givena

Examples

```
P_cge_ale(10,4:8,2:6,30)
```

`P_di` *Probability distribution of the distance to the ith nearest neighbour*

Description

This function gives the distribution of the distances to the ith nearest neighbour of a reference point.

Usage

```
P_di(i, a, N)
```

Arguments

`i` which neares neighbhour to calculate the probability for

`a` the distance at which the ith nearest neighbout was observed, can be a vector

`N` how many points in a sample

Value

returns the probability of observing the ith nearest neighbour at distance a in a sample of size N

Author(s)

Sebastian Dümcke <`duemcke@mpipz.mpg.de`>

See Also

`Pc_givena`,`Pc_givena4nn`

Examples

```
knnIndep:::P_di(4,3,10)
```

`roc.plot` *Plot a ROC*

Description

This functions uses the results of `generate.roc` to plot a ROC plot

Usage

```
roc.plot(pows, legend = NULL, cols = colorRampPalette(c(”blue”, ”gray”))(dim(pows)[3]),
mains = c(”Linear”, ”Quadratic”, ”Cubic”, ”Sine:period 1/2”,
”Sine: period 1/8”, ”X^(1/4)”, ”Circle”, ”Step function”, ”Torus”))
```

Arguments

`pows` array, as returned by `generate.roc`

`legend` `NULL` (default) to disable legend or a matrix with noise levels as used in `run.tests`

`cols` colours to use for the plots

`mains` main title for each plot

Value

This function is used solely for its side effect of plotting

Author(s)

Sebastian Dümcke `<duemcke@mpipz.mpg.de>`

See Also

`run.tests`

Examples

```
mycor = function(...) cor(...)^2
noises = cbind(lin=c(.1,.5,.8),circ=c(.2,.4,.6))
results.cor= run.tests(mycor,args=list(),types=c(1,7),noises=noises,nsim=100,size=50)
roc.plot(generate.roc(results.cor,pval=FALSE),legend=noises)
```

`run.tests` *Run several tests of independence on a benchmark of different functional relationships*

Description

This function runs a set of independence tests on a benchmark consisting of different functional dependence types (see `generate.benchmark.data`)

Usage

```
run.tests(fun, args, types, noises, size = 320, nsim = 500, ...)
```

Arguments

`fun` function or character naming a function. A function should have two vectors of coordinates as first two arguments

`args` list of additional arguments to the functions `fun`. If a function does not need any arguments use an empty list.

`types` numeric, which type of dependence to benchmark (see `generate.benchmark.data`) `noises` matrix of noise to add to each dependence. It should have `types` number of columns

`size` numeric, size of the data sets to generate (default 320 points)

`nsim` numeric, how many replicate simulations to run under the null model and H1, default 500

`...` additional arguments to pass on to function `generate.benchmark.data`

Details

This function makes use of `mclapply` so `MC_CORES` should be set to a number greater than 1 for parallelization

Value

This function returns a list data structure that can be further processed with the functions of this package, `calculate.power`,`generate.roc`

Author(s)

Sebastian Dümcke <duemcke@mpipz.mpg.de>

See Also

`calculate.power`,`generate.roc`, `generate.benchmark.data`

Examples

```
noises = cbind(lin=c(.1,.5,.8),circ=c(.2,.4,.6))
mycor = function(...) cor(...)^2
results.cor=run.tests(mycor,args=list(),types=c(1,7),noises=noises,nsim=50,size=100)
results = run.tests("novelTest.extreme",args=list(maxi=10),types=c(1,7),noises=noises,nsim=25,
size=100)
## Not run:
x11()
par(mfrow=c(1,ncol(noises)))
roc.plot(generate.roc(results,pval=FALSE),legend=noises)

## End(Not run)
power = t(drop(calculate.power(results,.95,`>`)))
power.cor = t(drop(calculate.power(results.cor,.95,`>`)))
#cor is excellent at linear relationships, not so much for circular relationships:
#(increasing power is an artifact of low number of simulation, increase nsim in run.tests)
power.plot(list(cor=power.cor, novelTest=power),noises,show.legend="topright",mains=c("Linear",
"Circle"))
```

Index