Package ‘subrank’

June 24, 2018

Type Package
Title Computes Copula using Ranks and Subsampling
Version 0.9.9.1
Date 2018-06-24
Author Jerome Collet
Maintainer Jerome Collet <jeromepcollet@gmail.com>
Description Estimation of copula using ranks and subsampling. The main feature of this method is that simulation studies show a low sensitivity to dimension, on realistic cases.
License GPL (>= 3)
LazyLoad yes
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-06-24 06:24:12 UTC

R topics documented:

subrank-package ................................................................. 2
corc ................................................................. 3
corc0 ................................................................. 4
desscop ................................................................. 5
dessceptous ............................................................... 6
estimdep ................................................................. 7
predictdep ............................................................... 8
predonfly ............................................................... 9
simany ................................................................. 11
simnul ................................................................. 12

Index 14
subrank-package  
Computes Copula using Ranks and Subsampling

Description

Estimation of copula using ranks and subsampling. The main feature of this method is that simulation studies show a low sensitivity to dimension, on realistic cases.

Details

The DESCRIPTION file:

Package: subrank  
Type: Package  
Title: Computes Copula using Ranks and Subsampling  
Version: 0.9.9.1  
Date: 2018-06-24  
Author: Jerome Collet  
Maintainer: Jerome Collet <jeromepcollet@gmail.com>  
Description: Estimation of copula using ranks and subsampling. The main feature of this method is that simulation studies show a low sensitivity to dimension, on realistic cases.  
License: GPL (>= 3)  
LazyLoad: yes  
NeedsCompilation: yes

Index of help topics:

corc Function to estimate copula using ranks and sub-sampling  
corc0 Function to estimate copula using ranks and sub-sampling, minimal version.  
desscop Discrete copula graph, a two-dimensional projection  
desscoptous Discrete copula graph, ALL two-dimensional projections  
estimdep Dependence estimation  
predictdep Probability forecasting  
predonfly Probability forecasting  
simany Test statistic distribution under any hypothesis  
simnul Test statistic distribution under independence hypothesis  
subrank-package Computes Copula using Ranks and Subsampling

Taking a sample, its dimension, and a sub-sample size, allows to estimate a discretized copula. This object has interesting features: convergence to copula, robustness with respect to dimension.
corc

Author(s)
Jerome Collet
Maintainer: Jerome Collet <jeromepcollet@gmail.com>

Examples
lon <- 31
a <- 2.85
x <- rnorm(lon)
y = a*x^2+rnorm(lon)
tablo = as.data.frame(cbind(x,y))
c=corc(tablo,c(1,2),8)
desscop(c,1,2)

---

**corc**

*Function to estimate copula using ranks and sub-sampling*

Description
Takes a sample, its dimension, a sub-sample size, and returns a discrete copula.

Usage
corc(dataframe, varnames, subsampsize, nbsafe=5,mixties=FALSE,nthreads=2)

Arguments
dataframe a data frame, containing the observations
varnames the name of the variables we want to estimate the dependence between
subsampsize the sub-sample size
nbsafe the ratio between the number of sub-samples and the cardinality of the discretized copula.
mixties if TRUE, put equal weight on tied values, using random permutations
nthreads number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using parallel::detectCores()

Value
cop an array representing the discretized copula
ties the number of sub-samples with a tie
nsubsampreal the effective number of sub-samples drawn
varnames the name of the variables studied
nnm the number of observations without missing values
Author(s)

Jerome Collet

Examples

lon <- 30
a <- 2
x <- rnorm(lon)
y = a*x^2+rnorm(lon)
datatable = as.data.frame(cbind(x,y))
c=corc(datatable,c("x","y"),8)
c
sum(c$cop)

corc0

Function to estimate copula using ranks and sub-sampling, minimal version.

Description

Minimal version of function corc.

