Package 'CopulaModel'

June 27, 2014

Type Package

Title CopulaModel: Dependence Modeling with Copulas

Version 0.6

Date 2014-06-27

Author H Joe; P Krupskii (factor copulas, tail-weighted dependence measures)

Maintainer Dependence Modeling <copula@stat.ubc.ca>

Depends R (>= 2.9.0), cubature

Description Software for book: Dependence Modeling with Copulas, Chapman & Hall/CRC, 2014.

License GPL-3

LazyLoad yes

R topics documented:

`CopulaModel-package` *CopulaModel: Dependence Modeling with Copulas*

Description

CopulaModel is the accompanying software for the book: Dependence Modeling with Copulas, by H. Joe, Chapman & Hall/CRC, 2014. With this software, a reader can check (almost) all numerical computations in the book. Much of the contributions for factor copula models is also described in the Ph.D. thesis of P. Krupskii (2014).

Details

All of the algorithms in Dependence Modeling with Copulas are shown within this software. There are templates for doing many numerical calculations and maximum likelihood estimation with copulas. Included are examples of faster code with links to Fortran90 and C. It is not possible to provide functions for all possible uses of copulas. But a user can adapt the templates and code within this software for many other applications. The source code (R, Fortran90, C) has more documentation than these help pages. The functions do not check on the domains of the arguments such as the copula parameter, but the domains are indicated in the source code. If a function doesn't seem to be working with your inputs, check the source code and look for examples in the tests subdirectories and examples in the book chapter subdirectories.

Some features include:

- For the 1-, 2-, 3-parameter bivariate copula families in the book, most have pcop, dcop, rcop, pcondcop, qcondcop for the copula cdf, copula density, copula random generation, conditional distribution $C_{2|1}$ and conditional quantile $C_{2|1}^{-1}$, where 'cop' is an abbreviated copula name. Also for many bivariate copula families, there are conversions among copula parameter, Kendall's tau, Spearman's rho, Blomqvist's beta, correlation of normal scores, and tail dependence parameters.
- R-vine (regular vine) models for continuous data with specified vine array and parametric pair-copula families.
- R-vine models for discrete response data with possibility of covariates.
- Factor copula models for continuous response (factor models here are truncated vines with latent variables).
- Factor copula models for ordinal (item) response.

Author(s)

H. Joe and P. Krupskii.

References

H. Joe (2014). Dependence Modeling with Copulas. Chapman & Hall/CRC. Boca Raton, FL.

P. Krupskii (2014). Structured Factor Copulas and Tail Inference. PhD thesis, University of British Columbia.

Other specific references are given within code files or help pages.

`ALAE600` *ALAE-loss subset of size 600*

Description

Subset of data set used in "Frees EW and Valdez EA (1998). Understanding relationships using copulas. North American Actuarial Journal, 2, 1-25." The subset has 600 uncensored observations.

Usage

```
data(alae600)
```

ALAEloss *ALAE-loss data set*

Description

Data set used in "Frees EW and Valdez EA (1998). Understanding relationships using copulas. North American Actuarial Journal, 2, 1-25."

Usage

```
data(ALAEloss)
```

ar2acf *Autocorrelation of Gaussian AR2 given lag1 and lag2 correlations*

Description

Autocorrelation of Gaussian AR2 given lag1 and lag2 correlations

Usage

```
ar2acf(rho1,rho2,d)
```

Arguments

rho1	correlation for lag 1
rho2	correlation for lag 2, so correlation matrix with rho1,rho1,rho2 should be positive definite
d	dimension of desired Toeplitz matrix

Value

acf function up to lag d-1

Examples

```
acf10=ar2acf(.6,.5,10)
toeplitz(acf10[1:6])
```

asianwklgret *Weekly returns from 4 Asian markets*

Description

Weekly returns from 4 Asian markets, July 1997 to May 2006; n=465 weeks. The indexes are: Hong Kong Hang Seng (HSI), Singapore Straights (STI), Seoul Kospi (KS11), and Taiwan Taiex (TWII). Original source is http://quote.yahoo.com

Usage

```
data(asianwklgret)  # object hksikotw
```

Format

A data matrix with 465 observations on the following 4 variables.

`hksikotw` columns are hk, si, ko, tw

asymgum *Bivariate asymmetric Gumbel and Galambos copulas*

Description

Bivariate asymmetric Gumbel and Galambos copulas

Usage

```
pasymgum(u,v,cpar)  # C(u,v)
pcondasymgum21(v,u,cpar) # C_{2|1}(v|u)
pcondasymgum12(u,v,cpar)  # C_{1|2}(u|v))
dasymgum(u,v,cpar)  # c(u,v)
asymgum.cpar2tau(cpar) # Kendall's tau
asymgum.cpar2rhoS(cpar) # Spearman's rho
Basymgum(ww,cpar,mxiter=30,eps=1.e-7,iprint=F)  # B(w)=A(w,1-w)
# similar to above with 'gum' replaced by 'gal' or 'gumMO' except
AasymgumMO(x,y,cpar)
```

Arguments

u	value in interval 0,1; could be a vector
v	value in interval 0,1; could be a vector
cpar	copula parameter vector, for asymgum, the two parameters are each in (0,1); for asymgal, the two parameters are each negative; for asymgumMO, the 3 parameters are delta>1, pi1 in (0,1] and pi2 in (0,1]
ww	vector of values in (0,1)
mxiter	maximum number of iterations
eps	tolerance for convergence
iprint	print flag for intermediate results
x	vector of positive values
y	vector of positive values

Details

The distribution in asymgum is also known as bilogistic in the multivariate extreme value literature – it is the bivariate Gumbel when the two parameters are equal and the Gumbel parameter is the reciprocal. The distribution in asymgal is also known as negative bilogistic in the multivariate extreme value literature – it is the bivariate Galambos when the two parameters are equal and the Galambos parameter is the negative reciprocal. The distribution in asymgumMO becomes bivariate Gumbel when the second and third parameters are both 1.

Value

cdf for pasymgum

conditional cdf for pcondasymgum21 and pcondasymgum12

pdf for dasymgum

Kendall's tau for asymgum.cpar2tau

Spearman's rho for asymgum.cpar2rhoS

list with $Bfn (function values), $Bder (first derivatives), $Bder2 (second derivatives) for Basymgum

Examples

```
ze=.3; eta=.2
cpar=c(ze,eta)
heps=1.e-5
u=.3; v=.8
chkcopderiv(u,v,cpar,bcdf=pasymgum,pcond=pcondasymgum21,bpdf=dasymgum,
  str="asymgum",eps=heps)
tau=asymgum.cpar2tau(cpar)
rho=asymgum.cpar2rhoS(cpar)
cat(cpar,tau,rho,"\n")
# special case of Gumbel
cpar=c(.5,.5)
cdf=pasymgum(.4,seq(.1,.9,.2),cpar)
cdf2=pgum(.4,seq(.1,.9,.2),1/cpar[1])
print(cbind(cdf,cdf2))
tau=asymgum.cpar2tau(cpar)
rho=asymgum.cpar2rhoS(cpar)
cat(tau,rho,"\n")
tau=gum.cpar2tau(1/cpar[1])
rho=gum.cpar2rhoS(1/cpar[1])
cat(tau,rho,"\n")
# asymmetric Galambos
cpar=c(-.5,-.5)
cdf=pasymgal(.4,seq(.1,.9,.2),cpar)
cdf2=pgal(.4,seq(.1,.9,.2),-1/cpar[1])
print(cbind(cdf,cdf2))
#and gumMO to add
```

`asymmetry` *Asymmetry / skewness measures for bivariate copulas*

Description

Reflection and permutation asymmetry / skewness measures for bivariate copulas

Usage

```
skewrefl(uu)  # third moment (u1+u2-1)^3
skewperm(uu)  # third moment  (u1-u2)^3
qskewperm(uu,p=.05,nrep=100)  # quantile-based measure for u1-u2
qskewrefl(uu,p=.05,nrep=100)  # quantile-based measure for u1+u2-1
```

Arguments

uu	nx2 data matrix of uniform scores
p	value between 0 and 0.5, leading to quantiles at p and 1-p
nrep	number of replications for bootstrap SE

Value

skewness/asymmetry measure, with a crude SE

References

Rosco J-F and Joe H (2014). Measures of tail asymmetry for bivariate copulas. Statistical Papers, 54, 709-726.

Examples

```
n=100
set.seed(123)
urefl=rbreflasym(n,.25)  # extreme reflection asymmetry
uperm=rbpermasym(n,.25)  # extreme permutation asymmetry
n=1000
set.seed(123)
uagum=rbasymgum1(n,c(2,.4))  # bivariate asymmetric Gumbel
umo=rbMO1(n,p=.4) # bivariate 1-parameter Marshall-Olkin
## Not run:
par(mfrow=c(2,2))
plot(urefl)
plot(uperm)
plot(uagum)
plot(umo)
## End(Not run)
sk1=skewrefl(urefl)
sk2=skewperm(uperm)
skagum=skewperm(uagum)
skmo=skewperm(umo)
print(rbind(sk1,sk2,skagum,skmo))
qsk1=qskewrefl(urefl)
qsk2=qskewperm(uperm)
qskagum=qskewperm(uagum)
qskmo=qskewperm(umo)
print(rbind(qsk1,qsk2,qskagum,qskmo))
```

bb1dep2cpar *BB1 Bivariate copula: mapping of measure of association and upper tail dependence lambda to 2-dimensional parameter vector*

Description

BB1 Bivariate copula: mapping of one of (beta,tau,rhoS,rhoN) and upper tail dependence lambda to 2-dimensional parameter vector (theta,delta); also mapping of Kendall's tau to copula parameter vector such that lower and upper tail dependence parameters are the same.

Usage

```
bb1.dep2cpar(value,lmU,type="tau",iprint=F)
bb1.tau2eqlm(tau,destart=1.5,mxiter=30,eps=1.e-6,iprint=F)
```

Arguments

value	value of dependence measure
lmU	value of upper tail dependence lambda
type	one of "tau", "beta", "rhoS" or "rhoN"
tau	Kendall's tau
iprint	print flag for intermediate results
destart	starting point of delta for iterations, default 1.5
mxiter	maximum number of iterations
eps	tolerance for stopping

Value

2-dimensional copula parameter (theta, delta) if a solution is found for bb1.dep2cpar

vector with th, de, lm1, lmu for bb1.tau2eqlm

See Also

deppar2taurhobetalambda

Examples

```
bb1.dep2cpar(.4,.1,"beta")
bb1.dep2cpar(.4,.1,"tau")
bb1.dep2cpar(.4,.1,"rhoS")
bb1.dep2cpar(.4,.1,"rhoN")
bb1.tau2eqlm(.4)
```

bb1rpow *Three-parameter bivariate copula families*

Description

Three-parameter bivariate copula families with a power parameter

Usage

```
pbb1rpow(u,v,cpar)
dbb1rpow(u,v,cpar)
pcondbb1rpow(v,u,cpar)
qcondbb1rpow(p,u,cpar,pvec=c(0.01,seq(.02,.98,.02),.99),icheck=F)
pbb7pow(u,v,cpar)
dbb7pow(u,v,cpar)
pcondbb7pow(v,u,cpar)
```

Arguments

u	value in interval 0,1; could be a vector
v	value in interval 0,1; could be a vector
p	quantile in interval 0,1; could be a vector
cpar	3-parameter vector: (theta,delta) or reflected BB1 or BB7 copula with a third parameter eta>0 for the power
pvec	for qcondbb1rpow(), a vector of quantiles to use with monotone interpolation
icheck	flag to output a check on inverse

Value

(conditional) cdf or pdf or quantile value(s)

Examples

```
cpar=c(1.1,1.5,0.5)
cpar1=c(1.1,1.5,1)
u=seq(.1,.9,.2)
v=u
pbb1rpow(u,v,cpar)
pbb7pow(u,v,cpar)
pbb1rpow(u,v,cpar1)
pbb1r(u,v,cpar1[1:2]) # same as above
pbb7pow(u,v,cpar1)
pbb7(u,v,cpar1[1:2]) # same as above
```

`bifct` *bi-factor and tri-factor structure*

Description

bi-factor and tri-factor structure: matrix inverse and determinant

Usage

```
bifct(grsize,rh1,rh2)
bifct2(grsize,rh1,rh2) # version 2, with inverse and det of a smaller matrix
trifct(grsize,sbgrsize,rh1,rh2,rh3)
subgr.consistent(grsize,sbgrsize) # check that grsize,sbgrsize are consistent
```

Arguments

`grsize`	vector of group sizes for mgrp groups with sum(grsize)=d
`sbgrsize`	vector of subgroup sizes by partitioning grsize vector
`rh1`	vector of correlations with global latent variable
`rh2`	vector of partial correlations with group latent variable given global; ith variable linked to group g
`rh3`	vector of partial correlations with subgroup latent variable given global and group latent; ith variable linked to group sg

Value

$fctmat (correlation matrix), $fctdet (determinant), $fctinv (inverse) for bifct, bifct2, trifct.

TRUE or FALSE for subgr.consistent depending on consistency

See Also

`mvtfact`

Examples

```
d=29
# bi-factor
grsize=c(5,5,3,3,4,2,2,2,3) # sum(grsize)=d
bifpar=c(0.84,0.63,0.58,0.78,0.79,
 0.87,0.80,0.74,0.71,0.57,   0.83,0.77,0.80, 0.74,0.78,0.71,
 0.71,0.58,0.63,0.64, 0.77,0.73, 0.72,0.72, 0.45,0.49, 0.66,0.70,0.69,
 0.67,0.58,0.15,0.70,0.47, 0.32,0.27,0.73,0.19,0.12, 0.35,0.23,0.53,
 0.89,0.63,0.63, 0.28,0.51,0.80,-0.02, 0.37,0.37, 0.82,0.82,
 0.70,0.70,-0.25,0.55,0.10)
bifobj=bifct(grsize,bifpar[1:d],bifpar[(d+1):(2*d)])
rmat=bifobj$fctmat
print(det(rmat)-bifobj$fctdet)
print(max(abs(solve(rmat)-bifobj$fctinv)))
# tri-factor
grsize=c(11,18) # sum(grsize)=d
```

```
sbgrsize = c(5,4,2,5,3,3,2,2,3) # sum(sbgrsize)=d
subgr.consistent(grsize,sbgrsize)
trifpar=c(0.86,0.77,0.73,0.71,0.54,0.70,
 0.55, 0.60,  0.64, 0.40, 0.44, 0.84, 0.65, 0.58,  0.78, 0.78, 0.83, 0.79,
 0.81, 0.77,  0.80, 0.73, 0.76, 0.73, 0.70, 0.70,  0.62, 0.72, 0.68, 0.25,
 0.37, 0.46,  0.23, 0.42, 0.28, 0.43, 0.45,-0.05,  0.61, 0.62, 0.54, 0.09,
 0.09, 0.34,  0.45, 0.07,-0.06,-0.02,-0.30,-0.12, -0.27, 0.13, 0.06, 0.56,
 0.50, 0.51, -0.31, 0.03, 0.81, 0.18, 0.21,-0.02, -0.16, 0.19, 0.37, 0.81,
 0.05, 0.48,  0.48, 0.50, 0.66, 0.10, 0.66, 0.29,  0.30, 0.11, 0.46, 0.86,
 0.60, 0.56,  0.37, 0.37, 0.76, 0.76, 0.14, 0.31,  0.18)
trifobj=trifct(grsize,sbgrsize,trifpar[1:d],trifpar[(d+1):(2*d)],
   trifpar[(2*d+1):(3*d)])
rmat=trifobj$fctmat
print(det(rmat)-trifobj$fctdet)
print(max(abs(solve(rmat)-trifobj$fctinv)))
```

`bivcopnllk` *Negative log-likelihood for bivariate copula model*

Description

Negative log-likelihood for bivariate copula model to input to numerical minimizer

Usage

```
bivcopnllk(param,udat,logdcop,ivect=T,LB=0,UB=1000)
bivmodnllk(param,xdat,logdcop,logpdf1,cdf1,np1,logpdf2,cdf2,np2,ivect=T,LB,UB)
bivcopnllk.ipol(param,udat,logbpdf,logupdf,uquant,iunivar,ppvec=NULL,LB,UB)
  # latter function used interpolation for a copula based on
  #  F_{12}(F^{-1}(u1),F^{-1}(u2)) where F^{-1} is computationally difficult
```

Arguments

`param`	parameter of model: copula parameter only for bivcopnllk; (par1,par2,par3) for bivmodnllk where par1 is the vector of parameters for univariate margin 1, par2 is for univariate margin 2, and par3 is for the bivariate copula.
`udat`	nx2 matrix, assume each column is U(0,1) distributed
`xdat`	nx2 matrix, original untransformed data
`logdcop`	function with log of copula density
`ivect`	flag that is T if logdcop can take vectorized inputs
`logpdf1`	function with log of univariate density of first variable
`cdf1`	function with univariate cdf of first variable
`np1`	dimension of par1 (parameters for first variable)
`logpdf2`	function with log of univariate density of second variable
`cdf2`	function with univariate cdf of second variable
`np2`	dimension of par2 (parameters for second variable)
`logbpdf`	function with log of density of F_{12}
`logupdf`	function with log of univariate marginal density

uquant	function for inverse of univariate marginal cdf
iunivar	vector with indices such that par1=param[iunivar] is the univariate parameter
ppvec	vector of quantiles to use for interpolation, a default is used in this is input as NULL; a possibility is something like ppvec=seq(min(udat),max(udat),length=100)
LB	vector of lower bounds on parameter values
UB	vector of upper bounds on parameter values

Value

negative log-likelihood

See Also

ordinal.bivcop

Examples

```
data(alae600)
alae=alae600$alae
loss=alae600$loss
ppareto=function(x,param)
{ alp=param[1]; s=param[2]
  u=1-(1+(x/s))^{-alp}
  u
}
logdpareto=function(x,param)
{ alp=param[1]; s=param[2]
  lpdf=log(alp/s)-(alp+1)*log(1+(x/s))
  lpdf
}
paretonllk=function(param,xdat)
{ alp=param[1]; s=param[2]
  if(alp<=0 | s<=0) return(1.e10)
  xs=xdat/s
  nllk=(alp+1)*log(1+xs) - log(alp/s)
  sum(nllk)
}
alae.pareto=nlm(paretonllk,p=c(2.39,1.6),hessian=TRUE,print.level=1,xdat=alae/10000)
loss.pareto=nlm(paretonllk,p=c(1.5,1.5),hessian=TRUE,print.level=1,xdat=loss/10000)
mle1p=alae.pareto$estimate
mle2p=loss.pareto$estimate
ualae.pareto=ppareto(alae/10000,mle1p)
uloss.pareto=ppareto(loss/10000,mle2p)
udat=cbind(ualae.pareto,uloss.pareto)
# IFM or inference functions for margins
ifm.pp=nlm(bivcopnllk,p=1.5,hessian=TRUE,print.level=1,
  udat=udat,logdcop=logdgum,LB=1,UB=20)
# mle is 1.434, nllk is -80.41
# full likelihood
parampp=c(mle1p,mle2p,ifm.pp$estimate)
full.pp=nlm(bivmodnllk,p=parampp,hessian=TRUE,print.level=1,
    xdat=cbind(alae/10000,loss/10000),logdcop=logdgum,
    logpdf1=logdpareto,cdf1=ppareto,np1=2,logpdf2=logdpareto,cdf2=ppareto,np2=2,
    LB=c(rep(0,4),1),UB=c(rep(100,4),20) )
# Pareto margin with bivariate Gaussian copula as a comparison
```

```
ml=nlm(bivcopnllk,p=.5,hessian=TRUE,print.level=1,
  udat=udat,logdcop=logdbvncop,LB=-1,UB=1)
# mle is 0.471, nllk is -72.25
logdnorm=function(x,par1) { dnorm(x,log=TRUE) }
uqnorm=function(p,par1) { qnorm(p) }
ppvec=seq(min(udat),max(udat),length=100)
mla=nlm(bivcopnllk.ipol,p=.5,udat,hessian=TRUE,print.level=1,
  logbpdf=logdbvn,logupdf=logdnorm,uquant=uqnorm,iunivar=NULL,
  ppvec=ppvec,LB=-1,UB=1)
# mle is 0.469, nllk is -72.09 with the interpolation
```

`bivdepmeas` *Kendall's tau, Spearman's rho and normal scores correlation for a bivariate copula*

Description

Kendall's tau, Spearman's rho and normal scores correlation for a bivariate copula

Usage

```
ktau(cpar,icond=T,pcond12,pcond21,zero=0,dcop,pcop,B=6,tol=0.0001)
rhoS(cpar,cop,zero=0,icond=F,tol=0.0001)
rhoN(cpar,icond=T,pcond,icdf=F,pcop,dcop,B=6,tol=0.0001)
```

Arguments

`cpar`	copula parameter: scalar or vector
`icond`	if T, numerical integration via conditional cdf of copula; otherwise via the copula cdf for rhoS and copula cdf/pdf for tau, rhoN
`icdf`	if T, numerical integration via the cdf and Hoeffding's identity, otherwise via the pdf (for rhoN)
`pcond12`	function for $C_{1\|2}(u\|v)$
`pcond21`	function for $C_{2\|1}(v\|u)$
`pcond`	function for $C_{2\|1}(v\|u)$, for rhoN
`cop`	for rhoS, copula conditional cdf if icond=T, and bivariate copula cdf if icond=F
`zero`	boundary is $[zero, 1-zero]^2$ for integration, default is 0, but choose smething like zero=1.e-6 if there is a boundary problem
`dcop`	function for bivariate copula density, needed if icond=F
`pcop`	function for bivariate copula cdf, needed if icond=F
`B`	upper limit for integration over $[0, B]^2$ with respect to normal density if icond=F
`tol`	desired accuracy for numerical integration, default 0.0001

Value

`depmeas`	dependence measure with value in (-1,1)

See Also

`depmeasAsympVar`

Examples

```
cpar=2
tau=ktau(cpar,pcond12=pcondpla,pcond21=pcondpla,zero=0)
spear=rhoS(cpar,cop=pfrk,zero=0)
rhoz=rhoN(cpar,pcond=pcondfrk)
```

---

bivpmf *bivariate probability mass function: cdf and correlation*

Description

bivariate probability mass function: cdf and correlation

Usage

```
bivpmf2cdf(bpmf)
corbivpmf(bpmf)
```

Arguments

bpmf	bivariate probability mass function in a matrix (non-negative and sums to 1)

Value

matrix with bivariate cdf for bivpmf2cdf()

means, variances, covariance, correlation for corbivpmf(), assuming categories are 0:(nrow(pbmf)-1) and 0:(ncol(pbmf)-1)

Examples

```
bpmf=matrix(c(.1,.2,.3,.2,.1,.1),2,3)
bivpmf2cdf(bpmf)
corbivpmf(bpmf)
```

---

blomq *Blomqvist's beta for bivariate data set*

Description

Blomqvist's beta for bivariate data set

Usage

```
blomq(bivdat,iunif=F)
```

Arguments

bivdat	data matrix with 2 columns
iunif	flag for whether bivdat has already been converted to uniform(0,1) scores, default=F in which case conversion is done

Value

empirical Blomqvist's beta

Examples

```
set.seed(1234)
n=300; cpar=3
udat=rgum(n,cpar)
blomq(udat)
4*pgum(.5,.5,cpar)-1  # theoretical value
```

---

btsampleStaty *Bootstrap from a stationary time series*

Description

Indexes for bootstrap from a stationary time series

Usage

```
btsampleStaty(size,p)
```

Arguments

size	length of time series
p	geometric rate such as size^(-1/3)

Value

vector of length 'size', each element is an integer between 1 and 'size'

References

Politis DN and Romano JP (1994). The stationary bootstrap. Journal of the American Statistical Association, 89, 1303–1313.

Examples

```
set.seed(123)
n=500
p=n^(-1/3)
for(isim in 1:5)
{ ii=btsampleStaty(n,p)
  print(length(unique(ii)))
}
```

bvnsemic *Semi-correlation for bivariate Gaussian*

Description

Semi-correlation for bivariate Gaussian

Usage

```
bvnsemic(rho)
```

Arguments

rho	(vector of) correlation parameter for bivariate Gaussian

Value

semi-correlation (same for lower and upper)

See Also

rhoNsemic

Examples

```
rho=seq(.1,.9,.1)
bvnsemic(rho)
# see function for the formula
```

bvtdep2cpar *Bivariate t copula: mapping of measure of association to copula parameter given shape parameter nu>0*

Description

Bivariate t copula: mapping of measure of association to copula parameter given shape parameter nu>0

Usage

```
bvt.dep2cpar(value,nu,type="tau",ngrid=15,iprint=F)
```

Arguments

value	value of dependence measure
nu	positive shape parameter for t distribution
type	one of "tau", "beta","lambda", "rhoS" or "rhoN"
ngrid	number of grid points to use in interpolation, default 15
iprint	print flag for intermediate results

Value

parameter in interval (-1.1)

See Also

deppar2taurhobetalambda

Examples

```
bvt.dep2cpar(.4,5,"beta")
bvt.dep2cpar(.4,5,"tau")
bvt.dep2cpar(.4,5,"lambda")
bvt.dep2cpar(.4,5,"rhoS")
bvt.dep2cpar(.4,5,"rhoN")
```

---

chkcop *Checks for pcop pcond dcop qcond functions*

Description

Check pcop pcond dcop qcond functions for copula cdf, conditional cdf, pdf, conditional quantile

Usage

```
chkcopderiv(u,vvec,cpar,bcdf,pcond,bpdf,str=" ",eps=1.e-4)
chkcopcond(uvec,vvec,cpar,pcond,qcond,str=" ",tol=1.e-5)
chkincrease(mat)
```

Arguments

u	scalar in (0,1)
vvec	vector of values in (0,1)
uvec	vector of values in (0,1), same length as vvec
cpar	copula parameter
bcdf	function for bivariate copula cdf = pcop
bpdf	function for bivariate copula pdf = dcop
pcond	function for conditional copula cdf
qcond	function for conditional copula quantile
str	string for copula name
eps	step size for numerical derivative in chkcopderiv
tol	tolerance for pcond/qcond composition versus identity in chkcopcond
mat	matrix or dataframe

Value

For chkcopderiv and chkcopcond, output is printed, nothing is returned. chkcopcond checks pcond vs numerical derivative of pcop; chkcopderiv checks also dcop vs numerical derivative of pcond.

For chkincrease, a value of 1 means the values are increasing in each row, otherwise 0 is returned.

Examples

```
# checks cdf, conditional cdf, pdf for Plackett copula
be=.8
par.pla=pla.b2cpar(be)
u=.3
vvec=seq(.4,.9,.1)
chkcopderiv(u,vvec,par.pla,bcdf=ppla,pcond=pcondpla,bpdf=dpla,str="pla")
uvec=seq(.1,.6,.1)
vvec=seq(.4,.9,.1)
chkcopcond(uvec,vvec,par.pla,pcondpla,qcondpla,"pla")
# check monotonicity for Plackett copula
u=seq(.1,.9,.2)
umat=matrix(rep(u,5),5,5,byrow=TRUE)
vmat=matrix(rep(u,5),5,5)
cdf=ppla(umat,vmat,par.pla)
chkincrease(cdf)
# check monotonicity of dependence measure for Galambos and Plackett copulas
data(deptabder)
chkincrease(gal.deptab[,1:6]) # cols cpar beta tau rhoS rhoN lambda
apply(gal.deptab,2,diff)
chkincrease(pla.deptab[,1:5]) # cols cpar beta tau rhoS rhoN l
```

---

contourBivCop — *Contour plot of bivariate copula density with standard Gaussian margins*

Description

Contour plot of bivariate copula density with standard Gaussian margins

Usage

```
contourBivCop(cpar,zvec,dcop,irefl=F)
```

Arguments

cpar	copula parameter
zvec	grid points of N(0,1) density for computing bivariate density
dcop	function for bivariate copula density with arguments u,v,cpar
irefl	flag to take reflection of dcop, default is F

Value

nothing is returned, a contour plot is drawn

Examples

```
zvec=seq(-3,3,.2)
# contour density plots for Frank, Gumbel, reflected Gumbel
contourBivCop(4,zvec,dcop=dfrk)
contourBivCop(2.5,zvec,dcop=dgum)
contourBivCop(2.5,zvec,dcop=dgum,irefl=TRUE)
```

`copderiv` *Derivatives of log copula density and copula conditional cdf*

Description

Derivatives of log copula density and copula conditional cdf

Usage

```
logdfrk.deriv(u,v,cpar)  # derivatives of c_{12}(u,v,cpar)
pcondfrk.deriv(v,u,cpar) # deriv of C_{2|1}(v+u;cpar)
# also frk can be replaced with gum or bb1 or ..
logdbvtcop.deriv(u,v,param,df=dfdefault)
pcondbvtcop.deriv(v,u,param,df=dfdefault)
```

Arguments

`u`	value in interval 0,1; could be a vector
`v`	value in interval 0,1; could be a vector
`cpar`	copula parameter: could be scalar or vector depending on the copula family
`param`	for t copula, this is either rho in (-1,1) or (rho,df) where df>0; in the former case, set dfdefault before using
`df`	global default shape parameter for t copula

Details

These are templates. A user can add other functions like these as needed. These could be useful for R-vine negative log-likelihoods with analytic derivatives for input into nlm. Result is faster with t copulas with fixed shape parameters.

Value

log pdf or $C_{2|1}$ value(s) with derivatives with respect to u, v and cpar. Output is a vector of length(3+length(cpar)) with u,v are scalars; otherwise output is a matrix of nx [length(3+length(cpar))] if at least one of u or v is a vector of length n.

See Also

`pcop` `pcond`

Examples

```
u=seq(.1,.9,.2)
v=c(.2,.2,.4,.6,.8)
dfdefault=5
logdfrk.deriv(u,v,2)
logdbvtcop.deriv(u,v,.6)
logdbvtcop.deriv(u,v,c(.6,5))
logdbb1.deriv(u,v,c(.5,2))
pcondfrk.deriv(u,v,2)
pcondbvtcop.deriv(u,v,.6)
pcondbvtcop.deriv(u,v,c(.6,5))
pcondbb1.deriv(u,v,c(.5,2))
```

copextreme *Boundary copula cdfs*

Description

Boundary copula cdfs of independence, comonotonicity and countermonotonicity

Usage

```
pindepcop(u,v=-1,cpar=0)
pcomonocop(u,v=-1,cpar=0)
pcountermono(u,v=-1,cpar=0)
pcondindep(v,u,cpar=0)
pcondcomono(v,u,cpar=0)
pcondcountermono(v,u,cpar=0)
```

Arguments

u	value in interval 0,1 for bivariate; vector of values in (0,1) for d-variate
v	value in interval 0,1 for bivariate; -1 for d-variate
cpar	copula parameter: could be anything but is not used; this is for consistency when pass to functions using copula cdfs

Value

cdf value(s): pcountermono uses only u[1],u[2] if v=-1

Examples

```
pindepcop(.1,.2)
pindepcop(seq(.1,.5,.2))
pcomonocop(.1,.2)
pcomonocop(seq(.1,.5,.2))
pcountermono(.5,.7)
pcountermono(seq(.5,.9,.2))
pcondindep(.1,.2)
pcondcomono(.1,.2)
pcondcountermono(.9,.2)
```

copMarkovts *maximum likelihood estimation and negative log-likelihood for copula-based Markov count time series models*

Description

maximum likelihood estimation and negative log-likelihood for copula-based Markov count time series models

Usage

```
mltscop1(y,x,pcop=pbvncop,start,family="Po",iprint=F,cparlb=0,cparub=30,prlevel=
mltscop2(y,x,pcop3=pmxid3ls,pcop2=pmxid2ls,start,family="Po",iprint=F,cparlb=0,c
rmspe.mccop1(param,yy,xdat=0,pcop=pbvncop,family="Po",iprint=F)
rmspe.mccop2(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,family="Po",iprint=F)
rmspe.tvn(param,yy,xdat=0,family="Po",iprint=F)
nb1mcnllk(param,yy,xdat=0,pcop=pbvncop,cparlb=0,cparub=30) # NB1, Markov order1
nb2mcnllk(param,yy,xdat=0,pcop=pbvncop,cparlb=0,cparub=30) # NB2, Markov order1
gp1mcnllk(param,yy,xdat=0,pcop=pbvncop,cparlb=0,cparub=30) # GP1, Markov order1
gp2mcnllk(param,yy,xdat=0,pcop=pbvncop,cparlb=0,cparub=30) # GP2, Markov order1
pomcnllk(param,yy,xdat=0,pcop=pbvncop,cparlb=0,cparub=30) # Poisson, Markov orde
# Markov order 2 count time series below
nb1mc2nllk(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,cparlb=0,cparub=30)
nb2mc2nllk(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,cparlb=0,cparub=30)
gp1mc2nllk(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,cparlb=0,cparub=30)
gp2mc2nllk(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,cparlb=0,cparub=30)
pomc2nllk(param,yy,xdat=0,pcop3=pmxid3ls,pcop2=pmxid2ls,cparlb=0,cparub=30)
# Markov order 2 count time series based on trivariate Gaussian below
nb1ar2nllk(param,yy,xdat)
nb2ar2nllk(param,yy,xdat)
gp1ar2nllk(param,yy,xdat)
gp2ar2nllk(param,yy,xdat)
poar2nllk(param,yy,xdat)
```

Arguments

y	count times series vector (nx1)
x	matrix of covariates (n x ncovar), where ncovar is number of covariates/predictors
yy	count times series vector
xdat	matrix of covariates, 0 by default for no covariates
param	parameter vector (includes univariate and dependence parameters)
start	starting point of param for nlm optimization
pcop	function for bivariate copula cdf for Markov order 1 models
pcop3	function for trivariate copula cdf for Markov order 2 models
pcop2	function for bivariate marginal copula cdf for Markov order 2 models
family	univariate count regression model: one of "Po", "NB1", "NB2", "GP1", "GP2"
cparlb	lower bound for copula parameters
cparub	upper bound for copula parameters
iprint	print flag for of intermediate results
prlevel	print.level for nlm()

Details

mltscop1 is a front end to rmspe.mccop1 and one of nb1mcnllk, nb1mcnllk, gp1mcnllk, gp2mcnllk, pomcnllk for maximum likelihood of copula-based Markov order 1 count time series model.

mltscop2 is a front end to rmspe.mccop2 and one of nb1mc2nllk, nb1mc2nllk, gp1mc2nllk, gp2mc2nllk, pomcnllk for maximum likelihood of copula-based Markov order 2 count time series model.

Value

negative log-likelihood for any of the nllk functions,

$rmse and $pred (root mean square prediction error and 1-step predictions for any of the rmspe functions,

$nllk, $mle (maximum likelihood estimate), $acov (asymptotic covariance matrix, $rmpse for the mltscop1, mltscop2 functions

Examples

```
data(wcb8594)
y=wcb8594$nclaims
cat("Frank NB2, 2 covariates\n")
ucpar2=c(1.7,-.2,-.2,.4,1.5)
frknb2=mltscop1(y,wcb8594[,2:3],pfrk,start=ucpar2,fam="NB2",iprint=TRUE)
cat("pos stable LT + Frank in trivariate mixture of max-id\n")
par1=c(1.7,-.2,-.2,.4,1.5,1.1)
lsfrknb1=mltscop2(y,wcb8594[,2:3],pcop3=pmxid3ls,pcop2=pmxid2ls,start=par1,
  family="NB1", iprint=TRUE,cparlb=c(0,0),cparub=30)
cat("AR(2) Gaussian/ NB1\n")
nb1param=c(1.7,-.2,-.2,.4,.5,.3)
out=nlm(nb1ar2nllk,p=nb1param,hessian=TRUE,steptol=1.e-4,yy=y,xdat=wcb8594[,2:3],
  print.level=1)
acov=solve(out$hess)
SE=sqrt(diag(acov))
cat("SEs=",SE,"\n")
outpred=rmspe.tvn(out$estimate,y,wcb8594[,2:3],family="NB1",iprint=TRUE)
cat("RMSPE=",outpred$rmse,"\n")
```

---

copmultiv *multivariate copula cdfs*

Description

multivariate copula cdfs

Usage

```
pmfrk(uu,cpar) # multivariate exchangeable Frank
pmgum(uu,cpar) # multivariate exchangeable Gumbel
pmgal(uu,cpar) # multivariate exchangeable Galambos
pmhr(uu,cpar) # multivariate Huesler-Reiss dimension d>=3
      # cpar is d*(d-1)/2 dimensional vector with order 12, 13, 23, 14, ...
mhrA(ww,param) # exponent function of multivariate Huesler-Reiss for d>=3
hrcondcor(thmat,j) # conditional correlation matrix in jth term of A function
```

Arguments

uu	vector of dimension d, each element in (0,1)
cpar	parameter of the d-variate copula
ww	vector of dimension d, each element in (0,Inf)
param	parameter of the multivariate distribution

thmat	d*(d-1)/2 dimensional Huesler-Reiss parameter vector as a symmetric matrix
j	integer index in 1:d, when d is nrow(thmat)

Details

pmhr() requires library mvtnorm

Value

cdf for the pm functions;

a non-negative value for mhrA;

a (d-1)x(d-1) correlation matrix for hrcondcor.

See Also

rectmult

Examples

```
pmgum(c(.8,.9,.7),2)
pmfrk(c(.8,.9,.7),2)
pmgal(c(.8,.9,.7),3)
library(mvtnorm)
pmhr(c(.8,.9,.7),c(.5,.8,1.2))
ww=-log(c(.8,.9,.7))
exp(-mhrA(ww,c(.5,.8,1.2))) # as above with pmhr()
thmat=corvec2mat(c(.5,.8,1.2))
hrcondcor(thmat,1)
hrcondcor(thmat,2)
hrcondcor(thmat,3)
```

---

coptrivmxid *Trivariate copula cdfs from mixture of max-id*

Description

Trivariate copula cdfs from mixture of max-id and bivariate (1,2) margin. These are functions for copula-based Markov order2 time series.

Usage

```
pmxid3ps(uu,param)    # positive stable LT and pfrk for H12,H32
pmxidr3ps(uu,param)   # reflected pmxid3ps
pmxid3ls(uu,param)    # log series LT and pfrk for H12,H32
ptmm1(uu,param)       # positive stable LT and pgum for H12,H32 (example of MM1)
pmxid2ps(u,v,param)   # bivariate (1,2) margin
pmxidr2ps(u,v,param)
pmxid2ls(u,v,param)
pbmm1(u,v,param)
pcondbmm1(v,u,param) # conditional cdf of pbmm1 with respect to u
dbmm1(u,v,param)      # density of pbmm1
```

Arguments

uu	vector of size 3 with numbers in (0,1) for trivariate copula
u	value in interval 0,1 for bivariate copula
v	value in interval 0,1 for bivariate copula
param	copula parameter =(LTpar,cpar12)

Value

cdf or pdf

Examples

```
u=seq(.1,.9,.2)
v=u
param=c(1.5,1.2)
pmxid2ps(u,v,param)
pmxidr2ps(u,v,param)
pmxid2ls(u,v,param)
pbmm1(u,v,param)
pcondbmm1(v,u,param)
uu=c(.1,.4,.6)
pmxid3ps(uu,param)
pmxidr3ps(uu,param)
pmxid3ls(uu,param)
ptmm1(uu,param)
```

---

cor2pcor *correlations to partial correlations and vice versa for vine array*

Description

correlations to partial correlations and vice versa for vine array (C vine, D vine or general R vine)

Usage

```
cor2pcor.cvine(rr)
pcor2cor.cvine(pc)
cor2pcor.dvine(rr)
pcor2cor.dvine(pc)
cor2pcor.rvine(rr,A)
pcor2cor.rvine(pc,A,byrow=T)
pcor2cor.1tr(pp,A)  # 1-truncated vine
pcor2cor.2tr(pp,A)  # 2-truncated vine
pcor2cor.3tr(pp,A,iprint=F)  # 3-truncated vine
pcor2cor.truncvine(pp,A,ntrunc,iprint=F)
vineResidVar(rmatobj,A,ntrunc)
```

Arguments

rr	correlation matrix, dimension d
pc	partial correlation array, dimension d, stored in C-vine or D-vine format
pp	partial correlation array, dimension d, stored in format of vine array A
byrow	TRUE if partial correlations by tree are stored by rows in pc, FALSE if partial correlation in location (j,k) of pc is $\rho_{j,k;S}$ for a conditioning set S.
A	dxd vine array with 1:d on diagonal; only upper triangle is used
ntrunc	truncation level of vine; only first ntrunc rows of A are used
rmatobj	object from pcor2cor.truncvine() with $rmat and $phmat
iprint	print flag for intermediate results

Details

pcor2cor.1tr, pcor2cor.2tr, pcor2cor.3tr are specific versions of pcor2cor.truncvine; pcor2cor.1tr is called by pcor2cor.truncvine and the other two are absorbed into pcor2cor.truncvine.

Value

pc	from cor2pcor.cvine and cor2pcor.dvine, partial correlation array
pcobj	from cor2pcor.rvine, list with $pctree and $pcmat, the former with partial correlations by tree in rows, the latter with $\rho_{j,k;S}$ in location (j,k).
rr	correlation matrix from pcor2cor.cvine, pcor2cor.dvine, pcor2cor.rvine
rmatobj	component $rmat for correlation matrix and $phmat for matrix of regression coefficients from pcor2cor.truncvine
psi2	vector of residual variances from vineResidVar

See Also

cor2reg

Examples

```
d=5
D=Dvinearray(d)
C=Cvinearray(d)
A=vnum2array(d,bnum=3)
# bnum in 0 to 2^dpow-1 where dpow=2^dcase, dcase=(d-2)*(d-3)/2
rr=toeplitz(c(1,.5,.25,.125,.05))
cor2pcor.dvine(rr)
cor2pcor.rvine(rr,D)
cor2pcor.cvine(rr)
cor2pcor.rvine(rr,C)
pcobj=cor2pcor.rvine(rr,A)
pcor2cor.rvine(pcobj$pctree,A)
pcor2cor.rvine(pcobj$pcmat,A,byrow=FALSE)
rmatobj=pcor2cor.truncvine(pcobj$pctree,A,ntrunc=3)
print(rmatobj$phmat)
print(rmatobj$rmat)
vineResidVar(rmatobj,A,3)
```

cor2reg *correlations to regression coefficients and vice versa for vine array*

Description

correlations to regression coefficients and vice versa for vine array (general R-vine); the matrix of regression coefficients provides a simple way for simulation for a multivariate normal distribution with correlation matrix with the structure of a truncated vine based on the order of variables in the vine array.

Usage

```
cor2reg(rr,A,iprint=F)
reg2cor(phm,A)
```

Arguments

rr	correlation matrix, dimension d
A	dxd vine array with 1:d on diagonal; only upper triangle is used.
phm	matrix of regression coefficients phi[j,k], dimension d
iprint	print flag for intermediate results

Value

phm	matrix of regression coefficients from cor2reg
rr	correlation matrix from reg2cor

See Also

cor2pcor

Examples

```
d=5
C=Cvinearray(d)
pp=matrix(0,d,d)
pp[1,2:d]=0.3
set.seed(123)
pp[1,2:d]=runif(d-1,0,1)
pp[2,3:d]=0.4
rr2=pcor2cor.cvine(pp)
print(rr2)
print(cor2reg(rr2,C,iprint=FALSE))
pp[3,4:d]=0.6
rr3=pcor2cor.cvine(pp)
print(rr3)
phm=cor2reg(rr3,C)
print(phm)
print(reg2cor(phm,C))
```

`cormat` *Functions on correlation matrices*

Description

Functions on correlation matrices

Usage

```
corvec2mat(rvec) # convert correlation vector to matrix
cormat2vec(rmat) # extract vector from correlation matrix
corDis(Rmod,Robs,n=0,npar=0) # discrepancy between model-based and observed corr
```

Arguments

`rvec`	vector of length d*(d-1)/2 with order r[1,2],r[1,3],r[2,3],r[1,4], ..., r[d-1,d]
`rmat`	dxd correlation matrix
`Rmod`	model-based correlation matrix
`Robs`	observed correlation matrix
`n`	sample size for Robs
`npar`	parameter vector size leading to Rmod

Value

dxd correlation matrix for corvec2mat

vector of length d*(d-1)/2 for cormat2vec

log(det(Rmod))-log(det(Robs))+sum(diag(solve(Rmod,Robs)))-nrow(Robs) for corDis assuming a Gaussian dependence model

Examples

```
rvec=c(.3,.4,.5,.4,.6,.7)
Rmod=corvec2mat(rvec)
print(Rmod); print(chol(Rmod))
print(cormat2vec(Rmod))
robsvec=c(.32,.38,.53,.41,.61,.67)
Robs=corvec2mat(robsvec)
print(corDis(Rmod,Robs))
print(corDis(Rmod,Robs,n=400,npar=3))
```

cparbound *Copula parameter bounds*

Description

Copula parameter bounds

Usage

```
cparbound(copname)
data(copparbounds)
```

Arguments

copname	string for abbreviation of copula name

Details

Functions for copula cdf, pdf, conditionals, simulation do not check on the domain of the parameter space (or other function arguments), because it is difficult to handle all of the possibilities on the input arguments being a scalar, vector or matrix. These functions can be used as arguments into functions for dependence measures and maximum likelihood. Use simple inputs to check how functions work. Numerical evaluation is different from mathematics; numerical instability (e.g., roundoffs, NaN) might occur when copula parameters are such that dependence is very strong (nearly comonotonic). For use of these functions for numerical maximum likelihood, if a copula parameter bound is Inf, use a parameter value where Kendall's tau is in the range 0.9 to 0.99.

Value

line number in the copparbounds object with bounds of the copula parameter(s); the lower and upper bound(s) are printed out.

See Also

pcop

Examples

```
cparbound("bvncop")
cparbound("pla")
cparbound("gum")
cparbound("bb1")
cparbound("bb1rpow")
cparbound("none")
data(copparbounds)
copparbounds # to see entire list of bounds
```

dbvn *bivariate normal (Gaussian) and t densities*

Description

bivariate normal (Gaussian) and t densities, and logarithm of densities

Usage

```
dbvn(x,rho)
dbvn2(x1,x2,rho)
logdbvn(x1,x2,rho)
dbvt(x1,x2,param,log=FALSE)
```

Arguments

x	2-dimensional vector
x1	scalar or vector
x2	scalar or vector
rho	correlation in (-1,1)
param	(rho,nu) with rho in (-1,1) and nu>0
log	T to return log density, =F to return density

Value

bivariate normal density (could be vector for dbvn2 if one or more of x1,x2,rho is a vector)

Examples

```
dbvn(c(1,.5),.5)
dbvn2(seq(.2,1,.2),seq(.1,.5,.1),.5)
dbvt(1,.5,c(.5,50))
jpdf=dbvt(1,.5,c(.5,5))
dbvt(1,.5,c(.5,5),log=TRUE)
log(jpdf)
```

depmeas2cpar *Convert from a dependence measure to copula parameter*

Description

Convert from a dependence measure to copula parameter

Usage

```
depmeas2cpar(values,type="beta",copname="gumbel")
```

Arguments

values	vector of values between 0 and 1
type	"beta", "tau", "rhoS" or "rhoN"; see deptab
copname	copula family: current choices are "plackett", "frank", "mtcj", "joe", "gumbel", "galambos", "huesler-reiss".

Details

The output values are obtained by interpolation from tables.

Value

copula parameters with the given dependence measure values

See Also

deptab deppar2taurhobetalambda pcinterpolate

Examples

```
values=c(0.29,0.79,0.41,0.88)
depmeas2cpar(values,"beta","gumbel")
depmeas2cpar(values,"tau","galambos")
depmeas2cpar(values,"rhoS","frank")
depmeas2cpar(values,"rhoN","plackett")
```

---

depmeasAsympVar *Asymptotic variance of Kendall's tau, Spearman's rho and Blomqvist's beta for a bivariate copula*

Description

Asymptotic variance of Kendall's tau, Spearman's rho and Blomqvist's beta for a bivariate copula

Usage

```
ktau.avar(cpar,pcop,pcond12,pcond21,zero=0,tol=1.e-5)
rhoS.avar(cpar,pcop,pcond12,pcond21,nq=25,zero=0,tol=1.e-5)
blomqvist.avar(cpar,pcop)
```

Arguments

cpar	copula parameter: scalar or vector
pcop	function for bivariate copula cdf
pcond12	function for $C_{1\|2}(u\|v)$
pcond21	function for $C_{2\|1}(v\|u)$
zero	boundary is [zero,1-zero]^2 for integration, default is zero, but choose something like zero=1.e-6 if there is a boundary problem
tol	desired accuracy for numerical integration, default 0.00001
nq	number of quadrature points per dimension for an inner integral. default is 25

Value

asymptotic variance = avar, so that approximate variance for sample size n is avar/n.

See Also

bivdepmeas

Examples

```
cpar=2
avarkt=ktau.avar(cpar,pfrk,pcondfrk,pcondfrk)
avarsp=rhoS.avar(cpar,pfrk,pcondfrk,pcondfrk,nq=25)
avarbl=blomqvist.avar(cpar,pfrk)
print(c(cpar,avarkt,avarsp,avarbl))
```

---

deppar2taurhobetalambda

Bivariate copulas: mappings of copula parameter and dependence measures

---

Description

Bivariate copulas: mappings of copula parameter (cpar) and Kendall's tau, Spearman's rho, Blomqvist's beta, tail dependence lambda where these are simple.

Usage

```
bvn.b2cpar(b)  # b=beta
bvn.cpar2b(rho)
bvn.cpar2rhoS(rho)
bvn.cpar2tau(rho)
bvt.cpar2b(cpar)
bvt.cpar2lm(cpar)
bvt.lm2cpar(lm,nu)  # lm=lambda
pla.b2cpar(b)
pla.cpar2rhoS(cpar)
pla.rhoS2cpar(rhoS, cpar0=1.5,mxiter=25,eps=1.e-6,iprint=F,mxstep=10)
frk.b2cpar(b, cpar0=0,mxiter=20,eps=1.e-8,iprint=F)
frk.cpar2rhoS(cpar)
frk.cpar2tau(cpar)
mtcj.b2cpar(b, cpar0=0,mxiter=20,eps=1.e-8,iprint=F)
mtcj.cpar2lm(cpar)
mtcj.cpar2tau(cpar)
mtcj.lm2cpar(lm)
mtcj.tau2cpar(tau)
joe.b2cpar(b, cpar0=0,mxiter=20,eps=1.e-8,iprint=F)
joe.cpar2lm(cpar)
joe.cpar2tau(cpar)
joe.lm2cpar(lm)
gum.b2cpar(b)
gum.cpar2lm(cpar)
```

```
gum.cpar2rhoS(cpar)
gum.cpar2tau(cpar)
gum.lm2cpar(lm)
gum.tau2cpar(tau)
gal.b2cpar(b)
gal.cpar2lm(cpar)
gal.cpar2rhoS(cpar)
gal.cpar2tau(cpar)
gal.lm2cpar(lm)
hr.b2cpar(b)
hr.cpar2lm(cpar)
hr.cpar2rhoS(cpar)
hr.cpar2tau(cpar)
hr.lm2cpar(lm)
bb1.b2cpar(b,de,thstart=1,mxiter=30,eps=1.e-6)
bb1.cpar2lm(cpar)
bb1.cpar2tau(cpar)
bb1.lm2cpar(lmpar)
#and other similar functions (see Details below)
```

Arguments

cpar	copula parameter (scalar or vector), rho and df for bvt.cpar2lm
rho	parameter for bivariate normal or t
tau	Kendall's tau for the copula
rhoS	Spearman's rho for the copula
b	Blomqvist's beta for the copula
de	second parameter for BB1 copula
lm	tail dependence for the copula if just upper (lower) tail dependence
nu	degree of freedom parameter for bivariate t
lmpar	vector with (lml,lmu) for lower and upper tail dependence parameters
cpar0	starting point for Newton-Raphson iterations; there is a default starting point in some cases if cpar=0 is specified
mxiter	maximum number of iterations
eps	tolerance for convergence
iprint	print flag for iterations
mxstep	bound on step size for Newton-Raphson iterations
thstart	starting point for Newton-Raphson iterations for bb1.b2cpar

Details

For abbreviations of names of copula families (before the dot in function names), see pcop help page.

Function names after the dot are abbreviations:

b2cpar for Blomqvist's beta to copula parameter (also for bvt)

lm2cpar for tail dependence lambda to copula parameter (also for bb4, bb7)

rhoS2cpar for Spearman's rho to copula parameter

tau2cpar for Kendall's tau to copula parameter (also for ipsA)

cpar2b for copula parameter to Blomqvist's beta (also for tev)

cpar2lm for copula parameter to tail dependence lambda (also for tev, bvt, bb4, bb7)

cpar2rhoS for copula parameter to Spearman's rho (also for tev, bb5)

cpar2tau for copula parameter to Kendall's tau (also for tev, bvt, bb2, bb3, bb5, bb6, bb7, bb8, bb9, bb10, imitlefA, ipsA)

Value

copula parameter or one of tau, rhoS, beta, lambda

See Also

pcop

Examples

```
pla.rhoS2cpar(.5,cpar0=1.84)
be=seq(.1,.9,.1)
cpar=pla.b2cpar(be)
beta=4*ppla(.5,.5,cpar)-1
print(rbind(cpar,beta))
bb1.lm2cpar(c(.4,.6))
frk.cpar2tau(2)
```

---

deptab *Tables of dependence measures for 1-parameter bivariate copula families*

Description

Tables (dataframes) in deptabder are pla.deptab, frk.deptab, mtcj.deptab, joe.deptab, gum.deptab, gal.deptab, hr.deptab with columns: cpar, beta, tau, rhoS, rhoN, lambda, tauder, rhoSder, rhoNder.

The bivariate copula families are Plackett, Frank, MTCJ=Mardia-Takahasi-Cook-Johnson, Joe-B5, Gumbel, Galambos, Huesler-Reiss.

cpar=copula parameter, beta=Blomqvist's beta, tau=Kendall's tau, rhoS=Spearman's rho, rhoN=correlation of normal scores, lambda=tail dependence parameter, tauder=(estimated) derivative of tau with respect to beta, rhosder=(estimated) derivative of rhos with respect to beta, rhoNder=(estimated) derivative of rhoN with respect to beta. The latter three are used for pcinterpolate() for interpolation in the function depmeas2cpar (dependence measure to copula parameter). Given one of tau, rhoS or rhoN, the corresponding value of beta is interpolated from these tables, and then inverted to get the copula parameter.

Usage

```
data(deptabder)
```

Format

Tables included are the following.

`pla.deptab` Dependence measure table for bivariate Plackett copula

`frk.deptab` Dependence measure table for bivariate Frank copula

`mtcj.deptab` Dependence measure table for bivariate MTCJ copula

`joe.deptab` Dependence measure table for bivariate Joe copula

`gum.deptab` Dependence measure table for bivariate Gumbel copula

`gal.deptab` Dependence measure table for bivariate Galambos copula

`hr.deptab` Dependence measure table for bivariate Huesler-Reiss copula

See Also

`pcinterpolate` `depmeas2cpar`

`discreteresponse` *Copula models for count response data*

Description

Copula models for count response data, including discretized MVN diagnostics from bivariate margins

Usage

```
latentBVNnllk1(rho,param,ucdf,xdat1,xdat2,y1,y2)
latentBVNnllk2(rho,par1,par2,cdf1,cdf2,xdat1,xdat2,y1,y2)
ieenllk(param,upmf,ydat,xdat,LB,UB) # independent estimating equations
MVNlatent1(ydat,xdat,nrep,upmf,ucdf,upmfcdf,mx,ustart,LB,UB,prlevel=0)
   # discretized multivariate normal/Gaussian with univariate marginal
   # regression model being specified by upmf/ucdf and common
   # regression parameters for each margin
MVNlatent2(ydat,xdat,nrep,unllks,upmfs,ucdfs,upmfcdfs,mx,ustart,LB,UB,prlevel=0)
   # discretized multivariate normal/Gaussian with univariate marginal
   # regression models being specified by upmfs/ucdfs and different
   # regression parameters for each margin
vinebivEvs01(ydat,xdat,nrep,mx,uparam,upmfcdf,cpar,A,pcop)
```

Arguments

`rho`	latent correlation parameter
`param`	common parameter vector for univariate margin of the latent bivariate normal model; assumed estimated in a previous step in latentBVNnllk1
`par1`	univariate parameter vector for univariate margin 1 of the latent bivariate normal model; assumed estimated in a previous step
`par2`	univariate parameter vector for univariate margin 2 of the latent bivariate normal model; assumed estimated in a previous step
`uparam`	univariate parameter vector, common to all univariate margin

ucdf	function for univariate cdf
upmf	function for common univariate pmf for ieenllk
upmfcdf	function that efficiently computes univariate pmf and cdf up to an upper limit mx
unllks	vector of strings for univariate negative log-likelihoods
upmfs	vector of strings for pmf functions for margins 1,...,d
ucdfs	vector of strings for cdf functions for margins 1,...,d
upmfcdfs	vector of strings for pmf/cdf functions for computing up to mx
mx	bound used for Expected vs Observed tables in univariate/bivariate margins
cdf1	function for univariate cdf for margin 1
cdf2	function for univariate cdf for margin 1
xdat1	covariates for margin 1 (nxq), q=#covariates
xdat2	covariates for margin 2 (nxq), q=#covariates
xdat	covariate matrix ((n*d)xq), q=#covariates, d=#repeated measurements
y1	count response 1, vector of length n
y2	count response 2, vector of length n
ydat	count response, vector of length n*d
nrep	#repeated measurements per subject
ustart	starting parameter point for univariate model
LB	lower bound for param
UB	upper bound for param
A	dxd vine array with 1:d on diagonal
pcop	function for copula cdf (with scalar parameter) in tree 1 of vine
cpar	parameter values for pair-copulas in tree 1 of vine
prlevel	print.level for nlm()

Value

negative log-likelihood value for latentBVNnllk1, latentBVNnllk2;

negative of sum of univariate log-likelihoods for ieenllk;

list(uparam vector or parmat matrix,rhvec,E1arr,O1arr,E2arr,O2arr) [univariate parameter estimates. latent correlation estimates, Expected and Observed counts for univariate/bivariate] for MVNlatent1,MVNlatent2;

list(rhosub,E1arr,O1arr,E2arr,O2arr) for mvnbivEvsO1 where rhosub is based on subset and rhopar is based on full data set;

list(E2arr,O2arr) for vinebivEvsO1 with Expected and Observed counts for bivariate margins in first tree of vine.

See Also

rvinediscrete

Examples

```
# count regression model with common betas for all margins: longitudinal count
data(rwmsubset)
rwm=rwmsubset
rwm$agec=(rwm$age-50)/10
rwm$ageq=(rwm$agec)^2
rwm$handfra=rwm$handper/100
xdat=cbind(rwm$sex,rwm$agec,rwm$ageq,rwm$hsat,rwm$handfra,rwm$univ)
xdat=as.matrix(xdat)
ydat=rwm$docvis
nc=ncol(xdat)
nc1=nc+1
out=MVNlatent1(ydat,xdat,nrep=5,upmf=nb1pmf,ucdf=nb1cdf,upmfcdf=nb1pmfcdf,
  mx=7,ustart=c(1.7,.3,.2,.1,-.2,.7,-.5,1.5),
  LB=c(rep(-20,nc1),0),UB=rep(10,nc1+1),prlevel=1)
print(out$uparam)
print(out$rhvec)
#
# GP1 count regression models for each margin: uncommon regression coefficients
data(kzrepmeas)
kz=kzrepmeas
kz$agehun=kz$age/100
xdat=cbind(kz$agehun,kz$sex,kz$msmok)
ydat=kz[,6]
nrep=4
outgp1=MVNlatent2(ydat,xdat,nrep,unllks=rep("gp1nllk",4),ucdfs=rep("gp1cdf",4),
  upmfcdfs=rep("gp1pmfcdf",4),
  mx=3,ustart=c(0,0,0,0,1),LB=c(-20,-20,-20,-20,0),UB=rep(10,5),prlevel=0)
print(outgp1$uparam)
print(outgp1$rhvec)
```

---

euro0306 *Log returns for some Euro markets*

Description

Log returns multipled by 100 for cac40 dax ftse oseax smi, years 2003-2006;

dimension 957x6, first column is date.

Original source is http://quote.yahoo.com

Usage

```
data(euro0306)
```

`euro07` *log returns and GARCH-filtered log returns for some Euro markets 2007*

Description

Log returns and GARCH filtered values of log returns for OSEAX FTSE AEX FCHI SSMI GDAXI ATX (market indexes in Norway, UK, Holland, France, Switzerland, Germany, Austria). This is a small data set with n=239 that can be used for illustration of functions for fitting vine and factor copulas. The original source is http://quote.yahoo.com

euro07lr has two objects: (i) euro07names has the above labels for the markets and (ii) euro07lr is a 239x7 matrix of log returns in 2007, one column for each market (the markets were merged to common dates) before returns were obtained

euro07gf has two objects: (i) euro07names has the above labels for the markets and (ii) euro07lr is a list with several matrices given below.

Usage

```
data(euro07lr) # objects euro07names and euro07lr
data(euro07gf) # objects euro07names and euro07gf
```

Format

The following are components.

filter 239x7 matrix of GARCH filtered returns

uscore 239x7 matrix of empirical U(0,1) scores of GARCH filtered returns

zscore 239x7 matrix of empirical normal scores of GARCH filtered returns

uscmodel 239x7 matrix of U(0,1) scores of GARCH filtered returns based on assuming standardized Student t distributions for the innovations

zscmodel 239x7 matrix of normal scores of GARCH filtered returns based on assuming standardized Student t distributions for the innovations

sigmat 239x7 matrix of volatilities

coef 5x7 matrix of GARCH parameter estimates rows are mu, omega, alpha1, beta1, shape, where 'shape' is the shape or degree of freedom parameter for the Student t innovations.

`exchmvn` *Exchangeable (positive) multivariate normal*

Description

Rectangle probability and derivatives of positive exchangeable multivariate normal, and trivariate normal

Usage

```
exchmvn(lb,ub,rho, mu=0,scale=1,eps=1.e-06)
exchmvn.deriv.margin(lb,ub,rho,k,ksign, eps=1.e-06)
exchmvn.deriv.rho(lb,ub,rho, eps=1.e-06)
pmnorm(lb,ub,mu,sigma, eps=1.e-05)
```

Arguments

lb	vector of lower limits of integral/probability
ub	vector of upper limits of integral/probability
rho	correlation (positive constant over pairs)
mu	mean vector
scale	standard deviation
eps	tolerance for numerical integration
k	margin for which derivative is to be taken, that is, derivative of exchmvn(lb,ub,rho) with respect to lb[k] or ub[k]; use exchmvn.deriv.rho for derivative of exchmvn(lb,ub,rho) with respect to rho
ksign	value is -1 for derivative of exchmvn(lb,ub,rho) with respect to lb[k], value is +1 for derivative of exchmvn(lb,ub,rho) with respect to ub[k]
sigma	covariance matrix

Details

The positive exchangeable multivariate normal distribution has a stochastic representation as a one-factor model from which rectangle probabilities can be written as 1-dimensional integrals. pmnorm() from Schervish (1984) is recommended only for dimension d=3; otherwise use pmvnorm() in library mvtnorm.

Value

rectangle probability or a derivative

References

Kotz S and Johnson NL (1972). Continuous Multivariate Distributions. Wiley, New York, page 48.

Schervish M (1984). Multivariate normal probabilities with error bound. Applied Statistics, 33, 81-94.