Usage

corc0(datavector,sampsize,dimension,subsampsize,nboot,u,mixties=FALSE,nthreads=2)

Arguments

datavector a vector, containing the observations
sampsize the sample size
dimension the sample dimension
subsampsize the sub-sample size
nboot the number of sub-samples (must be big)
u a random seed, integer
mixties if TRUE, put equal weight on tied values, using random permutations
nthreads number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using parallel::detectCores()

Value

the number of hits for each vector of ranks, plus 2 last values of the vector : number of ties and number of sub-samples really used.

Author(s)

Jerome Collet
Examples

```r
lon <- 30
a <- 2.85
x <- rnorm(lon)
\[ y = a \times x^2 + \text{rnorm}(\text{lon}) \]
\[ c = \text{corc}(c(x, y), \text{lon}, 2, 8, 1e5, 75014) \]
```

c


```r
set.seed(75013)
lon = 30
dimension = 3
sssize = 4
\[ c = \text{corc}(\text{rnorm}(\text{lon} \times \text{dimension}), \text{lon}, \text{dimension}, \text{sssize}, 1e5, 75014) \]
```

desscop

### desscop

*Discrete copula graph, a two-dimensional projection*

**Description**

Draws a discrete joint probability, for 2 variables, using bubbles

**Usage**

```r
desscop(copest, xname, yname, normalize = FALSE, axes = TRUE)
```

**Arguments**

- `copest`: the estimated copula (the whole structure resulting from `corc`)
- `xname`: the name of the variable we want to put on the horizontal axis
- `yname`: the name of the variable we want to put on the vertical axis
- `normalize`: if TRUE, the smallest probability is rescaled to 0, and the largest to 1
- `axes`: if TRUE, puts the name of the variables on the axes

**Author(s)**

Jerome Collet
**Examples**

```r
lon <- 31
a <- 2.85
x <- rnorm(lon)
y = a*x^2+rnorm(lon)
tablo = as.data.frame(cbind(x,y))
c=corc(tablo,c("x","y"),8)
desscop(c,"x","y")

tablo = as.data.frame(cbind(x=rep(0,each=lon),y=rep(0,each=lon)))
c=corc(tablo,c("x","y"),8,mixties=TRUE)
desscop(c,"x","y")
```

---

**desscoptous**  
*Discrete copula graph, ALL two-dimensional projections*

**Description**

Draws a discrete joint probability, for 2 variables, using bubbles

**Usage**

```r
desscoptous(copest, normalize = FALSE)
```

**Arguments**

- **copest**: the estimated copula (the whole structure resulting from `corc`)
- **normalize**: if TRUE, the smallest probability is rescaled to 0, and the largest to 1

**Author(s)**

Jerome Collet

**Examples**

```r
lon <- 31
a <- 2.85
x <- rnorm(lon)
y = a*x^2+rnorm(lon)
z = rnorm(lon)
tablo = as.data.frame(cbind(x,y,z))
c=corc(tablo,c("x","y","z"),8)
desscoptous(c)
```
estimdep  

__Dependence estimation__

**Description**

From a set of observations, builds a description of the multivariate distribution

**Usage**

`estimdep(dataframe, varnames, subsampsize, nbsafe=5, mixties=FALSE, nthreads=2)`

**Arguments**

dataframe  
a data frame containing the observations

varnames  
the name of the variables we want to estimate the multivariate distribution

subsampsize  
the sub-sample size

nbsafe  
the ratio between the discretized copula size and the number of sub-samples

mixties  
if TRUE, put equal weight on tied values, using random permutations

nthreads  
number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using parallel::detectCores()