See Also

fact1mvn

Examples

```
# The tests here show clearly what the function parameters are.
# step size for numerical derivatives (accuracy of exchmvn etc about 1.e-6)
heps = 1.e-4

cat("case 1: m=3\n")
m=3
a=c(-1,-1,-1)
```

```
b=c(2,1.5,1)
rho=.6
pr=exchmvn(a,b,rho)
cat("pr=exchmvn(avec,bvec,rho)=",pr,"\n")
cat("derivative wrt rho\n")
rho2=rho+heps
pr2=exchmvn(a,b,rho2)
drh.numerical= (pr2-pr)/heps
drh.analytic= exchmvn.deriv.rho(a,b,rho)
cat("   numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")

cat("derivative wrt a_k,b_k, k=1,...,",m,"\n")
for(k in 1:m)
{ cat("  k=", k, " lower\n")
  a2=a
  a2[k]=a[k]+heps
  pr2=exchmvn(a2,b,rho)
  da.numerical = (pr2-pr)/heps
  da.analytic= exchmvn.deriv.margin(a,b,rho,k,-1)
  cat("   numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
  cat("  k=", k, " upper\n")
  b2=b
  b2[k]=b[k]+heps
  pr2=exchmvn(a,b2,rho)
  db.numerical = (pr2-pr)/heps
  db.analytic= exchmvn.deriv.margin(a,b,rho,k,1)
  cat("   numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}


cat("\ncase 2: m=5\n")
m=5
a=rep(-1,m)
b=c(2,1.5,1,1.5,2)
rho=.6
pr=exchmvn(a,b,rho)
cat("pr=exchmvn(avec,bvec,rho)=",pr,"\n")
cat("derivative wrt rho\n")
rho2=rho+heps
pr2=exchmvn(a,b,rho2)
drh.numerical= (pr2-pr)/heps
drh.analytic= exchmvn.deriv.rho(a,b,rho)
cat("   numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")

cat("derivative wrt a_k,b_k, k=1,...,",m,"\n")
for(k in 1:m)
{ cat("  k=", k, " lower\n")
  a2=a
  a2[k]=a[k]+heps
  pr2=exchmvn(a2,b,rho)
  da.numerical = (pr2-pr)/heps
  da.analytic= exchmvn.deriv.margin(a,b,rho,k,-1)
  cat("   numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
  cat("  k=", k, " upper\n")
  b2=b
  b2[k]=b[k]+heps
  pr2=exchmvn(a,b2,rho)
```

```
  db.numerical = (pr2-pr)/heps
  db.analytic= exchmvn.deriv.margin(a,b,rho,k,1)
  cat("   numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}
```

`extremevalue` *Maximum likelihood for GEV and generalized Pareto*

Description

Maximum likelihood for generaluzed extreme value and generalized Pareto, using modified Newton-Raphson

Usage

```
gevmle(xdata, maxitn=20)
gpmle(xdata ,maxitn=20)
dgev(x, xi=1,mu=0,sigma=1)
pgev(x, xi=1,mu=0,sigma=1)
qgev(p, xi=1,mu=0,sigma=1)
logdgev(x, xi=1,mu=0,sigma=1)
dgpareto(x,xi, sigma=1)
pgpareto(x,xi, sigma=1)
qgpareto(p,xi, sigma=1)
```

Arguments

`xdata`	data set, should be positive-valued exceedances for gpmle
`maxitn`	maximum number of iterations for Newton-Raphson method
`x`	scalar of vector
`p`	scalar of vector, values in 0 to 1
`xi`	tail index or shape parameter of GEV
`mu`	location parameter
`sigma`	scale parameter

Value

list with $loglik, $params, $covar for gevmle and gpmle functions

density, cdf, quantile or log density values with the other functions

Examples

```
set.seed(123)
x=rnorm(2000)
xmat=matrix(x,40,50)
mxdat=apply(xmat,1,max)
gevmle(mxdat) # xi is negative because of sub-asymptotics
gpmle(x[x>1.3]-1.3)
rpareto=function(n,alp,s)
```

```
{ u=runif(n); tem=u^(-1/alp)-1; return(s*tem) }
set.seed(123)
x=rpareto(2000,alp=3,s=1)
xmat=matrix(x,40,50)
mxdat=apply(xmat,1,max)
gevmle(mxdat) # xi close to 1/alp=1/3
gpmle(x[x>1.2]-1.2)
```

`fact1mvn` *Rectangle probability for multivariate normal with 1-factor structure*

Description

Rectangle probability for multivariate normal with 1-factor structure

Usage

```
fact1mvn(lb,ub,load1,eps=1.e-6,B=6)
```

Arguments

`lb`	vector of lower limits of integral/probability
`ub`	vector of upper limits of integral/probability
`load1`	loading vector for factor 1
`eps`	tolerance for numerical integration
`B`	upper limit of integration and negative of lower limit

Details

For the 1-factor dxd correlation structure, the rectangle probability is a 1-dimensional integral. The positive exchangeable correlation matrix is a special case.

Value

rectangle probability

See Also

`exchmvn`

Examples

```
d=5
lb=rep(-1,d)
ub=c(2,1.5,1,1.5,2)
rho=.6
pr=exchmvn(lb,ub,rho)
cat("pr=exchmvn(lb,ub,rho)=",pr,"\n")
load1=rep(sqrt(rho),d)
pr2=fact1mvn(lb,ub,load1)
cat("pr=fact1mvn(lb,ub,load1)=",pr2,"\n")
# higher-dimensional example
```

```
load=c(.9,.8,.2,.2,.2,.2,.2,.2)
d=length(load)
uvec=seq(.3,.9,.2)
for(i in 1:length(uvec))
{ y=qnorm(uvec[i])
  jcdf=fact1mvn(rep(-6,d),rep(y,d),load)
  cat(y,jcdf,"\n")
}
```

`factanal.bi` *Gaussian factor analysis: common and bi-factor*

Description

Gaussian factor analysis: common and bi-factor

Usage

```
pfactnllk(rhvec,Robs,nsize)
bifactnllk(rhvec,grsize,Robs,nsize)
trifactnllk(rhvec,grsize,sbgrsize,Robs,nsize)
factanal.co(factors,start,data=1,cormat=NULL,n=100,prlevel=0,mxiter=100)
factanal.bi(grsize,start,data=1,cormat=NULL,n=100,prlevel=0,mxiter=100)
factanal.tri(grsize,sbgrsize,start,data=1,cormat=NULL,n=100,prlevel=0,mxiter=150
```

Arguments

`factors`	the number of factors p for factanal.co
`rhvec`	For pfactnllk: vector of length d*p of partial correlations, first d are correlations with factor 1, then partial correlations with factor k given previous factors for k=2,...,p. For bifactnllk: vector of length d*2; first d correlations with common factor, then partial correlations with group factor given common factor. For trifactnllk: vector of length d*3; first d correlations with common factor, then partial correlations with group factor given common factor, and finally partial correlations with subfactor given common and groups factors.
`Robs`	dxd correlation matrix
`nsize`	sample size
`grsize`	vector of group sizes for the bi-factor and tri-factor models
`sbgrsize`	vector of subgroup sizes for the tri-factor model
`start`	starting point for nlm() to find p-factor or bi-factor or tri-factor parameters to minimize the Gaussian negative log-likelihood
`data`	nsize x d data set to compute the correlation matrix
`cormat`	dxd correlation matrix
`n`	sample size
`prlevel`	print.level for nlm
`mxiter`	maximum number of iterations for nlm

Details

The output of factanal.co should be similar to that of factanal(). The algorithm is different so that it may fail to converge depending on the starting point (when p is larger).

pfactnllk, bifactnllk and trifactnllk return the gradient vector in order that gradient calculations are faster and maybe the number of iterations in nlm() can be reduced .

Value

For pfactnllk, bifactnllk and trifactnllk, a value for negative Gaussian log-likelihood and the gradient vector with respect to rhvec.

For factanal.co, a list with $nllk, $rhmat= dxp matrix of partial correlations, $loading=dxp loading matrix after varimax, $rotmat=pxp rotation matrix used by varimax. Note that the loading matrix and matrix of partial correlations are not unique for p>=2.

For factanal.bi, a list with $nllk, $rhmat= dx2 matrix of correlations and partial correlations.

For factanal.tri, a list with $nllk, $rhmat= dx3 matrix of correlations and partial correlations.

See Also

bifct mvtfact

Examples

```
rhpar=c( 0.81,  0.84,0.84, 0.54,0.57,0.49, 0.51,0.54,0.55,0.70,
  0.53,0.56,0.53,0.67,0.70)
Robs=corvec2mat(rhpar)
n=100
d=6
rhvec=c(rep(.5,d),rep(.4,d))
pfactnllk(rhvec,Robs,n)
factanal.co(2,start=rhvec,cormat=Robs,n=100,prlevel=1)
# bi-factor
grsize=c(3,3)
bifactnllk(rhvec,grsize,Robs,n)
factanal.bi(grsize,start=rhvec,cormat=Robs,n=100,prlevel=1)
# tri-factor
grsize=c(4,4)
sbgrsize=c(2,2,2,2)
d=sum(grsize)
p=3
tripar=((d*p):1)/(d*p+1)
param1=tripar[1:d]
param2=tripar[(d+1):(2*d)]
param3=tripar[(2*d+1):(3*d)]
n=50
robj=trifct(grsize,sbgrsize,param1,param2,param3)
achol=chol(robj$fctmat)
set.seed(123)
z=matrix(rnorm(n*d),n,d)
z=z
zdata=nscore(z,iopt=TRUE)
Robs=cor(zdata)
trifactnllk(tripar,grsize,sbgrsize,Robs,n)
factanal.tri(grsize,sbgrsize,start=tripar,cormat=Robs,n=50,prlevel=1,mxiter=150)
```

`factanal2nllk` *negative log-likelihood via factor analysis*

Description

negative log-likelihood via factanal, assuming data are standardized

Usage

```
factanal2nllk(rmat,mxfactor,n,iprint=F)
```

Arguments

`rmat`	correlation matrix of data
`mxfactor`	maximum number of factors
`n`	sample size
`iprint`	print flag for intermediate results

Value

vector of length mxfactor with the negative log-likelihoods for factor models with 1,...,mxfactor factors

See Also

`factanal.bi`

Examples

```
data(euro07gf);
zdat=euro07gf$zscore # 239 x 7
rr=cor(zdat)
n=nrow(zdat)
out=factanal2nllk(rr,3,n,iprint=TRUE)
```

`factorcopcdf` *Bivariate marginal copula cdfs for 1-factor and 2-factor copula models*

Description

Bivariate marginal copula cdfs for 1-factor and 2-factor copula models

Usage

```
pfact1cop(u1,u2,pcondcop,param,nq)
pcondfact1(u2,u1,pcondcop,dcop,param,nq)
dfact1cop(u1,u2,dcop,param,nq)
pfact2cop(u1,u2,pcondcop1,pcondcop2,param1,param2,nq)
pcondfact2(u2,u1,pcondcop1,pcondcop2,dcop1,dcop2,param1,param2,nq)
dfact2cop(u1,u2,pcondcop1,dcop1,dcop2,param1,param2,nq)
pfact1frk(u1,u2,param)  # nq defaulted to 35 etc
pfact2frk(u1,u2,pmatrix)  # nq defaulted to 35 etc
```

Arguments

u1	vector of values in interval 0,1;
u2	vector of values in interval 0,1; same length as u1
param	vector of length 2 or 2xq matrix where q is number of parameters
param1	vector of length 2 or 2xq matrix where q is number of parameters for the bivariate copula (e.g. BB1) in pcondcop; parameters that link observed variables to latent factor 1
param2	vector of length 2, parameters that link observed variables to latent factor 2
pcondcop	function for conditional cdf of linking copula for 1-factor model
pcondcop1	function for conditional cdf of linking copula for factor 1 for 2-factor model
pcondcop2	function for conditional cdf of linking copula for factor 1 for 2-factor model
dcop	function for pdf of linking copula for 1-factor model
dcop1	function for pdf of linking copula for factor 1 for 2-factor model
dcop2	function for pdf of linking copula for factor 2
nq	number of quadrature points for Gauss-Legendre quadrature
pmatrix	For most choices, first column has parameters that link observed variables to latent factor 1, second column has parameters that link observed variables to latent factor 2. For pfact2bb1frk: first two columns for BB1 parameters, third column for Frank for latent factor 2.

Value

cdf or conditional cdf or pdf value(s)

See Also

factorcopsim

Examples

```
th1a=frk.b2cpar(.7)
th1b=frk.b2cpar(.6)
th2a=frk.b2cpar(.5)
th2b=frk.b2cpar(.4)
u1=seq(.1,.9,.2)
u2=seq(.3,.7,.1)
pfact1frk(u1,u2,c(th1a,th1b))
pfact2frk(u1,u2,matrix(c(th1a,th1b,th2a,th2b),2,2))
```

`factorcopmle` *MLE and negative log-likelihood for factor copula models*

Description

MLE and negative log-likelihood for factor copula models (both common and structured factors). f90 in function name means the log-likelihood and derivatives are computed in fortran90 code; note that ml2fact is same as f90ml2fact.

Usage

```
ml1fact(nq,start,udata,dcop,LB=0,UB=1.e2,prlevel=0,mxiter=100)
ml1factb(nq,start,ifixed,udata,dcop,LB=0,UB=1.e2,prlevel=0,mxiter=100)
f90ml1fact(nq,start,udata,copname,LB=0,UB=40,ihess=F,prlevel=0,mxiter=100,nu=3)
f90ml2fact(nq,start,udata,copname,LB=0,UB=40,repar=0,ihess=F,prlevel=0,mxiter=10
f90ml2factb(nq,start,ifixed,udata,copname,LB=0,UB=40,ihess=F,prlevel=0,mxiter=10
f90cop1nllk(param,dstruct, iprfn=F) # 1-factor
f90cop2nllk(param,dstruct, iprfn=F) # 2-factor
```

Arguments

`param`	parameter for f90cop1nllk and f90cop2nllk, these functions are input to pdhessmin or pdhessminb
`nq`	number of quadrature points
`start`	starting point of param for nlm optimization
`ifixed`	logical vector of same length as param, ifixed[i]=TRUE iff param[i] is fixed at the given value
`udata`	nxd matrix of values in (0,1)
`dcop`	function for bivariate copula density
`copname`	"frank", "gumbel", "gumfrank" (see below)
`dstruct`	structure that includes $quad for the gauss-legendre nodes and weights, $copname for the model, and $data of form udata For t-factor model, also $nu for degree of freedom parameter. Also $repar as a code for reparametrization (check examples).
`LB`	lower bound of components of param, usually of length(param), could also be a scalar for a common lower bound
`UB`	upper bound of components of param, usually of length(param), could also be a scalar for a common upper bound
`nu`	degree of freedom parameter for 1-factor model with t copulas
`nu1`	degree of freedom parameter for first factor of 2-factor model with t copulas
`nu2`	degree of freedom parameter for second factor 2-factor model with t copulas
`repar`	repar=1 for reparametrization for Gumbel copula (cpar->1+param^2) and repar=2 for BB1 copula with second parameter (delta->1+param^2)
`ihess`	option for hessian in nlm()
`prlevel`	print.level in nlm()
`mxiter`	max number of iterations or iterlim in nlm()
`iprfn`	flag for printing of function value and derivatives

Details

ml1fact (ml1factb) uses only R code; f90ml1fact, f90ml2fact, f90ml2factb link to Fortran 90 code; ml2fact (ml2factb) is the same as f90ml2fact (f90ml2factb).

Current options are the following (but user can use the f90 code as templates for adding other linking copulas).

(1a) f90ml1fact: "frank", "gumbel", "t", "bb1"

(1b) f90cop1nllk: "frank", "lfrank", "gumbel", "t", "bb1"

(2) f90cop2nllk and f90ml2fact (f90ml2factb): "frank", "lfrank", "gumbel", "gumfrank", "bb1frank", "t", "tapprox" (latter uses monotone interpolation for the Student t cdf).

For BB1, the order of parameters is theta[1],delta[1],...,theta[d],delta[d], where thetas>0 and deltas>0.

Value

$fnval, $grad, $hess for f90cop1nllk and f90cop2nllk;

MLE etc for ml1fact and ml2fact.

References

Krupskii P and Joe H (2013). Factor copula models for multivariate data. Journal of Multivariate Analysis, 120, 85-101.

See Also

factorcopsim mvtfact structcop

Examples

```
# 1-factor
cpar.frk=c(12.2,3.45,4.47,4.47,5.82); d=5
n=300
set.seed(123)
frkdat=sim1fact(n,cpar.frk,qcondfrk,"frk")
cat("\nFrank 1-factor MLE: standalone R and then f90\n")
out.frk=ml1fact(nq=21,cpar.frk,frkdat,dfrk,LB=-30,UB=30,prlevel=1,mxiter=100)
gl21=gausslegendre(21)
dstrfrk=list(copname="frk",data=frkdat,quad=gl21,repar=0)
out=pdhessminb(cpar.frk,f90cop1nllk,ifixed=rep(FALSE,d),dstruct=dstrfrk,
  LB=rep(-30,d),UB=rep(30,d),iprint=TRUE,eps=1.e-5)
set.seed(123)
dfdefault=5
tdat=sim1fact(n,c(.6,.7,.6,.8,.65),qcondt,"t")
dstrt=list(copname="t",data=tdat,quad=gl21,repar=0,nu=10)
out=pdhessminb(rep(.6,d),f90cop1nllk,ifixed=rep(FALSE,d),dstruct=dstrt,
  LB=rep(-1,d),UB=rep(1,d),iprint=TRUE,eps=1.e-5)
dstrt=list(copname="t",data=tdat,quad=gl21,repar=0,nu=5)
out=pdhessminb(rep(.6,d),f90cop1nllk,ifixed=rep(FALSE,d),dstruct=dstrt,
  LB=rep(-1,d),UB=rep(1,d),iprint=TRUE,eps=1.e-5)
# 2-factor
d=7; be1=c(.7,.6,.7,.6,.7,.6,.7); be2=c(.4,.4,.4,.4,.3,.3,.3)
cpar1.frk=frk.b2cpar(be1); cpar2.frk=frk.b2cpar(be2)
n=300
set.seed(123)
frkdat=sim2fact(n,cpar1.frk,cpar2.frk,qcondfrk,qcondfrk,"frk","frk")
```

```
cat("\nFrank 2-factor MLE: nlm and then pdhessmin\n")
out.frk=ml2fact(nq=21,c(cpar1.frk,cpar2.frk),frkdat,copname="frank",
  LB=-30,UB=30,prlevel=1,mxiter=100)
dstrfrk=list(copname="frank",data=frkdat,quad=gl21,repar=0)
out=pdhessminb(c(cpar1.frk,cpar2.frk),f90cop2nllk,ifixed=rep(FALSE,2*d),
  dstruct=dstrfrk, LB=rep(-20,2*d),UB=rep(20,2*d),iprint=TRUE,eps=1.e-5)
```

`factorcopsim` *Simulation for factor copulas*

Description

Simulation for 1-factor copula and 2-factor copula, all bivariate linking copulas in same parametric family for each factor

Usage

```
sim1fact(n,parobj1,qcond1,copname1,ivect=F)
sim2fact(n,parobj1,parobj2,qcond1,qcond2,copname1="a1",copname2="a2",ivect=F)
```

Arguments

`n`	sample size
`parobj1`	parameter vector of dimension d or parameter matrix with d rows, where d is dimension of factor copula; parobj is dx2 for something like BB1 copula
`parobj2`	parameter vector of dimension d or parameter matrix with d rows, for linking copulas to factor 2
`qcond1`	name of conditional copula function $C^{-1}_{U\|V}(u\|v)$, choices include qfrk, qgum, qgumr, qbb1, qt with fixed nu1.
`qcond2`	name of conditional copula function $C^{-1}_{U\|V}(u\|v)$ for second factor, choices include qfrk, qgum, qgumr, qt with fixed nu2.
`copname1`	copula name: the function checks on "frank", "mtcj", "mtcjr", "fgm" for which qcond has closed form.
`copname2`	copula name for factor 2
`ivect`	flag that is T if qcond1 and qcond2 have vectorized forms

Value

data matrix of dimension nxd

References

Krupskii P and Joe H (2013). Factor copula models for multivariate data. Journal of Multivariate Analysis, 120, 85-101.

See Also

`factorcopmle` `rfactcop`

Examples

```
bevec=c(.8,.7,.6,.5,.5)
cpar.frk=frk.b2cpar(bevec)
lmbb1=matrix(c(.3,.4,.5,.3,.5, .6,.6,.6,.7,.7),5,2)
cpar.bb1=lmbb1
for(i in 1:nrow(lmbb1))
{ cpar.bb1[i,]=bb1.lm2cpar(lmbb1[i,]) }
n=300
set.seed(123)
frkdat=sim1fact(n,cpar.frk,qcondfrk,"frank")
print(cor(frkdat))
set.seed(123)
bb1dat=sim1fact(n,cpar.bb1,qcondbb1,"bb1",ivect=FALSE)
print(cor(bb1dat))
set.seed(123)
bb1dat=sim1fact(n,cpar.bb1,qcondbb1,"bb1",ivect=TRUE)
print(cor(bb1dat))
# pairs(bb1dat)
bevec2=c(.0001,.6,.6,.6,.7)
cpar.frk2=frk.b2cpar(bevec2)
frk2dat=sim2fact(n,cpar.frk,cpar.frk2,qcond1=qcondfrk,qcond2=qcondfrk,"frank","frank")
print(cor(frk2dat))
```

`fmdepmeas` *Spearman rho matrix for 1-factor, 2-factor, nested factor copulas and multivariate Gaussian factor model*

Description

Spearman rho matrix for 1-factor, 2-factor, nested factor copulas and multivariate Gaussian factor model

Usage

```
srho1fact(param1,pcondcop,nq)
srho2fact(param1,param2,pcondcop1,pcondcop2,nq)
rho2nestfact(param1,param2,dcop1,pcondcop2,nq)
rhomvn(fct,param) # matrix of Spearman rho for MVN factor model
cop2srho(param,pcop,nq) # Spearman rho via Gauss-Legendre quadrature
```

Arguments

`param1`	vector/matrix of copula parameters for linking copulas to factor 1 or global latent variable
`param2`	vector of copula parameters for linking copulas to factor 2 or group latent variable
`fct`	number of factors
`param`	dxfct matrix of partial correlation parameters for multivariate normal/Gaussian factor model with fct factors
`pcondcop`	function for conditional cdf of copula family for factor 1

pcondcop1	function for conditional cdf of copula family for factor 1
pcondcop2	function for conditional cdf of copula family for factor 2 or group latent variable
dcop1	function for bivariate copula pdf for global latent variable
pcop	function for bivariate copula cdf
nq	number of quadrature points for Gauss-Legendre quadrature

Details

2-dimensional Gauss-Legendre quadrature for 1-factor model, 4-dimensional Gauss-Legendre quadrature for 2-factor model.

Value

dxd matrix for srho1fact, srho2fact, rhomvn

single number for rho2nestfact, cop2srho

See Also

factorcopsim structcop

Examples

```
n=40000
cpar=frk.b2cpar(seq(.7,.8,.02)); cpar2=frk.b2cpar(seq(.5,.6,.02))
set.seed(123)
frk1dat=sim1fact(n,cpar,qcondfrk,"frank")
set.seed(123)
frk2dat=sim2fact(n,cpar,cpar2,qcond1=qcondfrk,qcond2=qcondfrk,"frank","frank")
# 1-factor
r1=cor(frk1dat)
print(r1)
r1t=srho1fact(cpar,pcondfrk,nq=15)
print(r1t)
print(max(abs(r1-r1t)))
# 2-factor
r2=cor(frk2dat)
print(r2)
r2t=srho2fact(cpar,cpar2,pcondfrk,pcondfrk,nq=15)
print(r2t)
print(max(abs(r2-r2t)))
# nested-factor
grsize=c(4,4)
set.seed(123)
parne=c(c(4,3),rep(6,4),rep(6.5,4))
udatne=simnestfact(n=600, grsize, cop=5, parne) # Frank pair-copulas
parwglobal=c(4,3)
parwgrp=c(6,6.5)
srh=rho2nestfact(parwglobal,parwgrp,dfrk,pcondfrk,21)
print(srh)
print(cor(udatne[,1],udatne[,5:8]))  # should be round same as above
# Spearman's rho for bivariate copula (which could be margin of factor copula)
cop2srho(c(2.5,2.8), pfact1gum, 35)  # Gauss-Legendre
rhoS(c(2.5,2.8),pfact1gum)  # cubature
cop2srho(matrix(c(2.5,2.8,1.5,1.6),2,2), pfact2gum, 35)
```

`gammaconvfactor` *Gamma convolution factor model*

Description

Gamma convolution factor model and its copula

Usage

```
rgammaconv(n,th0,thvec)
pmgamfact(xvec,th0,thvec,zero=0)
dmgamfact(xvec,th0,thvec,zero=0)
dmgamfact.gl(xvec,th0,thvec,gl)
pbgamfact(x1,x2,th0,th1,th2,zero=0)
dbgamfact(x1,x2,th0,th1,th2,zero=0)
dbgamfact.gl(x1,x2,th0,th1,th2,gl)
pbgamfcop(u,v,param)
dbgamfcop(u,v,param,zero=0)
dbgamfcop.gl(u,v,param,gl=gldefault)
pcondbgamfcop21(v,u,param,zero=0)
pcondbgamfcop12(u,v,param,zero=0)
pmgamfcop(uvec,param)
dmgamfcop(uvec,param,zero=0)
dmgamfcop.gl(uvec,param,gl)
```

Arguments

`n`	sample size for simulation
`th0`	scalar for shape parameter of the shared/common component
`thvec`	vector of shape parameters of individual components, length d
`xvec`	vector of length d with positive values
`uvec`	vector of length d with values in (0,1)
`gl`	Gauss-Legendre object with components $nodes and $weights
`x1`	positive value for first variable (bivariate case)
`x2`	positive value for second variable (bivariate case)
`th1`	scalar for shape parameter of first variable (bivariate case)
`th2`	scalar for shape parameter of second variable (bivariate case)
`zero`	tolerance for numerical integration, set as 0.0001 if there are problems
`u`	value in interval 0,1; could be a vector
`v`	value in interval 0,1; could be a vector
`param`	parameter vector with length d+1 with th0,thvec

Value

random sample (nxd matrix) for rgammaconv

cdf or conditional cdf or pdf for remaining functions

See Also

invGaussconvfactor

Examples

```
n=1000
th0=2
thvec=c(.3,.3)
set.seed(123)
xdat=rgammaconv(n,th0,thvec)
cor(xdat)
#plot(xdat)
gl=gausslegendre(35)
pmgamfact(c(1,1.1),th0,c(.4,.4),zero=0)
dbgamfact(1,1.1,th0,th1=.4,th2=.4,zero=0)
dbgamfact.gl(1,1.1,th0,th1=.4,th2=.4,gl) # could be inaccurate for th1,th2<1
dbgamfact(1,1.1,th0,th1=1.2,th2=1.4,zero=0)
dbgamfact.gl(1,1.1,th0,th1=1.2,th2=1.4,gl)
dmgamfact(c(1,1.1),th0,c(1.2,1.4),zero=0)
dmgamfact.gl(c(1,1.1),th0,c(1.2,1.4),gl)
# density can be finite on diagonal
try(dbgamfact(1,1.0001,0.1,th1=.4,th2=.4,zero=0))
try(dbgamfact(1,1,0.1,th1=.4,th2=.4,zero=0))
# copula
pmgamfcop(c(.5,.6),c(2,1.2,1.4))
dmgamfcop(c(.5,.6),c(2,1.2,1.4),zero=0)
dmgamfcop.gl(c(.5,.6),c(2,1.2,1.4),gl)
pcondbgamfcop21(.6,.5,c(2,1.2,1.4),zero=0)
pcondbgamfcop12(.5,.6,c(2,1.2,1.4),zero=0)
pcondbgamfcop21(.5,.6,c(2,1.4,1.2),zero=0)
```

---

garchfilter *GARCH filter applied separately to log returns*

Description

GARCH filter applied separately to a matrix of log returns, the d log returns should have the same dates, a subset of rows can be chosen

Usage

```
gfiltersubset(lgret,ar,m1,m2,iprint=F)
```

Arguments

lgret	Nxd matrix of log returns of d financial assets
ar	TRUE for univariate GARCH(1,1)-AR(1) for model, FALSE for GARCH(1,1) model
m1	first row of lgret to use for the subset
m2	last row of lgret to use for the subset
iprint	print flag for GARCH estimates, F by default

Details

This function requires library fGarch and its dependent libraries.

Value

filter	GARCH filtered data (nxd, where n=m2-m1+1)
uscore	empirical uniform scores (nxd)
zscore	empirical normal scores (nxd)
uscmodel	model-based uniform scores (nxd)
zscmodel	model-based normal scores (nxd)
sigmat	matrix of estimated volatilities (nxd)
coef	matrix of GARCH parameters (6xd or 5xd for ar=T or ar=F respectively), the parameters are mu, (ar1), omega, alpha1, beta1, shape.

Examples

```
## Not run:
data(euro07lr) # euro07lr data set is 239x7
library(fGarch)
out=gfiltersubset(euro07lr,ar=FALSE,1,200,iprint=TRUE)
## End(Not run)
```

---

gausslegendre *Gauss-Legendre quadrature*

Description

Gauss-Legendre quadrature nodes and weights

Usage

```
gausslegendre(nq)
```

Arguments

nq	number of quadrature points

Value

nodes	vector of length nq of nodes in the interval (0,1)
weights	vector of length nq, sum of weights is 1

References

Original source for the code is "Stroud A and Secrest D (1966), Gaussian Quadrature Formulas, Prentice-Hall."

Examples

```
out=gausslegendre(15)
# same as gauss.quad.prob(15,dist="uniform") in library(statmod)
print(sum(out$weights))  # should be 1
print(sum(out$weights*out$nodes))  # should be 0.5
```

---

gausstrvine *Best Gaussian truncated vines*

Description

Best Gaussian truncated d-dimensional vines up to d-2 trees

Usage

```
gausstrvine(rmat,iprint=F)
gausstrvine.nonuniq(rmat,jtrunc=3,eps=1.e-7,iprint=F) # check non-uniqueness
```

Arguments

rmat	dxd correlation matrix, 4<=d<=8
iprint	print flag for intermediate steps (in f90 code)
jtrunc	truncation level to check on degree of non-uniqueness
eps	tolerance to check on degree of non-uniqueness, default 1e-7

Details

Note that even if the optimal ell-truncated vine is unique, the vine array leading to it is not unique. The output of gausstrvine.nonuniq() is not saved into R variables, so it should be looked at for other truncated vines that lead to the same determinant as the optimal.

Value

bnum	d-2 dimensional vector with indices of best vine arrays; can get vine arrays with something like vnum2array(d, bnum[ell])
logdetmx	d-1 dimensional vector with max log determinants for truncated vines of order 1,...,d-2; the last entry in position d-1 is the log determinant of rmat
permmat	dx(d-2) matrix, with permutation leading to a best ell-truncated vine in column ell
pcarr	dxdx(d-2) partial correlation array with matrix of partial correlations in pcarr[,,ell] for the best ell-truncated vine

See Also

gausstrvineMST

Examples

```
## Not run:
rmat=matrix(c(
1.00000,0.69965,0.70477,0.66536,0.65967,
0.69965,1.00000,0.65499,0.61713,0.61202,
0.70477,0.65499,1.00000,0.62967,0.62798,
0.66536,0.61713,0.62967,1.00000,0.57398,
0.65967,0.61202,0.62798,0.57398,1.00000), 5,5)
out=gausstrvine(rmat,iprint=FALSE)
print(out)
outnonuniq=gausstrvine.nonuniq(rmat,jtrunc=3,eps=1.e-7,iprint=TRUE)
# some checks
d=nrow(rmat)
for(ell in 1:(d-2))
{ A=vnum2array(d,out$bnum[ell])
  cat("truncation level ", ell,"\n")
  print(A)
  cat("check on log determinant\n")
  pcmat=out$pcarr[,,ell]
  logdet=sum(log(1-pcmat[1:ell,]^2))
  print(logdet)
}
print(determinant(rmat,log=TRUE)$modulus)
## End(Not run)
```

`gausstrvineMST` *Gaussian truncated d-dimensional vines based on sequential minimum spanning trees*

Description

Gaussian truncated d-dimensional vines based on sequential minimum spanning trees with weights of one minus squared partial correlation

Usage

```
gausstrvine.mst(rmat,ntrunc,iprint=F)
```

Arguments

`rmat`	dxd correlation matrix
`ntrunc`	specified upper bound in truncation level to consider
`iprint`	print flag for intermediate results

Details

This function depends on the minimum spanning tree algorithm in the library igraph0.

Value

RVM	object with RVMVineA = d-dimensional vine array, RVMpc = partial correlations by tree, RVMMatrix = vine array in VineCopula format [d:1,d:1], RVMCor = partial correlations in VineCopula format [d:1,d:1]
mst	spanning trees 1,2,...d-1: $mst[[1]], $mst[[2]], ...
treeweight	vector of length d-1 with sum_edge log(1-rho[edge]^2) for trees 1,...d-1
trunclevel	same as inputted ntrunc
truncval	sum treeweight[1:trunclevel] / sum treeweight[1:(d-1)]

See Also

gausstrvine

Examples

```
## Not run:
d=5
library(igraph0) # version 0.5.6 works
rmat=matrix(c(
1.00,0.76,0.76,0.74,0.67,
0.76,1.00,0.91,0.93,0.86,
0.76,0.91,1.00,0.94,0.85,
0.74,0.93,0.94,1.00,0.88,
0.67,0.86,0.85,0.88,1.00), d,d)
colnames(rmat) = rownames(rmat) = paste("V",1:d,sep="")
out=gausstrvine.mst(rmat,ntrunc=3,iprint=TRUE)
print(out)
## End(Not run)
```

---

genbeta2 *Copula based on generalized beta of order 2*

Description

Copula based on generalized beta of order 2

Usage

```
dbgb2(y1,y2,param) # bivariate density
pbgb2(y1,y2,param)  # bivariate cdf
dbgb2cop(u,v,param) # bivariate copula density
pbgb2cop(u,v,param)  # bivariate copula cdf
pcondbgb2cop(v,u,param)  # bivariate copula conditional cdf
dmgb2(yvec,param)    # multivariate density
dmgb2cop(uvec,param)    # multivariate copula density
logdmgb2cop(uvec,param) # log of multivariate copula density
rmgb2cop(n,param)  # simulation
bgb2.cpar2lm(param) # upper tail dependence
mgb2.cpar2cor(param) # correlation matrix if zeta>2
```

Arguments

y1	positive value; could be a vector
y2	positive value; could be a vector
yvec	positive vector of length d
u	value in interval (0,1); could be a vector
v	value in interval (0,1); could be a vector
uvec	vector of length d; values in (0,1)
param	parameter vector of dimension d+1; (eta1,eta2,zeta) for bivariate and (etavec,zeta) for d-variate; all parameters are positive
n	simulation sample size

Value

pdf for dbgb2, dmgb2, dbgb2cop, dmgb2cop;

cdf for pbgb2, pbgb2cop;

conditional cdf for pcondbgb2cop;

log density for logdmgb2cop (use for maximum likelihood);

random d-vectors for rmgb2cop;

upper tail dependence parameter for bgb2.cpar2lm;

correlation matrix for mgb2.cpar2cor

References

Yang X, Frees EW, Zhang Z (2011). A generalized beta copula with applications in modeling multivariate long-tailed data. Insurance: Mathematics and Economics 49, 265-284.

Examples

```
y1=2; y2=3;
param=c(1.5,1.8,2.3)
dbgb2(y1,y2,param)
dmgb2(c(y1,y2),param) # same as above
u1=.2; u2=.3;
dbgb2cop(u1,u2,param)
dmgb2cop(c(u1,u2),param) # same as above
n=1000
param=c(4.5,4.8,2.3)
set.seed(124)
udat=rmgb2cop(n,param)
print(cor(udat))
cat("compare theoretical\n")
print(mgb2.cpar2cor(param))
dmgb2copnllk= function(param,udat)
{ n=nrow(udat)
  if(any(param<=0)) return(1.e10)
  nllk=0
  for(i in 1:n) nllk=nllk-logdmgb2cop(udat[i,],param)
  nllk
}
out=nlm(dmgb2copnllk,p=param,udat=udat,hessian=TRUE,print.level=1,iterlim=20)
```

```
print(sqrt(diag(solve(out$hess))))
## Not run:
# contour of density with N(0,1) margins
zvec=seq(-3,3,.2)
contourBivCop(param,zvec,dcop=dbgb2cop)
## End(Not run)
```

---

gpois *Generalized Poisson cdf and pmf*

Description

Generalized Poisson cdf and pmf

Usage

```
gpoispmfcdf(ub,theta,vrh)
dgpois(y,param)
pgpois(y,param)
```

Arguments

param	2-vector with theta=convolution parameter and vrh=second parameter in (0,1)
theta	convolution parameter >0
vrh	second parameter between 0 and 1
y	non-negative integer (or vector for dgpois)
ub	non-negative integer

Value

table of size (ub+1)x3 with integer, pmf and cdf for gpoispmfcdf;

pmf for dgpois (can be vectorized);

cdf for pgpois;

See Also

gpoisson

Examples

```
theta=2.5
vrh=0.5
gpoispmfcdf(6,theta,vrh)
dgpois(0:6,c(theta,vrh))
pgpois(6,c(theta,vrh))
```

gpoisson *Generalized Poisson regression for count data*

Description

Generalized Poisson (GP) regression for count data; 2 versions. GP(theta,vrho) has pmf f(y;theta,vrho)= [theta*(theta+vrho y)^(y-1)]*exp(-theta-vrho*y) / y! .

Usage

```
gp1nllk(param,y,xdat)
gp2nllk(param,y,xdat)
gp1pmfcdf(ub,param,x)
gp2pmfcdf(ub,param,x)
gp1cdf(y,param,x)
gp2cdf(y,param,x)
gp1pmf(y,param,x)
gp2pmf(y,param,x)
```

Arguments

param	parameter of GP model, length is 2+number of covariates; the parameters are: b0=intercept, bvec= vector regression coefficients (length(bvec)=length(x)=ncol(xdat), and xi (for GP1) or theta (for GP2). For GP1, mu(x)= exp(b[0]+bvec^T x), xi=(overdispersion index minus one) and 1-vrho=sqrt(1/(1+xi)) are fixed, and theta(x)=mu(x)*(1-vrh). For GP2, mu(x)= exp(b[0]+bvec^T x), theta=convolution parameter is fixed and 1-vrho(x)=theta/mu(x).
xdat	matrix for gp1nllk and gp2nllk
x	vector for gp1pmfcdf, gp2pmfcdf, gp1cdf, gp2cdf, gp1pmf, gp2pmf
y	vector for gp1nllk and gp2nllk (with length(y)=nrow(xdat)); non-negative integer for the other functions
ub	upper bound integer for which pmf and cdf are computed

Value

negative log-likelihood for gp1nllk and gp2nllk;

matrix with columns (0:ub,pmf,cdf) for gp1pmfcdf and gp2pmfcdf, computed in an efficient way (parameters assumed to be such that most probability is on small counts);

cdf for gp1cdf and gp2cdf;

pmf for gp1pmf and gp2pmf.

See Also

gpois negbinom

Examples

```
y= c(
 2, 1, 1, 0,35, 9, 0, 1, 4, 0, 0, 1, 4, 0, 0, 8, 7, 2, 0, 7, 0, 0, 3, 4, 0,
 4, 1, 3, 0, 6, 1, 0, 2, 8, 0,12, 0, 4, 2, 1, 3, 0, 9, 0, 0, 0, 2, 0, 8, 1,
 2, 4, 2, 0, 0, 2, 1, 3, 2, 1, 3, 4, 4, 5, 0, 4, 0, 2, 0,28, 1,24, 1, 0,10,
 3, 3, 0, 0, 7, 2, 4, 6, 4,13, 5, 8, 0, 1, 6, 0,24, 9, 0,10, 0, 0, 8, 5, 3,
16, 0, 4, 1, 1, 4,12, 4, 3, 5, 0, 2, 1, 5, 3, 0, 0, 6, 4, 2, 0, 2, 0,15, 3,
 0, 2, 3, 4, 5, 0, 3, 0, 0, 6, 0, 0,15, 0, 0, 0, 1, 3, 0, 1, 0, 4, 2,10, 4,
 1, 0, 0, 0, 5, 0, 0, 2, 0, 4, 0, 0, 2,25, 0, 0,13, 0, 0,21, 3, 0, 0, 0, 2,
 2, 0, 4,13, 2, 9, 9, 2, 0, 1, 2, 2, 8, 6, 0, 4, 1, 2, 0, 0, 0, 0, 0, 0, 2,
 2, 0, 3, 1, 1, 7, 3, 0, 2, 2, 1, 3, 2, 2, 1, 3, 3, 0, 0, 0, 2, 0, 0, 0, 0,
 1, 2, 2, 0, 0, 9, 0, 0, 1, 1, 0, 2,10, 0,17, 2, 0,14, 0, 5, 9, 2, 0, 6, 3,
 3, 1, 0,11, 4, 9, 0, 1, 0, 0,12, 4, 0, 1,21, 0, 3, 2, 0, 1, 0, 1, 3, 8,10,
19, 0, 2, 7, 1, 0, 2, 0, 4, 0, 6, 4, 7, 1, 0, 1, 3, 4, 0, 4)
hsat=c(
 8, 7, 3,10, 6, 5, 8, 9, 9, 8,10, 8, 6, 7,10, 8, 5, 8, 8, 6, 8, 8, 8, 9,10,
 7, 9,10, 8, 6, 6, 9, 7, 5,10, 4, 8, 4, 5, 5, 7, 6, 7,10, 9, 9, 5, 7, 4, 7,
 6, 6, 7, 5,10, 9,10, 7, 8, 6, 5, 5, 0, 5, 7, 3, 8, 8, 7, 5, 5, 0, 7, 6, 3,
10, 7, 7,10, 5, 5, 4, 2, 7, 6, 2, 5,10, 7, 8, 5, 5, 5,10, 3, 9, 6, 8,10,10,
 4, 7, 2, 8, 9, 0, 0, 5, 8, 3, 7, 6,10, 4, 5, 7, 6, 7, 3, 4,10, 4, 8, 8, 3,
 9, 5,10, 9, 5,10,10, 8,10, 5,10, 6, 5, 9, 8,10, 7, 8, 9, 7, 8, 4, 8, 3, 5,
 5, 7,10, 8, 1, 3, 3, 8,10, 3, 5, 5, 7, 5,10, 8, 5, 8, 5, 0, 6, 8, 2, 5, 6,
 7,10, 5, 0, 5, 2, 0, 3,10, 7, 4, 6, 9, 2, 8, 5, 9, 7, 5,10, 8, 8, 7, 7, 7,
10,10, 2, 5, 7, 5, 9, 6, 7, 6, 9, 9, 6, 8,10, 7, 8, 8,10,10, 5,10, 5, 8,10,
 8, 7,10, 9,10, 4, 6, 9, 5, 9, 9, 6, 8, 8, 2, 5, 8, 3, 7, 0, 8, 8,10, 5, 7,
 6, 7,10, 5, 5, 1, 5, 6, 4,10, 5, 5, 5, 7, 2, 8, 5,10,10,10,10, 6, 6, 6, 6,
 7, 8, 8,10,10, 8, 7, 8, 3, 8, 8, 8, 6, 3, 7,10,10, 2, 9, 2)
fit=nlm(ieenllk,p=c(2.5,-.2,5),hessian=TRUE,print.level=1,upmf=gp1pmf,
    xdat=hsat,ydat=y,LB=c(-1,-2,0),UB=c(10,10,10))
```

`imitlefA` *Bivariate Archimedean copula based on integrated Mittag-Leffler LT*

Description

Bivariate Archimedean copula based on integrated Mittag-Leffler Laplace transform

Usage

```
pimitlefA(u,v,cpar)
dimitlefA(u,v,cpar)
pcondimitlefA(v,u,cpar)  # C_{2|1}(v|u;cpar)
qcondimitlefA(p,u,cpar,eps=1e-08,mxiter=30,iprint=F) # C_{2|1}^{-1}(p|u;cpar)
imitlefA.cpar2tau(cpar)
pimitlefAr(u,v,cpar)  # reflected/survival version of pimitlefA
dimitlefAr(u,v,cpar)
pcondimitlefAr(v,u,cpar)
```

Arguments

`u`	value in interval 0,1; could be a vector
`v`	value in interval 0,1; could be a vector

p	quantile in interval 0,1; could be a vector
cpar	parameter vector: 2-dimensional vector (or 2-column matrix) with parameters vth>0 and de>1.
eps	tolerance for convergence
mxiter	maximum number of iterations
iprint	print flag for iterations

Value

(conditional) cdf or pdf or quantile value(s), or Kendall's tau.

References

Joe H (2014). Dependence Modeling with Copulas. Chapman & Hall/CRC.

Examples

```
u=seq(.1,.6,.1)
v=seq(.4,.9,.1)
vth=.5; de=1.6; cpar=c(vth,de)
pp=pcondimitlefA(v,u,cpar)
print(pp)
qcondimitlefA(pp,u,cpar)
cdf=pimitlefA(u,v,cpar)
pdf=dimitlefA(u,v,cpar)
tau=imitlefA.cpar2tau(cpar)
print(tau)
```

---

invgamA *Bivariate Archimedean copula based on inverse gamma LT*

Description

Bivariate Archimedean copula based on inverse gamma Laplace transform

Usage

```
pinvgamA(u,v,cpar)
dinvgamA(u,v,cpar)
pcondinvgamA(v,u,cpar)  # C_{2|1}(v|u;cpar)
rminvgamA(n,d,cpar)
logdinvgamA(u,v,cpar,pgrid=0)
invgamA.cpar2tau(cpar)
invgamA.tau2cpar(tau)
```

Arguments

u	value in interval 0,1; could be a vector
v	value in interval 0,1; could be a vector
cpar	parameter: could be scalar or vector (positive-valued)
n	sample size for ripsA, positive integer
d	dimension
pgrid	grid of values in (0,1) to use for monotone interpolation; see code for the default vector when pgrid is input as 0
tau	Kendall value in (0,1)

Value

cdf, pdf, conditional cdf, conditional quantile value(s) for pinvgamA, dinvgamA, pcondinvgamA, qcondinvgamA respectively;

log density for logdinvgamA (use for maximum likelihood);

random d-vectors for rminvgamA;

Kendall's tau for invgamA.cpar2tau;

copula parameter for invgamA.tau2cpar

References

Joe H and Hua L (2010). Tail order and intermediate tail dependence of multivariate copulas. Journal of Multivariate Analysis, v 102, 1454–1471

Examples

```
u=seq(.1,.6,.1)
v=seq(.4,.9,.1)
cpar=.5
pp=pcondinvgamA(v,u,cpar)
print(pp)
tau=invgamA.cpar2tau(cpar)
print(tau)
set.seed(123)
udata=rminvgamA(500,d=2,cpar=2) # tau=0.5
print(taucor(udata[,1],udata[,2]))
print(semicor(udata,inscore=FALSE))
ml=nlm(bivcopnllk,p=1.5,hessian=TRUE,print.level=1,
  udat=udata,logdcop=logdinvgamA,LB=.0001,UB=10)
## Not run:
# contour of density with N(0,1) margins
zvec=seq(-3,3,.2)
contourBivCop(2,zvec,dinvgamA)
## End(Not run)
```

`invGauss` *Inverse Gaussian distribution*

Description

Inverse Gaussian distribution

Usage

```
pIG(x,mu,vsi)  # cdf with mean parameter mu and second parameter vsi
dIG(x,mu,vsi)  # density
qIG(p,mu,vsi,mxiter=10,eps=1e-06,mxstep=5,iprint=F)  # inverse cdf
rIG(n,mu,vsi)  # simulation of random variables
```

Arguments

`x`	positive value; could be a vector
`p`	value in (0,1); could be a vector
`mu`	mean parameter m of inverse Gaussian
`vsi`	second parameter varsigma=lambda of inverse Gaussian
`mxiter`	maximum number of iterations
`eps`	tolerance for convergence
`mxstep`	bound on step size for Newton-Raphson iterations
`iprint`	print flag for iterations
`n`	simulation sample size

Details

Seshadri (1993): with mu>0, vsi>0 means the pdf is: f(x;mu,vsi)= [sqrt(vsi)/sqrt(2*pi*x^3)] * exp[-(vsi/[2mu^2])*(x-mu)^2/x] for x>0.

Reparametrization has mu=zeta*eta as the mean and vsi=eta^2 where eta=convolution parameter; zeta is like a scale parameter that can be set to 1 for the copula

Property of the cdf is pIG(x,mu,vsi)=pIG(x/mu,1,vsi/mu)

Value

cdf or pdf or quantile or random sample

References

Seshadri V (1993). The Inverse Gaussian Distribution. Clarendon Press.

See Also

`invGaussconvfactor`

Examples

```
mu=3; vsi=2
x=1:10
p=pIG(x,mu,vsi)
xx=qIG(p,mu,vsi)
print(cbind(x,p,xx))
eps=1.e-5
peps=pIG(x+eps,mu,vsi)
print(cbind((peps-p)/eps,dIG(x,mu,vsi)))
set.seed(123)
n=10000
x=rIG(n,mu,vsi)
print(summary(x))
print(var(x))
cat("theoretical values mean=", mu, " var=", mu^3/vsi,"\n")
```

---

invGaussconvfactor *Inverse Gaussian convolution factor model*

Description

Inverse Gaussian convolution factor model

Usage

```
rIGconv(n,th0,thvec,ze=1)
pmIGfact(xvec,th0,thvec,zero=0)
pmIGfact.gl(xvec,th0,thvec,gl)
dmIGfact(xvec,th0,thvec,zero=0)
dmIGfact.gl(xvec,th0,thvec,gl)
dbIGfact(x1,x2,th0,th1,th2,zero=0)
pmIGfcop.gl(uvec,param,gl)
dmIGfcop.gl(uvec,param,gl)
```

Arguments

n	sample size for simulation
th0	scalar for shape parameter of the shared/common component
thvec	vector of shape parameters of individual components, length d
param	parameter vector with length d+1 with th0,thvec
xvec	vector of length d with positive values
uvec	vector of length d with values in (0,1)
gl	Gauss-Legendre object with components $nodes and $weights
ze	non-convolution parameter zeta, can be set to 1 for copula
zero	tolerance for numerical integration, set as 0.0001 if there are numerial problems
x1	positive value for first variable (bivariate case)
x2	positive value for second variable (bivariate case)
th1	scalar for shape parameter of first variable (bivariate case)
th2	scalar for shape parameter of second variable (bivariate case)

Value

random sample (nxd matrix) for rIGconv

cdf or pdf for remaining functions

See Also

invGauss gammaconvfactor

Examples

```
n=1000
th0=2
thvec=c(.3,.3)
set.seed(123)
xdat=rIGconv(n,th0,thvec)
cor(xdat)
#plot(xdat)
gl=gausslegendre(25)
pmIGfact(c(1,1.1),th0,c(.4,.4),zero=0)
pmIGfact.gl(c(1,1.1),th0,c(.4,.4),gl)
# check that density is finite on diagonal
dbIGfact(1,1.1,th0,th1=.4,th2=.4,zero=0)
dmIGfact(c(1,1.1),th0,c(.4,.4),zero=0)
dmIGfact.gl(c(1,1.1),th0,c(.4,.4),gl)
dbIGfact(1,1.0001,th0,th1=.4,th2=.7,zero=0)
# copula
pmIGfcop.gl(c(.5,.6),c(2,.4,.4),gl)
dmIGfcop.gl(c(.5,.6),c(2,.4,.4),gl)
```

---

ipsA *Bivariate Archimedean copula based on integrated positive stable LT*

Description

Bivariate Archimedean copula based on integrated positive stable Laplace transform

Usage

```
pipsA(u,v,cpar)
dipsA(u,v,cpar)
pcondipsA(v,u,cpar)  # C_{2|1}(v|u;cpar)
qcondipsA(p,u,cpar)  # C_{2|1}^{-1}(p|u;cpar)
ripsA(n,cpar)
logdipsA(u,v,cpar)
ipsA.cpar2tau(cpar)
ipsA.tau2cpar(tau,mxiter=20,eps=1e-06, cparstart=0, iprint=F)
```

Arguments

u	value in interval 0,1; could be a vector
v	value in interval 0,1; could be a vector
p	quantile in interval 0,1; could be a vector
cpar	parameter: could be scalar or vector (positive-valued)
n	sample size for ripsA, positive integer
tau	tau in interval (-1,1), could be a vector
mxiter	maximum number of Newton-Raphson iterations
eps	tolerance for convergence of Newton-Raphson iterations
cparstart	starting point for Newton-Raphson iterations
iprint	print flag for Newton-Raphson iterations

Details

For the reflected copula, the functions are pipsAr, dipsAr, pcondipsAr, qcondipsAr, logdipsAr.

Value

cdf, pdf, conditional cdf, conditional quantile value(s) for pipsA, dipsA, pcondipsA, qcondipsA respectively;

log density for logdipsA;

random pairs for ripsA;

Kendall's tau for psA.cpar2tau;

copula parameter for ipsA.tau2cpar or parameter value for a given Kendall's tau.

References

Joe H and Ma C (2000). Multivariate survival functions with a min-stable property. Journal of Multivariate Analysis, 75, 13-35.

Examples

```
u=seq(.1,.6,.1)
v=seq(.4,.9,.1)
cpar=.6
pp=pcondipsA(v,u,cpar)
vv=qcondipsA(pp,u,cpar)
print(pp)
print(vv)
tau=ipsA.cpar2tau(cpar)
print(tau)
tauv=seq(-.9,.9,.1)
cpar=ipsA.tau2cpar(tauv)
print(cpar)
set.seed(123)
udata=ripsA(500,cpar=cpar[15]) # tau=0.5
print(taucor(udata[,1],udata[,2]))
print(cor(udata,method="kendall"))
```

IRfactormle *MLE for discrete factor models for item response*

Description

MLE for discrete factor copula models for item response; f90 in function name means the log-likelihood and derivatives are computed in fortran90 code, ir1nllk and ir2nllk are also based on f90.

Usage

```
ml1irfact(nq,start,ydata,pcond,LB=0,UB=50,ihess=F,prlevel=0,mxiter=50)
f90ml1irfact(nq,start,ydata,copname,LB=0,UB=40,ihess=F,prlevel=0,
   mxiter=50,nu1=3)
ml2irfact(nq,start,ydata,pcond1,pcond2,LB=0,UB=40,ihess=F,prlevel=0,mxiter=50)
f90ml2irfact(nq,start,ydata,copname,LB=0,UB=40,ihess=F,prlevel=0,
   mxiter=50,nu1=5,nu2=5)
f90ml2irfactb(nq,start,ifixed,ydata,copname,LB=0,UB=40,ihess=F,prlevel=0,
   mxiter=50,nu1=5,nu2=5)
ir1nllk(param,dstruct,iprfn=F)
ir2nllk(param,dstruct,iprfn=F)
f90irfisherinfo1(param,ucutp,nq=15,copname,nu1=3,nn=1000)
f90irfisherinfo2(param,ucutp,copname,nq=15,ifixed,nu1=3,nu2=3,nn=1000)
```

Arguments

param	parameter for ir1nllk and ir2nllk, these functions are input to pdhessmin or pdhessminb
nq	number of quadrature points
start	starting point of param for nlm optimization
ifixed	logical vector of same length as param, ifixed[i]=TRUE iff param[i] is fixed at the given value
ydata	nxd integer-valued matrix with values in 0,1,...,ncat-1, ncat is number of ordinal categories.
pcond	function for bivariate copula conditional cdf, linking to latent variable
pcond1	function for copula conditional cdf for factor 1
pcond2	function for copula conditional cdf for factor 2
copname	"gumbel" or "gaussian" or "t" or "gumbelt" (Gumbel for factor 1 and t for factor 2) or "tgumbel" (t for factor 1 and Gumbel for factor 2)
dstruct	structure that includes $quad for the Gauss-Legendre nodes and weights, $copname for the model, $data for ydata, $cutp for the ordinal cutpoints
LB	lower bound of components of param, usually of length(param), could also be a scalar for a common lower bound
UB	upper bound of components of param, usually of length(param), could also be a scalar for a common upper bound
ihess	option for hessian in nlm()
prlevel	print.level in nlm()

mxiter	maximum number of iterations iterlim in nlm()
ucutp	(ncat-1)xd matrix of cutpoints in (0,1)
nu1	df parameter for factor 1 if copname="t" or "gumbelt" or "tgumbel"
nu2	df parameter for factor 2 if copname="t" or "gumbelt" or "tgumbel"
iprfn	flag for printing of function value and derivatives
nn	nominal sample size for inverse of Fisher information

Value

$fnval, $grad, $hess for ir1nllk and ir2nllk; MLE etc for ml1irfact and ml2irfact.

$finfo with Fisher information matrix, $SE with sqrt(diagonal of inverse Fisher information matrix divided by nn) for f90irfisherinfo1 and f90irfisherinfo2.

References

Nikoloulopoulos A K and Joe H (2014). Factor copula models for item response data, Psychometrika.

See Also

IRfactorsim factorcopmle

Examples

```
data(ltmconv)
d=ncol(sci)
#1-factor (3 methods)
nq=21
## Not run:
library(abind)  # need abind() for ir1factpmf and ir2factpmf
#                 called by ml1irfact and ml2irfact respectively
ml1a=ml1irfact(nq,start=rep(2,d),sci,pcond=pcondgum,LB=1,UB=20,ihess=TRUE,
  prlevel=1,mxiter=50)
ml1b=f90ml1irfact(nq,start=rep(2,d),sci,copname="gumbel",LB=1,UB=20,ihess=TRUE,
  prlevel=1,mxiter=50)
ucutp=unifcuts(sci)
gl=gausslegendre(nq)
dstrgum=list(copname="gumbel",dat=sci,quad=gl,cutp=ucutp)
ml1c=pdhessmin(param=rep(2,d),ir1nllk,dstruct=dstrgum,LB=rep(1,d),UB=rep(20,d),
  iprint=TRUE,eps=1.e-5);
#2-factor (2 methods)
param=c(1.5,1.1,1.6,2.5,1.05,1.2,1.5,rep(.4,d))
dfdefault=2
ml2a=ml2irfact(nq,start=param,sci,pcond1=pcondgum,pcond2=pcondt,
  LB=c(rep(1,d),rep(-1,d)),UB=c(rep(20,d),rep(1,d)),prlevel=1,mxiter=50)
dstrgumt=list(copname="gumbelt",dat=sci,quad=gl,cutp=ucutp,nu2=2)
ml2b = pdhessmin(param,ir2nllk,dstruct=dstrgumt,
  LB=c(rep(1,d),rep(-1,d)),UB=c(rep(20,d),rep(1,d)),iprint=TRUE,eps=1.e-5);
## End(Not run)
# Fisher information (check for near non-identifiability)
theta=c(0.5,0.6,0.5,0.6,0.4)
delta=c(0.3,0.4,0.3,0.4,0.2)
ifixed=rep(FALSE,2*d)
nq=21
```

```
K=4
d=5
ucut=seq(1:(K-1))/K
ucuts=matrix(rep(ucut,d),ncol=d)
nn=1000
ifixed=rep(FALSE,2*d)
cat("\nt5/t5\n")
finft=f90irfisherinfo2(c(theta,delta),ucutp=ucuts,copname="t",nq=nq,ifixed,
   nu1=5,nu2=5,nn=nn)
print(finft$SE)
cat("\nt20/t20\n")
finft=f90irfisherinfo2(c(theta,delta),ucutp=ucuts,copname="t",nq=nq,ifixed,
   nu1=20,nu2=20,nn=nn)
print(finft$SE)  # larger as cliser to Gaussian non-identifiable case
```

`IRfactorsim` *Simulation for factor copulas for ordinal item response data*

Description

Simulation for factor copulas for ordinal item response data, all bivariate linking copulas in same parametric family for each factor

Usage

```
sim1irfact(ucuts,n,parobj1,qcond1,copname1,ivect=F)
sim2irfact(ucuts,n,parobj1,parobj2,qcond1,qcond2,copname1="",copname2="",ivect=F
```

Arguments

`ucuts`	(ncat-1)xd matrix of cutpoints, increasing in each column and bounded in (0,1); ncat=number of categories
`n`	sample size
`parobj1`	parameter vector of dimension d or parameter matrix with d rows for factor 1, where d is dimension of factor copula; this is dx2 for something like BB1 copula
`parobj2`	parameter vector of dimension d or parameter matrix with d rows for factor 2
`qcond1`	function for copula conditional inverse cdf $C^{-1}_{U\|V}(u\|v)$, choices include qcondfrk, qcondgum, qcondgumr, qcondbb1, qcondbvtcop with fixed nu1.
`qcond2`	function for copula conditional inverse cdf $C^{-1}_{U\|V}(u\|v)$ for second factor, choices include qcondfrk, qcondgum, qcondgumr, qcondbvtcop with fixed nu2.
`copname1`	copula name: the function checks on "frank", "mtcj", "mtcjr", "fgm" for which qcond has closed form.
`copname2`	copula name for factor 2
`ivect`	flag that is T if qcond1 and qcond2 have vectorized forms

Details

These interface to sim1fact and sim1fact, with discretization based on ucuts.

Value

data matrix of dimension nxd with values in 0,...,ncat-1

References

Nikoloulopoulos A K and Joe H (2014). Factor copula models for item response data, Psychometrika.

See Also

`IRfactormle` `factorcopsim`

Examples

```
ucuts=matrix(c(.3,.6,.4,.7,.5,.8),2,3)
param=c(5.5,6.5,4)
set.seed(123)
ydat=sim1irfact(ucuts,n=1000,param,qcond1=qcondfrk,copname1="frank")
print(cor(ydat))
for(j in 1:length(param)) print(table(ydat[,j]))
ydat2=sim2irfact(ucuts,n=1000,param,c(2,2,2),qcond1=qcondgum,qcond2=qcondgum,ivect=TRUE)
print(cor(ydat2))
for(j in 1:length(param)) print(table(ydat2[,j]))
```

---

isposdef *Check if Hessian matrix is positive definite*

Description

Check if Hessian matrix is positive definite via attempted Cholesky decomposition

Usage

```
isposdef(amat)
```

Arguments

`amat`	dxd symmetric matrix

Value

True or False; True if matrix is positive definite

Examples

```
a1=matrix(c(1,.5,.5,1),2,2)
a2=matrix(c(1,1.5,1.5,1),2,2)
t1=try(chol(a1))
t2=try(chol(a2))
print(isposdef(a1))
print(isposdef(a2))
```

KLdiv *Kullback-Leibler divergence and sample size for two bivariate copula densities*

Description

Kullback-Leibler divergence and sample size for two bivariate copula densities

Usage

```
KLcopvsbvn(rho,dcop2,param2,copname2="bivcop",UB=7,iprint=F)
KLcopvscop(copname1="cop1",param1,dcop1,copname2="cop2",param2,dcop2,UB=7,iprint
KL12gl(par2,dcop2,par1,dcop1,parlb=0,gl)
KLoptgl(dcop1,dcop2,par1,par2,name1,name2,par2lb=0,gl,pcop1,pcop2,
   ccdf1a,ccdf1b,ccdf2a,ccdf2b,prlevel=0)
```

Arguments

dcop1	function for first bivariate copula density; form is dcop(u,v,param)
dcop2	function for second bivariate copula density
copname1	name of first bivariate copula density
copname2	name of second bivariate copula density
rho	parameter of dcop1 if it is bivariate normal
param1	parameter of dcop1
param2	parameter of dcop2
UB	limit to use for 2-dimensional integration with respect to bivariate normal density
iprint	print flag for intermediate results, FALSE by default
par1	copula parameter for family 1
par2	copula parameter for family 2
parlb	lower bound for copula parameter for family 2
par2lb	lower bound for copula parameter for family 2
gl	Gauss-Legendre quadrature object with $nodes and $weights
name1	name of first bivariate copula density
name2	name of second bivariate copula density
pcop1	function for first bivariate copula cdf; form is pcop(u,v,param)
pcop2	function for second bivariate copula cdf
ccdf1a	function for $C_{2\|1}$ for pcop1
ccdf1b	function for $C_{1\|2}$ for pcop1
ccdf2a	function for $C_{2\|1}$ for pcop2
ccdf2b	function for $C_{1\|2}$ for pcop2
prlevel	print.level for nlm

Value

For KLcopvsbvn and KLcopvscop, a vector with 5 elements: KLcop1true, KLcop2true, Jeffreys, sampsize1, sampsize2, where KLcop1true (KLcop2true) is the KL divergence when copula 1 (respectively copula 2) is the true model; Jeffreys=KLcop1true+KLcop2true is Jeffreys' divergence, sampsize1 (sampsize2) is the sample size needed to distinguish with the other copula family wth probability 0.95 when copula 1 (respectively copula 2) is the true model.