**Value**

the description of the dependence, it is an object with the following parts:

cop  
the array representing the discretized copula

margins  
the matrix representing the margins, estimated using kernel density estimation

varnames  
the names of the variables

**Author(s)**

Jerome Collet

**Examples**

```r
lon=3000
plon=3000
subsampsize=20

#-------------------
x=(runif(lon)-1/2)*3
y=x^2+rnorm(lon)
z=rnorm(lon)
donori=as.data.frame(cbind(x,y,z))
depori=estimdep(donori,c("x","y","z"),subsampsize)

knownvalues=data.frame(z=rnorm(plon))
```
predictdep <- predictdep(knownvalues, depor)  # Predict probabilities
plot(prev$x, prev$y, xlim=c(-2, 2), ylim=c(-2, 5), pch=20, cex=0.5)
points(donori[,1:2], col='red', pch=20, cex=.5)

knownvalues = data.frame(x=(runif(lon)-1/R)*S)
prev <- predictdep(knownvalues, depor)  # Predict probabilities
plot(prev$x, prev$y, xlim=c(-2, 2), ylim=c(-2, 5), pch=20, cex=0.5)
points(donori[,1:2], col='red', pch=20, cex=.5)

knownvalues = data.frame(y=runif(plon, min=-2, max=4))
prev <- predictdep(knownvalues, depor)  # Predict probabilities
plot(prev$x, prev$y, xlim=c(-2, 2), ylim=c(-2, 5), pch=20, cex=0.5)
points(donori[,1:2], col='red', pch=20, cex=.5)

---

**predictdep**

**Probability forecasting**

**Description**

From a set of incomplete observations, and a description of the dependence, provides simulated values of the unknown coordinates. It is also possible to simulate unconditionally, with empty observations.

**Usage**

```
predictdep(knownvalues, dependence, smoothing=c("Uniform","Beta"), nthreads=2)
```

**Arguments**

- **knownvalues**: in case of conditional simulation, a matrix containing incomplete observations, the known coordinates being the same for all observations. If no variable name in knownvalues appears in dependence$varnames, then the simulation is unconditional.
- **dependence**: the description of the dependence we want to use to forecast, as built by function `estimdep`
- **smoothing**: the smoothing method for input and output ranks.
- **nthreads**: number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using `parallel::detectCores()`

**Value**

the matrix of the completed observations

**Author(s)**

Jerome Collet
Examples

lon=100
plon=100
subsampsize=10

shift=0
noise=0
knowndims=1

x=rnorm(lon)
y=2*x+noise+rnorm(lon)
donori=as.data.frame(cbind(x,y))
depori=estimdep(donori,c("x","y"),subsampsize)
##
knownvalues=data.frame(x=rnorm(plon)+shift)
prev <- predictdep(knownvalues,depori)
##
plot(prev$x,prev$y,xlim=c(-2,2),ylim=c(-2,5),pch=20,cex=0.5)
points(donori[,1:2],col='red',pch=20,cex=.5)
##
knownvalues=data.frame(x=rnorm(plon)+shift)
prev <- predictdep(knownvalues,depori,smoothing="Beta")
##
plot(prev$x,prev$y,xlim=c(-2,2),ylim=c(-2,5),pch=20,cex=0.5)
points(donori[,1:2],col='red',pch=20,cex=.5)
# souci normal si |shift|>>1

knownvalues=data.frame(z=rnorm(plon)+shift)
prev <- predictdep(knownvalues,depori)
##
plot(prev$x,prev$y,xlim=c(-2,2),ylim=c(-2,5),pch=20,cex=0.5)
points(donori[,1:2],col='red',pch=20,cex=.5)

predonfly

**Probability forecasting**

Description

From two sets of observations, first one of complete observations and second one of incomplete observations, provides simulated values of the unknown coordinates.