For KL12gl, a single value for divergence; this function is optimized by KLoptgl.

For KLoptgl, a list with $cpar2 for the parameter of copula2 leading to small KL divergence with copula1 with parameter param1, $depm1=c(be1,tau1,rhoS1,rhoN1) is a vector of beta,tau,rhoS,rhoN for copula1 with parameter param1, $depm2=c(be2,tau2,rhoS2,rhoN12 is a vector of beta,tau,rhoS,rhoN for copula2 with parameter cpar2.

See Also

pcop pcond deppar2taurhobetalambda

Examples

```
rho=bvn.b2cpar(0.5)
par.gum=gum.b2cpar(0.5)
par.gal=gal.b2cpar(0.5)
KLcopvsbvn(rho,dcop2=dgum,param2=par.gum,copname2="Gumbel",UB=6,iprint=TRUE)
KLcopvscop("Gumbel",par.gum,dgum,"Galambos",par.gal,dgal,UB=6,iprint=TRUE)
# parameter from Plackett family closed to bivariate Gaussian copula with
#   rhoS=0.5
cpar1=bvn.rhoS2cpar(.5)
cpar2=pla.rhoS2cpar(.5)  # 5.115661
gl25=gausslegendre(25)
kl=KLoptgl(dbvncop,dpla,cpar1,cpar2,"bvn","plackett",0,gl25,pbvncop,ppla,
   pcondbvncop,pcondbvncop,pcondpla,pcondpla)
print(kl$cpar2)  # 4.671857
```

---

kzrepmeas *Karim-Zeger data set*

Description

73 subjects, with 4 repeated measures in quarters 1 to 4; variables are id, time, age, sex, msmok, y. For the variables, y is the count response variable for the number of visits to the hospital, quarterly over a one-year period; id=identification code, time=quarter, age=age in months at the beginning of the study, sex=0 for male and 1 for female, msmok=maternal smoking status (1 for yes, 0 for no).

This data set is on page 342, Table 9.22, Problem 9.12 of "Statistical methods for the analysis of repeated measurements", by Charles S. Davis.

The original source is: "Karim MR and Zeger SL (1988). GEE: A SAS macro for longitudinal data analysis. Technical Report 674, Department of Biostatistics, The Johns Hopkins University, Baltimore."

Usage

```
data(kzrepmeas)
```

load2pcor *Operations of loading matrix of Gaussian factor analysis*

Description

Operations of loading matrix of Gaussian factor analysis

Usage

```
load2pcor(amat)
pcor2load(rhmat)
grotate2(amat,row,iprint=F) # apply one Givens rotation
grotate3(amat,row1=1,row2=2,iprint=F) # apply three Givens rotations
```

Arguments

amat	dxp matrix of loadings for load2pcor, dx2 for grotate2, dx3 matrix for grotate3
rhmat	dxp matrix for pcor2load, correlations with factor 1 in column 1, partial correlations with factor k given previous factors in column k
row	index of row to set to 0 in second column
row1	index of row to set to 0 in second and third columns
row2	index of row to set to 0 in third column
iprint	print flag for intermediate results

Value

For load2pcor, dxp matrix with correlations with factor 1 in the first column and partial correlations with factor k given previous factors in the kth column.

For grotate2 and grotate3, a rotated loading matrix with same dimension as inputted matrix.

For pcor2load, a matrix of loadings.

Examples

```
d=7
be1=c(.7,.6,.7,.6,.7,.6,.7)
be2=c(.4,.4,.4,.4,.3,.3,.3)
cpar1.gum=gum.b2cpar(be1)
cpar2.gum=gum.b2cpar(be2)
n=300
set.seed(123)
gumdat=sim2fact(n,cpar1.gum,cpar2.gum,qcondgum,qcondgum,"gum","gum")
zdat=nscore(gumdat,iopt=TRUE)
rmat=cor(zdat)
out2=factanal(covmat=rmat,factors=2)
amat2=matrix(c(out2$loadings),d,2)
amat2=grotate2(amat2,row=1)
print(amat2) # 0 in (1,2) position
rhmat=load2pcor(amat2)
amat=pcor2load(rhmat)
out3=factanal(covmat=rmat,factors=3)
amat3=matrix(c(out3$loadings),d,3)
```

```
amat3=grotate3(amat3,row1=1,row2=2)
print(amat3) # 0 in (1,2), (1,3) and (2,3) positions
load2pcor(amat3)
```

`loglikvector` *Vector of log-likelihoods for a model*

Description

Vector of log-likelihoods for a model for use with Vuong's procedure

Usage

```
vuongllkr(llkv1,llkv2)
vuong2llkr(llkv1,llkv2,dim1,dim2)
mdiscretellkv(param,uudat,mrectpr)  # multivariate discrete
emvndiscretellkv(param,zzdat)  # exchangeable discretized multivariate normal
rvinediscretellkv(param,uudat,A,pcopnames,iprint=FALSE) # discrete R-vine
ir1factpmf(param,dstruct,pcondcop)
ir2factpmf(param,dstruct,pcondcop1,pcondcop2)
strfactllkv(param,udat,strmodel,copname,nq,grsize=0,nu=0,ipdf=1)
mvtbifactllkv(param,tdata,grsize,df,full=TRUE)
mvttrifactllkv(param,tdata,grsize,sbgrsize,df,full=TRUE)
rvinellkv.trunc(parvec,udat,A,logdcopnames,pcondnames,np)
```

Arguments

`llkv1`	vectors of log-likelihoods for model 1, length n is same as that for the sample size
`llkv2`	vectors of log-likelihoods for model 2, length n is same as that for the sample size
`dim1`	parameter vector dimension for model 1
`dim2`	parameter vector dimension for model 2
`param`	parameter vector for the model
`parvec`	parameter vector for the model
`uudat`	dimension nx(2d) with corners of rectangle probabilities for each discrete vector observation on U(0,1) scale; uudat[,1:d]<uudat[(d+1):(2*d)]
`zzdat`	dimension nx(2d) with corners of rectangle probabilities for each discrete vector observation on N(0,1) scale; zzdat[,1:d]<zzdat[(d+1):(2*d)]
`mrectpr`	function for multivariate rectangle probability
`A`	dxd vine array with 1:d on diagonal
`pcopnames`	string vector with names of copula cdfs of length ntrunc, ntrunc=truncation level
`pcondcop`	pcond function 1-factor ordinal model
`pcondcop1`	pcond function first factor of 2-factor ordinal model
`pcondcop2`	pcond function second factor of 2-factor ordinal model
`dstruct`	structure with $dat for dataset, $cutp for cutpounts on (0,1) scale, and $quad for Gauss-Legendre object with quadrature points and nodes

udat	nxd matrix of uniform scores
strmodel	one of "1factor", "2factor", "bifactor", "nestedfactor"
copname	something like "frank", "bb1", "bb1frank", "bb1frk", "t", "tapprox"
nq	number of quadrature points
grsize	vector of group sizes for mgrp groups with sum(grsize)=d
sbgrsize	vector of subgroup sizes by partitioning grsize vector
nu	Student t df parameters of copname is "t"
ipdf	default to 1 to compute log-likelihood only with gradients
tdata	nxd matrix of Student t scores
df	df parameter for multivariate Student t
iprint	print flag for intermediate calculations
full	T for bi-factor and F for nested-factor structure for multivariate t
logdcopnames	string vector of names of logcopula densities for trees 1,...,ntrunc
pcondnames	string vector of names of cond cdfs for trees 1,...,ntrunc
np	dxd where np[ell,j] is size for parameter th[ell,j] for bivariate copula in tree ell, variables j and A[ell,j]

Value

vector of log-likelihood values at parameter estimate, one for each observation, for the llkv functions;

vector of discrete probabilities or likelihoods for ir1factpmf and ir2factpmf;

95 percent interval for the mean of llkv2-llkv1 for the vuongllkr and vuong2llkr function (the latter adjusts for the Schwarz/BIC correction); negative interval means that model 1 is better, positive interval means that model 2 is better, and interval that includes 0 implies models not significantly different.

See Also

bivcopnllk factorcopmle IRfactormle mdiscretenllk mvtfact rvinediscrete structcop

Examples

```
# example with discrete ordinal data
data(ltmconv)
d=ncol(sci)
n=nrow(sci)
ucutp=unifcuts(sci)
nq=21
gl = gausslegendre(nq)
# factor models
library(abind)
dstrsci=list(dat=sci,quad=gl,cutp=ucutp)
par1.gum=c(1.467402,1.071322,1.589827,2.476544,1.049539,1.191183,1.506965)
llkv1fgum=ir1factpmf(par1.gum,dstrsci,pcondgum)
llkv1fgum=log(llkv1fgum)
par2.gumt=c(1.3632887,1.5646628,1.1770374,1.3801555,1.5733537,
 1.7975962,1.2583718,
```

```
-0.3123371,0.4745622,-0.5367812,-0.6948827,0.5530323,
 0.3288671,-0.4507251)
dfdefault=2
llkv2fgumt=ir2factpmf(par2.gumt,dstrsci,pcondgum,pcondt)
llkv2fgumt=log(llkv2fgumt)
# truncated vine model
perm=c(6,4,3,1,5,7,2)
sciperm2=sci[,perm]
d=ncol(sciperm2)
n=nrow(sciperm2)
A2=vnum2array(d,320)
out=varray2M(A2)
M2=out$mxarray
ucuts2=unifcuts(sciperm2)
ucuts2=rbind(rep(0,d),ucuts2,rep(1,d))
# gumbel/t(5)
pbvtcop1=function(u,v,rh,df=dfdefault)
{ param=c(rh,df)
  u[u>=1]=.9999999; v[v>=1]=.9999999
  u[u<=0]=.0000001; v[v<=0]=.0000001
  xt=qt(u,df); yt=qt(v,df);
  pbvt(xt,yt,param)
}
pcopnames2=c("pgum","pbvtcop1")
dfdefault=5
par2.rvine=c(1.1317132,1.0337752,1.2786283,1.4919012,1.3498679,1.5146284,
 0.4852748,0.1611857,-0.1258033,0.3406340,0.2977515)
parmat2=matrix(0,d,d)
parmat2[1,2:d]=par2.rvine[1:6]
parmat2[2,3:d]=par2.rvine[7:11]
llkvrvine2t=rep(0,n)
for(i in 1:n)
{ llkvrvine2t[i]=rvinepmf.ordinal(parmat2,sciperm2[i,],A2,M2,pcopnames2,ucuts2)
}
llkvrvine2t=log(llkvrvine2t)
cmpa=vuongllkr(llkvrvine2t,llkv1fgum)
print(cmpa)
cmpb=vuongllkr(llkvrvine2t,llkv2fgumt)
print(cmpb)

# example with continuous data
bevec=c(.8,.7,.6,.5,.5)
cpar.frk=frk.b2cpar(bevec)
cpar.gum=gum.b2cpar(bevec)
set.seed(123)
udat=sim1fact(100,cpar.frk,qcondfrk,"frank")
out.frk=ml1fact(nq=21,cpar.frk,udat,dfrk,LB=-30,UB=30,prlevel=0,mxiter=50)
out.gum=ml1fact(nq=21,cpar.gum,udat,dgum,LB=1,UB=30,prlevel=0,mxiter=50)
llkvfrk=strfactllkv(out.frk$estimate,udat,"1factor","frank",nq=21,ipdf=1)
llkvgum=strfactllkv(out.gum$estimate,udat,"1factor","gumbel",nq=21,ipdf=1)
print(c(out.frk$min,-sum(llkvfrk)))
print(c(out.gum$min,-sum(llkvgum)))
cmp=vuongllkr(llkvgum,llkvfrk)
print(cmp) # second model is "true" model so interval should be positive,
```

`ltmconv` *Item response data sets from ltm R package*

Description

item response data sets converted from ltm R package: (a) science, (b) environment.

(a) sci: Science data set, size 392x7 (7 items, categories 0,1,2,3); "0=strongly disagree", "1=disagree to some extent", "2=agree to some extent" and "3=strongly agree". Sample size n=392. The items are:

Y1: Science and technology are making our lives healthier, easier and more comfortable;

Y2: Scientific and technological research cannot play an important role in protecting the environment and repairing it;

Y3: The application of science and new technology will make work more interesting;

Y4: Thanks to science and technology, there will be more opportunities for the future generations;

Y5: New technology does not depend on basic scientific research;

Y6: Scientific and technological research do not play an important role in industrial development;

Y7: The benefits of science are greater than any harmful effect it may have.

(b) env: Environment data set, size 291x6 (6 items, categories 0,1,2). "0=very concerned", "1=slightly concerned" and "2=not very concerned". Sample size n=291. The items are:

Y1: Lead from petrol;

Y2: River and sea pollution;

Y3: Transport and storage of radioactive waste;

Y4: Air pollution;

Y5: Transport and disposal of poisonous chemicals;

Y6: Nuclear Risks from nuclear power station.

Usage

```
data(ltmconv) # objects are sci and env
```

`makedeptable` *Make table of dependence measures for a 1-parameter bivariate copula family*

Description

Make table of dependence measures for a 1-parameter bivariate copula family

Usage

```
makedeptable(bvec,bfn,dcop,pcop,pcond12,pcond21,LBcpar=0,UBcpar=Inf,
   itaildep=F,lmfn,zero=0,zbd=6,iprint=F)
```

Arguments

bvec	vector of Blomqvist beta values in increasing order, first is 0, last is 1
bfn	function to get copula parameter cpar given Blomqvist beta
dcop	function for copula density c
pcop	function for copula cdf C
pcond12	function for copula conditional $C_{1\|2}(u\|v)$
pcond21	function for copula conditional $C_{2\|1}(v\|u)$
LBcpar	parameter for independence
UBcpar	parameter for comonotonicity
itaildep	T to compute tail dependence parameter lambda=lm
lmfn	function to compute lambda given copula parameter
zero	0 or something like 1.e-6, used with pcond12, pcond21
zbd	integration bound with respect to N(0,1) margins for rhoN
iprint	print flag for intermediate results, default=F

Value

table with column headings of "cpar","beta","tau","rhoS","rhoN","lambda"

See Also

deptab depmeas2cpar

Examples

```
bvec=seq(0,.9,.02)  # more grid points
bvec=c(bvec,.95)
bvec=c(bvec,1)
np=length(bvec)
bvec=bvec[-(np-1)]  # 0.95 can be a problem for some copula families
np=length(bvec)
frk.deptab=makedeptable(bvec,bfn=frk.b2cpar,pcop=pfrk,
  pcond12=pcondfrk,pcond21=pcondfrk,LBcpar=0,
  UBcpar=Inf,itaildep=FALSE,zero=0,zbd=7,iprint=TRUE)
tauinteg=frk.deptab[,3]
for(i in 2:(np-1))
{ frk.deptab[i,3]=frk.cpar2tau(frk.deptab[i,1]) }
print(frk.deptab)
cat("accuracy of 2-dimensional numerical integration\n")
print(abs(tauinteg-frk.deptab[,3]))
```

`mdiscretenllk` *negative log-likelihood of multivariate copula with discrete margins*

Description

negative log-likelihood of multivariate copula with discrete margins

Usage

```
mdiscretenllk(cpar,uudat,pmcop,LB,UB)
emvndiscretenllk(cpar,zzdat)  # exchangeable multivariate normal
```

Arguments

`cpar`	copula parameter for pmcop
`uudat`	n x (2d) matrix with uudat[,1:d] lower corner and uudat[(d+1):(2d)] upper corner of rectangle, after fitting univariate models and converting margin via cdf
`pmcop`	function for the cdf of the d-variate copula
`zzdat`	n x (2d) matrix with zzdat[,1:d] lower corner and zzdat[(d+1):(2d)] upper corner of rectangle, after fitting univariate models and converting margin to standard normal (for emvndiscretenllk which is positive exchangeable multivariate normal)
`LB`	lower bound vector for cpar
`UB`	upper bound vector for cpar

Value

negative log-likelihood

See Also

`discreteresponse`

Examples

```
# d=4 dimensional data with transformed univariate
uu=matrix(c(.1,.1,.1,.1, .5,.6,.7,.8, 0,0,0,0, .4,.5,.4,.3),2,8,byrow=TRUE)
zz=qnorm(uu)
zz[zz< -6]=-6
mdiscretenllk(2,uu,pmfrk,0,30)
emvndiscretenllk(0.3,zz)
```

mprobit *multivariate ordinal probit and approximation by vine distributions*

Description

multivariate ordinal probit and approximation by vine distributions

Usage

```
pmfmordprobit(zcuts,rmat,iprint=F,ifixseed=F) # multivariate ordinal probit pmf
dvineKLfn(parvec,ucuts,pr)  # KL divergence of multivariate probit and D-vine
rvineKLfn(parvec,ucuts,A,pr) # KL divergence of multivariate probit and R-vine
dvineKLss(parvec,ucuts,pr,iprint=F) # KL sample size of multiv probit and D-vine
rvineKLss(parvec,ucuts,A,pr,iprint=F) # KL sample size of multiv probit and R-vi
f90rvineKL(parvec,ucuts,A,M,pr) # KL divergence of multiv probit and R-vine
ARprobitvsDvine(ucuts,rmat,iprint=F,prlevel=1,mxiter=50,ifixseed=F) # best D-vin
mprobitvsRvine(ucuts,rmat,A,iprint=F,prlevel=1,mxiter=50,ifixseed=F) # best R-vi
f90mprobitvsRvine(ucuts,rmat,A,iprint=F,prlevel=1,mxiter=50,ifixseed=F)
```

Arguments

parvec	parameter vector of partial correlations with length d*(d-1)/2
rmat	dxd correlation matrix
ucuts	(ncateg+1)xd matrix of cut points for ordinal, e.g., computed from unifcuts via ucuts=unifcuts(y), ucuts=rbind(rep(0,d),ucuts,rep(1,d))
zcuts	cutpoints on N(0,1) for ordinal responses: (ncat+1)xd, where first row is a substitute for -Inf and last row is a substitute for +Inf
A	dxd vine array with 1:d on diagonal
M	dxd maximal array for vine array A
pr	vector outputted from pmfmordprobit()
iprint	print flag for intermediate results
mxiter	max iterations for nlm()
ifixseed	F by default, if T, then seed is fixed before each call to pmvnorm within the function
prlevel	print.level for nlm() minimization

Details

pmfmordprobit() requires library mvtnorm for the function pmvnorm for multivariate normal rectangle probabilities for dimensions 3 and higher. f90mprobitvsRvine is the faster version of mprobitvsRvine, with some computations in fortran90. f90mprobitvsRvine() and mprobitvsRvine() require library combinat.

Value

vector of pmf in ordinal categories in lexicographic order, for pmfmordprobit()

Kullback-Leibler divergence for dvineKLfn and rvineKLfn

KL divergence and sample size and probability vector from vine approximation, for dvineKLss and rvineKLsss

for ARprobitvsDvine, there are several components:

$mordprobitpr = probability vector from multivariate ordinal probit (length= ncateg^d)

$dvineparam = vector of parameters for D-vine approximation with bivariate Gaussian pair-copulas

$KLdiv = KL divergence of mult probit and D-vine approximation

$KLss = KL sample size

$vinepr = probability vector from D-vine approximation

for mprobitvsRvine and f90mprobitvsRvine, there are several components:

$mordprobitprmat = prob vectors from multivariate ordinal probit (ncateg^d x d!/2) for different permutations

$parmat = parameter of R-vine approximation for each permutation (C(d,2) x d!/2)

$vKLdiv = vector of KL divergences of multivariate probit and R-vine

$vKLss = vector of KL sample sizes

$rvineprmat= columns of probability vectors from R-vine approximation (ncateg^d x d!/2)

Examples

```
# D-vine
ncateg=3
ucuts3=matrix(c(.4,.5,.4, .7,.8,.6),ncateg-1,3,byrow=TRUE)
zcuts3=qnorm(ucuts3)
zcuts3=rbind(rep(-6,3),zcuts3,rep(6,3))
ucuts3=rbind(rep(0.00001,3),ucuts3,rep(0.99999,3))
rh=.6
rmat3=toeplitz(rh^(0:2))
library(mvtnorm)
pmf3=pmfmordprobit(zcuts3,rmat3,iprint=TRUE)
dvineappr=nlm(dvineKLfn,c(rh,rh,0),hessian=TRUE,iterlim=40,print.level=1,ucuts=ucuts3,pr=
dvineKLss(dvineappr$estimate,ucuts3,pmf3)
# multivariate AR probit versus discrete D-vine
ARprobitvsDvine(ucuts3,rmat3,iprint=FALSE,prlevel=1,mxiter=50)
# R-vine
d=4
A=Dvinearray(d)
out=varray2M(A); M=out$mxarray
ucuts4=matrix(c(.4,.5,.4,.3, .7,.8,.6,.6),2,4,byrow=TRUE)
ucuts4=rbind(rep(0.00001,d),ucuts4,rep(.99999,d))
parvec=c(.5,.5,.5,.1,.1,0)
pr=rep(1/81,81)
out=f90rvineKL(parvec,ucuts4,A,M,pr)
print(out)
rvineKLfn(parvec,ucuts4,A,pr)
# multivariate probit versus discrete R-vine
C3=Cvinearray(3)
ncateg=3
ucuts3=matrix(c(.4,.5,.4, .7,.8,.6),ncateg-1,3,byrow=TRUE)
```

```
ucuts3=rbind(rep(0.00001,3),ucuts3,rep(.99999,3))
rmat3=toeplitz(0.5^(0:2))
library(combinat)
out3=mprobitvsRvine(ucuts3,rmat3,C3,iprint=TRUE,prlevel=1)
out3b=f90mprobitvsRvine(ucuts3,rmat3,C3,iprint=TRUE,prlevel=1) # much faster
# these are not exactly the same output because
# pmfmordprobit uses pmvnorm() in library mvtnorm and
#  pmvnorm() has a quasi-random component
out3=mprobitvsRvine(ucuts3,rmat3,C3,iprint=TRUE,prlevel=1,ifixseed=TRUE)
out3b=f90mprobitvsRvine(ucuts3,rmat3,C3,iprint=TRUE,prlevel=1,ifixseed=TRUE)
# the above are the same if ifixseed=TRUE
```

`mvtfact` *multivariate t with common p-factor, bi-factor and tri-factor correlation structures*

Description

multivariate t with common p-factor, bi-factor and tri-factor correlation structures: negative log-likelihoods and gradients

Usage

```
mvtpfactnllk(rhvec,tdata,df)
mvtbifactnllk(rhvec,grsize,tdata,df)
mvttrifactnllk(rhvec,grsize,sbgrsize,tdata,df)
mvtpfact(tdata,start,pfact,df,prlevel=0,mxiter=100)
mvtbifact(tdata,start,grsize,df,prlevel=0,full=T,mxiter=100)
mvttrifact(tdata,start,grsize,sbgrsize,df,prlevel=0,full=T,mxiter=150)
```

Arguments

`rhvec`	vector of correlation/partial correlation parameters with latent variables, length is d*p for mvtpfactnllk and d*2 for mvtbifactnllk
`tdata`	nxd data set, univariate margins are Student t(df)
`pfact`	number of factors for mvtpfact
`grsize`	vector of group sizes for the bi-factor and tri-factor models, length mgrp for mgrp groups with sum(grsize)=d
`sbgrsize`	vector of subgroup sizes for the tri-factor model, length msbgrp for msbgrp groups with sum(sbgrsize)=d; sbgrsize must be consistent with grsize as groups are split into subgroups
`df`	degree of freedom parameter for multivariate Student t; df>300 to get multivariate Gaussian
`start`	starting point for numerical maximum likelihood, length is d*p for mvtpfact and d*2 for mvtbifact
`full`	T for bi-factor, F for nested-factor as special case
`prlevel`	print.level for nlm
`mxiter`	max number of iterations iterlim for nlm

Details

When parameters are converted to a dxp matrix (p=2 for bi-factor and p=3 for tri-factor), the correlations or partial correlations in any column are unique up to sign.

There is further non-uniqueness for the p-factor model with p>=2, as the loading matrix can be rotated. To get a parameter matrix with one 0 in the second column, two 0s in the third column etc., use pcor2load() to convert to a loading matrix, then something like grotate2() and grotate3() to rotate the loading matrix, then convert back to a partial correlation representation with load2pcor().

For bi-factor and tri-factor, there is non-uniquess in the second (or third) column if some group (subgroup) sizes are 1 or 2.

However, if the numerial optimization converges from different starting points, the final negative log-likelihood should be the same even if the point of convergence is not and the gradient is not close to a zero vector.

Value

list with negative log-likelihood $nllk and gradient vector $lgrad for mvtpfactnllk and mvtbifactnllk

mle object (output of nlm) for mvtpfact and mvtbifact

References

Krupskii P (2014). Structured Factor Copulas and Tail Inference. PhD thesis, University of British Columbia.

See Also

bifct factanal.bi factorcopmle structcop

Examples

```
data(euro07gf)
udat=euro07gf$uscore
d=ncol(udat)
st1=rep(0.4,d)
st2=rep(0.4,2*d)
for(df in c(5,15))
{ tdata=qt(udat,df)
  cat("\ndf=", df,"\n")
  cat("1-factor MVt\n")
  out1t=mvtpfact(tdata,st1,pfact=1,df=df,prlevel=1)
  cat("\n2-factor MVt\n")
  out2t=mvtpfact(tdata,st2,pfact=2,df=df,prlevel=1)
  st1=out1t$estimate
  st2=out2t$estimate
}
# non-uniqueness for 2-factor
st2=matrix(st2,ncol=2)
load2=pcor2load(st2)
load2rot=grotate2(load2,row=1)
st2b=c(rep(.7,d),rep(.2,d))
out2b=mvtpfact(tdata,st2b,pfact=2,df=15,prlevel=1)
load2b=pcor2load(matrix(out2b$estimate,ncol=2))
load2brot=grotate2(load2b,row=1)
print(max(abs(load2b-load2)))
```

```
print(max(abs(load2brot-load2rot)))
print(out2t$min-out2b$min)
# bi-factor and nested-factor
df=10
tdata=qt(udat,df)
grsize=c(4,3)
bif=mvtbifact(tdata, c(rep(.8,d),.2,.2,.9,.2,.2,.8,.2),grsize,df=df,
  prlevel=1,full=TRUE,mxiter=100)
nestf=mvtbifact(tdata, c(.9,.2,rep(.8,d)),grsize,df=df,
  prlevel=1,full=FALSE,mxiter=100)
# tri-factor, simulated example
grsize=c(6,6)
sbgrsize=c(3,3,3,3)
d=sum(grsize)
p=3
tripar=((d*p):1)/(d*p+1)
param1=tripar[1:d]
param2=tripar[(d+1):(2*d)]
param3=tripar[(2*d+1):(3*d)]
n=100
df=10
robj=trifct(grsize,sbgrsize,param1,param2,param3)
achol=chol(robj$fctmat)
set.seed(123)
z=matrix(rnorm(n*d),n,d)
z=z
udata=uscore(z)
tdata=qt(udata,df)
out=mvttrifactnllk(tripar,grsize,sbgrsize,tdata,df)
print(out)
ml=mvttrifact(tdata,start=tripar,grsize,sbgrsize,df,prlevel=1,mxiter=150)
st2=tripar; tripar[1:d]=.8
ml2=mvttrifact(tdata,start=st2,grsize,sbgrsize,df,prlevel=1,mxiter=150)
```

`negbinom` *Negative binomial regression for count data*

Description

Negative binomial (NB) regression for count data; 2 versions. NB(theta,xi) has pmf f(y;theta,xi)= [Gamma(theta+y) xi^y] / [Gamma(theta) y! (1+xi)^(theta+y)]

Usage

```
nb1nllk(param,y,xdat)
nb2nllk(param,y,xdat)
nbpmfcdf(ub,theta,p)
nb1pmfcdf(ub,param,x)
nb2pmfcdf(ub,param,x)
nb1cdf(y,param,x)
nb2cdf(y,param,x)
nb1pmf(y,param,x)
nb2pmf(y,param,x)
```

Arguments

param	parameter of NB model, length is 2+number of covariates; the parameters are: b0=intercept, bvec= vector regression coefficients (length(bvec)=length(x)=ncol(xdat), and finally xi or theta. For NB1, mu(x)= exp(b[0]+bvec^T x), xi=(overdispersion index minus one) is fixed, and theta(x)=mu(x)/xi. For NB2, mu(x)= exp(b[0]+bvec^T x), theta=convolution parameter is fixed and xi(x)=mu(x)/theta.
theta	convolution parameter
p	probability parameter between 0 and 1
xdat	matrix for nb1nllk and nb2nllk
x	vector for nb1pmfcdf, nb2pmfcdf, nb1cdf, nb2cdf, nb1pmf, nb2pmf
y	vector for nb1nllk and nb2nllk (with length(y)=nrow(xdat)); non-negative integer for the other functions
ub	upper bound integer for which pmf and cdf are computed

Value

negative log-likelihood for nb1nllk and nb2nllk; matrix with columns (0:ub,pmf,cdf) for nbpmfcdf, nb1pmfcdf and nb2pmfcdf, computed in an efficient way (parameters assumed to be such that most probability is on small counts); cdf for nb1cdf and nb2cdf; pmf for nb1pmf and nb2pmf.

See Also

gpoisson

Examples

```
y= c(
 2, 1, 1, 0,35, 9, 0, 1, 4, 0, 0, 1, 4, 0, 0, 8, 7, 2, 0, 7, 0, 0, 3, 4, 0,
 4, 1, 3, 0, 6, 1, 0, 2, 8, 0,12, 0, 4, 2, 1, 3, 0, 9, 0, 0, 0, 2, 0, 8, 1,
 2, 4, 2, 0, 0, 2, 1, 3, 2, 1, 3, 4, 4, 5, 0, 4, 0, 2, 0,28, 1,24, 1, 0,10,
 3, 3, 0, 0, 7, 2, 4, 6, 4,13, 5, 8, 0, 1, 6, 0,24, 9, 0,10, 0, 0, 8, 5, 3,
16, 0, 4, 1, 1, 4,12, 4, 3, 5, 0, 2, 1, 5, 3, 0, 0, 6, 4, 2, 0, 2, 0,15, 3,
 0, 2, 3, 4, 5, 0, 3, 0, 0, 6, 0, 0,15, 0, 0, 0, 1, 3, 0, 1, 0, 4, 2,10, 4,
 1, 0, 0, 0, 5, 0, 0, 2, 0, 4, 0, 0, 2,25, 0, 0,13, 0, 0,21, 3, 0, 0, 0, 2,
 2, 0, 4,13, 2, 9, 9, 2, 0, 1, 2, 2, 8, 6, 0, 4, 1, 2, 0, 0, 0, 0, 0, 0, 2,
 2, 0, 3, 1, 1, 7, 3, 0, 2, 2, 1, 3, 2, 2, 1, 3, 3, 0, 0, 0, 2, 0, 0, 0, 0,
 1, 2, 2, 0, 0, 9, 0, 0, 1, 1, 0, 2,10, 0,17, 2, 0,14, 0, 5, 9, 2, 0, 6, 3,
 3, 1, 0,11, 4, 9, 0, 1, 0, 0,12, 4, 0, 1,21, 0, 3, 2, 0, 1, 0, 1, 3, 8,10,
19, 0, 2, 7, 1, 0, 2, 0, 4, 0, 6, 4, 7, 1, 0, 1, 3, 4, 0, 4)
hsat=c(
 8, 7, 3,10, 6, 5, 8, 9, 9, 8,10, 8, 6, 7,10, 8, 5, 8, 8, 6, 8, 8, 8, 9,10,
 7, 9,10, 8, 6, 6, 9, 7, 5,10, 4, 8, 4, 5, 5, 7, 6, 7,10, 9, 9, 5, 7, 4, 7,
 6, 6, 7, 5,10, 9,10, 7, 8, 6, 5, 5, 0, 5, 7, 3, 8, 8, 7, 5, 5, 0, 7, 6, 3,
10, 7, 7,10, 5, 5, 4, 2, 7, 6, 2, 5,10, 7, 8, 5, 5, 5,10, 3, 9, 6, 8,10,10,
 4, 7, 2, 8, 9, 0, 0, 5, 8, 3, 7, 6,10, 4, 5, 7, 6, 7, 3, 4,10, 4, 8, 8, 3,
 9, 5,10, 9, 5,10,10, 8,10, 5,10, 6, 5, 9, 8,10, 7, 8, 9, 7, 8, 4, 8, 3, 5,
 5, 7,10, 8, 1, 3, 3, 8,10, 3, 5, 5, 7, 5,10, 8, 5, 8, 5, 0, 6, 8, 2, 5, 6,
 7,10, 5, 0, 5, 2, 0, 3,10, 7, 4, 6, 9, 2, 8, 5, 9, 7, 5,10, 8, 8, 7, 7, 7,
10,10, 2, 5, 7, 5, 9, 6, 7, 6, 9, 9, 6, 8,10, 7, 8, 8,10,10, 5,10, 5, 8,10,
 8, 7,10, 9,10, 4, 6, 9, 5, 9, 9, 6, 8, 8, 2, 5, 8, 3, 7, 0, 8, 8,10, 5, 7,
 6, 7,10, 5, 5, 1, 5, 6, 4,10, 5, 5, 5, 7, 2, 8, 5,10,10,10,10, 6, 6, 6, 6,
 7, 8, 8,10,10, 8, 7, 8, 3, 8, 8, 8, 6, 3, 7,10,10, 2, 9, 2)
fit1=nlm(ieenllk,p=c(2.5,-.2,4),hessian=TRUE,print.level=1,upmf=nb1pmf,
```

```
    xdat=hsat,ydat=y,LB=c(-1,-2,0),UB=c(10,10,10))
fit2=nlm(ieenllk,p=c(2.5,-.2,0.8),hessian=TRUE,print.level=1,upmf=nb2pmf,
    xdat=hsat,ydat=y,LB=c(-1,-2,0),UB=c(10,10,10))
```

`nscore` *Transform to normal scores*

Description

Transform each variable in a data matrix to normal scores and finding the adjustment so that mean of squares of scores is 1.

Usage

```
nscore(data,iopt=F)
nscoreOpta(n,mxiter=20, eps=1.e-4,iprint=F)
```

Arguments

`data`	data matrix or data frame
`iopt`	iopt=T to use adjustment 'a' in nscoreOpta that makes normal scores have mean of 0 and variance of 1; default is iopt=F in which case a=-0.5 is used
`n`	sample size
`mxiter`	maximum number of iterations
`eps`	tolerance for stopping
`iprint`	print flag for iterations

Value

data matrix with same number of columns as input for nscore

number near -0.5 for nscoreOpta

See Also

`uscore`

Examples

```
set.seed(123)
x=matrix(rnorm(40),20,2)
z=nscore(x)
nscoreOpta(100) # -0.5766102
nscoreOpta(1000) # -0.5522775
```

`ordinal` *multivariate ordinal response*

Description

multivariate ordinal response with no covariates: some utility functions

Usage

```
d2v(d,ncat,ii,izero=F)  # decimal to category vector
v2d(jj,ncat)  # vector to decimal
d2b(d,ii)  # decimal to binary vector
unifcuts(odat)
ordinal2fr(odat,ncat)
bprobitnllk(rho,zcuts,bfr,jj1,jj2)
bprobitwPrednllk(rho,zzdat,jj1,jj2)
polychoric.bivtab(bivtab,iprint=F,prlevel=0)
polychoric0(odat,iprint=F,prlevel=0)   # number of categories can vary
polychoric(odatfr,zcuts,iprint=F,prlevel=0) # same categories for all variables
polychoric.wPred(zzdat,iprint=F,prlevel=0)
```

Arguments

`d`	dimension d of ordinal response vector
`ncat`	#categories (assumed labeled 0,1,...,(ncat-1) or 1,...,ncat)
`rho`	latent correlation parameter in (-1,1)
`ii`	non-negative integer in 0 to ncat^2-1
`jj`	d-vector, each element in 0,1,...,(ncat-1)
`izero`	if T, categories start at 0, otherwise 1
`odat`	nxd matrix of ordinal responses in 0,...,(ncat-1) or 1,...,ncat
`odatfr`	nx(d+1) matrix: d columns of ordinal responses and final column with frequency of each distinct observed d-vector
`zcuts`	cutpoints on N(0,1) for ordinal responses: (ncat+1)xd, where first row is a substitute for -Inf and last row is a substitute for +Inf
`bivtab`	bivariate table of counts for 2 ordinal variables
`zzdat`	nx(2d) matrix with corners of rectangle for each vector observation in N(0,1) scale; for ordinal data, ordprobit.univar and mord2uu can been used to get zzdat
`bfr`	vector of bivariate frequencies
`jj1`	index of first variable
`jj2`	index of second variable
`iprint`	flag for printing of intermediate results, including bivariate tables for observed versus expected assuming discretized bivariate Gaussian
`prlevel`	print.level for nlm for numerical optimization

Details

The intermediate bivariate tables from iprint=T are currently not saved in the output data structure. bprobitnllk is optimized by polychoric(). Currently prdinal2fr() and polychoric() assume that all ordinal variables have the same number of categories but the code could be generalized. bprobitwPrednllk is optimized by polychoric.wPred().

polychoric0() might fail if some ordinal category has zero counts for one or more of the ordinal variables; in this case, use polychoric() with some preprocessing should be better.

Value

d-vector for d2v(); decimal for v2d() (as way to index ordinal vectors;

bivary d-vector for d2b();

odatfr is output of ordinal2fr() (ordinal to frequencies);

a (ncat-1)xd matrix of cutpoints on the uniform(0,1) scale for unifcuts(), where ncat is number of ordinal categories and d is number of variables;

a polychoric correlation and SE for polychoric.bivtab();

for polychoric() and polychoric0(), $polych is a polychoric correlation matrix based on two-stage estimate; $iposdef is an indicator if the 2-stage correlation matrix estimate is positive definite;

for polychoric(), also $zcuts is the matrix of cutpoints on N(0,1) scale.

for polychoric.wPred(), $polych is a polychoric correlation matrix based on two-stage estimate; $iposdef is an indicator if the 2-stage correlation matrix estimate is positive definite

See Also

ordinal.bivcop ordprobit.univar

Examples

```
d2v(3,3,0) # 1 1 1 since ii=0 -> 111, ii=1 -> 112, ii=2 -> 113 etc
d2v(3,3,0,izero=TRUE) # 0 0 0 since ii=0 -> 000, ii=1 -> 001, ii=2 -> 002 etc
d2v(3,3,10) # 2 1 2 since ii=8 -> 133, next in lexicographic order are 211,212
d2v(3,2,6)  # 2 2 1  (1s and 2s)
d2v(3,2,6,izero=TRUE) # 1 1 0 (0s and 1s)
d2b(3,6)  # 1 1 0  (0s and 1s)
v2d(c(1,1,0,1),2)  # 1*2^3+ 1*2^2+ 1= 13
v2d(c(1,1,0,1),3)  # 1*3^3+ 1*3^2+ 1= 37
# examples for unifcuts and ordinal2fr
set.seed(12345)
x=rnorm(1000)
y=0.5*x+sqrt(.75)*rnorm(1000)
x=cut(x,c(-Inf,-.75,1.3,Inf))
y=cut(y,c(-Inf,-.5,1.5,Inf))
x=as.numeric(x)
y=as.numeric(y)
odat0=cbind(x,y)-1
odat1=cbind(x,y)
print(unifcuts(odat0))
print(unifcuts(odat1)) # same as above
odat0fr=ordinal2fr(odat0,3)
odat1fr=ordinal2fr(odat1,3) # same as above
# example for polychoric
data(ltmconv)
```

```
nitem=ncol(sci) # sci is 392x7
nc=4
odatfr=ordinal2fr(sci,nc) # 298x8
d=ncol(odatfr)-1 # 7
ucuts=unifcuts(sci)
zcuts=qnorm(ucuts)
zcuts=rbind(rep(-6,d),zcuts,rep(6,d))
cat("\ncut points on N(0,1) scale\n")
print(zcuts)
polyr=polychoric(odatfr,zcuts,iprint=TRUE,prlevel=0)
polyr0=polychoric0(sci,iprint=TRUE) # without preprocessing
print(polyr0$polych-polyr$polych)
# example with bivariate table that is not square
btab=matrix(c(20,14,3,12,15,10,3,14,21,4,6,20),3,4)
polychoric.bivtab(btab,iprint=TRUE)
# example for polychoric.wPred
data(ordinalex)
xvec=c(t(ordinalex$xx))
yvec=c(t(ordinalex$yy))
ord.univar=ordprobit.univar(xvec,yvec,iprint=TRUE)
print(ord.univar)
ordtr=mord2uu(xvec,yvec,4,ord.univar$cutpts,ord.univar$beta)
polyr2=polychoric.wPred(ordtr$zzdat,iprint=TRUE)
```

`ordinal.bivcop` *Negative log-likelihood for bivariate marginal copula model for discrete variables*

Description

Negative log-likelihood for bivariate marginal copula model for discrete variables; this function is a counterpart of polychoric() when the bivariate Gaussian copula is replaced by another bivariate copula.

Usage

```
ordinal.bivcop(odatfr,ucuts,pcop,cparstart,LB=0,UB=10,iprint=F,prlevel=0)
bivcopOrdinalnllk(cpar,ucuts,bfr,jj1,jj2,pcop,LB=0,UB=10)
```

Arguments

`odatfr`	nrecx(d+1) matrix: d columns of ordinal responses and final column with frequency of each distinct observed d-vector (it could be just a vector of 1s)
`ucuts`	cutpoints in U(0,1) scale as (ncateg+1)xd matrix, obtained via unifcuts; so ncateg=nrow(ucuts)-1 and d=ncol(ucuts)
`pcop`	function with pair-copula cdf
`cpar`	copula parameter for pcop
`cparstart`	vector of starting points, dimension is d*(d-1)/2 times the dimension of the parameter for pcop
`bfr`	vector of bivariate frequencies
`jj1`	index of first variable when enumerating through pairs

jj2	index of second variable
LB	lower bound on parameter of pcop
UB	upper bound on parameter of pcop
iprint	flag for printing of intermediate results
prlevel	print.level for nlm for numerical optimization

Value

List with

$nllkvec = vector of length d*(d-1)/2 with negative log-likelihoods at the bivariate MLEs for each pair

$cparvec = vector of length d*(d-1)/2 times the dimension of the pcop parameter, with the bivariate MLEs for each pair

$summary = ncateg x (2*ncateg) x d*(d-1)/2 array with observed and expected bivariate marginal counts.

Compare output of function polychoric() with the bivariate normal/Gaussian copula as a latent model.

See Also

bivcopnllk ordinal

Examples

```
# convert bivariate t with fixed degree of freedom to use with above
pbvtcop5=function(u,v,rho) { pbvtcop(u,v,c(rho,5)) }
pbvtcop10=function(u,v,rho) { pbvtcop(u,v,c(rho,10)) }
data(ltmconv) # to use data set 'env'
d=3
nc=4  # ncateg
ucuts=unifcuts(env[,1:d])
bd=pnorm(-6)
ucuts=rbind(rep(bd,d),ucuts,rep(1-bd,d))
zcuts=qnorm(ucuts)
odatfr=ordinal2fr(env[,1:d],nc)
polyr=polychoric(odatfr,zcuts,iprint=FALSE,prlevel=0)
rhst=cormat2vec(polyr$polych)
rhstgal=rhst
cparst=depmeas2cpar(rhstgal,"rhoN","galambos")
cat("\nGalambos\n")
outgal=ordinal.bivcop(odatfr,ucuts,pcop=pgal,cparstart=cparst,iprint=TRUE,
   prlevel=0,LB=0,UB=10)
cat("\nt(5)\n")
outbvt5=ordinal.bivcop(odatfr,ucuts,pcop=pbvtcop5,cparstart=rhst,iprint=TRUE,
   prlevel=0,LB=-1,UB=1)
cat("\nt(10)\n")
outbvt10=ordinal.bivcop(odatfr,ucuts,pcop=pbvtcop10,cparstart=rhst,iprint=TRUE,
   prlevel=0,LB=-1,UB=1)
cat("\nBB1\n")
taust=bvn.cpar2tau(rhst)
dd=d*(d-1)/2
cparst=NULL
for(ii in 1:dd)
```

```
{ cpar=bb1.tau2eqlm(taust[ii])
  cparst=c(cparst,cpar[1:2])
}
outbb1=ordinal.bivcop(odatfr,ucuts,pcop=pbb1,cparstart=cparst,iprint=TRUE,
   prlevel=0,LB=rep(c(0,1),dd),UB=5)
cat("\nnllk comparison\n")
print(cbind(outgal$nllkvec,outbvt5$nllkvec,outbvt10$nllkvec,outbb1$nllkvec))
```

`ordinalex` *Multivariate ordinal data set*

Description

A simulated data set yy (200x4) with a repeated measures ordinal response with 4 measurements and 3 categories, a covariate matrix xx (200x4) with values in (-1,1). The values used for the simulation are: b0cut[1]=-0.5, b0cut[2]=0.5, b1=0.4, latent AR(1) correlation=0.5

Usage

```
data(ordinalex) # components b0cut, b1, d, ncateg, ncl, rmat, xx, yy
```

Format

The following are components.

`b0cut` cutpoints for the intercept b0

`b1` slope b1

`d` dimension of repeated measures

`ncateg` number of ordinal categories

`ncl` number of clusters

`rmat` latent correlation matrix

`xx` 200x4 covariate matrix with values in (-1,1)

`yy` 200x4 response variable matrix with values in 1,2,3

`ordprobit.univar` *Maximum likelihood for ordinal probit model*

Description

Maximum likelihood for ordinal probit: Newton-Raphson minimization of negative log-likelihood, and conversion to uniform/normal scales for fitting copula model in case of repeated measures with fixed cluster size

Usage

```
ordprobit.univar(x,y,iprint=F,mxiter=20,toler=1.e-6)
mord2uu(xmat,yvec,nrep,b0cut,bvec) # multivariate ordinal to (0,1) vector
```

Arguments

x	vector or matrix of explanatory variables. Each row corresponds to an observation and each column to a variable. The number of rows of x should equal the number of data values in y, and there should be fewer columns than rows. Missing values are not allowed.
y	numeric vector containing the ordinal response. The values must be in the range 1,2,...,ncateg or 0,1,...(ncateg-1), where ncateg is the number of categories. Missing values are not allowed.
iprint	print flag for the iterations for numerical maximum likelihood, default is FALSE
mxiter	maximum number of Newton-Raphson iterations
toler	tolerance for convergence in Newton-Raphson iterations
xmat	vector or matrix of explanatory variables; like above x
yvec	similar to above y
nrep	number of repeated measures or cluster size for each subject/unit
b0cut	vector of cutpoints
bvec	vector of regression coefficients

Details

If ordprobit for repeated measures ordinal probit fails to converge from the simple starting point in that function, this function ordprobit.univar should provide a better starting point. It is also equivalent to ordprobit with an identity latent correlation matrix.

The ordinal probit model is similar to the ordinal logit model (proportion odds logistic regression : polr in library MASS), The parameter estimate of ordinal logit are roughly 1.8 to 2 times those of ordinal probit (the signs of the parameters in polr may be different, as this function may be using a different orientation for the latent variable).

Value

For ordprobit.univar(), list of MLE of parameters and their associated standard errors, in the order cutpt1,...,cutpt(number of categ-1),b1,...b(number of covariates). $negloglik for value of negative log-likelihood, evaluated at MLE; $cutpts for MLE of ordered cutpoint parameters; $beta for MLE of regression parameters; $cov for estimated covariance matrix of the parameters.

For mord2uu, a list with components $uudat for transform of cdf to U(0,1), $zzdat for transform of cdf to N(0,1).

References

Anderson JA and Pemberton JD (1985). The grouped continuous model for multivariate ordered categorical variables and covariate adjustment. Biometrics, 41, 875-885.

See Also

ordinal

Examples

```
data(ordinalex)
xvec=c(t(ordinalex$xx))
yvec=c(t(ordinalex$yy))
ord.univar=ordprobit.univar(xvec,yvec,iprint=TRUE)
print(ord.univar)
ord.univar2=ordprobit.univar(xvec,yvec-1,iprint=TRUE)
print(ord.univar2) # same as ord.univar
ordtr=mord2uu(xvec,yvec,4,ord.univar$cutpts,ord.univar$beta)
ordtr2=mord2uu(xvec,yvec-1,4,ord.univar$cutpts,ord.univar$beta) #same
max(abs(ordtr$uudat-ordtr2$uudat))
```

---

partialcor *partial correlations from a correlation matrix*

Description

partial correlations from a correlation matrix, (all possible) or (single)

Usage

```
allpcor(rr)
partcor(S,given,j,k)
```

Arguments

rr	dxd correlation matrix
S	dxd covariance or correlation matrix
given	the indices of the "given" or conditioning variables
j	index of first conditioned variable
k	index of second conditioned variable

Value

pc	single partial correlation from partcor of variables j,k given indices in 'given'
pcobj	from allpcor, list with partial correlations in 2 forms: a 3dim array named $pc3.array and a 2dim array named $pc2.array, also $mnmx for min/max partial correlation by conditioning set size; for pc3.array: third dimension comes from conditioning on sets in the order 1 2 12 3 13 23 123 4 14 24 124 34 134 234 1234 5 15 25 125 35 135 235 1235 45 145 245 1245 345, ... (third dimension has length 2^d-4)

Examples

```
d=5
rr=toeplitz(c(1,.5,.25,.125,.05))
pcobj=allpcor(rr)
print(pcobj$pc3.array[,,1])
partcor(rr,c(1),3,4)
partcor(rr,c(1),3,5)
print(pcobj$pc3.array[,,3])
partcor(rr,c(1,2),3,4)
partcor(rr,c(1,2),3,5)
```

pbnorm *Bivariate normal and Student cdfs with vectorized inputs*

Description

Bivariate normal and Student cdfs with vectorized inputs

Usage

```
pbnorm(z1,z2,rho,icheck=F)
pbvt(z1,z2,param,icheck=F)
```

Arguments

z1	scalar or vector of reals
z2	scalar or vector of reals
rho	scalar or vector parameter in (-1,1); vectors cannot have different lengths if larger than 1, each of z1,z2,rho either has length 1 or a constant n greater than 1
param	vector of length 2, or matrix with 2 columns; vectors and number of rows of matrix cannot be different if larger than 1; for param, first column is rho, second column is df.
icheck	T if checks are made for proper inputs, default of F

Details

Donnelly's code can be inaccurate in the tail when the tail probability is 2.e-9 or less (it sometimes returns 0). In the case the exchmvn code is used with dimension 2. Alternatively a user can use vectorized function pbivnorm() in the library pbivnorm, and write a function pbvncop based on it.

Value

cdf value(s)

References

Donnelly TG (1973). Algorithm 462: bivariate normal distribution, Communications of the Association for Computing Machinery, 16, 638;

pbvt is taken from the source of the mvtnorm R package, with a different interface to R; the degree of freedom parameter for the multivariate t cdf is integer value in that source. One could use linear interpolation for non-integer-valued degree of freedom parameter.

See Also

exchmvn

Examples

```
cat("\n pbnorm rho changing\n")
z1=.3; z2=.4; rho=seq(-1,1,.1)
out1=pbnorm(z1,z2,rho)
print(cbind(rho,out1))

cat("\n pbnorm matrix inputs for z1, z2\n")
rho=.4
z1=c(-.5,.5,10.)
z2=c(-.4,.6,10.)
z1=matrix(z1,3,3)
z2=matrix(z2,3,3,byrow=TRUE)
out3=pbnorm(z1,z2,rho)
print(out3)
cdf2=rbind(rep(0,3),out3)
cdf2=cbind(rep(0,4),cdf2)
pmf=apply(cdf2,2,diff)
pmf2=apply(t(pmf),2,diff)
pmf2=t(pmf2)  # rectangle probabilities
print(pmf2)

cat("\n pbvt rho changing\n")
z1=.3; z2=.4; rho=seq(-.9,.9,.1); nu=2
param=cbind(rho,rep(nu,length(rho)))
out1=pbvt(z1,z2,param)
print(cbind(rho,out1))
cat("\n pbvt z1 changing\n")
z1=seq(-2,2,.4)
z2=.4; rho=.5; nu=2
out2=pbvt(z1,z2,c(rho,nu))
print(cbind(z1,out2))
```

---

pcinterpolate *Interpolate a monotone function (via piecewise cubic Hermite)*

Description

Interpolate a monotone function from a two-column table

Usage

```
pcderiv(x,fn)
pcinterpolate(x,fn,deriv,xnew)
```

Arguments

x	vector of x values
fn	function values corresponding to each value in x
deriv	vector of (estimated) derivative values assuming monotonicity, output of pcderiv and input to pcinterpolate
xnew	vector of new x values to interpolate values of the function.

Details

The monotone piecewise cubic interpolation algorithm is from the references given below.

Value

deriv	for pcderiv, estimated derivatives at x values
y	for pcinterpolate, length(xnew) x 2 matrix of interpolated fn values and derivatives

References

Fritsch FN and Carlson RE (1980), Monotone piecewise cubic interpolation. Siam J Numerical Analysis, 17, 238-246.

Kahaner D, Moler CB and Nash S (1989). Numerical Methods and Software Prentice Hall.

See Also

makedeptable

Examples

```
n=21
x=seq(0,pi/2,length=n)
fn=sin(x)
der=pcderiv(x,fn)
print(cbind(der,cos(x)))
xnew=seq(.05,1.,.05)
out=pcinterpolate(x,fn,der,xnew)
fval=sin(xnew)
print(cbind(out[,1],fval,abs(out[,1]-fval)))
cat("max err in deriv : ", max(abs(der-cos(x))), "\n")
cat("max err in interp : ", max(abs(out[,1]-fval)), "\n")
```

---

pcond — *Bivariate copula conditional cdfs and quantile functions*

Description

Bivariate copula conditional cdfs and quantile functions

Usage

```
pcond(v,u,cpar)
qcond(p,u,cpar)
```

Arguments

v	conditioned value in interval 0,1; could be a vector
u	conditioning value in interval 0,1; could be a vector
p	quantile in interval 0,1; could be a vector
cpar	copula parameter: could be scalar or vector depending on the copula family, could be a matrix with m columns if copula family has m parameters.

Details

Choices appending 'pcond' and 'qcond' are pla, frk, mtcj, mtcjr (reflected mtcj), joe, gum, gumr, gal, hr, fgm, bMO21, bb1, bb2, bb3, bb4, bb4r, bb5, bb6, bb7, bb7r, bb8, bb9, bb10, tev, ipsA, ipsAr, imitlefA, imitlefAr, etc. Use pcondbvncop and pcondbvtcop (or pcondt) for the conditional cdfs of the bivariate normal and t copulas.

See help page for pcop for the abbreviations of the copula names.

Value

conditional cdf value(s) or inverse conditional cdf value(s)

References

Joe H (1997). Multivariate Models and Dependence Concepts. Chapman & Hall.

See Also

pcond cparbound

Examples

```
u=seq(.1,.9,.2)
v=u
pcondpla(u,v,2)
pcondfrk(u,v,2)
pcondmtcj(u,v,2)
pcondbb1(u,v,c(.5,1.2))
qcondpla(u,v,2)
qcondfrk(u,v,2)
qcondmtcj(u,v,2)
qcondbb1(u,v,c(.5,1.2))
qcondjoe(u,v,2)
qcondgum(u,v,2)
qcondgal(u,v,2)
qcondhr(u,v,2)
```

---

pcop *Bivariate copula cdfs and densities*

Description

Bivariate copula cdfs and densities, for parametric families Log of copula densities.

Usage

```
pcop(u,v,cpar)
dcop(u,v,cpar)
logdcop(u,v,cpar)
```

Arguments

u	value in interval 0,1; could be a vector
v	value in interval 0,1; could be a vector
cpar	copula parameter: could be scalar or vector depending on the copula family, could be a matrix with m columns if copula family has m parameters.

Details

Choices are 'cop' in pcop and dcop are bvncop, bvtcop, pla, frk, mtcj, mtcjr (reflected mtcj), joe, gum, gumr, gal, hr, fgm, bMO, tev, bb1, bb1r, bb2, bb3, bb4, bb4r, bb5, bb6, bb7, bb7r, bb8, bb9, bb10, ipsA, ipsAr, imitlefA, imitlefAr, etc. Use dbvncop and dbvtcop for the bivariate normal and t copula densities. Note that pbvtcop assumes that the degree of freedom parameter is a positive integer.

The bounds for the copula parameter(s) are in the source R files, or can get obtained from the function cparbound(). The copula names are abbreviations for:

bvn = bivariate normal or Gaussian

bvt = bivariate t

pla = Plackett

frk = Frank

mtcj = Mardia-Takahasi-Clayton-Cook-Johnson

joe = Joe/B5

gum = Gumbel

gal = Galambos

hr = Huesler-Reiss

fgm = Farlie-Gumbel-Morgenstern

bMO = bivariate Marshall-Olkin (cdf only, it has not absolutely continuous)

basymgum1 = bivariate asymmetric Gumber with one skew parameter (cdf only)

tev = t-EV = extreme value limit of bivariate t

bb1 = BB1 etc

ipsA = Archimedean based on integrated positive stable LT

imitlefA = Archimedean based on integrated Mittag-Leffler LT

Choices are 'cop' in logdcop are pla, frk, mtcj, mtcjr, joe, gum, gal, hr, fgm, bvncop, bvtcop, bb1, bb1r, bb7, ipsA, ipsAr. These are included if it is more efficient to code logdcop directly, Otherwise create your own function from log of the appropriate dcop function. Another possibility is to write dcop functions with log=T option.

Value

cdf or pdf or log pdf value(s)

References

Joe H (1997). Multivariate Models and Dependence Concepts. Chapman & Hall.

See Also

pcond cparbound

Examples

```
u=seq(.1,.9,.1)
v=u
ppla(u,v,2)
pfrk(u,v,2)
pmtcj(u,v,2)
pbb1(u,v,c(.5,1.2))
dpla(u,v,2)
logdpla(u,v,2)
log(dpla(u,v,2))
```

`pdhessmin` *Minimization with modified Newton-Raphson and positive definite Hessian*

Description

pdhessmin: Minimization with modified Newton-Raphson and positive definite Hessian

pdhessminb: Minimization with modified Newton-Raphson and positive definite Hessian, with some parameters fixed (at bounds).

Usage

```
pdhessmin(param,objfn,dstruct,LB,UB,mxiter=30,eps=1.e-6,bdd=5,iprint=F)
pdhessminb(param,objfn,ifixed,dstruct,LB,UB,mxiter=30,eps=1.e-6,bdd=5,iprint=F)
```

Arguments

`param`	starting point for minimization of function objfn()
`objfn`	objective function of form objfn(param,dstruct,iprint=F); use iprint=T to print out extra information for debugging your function. objfn returns a list with fnval=functionvalue, grad=gradient, hess=hessian; that is, objfn computes the first and second order derivatives of objfn().
`dstruct`	data structure with data sets and other variables/controls to be passed and used by objfn()
`ifixed`	logical vector of same length as param, ifixed[i]=TRUE iff param[i] is fixed at the given value
`LB`	lower bound of components of param, usually of length(param), could also be a scalar for a common lower bound
`UB`	upper bound of components of param, usually of length(param), could also be a scalar for a common upper bound
`mxiter`	maximum number of Newton-Raphson iterations
`eps`	tolerance for Newton-Raphson iterations, stop when two consecutive iterations with eps in absolute value
`bdd`	bound on difference of 2 consecutive iterations, default 5
`iprint`	print flag for intermediate output for each iteration of the Newton-Raphson method

Details

The algorithm is due to P Krupskii.

Value

parmin	parameter value at point of minimum
fnval	function value at the minimum
invh	inverse Hessian at the minimum, estmated covariance matrix at MLE if objfn is negative log-likelihood
iconv	1 for convergence, 0 for not
iposdef	1 for positive definite Hessian at last iteration, 0 for not

Examples

```
data(euro07gf)
udat=euro07gf$uscore
n=nrow(udat)
d=ncol(udat)
np=2*d
stfrk2=rep(3,np);
LB.frk2=rep(-60,np);  UB.frk2=rep(60,np);
gl=gausslegendre(15)
dstructfrk=list(copname="frank",data=udat,quad=gl,repar=0);
ifixed=rep(FALSE,np);
ml= pdhessminb(stfrk2,f90cop2nllk,ifixed=ifixed,dstruct=dstructfrk,
  LB=LB.frk2,UB=UB.frk2,iprint=TRUE,eps=1.e-4);
```

---

pnestfactcop *Bivariate marginal copula cdfs for nested-factor copula models*

Description

Bivariate marginal copula cdfs for nested-factor copula models

Usage

```
pnest2cop(u1,u2,dcop1,pcondcop2,param1,param2,nq)
pnest2frk(u1,u2,param)  # nq defaulted to 35 etc
pnest2gum(u1,u2,param)  # nq defaulted to 35 etc
pnest2t(u1,u2,param,df)  # nq defaulted to 35 etc
pnest2tgum(u1,u2,param,df)  # nq defaulted to 35 etc
pnest2tbb1(u1,u2,param,df)  # nq defaulted to 35 etc
pnest2gumbb1(u1,u2,param)  # nq defaulted to 35 etc
```

Arguments

u1	vector of values in interval 0,1;
u2	vector of values in interval 0,1; same length as u1
param1	vector of length 2 or 2xq matrix where q is number of parameters for the bivariate copula (e.g. BB1) in dcop1; parameters that link observed variables to common latent

param2	vector of length 2 or 2xq matrix where q is number of parameters for the bivariate copula in pcondcop2; parameters that link observed variables to nested group latent variable
param	column 1 has parameters for global/common latent, column 2 (and column 3 for pnest2tbb1,pnest2gumbb1) has parameters for group latent
df	shape or df parameter for bivariate t linking copula
dcop1	function for pdf of copula family for global/common latent
pcondcop2	function for conditional cdf of copula family for nested group factor
nq	number of quadrature points for Gauss-Legendre quadrature

Details

This function is the bivariate marginal cdf of nested copula for two variables in different groups (for within the same group, the bivariate maeginal cdf is a 1-factor copula margin.

Value

cdf value(s)

See Also

structcop

Examples

```
pnest2fgm=function(u1,u2,param)
{ f=pnest2cop(u1,u2,dfgm,pcondfgm,param[,1],param[,2],35);
  f=f*(f<=1)+(f>1)
  f
}
th1a=frk.b2cpar(.7)
th1b=frk.b2cpar(.6)
th2a=frk.b2cpar(.5)
th2b=frk.b2cpar(.4)
u1=seq(.1,.9,.2)
u2=seq(.3,.7,.1)
pnest2frk(u1,u2,matrix(c(th1a,th1b,th2a,th2b),2,2))
pnest2gum(u1,u2,matrix(c(th1a,th1b,th2a,th2b),2,2))
pnest2t(u1,u2,matrix(c(.5,.6,.5,.4),2,2),c(5,5))
pnest2tgum(u1,u2,matrix(c(.5,.6,1.5,1.4),2,2),5)
pnest2tbb1(u1,u2,matrix(c(.5,.6,.4,.5,1.5,1.4),2,3),5)
pnest2gumbb1(u1,u2,matrix(c(1.5,1.6,.4,.5,1.5,1.4),2,3))
pnest2fgm(u1,u2,matrix(c(.5,.4,.6,.7),2,2))
```

---

rbivcop2param *Bivariate 2-parameter copula families: simulation*

Description

Bivariate 2-parameter copula families: simulation

Usage

```
rbb1(n,cpar,type="qcond",icheck=F)
rbb2(n,cpar,type="qcond",icheck=F)
rbb3(n,cpar,type="qcond",icheck=F)
rbb6(n,cpar,type="qcond",icheck=F)
rbb7(n,cpar,type="qcond",icheck=F)
rbb4(n,cpar,icheck=F)
rbb5(n,cpar,icheck=F)
rbb8(n,cpar,icheck=F)
rbb9(n,cpar,icheck=F)
rbb10(n,cpar,icheck=F)
```

Arguments

`n`	sample size
`cpar`	copula parameter: vector of length 2
`type`	"qcond" for conditional approach using $C_{2\|1}^{-1}$; "mix" for stochastic representation based on mixture of max-id
`icheck`	flag to output means and estimated correlation of sample

Value

nx2 matrix with values in (0,1)

References

Joe H (1997). Multivariate Models and Dependence Concepts. Chapman & Hall.