Usage

predonfly(completeobs,incompleteobs,varnames,subsampsize,nbpsrts=1,mixties=FALSE, maxtirs=1e5,complete=TRUE,nthreads=2)
Arguments

- completeobs: the set of complete observations.
- incompleteobs: the set of incomplete observations.
- varnames: the modeled variables.
- subsampsize: the sub-sample size.
- nbpreds: the number of predictions for each incomplete observation.
- mixties: if TRUE, should put equal weight on tied values, using random permutations (not yet implemented)
- maxtirs: the maximum number of sub-samples, to stop the computation even if they did not provide nbpreds predictions for each incomplete observation.
- complete: If TRUE, predictions are completed with incomplete observations.
- nthreads: number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using parallel::detectCores()

Value

the matrix of the completed observations

Author(s)

Jerome Collet

Examples

```r
lon=100
plon=30
subsampsize=10

x=rnorm(lon)
y=2*x+rnorm(lon)*0
donori=as.data.frame(cbind(x,y))
##
knownvalues=data.frame(x=rnorm(plon))
prev <- predonfly(donori,knownvalues,c("x","y"),subsampsize,100)
##
plot(prev$x,prev$y,pch=20,cex=0.5,
    ylim=range(c(prev$y,donori$y),na.rm=TRUE),xlim=range(c(prev$x,donori$x)))
points(donori[,1:2],col='red',pch=20,cex=.5)

lon=3000
mg=20
dimtot=4
rayon=6

genboules <- function(lon,a,d)
{
  ss <- function(vec)
```
simany

Test statistic distribution under any hypothesis

Description
Simulates the test statistic, under independence

Usage
simany(sampsize, dimension, subsampsizes, sampnum, nbsafe=5, nthreads=2, fun=NULL, ...)

Arguments
sampsize      sample size
dimension     sample dimension
subsampsizes  vector of sub-sample sizes
sampnum       number of samples
nbsafe        the ratio between the number of sub-samples and the cardinality of the discretized copula.
nthreads      number of number of threads, assumed to be strictly positive. For "full throttle" computations, consider using parallel::detectCores()
fun           the function describing the dependence.
...           optional arguments to fun
**Value**

- `lrs` the distances with independent case
- `lrs2mean` the distances with theoretical value, given dependence `fun`
- `scarcities` the proportions of non-reached vector ranks
- `DistTypes` a recall of the list of the distance types: "KL", "L2", "L1", "APE"

**Author(s)**

Jerome Collet

**Examples**

```r
depquad <- function(lon, dd, a)
{
  x <- rnorm(lon)
  y0 <- a*x^2
  y <- y0 + rnorm(lon)
  reste <- rnorm((dd-2)*lon)
  return(c(x, y, reste))
}
sims0 <- simany(101, 3, 8, 50, nbsafe = 1)
seuil <- apply(sims0$lrs, 3, quantile, 0.95)
seuil <- matrix(ncol = 4, nrow = 50, seuil, byrow = TRUE)
sims1 <- simany(101, 3, 8, 50, nbsafe = 1, fun = depquad, a = 0.5)
apply(sims1$lrs[, 1] > seuil, 2, mean)
```

---

**simnul**

*Test statistic distribution under independence hypothesis*

**Description**

Simulates the test statistic, under independence

**Usage**

```r
simnul(sampsize, dimension, subsampsizes, sampnum, KL = TRUE, nbsafe = 5, nthreads = 2)
```

**Arguments**

- `sampsize` sample size
- `dimension` sample dimension
- `subsampsizes` vector of sub-sample sizes
- `sampnum` number of samples
- `KL` if TRUE, returns the Kullback-Leibler divergence with the independent case, if FALSE, the L2 distance. There is no re-normalization, contrary to what happens for simany.
**Value**

- `lrs` the distances with independent case
- `scarcities` the proportions of non-reached vector ranks

**Author(s)**
Jerome Collet

**Examples**

```r
library(datasets)
# plot(swiss)
c=cocor(swiss,1:3,8)
c
RV=sum(c$cop*log(c$cop),na.rm=TRUE)+3*log(8)
sims=simmul(47,3,8,100)
pvalue=mean(RV<=sims$lrs)
pvalue
RV
summary(sims$lrs)
```
Index

corc, 3, 5, 6
corc0, 4
desscop, 5
desscoptous, 6
estimdep, 7, 8
predictdep, 8
predonfly, 9
simany, 11
simmul, 12
subrank (subrank-package), 2
subrank-package, 2