See Also

rLTstochrep

Examples

```
n=500
set.seed(12345)
udat1=rbb1(n,c(.5,1.2),icheck=TRUE)
set.seed(12345)
udat2=rbb1(n,c(.5,1.2),type="mix",icheck=TRUE)
# another method based on Mittag-Leffler LT of the BB1 Archimedean copula
set.seed(12345)
udat3=rmbb1(n,2,c(.5,1.2))
print(summary(udat3))
print(cor(udat3)[1,2])
```

`rcop` *Simulation from parametric bivariate copula families*

Description

Simulation from parametric bivariate copula families (one-parameter and two-parameter)

Usage

```
#rcop(n,cpar)
rcop(n,cpar,icheck=F)
```

Arguments

`n`	sample size
`cpar`	copula parameter: could be scalar or vector depending on the copula family
`icheck`	if T, output average u, average v and (Spearman) correlation

Details

Choices are 'cop' in rcop

pla, frk, mtcj, joe, gum, gumr (reflected Gumbel), gal, hr, bb1, bb2, bb3, bb4, bb5, bb6, bb7, bb8, bb9, bb10 for bivariate families;

See help page for pcop for the abbreviations of the copula names.

breflasym, bpermasym, basymgum1, bMO1 for 'cop' in rcop for extreme cases of bivariate reflection or permutation asymmetric:

bivariate permutation asymmetric,

bivariate asymmetric Gummbel with one skew parameter,

bivariate Marshall-Olkin skew 1-parameter subfamily.

Value

nx2 matrix with values in (0,1)

See Also

`pcop` `rbivcop2param`

Examples

```
n=1000
set.seed(12345)
cpar=gum.tau2cpar(.45)
udat=rgum(n,cpar)
taucor(udat[,1],udat[,2])
cpar=depmeas2cpar(.45,"rhoS","gumbel")
udat=rgum(n,cpar)
cor(udat[,1],udat[,2],method="spearman")
cor(udat[,1],udat[,2])
# see help page for skewrefl for use of
# rbreflasym, rbpermasym, rbsymgum1, rbMO1
```

`rcopulaGARCH` *Simulation for copula GARCH models with factor structure*

Description

Simulation for copula GARCH models with factor structure, with links to C code

Usage

```
rgarch1fact(n,garchpar,cpar,sigma0,copcode)
rgarch2fact(n,garchpar,cpar,sigma0,copcode)
rgarchbifact(n,grsize,garchpar,cpar,sigma0,copcode)
rgarchnestfact(n,grsize,garchpar,cpar,sigma0,copcode)
rgarchbifactmvt(n,grsize,garchpar,cpar,sigma0,copcode=2) # monotone interpolatio
rgarchnestfactmvt(n,grsize,garchpar,cpar,sigma0,copcode=2) # monotone interpolat
```

Arguments

`n`	sample size
`garchpar`	6xd matrix, where d is the number of assets, rows are mu,ar1,omega,alpha1,beta1,nu; set ar1 to be 0 vector if AR term not used in GARCH model
`cpar`	copula parameter vector
`grsize`	vector of group sizes for mgrp groups with sum(grsize)=d
`sigma0`	d-vector with starting values for conditional SDs (from GARCH output)
`copcode`	current options are 1 for Gaussian, 2 for t, 3 for Gumbel, -3 for reflected Gumbel, 5 for Frank; 9 for BB1 1-factor, 9 for BB1/Frank 2-factor and b-factor; 11 for Gumbel/BB1 nested factor

Details

rgarchbifactmvt and rgarchnestfactmvt use monotone interpolation for the Student t univariate cdf, for faster computations

Value

`lgret`	dxn matrix of log returns
`portfret`	nx1 portfolio return vector, assuming equally weighted

See Also

`rfactcop` `structcop`

Examples

```
garchpar=matrix(c(0.094, 0.125, 0.068, 0.212, 0.092,
 -0.062,-0.040,-0.082,-0.020,-0.051,
  0.019, 0.014, 0.016, 0.082, 0.014,
  0.071, 0.075, 0.088, 0.092, 0.080,
  0.909, 0.914, 0.882, 0.842, 0.902,
```

```
 9.4, 8.3,9.5, 7.9,9.6), 6,5,byrow=TRUE)
sigma0=c(1.113,1.286,0.857,1.176,0.968)
garchpar6=cbind(garchpar,c(0.1, -0.05,0.01,0.08,0.9,10.))
sigma6=c(sigma0,1)
grsize=c(2,2,2)
cpar1=seq(1.1,1.5,.1)
cpar2=c(seq(1.1,1.5,.1),rep(1.1,5))
cparbi=c(seq(1.1,1.6,.1),rep(1.1,6))
cparne=c(rep(1.1,3),seq(1.1,1.6,.1))
#
set.seed(123)
out=rgarch1fact(3,garchpar,cpar1,sigma0,copcode=3)
print(out)
set.seed(123)
out=rgarch2fact(3,garchpar,cpar2,sigma0,copcode=3)
print(out)
set.seed(123)
out=rgarchbifact(3,grsize,garchpar6,cparbi,sigma6,copcode=3)
print(out)
set.seed(123)
out=rgarchnestfact(3,grsize,garchpar6,cparne,sigma6,copcode=3)
print(out)
```

`rectmult` *multivariate rectangle probabilities for copula models*

Description

multivariate rectangle probabilities for copula models with discrete responses

Usage

```
rectmult(u1vec,u2vec,cpar,pmcop)
rectmfrk(u1vec,u2vec,cpar)  # multivariate exchangeable Frank
rectmgum(u1vec,u2vec,cpar)  # multivariate exchangeable Gumbel
rectmgal(u1vec,u2vec,cpar)  # multivariate exchangeable Galambos
rectemvn(u1vec,u2vec,cpar)  # multivariate exchangeable normal
```

Arguments

`u1vec`	vector of dimension d, lower corner of rectangle
`u2vec`	vector of dimension d, upper corner of rectangle
`cpar`	parameter of the d-variate copula
`pmcop`	function for the cdf of the d-variate copula

Value

rectangle probability

See Also

`copmultiv`

Examples

```
rectmgum(c(.1,.2,.1),c(.8,.9,.7),2)
rectmgum(c(.1,.2,.1),c(.8,.9,.7),4)
rectmult(c(.1,.2,.1),c(.8,.9,.7),4,pmcop=pmgum)
rectmfrk(c(.1,.2,.1),c(.8,.9,.7),2)
rectmfrk(c(.1,.2,.1),c(.8,.9,.7),4)
rectmgal(c(.1,.2,.1),c(.8,.9,.7),3)
rectemvn(c(.1,.2,.1),c(.8,.9,.7),0.6)
```

---

rfactcop *Simulation for copula models with factor structure*

Description

Simulation for copula models with factor structure, with links to C code

Usage

```
r1fact(n,d,cpar,copcode)
r2fact(n,d,cpar,copcode)
rbifact(n,grsize,cpar,copcode)
rnestfact(n,grsize,cpar,copcode)
```

Arguments

n	sample size
d	number of variables
cpar	copula parameter vector
grsize	vector of group sizes for mgrp groups with sum(grsize)=d
copcode	current options are 1 for Gaussian, 2 for t, 3 for Gumbel, -3 for reflected Gumbel, 5 for Frank; 9 for BB1 1-factor, 9 for BB1/Frank 2-factor and bi-factor; 11 for Gumbel/BB1 nested factor

Details

These do simulations via C code; currently there are not as many possible options compared with sim1fact(), sim2fact(), simbifact() and simnestfact()

Value

nxd matrix of d dependent U(0,1) variables

See Also

factorcopsim structcop

Examples

```
set.seed(123)
rhvec=c(.8,.7,.6,.5,.5)
udat1=r1fact(3,5,rhvec,1); print(udat1)
udat2=r1fact(3,5,c(rhvec,4),2); print(udat2)
cpar1=seq(1.1,1.5,.1)
udat3=r1fact(3,5,cpar1,3); print(udat3)
cparbb1=c(.2,1.1,.2,1.2,.2,1.3,.2,1.4,.2,1.5)
udat9=r1fact(3,5,cparbb1,9); print(udat9)
#
set.seed(123)
cpar2=c(seq(1.1,1.5,.1),rep(1.1,5))
udat3=r2fact(3,5,cpar2,3); print(udat3)
cpar2bb1=c(cparbb1,seq(1.1,1.5,.1))
udat9=r2fact(3,5,cpar2bb1,9); print(udat9)
#
grsize=c(2,2,2)
set.seed(123)
cparbi=c(seq(1.1,1.6,.1),rep(1.1,6))
udat3=rbifact(3,grsize,cparbi,3); print(udat3)
cparbibb1=c(cparbb1,.2,1.6,seq(1.1,1.6,.1))
udat9=rbifact(3,grsize,cparbibb1,9); print(udat9)
#
grsize=c(2,2,2)
set.seed(123)
cparne=c(rep(1.1,3),seq(1.1,1.6,.1))
udat3=rnestfact(3,grsize,cparne,3); print(udat3)
cparnebb1=c(seq(0.6,1.1,.1),cparbb1)
udat9=rnestfact(3,grsize,cparnebb1,9); print(udat9)
```

---

rhoNsemic *Semi-correlations for bivariate copula*

Description

Semi-correlations for bivariate copula

Usage

```
rhoNsemic(cpar,dcop,pcop,pcond,B=6,isym=F,iinfbd=T,iprint=F)
```

Arguments

cpar	copula parameter: scalar or vector
dcop	function for bivariate copula pdf
pcop	function for bivariate copula cdf
pcond	function for $C_{2\|1}(v\|u)$
B	bound on numerical integration on $[0, B]^2$ with respect to standard normal densities
isym	flag for reflection symmetric, default is F

iinfbd	flag for infinity as limit; if T upper bound is Inf for 1-dimensional integral, otherwise B; default is T
iprint	print flag for intermediate results

Value

vector of length 2 with lcorr = lower normal scores semi-correlation of bivariate copula, ucorr = upper normal scores semi-correlation of bivariate copula.

See Also

pcop semicor

Examples

```
rhoNsemic(2,dfrk,pfrk,pcondfrk,B=6,isym=TRUE)
rhoNsemic(2,dgum,pgum,pcondgum,B=6,isym=FALSE)
rhoNsemic(2,dgumr,pgumr,pcondgumr,B=6,isym=FALSE)
```

---

rLTstochrep *Simulation of random variables from LTs used in Archimedean copulas. Simulation from multivariate Archimedean copulas.*

Description

Simulation of random variables from LTs used in Archimedean copulas. Simulation from multivariate Archimedean copulas.

Usage

```
rpostable(n,alp) # 0<alp<1
rsibuya(n,alp) # 0<alp<1
rlogseries(n,cpar) # cpar as for Frank leads to better parametrization
rmitlef(n,param)
rgammaSgamma(n,param)
rpostableSgamma(n,param)
rsibuyaSpostable(n,param)
rsibuyaSgamma(n,param)
rmfrk0(n,d,cpar) # R version
rmfrk(n,d,cpar,icheck=F) # link to C
rmcop(n,d,cpar)  # choices for 'cop' for mtcj,joe,gum,bb1,bb2,bb3,bb6,bb7,bb10
```

Arguments

n	sample size
d	dimension for multivariate Archimedean
alp	parameter of Laplace transform (LT)
param	(vector) parameter of Laplace transform (LT)
cpar	copula parameter: could be scalar or vector depending on the copula family
icheck	flag to print out means and correlation as checks

Details

The LT families matching the Archimedean copula families are:

logseries for Frank;

gamma for MTCJ=Mardia-Takahasi-Cook-Johnson;

Sibuya for Joe;

positive stable for Gumbel;

Mittag-Leffler (or gamma stopped positive stable) for multivariate version of BB1;

gammaSgamma (gamma stopped gamma) for multivariate version of BB2;

postableSgamma (positive stable stopped gamma) for multivariate version of BB3;

sibuyaSpostable (Sibuya stopped positive stable) for multivariate version of BB6;

sibuyaSgamma (Sibuya stopped gamma) for multivariate version of BB7;

shifted negative binomial (see code) for multivariate version of BB10.

Value

vector for rpostable to rsibuyaSgamma, nxd matrix for rmfrk to rmbb10.

References

Joe H (2014). Dependence Modeling with Copulas. Chapman&Hall/CRC. See Appendix for the names of some of the LTs and the source of the algorithms.

See Also

rcop rbivcop2param

Examples

```
cpar=c(1,2); n=1000
r=rmitlef(n,cpar)
print(summary(r))
uu=rmbb1(n,d=3,cpar)
print(summary(uu))
print(taucor(uu[,1],uu[,2]))
print(taucor(uu[,1],uu[,3]))
print(taucor(uu[,2],uu[,3]))
tau=bb1.cpar2tau(cpar)
cat("theor.tau=",tau,"\n")
```

---

rvinediscbvnnllk *Negative log-likelihood for discrete R-vine with Gaussian pair-copulas and ordinal response*

Description

Negative log-likelihood for discrete R-vine with Gaussian pair-copulas and ordinal response

Usage

```
rvinediscbvnnllk(parvec,zzdat,A)  # univariate margins converted to zzdat
rvinediscbvnfullnllk(parvec,A,xmat,yvec,nrep,ncateg) # full max likelihood
```

Arguments

parvec	parameter vector of partial correlations with length d*(d-1)/2
zzdat	dimension nclx(2d) with corners of rectangle , N(0,1) scale
A	dxd vine array with 1:d on diagonal
xmat	nn*npred matrix, nn=nrep*ncl=d*ncl, ncl=#clusters
yvec	integer-valued vector of length nn, values in 0:(ncateg-1) or 1:ncateg
nrep	cluster size d or #repeated ordinal measures
ncateg	number of ordinal categories

Value

negative log-likelihood

See Also

rvinediscrete

Examples

```
data(ordinalex)
xvec=c(t(ordinalex$xx))
yvec=c(t(ordinalex$yy))
uni=ordprobit.univar(xvec,yvec,iprint=FALSE)
latentdat=mord2uu(xvec,yvec,nrep=4,uni$cutpts,uni$beta)
uudat=latentdat$uudat
zzdat=latentdat$zzdat
D4=Dvinearray(4)
param=c(.5,.5,.5,.1,.1,.1)
temz=rvinediscbvnnllk(param,zzdat,D4)
print(temz)
mlz=nlm(rvinediscbvnnllk,p=param,zzdat=zzdat,A=D4,hessian=TRUE,print.level=1)
fparam=c(uni$cutpts,uni$beta, mlz$estimate)
fnllk=rvinediscbvnfullnllk(fparam,D4,xvec,yvec,nrep=4,ncateg=3)
print(fnllk)
fnllk2=rvinediscbvnfullnllk(fparam,D4,xvec,yvec-1,nrep=4,ncateg=3)
print(fnllk2)
mlf=nlm(rvinediscbvnfullnllk,fparam,A=D4,xmat=xvec,yvec=yvec,nrep=4,ncateg=3,
   hessian=TRUE,print.level=1)
```

`rvinediscrete` *Negative log-likelihoods for regular discrete vine models*

Description

Probabilities and negative log-likelihoods for regular discrete vine models

Usage

```
rvinepmf.discrete(parmat,u1vec,u2vec,A,M,pcopnames,iprint=F)
dvinepmf.discrete(parmat,u1vec,u2vec,pcopnames,iprint=F)
rvinediscretenllk(parvec,uudat,A,pcopnames,LB=0,UB=10,iprint=F)
rvinediscfullnllk(parvec,ydat,xdat,nrep,upmfcdf,npar1,A,pcopnames,
  LB=0,UB=10,zero=0.00001,one=0.99999,iprint=F) # repeated measures
rvinepmf.ordinal(parmat,yvec,A,M,pcopnames,ucuts,iprint=F)
rvineordinalnllk(parvec,ydat,ncateg,A,pcopnames,LB=0,UB=10,iprint=F)
# all of these assume 1-dimensional parameter for each edge of vine
```

Arguments

`parmat`	dxd parameter matrix for the model, 1-parameter pair-copulas, position according to A
`parvec`	parameter vector for the model
`u1vec`	d-dimension lower vector of hyperrectangle
`u2vec`	d-dimension upper vector of hyperrectangle
`uudat`	dimension nx(2d) with corners of rectangle probabilities for each discrete vector observation for rvinediscretenllk
`yvec`	integer-valued d-vector, values in 0:(ncateg-1) for rvinepmf.ordinal
`ncateg`	number of categories for rvineordinalnllk
`ydat`	d-dimension ordinal or discrete response; nxd matrix, values in 0:(ncateg-1) or 1:ncateg
`xdat`	covariate matrix ((n*d)xq), q=#covariates, d=#repeated measurements, for rvinediscfullnllk
`nrep`	#repeated measurements per subject for rvinediscfullnllk
`ucuts`	(ncateg+1)xd matrix of cut points for ordinal, computed from unifcuts via ucuts=unifcuts(y), ucuts=rbind(rep(0,d),ucuts,rep(1,d)), for rvinepmf.ordinal
`upmfcdf`	function that computes the pmf,cdf up to the value mx
`npar1`	number of parameters in upmfcdf
`A`	dxd vine array with 1:d on diagonal
`M`	dxd vine maximum array, $mxarray component of varray2M(A)
`pcopnames`	string vector with names of pair-copula cdfs of length ntrunc, ntrunc=truncation level
`LB`	lower bound of components of param
`UB`	upper bound of components of param
`zero`	either 0 or epsilon depending on the copula
`one`	either 1 or 1-epsilon depending on the copula
`iprint`	print flag for intermediate steps

Details

dvinepmf.discrete() was written before rvinepmf.discrete() because the algorithm looks simpler for a D-vine versus the general R-vine. rvinepmf.discrete() can be tested with a D-vine and matched to the output of rvinepmf.discrete().

Value

probability for rvinepmf.discrete and rvinepmf.ordinal

negative log-likelihood value for the nllk functions

References

Panagiotelis A, Czado C and Joe H (2012). Pair Copula Constructions for Multivariate Discrete Data. Journal of the American Statistical Association, 107, 1063-1072.

See Also

discreteresponse rvinediscbvnnllk

Examples

```
# discrete response
D=Dvinearray(4); out=varray2M(D); M=out$mxarray
parmat=matrix(0,4,4); parmat[1,2:4]=2; parmat[2,3:4]=1.7; parmat[3,4]=1.1
u1vec=c(.1,.2,.1,.2); u2vec=c(.5,.6,.5,.4)
pcopnames=rep("pgum",3)
rvinepmf.discrete(parmat,u1vec,u2vec,D,M,pcopnames,iprint=FALSE)
d=5
dd=d*(d-1)/2; n=5
uudat=matrix(c(
0.50,0.49,0.28,0.43,0.43,0.63,0.61,0.47,0.56,0.56,
0.31,0.50,0.73,0.39,0.17,0.51,0.63,0.78,0.52,0.33,
0.06,0.66,0.48,0.49,0.00,0.15,0.72,0.58,0.59,0.07,
0.00,0.00,0.00,0.00,0.00,0.58,0.57,0.55,0.52,0.53,
0.99,0.98,0.98,0.99,0.98,0.99,0.98,0.98,0.99,0.98),n,2*d,byrow=TRUE)
D5=Dvinearray(d)
out=varray2M(D5); M=out$mxarray
rvinediscretenllk(rep(2.2,dd),uudat,D5,pcopnames=rep("pfrk",4),LB=-10,UB=30)
mle=nlm(rvinediscretenllk,p=rep(2.2,dd),hessian=TRUE,print.level=1,
  uudat=uudat,A=D5,pcopnames=rep("pfrk",4),LB=rep(-10,dd),UB=rep(20,dd))
# ordinal response
data(ltmconv)
d=ncol(sci); nitem=ncol(sci); nc=4
perm=c(6,4,3,1,5,7,2)
sciperm=sci[,perm]
A=vnum2array(d,320); out=varray2M(A); M=out$mxarray
ucuts=unifcuts(sciperm); ucuts=rbind(rep(0,d),ucuts,rep(1,d))
par3=c(2,2,2,2,2,2, 1.3,1.3,1.3,1.3,1.3, 1.1,1.1,1.1,1.1,1.1)
parmat3=matrix(0,d,d)
parmat3[1,2:d]=par3[1:6]
parmat3[2,3:d]=par3[7:11]
parmat3[3,4:d]=par3[12:15]
pcopnames=rep("pgum",3)
rvinepmf.ordinal(parmat3,sciperm[16,],A,M,pcopnames,ucuts)
nllk=rvineordinalnllk(par3,sciperm,nc,A,pcopnames,LB=rep(1,15),UB=rep(10,15),iprint=FALSE
print(nllk)
```

`rvinenllk` *Negative log-likelihoods for regular vine models*

Description

Log probability density and negative log-likelihoods for regular vine models

Usage

```
rvinelogpdf(uvec,A,parmat,logdcop,pcond,iprint=F) # this is a template
cvinelogpdf(uvec,parmat,logdcop,pcond,iprint=F) # this is a template
dvinelogpdf(uvec,parmat,logdcop,pcond,iprint=F) # this is a template
rvinenllk.trunc(parvec,udat,A,logdcopnames,pcondnames,np,ifixed,parfixed,LB=0,UB
   # common pair-copula family for each tree
#rvinellkv.trunc(parvec,udat,A,logdcopnames,pcondnames,np)
rvinenllk1.trunc(parvec,udat,A,logdcopnames,pcondnames,LB=0,UB=10)
   # scalar parameter for each edge of vine, can be used with t copulas with
   #  fixed shape parameters
rvinenllk.trunc2(parvec,udat,A,ntrunc,logdcopmat,pcondmat,np,
  ifixed,parfixed, LB=0,UB=10)
   # can be different pair-copula family for each edge of vine
rvinellkv.trunc2(parvec,udat,A,ntrunc,logdcopmat,pcondmat,np)
rvinenllk1.nonsimpl(parvec,udat,A,logdcopnames,pcondnames,ib0fixed=F,
  b0fixed=0,iprint=F,LB=0,UB=10)
```

Arguments

`parmat`	dxd parameter matrix for the model, 1-parameter pair copulas, position according to A
`parvec`	parameter vector for the model
`uvec`	d-vector of uniform scores for rvinelogpdf
`udat`	nxd matrix of uniform scores for rvinenllk functions
`A`	dxd vine array with 1:d on diagonal
`ntrunc`	truncation level between 1 and d-1
`logdcop`	function for log copula pdf; same for all edge of a vine in rvinelogpdf
`pcond`	function for copula conditional cdf; same for all edge of a vine in rvinelogpdf
`logdcopnames`	string vector with names of log copula pdfs of length ntrunc, ntrunc=truncation level
`pcondnames`	string vector with names of copula conditional cdfs of length ntrunc, ntrunc=truncation level
`logdcopmat`	matrix of names of log copula pdfs for trees 1,...,ntrunc
`pcondmat`	matrix of names of conditional cdfs for trees 1,...,ntrunc
`np`	dxd where np[ell,j] is size for parameter[ell,j] for bivariate copula in tree ell, linking variables j and A[ell,j]
`ifixed`	length equal to length(param)+length(parfixed)
`parfixed`	dimension equal to sum(ifixed) for the positions where the ifixed vector is T, parfixed[1] goes to the first fixed position

LB	lower bound of components of parvec
UB	upper bound of components of parvec; scalar or same length as parvec
ib0fixed	T or F for whether intercept b0 is fixed
b0fixed	fixed intercept when copula parameter has form cpar=exp(b0fixed+b1*sum(ucond)) for trees 2,3.. for rvinenllk1.nonsimpl
iprint	print flag for intermediate calculations

Value

log probability density function for rvinelogpdf, cvinelogpdf, dvinelogpdf;

vector of log pdf or log-likelihood (one element for each row of udat, for llkv functions, to be used in Vuong's procedure

negative log-likelihood value for the nllk functions;

See Also

rvinenllkderiv rvinenllkpseud rvinesim

Examples

```
uvec=c(.1,.3,.4,.5,.7)
parmat= matrix(c(0,0,0,0,0, 1.5,0,0,0,0, 1.5,1.2,0,0,0, 1.5,1.2,1.3,0,0,
   1.5,1.2,1.3,1.4,0), 5,5)
A=vbin2array(5,6) # not C or D-vine
C=Cvinearray(5); D=Dvinearray(5)

cat("\nR-vine\n")
lpdf=rvinelogpdf(uvec,A,parmat,logdcop=logdgum,pcond=pcondgum,iprint=FALSE)
print(lpdf)

cat("\nC-vine\n")
lpdf=rvinelogpdf(uvec,C,parmat,logdcop=logdgum,pcond=pcondgum,iprint=FALSE)
print(lpdf)
lpdfc=cvinelogpdf(uvec,parmat,logdcop=logdgum,pcond=pcondgum,iprint=TRUE)

cat("\nD-vine\n")
lpdf=rvinelogpdf(uvec,D,parmat,logdcop=logdgum,pcond=pcondgum,iprint=FALSE)
print(lpdf)
lpdfd=dvinelogpdf(uvec,parmat,logdcop=logdgum,pcond=pcondgum,iprint=TRUE)

d=5
qcondnames=rep("qcondfrk",4)
pcondnames=rep("pcondfrk",4)
logdcopnames=rep("logdfrk",4)
pcondnames=rep("pcondfrk",4)
parvec=c(3.6,3.6,3.6,3.6, 1.5,1.5,1.5, 1.4,1.4, 0.3)
A=vnum2array(d,3)
set.seed(123)
nsim=20
np=matrix(0,d,d)
np[1,2:d]=1; np[2,3:d]=1; np[3,4:d]=1; np[4,5]=1
udat=rvinesimvec(nsim,A,parvec,np,qcondnames,pcondnames,iprint=FALSE)
pcondmat=matrix(c("",rep("pcondfrk",4),"","",rep("pcondfrk",3),
  "","","",rep("pcondfrk",2),"","","","","pcondfrk",rep("",5)),5,5,byrow=TRUE)
```

```
logdcopmat=matrix(c("",rep("logdfrk",4),"","",rep("logdfrk",3),
  "","","",rep("logdfrk",2),"","","","","logdfrk",rep("",5)),5,5,byrow=TRUE)
rvinenllk1.trunc(parvec,udat,A,logdcopnames,pcondnames,LB=-10,UB=30)
rvinenllk.trunc(parvec,udat,A,logdcopnames,pcondnames,np,ifixed=rep(FALSE,10),
   parfixed=NA,LB=-10,UB=30)
rvinenllk.trunc2(parvec,udat,A,ntrunc=4,logdcopmat,pcondmat,np,
   ifixed=rep(FALSE,10),parfixed=NA,LB=-10,UB=30) # same as above
mle=nlm(rvinenllk1.trunc,p=parvec,
  udat=udat,A=A,logdcopnames=logdcopnames,pcondnames=pcondnames,
  hessian=TRUE,iterlim=30,print.level=1,LB=-10,UB=30)
rvinellkv.trunc(mle$estimate,udat,A,logdcopnames,pcondnames,np)
rvinellkv.trunc2(mle$estimate,udat,A,ntrunc=4,logdcopmat,pcondmat,np)
```

`rvinenllkderiv` *Negative log-likelihood and gradient for regular vine models*

Description

Negative log-likelihood and gradient for regular vine models

Usage

```
rvinenllkder1.trunc(parvec,udat,A,logdcopdernames,pconddernames,LB=0,UB=10)
dvinenllkder1.trunc(parvec,udat,logdcopdernames,pconddernames,LB=0,UB=10)
rvinenllkder2.trunc(parvec,udat,A,logdcopdernames,pconddernames,LB=0,UB=10)
```

Arguments

`parvec`	parameter vector for the model
`udat`	nxd matrix of uniform scores for rvinenllk.trunc
`A`	dxd vine array with 1:d on diagonal
`logdcopdernames`	string vector with names of log copula pdfs and derivatives, length ntrunc, ntrunc=truncation level
`pconddernames`	string vector with names of copula conditional cdfs and derivatives, length ntrunc, ntrunc=truncation level
`LB`	lower bound of components of parvec
`UB`	upper bound of components of parvec; scalar or same length as parvec

Details

dvinenllkder1.trunc() was written before rvinenllkder1.trunc() because the algorithm looks simpler for a D-vine versus the general R-vine. rvinenllkder1.trunc() can be tested with a D-vine and matched to the output of rvinenllkder1.trunc().

Value

negative log-likelihood value with gradient as an atribute; suitable for use with nlm.

See Also

rvinenllk

Examples

```
d=5
A=vnum2array(d,3)
nsim=20
np=matrix(0,d,d)
# example 1
qcondnames=rep("qcondfrk",4)
pcondnames=rep("pcondfrk",4)
logdcopdernames=rep("logdfrk.deriv",4)
pconddernames=rep("pcondfrk.deriv",4)
parvec=c(3.6,3.6,3.6,3.6, 1.5,1.5,1.5, 1.4,1.4, 0.3)
set.seed(123)
np[1,2:d]=1; np[2,3:d]=1; np[3,4:d]=1; np[4,5]=1
udat=rvinesimvec(nsim,A,parvec,np,qcondnames,pcondnames,iprint=FALSE)
mle=nlm(rvinenllkder1.trunc,p=parvec,udat=udat,A=A,
  logdcopdernames=logdcopdernames,pconddernames=pconddernames,
  hessian=TRUE,iterlim=30,print.level=1,LB=-10,UB=30,check.analyticals=FALSE)
mle2=nlm(rvinenllkder1.trunc,p=parvec[1:7],udat=udat,A=A,
  logdcopdernames=logdcopdernames[1:2],pconddernames=pconddernames[1:2],
  hessian=TRUE,iterlim=30,print.level=1,LB=-10,UB=30,check.analyticals=FALSE)

# example 2
qcondnames=c("qcondbb1",rep("qcondfrk",3))
pcondnames=c("pcondbb1",rep("pcondfrk",3))
logdcopdernames=c("logdbb1.deriv",rep("logdfrk.deriv",3))
pconddernames=c("pcondbb1.deriv",rep("pcondfrk.deriv",3))
parvec=c(0.5,1.6,0.5,1.6,0.5,1.6,0.5,1.6, 1.5,1.5,1.5, 1.4,1.4, 0.3)
np[1,2:d]=2; np[2,3:d]=1; np[3,4:d]=1; np[4,5]=1
set.seed(123)
udat=rvinesimvec(nsim,A,parvec,np,qcondnames,pcondnames,iprint=FALSE)
lb=c(rep(c(0,1),4),rep(-10,6))
ub=c(rep(c(6,6),4),rep(30,6))
mle=nlm(rvinenllkder2.trunc,p=parvec,udat=udat,A=A,
 logdcopdernames=logdcopdernames,pconddernames=pconddernames,
 hessian=TRUE,iterlim=30,print.level=1,LB=lb,UB=ub,check.analyticals=FALSE)
mle2=nlm(rvinenllkder2.trunc,p=parvec[1:11],udat=udat,A=A,
 logdcopdernames=logdcopdernames[1:2],pconddernames=pconddernames[1:2],
 hessian=TRUE,iterlim=30,print.level=1,LB=lb[1:11],UB=ub[1:11],check.analyticals=FALSE)
```

---

rvinenllkpseud *Pseudo observations after fitting truncated R-vine copulas*

Description

Pseudo observations after fitting truncated R-vine copulas

Usage

```
rvinenllkpseud(parvec,udat,A,logdcopnames,pcondnames,np)
   # common pair-copula family for each tree
rvinenllkpseud2(parvec,udat,A,ntrunc,logdcopmat,pcondmat,np)
   # can be different pair-copula family for each edge of vine
```

Arguments

parvec	parameter vector for the model
udat	nxd matrix of uniform scores for rvinenllk functions
A	dxd vine array with 1:d on diagonal
ntrunc	truncation level between 1 and d-1
logdcopnames	string vector with names of log copula pdfs of length ntrunc, ntrunc=truncation level
pcondnames	string vector with names of copula conditional cdfs of length ntrunc, ntrunc=truncation level
logdcopmat	matrix of names of log copula pdfs for trees 1,...,ntrunc
pcondmat	matrix of names of conditional cdfs for trees 1,...,ntrunc
np	dxd where np[ell,j] is size for parameter[ell,j] for bivariate copula in tree ell, linking variables j and A[ell,j]

Value

nllk	negative log-likelihood
condforw	nx(d-ntrunc) matrix with $C_{j\|a_{ntrunc,j};S}$ in the forward direction
condbackw	nx(d-ntrunc) matrix with $C_{a_{ntrunc,j}\|j;S}$ in the backward direction

See Also

rvinenllk

Examples

```
parmat=matrix(c(0,1.5,2,2.5,2.2,0,0,.5,.4,.6,0,0,0,.2,.2,0,0,0,.2,.2,0,0,0,0,.2),5,5,byrc
parvec=c(parmat[1,2:5],parmat[2,3:5],parmat[3,4:5],parmat[4,5])
pcondnames=c("pcondgum","pcondbvncop","pcondbvncop","pcondbvncop")
qcondnames=c("qcondgum","qcondbvncop","qcondbvncop","qcondbvncop")
np=matrix(1,5,5)
C5=Cvinearray(5)
set.seed(123)
udat=rvinesimvec(300,C5,parvec,np,qcondnames,pcondnames)
# fit 1-truncated
logdcopnames1="logdgum"
mle1=nlm(rvinenllk1.trunc,p=parvec[1:4],
    udat=udat,A=C5,logdcopnames=logdcopnames1,pcondnames=pcondnames[1],
    hessian=TRUE,iterlim=30,print.level=1,LB=1,UB=20)
pseud1=rvinenllkpseud(mle1$estimate,udat,C5,logdcopnames1,pcondnames[1],np)
# should be 2|1 3|1 4|1 5|1 in $condforw
zdat1=nscore(pseud1$condforw)
out=semicortable(zdat1,inscore=TRUE)
out[,1]=out[,1]+1
```

```
out[,2]=out[,2]+1
print(out)
# fit 2-truncated
logdcopnames2=c("logdgum","logdbvncop")
mle2=nlm(rvinenllk1.trunc,p=parvec[1:7],
    udat=udat,A=C5,logdcopnames=logdcopnames2,pcondnames=pcondnames[1:2],
    hessian=TRUE,iterlim=30,print.level=1,LB=c(1,1,1,1,-1,-1,-1),
    UB=c(20,20,20,20,1,1,1))
pseud2=rvinenllkpseud(mle2$estimate,udat,C5,logdcopnames2,pcondnames[1:2],np)
# should be 3|12 4|12 5|12 in $condforw
zdat2=nscore(pseud2$condforw)
out=semicortable(zdat2,inscore=TRUE)
out[,1]=out[,1]+2
out[,2]=out[,2]+2
print(out)
```

`rvinesim` *Simulation for R-vine copulas*

Description

Simulation for R-vine copulas, bivariate copulas in the same parametric family for each tree; current version assume each bivariate copula family is permutation symmetric

Usage

```
cvinesim(p,parmat,qcond,pcond)  # same pcond for all trees
dvinesim(p,parmat,qcond,pcond,iprint=F)  # same pcond for all trees
rvinesim0(p,A,parmat,qcond,pcond,iprint=F)  # same pcond for all trees
rvinesim1(p,A,parmat,qcondnames,pcondnames,iprint=F)
rvinesim2(p,A,parvec,np,qcondnames,pcondnames,iprint=F)
rvinesimvec(nsim,A,parvec,np,qcondnames,pcondnames,iprint=F)
  # common pair-copula family for each tree
rvinesimvec2(nsim,A,ntrunc,parvec,np,qcondmat,pcondmat,iprint=F)
  # can be different pair-copula family for each edge of vine
```

Arguments

`p`	vector of length d, e.g. runif(d)
`nsim`	sample size for simulation
`A`	dxd vine array with 1:d on diagonal, or ntrunc x d vine array as only ntrunc rows are used
`ntrunc`	truncation level between 1 and d-1
`qcond`	function for inverse conditional cdf $C^{-1}_{U\|V}(u\|v)$
`pcond`	function for conditional cdf $C_{U\|V}(u\|v)$
`qcondnames`	names of inverse conditional cdf functions $C^{-1}_{U\|V}(u\|v)$, for trees 1,...,ntrunc (so ntrunc=length(qcondnames) for these implementations)
`pcondnames`	names of conditional cdf functions $C_{U\|V}(u\|v)$, for trees 1,...,ntrunc
`qcondmat`	matrix of names of conditional quantile functions for trees 1,...,ntrunc

pcondmat	matrix of names of conditional cdfs for trees 1,...,ntrunc
parmat	dxd matrix: for rvinesim1, where all bivariate copula families have 1 parameter, parameter in parmat[ell,j] for ell<j is the parameter of the copula associated with A[ell,j]
parvec	vector: for rvinesim2 and rvinesimvec, with the union of the parameters associated with the copulas in A[ell,j], j=ell+1,...,d. ell=1,...,ntrunc
np	dxd matrix: for rvinesim2 and rvinesimvec, the dimension of the vector for the copulas in A[ell,j], j=ell+1,...,d. ell=1,...,ntrunc; the function will determine parvec[ip1:ip2] for the copula associated with A[ell,j]
iprint	print flag for intermediate results

Value

vector of length d of values on (0,1) for cvinesim, dvinesim, rvinesim0, rvinesim1 and rvinesim2.

data matrix (each element between 0 and 1) of dimension nsimxd for rvinesimvec

See Also

rvinenllk

Examples

```
d=5
C=Cvinearray(d); D=Dvinearray(d)
p=c(.4,.5,.6,.4,.7)
parmat=matrix(c(0,2,3,4,5,0,0,6,6,3,0,0,0,4,3,0,0,0,3,3,0,0,0,0,2),5,5,byrow=TRUE)
cvinesim(p,parmat,qcondfrk,pcondfrk)
rvinesim0(p,C,parmat,qcondfrk,pcondfrk) # same as above line
dvinesim(p,parmat,qcondfrk,pcondfrk)
rvinesim0(p,D,parmat,qcondfrk,pcondfrk) # same as above line
#
d=7
qcondnames=rep("qcondfrk",3)
pcondnames=rep("pcondfrk",3)
parvec=c(2,3,4,5,6,6, 3,4,3,3,3, 2,2,2,2)
np=matrix(0,d,d)
np[1,2:d]=1; np[2,3:d]=1; np[3,4:d]=1
A3=vnum2array(d,300)
nsim=10
uu1=matrix(0,nsim,d)
for(i in 1:nsim)
{ pp=runif(d)
  uu1[i,]=rvinesim2(pp,A3,parvec,np,qcondnames,pcondnames,iprint=FALSE)
}
print(summary(uu1))
print(cor(uu1))
# vectorized rvinesim
uu2=matrix(0,nsim,d)
uu2=rvinesimvec(nsim,A3,parvec,np,qcondnames,pcondnames,iprint=FALSE)
print(summary(uu2))
print(cor(uu2))
# BB1/Frank
qcondnames3=c("qcondbb1","qcondfrk","qcondfrk")
```

```
pcondnames3=c("pcondbb1","pcondfrk","pcondfrk")
parvec=c(.8,2,.7,1.6,.3,2.1,.6,1.5,.9,2.3,.9,2.4, 3,4,3,3,3, 2,2,2,2)
np2=matrix(0,d,d)
np2[1,2:d]=2; np2[2,3:d]=1; np2[3,4:d]=1
uu3=rvinesimvec(nsim,A3,parvec,np2,qcondnames3,pcondnames3,iprint=FALSE)
print(uu3)
# version with different pair-copula on different edges
d=4
D=Dvinearray(d)
np=matrix(0,d,d)
np[1,2:d]=1; np[2,3:d]=1; np[3,4:d]=1
qcondnames=c("qcondgum","qcondfrk","qcondfrk")
pcondnames=c("pcondgum","pcondfrk","pcondfrk")
parvec=c(2.0,2.2,1.6, 1.5,1.5, 0.3)
set.seed(1234)
rvinesimvec(2,D,parvec,np,qcondnames,pcondnames,iprint=FALSE)
pcondmat=matrix(c("",rep("pcondgum",3),"","",rep("pcondfrk",2),
  "","","","pcondfrk",rep("",4)),4,4,byrow=TRUE)
qcondmat=matrix(c("",rep("qcondgum",3),"","",rep("qcondfrk",2),
  "","","","qcondfrk",rep("",4)),4,4,byrow=TRUE)
set.seed(1234)
rvinesimvec2(2,D,ntrunc=3,parvec,np,qcondmat,pcondmat,iprint=FALSE) #as above
```

`rwmsubset` *Riphahn-Wambach-Million subset*

Description

50 subjects, with 5 repeated measures in years 1984, 1985, 1986, 1987, 1988 (5 lines per subject); variables are id, docvis, sex, year, age, hsat, handper, univ, public, addon, married, self.

This data set is a subset (of cases and variables) from "Riphahn RT, Wambach A, and Million A (2003). Incentive Effects in the Demand for Health Care: A Bivariate Panel Count Data Estimation. Journal of Applied Econometrics, 18 (4), 387-405."

Usage

```
data(rwmsubset)
```

`semicor` *Semi-correlations for bivariate or multivariate data set*

Description

Semi-correlations for bivariate or multivariate data set

Usage

```
semicor(bivdat,inscore=T)
semicortable(mdat,inscore=F)
```

Arguments

bivdat	data matrix with 2 columns
mdat	data matrix with d columns
inscore	T if bivdat or mdat has already been converted to normal scores

Value

For semicor, $ncorr: correlation of normal scores; $lcorr: lower semi-correlation of normal scores; $ucorr: upper semi-correlation of normal scores.

For semicortable, d*(d-1)/2 by 6 table with columns j1,j2,ncorr,lcorr,ucorr,bvnsemic

See Also

bvnsemic rhoNsemic tailweightedDepmeas

Examples

```
set.seed(1234)
n=300
x=matrix(rnorm(2*n),n,2)
rho=0.6
x[,2]=rho*x[,1]+sqrt(1-rho^2)*x[,2]
semicor(x,inscore=TRUE)
semicor(x,inscore=FALSE)
bvnsemic(rho)  # theoretical value
# log return data
data(euro0306)
semicortable(euro0306[,2:6],inscore=FALSE)
```

---

structcop *Simulation and maximum likelihood for structured factor copulas*

Description

Simulation and maximum likelihood for structured nested and bi-factor copulas

Usage

```
simnestfact(nn,grsize,cop,param)
simbifact(nn,grsize,cop,param)
f90str1nllk(param,dstruct,iprfn=F) # nested-factor
f90str2nllk(param,dstruct,iprfn=F) # bi-factor
```

Arguments

nn	sample size
grsize	vector of group sizes for mgrp groups with sum(grsize)=d
cop	number code for a copula model: 1 for Gaussian/normal, 2 for Student t, 3 for Gumbel, 5 for Frank; 10 for Gumbel/BB1 for simnestfact; 9 for BB1/Frank for simbifact

param	parameter vector; length is d+mgrp+1(cop==2)
dstruct	structure that includes $quad for the gauss-legendre nodes and weights, $copname for the model, $data for data set of dependent U(0,1), $grsize for grsize. For t-factor model, also $nu for degree of freedom parameter. Also $repar is a code for reparametrization (check examples).
iprfn	flag for printing of function value and derivatives

Details

f90str1nllk: "t", "tbb1" (t for group latent to global latent, BB1 for observed to group latent), "tgum", "frank", "gumbel", "frkgum", "frkbb1", "gumbb1", "tgum",

f90str2nllk: "frank", "gumbel", "gumfrk", "bb1frk", "bb1gum", "t" "tapprox" (latter uses monotone interpolation for the Student t cdf)

The order of BB1 parameters for all of the models with a BB1 component is theta1,theta2,...thetad,delta1,....deltad, with thetas>0 and deltas>1.

This is different for BB1 parameters for the 1-factor BB1 and 2-factor BB1/Frank, where it is theta1,delta1,theta2,delta2,...,thetad.deltad.

The difference is due to how to handle the f90 code.

Value

data matrix of dimension nn x d for simnestfact() and simbifact() with U(0,1) or N(0,1) or t(df) margins.

$fnval, $grad, $hess for f90str1nllk and f90str2nllk.

References

Krupskii P and Joe H (2013). Structured factor copula models: theory, inference and computation.

See Also

factorcopmle factorcopsim mvtfact

Examples

```
gl=gausslegendre(25)
grsize=c(4,4,3)
d=sum(grsize)
n=500
# nested-factor copula
mgrp=length(grsize)
set.seed(123)
parne=c(rep(4,3),rep(6,4),rep(6.5,4),rep(7,3))
udatne=simnestfact(n,grsize,cop=5,parne)
dstrfrk=list(data=udatne,copname="frank",quad=gl,repar=0,grsize=grsize)
npar=mgrp+d
outn= pdhessminb(rep(3,npar),f90str1nllk, ifixed=rep(FALSE,npar), dstrfrk,
  LB=rep(0,npar), UB=rep(30,npar), mxiter=30, eps=5.e-5,iprint=TRUE)
# bi-factor copula
set.seed(123)
parbi=c(rep(4,11),rep(6,4),rep(6.5,4),rep(7,3))
udatbi=simbifact(n,grsize,cop=5,parbi)
```

```
npar=2*d
dstrfrk=list(data=udatbi,copname="frank",quad=gl,repar=0,grsize=grsize,pdf=0)
nllk=f90str2nllk(parbi,dstrfrk)
outb=pdhessminb(c(rep(2,d),rep(3,d)),f90str2nllk,ifixed=rep(FALSE,npar),dstrfrk,
  LB=rep(0,npar), UB=rep(20,npar), mxiter=30, eps=5.e-5,iprint=TRUE)
```

`tailweightedDepmeas` *Tail-weighted dependence measures*

Description

Tail-weighted dependence measures: (a) bivariate copulas and (b) empirical data

Usage

```
twdm(pcop,param,power,nq,tscore=F)
twdm.emp(data,power)
twdm.emp.vec(data,power)  # vectorized, faster version
twdmnestcop(dcop,pcondcop,param1,param2,power,nq) # bivariate margin of
#                        nested-factor copula, 2 different groups
twdm1factcop(pcondcop,param,power,nq) # bivariate margin of 1-factor copula
twdm2factcop(pcondcop1,pcondcop2,param1,param2,power,nq) # bivariate margin of 2
```

Arguments

`pcop`	function for bivariate copula cdf
`param`	dependence parameter of pcop, or pcondcop in the two variables
`power`	power to use for tail-weighted dependence measure, good choice is 6 for twdm
`nq`	number of quadrature points for Gauss-Legendre quadrature
`tscore`	if T, Student t transform or normal transform of Gauss-Legendre quadrature points are used; this is faster is pcop is pbvncop or pbvtcop
`data`	data matrix with dimensions nxd
`dcop`	function for bivariate copula density for global with group1 and group2 latent
`pcondcop`	function for copula conditional cdf given latent
`pcondcop1`	function for copula conditional cdf for first latent
`pcondcop2`	function for copula conditional cdf for second latent
`param1`	dependence parameter of pcondcop1 in the two variables for twdm2factcop; dependence parameter of dcop for two group variables for twdmnestcop
`param2`	dependence parameter of pcondcop2 in the two variables; dependence parameter of pcondcop for two observed variables in twdmnestcop

Value

`twdm`	for twdm, twdmnestcop, twdm1factcop, twdm2factcop: vector of length 2 lower and upper tail-weighted dependence measure values
`ltwdm`	for twdm.emp and twdm.emp.vec: dxd matrix of empirical lower tail-weighted dependence measure values; for upper tail-weighted values, input with negation of the data set.

References

Krupskii P (2014). Structured Factor Copulas and Tail Inference. PhD thesis, University of British Columbia.

See Also

factorcopcdf factorcopsim pcond structcop

Examples

```
th1=gum.b2cpar(.7)
th2=gum.b2cpar(.6)
gum0.tw=twdm(pgum,th1,power=6,nq=15)
# 1-factor and 2-factor
gum1.tw=twdm(pfact1gum,c(th1,th1),power=6,nq=15)
gum2.tw=twdm(pfact2gum,matrix(c(th1,th1,th2,th2),2,2),power=6,nq=15)
gum1b=twdm1factcop(pcondgum,c(th1,th1),6,35)
gum2b=twdm2factcop(pcondgum,pcondgum,c(th1,th1),c(th2,th2),6,35)
# theoretical
cat(gum0.tw,"\n")
cat(gum1.tw,gum1b,"\n") # same from the two methods
cat(gum2.tw,gum2b,"\n") # same from the two methods
#
n=1000
set.seed(123)
gumdat1=sim1fact(n,c(th1,th1),qcondgum,"gumbel",ivect=TRUE)
set.seed(124)
gumdat2=sim2fact(n,c(th1,th1),c(th2,th2),qcondgum,qcondgum,"gumbel","gumbel",ivect=TRUE)
# empirical
gum1.ltw=twdm.emp(gumdat1,power=6)
gum1.utw=twdm.emp(1-gumdat1,power=6)
print(c(gum1.ltw[1,2],gum1.utw[1,2]))
gum2.ltw=twdm.emp.vec(gumdat2,power=6)
gum2.utw=twdm.emp.vec(1-gumdat2,power=6)
print(c(gum2.ltw[1,2],gum2.utw[1,2]))
# nested-factor
gumn=twdmnestcop(dgum,pcondgum,c(th1,th1),c(th2,th2),6,55)
cat(gumn,"\n")
n=1000
set.seed(123)
grsize=c(2,2)
gumdatn=simnestfact(n,grsize, cop=3, c(th1,th1,th2,th2,th2,th2))
gumn.ltw=twdm.emp.vec(gumdatn,power=6)
gumn.utw=twdm.emp.vec(1-gumdatn,power=6)
print(c(gumn.ltw[1,2],gumn.utw[1,2]))
```

---

taucor *Kendall's tau for two variables*

Description

Data version of Kendall's tau for two variables, using Knight's algorithm

Usage

```
taucor(x,y=0,pflag=0)
taub(otab) # Kendall's taub, adjusted for ties for 2-way table
```

Arguments

x	first variable, ties allowed; n x d data matrix if y=0
y	second variable, ties allowed
pflag	print flag, 0 = silent, 1 = print number of ties, exchange count and other summeries 2 = print sorted data 3 = print sorting iterations
otab	ordinal 2-way table/matrix

Value

tau	Kendall's tau with value in (-1,1); denominator adjusts for ties in case of ordinal variables

References

Knight WR (1966). A computer method for calculating Kendall's tau with ungrouped data. Journal of the American Statistical Association, 61, 436-439.

Examples

```
set.seed(1234)
x=rnorm(50)
y=.6*x+.8*rnorm(50)
tau=taucor(x,y)
tau2=cor(x,y,method="kendall")  # this is too slow when length(x) is large
cat(tau,tau2,"\n")
z=.5*x+.5*y+sqrt(.5)*rnorm(50)
tau=taucor(cbind(x,y,z))
tau2=cor(cbind(x,y,z),method="kendall")
print(max(abs(tau-tau2)))
# check for two ordinal variables
n=25
set.seed(123)
x1=floor(runif(n,0,3))
x2=floor(runif(n,0,3))
otab=table(x1,x2)
taub(otab) # should be same as next two lines
taucor(x1,x2)
cor(x1,x2,method="kendall")
```

---

uscore *Transform to U(0,1) scores*

Description

Transform each variable in a data matrix to uniform scores

Usage

```
uscore(data,aunif=-0.5)
```

Arguments

data	data matrix or data frame
aunif	rank i -> (i+aunif)/(n+1+2*aunif) for sample size n; default is -0.5

Value

data matrix with same number of columns as input

See Also

nscore

Examples

```
set.seed(123)
xx=matrix(rnorm(40),20,2)
uu=uscore(xx)
```

---

varray2M *Vine array to maximum matrix*

Description

Vine array to maximum matrix

Usage

```
varray2M(A,iprint=F,str="")
```

Arguments

A	dxd vine array with 1:d on diagonal; only upper triangle is used
iprint	print flag for some intermediate steps
str	string to describe the vine if iprint=T

Details

In the VineCopula R package and in "Dissmann J, Brechmann EC, Czado C and Kurowicka D (2013). Computational Statistics and Data Analysis, 59, 52-69", the vine array is abbreviation as RVM and it is A[d:1,d:1] where d=ncol(A). This conversion is needed if this package and VineCopula are used together.

Value

mxarray	array with M[k,j]= max A[k,1],..,A[k,j]
icomp	array with indicators for use in log-likelihood and simulation

See Also

vinearray

Examples

```
C= matrix(c(1,0,0,0,0, 1,2,0,0,0, 1,2,3,0,0, 1,2,3,4,0, 1,2,3,4,5), 5,5)
D= matrix(c(1,0,0,0,0, 1,2,0,0,0, 2,1,3,0,0, 3,2,1,4,0, 4,3,2,1,5), 5,5)
B0=matrix(c(1,0,0,0,0, 1,2,0,0,0, 1,2,3,0,0, 1,2,3,4,0, 1,3,2,4,5), 5,5)
B1=matrix(c(1,0,0,0,0, 1,2,0,0,0, 1,2,3,0,0, 1,2,3,4,0, 2,1,3,4,5), 5,5)
B2=matrix(c(1,0,0,0,0, 1,2,0,0,0, 2,1,3,0,0, 1,2,3,4,0, 1,2,3,4,5), 5,5)
B3=matrix(c(1,0,0,0,0, 1,2,0,0,0, 1,2,3,0,0, 1,3,2,4,0, 2,1,3,4,5), 5,5)
D6= matrix(c(1,0,0,0,0,0, 1,2,0,0,0,0, 2,1,3,0,0,0, 3,2,1,4,0,0, 4,3,2,1,5,0,
   5,4,3,2,1,6),6,6)
varray2M(C ,iprint=TRUE,"C ")
varray2M(D ,iprint=TRUE,"D ")
varray2M(B0,iprint=TRUE,"B0")
varray2M(B1,iprint=TRUE,"B1")
varray2M(B2,iprint=TRUE,"B2")
varray2M(B3,iprint=TRUE,"B3")
varray2M(D6,iprint=TRUE,"D6")
```

---

vinearray *Vine array conversion, validation and generation*

Description

Vine array conversion, validation and generation

Usage

```
varray2NO(A,irev=F,iprint=F) # vine array to natural order
vbin2array(d,b=0,iprint=F)   # binary representation to vine aray
varray2bin(A)  # vine array to binary representation
varraycheck(A) # vine array validation check
genVineArray(d)  # generate random vine array
varrayperm(A,perm) # vine array with permutation of indices
vnum2array(d,bnum=0,iprint=F) # decimal (of binary representation) to vine array
Dvinearray(d,iNO=F) # D-vine array
Cvinearray(d)       # C-vine array
```

Arguments

A	dxd vine array: only the upper triangle is used. For varray2bin, A should be in natural order, 1:d on diagonal. and also A[j-1,j]=j-1 (j=2,...,d). For varray2NO, varraycheck and varrayperm, A is not required to have 1:d on diagonal.
irev	irev=F means A1[d,d]=A[d,d], irev=T means A1[d,d]=A[d-1,d]
iprint	print flag for some intermediate steps
d	dimension of vine array
b	binary dxd matrix, the important entries are in columns 4 to d; rows 2 to j-2 in column j.
perm	d-dimension vector, permutation of 1:d

bnum	for vnum2array, integer in the range 0 and 2^((d-2)*(d-3)/2)-1, used to create the binary matrix b; vnum2array allows easier enumeration through all vine arrays in natural order for small d=5,6,7.
iNO	Dvinearray in natural order if iNO=T, and in standard 1-2-...-d order if iNO=F

Details

In the VineCopula R package and in "Dissmann J, Brechmann EC, Czado C and Kurowicka D (2013). Computational Statistics and Data Analysis, 59, 52-69", the vine array is abbreviation as RVM and it is A[d:1,d:1] where d=ncol(A). This conversion is needed if this package and VineCopula are used.

Value

Aobj	varray2NO returns a list with $NOa which is a vine array with A[i,i]=A[i,i+1] without necessarily a sorted diagonal, $NO with vine array in natural order with sorted diagonal 1:d, $perm is the permutation to get to from the NOa to NO, $diag is the diagonal of NOa.
ANO	if b is an array in vbin2array, then a vine array in NO=natural order corresponding to b is returned; if b=0 in vbin2array (or equivalently genVineArray, then a random vine array is returned.
bNO	if A is a vine array in NO, varray2bin the binary matrix, otherwise varray2bin returns -1 to indicate that the array is not in NO.
code	for varraycheck, code= 1 for valid vine array; code= -3 for diagonal not 1:d; code= -2 for not permutation of 1:j in column j; code= -1 if cannot find proper binary array from array in natural order.
A	for Dvinearray, Cvinearray, genVineArray and vnum2array, the output is a dxd vine array.
Aperm	for varrayperm, the output is a dxd vine array with permuted indices, perm[] is on the diagonal.

References

Joe H, Cooke RM and Kurowicka D (2011). Regular vines: generation algorithm and number of equivalence classes. In Dependence Modeling: Vine Copula Handbook, pp 219–231. World Scientific, Singapore.

See Also

gausstrvine rvinesim

Examples

```
C5= matrix(c(1,0,0,0,0, 1,2,0,0,0, 1,2,3,0,0, 1,2,3,4,0, 1,2,3,4,5), 5,5)
D5= matrix(c(1,0,0,0,0, 1,2,0,0,0, 2,1,3,0,0, 3,2,1,4,0, 4,3,2,1,5), 5,5)
D5perm= matrix(c(5,0,0,0,0, 5,4,0,0,0, 4,5,3,0,0, 3,4,5,2,0, 2,3,4,5,1), 5,5)
varrayperm(D5,perm=c(5,4,3,2,1))  # same as D5perm above
print(varray2NO(C5))
print(varray2NO(D5))
DNO=varray2NO(D5perm)
print(DNO)
set.seed(123)
vbin2array(d=5,b=0,iprint=FALSE)  # random vine array in NO=natural order
```

```
set.seed(123)
genVineArray(d=5)
b=matrix(0,5,5)
b[2:3,5]=1
vbin2array(d=5,b=b,iprint=F)  # vine array with binary matrix b
for(i in 0:7) { print(vnum2array(5,bnum=i,iprint=FALSE)) }
# more checks
for(i in 1:3)
{ d=floor(runif(1,5,10))
  A=vbin2array(d,0,iprint=TRUE)
  print(A)
  b=varray2bin(A)
  print(b)
}

# interface varraycheck to varray2NO, rename varray2bin
varraycheck(C5)
varraycheck(D5)
varraycheck(DNO$NO)
# proper vine array
A1=matrix(c(1,0,0,0,0,0, 1,2,0,0,0,0, 1,2,3,0,0,0, 1,2,3,4,0,0, 2,1,3,4,5,0,
   2,1,4,3,5,6),6,6)
b1=varraycheck(A1)
print(b1)
A1perm=varrayperm(A1,c(1,4,5,2,6,3))
varraycheck(A1perm) # also OK
# improper vine array, code=-1
A2=matrix(c(1,0,0,0,0,0, 1,2,0,0,0,0, 1,2,3,0,0,0, 1,2,3,4,0,0, 2,1,3,4,5,0,
   2,3,1,4,5,6),6,6)
b2=varraycheck(A2)
print(b2)
# improper vine array, code=-2
A3=matrix(c(1,0,0,0,0,0, 1,2,0,0,0,0, 1,2,3,0,0,0, 1,2,3,4,0,0, 2,1,3,4,5,0,
   2,1,4,3,3,6),6,6)
b3=varraycheck(A3)

# Dvinearray
Dvinearray(5)
Dvinearray(5,iNO=TRUE)
vnum2array(5,bnum=0) # same as above
# Cvinearray
Cvinearray(5)
vnum2array(5,bnum=7) # same as above
```

`vinemargincdf` *Bivariate marginal copulas for trees 2 and 3 of vine copulas*

Description

Bivariate marginal copulas for trees 2 and 3 of vine copulas

Usage

```
ptree2cop(ub,uc,param,pcondba,pcondca,pcopbc,nq)
```

```
dtree2cop(ub,uc,param,pcondba,pcondca,dcopbc,dcopab,dcopac,nq)
pcondtree2(uc,ub,param,pcondba,pcondca,pcondcb,dcopab,nq)
ptree3cop.cvine(uc,ud,param,pcondba,pcondca,pcondda,pcondcb,pconddb,
  pcopcd,dcopab,nq)
pcondtree3.cvine(ud,uc,param,pcondba,pcondca,pcondda,pcondcb,pconddb,
  pconddc,dcopab,dcopac,dcopbc,nq)
dtree3cop.cvine(uc,ud,param,pcondba,pcondca,pcondda,pcondcb,pconddb,
  dcopcd,dcopab,dcopac,dcopad,dcopbc,dcopbd,nq)
ptree3cop.dvine(ua,ud,param,pcondab,pcondbc,pcondcb,pconddc,pcondac,
  pconddb,pcopad,dcopbc,nq)
pcondtree3.dvine(ud,ua,param,pcondab,pcondbc,pcondcb,pconddc,pcondac,
  pconddb,pcondda,dcopab,dcopbc,dcopac,nq)
dtree3cop.dvine(ua,ud,param,pcondab,pcondbc,pcondcb,pconddc,pcondac,
  pconddb,dcopad,dcopab,dcopbc,dcopcd,dcopac,dcopbd,nq)
```

Arguments

ub	vector for variable b, each element in interval 0,1
param	vector of length 3 for tree2, and length 6 for tree3
pcopbc	function for copula cdf for variables b,c; similar for pcopcd, pcopad
pcondba	function for conditional cdf of variable b given variable a; similar for pcondca, pcondcb, pcondda, pconddb, pconddc, pcondab, pcondac, pcondbc
dcopbc	function for copula pdf of variables b,c; similar for dcopab, dcopac, dcopad, dcopbd, dcopcd
nq	number of quadrature points for Gauss-Legendre quadrature
ua	vector for variable a, each element in interval 0,1
uc	vector for variable c, each element in interval 0,1
ud	vector for variable d, each element in interval 0,1
pcopcd	similar
pcopad	similar
pcondca	similar
pcondcb	similar
pcondda	similar
pconddb	similar
pconddc	similar
pcondab	similar
pcondac	similar
pcondbc	similar
dcopab	similar
dcopac	similar
dcopad	similar
dcopbd	similar
dcopcd	similar

Value

cdf or conditional cdf or pdf value(s) of bivariate margin of vine copula

See Also

rvinenllk pcond

Examples

```
ptree2gumfrk=function(u,v,param)
{ ptree2cop(u,v,param,pcondgum,pcondgum,pfrk,nq=35) }
pcondtree2gumfrk21=function(v,u,param)
{ pcondtree2(v,u,param,pcondgum,pcondgum,pcondfrk,dgum,nq=35) }
pcondtree2gumfrk12=function(v,u,param)
{ pcondtree2(v,u,param[c(2,1,3)],pcondgum,pcondgum,pcondfrk,dgum,nq=35) }
d=4
parmat=matrix(c(0,1.4,1.4,1.6, 0,0,2.6,3.4, 0,0,0,2, 0,0,0,0),4,4,byrow=TRUE)
A=Cvinearray(d)
cat("theoretical rhoS and tau for tree 2\n")
for(j in 3:d)
{ j0=A[1,j]
  j1=A[2,j]; j2=A[j,j]
  jmin=min(j0,j1); jmax=max(j0,j1)
  param3=c(parmat[1,jmax],parmat[1,j],parmat[2,j])
  rho=rhoS(param3,ptree2gumfrk)
  tau=ktau(param3,pcond12=pcondtree2gumfrk12,pcond21=pcondtree2gumfrk21)
  cat(j,rho,tau,"\n")
}
# similar for the one pair in tree 3
```

---

wcb8594 *Workmans' Compensation Board monthly claims 1985-1994*

Description

Data set from "Freeland RK (1998), Statistical Analysis of Discrete Time Series with Application to the Analysis of Workers' Compensation Claims Data, PhD thesis, University of British Columbia."

variable 1: nclaims is the monthly claims of short-term disability benefits made by injured workers from 1985 to 1994; category is cut injury in the logging industry.

variable 2: x1 has sin(2*pi*t/12), t=1,...,120.

variable 3: x2 has cos(2*pi*t/12), t=1,...,120.

x1 and x2 are considered as vectors of seasonal covariates.

Usage

```
data(wcb8594)
```

Index